Soft Servo
SYSTEMS, INC

ServoWorksPLC Manual

Version 1.1

Table of Contents

Table of Contents ..... **i**
List of Figures ..... **v**
List of Tables ..... **viii**
Chapter 1: What is a PLC Sequence Program? **1-1**
1.1 Overview of PLC in the ServoWorks System 1-1
1.2 The ServoWorksPLC Application Suite 1-3
1.3 PLC Sequence Programs 1-4
Chapter 2: Installing and Initializing the ServoWorksPLC Application Suite **2-1**
2.1 Overview 2-1
2.2 Starting the Installation 2-1
2.3 Installing and Initializing the ServoWorksPLC Application Suite 2-2
2.4 Finishing Your Installation 2-5
2.5 Uninstalling the ServoWorksPLC Application Suite 2-6
Chapter 3: Using the ServoWorksPLC Utility Programs **3-1**
3.1 Overview 3-1
3.2 Using the PLC Control Console Application to Compile Sequence Programs 3-2
3.2.1 Starting the PLC Control Console Application 3-2
3.2.2 Compiling Your Sequence Program 3-4
3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration) 3-5
3.2.4 Setting Up Your Keep Relay for PLC 3-6
3.2.5 Setting Up Your Timer for PLC 3-7
3.2.6 Setting Up Your Counters for PLC 3-8
3.3 Verifying Sequence Programs Using the ServoWorksPLC Ladder Monitor/Debugger 3-9
3.3.1 Overview of the Ladder Monitor/Debugger 3-9
3.3.2 Using the Ladder Monitor/Debugger 3-9
3.3.3 Ladder Diagram Format (Interpreting the Ladder Diagram) 3-10
3.3.3.1 Overview 3-10
3.3.3.2 Addresses 3-11
3.3.3.3 Signal Names 3-12
3.3.3.4 Commenting 3-12
3.3.3.5 Symbols 3-12
3.3.3.6 Rows 3-13
3.3.3.7 Relay Junction Labeling 3-14
3.3.3.8 Infinite Number of Relays 3-14
3.3.4 Changing the Display of the Ladder Monitor/Debugger 3-15
3.3.5 Using the Search Function of the PLC Ladder Monitor/Debugger 3-15
3.4 Using the PLC Bit Pattern Utility 3-16
3.5 Using the PLC Time Chart Utility 3-17
Chapter 4: Setting Up A Sequence Program **4-1**
4.1 Description of the Sequence Program and Other PLC Files 4-1
4.1.1 Overview 4-1
4.1.2 .lad Files 4-1
4.1.3 .mod Files 4-4

4.1.4 .bin Files, .div Files, .fig Files and .lst Files 4-4
4.2 Overview of File Structure 4-5
4.3 Coding Convention 4-6
Chapter 5: Inside the PLC Engine (How the PLC Engine Operates) 5-1
5.1 The Sequential Processing of the Sequence Program 5-1
5.2 Repetitive Sampling 5-2
5.3 I/O Signals 5-3
5.3.1 Input 5-3
5.3.2 Output 5-3
5.4 PLC Code Execution 5-4
Chapter 6: Memory Addresses 6-1
6.1 What Are Memory Addresses? 6-1
6.2 Addresses Related to the PLC 6-1
6.3 Address Specifications 6-2
6.4 PLC and Machine Tool Addresses (PLC ↔ MT) 6-3
6.5 PLC Engine and ServoWorks Motion Engine Addresses (PLC ↔ NC) 6-3
6.6 Internal Relay Addresses (R) 6-4
6.7 Counter Addresses (C) 6-5
6.8 Keep Relay and Static Memory Control Addresses (K) 6-6
6.9 Data Table Addresses (D) 6-6
6.10 Timer Addresses (T) 6-7
6.11 Alarm Relay Addresses (A) 6-8
Chapter 7: Static Memory 7-1
7.1 Timer, Counter, Keep Relay, Static Memory Control, Data Table 7-1
7.1.1 Overview of Static Memory 7-1
7.1.2 Timer 7-1
7.1.3 Counter (Addresses C0~C79) 7-1
7.1.4 Keep Relay (Addresses K0~K99) 7-2
7.1.5 Data Table (Addresses D0~D1999) 7-2
7.2 Reading and Writing Static Memory 7-3
7.3 PLC Data Table 7-3
7.3.1 Overview 7-3
7.3.2 Creation of Data in the Data Table 7-4
Chapter 8: PLC Basic Commands 8-1
8.1 Overview 8-1
8.1.1 Signal Addresses 8-2
8.1.2 Types of Commands (Basic and Functional) 8-2
8.1.3 Storing the Results of Logic Operations in the Result History Register 8-2
8.2 Basic Commands 8-3
8.2.1 Summary of Basic Commands 8-3
8.2.2 RD Command 8-4
8.2.3 RD.NOT Command 8-6
8.2.4 WRT Command 8-8
8.2.5 WRT.NOT Command 8-9
8.2.6 AND Command 8-10
8.2.7 AND.NOT Command 8-10

8.2.8 OR Command 8-11
8.2.9 OR.NOT Command 8-11
8.2.10 RD.STK Command 8-11
8.2.11 RD.NOT.STK Command 8-13
8.2.12 AND.STK Command 8-15
8.2.13 OR.STK Command 8-15
Chapter 9: PLC Functional Commands 9-1
9.1 Overview 9-1
9.1.1 Functional Command Format 9-3
9.1.2 Control Values 9-4
9.1.3 Command 9-5
9.1.4 Parameters 9-5
9.1.5 W1 9-5
9.1.6 Operation Data – Binary Coded Decimal or Binary Format 9-5
9.1.7 Numerical Data Examples 9-6
9.1.7.1 BCD Format Data 9-6
9.1.7.2 Binary Format Data 9-6
9.1.8 Addresses for the Numerical Data Handled by Functional Commands 9-8
9.1.9 Functional Command Register (R9000 ~ R9005) 9-9
9.2 Descriptions of Functional Commands 9-10
9.2.1 TMR (Timer) 9-10
9.2.2 TMRB (Fixed Timer) 9-12
9.2.3 TMRC (Timer) 9-14
9.2.4 DEC (Decoding) 9-16
9.2.5 DECB (Binary Decoding Processing) 9-18
9.2.6 CTR (COUNTER) 9-20
9.2.7 CTRC (Counter) 9-26
9.2.8 ROT (Rotational Control) 9-29
9.2.9 ROTB (Binary Rotational Control) 9-33
9.2.10 COD (Code Transformation) 9-37
9.2.11 CODB (Binary Code Conversion) 9-41
9.2.12 MOVE (Masked Data Transfer) 9-43
9.2.13 MOVOR (Bit-Wise Sum Data Transfer) 9-46
9.2.14 COM (Common Line Control) 9-47
9.2.15 COME (Common Line Control Termination) 9-50
9.2.16 JMP (Jump) 9-51
9.2.17 JMPE (Jump Termination) 9-54
9.2.18 PARI (Parity Check) 9-55
9.2.19 DCNV (Data Conversion) 9-58
9.2.20 DCNVB (Extended Data Conversion) 9-60
9.2.21 COMP (Compare) 9-62
9.2.22 COMPB (Binary Compare) 9-64
9.2.23 COIN (Equality Check) 9-66
9.2.24 SFT (Shift Register) 9-68
9.2.25 DSCH (Data Search) 9-71
9.2.26 DSCHB (Binary Data Search) 9-74

9.2.27 XMOV (Index Modification Data Transfer)....9-76
9.2.28 XMOVB (Binary Index Modification Data Transfer)....9-80
9.2.29 ADD (Addition)....9-82
9.2.30 ADDB (Binary Addition)....9-84
9.2.31 SUB (Subtraction)....9-86
9.2.32 SUBB (Binary Subtraction)....9-89
9.2.33 MUL (Multiplication)....9-91
9.2.34 MULB (Binary Multiplication)....9-94
9.2.35 DIV (Division)....9-96
9.2.36 DIVB (Binary Division)....9-99
9.2.37 NUME (Constant Declaration)....9-101
9.2.38 NUMEB (Binary Constant Declaration)....9-103
Appendix A: S-100T Data Mapping Tables....A-1
Overview of Mapping Tables....A-1
F Data Mapping Table....A-2
G Data Mapping Table....A-12
X Data Mapping Tables....A-14
HandWheel I/P (FP-60)....A-14
Home & Limit Switches (DC-120)....A-14
Appendix B: S-100M and General Motion Applications (MC-Quad, MotionPro and SWSDK) Data Mapping Tables....B-1
Overview of Mapping Tables....B-1
F Data Mapping Table....B-2
G Data Mapping Table....B-5
X Data Mapping Table....B-8
Y Data Mapping Table....B-24
Index....I

List of Figures

Figure 1-1: Overview of the PLC Engine in the ServoWorks System 1-2
Figure 2-1: Welcome to ServoWorks MC-Quad Window 2-2
Figure 2-2: ServoWorksPLC Installation Window 2-3
Figure 2-3: ServoWorksPLC Installation Summary Window 2-3
Figure 2-4: PLC Control Screen 2-4
Figure 2-5: PLC Ladder Compiler Screen (1 of 2) 2-4
Figure 2-6: Select Module Definition Window 2-5
Figure 2-7: Compile Finish Dialog Box 2-5
Figure 2-8: ServoWorksPLC Installation Window 2-6
Figure 3-1: Architecture of the ServoWorksPLC Application Suite 3-2
Figure 3-2: PLC Control Screen Window for “Run” Status 3-3
Figure 3-3: PLC Control Screen Window for “Stopped” Status 3-3
Figure 3-4: PLC Ladder Compiler Screen 3-4
Figure 3-5: Compile Finish Dialog Box 3-5
Figure 3-6: PLC Table Setting Screen 3-5
Figure 3-7: PLC Table Setting Screen Dialog Box 3-6
Figure 3-8: Edit Keep Relay Window 3-6
Figure 3-9: Edit Timer Window 3-7
Figure 3-10: Edit Counter Window 3-8
Figure 3-11: PLC Diagnose Window 3-9
Figure 3-12: Module Selection Window 3-10
Figure 3-13: PLC Diagnose Window With Comments for an Example Module 3-10
Figure 3-14: Format for an Address in a Ladder Diagram 3-11
Figure 3-15: Ladder Diagram Rows 3-14
Figure 3-16: Format for Relay Junction Labeling in Ladder Diagrams 3-14
Figure 3-17: Ladder Diagram Format for a Limited Number of Relays 3-14
Figure 3-18: Ladder Diagram Format for an Infinite Number of Relays 3-15
Figure 3-19: PLC Diagnose Window Without Comments for an Example Module 3-15
Figure 3-20: PLC Diagnose Search Window 3-16
Figure 3-21: Bit Pattern Window 3-16
Figure 3-22: Time Chart Window 3-17
Figure 4-1: Typical .lad File: SoftServo_0 (1 of 2) 4-2
Figure 4-2: Typical .lad File: SoftServo_0.lad (2 of 2) 4-3
Figure 4-3: Example .mod File: SoftServo_0.mod 4-4
Figure 4-4: Sequence Program Setup Procedure (1 of 2) 4-5
Figure 4-5: Sequence Program Setup Procedure (2 of 2) 4-6
Figure 4-6: PLC Coding Example – Instruction List Format 4-7
Figure 4-7: PLC Coding Example – Ladder Diagram Format 4-8
Figure 5-1: First Example of a Circuit 5-1
Figure 5-2: Second Example of a Circuit 5-2
Figure 5-3: Execution of Sequence Program by the PLC Engine 5-5
Figure 6-1: Addresses Related to the PLC Engine 6-1
Figure 6-2: Internal Relay Usable Region 6-4
Figure 6-3: Counter Addresses 6-5

Figure 6-4: Keep Relay and Static Memory Control Addresses 6-6
Figure 6-5: Data Table Addresses 6-6
Figure 6-6: Timer Addresses 6-7
Figure 6-7: Alarm Relay Addresses 6-8
Figure 7-1: Example of Counter Addresses 7-2
Figure 8-1: Signal Addresses 8-2
Figure 8-2: Structure of the Result History Register 8-2
Figure 8-3: Ladder Diagram Example for the RD Command 8-4
Figure 8-4: Ladder Diagram Example for the RD.NOT Command 8-6
Figure 8-5: Ladder Diagram Example for the WRT Command 8-8
Figure 8-6: Ladder Diagram Example for the WRT.NOT Command 8-9
Figure 8-7: Ladder Diagram Example for the RD.STK Command 8-12
Figure 8-8: Ladder Diagram Example for the RD.NOT.STK Command 8-13
Figure 9-1: Functional Command Format – Ladder Diagram and Functional Command Register 9-3
Figure 9-2: Example of BCD Format Data 9-6
Figure 9-3: Memory Storage of Binary Format Data 9-7
Figure 9-4: Examples of Binary Format Data for 1 Byte Data 9-8
Figure 9-5: Addresses of Numeric Data 9-9
Figure 9-6: Functional Command Register 9-9
Figure 9-7: Format for the TMR Command 9-10
Figure 9-8: Timer Behavior for the TMR Command 9-11
Figure 9-9: Format for the TMRB Command 9-12
Figure 9-10: Timer Behavior for the TMRB Command 9-12
Figure 9-11: Format for the TMRC Command 9-14
Figure 9-12: TMRC Address of the Time Set of the Timer 9-15
Figure 9-13: Timer Register Address for the TMRC Command 9-15
Figure 9-14: Timer Behavior for the TMRC Command 9-15
Figure 9-15: Format for the DEC Command 9-16
Figure 9-16: Ladder Diagram Example Using the DEC Command 9-17
Figure 9-17: Function for the DECB Command 9-18
Figure 9-18: Format for the DECB Command 9-18
Figure 9-19: Ring Counter Created Using the CTR Command 9-20
Figure 9-20: Format for the CTR Command 9-21
Figure 9-21: Count Signal (Action Command) for the CTR Command 9-22
Figure 9-22: Ladder Diagram For Counter Example #1 9-23
Figure 9-23: Ladder Diagram For Counter Example #2 9-24
Figure 9-24: Division of a Rotational Body for Counter Example #2 9-24
Figure 9-25: Format for the CTRC Command 9-26
Figure 9-26: Count Signal (Action Command) for the CTRC Command 9-28
Figure 9-27: Address of the Counter Preset Value for the CTRC Command 9-28
Figure 9-28: Address of the Up Counter Output for the CTRC Command 9-28
Figure 9-29: Format for the ROT Command 9-29
Figure 9-30: Rotation Direction Rule – 12-Division Example 9-32
Figure 9-31: Format for the ROTB Command 9-33
Figure 9-32: Ladder Diagram Example Using the ROTB Command 9-36

Figure 9-33: Code Transformation Using the COD Command 9-37
Figure 9-34: Format for the COD Command 9-38
Figure 9-35: Code Transformation Using the CODB Command 9-41
Figure 9-36: Format for the CODB Command 9-41
Figure 9-37: Input Data and Logic Data for the MOVE Command 9-43
Figure 9-38: Format for the MOVE Command 9-44
Figure 9-39: Ladder Diagram Example Using the MOVE Command 9-45
Figure 9-40: Function of the MOVOR Command 9-46
Figure 9-41: Format for the MOVOR Command 9-46
Figure 9-42: Function of the COM Command 9-47
Figure 9-43: Format for the COM Command 9-47
Figure 9-44: Relay Circuit 9-48
Figure 9-45: Ladder Diagram Using the COM Command 9-49
Figure 9-46: Ladder Diagram Example Using COM, MOVE and COIN Commands .. 9-49
Figure 9-47: Format for the COME Command 9-50
Figure 9-48: Function of the JMP Command 9-51
Figure 9-49: Format for the JMP Command 9-51
Figure 9-50: Ladder Diagram Example Using the JMP Command 9-53
Figure 9-51: Format for the JMPE Command 9-54
Figure 9-52: Format for the PARI Command 9-55
Figure 9-53: Ladder Diagram Example Using the PARI Command 9-57
Figure 9-54: Format for the DCNV Command 9-58
Figure 9-55: Format for the DCNVB Command 9-60
Figure 9-56: Calculation Result Register for the DCNVB Command 9-61
Figure 9-57: Format for the COMP Command 9-62
Figure 9-58: Format for the COMPB Command 9-64
Figure 9-59: Parameters Format Specification for the COMPB Command 9-64
Figure 9-60: Calculation Result Register for the COMPB Command 9-65
Figure 9-61: Format for the COIN Command 9-66
Figure 9-62: Format for the SFT Command 9-68
Figure 9-63: Condition Specification CONT = 0 for the SFT Command – Shift Left Example 9-69
Figure 9-64: Condition Specification CONT = 1 for the SFT Command – Shift Left Example 9-69
Figure 9-65: Shift Data Address for the SFT Command 9-70
Figure 9-66: Function of the DSCH Command 9-71
Figure 9-67: Format for the DSCH Command 9-72
Figure 9-68: Function of the DSCHB Command 9-74
Figure 9-69: Format for the DSCHB Command 9-74
Figure 9-70: Reading from and Writing to the Data Table for the XMOV Command . 9-76
Figure 9-71: Format for the XMOV Command 9-77
Figure 9-72: Reading from and Writing to the Data Table for the XMOVB Command... 9-80
Figure 9-73: Format for the XMOVB Command 9-80
Figure 9-74: Format for the ADD Command 9-82
Figure 9-75: Format for the ADDB Command 9-84

Figure 9-76: Parameters Format Specification for the ADDB Command 9-84
Figure 9-77: Calculation Result Register for the ADDB Command 9-85
Figure 9-78: Format for the SUB Command 9-86
Figure 9-79: Format for the SUBB Command 9-89
Figure 9-80: Parameters Format Specification for the SUBB Command 9-89
Figure 9-81: Calculation Result Register for the SUBB Command 9-90
Figure 9-82: Format for the MUL Command 9-91
Figure 9-83: Format for the MULB Command 9-94
Figure 9-84: Parameters Format Specification for the MULB Command 9-95
Figure 9-85: Calculation Result Register for the MULB Command 9-95
Figure 9-86: Format for the DIV Command 9-96
Figure 9-87: Format for the DIVB Command 9-99
Figure 9-88: Parameters Format Specification for the DIVB Command 9-99
Figure 9-89: Calculation Result Register for the DIVB Command 9-100
Figure 9-90: Format for the NUME Command 9-101
Figure 9-91: Format for the NUMEB Command 9-103

List of Tables

Table 3-1: Signal Letters and Their Meanings 3-11
Table 3-2: Ladder Diagram Symbols (1 of 2) 3-12
Table 3-3: Ladder Diagram Symbols (2 of 2) 3-13
Table 4-1: PLC Coding Example – Program Breakdown 4-7
Table 6-1: Symbols for the Signal Types 6-2
Table 8-1: PLC Basic Commands and Their Functions 8-3
Table 8-2: Coding of the RD Command Example (Alternative #1) 8-5
Table 8-3: Coding of the RD Command Example (Alternative #2) 8-5
Table 8-4: Coding of the RD.NOT Command Example (Alternative #1) 8-7
Table 8-5: Coding of the RD.NOT Command Example (Alternative #2) 8-7
Table 8-6: Coding of the WRT Command 8-8
Table 8-7: Coding of the WRT.NOT Command 8-9
Table 8-8: Coding of the RD.STK Command 8-12
Table 8-9: Coding of the RD.NOT.STK Command 8-14
Table 9-1: Summary of Functional Commands (1 of 2) 9-1
Table 9-2: Summary of Functional Commands (2 of 2) 9-2
Table 9-3: Functional Command Format – Coding 9-4
Table 9-4: Coding Format of the TMR Command 9-10
Table 9-5: Coding Format of the TMRC Command 9-14
Table 9-6: Coding Format of the DEC Command 9-16
Table 9-7: Coding Example of the DEC Command 9-17
Table 9-8: Coding Format of the CTR Command 9-21
Table 9-9: Coding Format of the CTRC Command 9-27
Table 9-10: Coding Format of the ROT Command 9-30
Table 9-11: Coding Format of the COD Command 9-39

Table 9-12: Coding Format of the MOVE Command 9-44
Table 9-13: Coding Format of the JMP Command 9-52
Table 9-14: Coding Format of the PARI Command 9-56
Table 9-15: Coding Format of the DCNV Command 9-58
Table 9-16: Coding Format of the COMP Command 9-62
Table 9-17: Coding Format of the COIN Command 9-66
Table 9-18: Coding Format of the DSCH Command 9-72
Table 9-19: Coding Format of the XMOV Command 9-78
Table 9-20: Coding Format of the ADD Command 9-82
Table 9-21: Coding Format of the SUB Command 9-87
Table 9-22: Coding Format of the MUL Command 9-92
Table 9-23: Coding Format of the DIV Command 9-97
Table 9-24: Coding Format of the NUME Command 9-101
Table B-1: SwPLC F Address Map for NC Mode Settings B-30
Table B-2: SwPLC F/G Address Map for HandWheel Multiple Selection B-30
Table B-3: SwPLC F/G Address Map for HandWheel and HandWheel Interrupt Axis Selection B-30
Table B-4: SwPLC G Address Map for Manual Feedrate Override B-31
Table B-5: SwPLC G Address Map for Feedrate Override B-32
Table B-6: SwPLC G Address Map for Rapid Override B-32
Table B-7: SwPLC G Address Map for NC Mode Settings B-33
Table B-8: SwPLC G Address Map for Jog Axis Control B-33

Chapter 1: What is a PLC Sequence Program?

1.1 Overview of PLC in the ServoWorks System

Programmable logic control is the process of automating the monitoring and sequence control of machines (machine tools, in this case). In the ServoWorks system, this process is performed by the PLC Engine, which is included as part of ServoWorksPLC application suite. In this text, the term "PLC" refers to the PLC programming language.

The ServoWorks Motion Engine controls the high-performance, multi-axis servo loops: the motion of the machine tool. Based on its feedback loops for motion control, it may want to send a command to the machine tool to slow down, speed up, etc. But let's imagine that there is a door open on the machine tool. The ServoWorks Motion Engine doesn't have access to the signal that would tell it about the open door. This is where the PLC Engine comes in.

The PLC Engine has access to *all* the information about not only the ServoWorks system (hardware and software), but also the machine tool inputs and outputs. The PLC Engine is the one element that has access to everything there is to know about the ServoWorks system and the machine tool. As such, it functions as the "central headquarters" for all decision-making regarding the motion and the machine tool. It controls the sequencing of everything that happens with the machine tool, the servomotors, etc.

The relationships between the ServoWorks application software, the ServoWorks Motion Engine, the PLC Engine and the machine tool are as follows:

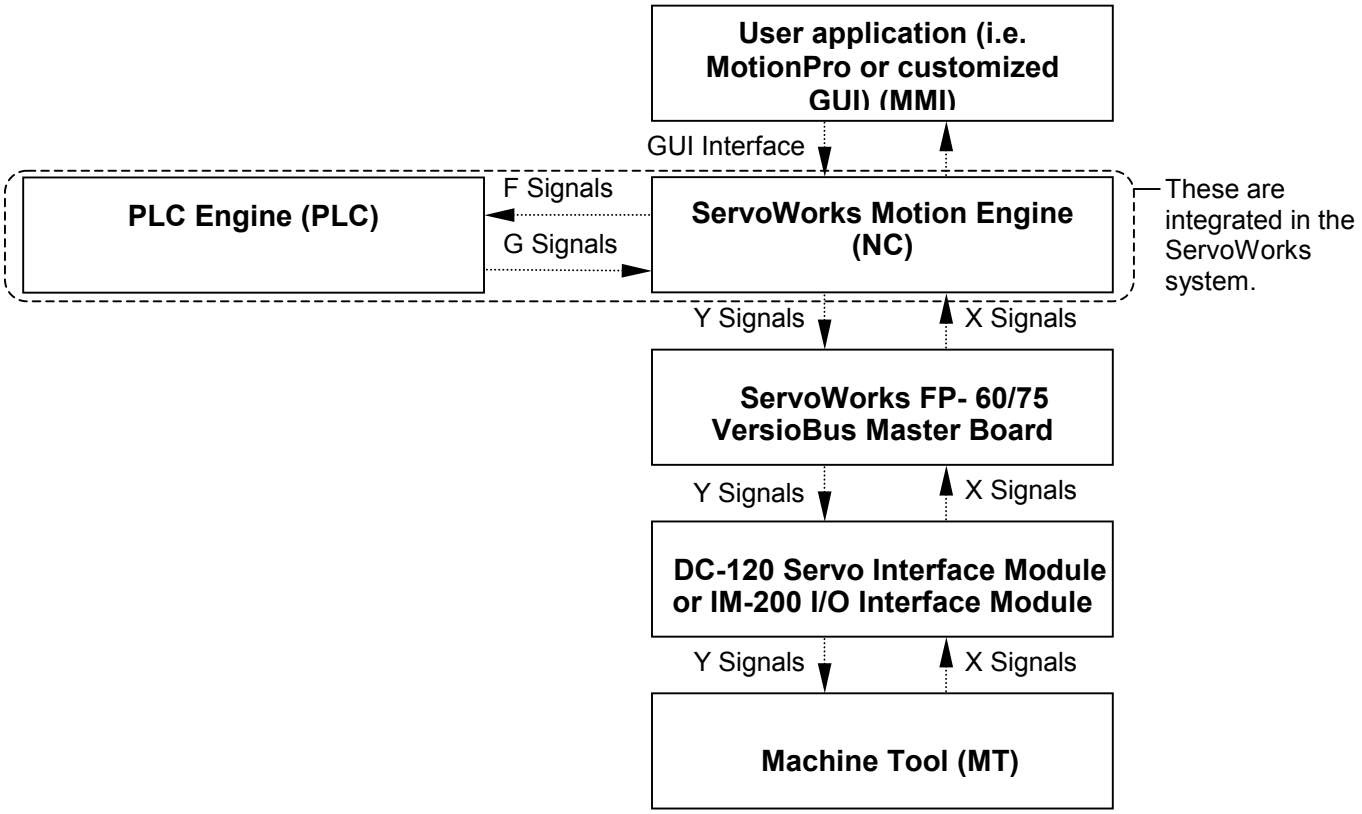


Figure 1-1: Overview of the PLC Engine in the ServoWorks System

The PLC Engine executes the sequence program in a cyclical fashion. The PLC Engine has a timer, and uses it to run the following PLC cycle every scan time (by default, the scan time is 8 msec):

1) The PLC Engine performs a full scan of inputs from both the ServoWorks Motion Engine and inputs from the machine tool (which go through the ServoWorks Motion Engine). In other words, it checks the status of each bit ("0" or "1") for each F and each X input signal (which will be discussed later).

2) The PLC Engine runs the executable sequence program based on these new input values. It executes the program by reading and executing each command sequentially, at high speed. The command may specify reading or writing inputs or outputs, or performing logical operations such as AND or OR (arithmetic processing).

3) The PLC Engine sends any outputs or commands generated by the sequence program to the ServoWorks Motion Engine or to the machine tool (using the G and Y signals) via the ServoWorks Motion Engine.

This cycle of scan, execute sequence program, and generate outputs or commands is repeated every 8 ms (the standard scan time for the PLC Engine, which can also be user-defined). Every time the sequence program finishes executing, it starts again almost immediately. This means that the PLC Engine is very responsive to any inputs or commands.

For example, if a lathe operator gives a command (using the S-100T program) to start the spindle on a lathe, that command gets relayed through the ServoWorks Motion Engine to the PLC Engine. The command is picked up by the PLC Engine (in the form of an F signal, which will be discussed later) the next time it scans all the inputs from the ServoWorks Motion Engine and the machine tool. The PLC Engine then executes the sequence program, which will check that the door is closed on the machine tool, the collet is in position, etc. (It checks these by looking at the status of X signals from the machine tool.) If all the conditions for starting the spindle are met, the PLC Engine will issue a command to the ServoWorks Motion Engine, which is sent to the machine tool (in the form of a Y signal) to start the spindle on the lathe.

1.2 The ServoWorksPLC Application Suite

The ServoWorksPLC application suite has three parts:

- The ServoWorksPLC Control Console Application (Win32).
- The ServoWorksPLC Monitor/Debugger.
- ServoWorksPLC Utility Tools.

The ServoWorks PLC Engine is a real-time soft PLC module that executes PLC sequence programs, and is included in the base ServoWorks packages. This programmable logic controller (PLC) for machine tools reads and executes the binary PLC file every 8 ms (or some other user-defined scan time), decides if it needs to take any action based on these inputs or changes in these inputs, and issues commands to the ServoWorks Motion Engine or the machine tool, if necessary. The PLC Engine is seamlessly integrated with the ServoWorks Motion Engine that performs the motion control into a single motion / machine control application.

The ServoWorksPLC Control Console application is a stand-alone application that you can use to edit and compile your sequence programs in PLC Instruction List (IL) format into executable binary files, which can then be understood and executed by the PLC Engine.

The ServoWorksPLC Monitor/Debugger is for verifying sequence programs with ladder diagrams. This visual environment makes it easy for the PLC programmer to debug sequence programs.

The ServoWorksPLC Utility Tools includes utility tools such as bit pattern display, and time charts showing the history of bit signals.

1.3 PLC Sequence Programs

A sequence program is a program that is written in PLC code (Instruction List format) . This sequence program tells the PLC Engine how to control the machine tool and how to control the computer numerical control (ServoWorks Motion Engine).

The PLC language used by the ServoWorksPLC and described in this manual is compatible with Fanuc's PLC ladder logic.

Because each machine tool is different, you will need to write a unique PLC sequence program for each machine tool setup, although sequence programs can be reused for machine tool that are the exact same make and model.

Chapter 2: Installing and Initializing the ServoWorksPLC Application Suite

2.1 Overview

The ServoWorksPLC Application Suite may be provided with the MC-Quad CD-ROM, the S-100T CD-ROM, the S-100M CD-ROM or the MotionPro CD-ROM.

The software installation procedure shown here for the ServoWorksPLC Application Suite uses screen shots from the MC-Quad CD-ROM, but the same procedure applies to the S-100T CD-ROM, the S-100M CD-ROM or the MotionPro CD-ROM.

2.2 Starting the Installation

The following steps will guide you through installing the ServoWorksPLC Application Suite:

1) Close all running applications.

2) Place the CD-ROM labeled "MC-Quad" (or "S-100T," "S-100M" or "MotionPro") in the CD-ROM drive. The installation software may start automatically – if it doesn't start automatically, go to the Windows directory and click on the "Install.exe" file. (This file may be called "MC-Quad_Install.exe,", S-100T_Install.exe, etc.) You will see the "Welcome to ServoWorks" window appear, as shown in Figure 2-1:

Figure 2-1: Welcome to ServoWorks MC-Quad Window

Again, your "Welcome to ServoWorks" window will look different if you purchased the ServoWorksPLC Application Suite as part of the S-100T, as part of the S-100M or as part of MotionPro.

3) Click the "Read Me" button to view the brief installation instructions. We recommend that you print these instructions, so that you can refer to them for the ServoWorksPLC Application Suite initialization that is the last step of your installation procedure.

2.3 Installing and Initializing the ServoWorksPLC Application Suite

1) Click the "ServoWorksPLC" button. You will see the following window appear:

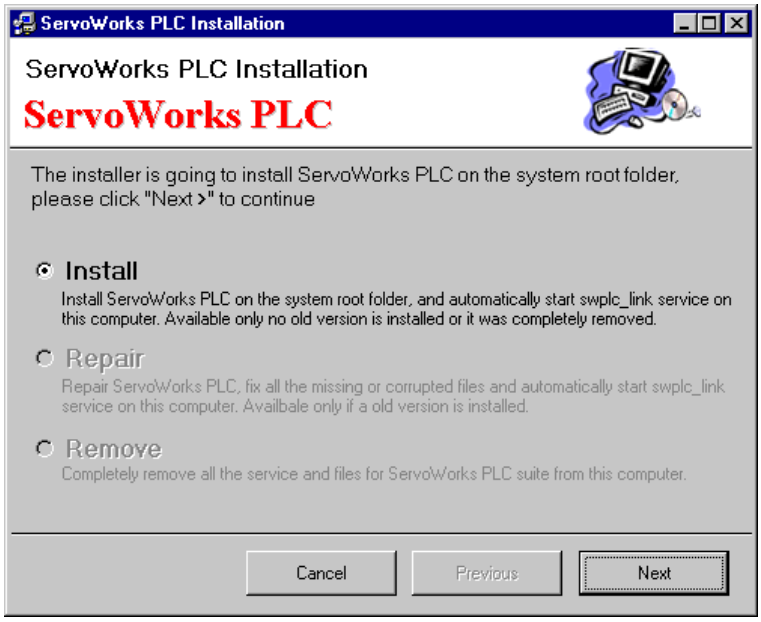


Figure 2-2: ServoWorksPLC Installation Window

2) Make sure the "Install" option is selected, and click the "Next" button to install the ServoWorksPLC application suite. The installation software will copy the ServoWorksPLC application suite files, and then you will see the window shown in Figure 2-3 appear.

NOTE: If the "Install" option is not available, you should choose the "Remove" option to remove any existing copies of the ServoWorksPLC Application Suite from your computer, then restart this installation process from step #1.

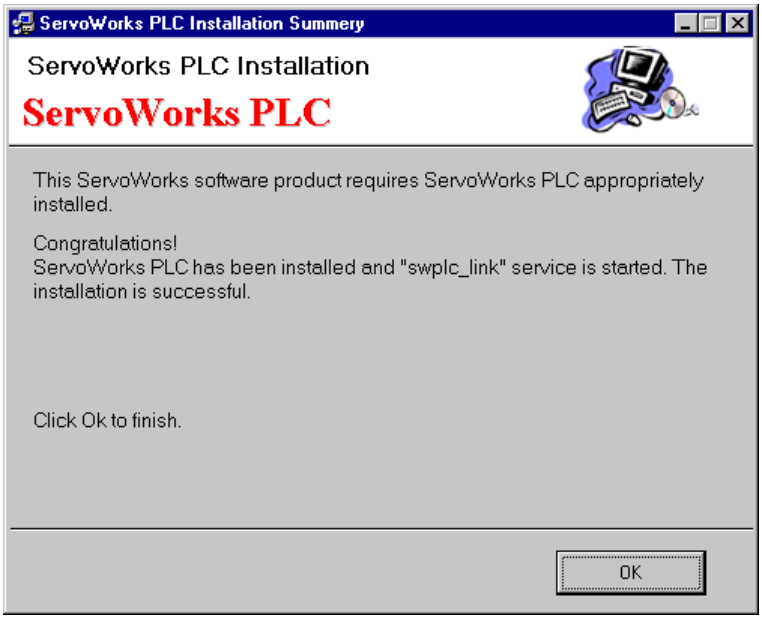


Figure 2-3: ServoWorksPLC Installation Summary Window

3) Click the "OK" button. You will see the following window appear:

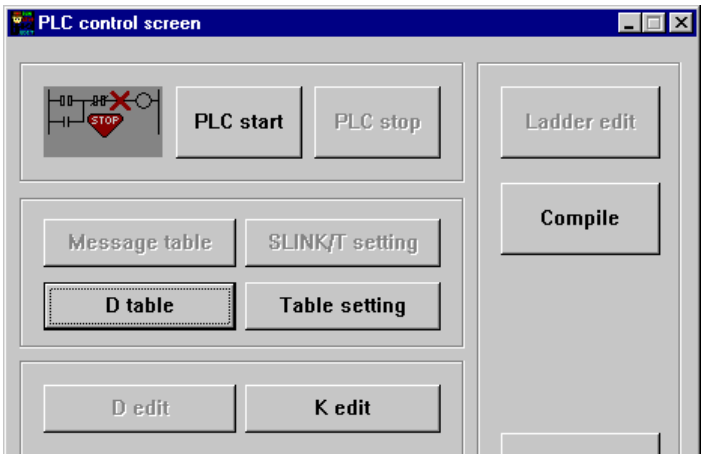


Figure 2-4: PLC Control Screen

4) Click the "Compile" button. You will see the PLC Ladder Compiler Screen appear.

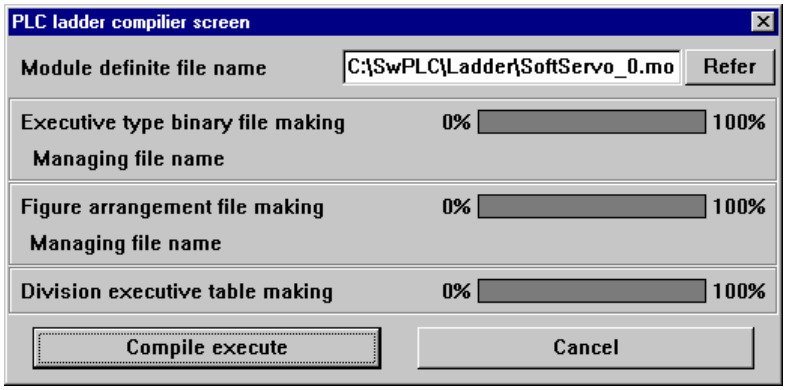


Figure 2-5: PLC Ladder Compiler Screen (1 of 2)

5) Click the "Refer" button. You will see the "Select Module Definition File" window appear.

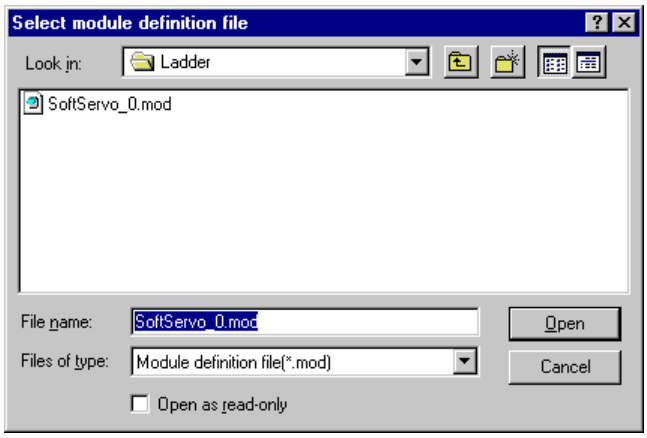


Figure 2-6: Select Module Definition Window

6) Browse to the SoftServo_0.mod file in the "C:\SwPLC\Ladder" folder (C:\SwPLC\Ladder\\SoftServo_0.mod) and click on this folder. Click the "Open" button once the correct file is selected. You will be back to the PLC Ladder Compiler Screen.

7) Click the "Compile execute" button. You will see the "Compile Finish" window appear.

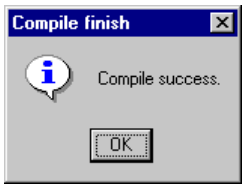


Figure 2-7: Compile Finish Dialog Box

8) Click the "OK" button and your initialization of the ServoWorksPLC Application Suite will be complete.

9) To exit ServoWorksPLC, click the "Cancel" button in the PLC Ladder Compiler Screen, and then click the exit button (✕) in the upper right hand corner of the PLC Control Screen.

Your ServoWorksPLC Application Suite is now properly initialized.

2.4 Finishing Your Installation

To finish your installation (after you have also installed either MC-Quad, the S-100T, the S-100M or MotionPro), just click the "Exit" button in the "Welcome to ServoWorks" window.

2.5 Uninstalling the ServoWorksPLC Application Suite

To uninstall the ServoWorksPLC Application Suite:

1) Click the "ServoWorksPLC" button, and you will see the following "ServoWorksPLC Installation" dialog box appear:

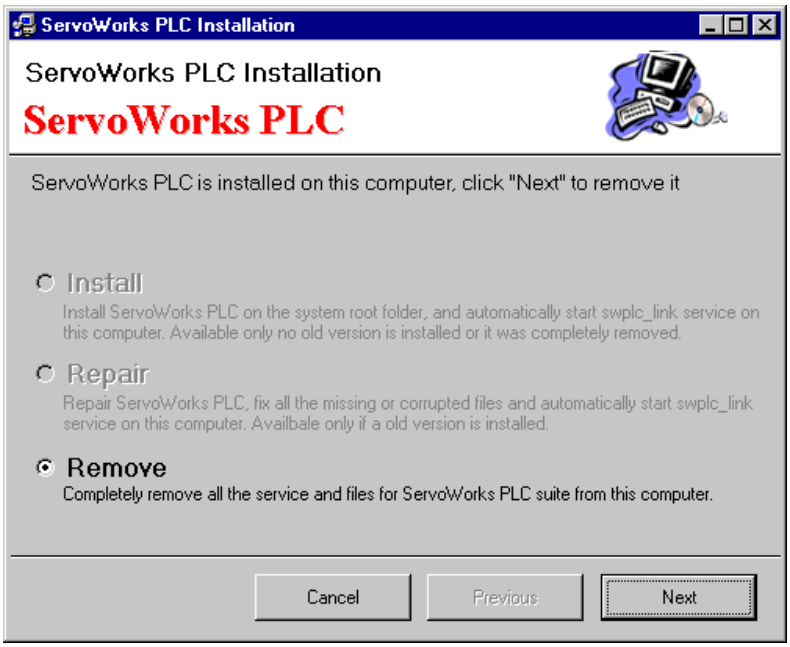


Figure 2-8: ServoWorksPLC Installation Window

2) Select "Remove" and click the "OK" button.

3) In some cases, you may need to manually delete the "SwPLC" folder that was automatically created by the original ServoWorksPLC Application Suite installation (typically "C:\SwPLC"). Manually deleting this folder may require you to reboot your computer first.

Chapter 3: Using the ServoWorksPLC Utility Programs

3.1 Overview

The ServoWorksPLC Application Suite consists of four independent utility programs:

- The ServoWorksPLC Control Console Application (Win32).
- The ServoWorksPLC Monitor/Debugger.
- PLC Bit Pattern Display Utility.
- PLC Time Chart Utility for showing the history of bit signals.

These four PLC utility programs interact with the ServoWorks applications and the ServoWorks Real-Time Modules as shown in the following figure:

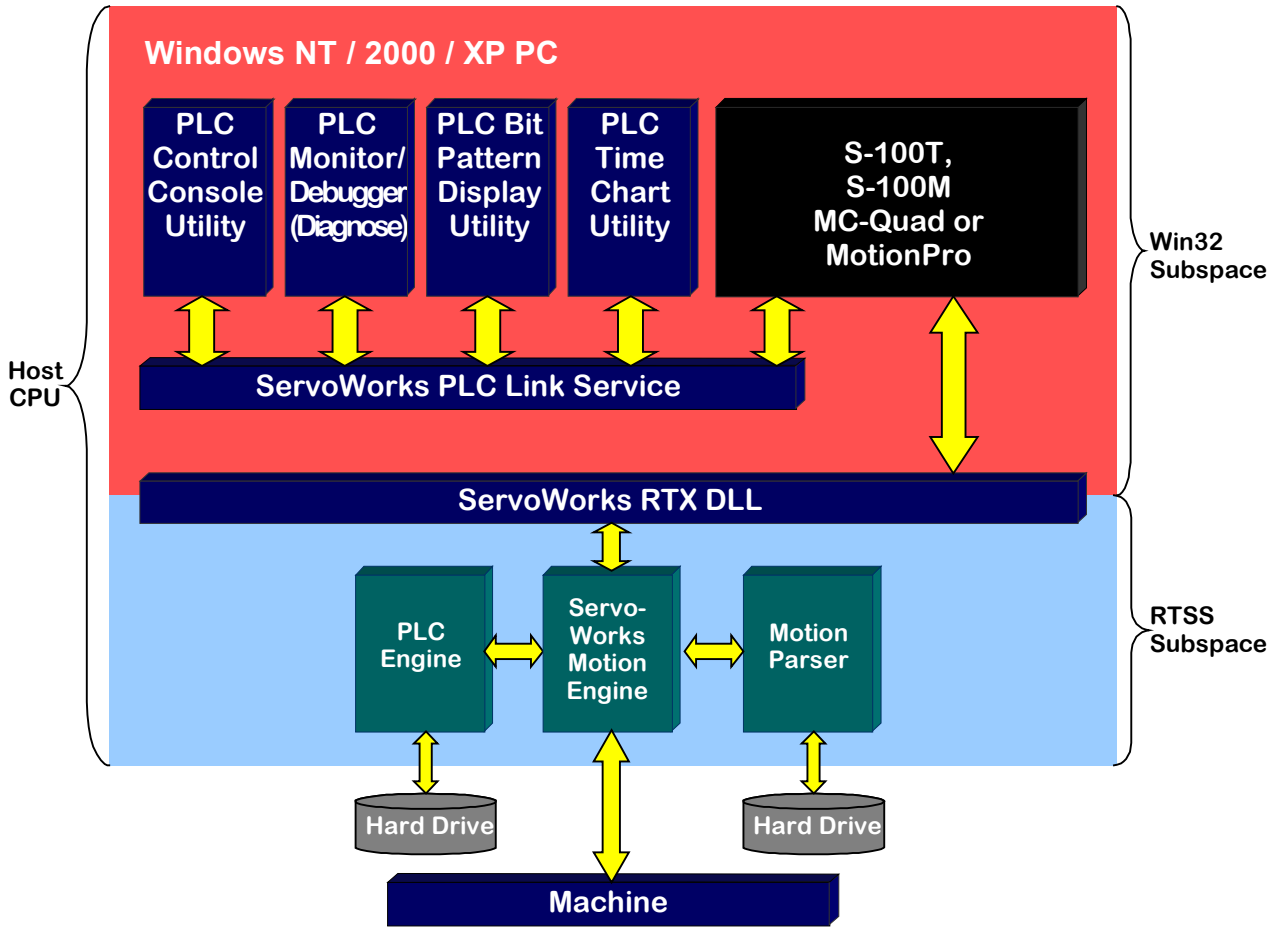


Figure 3-1: Architecture of the ServoWorksPLC Application Suite

3.2 Using the PLC Control Console Application to Compile Sequence Programs

3.2.1 Starting the PLC Control Console Application

This application enables you to start, stop and compile ladder logic, and set up the timer, the counter, and table setup data. These are all necessary operations for setting up PLC for your machine.

Start the PLC Control Console application by double clicking on the

SwPLC_CONTROL.exe

file in C:\SwPLC.

If the status of ladder logic is "RUN," you will see the PLC Control Screen window shown in the following figure:

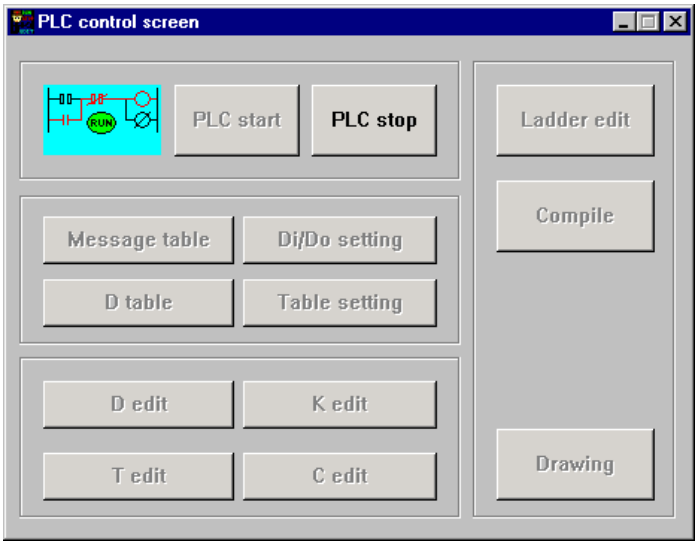


Figure 3-2: PLC Control Screen Window for "Run" Status

If the status of ladder logic is "STOPPED," you will see the PLC Control Screen window shown in the following figure:

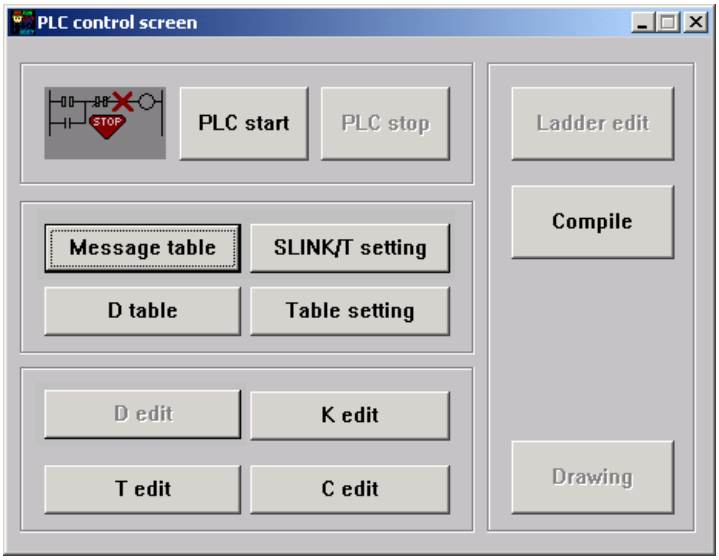


Figure 3-3: PLC Control Screen Window for "Stopped" Status

The first time you run the PLC Control Console application, the "K edit," "T edit" and "C edit" buttons will be disabled. These three buttons will be enabled only after the SwPLC table has been set up – see *Section 3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration)*.

To exit the PLC Control Console application at any time, click on the ✕ button in the upper right hand corner.

3.2.2 Compiling Your Sequence Program

Before you can execute the ladder logic in PLC, you will have to compile the PLC sequence program files. The following steps will guide you through compiling your sequence program.

1) The PLC Control Console application must be started, and the PLC status must be "Stopped." If the status is "Run," you must click on the "PLC stop" button, and the ladder logic process will stop immediately.

⚠ CAUTION

The Emergency Stop for your machine MUST be activated before you stop the PLC ladder logic process by clicking on the "PLC stop" button.

2) Click on the "Compile" button, and you will see the following window:

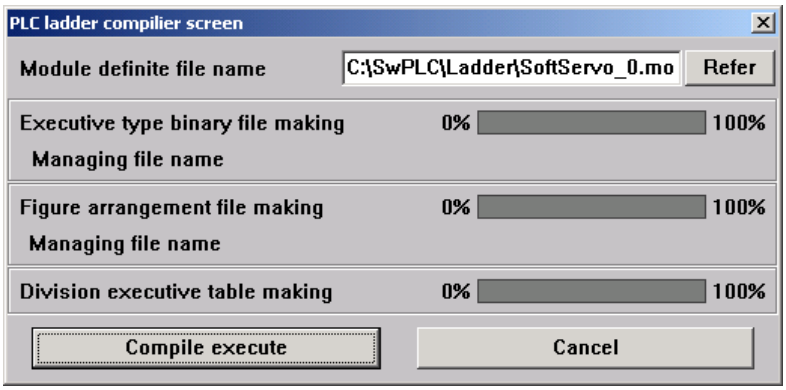


Figure 3-4: PLC Ladder Compiler Screen

3) In the above window, make sure the `.mod` file you want to compile is listed in the "Module definite file name" text box. If you to compile a different file, click on the "Refer" button and browse to the file you do want to compile.

4) Click on the "Compile execute" button to start compiling process. When it is finished successfully, you will see the following dialog box:

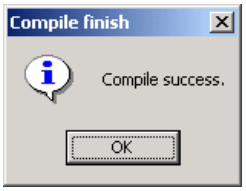


Figure 3-5: Compile Finish Dialog Box

3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration)

One of the steps necessary to convert a sequence program into machine language is I/O declaration. After you define the specifications (commands) for the program, you need to create the interface settings. In the interface table, you assign each I/O signal a name (under six characters), depending on the signal type. This procedure also enables the "K edit," "T edit" and "C edit" buttons in the PLC Control Console application. If you don't create the interface settings, you won't be able to edit the keep relays, the timers and the counters.

The following steps will guide you through setting up your tables for PLC:

1) Click the "Table setting" button on the PLC Control Screen, and the following window will appear:

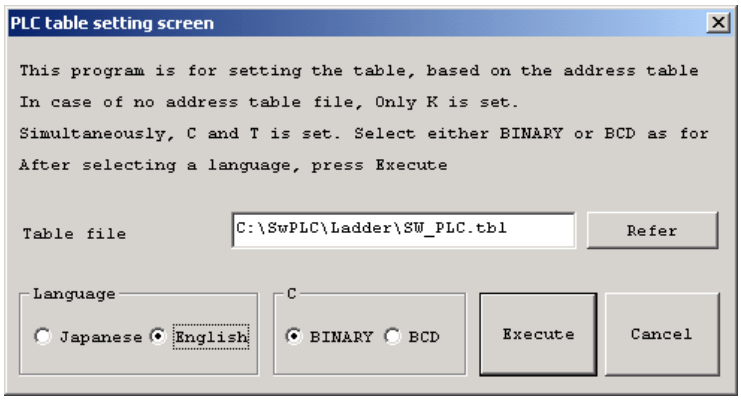


Figure 3-6: PLC Table Setting Screen

2) Make sure the comment setup table shown in the "Table file" text box is the file you want. If you to use a different file for your table, click on the "Refer" button and browse to the file you do want to use as the comment setup table file for the PLC Diagnose Screen of the PLC Monitor/Debugger.

3) Select either "BINARY" or "BCD" in the "C" frame, for the comment display.

4) Click on the "Execute" button. The setting process will start automatically. When the process has finished successfully, you will see the following dialog box:

Figure 3-7: PLC Table Setting Screen Dialog Box

Your tables are now successfully set up.

3.2.4 Setting Up Your Keep Relay for PLC

To set up your keep relay for PLC, click the "K edit" button on the PLC Control Screen, and the following window will appear:

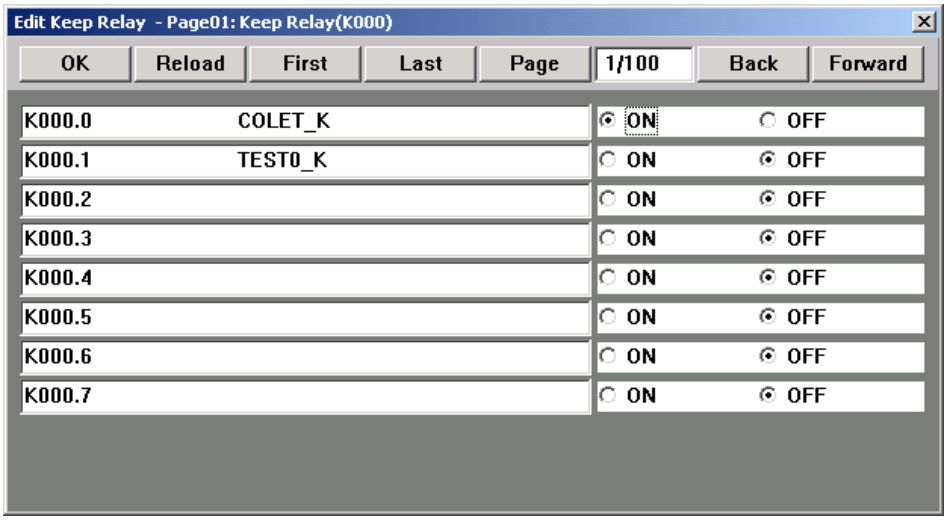


Figure 3-8: Edit Keep Relay Window

If the "K edit" button is disabled, you need to set up the SwPLC table – see *Section 3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration)*.

For each keep relay address, you can select "**ON**" or "**OFF**" for the status of the keep relay.

To set all the keep relays back to "**OFF**," click on the "Reload" button.

You can use the "Back" and "Forward" buttons to scroll through the keep relays. Clicking on the "First" button displays the first page of keep relays, while clicking on the

"Last" button displays the last page of keep relays. If you click on the "Page" button, a Page Selection Window will appear, and you can scroll through all pages of keep relays, and select one by clicking on the "OK" button.

When you have finished setting up your keep relays, click on the "OK" button to save the settings.

3.2.5 Setting Up Your Timer for PLC

To set up your timer for PLC, click the "T edit" button on the PLC Control Screen, and the following window will appear:

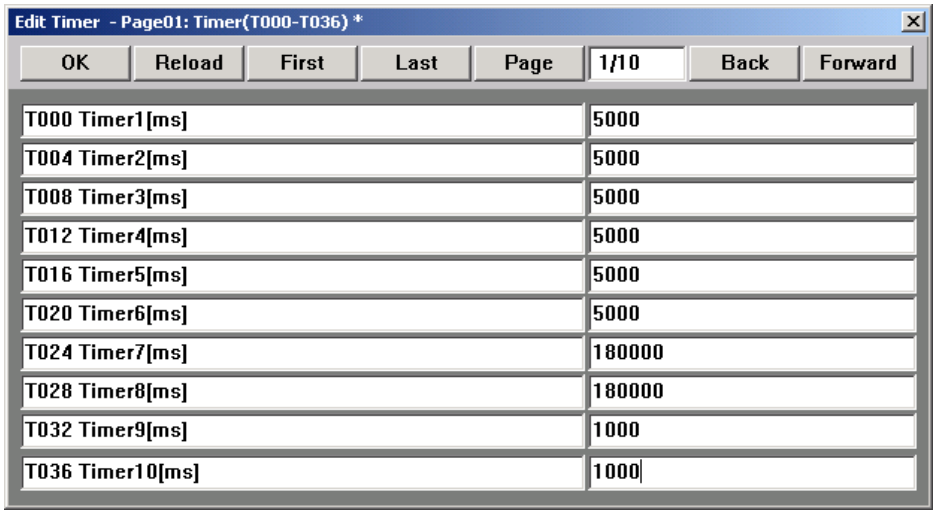


Figure 3-9: Edit Timer Window

If the "T edit" button is disabled, you need to set up the SwPLC table – see *Section 3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration)*.

For each timer, you can specify the time in milliseconds.

To set all the timers back to "**0**," click on the "Reload" button.

You can use the "Back" and "Forward" buttons to scroll through the timers. Clicking on the "First" button displays the first page of timers, while clicking on the "Last" button displays the last page of timers. If you click on the "Page" button, a Page Selection Window will appear, and you can scroll through all pages of timers, and select one by clicking on the "OK" button.

When you have finished setting up your timers, click on the "OK" button to save the settings.

3.2.6 Setting Up Your Counters for PLC

To set up your counters for PLC, click the "C edit" button on the PLC Control Screen, and the following window will appear:

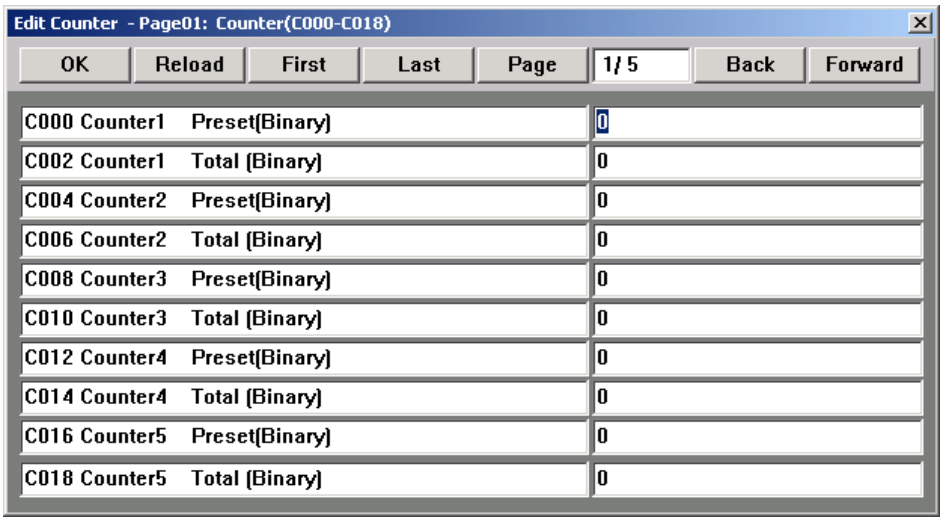


Figure 3-10: Edit Counter Window

If the "C edit" button is disabled, you need to set up the SwPLC table – see *Section 3.2.3 Setting Up Your Tables for PLC (Input/Output Declaration)*.

For each counter, you can specify the number of counts.

To set all the counters back to "**0**," click on the "Reload" button.

You can use the "Back" and "Forward" buttons to scroll through the counters. Clicking on the "First" button displays the first page of counters, while clicking on the "Last" button displays the last page of counters. If you click on the "Page" button, a Page Selection Window will appear, and you can scroll through all pages of counters, and select one by clicking on the "OK" button.

When you have finished setting up your counters, click on the "OK" button to save the settings.

3.3 Verifying Sequence Programs Using the ServoWorksPLC Ladder Monitor/Debugger

3.3.1 Overview of the Ladder Monitor/Debugger

This utility shows the status of an executing Ladder Logic as a Ladder circuit in real time. By monitoring the real-time status of contacts, you can verify that you have properly set up your PLC sequence program for your machine.

3.3.2 Using the Ladder Monitor/Debugger

The following steps will guide you in using the ServoWorksPLC Ladder Monitor/Debugger:

1) Start the PLC Ladder Monitor/Debugger utility by double clicking on the

file in C:\SwPLC. You will see the PLC Diagnose window shown in the following figure:

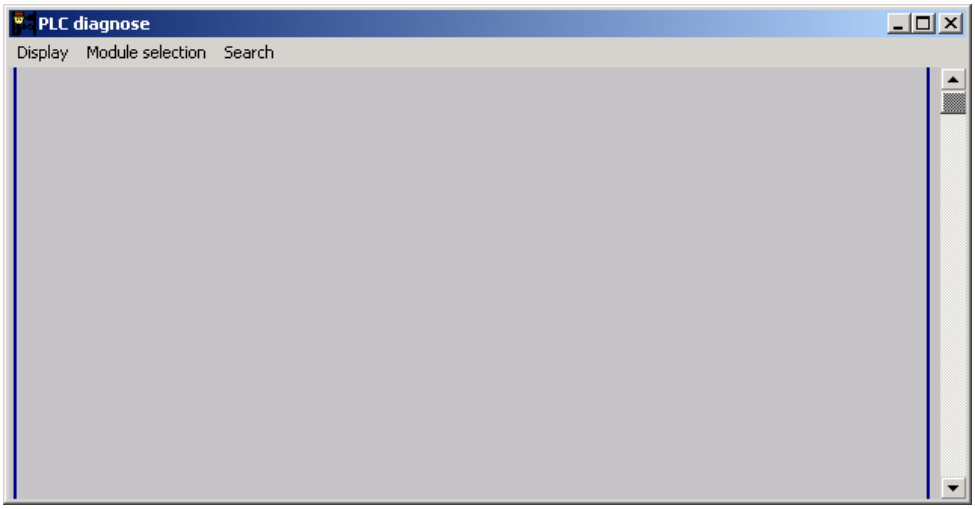


Figure 3-11: PLC Diagnose Window

2) Click on "Module selection" from the menu, and you will see the window shown in the following figure:

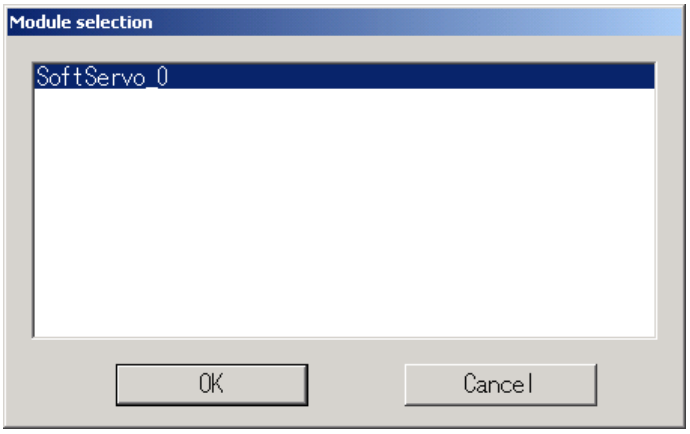


Figure 3-12: Module Selection Window

3) Select your module, and click the "OK" button. A Ladder window will appear for that module, similar to that shown in the following figure:

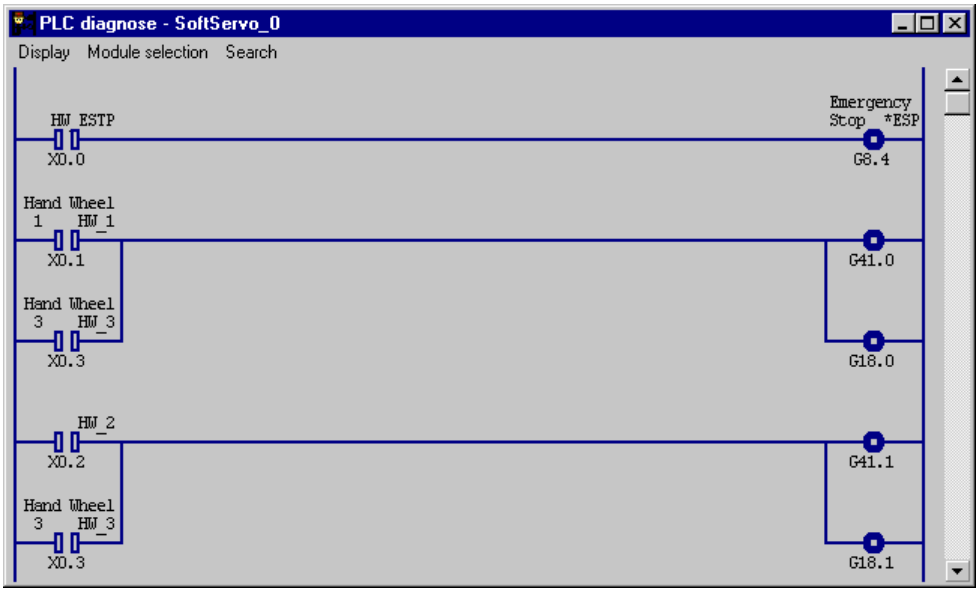


Figure 3-13: PLC Diagnose Window With Comments for an Example Module

3.3.3 Ladder Diagram Format (Interpreting the Ladder Diagram)

3.3.3.1 Overview

The ladder diagram is interpreted by you. There are a set of consistent symbols described below. The ladder diagram will contain addresses, signal names, and comments.

3.3.3.2 Addresses

An address is composed of an address number and a bit number, as described below.

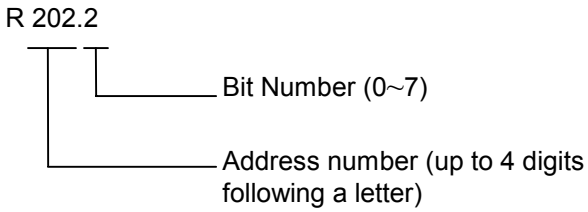


Figure 3-14: Format for an Address in a Ladder Diagram

There is always a letter of the alphabet in front of the address number to indicate the kind of signal it is. See Table 3-1 for letters used and their meanings.

Signal Letter	Signal Description
X	Input signal from the machine to the PLC Engine (MT → PLC)
Y	Output signal from the PLC Engine to the machine (PLC → MT)
F	Input signal from the ServoWorks Motion Engine (NC) to the PLC Engine (NC → PLC)
G	Output signal from the PLC Engine to the ServoWorks Motion Engine (NC) (PLC → NC)
R	Internal Relay
A	Requesting Message Display
C	Counter
K	Keep Relay
D	Data Table
T	Variable Timer

Table 3-1: Signal Letters and Their Meanings

3.3.3.3 Signal Names

All signal names will be at most 8 characters. Each character can be a letter, a number, or a special symbol.

3.3.3.4 Commenting

Up to 30 characters may be displayed to comment each relay coil and signal in the sequence program.

3.3.3.5 Symbols

The ladder diagram utilizes the following symbols:

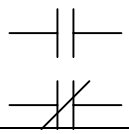
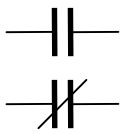
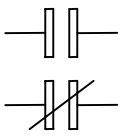
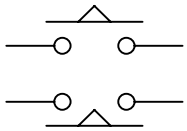
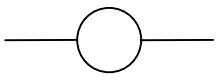
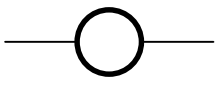
Symbol	Description
	Relay (for an internal variable) inside of the PLC Engine.
	Input signal from the ServoWorks Motion Engine.
	Input signal from the machine tool.
	Timer inside the PLC Engine.
	Relay coil used only inside the PLC Engine (for an internal variable).
	Relay coil that is output to the ServoWorks Motion Engine.

Table 3-2: Ladder Diagram Symbols (1 of 2)

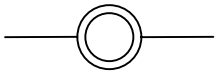
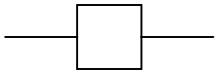
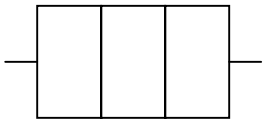
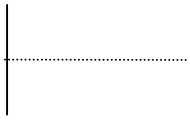
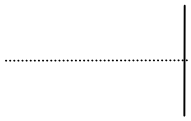
Symbol	Description
	Relay coil that is output to the machine tool.
	Timer coil inside the PLC Engine.
	Functional command of the PLC language. Symbols differ slightly depending on the kind of functional command.
	Left vertical power rail.
	Right vertical power rail.
	Horizontal line connection.
	Vertical line connection.

Table 3-3: Ladder Diagram Symbols (2 of 2)

3.3.3.6 Rows

The ladder diagram has rows along the longer side.

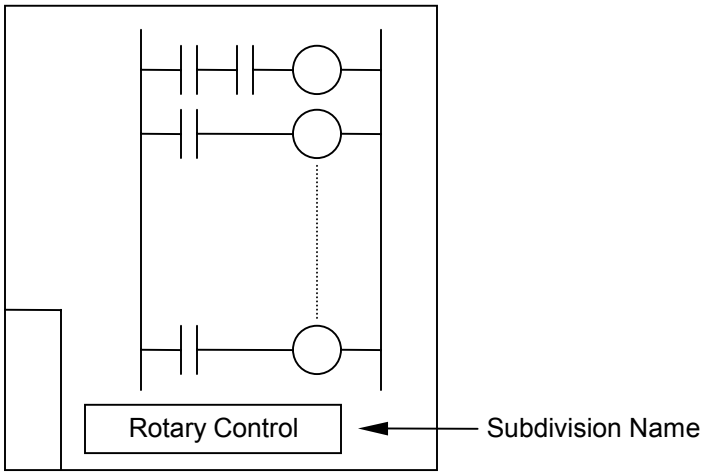


Figure 3-15: Ladder Diagram Rows

3.3.3.7 Relay Junction Labeling

The relay junction will contain the name of the relay coil and either the line number or the address of the signal.

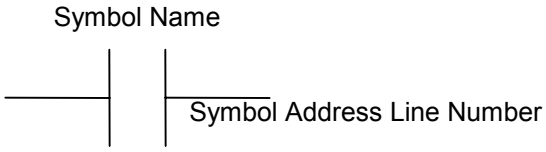


Figure 3-16: Format for Relay Junction Labeling in Ladder Diagrams

3.3.3.8 Infinite Number of Relays

In a usual relay sequence circuit, the number of relay junctions is limited, so in order to minimize the number of relays, relays are reused within the circuit.

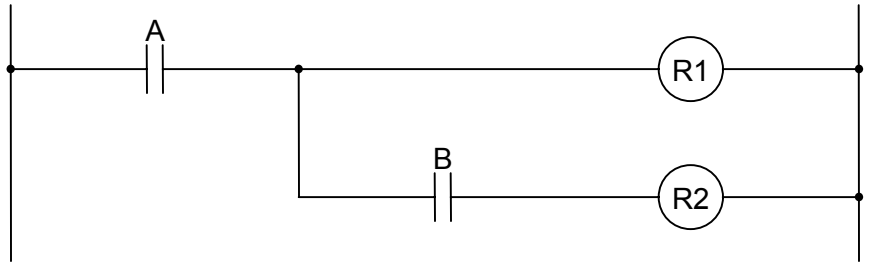


Figure 3-17: Ladder Diagram Format for a Limited Number of Relays

In the PLC Engine, there are an infinite number of relays, so the ladder diagram is written as shown below.

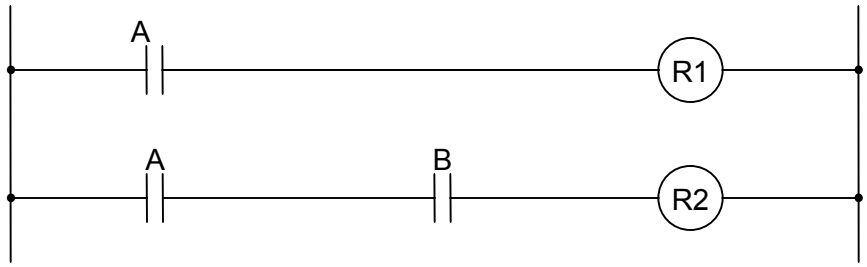


Figure 3-18: Ladder Diagram Format for an Infinite Number of Relays

3.3.4 Changing the Display of the Ladder Monitor/Debugger

You can click on the ☐ symbol in the upper right hand corner of the window to maximize the window, or use the scroll bar to scroll through the complete display.

You can also click on "Display" from the menu and uncheck the "comment display", to hide the comments to make the display more condensed, as shown in the following window:

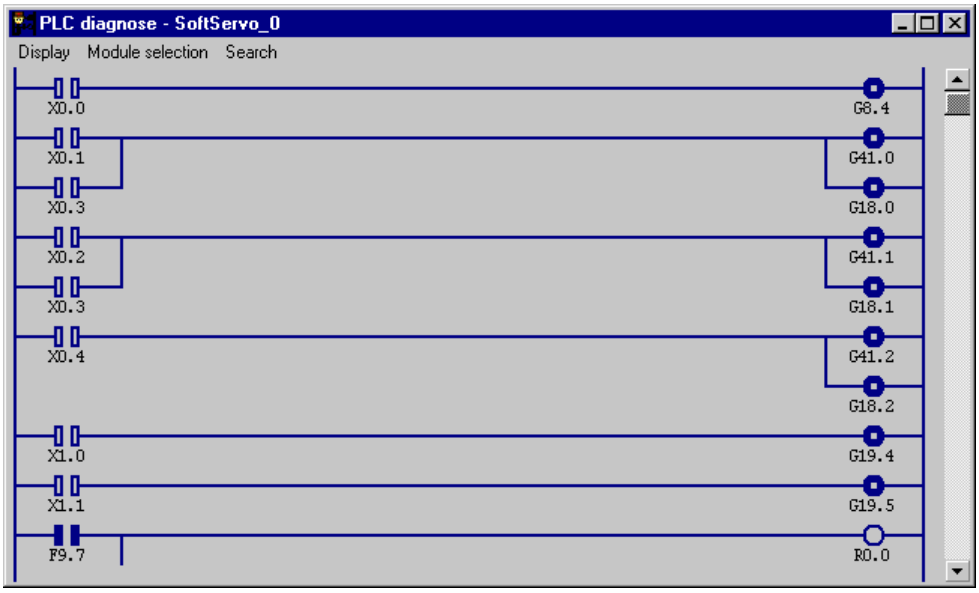


Figure 3-19: PLC Diagnose Window Without Comments for an Example Module

3.3.5 Using the Search Function of the PLC Ladder Monitor/Debugger

Click on "Search" from the menu in the PLC Diagnose Window, and you will see the window shown in the following figure:

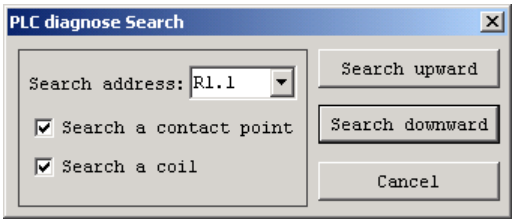


Figure 3-20: PLC Diagnose Search Window

To scan for an address in the PLC Diagnose Window, type the address you want to find in the "Search address" text box. You can search for a contact point, or a coil, or both. Click on either "Search upward" or "Search downward" buttons – if the address exists, the first instance of it will be highlighted in the PLC Diagnose Window.

3.4 Using the PLC Bit Pattern Utility

The PLC Bit Pattern utility shows the PLC bite data in binary format. You can see the 8-bit address in a byte address that you specify.

Start the PLC Ladder Monitor/Debugger utility by double clicking on the

file in C:\SwPLC. You will see the Bit Pattern window shown in the following figure:

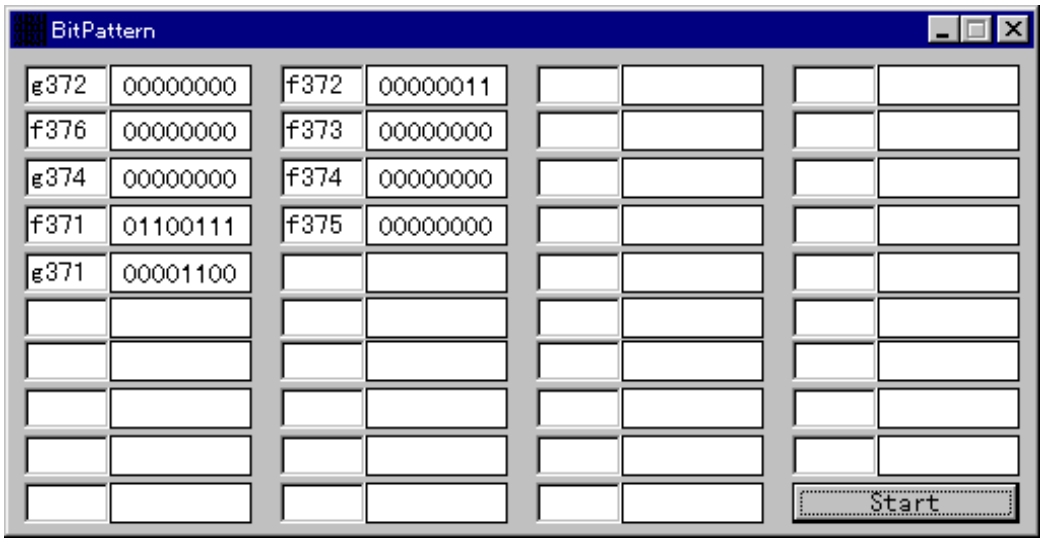
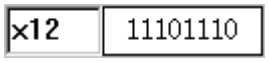


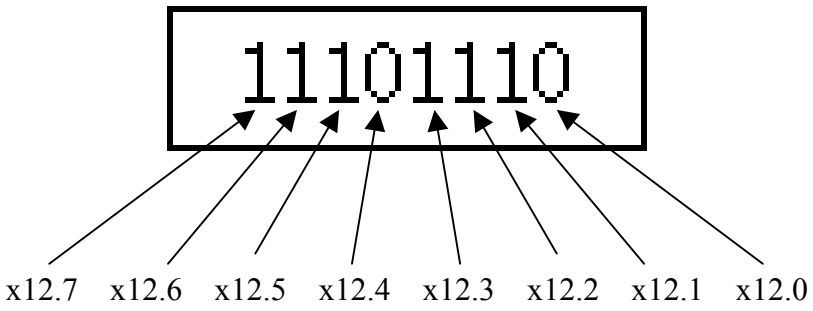
Figure 3-21: Bit Pattern Window

Type in a byte address, click the "Start" button, and you will see the current bit pattern.

For example, let's say you type in the address "x12," click on the "Start" button, and see the bit pattern "11101110," as shown:



In this case, the data is showing from the left side:



Therefore, the "ON" bits are the following six values: x12.7, x12.6, x12.5, x12.3, x12.2, and x12.1.

3.5 Using the PLC Time Chart Utility

The PLC Time Chart Utility can show PLC internal data as a time chart.
Start the PLC Ladder Monitor/Debugger utility by double clicking on the

file in C:\SwPLC. You will see the PLC Time Chart window shown in the following figure:

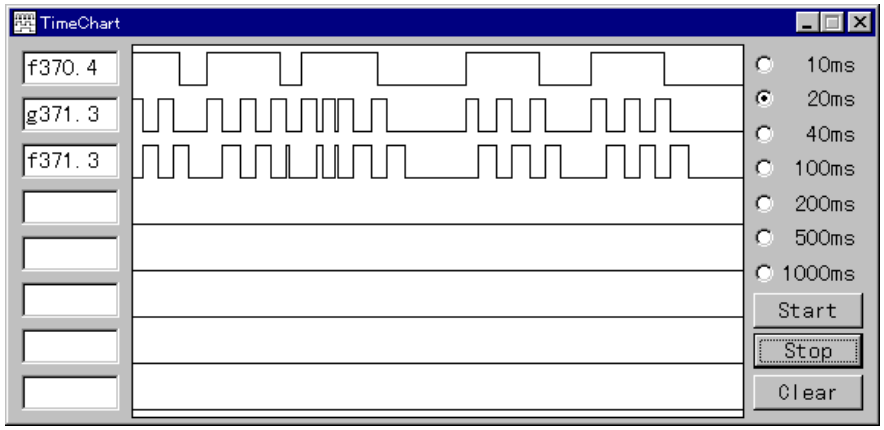


Figure 3-22: Time Chart Window

The following steps will guide you in using the PLC Time Chart Utility:

1) Input the address(es) for which you need to see the status.

2) At the right side of the window, set up the sampling interval by selecting the time in milliseconds.

3) Click on the "Start" button, and you will see the ON/OFF status in the center of the screen. During the sampling, you can change the sampling intervals by changing interval time.

4) Click on the "Stop" button to stop the sampling.

At any time, you can click on the "Clear" button to clear the window and delete the graph(s).

Chapter 4: Setting Up A Sequence Program

4.1 Description of the Sequence Program and Other PLC Files

4.1.1 Overview

The actual sequence program (the "`.lad`" file) is one of a series of files with the same name, but different file extensions, located by default in the folder "C:\SwPLC\Ladder." All but two of these files are automatically generated when you compile your sequence program. The two that are not automatically generated, which you must write, are the "`.lad`" and the "`.mod`" files.

4.1.2 .lad Files

A sequence program (the "`.lad`" file) is a way of expressing sequential logic to control a machine. The ServoWorks PLC Control Console application takes a representation of the system requirements (the PLC sequence program, in Instruction List Format) and converts it to an executable binary format so that it can be executed by the PLC Engine.

The procedure of creating a sequence program usually starts with writing PLC code in Instruction List (IL) Format using a text editor. This sequence program must be a text file with a "`.lad`" extension. The `.lad` file that comes with the ServoWorksPLC Application Suite needs to be modified or replaced with a sequence program that is customized for your machine.

Any text editor or word processor can be used to create a sequence program. That text file should then be saved as a `.lad` file.

A `.lad` file must have "%@3" for the first line, and "%" for the last line. Comments can be included in sequence programs by beginning a line with ";" or "//". Comments must be on separate lines. Each command should be on a separate line.

You can view the PLC code of your sequence program in a ladder diagram format to verify and debug your sequence program using the ServoWorksPLC Monitor/Debugger (described in *Chapter 3: Using the ServoWorksPLC Utility Programs*).

We strongly recommend that you begin with our default ladder file, included in the ServoWorksPLC Application Suite. A typical `.lad` file (which demonstrates how to correctly handle the HandWheel E-STOP) is shown as follows:

```
%@3

// E-Stop using HandWheel E-Stop input
RD      X0.0
WRT      G8.4

// Handwheel axis selection
RD      X0.1
OR      X0.3
WRT     G41.0
WRT     G18.0

RD      X0.2
OR      X0.3
WRT     G41.1
WRT     G18.1

RD      X0.4
WRT     G41.2
WRT     G18.2

// Handwheel multiple
RD      X1.0
WRT     G19.4
RD      X1.1
WRT     G19.5

////////////////////////////////////////////////////////////////
// The following three sections work for M30 program restart  //
// after rewind. If you would like to have this function,     //
// Please enable these three sections by removing the "//"    //
// comment signals and re-compile.                            //
////////////////////////////////////////////////////////////////
// Cycle Start Rising Edge Relay
RD      F0.5
AND.NOT R1.5
WRT     R1.6

RD      F0.5
WRT     R1.5

// M30 Falling Edge Relay
RD.NOT  F9.4
AND     R1.0
RD.STK  R1.4
AND.NOT R1.6
OR.STK
WRT     R1.4

RD      F9.4
WRT     R1.0
```

Figure 4-1: Typical .lad File: SoftServo_0 (1 of 2)

```
// Cycle start
RD      R1.4
//TMR   1           // With Timer delay
AND.NOT F0.5
WRT     G7.2

// Feed hold (M00, M01)
RD      F9.6
AND     F2.5
WRT     R2.0

RD      F9.7
AND     F7.0
OR      R2.0
WRT     G8.5

// MFIN (M00, M01, M02, M03, M04, M05, M30)
RD      F9.7
OR      F9.6
OR      F9.5
OR      F9.4
WRT     R0.0

RD      F7.0
AND     R0.0
WRT     R0.2

RD.NOT  F7.0
AND     R0.1
WRT     R0.3

RD      R0.2
AND.NOT R0.3
WRT     G5.0

RD      R0.0
WRT     R0.1

// SFIN (By S Strobe)
RD      F7.2
WRT     G5.2

// TFIN (By T Strobe)
RD      F7.3
WRT     G5.3

%
```

Figure 4-2: Typical .lad File: SoftServo_0.lad (2 of 2)

4.1.3 .mod Files

The "`.mod`" is a text file which you must create or verify with a "`.mod`" extension that tells the ServoWorksPLC Application Suite which files to reference. Specifically, the ".mod" file refers to the ".lad" file, and tells the compiler to compile the ".lad" file and generate a series of files with the same name as the ".lad" file, but with different extensions. It is the "`.mod`" file that is actually compiled when you compile your sequence program. The `.mod` file will always be one line. The default `.mod` file is shown as follows:

```
SoftServo_0   1
```

Figure 4-3: Example .mod File: SoftServo_0.mod

In this case, the name of the series of files will all be named "SoftServo_0," but will have different extensions: SoftServo_0.lad, SoftServo_0.bin, SoftServo_0.div, SoftServo_0.fig and SoftServo_0.lst.

4.1.4 .bin Files, .div Files, .fig Files and .lst Files

These files are automatically generated when you compile the .mod file.

The .bin file is the binary file for execution. The .div file contains the module divide table. The .fig table is the file used for real-time ladder diagram display. The .lst file has the assembly-code format and machine codes.

4.2 Overview of File Structure

In order to control the machine using a PLC, the sequence program files will be automatically set up according to Figures 4-4 and 4-5.

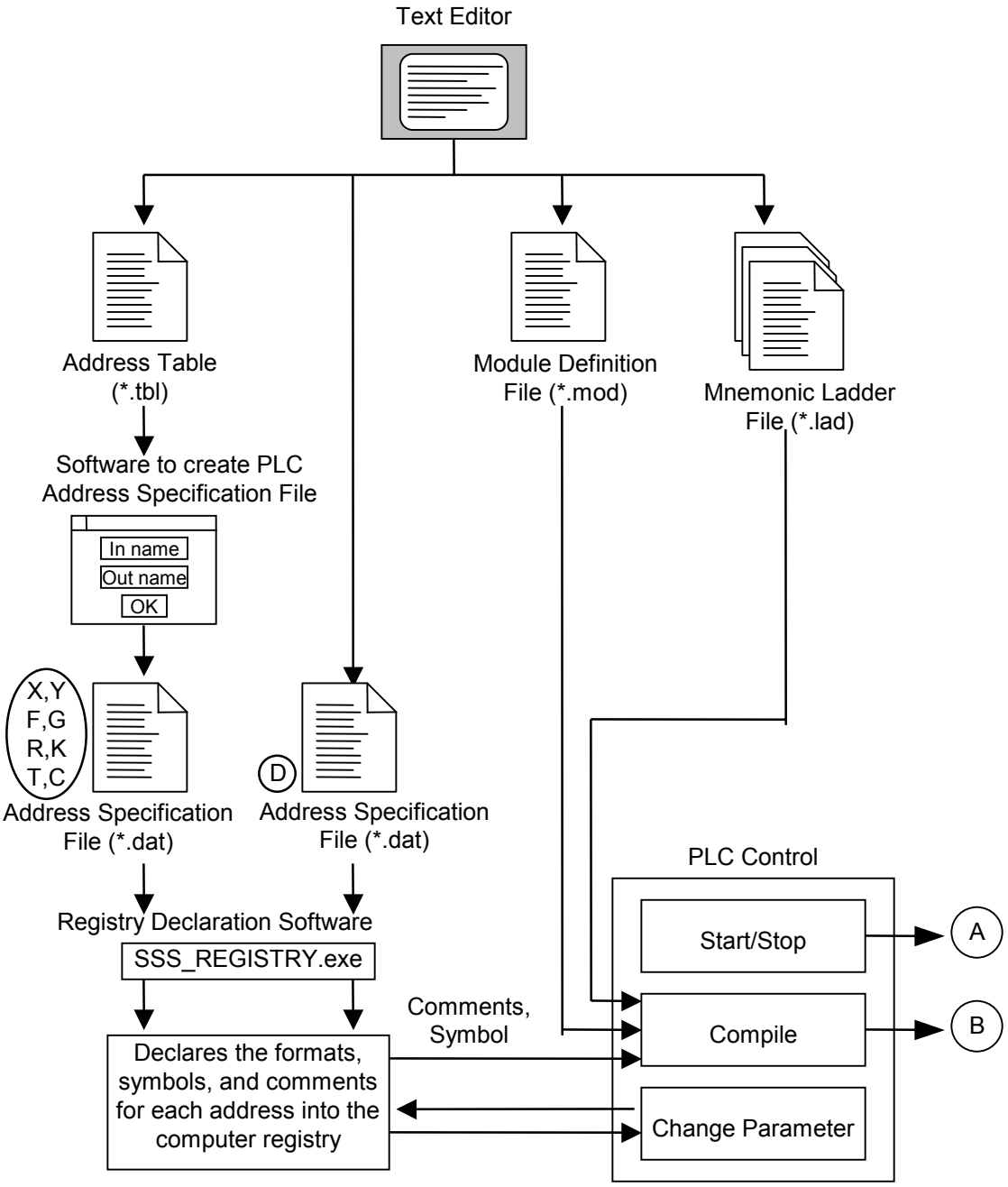


Figure 4-4: Sequence Program Setup Procedure (1 of 2)

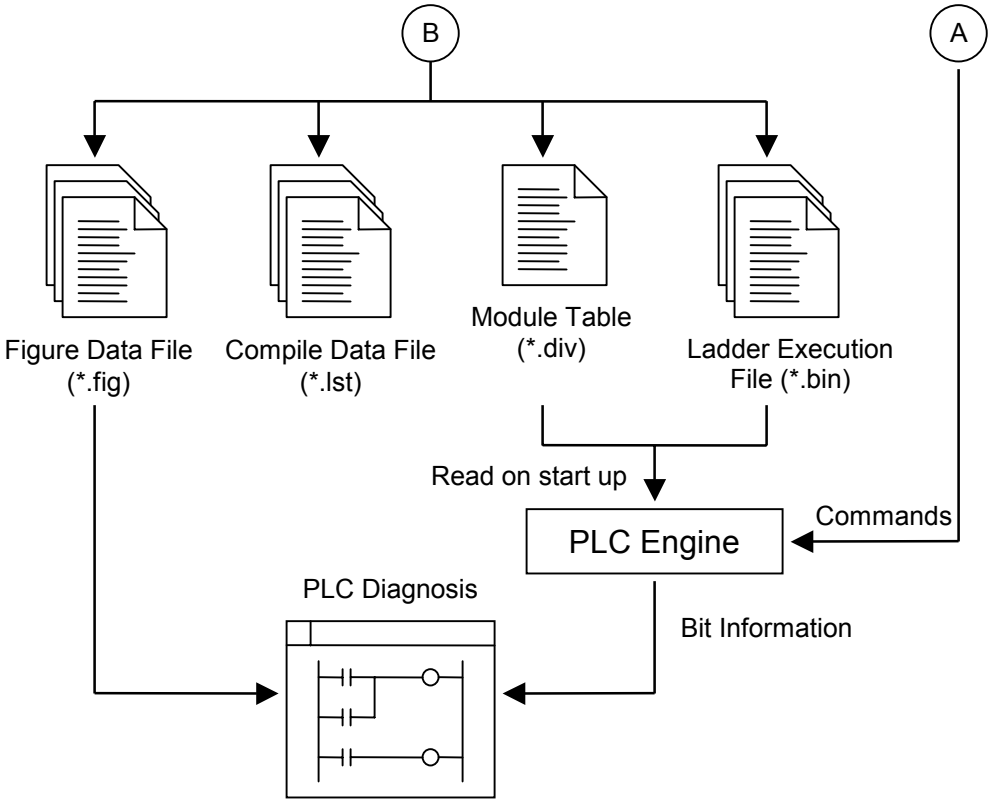


Figure 4-5: Sequence Program Setup Procedure (2 of 2)

4.3 Coding Convention

Writing the PLC commands in Instruction List Format that make up your sequence program and that represent the specifications of your machine is called "coding." An example of PLC code and the corresponding ladder diagram is shown in Figures 4-6 and 4-7, and Table 4-1.

```
%@3

// E-Stop using HandWheel E-Stop input
RD    X0.0
WRT   G8.4


// Handwheel axis selection
RD    X0.1
OR    X0.3
WRT   G41.0
WRT   G18.0

RD    X0.2
OR    X0.3
WRT   G41.1
WRT   G18.1

RD    X0.4
WRT   G41.2
WRT   G18.2
```

Figure 4-6: PLC Coding Example – Instruction List Format

Step No.	Command	Address No.	Bit No.	Description
1	RD	X0	. 0	HW_ESTP
2	WRT	G8	. 4	*ESP
3	RD	X0	. 1	HW_1
4	OR	X0	. 3	HW_3
5	WRT	G41	. 0	HS1IA
6	WRT	G18	. 0	HS1A
7	RD	X0	. 2	HW_2
8	OR	X0	. 3	HW_3
9	WRT	G41	. 1	HS1IB
10	WRT	G18	. 1	HW1B
11	RD	X0	. 4	HW_4
12	WRT	G41	. 2	HS1IC
13	WRT	G18	. 2	HS1C

Table 4-1: PLC Coding Example – Program Breakdown

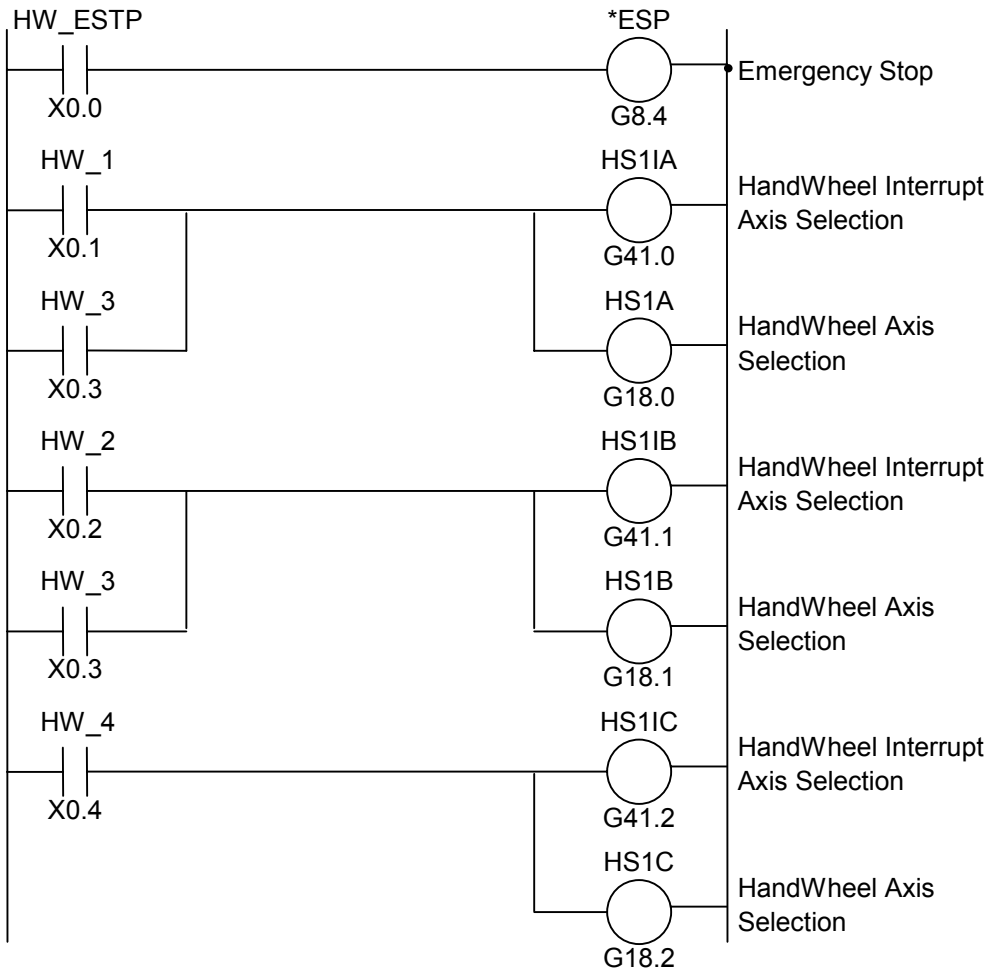


Figure 4-7: PLC Coding Example – Ladder Diagram Format

Chapter 5: Inside the PLC Engine (How the PLC Engine Operates)

The execution of a PLC sequence program by the PLC Engine is different from a usual relay circuit because it is simulated by software. Therefore, in designing a PLC sequence program, it is important that you understand the sequence of execution.

5.1 The Sequential Processing of the Sequence Program

In a usual relay sequence circuit, all relays can work simultaneously. For example, in Figure 5-1, when the relay A is turned on (and both B and C are turned off), both relays D and E work at the exact same time.

With the PLC Engine, each relay in a circuit works sequentially. For example in Figure 5-1, when relay A is on (and both B and C are turned off), first relay D, and then relay E is activated.

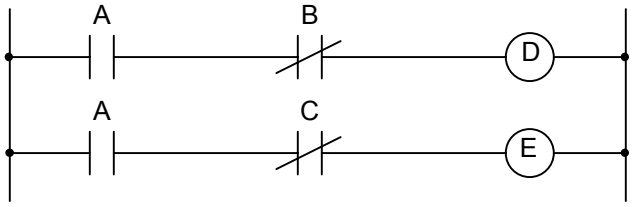


Figure 5-1: First Example of a Circuit

In other words, the sequence in a PLC program follows the sequence drawn on the ladder diagram (programming order). Though the sequence's execution is done very rapidly, it may affect the order of execution. Therefore, in a ladder diagram as in Figure 5-2, you can see a difference in execution between the PLC sequence and the relay circuit sequence.

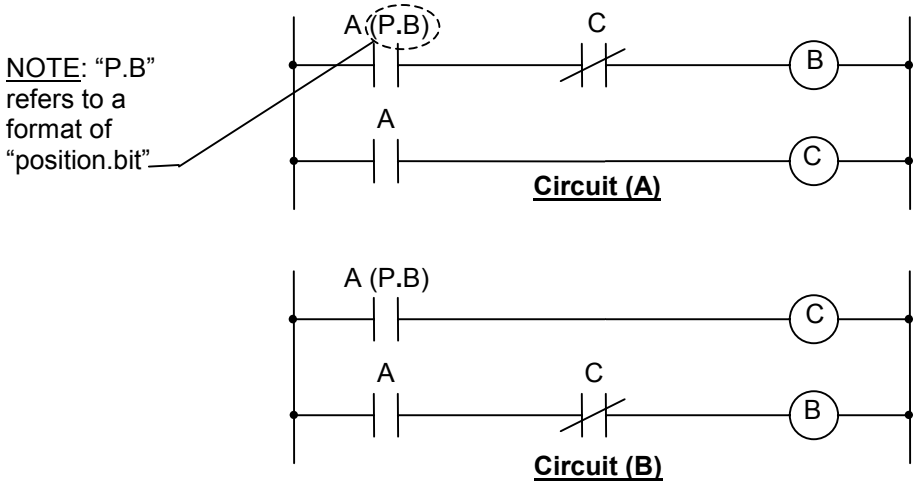


Figure 5-2: Second Example of a Circuit

In the relay circuit sequence: In Figure 5-2, both circuits (A) and (B) work simultaneously. When A (P.B) is turned on, current runs through coils B and C, and B and C are turned on at the same time. After C is turned on (after the relay execution), B turns off the circuit.

In the PLC sequence: In circuit (A), as in the relay circuit, when A (P.B) is turned on, B and C are both turned on, and after a certain period of time (1 cycle of PLC sequence), B turns the circuit off. However, in circuit (B), when A (P.B) is turned on, C is turned on, but B is not.

5.2 Repetitive Sampling

The ServoWorks PLC Engine samples values at discrete intervals (8 msec by default), so the sequence is run until the end of the ladder diagram; the sequence is then executed again from the beginning of the ladder diagram. The time it takes to complete the execution of a program from beginning to end (1 cycle) is called the sequence program's execution time. The execution time is determined by the control level (number of steps), and the size of the Number 1 level sequence, defined later. The faster the execution time, the more responsive the program gets.

5.3 I/O Signals

There are two kinds of input signals to the PLC Engine: input signals from the ServoWorks Motion Engine (i.e. signals sent as a result of M functions and T functions) and input signals from the machine (i.e. cycle start and feed/hold signals). There are also two kinds of output signals from the PLC Engine: output signals to the ServoWorks Motion Engine (i.e. cycle start, feed/hold) and output commands to the machine (i.e. talet rotation, spindle suspension). The input signals are fed into the input memory in the PLC Engine, and the output signals are the output of the PLC Engine.

5.3.1 Input

1) Input Memory from the ServoWorks Motion Engine: The input signals from the ServoWorks Motion Engine to the PLC Engine originate from the numerical control (NC) input memory specified by the ServoWorks Motion Engine, and are usually transferred to the PLC Engine in 5 ms periods.

2) Input Signals from the Machine: These signals are transferred from the input circuit to the input signal memory.

3) Input Signal Memory: The input signal memory holds the signals transferred from the machine to the PLC Engine in 8 ms periods.

5.3.2 Output

1) Output Memory to the ServoWorks Motion Engine: The output signal from the PLC Engine to the ServoWorks Motion Engine is sent to the NC output memory specified by the ServoWorks Motion Engine. The PLC Engine sends the data in 5 ms periods.

2) Output Signals to the Machine: The output signals to the machines are transferred from the PLC's output signal memory to the output circuit.

3) Output Signal Memory: Output signal memory is the memory specified by the PLC sequence program. The signals to the output signal memory are sent to the machine in 8 ms periods by the PLC Engine.

5.4 PLC Code Execution

PLC code is executed one command at a time. For example, the instruction sequence "RD X0.0, AND R10.1, WRT Y0.0" does the following:

1) On the first command (RD X0.0), the input signal at address X0.0 is inserted into the computation register.

2) On the next command (AND R10.1), it takes the internal relay value at the address R10.1, computes the logical AND with the current computational register value, and puts it back into the computational register.

3) On the last command (WRT Y0.0), it stores the value in the computational register to the output signal at address Y0.0.

Once you write the PLC code for a sequence program, you can use the ServoWorksPLC Control Console application to convert it into machine language for the computer (Figures 4-4 and 4-5).

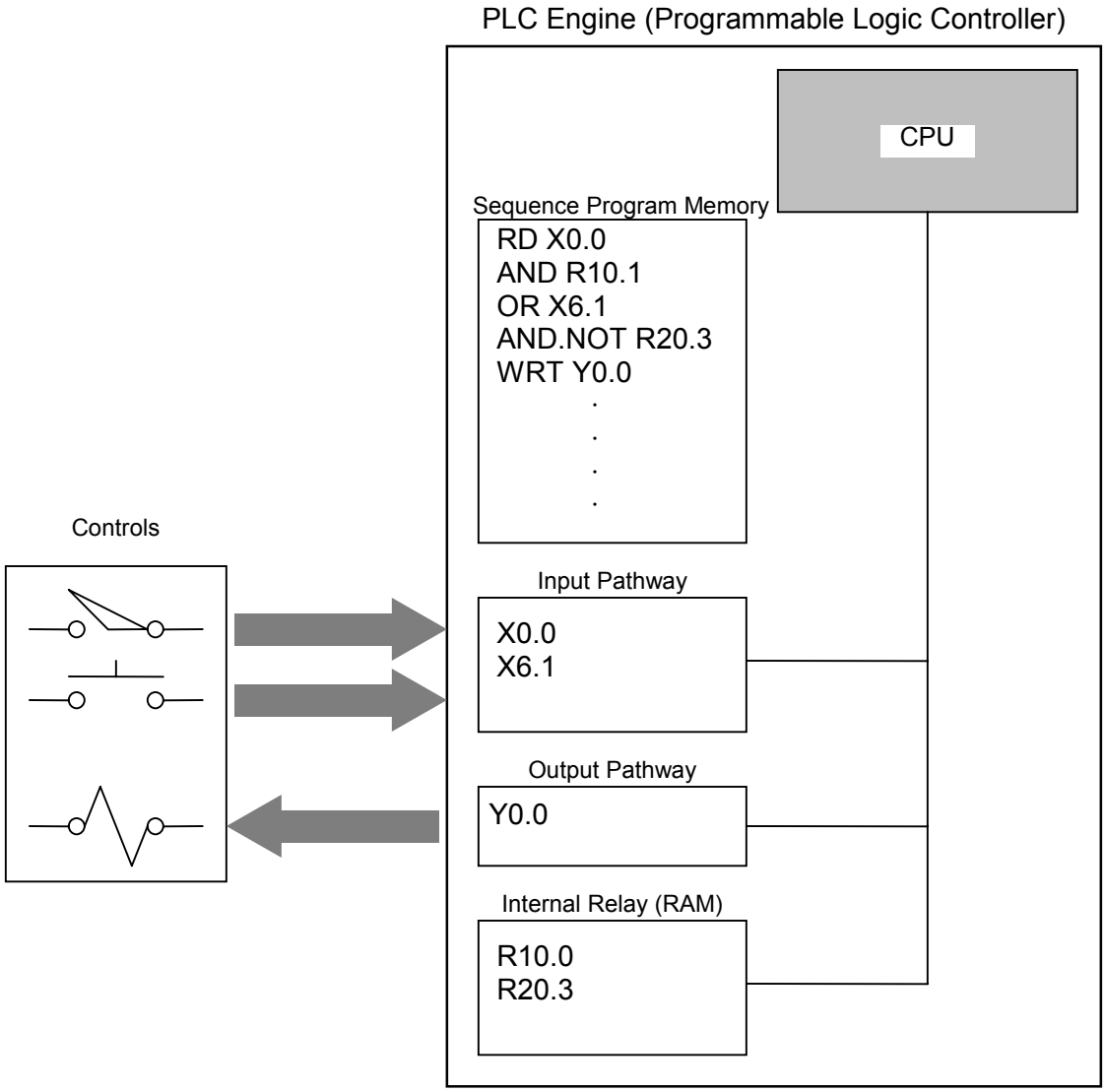


Figure 5-3: Execution of Sequence Program by the PLC Engine

Chapter 6: Memory Addresses

6.1 What Are Memory Addresses?

An address is the number that represents the location in memory of where I/O signals to and from the machine, I/O signals to and from the ServoWorks Motion Engine, internal relays, counters, keep relays (parameters for the PLC sequence program), and data cables reside. An address consists of an address number (each one contains eight signals) and a bit number (0-7) to specify which of the eight. The symbol table that shows the signal name and the respective address is created using the PLC Control Console Application.

6.2 Addresses Related to the PLC

The addresses used by the PLC's sequence program can be divided roughly into 4 different types, as shown in Figure 6-1:

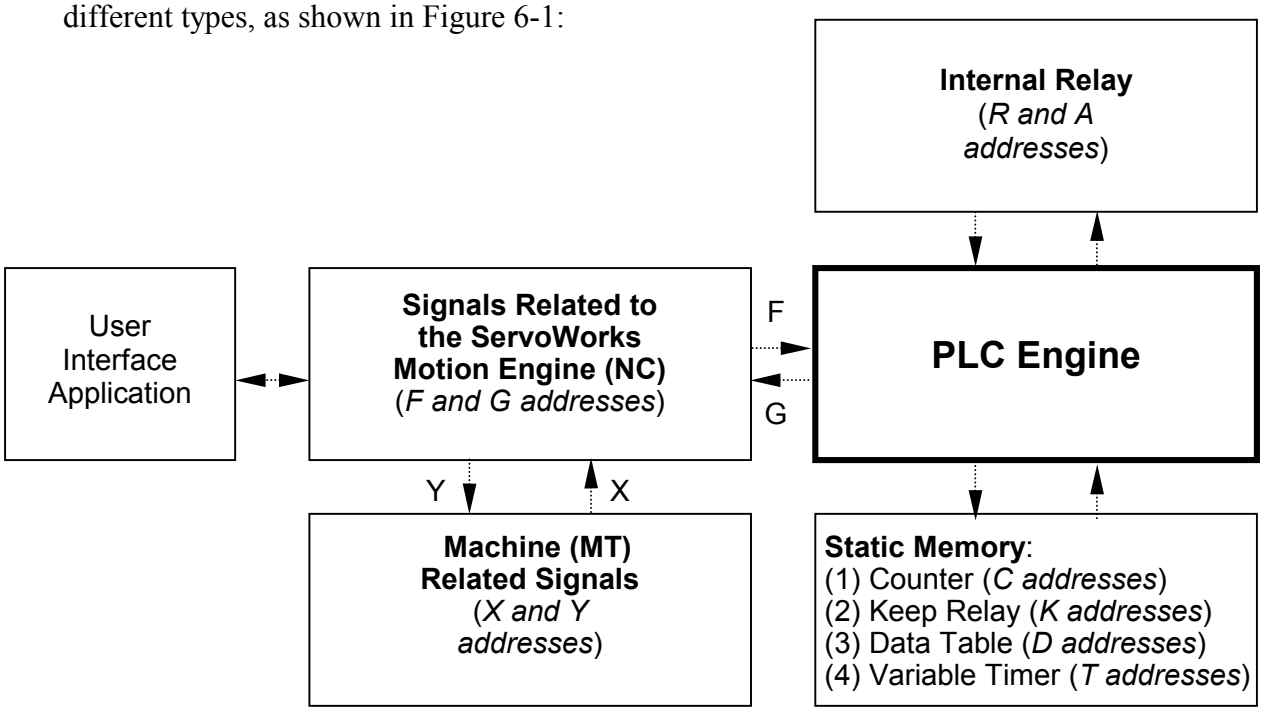
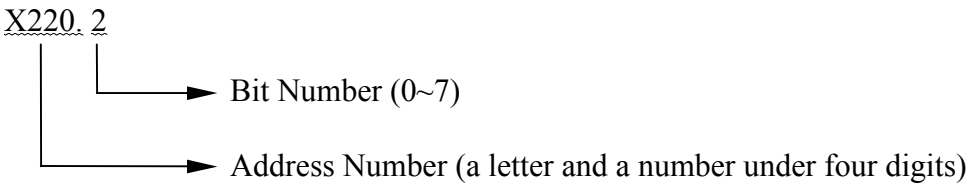


Figure 6-1: Addresses Related to the PLC Engine

6.3 Address Specifications

An address consists of an address and a bit number in the following format:



The first character of the address is always a letter of the alphabet representing the type of signal as shown in Table 6-1. When performing byte-level addressing within a function command, use X220. The "." and the bit number is not necessary in this case.

Symbol	Type of Signal
X	Input signals from the machine to the PLC Engine. (MT → PLC)
Y	Output signals (commands) from the PLC Engine to the machine. (PLC → MT)
F	Input signals from the ServoWorks Motion Engine to the PLC Engine. (NC→ PLC)
G	Output signals from the PLC Engine to the ServoWorks Motion Engine. (PLC → NC)
R	Internal Relay
C	Counter
K	Keep Relay
D	Data Table
T	Variable Timer
A	Alarm

Table 6-1: Symbols for the Signal Types

6.4 PLC and Machine Tool Addresses (PLC ↔ MT)

Addresses related to the machine are as follows:

a) PLC Engine ←MT (machine tool):
Address Range: X0~X99

b) PLC Engine →MT (machine tool):
Address Range: Y0~Y99

Within the byte ranges above, a maximum of 800/800 I/O signals can be declared, though they must be allocated in one-byte increments (space for 8 signals).

For the addresses of specific signals, refer to *Appendix A: S-100T Data Mapping Tables* or *Appendix B: S-100M and General Motion Data Mapping Tables*, which lists the name and address of each specific signal included in the ServoWorks system. Input signals from the ServoWorks Motion Engine have fixed addresses.

6.5 PLC Engine and ServoWorks Motion Engine Addresses (PLC ↔ NC)

Addresses related to the ServoWorks Motion Engine are as follows:

a) PLC Engine ←ServoWorks Motion Engine Signals:
Address Range: F0~F399

b) PLC Engine →ServoWorks Motion Engine Signals:
Address Range: G0~G399

For the addresses of specific signals, refer to *Appendix A: S-100T Data Mapping Tables* or *Appendix B: S-100M and General Motion Data Mapping Tables*, which lists the name and address of each specific signal included in the ServoWorks system.

6.6 Internal Relay Addresses (R)

The internal relay can use 1000 bytes, at addresses R0~R999 (see Figure 6-2). This region will be refreshed every time the power is turned on.

The addresses R9000~R9099 are reserved by the PLC, so they cannot be used in the sequence program. The results of the function commands are stored in this reserved address space. Refer to the following figure:

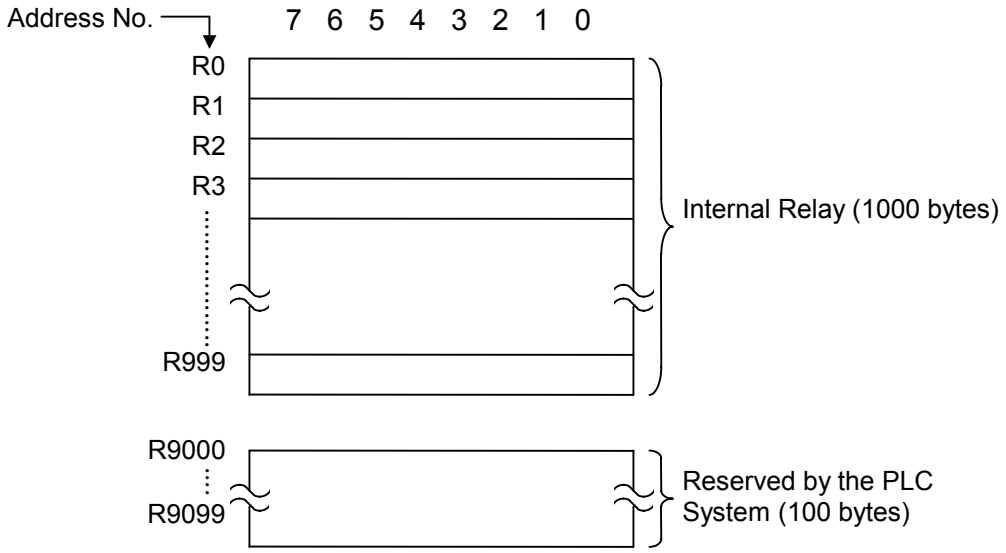


Figure 6-2: Internal Relay Usable Region

6.7 Counter Addresses (C)

Addresses C0~C79 (80 bytes of memory) are designated for the counters. The contents in this memory are not erased when the power is off because this region holds static memory. Refer to the following figure:

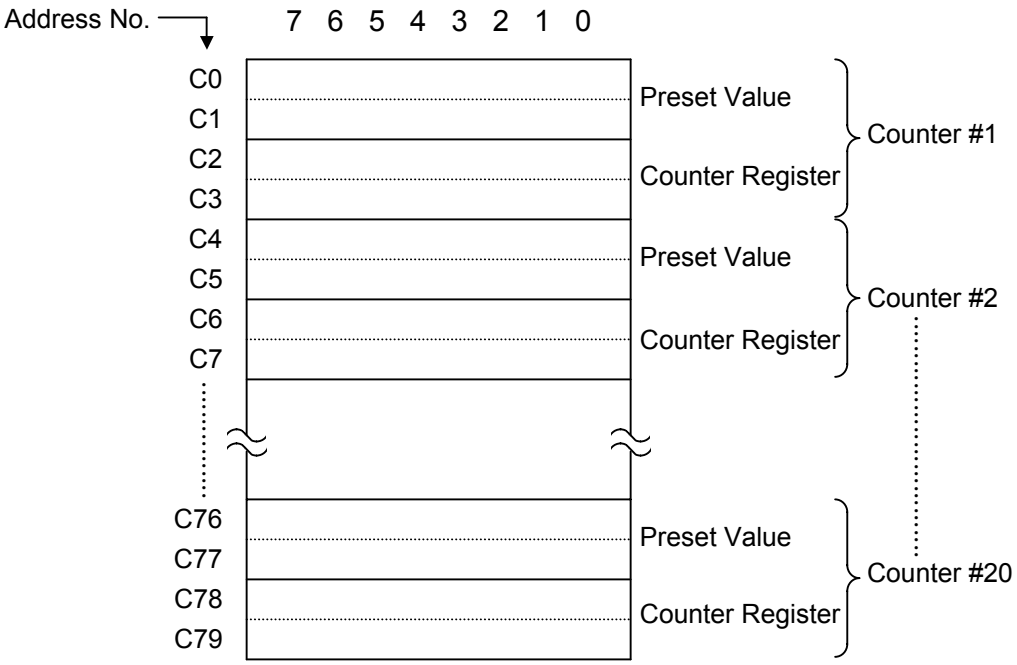


Figure 6-3: Counter Addresses

6.8 Keep Relay and Static Memory Control Addresses (K)

Addresses K0~K99 (100 bytes of memory) are used to store the keep relay data and the PLC parameters. This is also static memory. Refer to the following figure:

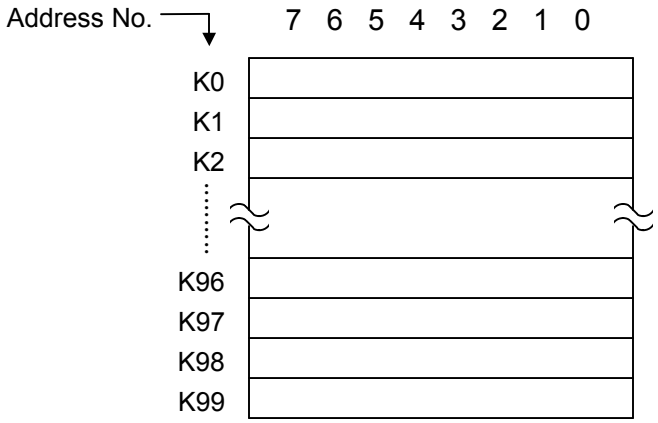


Figure 6-4: Keep Relay and Static Memory Control Addresses

6.9 Data Table Addresses (D)

The data table is stored in static memory. The basic data table resides at addresses D0~D1999 (2000 bytes of memory). Refer to the following figure:

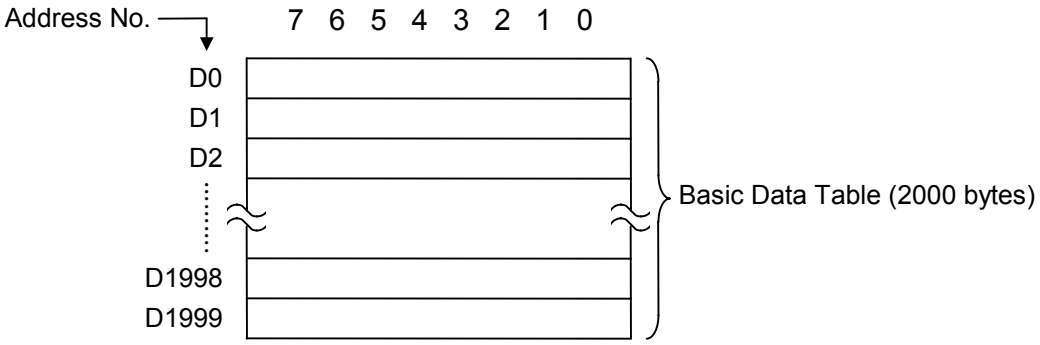


Figure 6-5: Data Table Addresses

6.10 Timer Addresses (T)

The variable timers used in TMR commands are allocated 400 bytes of static memory, at addresses T0~T399.

The addresses corresponding to each timer are shown in Figure 6-6. Since this region is static memory, the contents are not erased when the power is turned off. Refer to the following figure:

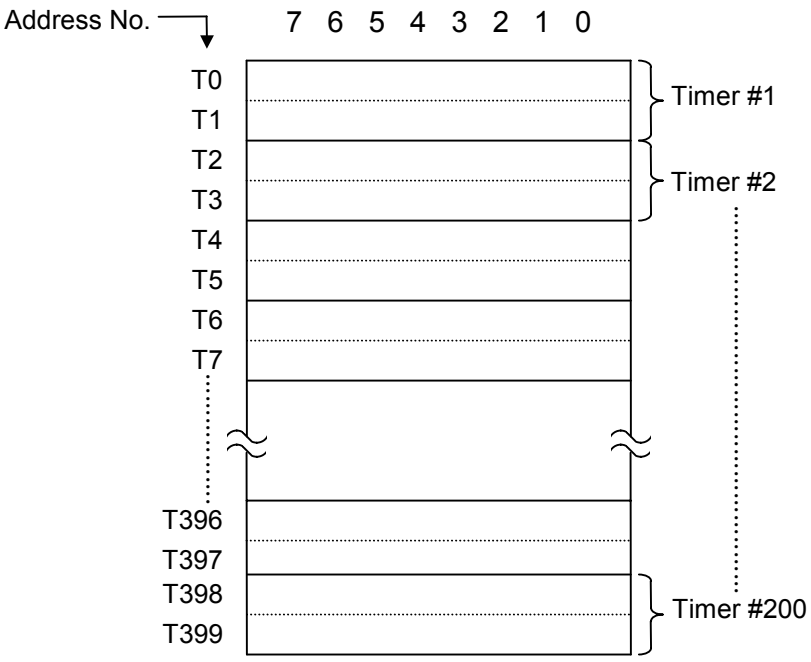


Figure 6-6: Timer Addresses

6.11 Alarm Relay Addresses (A)

The alarm relay has 100 bytes of memory, at addresses A0~A99. Refer to the following figure:

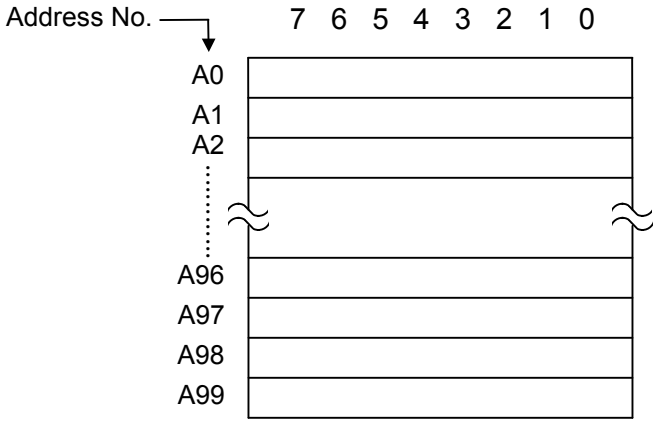


Figure 6-7: Alarm Relay Addresses

Chapter 7: Static Memory

7.1 Timer, Counter, Keep Relay, Static Memory Control, Data Table

7.1.1 Overview of Static Memory

Static memory is the memory where the information contained in that memory is not erased even when the power is off. The PLC Engine uses static memory for the following:

- Timer
- Counter
- Keep Relay
- Static Memory Control
- Data Table

7.1.2 Timer

Static memory is used to specify the time for the timer. You can display and set the timer with the PLC Control Console application (the T Table). You can also read and write the time using the sequence program.

7.1.3 Counter (Addresses C0~C79)

Static memory is used to store the counter preset value and the increment value. You can display and set the counter with the PLC Control Console application (the C Table). You can also read and write into the counter using the sequence program. For details about the counter addresses, refer back to *Section 6.7: Counter Addresses (C)*. The format of the data is either 2 bytes of BCD or binary with the higher digits corresponding to the higher addresses. The address can be in BCD or binary, as specified in the PLC system parameters.

Example: When the PLC counter addresses are C0 and C1 and the preset value is 1578.

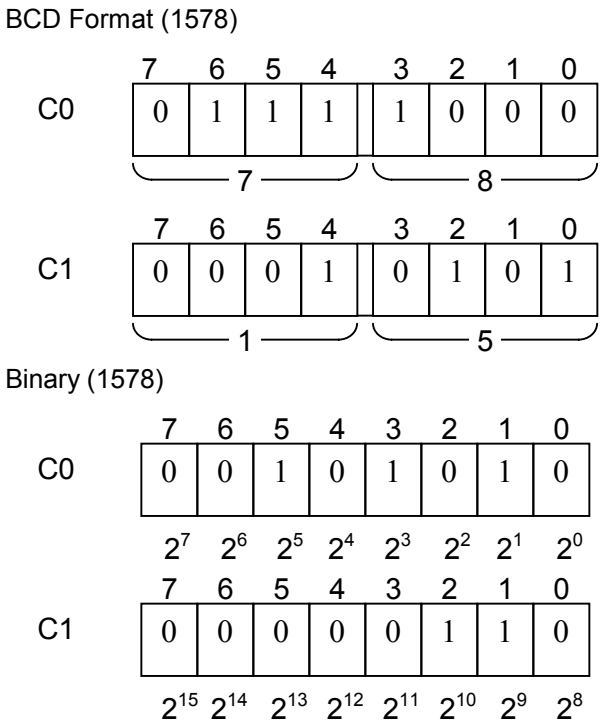


Figure 7-1: Example of Counter Addresses

In order to set the lower 2 digits of the value to the output of a command that outputs 1-byte data, then specify C0 as the output address for that command.

7.1.4 Keep Relay (Addresses K0~K99)

Static memory holds parameters for the program such as the keep relay. You can read out or set the value with the PLC Control Console application (the K Table). You can also read and write from within the sequence program. The PLC control screen handles data as an 8-bit binary, so each digit holds the value of "0" or "1."

7.1.5 Data Table (Addresses D0~D1999)

You can use a set of numeric data (a data table) for PLC sequence control. For details, refer to *Section 7.3: PLC Data Table*.

7.2 Reading and Writing Static Memory

The sequence program can read and write any data in static memory. However, the memory accessed by the sequence program is not the static memory, but an exact image of the static memory (RAM). Therefore, data inside the image disappears when the power is off, but that data is copied again from static memory when the power is turned on, restoring properly.

The system uses a write-through cache, where data modified in RAM by the sequence program is automatically transferred to the static memory.

Modifications of data in the image can be done at any time and at any frequency, and the data will be transferred to static memory. Therefore, writing to static memory does not require any special handling. However, it does take some time for the data to propagate, or be written, to static memory (about 512 ms).

7.3 PLC Data Table

7.3.1 Overview

PLC sequential control sometimes requires a set of numeric data (henceforth called a data table). Being able to write to and read from this data table is useful. For example, it can supply the tool number of each tool in the ATC Magazine to the program. You can create a maximum of 50 data tables.

Within the memory constraints of a table, you can set the data table to 1-, 2-, or 4-byte length words encoded either in binary or BCD. Therefore, you can easily make useful data structures that do not waste space.

You can access data in the data table through the static memory, which in turn is accessed by the PLC control screen.

This data can also be accessed using functional commands in the program, such as Data Search (DSCHB) and Index Modification Data Transfer (XMOVB).

7.3.2 Creation of Data in the Data Table

The data in the data table is created with the PLC Control Console application (the D Table).

NOTE: You can also read and write into the data table using the sequence program.

Chapter 8: PLC Basic Commands

8.1 Overview

The first step in designing a sequence program is to sketch out a ladder diagram. The ladder diagram consists of relay junctions and functional command symbols described later, in *Chapter 9: PLC Functional Commands*. Then you code the ladder diagram into Instruction List format, and verify the coding by using the ServoWorks Monitor/Debugger to view the ladder diagram representation of that Instruction List code, and make sure the logic represented by the ladder diagram is what you intended. When you are satisfied with your code, compile your Instruction List sequence program into machine code with the ServoWorksPLC Control Console application. This machine code is then fed into the PLC Engine as the sequence program.

You will be coding using mnemonic representation (PLC commands such as RD, AND, and OR) of PLC logic. You should, however, understand relay symbols (such as ⊣⊢, ⊣/⊢, -◯-) and the functional command symbols used in the ladder diagram.

You should have a thorough understanding of PLC basic commands to understand the details of the functional commands presented later. Therefore, you should read carefully through the rest of this chapter, and *Chapter 9: PLC Functional Commands* before coding your sequence program for your ServoWorks system.

This chapter concerns itself with basic PLC commands. For each basic command, we must concern ourselves with the following items:

- Signal addresses.
- Types of commands (basic and functional).
- Storing the results of logical operations.

8.1.1 Signal Addresses

An address can be assigned to all signals, relay coils and junctions, drawn in the ladder diagram shown in Figure 8-1. An address consists of an address number and a bit number. The leading zero can optionally be suppressed. For more details about addresses, see *Chapter 6: Memory Addresses*.

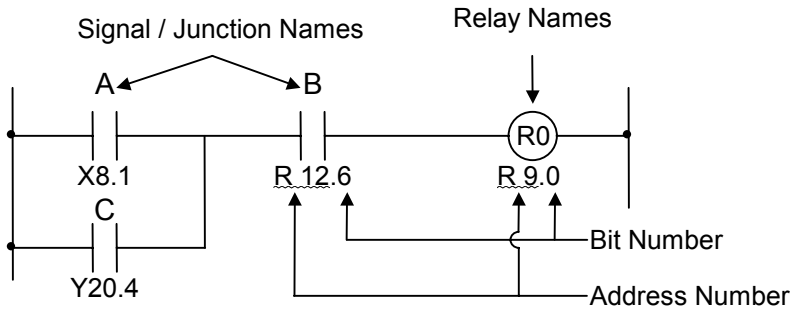


Figure 8-1: Signal Addresses

8.1.2 Types of Commands (Basic and Functional)

Basic commands are the commands you will use most frequently in sequence programs. There are twelve basic commands including AND, OR, and other byte level operations.

Functional commands are commands that make the programming of the complex controls of machinery much easier than just using basic commands. Refer to *Chapter 9: PLC Functional Commands* for descriptions of the functional commands.

8.1.3 Storing the Results of Logic Operations in the Result History Register

A sequence program can store intermediate results in a FIFO (first-in, first-out) stack register known as the Result History Register. This register contains 1 bit + 8 bits = 9 bits (see Figure 8-2).

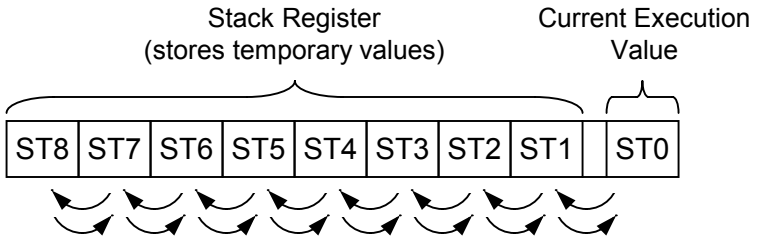


Figure 8-2: Structure of the Result History Register

As a push command (i.e. RD.STK) is executed, the current execution value is stored in ST1 and the other values shift to the left as shown in Figure 8-2. When a pop command (i.e. AND.STK) is executed, the values shift to the right, and the value on the top of the stack (ST1) is moved into the current execution environment. For the specifics of each command, refer to their respective sections later in this chapter.

8.2 Basic Commands

8.2.1 Summary of Basic Commands

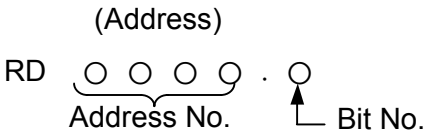
Table 8-1 shows the kinds of basic commands and their functions. Detailed descriptions follow.

No.	Command	Function
1	RD	Reads the value of the signal and puts it in ST0.
2	RD.NOT	Reads the inverted value of the signal and puts it in ST0.
3	WRT	Outputs the result (value of ST0) into the specified address.
4	WRT.NOT	Outputs the inverted result (value of ST0) into the specified address.
5	AND	Logical AND (Product). Performs a logical AND with the specified signal and the existing value (ST0).
6	AND.NOT	Inverts the value of the specified signal and performs a logical AND with the existing value.
7	OR	Logical OR (Sum). Performs a logical OR with the specified signal and the existing value (ST0).
8	OR.NOT	Inverts the value of the specified signal and performs a logical OR with the existing value.
9	RD.STK	Shifts the register contents left one bit and puts the value of the signal with the specified address into ST0.
10	RD.NOT.STK	Same as RD.STK, but stores the inverted signal value into ST0.
11	AND.STK	Stores AND of ST0 and ST1 into ST1, then shifts all of the bits in the register to the right one bit.
12	OR.STK	Stores OR of ST0 and ST1 into ST1, then shifts all of the bits in the register to the right one bit.

Table 8-1: PLC Basic Commands and Their Functions

8.2.2 RD Command

Format



Function

This command reads the value of a signal at a specified address ("1" or "0"), and puts it into the ST0 bit in the result history register.

Use

When the code starts with junction A (⊣⊢), RD is used. For an example, see the ladder diagram in Figure 8-3 and the coding sheet entry example in Table 8-2.

Signal

The signal (junction) read by the RD command could be any signal used in the logical expression for a coil (output).

Example of RD Command Usage

Tables 8-2 and 8-3 show two alternative ways to code the ladder diagram in Figure 8-3, in different orders. Both orders lead to the same result.

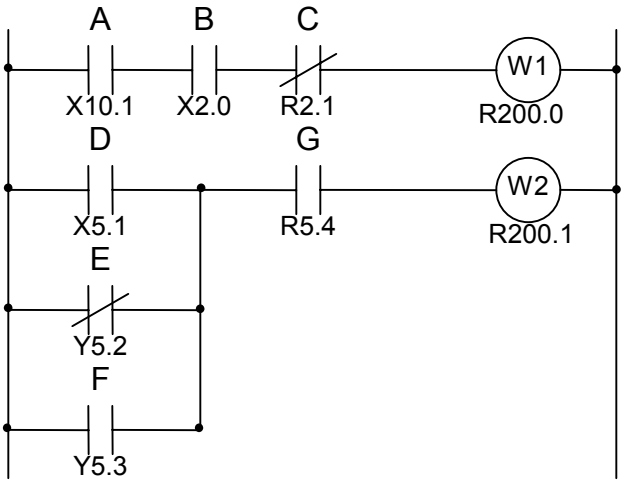


Figure 8-3: Ladder Diagram Example for the RD Command

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD	X10 . 1				A
2	AND	X2 . 0				A • B
3	AND.NOT	R2 . 1				A • B • $\overline{C}$
4	WRT	R200 . 0	W1 out			A • B • $\overline{C}$
5	RD	X5 . 1				D
6	OR.NOT	Y5 . 2				D + $\overline{E}$
7	OR	Y5 . 3				D + $\overline{E}$ + $\overline{F}$
8	AND	R5 . 4				(D + $\overline{E}$ + $\overline{F}$) • G
9	WRT	R200 . 1	W2 out			(D + $\overline{E}$ + $\overline{F}$) • G

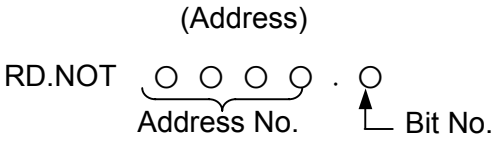
Table 8-2: Coding of the RD Command Example (Alternative #1)

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD	X2 . 0				B
2	AND	X10 . 1				B • A
3	AND.NOT	R2 . 1				B • A • $\overline{C}$
4	WRT	R200 . 0	W1 out			B • A • $\overline{C}$
5	RD	Y5 . 3				F
6	OR.NOT	Y5 . 2				F + $\overline{E}$
7	OR	X5 . 1				F + $\overline{E}$ + D
8	AND	R5 . 4				(F + $\overline{E}$ + D) • G
9	WRT	R200 . 1	W2 out			(F + $\overline{E}$ + D) • G

Table 8-3: Coding of the RD Command Example (Alternative #2)

8.2.3 RD.NOT Command

Format



Function

This command reads the inverted value of a specified signal and puts it into the ST0 bit in the result history register.

Use

When the code starts with junction B (⊣/⊢), use the RD.NOT command. For an example, see the ladder diagram in Figure 8-4 and the coding sheet entry example in Table 8-4.

Signal

The signal (junction) read by the RD.NOT command could be any signal used in the logical expression for a coil (output).

Example of RD.NOT Command Usage

Tables 8-4 and 8-5 show two alternative ways to code the ladder diagram in Figure 8-4, in different orders. Both orders lead to the same result.

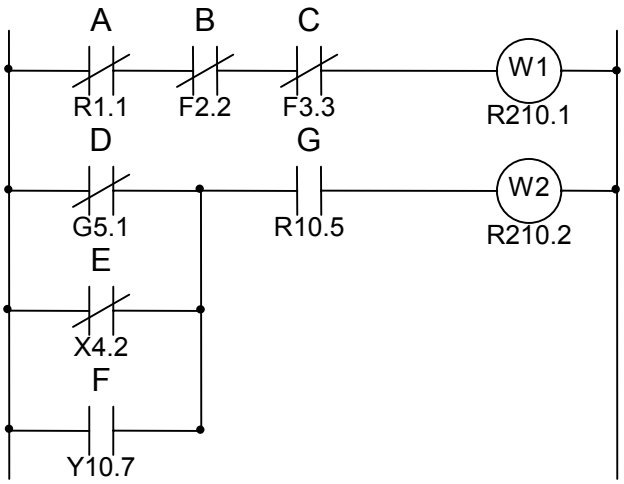


Figure 8-4: Ladder Diagram Example for the RD.NOT Command

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD.NOT	R1 . 1				$\overline{A}$
2	AND.NOT	F2 . 2				$\overline{A} \cdot \overline{B}$
3	AND.NOT	F3 . 3				$\overline{A} \cdot \overline{B} \cdot \overline{C}$
4	WRT	R210 . 1	W1 out			$\overline{A} \cdot \overline{B} \cdot \overline{C}$
5	RD.NOT	G5 . 1				$\overline{D}$
6	OR.NOT	X4 . 2				$\overline{D} + \overline{E}$
7	OR	Y10 . 7				$\overline{D} + \overline{E} + F$
8	AND	R10 . 5				$(\overline{D} + \overline{E} + F) \cdot G$
9	WRT	R210 . 2	W2 out			$(\overline{D} + \overline{E} + F) \cdot G$

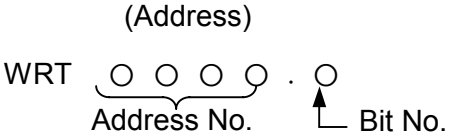
Table 8-4: Coding of the RD.NOT Command Example (Alternative #1)

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD.NOT	F2 . 2				$\overline{B}$
2	AND.NOT	F3 . 3				$\overline{B} \cdot \overline{C}$
3	AND.NOT	R1 . 1				$\overline{B} \cdot \overline{C} \cdot \overline{A}$
4	WRT	R210 . 1	W1 out			$\overline{B} \cdot \overline{C} \cdot \overline{A}$
5	RD.NOT	X4 . 2				$\overline{E}$
6	OR	Y10 . 7				$\overline{E} + F$
7	OR.NOT	G5 . 1				$\overline{E} + F + \overline{D}$
8	AND	R10 . 5				$(\overline{E} + F + \overline{D}) \cdot G$
9	WRT	R210 . 2	W2 out			$(\overline{E} + F + \overline{D}) \cdot G$

Table 8-5: Coding of the RD.NOT Command Example (Alternative #2)

8.2.4 WRT Command

Format



Function

This command writes the result of the logic operation, the value of the ST0 bit in the result history register ("1" or "0"), into the specified address.

Use

See Tables 8-2, 8-3, 8-4 and 8-5 for examples of how to use the WRT command.

Signal

You can output a logic operation result into two or more addresses, as shown in the example that follows.

Example of WRT Command Usage

See Figure 8-5 and Table 8-6 for examples of how to use the WRT command.

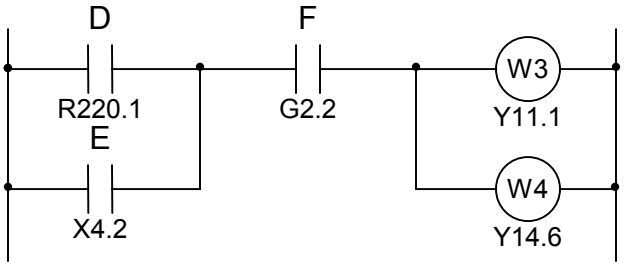


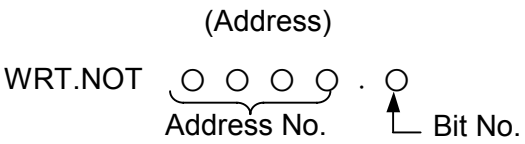
Figure 8-5: Ladder Diagram Example for the WRT Command

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD	R220 . 1				D
2	OR	X4 . 2				D • E
3	AND	G2 . 2				(D + E) • F
4	WRT	Y11 . 1	W3 out			(D + E) • F
5	WRT	Y14 . 6	W4 out			(D + E) • F

Table 8-6: Coding of the WRT Command

8.2.5 WRT.NOT Command

Format



Function

This command writes the inverse of the result (value of the ST0 bit in the Result History Register) into the specified address.

Example of WRT.NOT Command Usage

See Figure 8-6 and Table 8-7 for an example of how to use the WRT.NOT command.

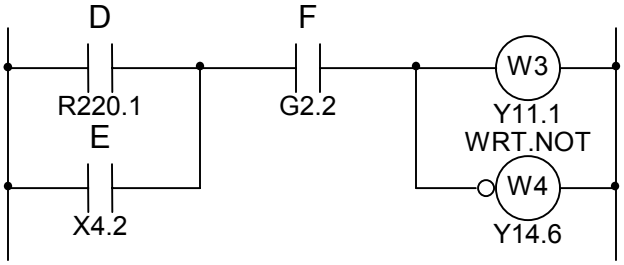


Figure 8-6: Ladder Diagram Example for the WRT.NOT Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	R220	1	
2	OR	X4	2	
3	AND	G2	2	
4	WRT	Y11	1	W3 out
5	WRT.NOT	Y14	6	W4 out

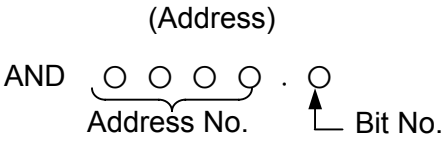
Result History Register

ST2	ST1	ST0
		D
		D • E
		(D + E) • F
		(D + E) • F
		$\overline{(D + E) \cdot F}$

Table 8-7: Coding of the WRT.NOT Command

8.2.6 AND Command

Format



Function

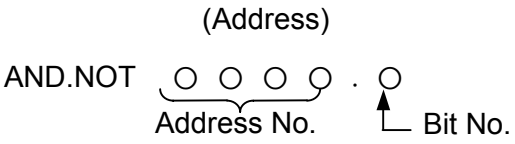
This command takes the value of a specified signal and executes the logical AND (product) with the existing value.

Example of AND Command Usage

See Figure 8-3 and Table 8-2 for an example using the AND command.

8.2.7 AND.NOT Command

Format



Function

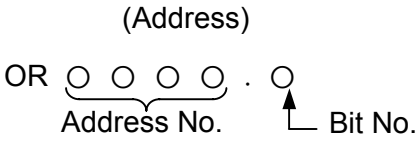
This command inverts the value of a specified signal and executes the logical AND.NOT (inverse product) with the existing value.

Example of AND.NOT Command Usage

See Figure 8-3 and Table 8-2 for an example of how to use the AND.NOT command.

8.2.8 OR Command

Format



Function

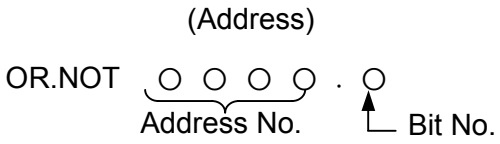
This command takes the value of a specified signal and executes the logical OR (sum) with the existing value.

Example of OR Command Usage

See Figure 8-3 and Table 8-2 for an example using the OR command.

8.2.9 OR.NOT Command

Format



Function

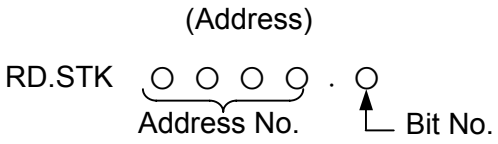
This command inverts the value of a specified signal and executes the logical OR.NOT (inverse sum) with the existing value.

Example of OR.NOT Command Usage

See Figure 8-3 and Table 8-2 for an example of how to use the OR.NOT command.

8.2.10 RD.STK Command

Format



Function

This command pushes the intermediate calculation onto the stack. It shifts each bit in the register one bit to the left and puts the value of the signal with the specified address into the ST0 bit of the result history register.

Use

Use this command when the specified address is the A junction (⊣⊢).

Example of RD.STK Command Usage

See Figure 8-7 and Table 8-8 for examples of how to use the RD.STK command.

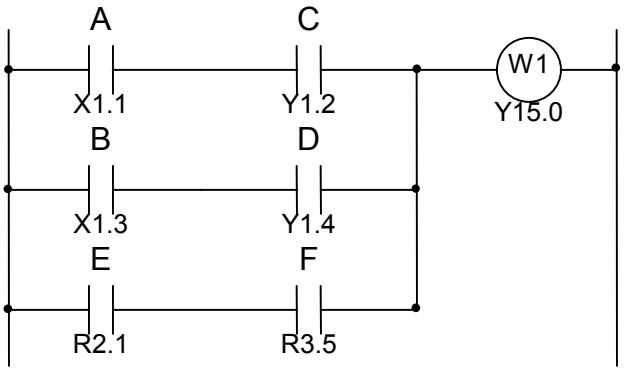


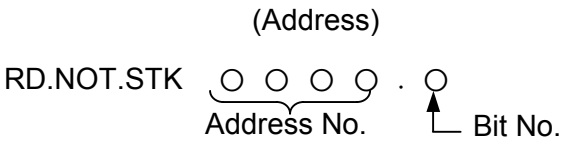
Figure 8-7: Ladder Diagram Example for the RD.STK Command

Coding Sheet				Result History Register		
Step No.	Command	Address No. Bit No.	Description	ST2	ST1	ST0
1	RD	X1 . 1				A
2	AND	Y1 . 2				A • C
3	RD.STK	X1 . 3			A • C	B
4	AND	Y1 . 4			A • C	B • D
5	OR.STK					A • C + B • D
6	RD.STK	R2 . 1			A • C + B • D	E
7	AND	R3 . 5			A • C + B • D	E • F
8	OR.STK					A • C + B • D + E • F
9	WRT	Y15 . 0				A • C + B • D + E • F

Table 8-8: Coding of the RD.STK Command

8.2.11 RD.NOT.STK Command

Format



Function

This command pushes the intermediate calculation onto the stack. It shifts each bit in the register one bit to the left and puts the inverse of the value of the signal with the specified address into the ST0 bit of the result history register.

Use

Use this command when the specified address is the B junction (⊣/⊢).

Example of RD.NOT.STK Command Usage

See Figure 8-8 and Table 8-9 for examples of how to use the RD.NOT.STK command.

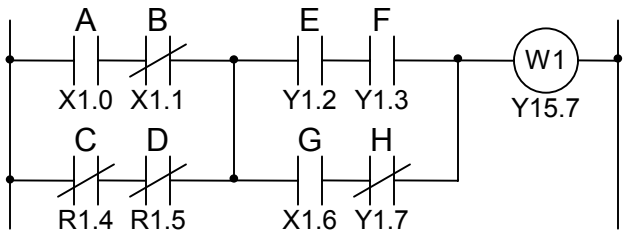


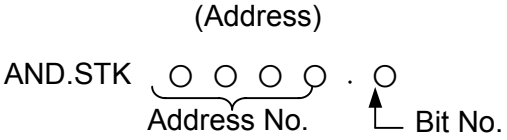
Figure 8-8: Ladder Diagram Example for the RD.NOT.STK Command

Coding Sheet				Result History Register		
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST2**	**ST1**	**ST0**
1	RD	X1 . 0	A			A
2	AND.NOT	X1 . 1	B			$A \cdot \overline{B}$
3	RD.NOT.STK	R1 . 4	C		$A \cdot \overline{B}$	$\overline{C}$
4	AND.NOT	R1 . 5	D		$A \cdot \overline{B}$	$\overline{C} \cdot \overline{D}$
5	OR.STK					$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$
6	RD.STK	Y1 . 2	E		$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$	E
7	AND	Y1 . 3	F		$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$	$E \cdot F$
8	RD.STK	X1 . 6	G	$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$	$E \cdot F$	G
9	AND.NOT	Y1 . 7	H	$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$	$E \cdot F$	$G \cdot \overline{H}$
10	OR.STK				$A \cdot \overline{B} + \overline{C} \cdot \overline{D}$	$E \cdot F + G \cdot \overline{H}$
11	AND.STK					$(A \cdot \overline{B} + \overline{C} \cdot \overline{D}) \cdot (E \cdot F + G \cdot \overline{H})$
12	WRT	Y15 . 7	W1 out			$(A \cdot \overline{B} + \overline{C} \cdot \overline{D}) \cdot (E \cdot F + G \cdot \overline{H})$

Table 8-9: Coding of the RD.NOT.STK Command

8.2.12 AND.STK Command

Format



Function

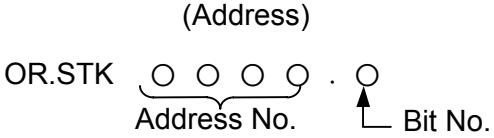
This command sets the logical product of ST0 and ST1 into ST1, and shifts the bits of the register one bit to the right to put the result into ST0.

Example of AND.STK Command Usage

See Figure 8-8 and Table 8-9 for examples on how to use the AND.STK command.

8.2.13 OR.STK Command

Format



Function

This command sets the logical sum of ST0 and ST1 into ST1, and shifts the contents of the register one byte to the right to take the result from ST0.

Examples of OR.STK Command Usage

See Figure 8-7 and Table 8-8, and Figure 8-8 and Table 8-9 for examples using the OR.STK command.

Note

In the example shown in Table 8-8, the results are the same even if the OR.STK in step number 5 is moved between steps number 7 and number 8. However, if you code with OR.STK and AND.STK repeatedly it is easier to make mistakes.

Chapter 9: PLC Functional Commands

9.1 Overview

When you are already dealing with a program complex enough to control an NC machine, creating a sequence program using only bit operation functions can be very difficult. For example, digital controls of a rotational system can be made simpler with these functional commands. So, for such programs, we have provided functional commands (also known as "machine commands"). The different types of functional commands and their descriptions are shown in Tables 9-1 and 9-2.

No.	Command		Description
	Ladder Format	Code Format	
1	TMR	TMR	Timer
2	TMRB	SUB 24	Static Timer
3	TMRC	SUB 54	Timer
4	DEC	DEC	Decode
5	DECB	SUB 25	Binary Decode
6	CTR	SUB 5	Counter
7	CTRC	SUB 55	Counter
8	ROT	SUB 6	Rotational Control
9	ROTB	SUB 26	Binary Rotational Control
10	COD	SUB 7	Code Transformation
11	CODB	SUB 27	Binary Code Transformation
12	MOVE	SUB 8	Masked Data Transfer
13	MOVOR	SUB 28	Bit-Wise Sum Data Transfer
14	COM	SUB 9	Common Line Control
15	COME	SUB 29	Common Line Control Termination
16	JMP	SUB 10	Jump
17	JMPE	SUB 30	Jump Termination
18	PARI	SUB 11	Parity Check
19	DCNV	SUB 14	Data Conversion
20	DCNVB	SUB 31	Extended Data Conversion
21	COMP	SUB 15	Comparison
22	COMPB	SUB 32	Binary Comparison

Table 9-1: Summary of Functional Commands (1 of 2)

No.	Command		Description
	Ladder Format	Code Format	
23	COIN	SUB 16	Equality Check
24	SFT	SUB 33	Shift Register
25	DSCH	SUB 17	Data Search
26	DSCHB	SUB 34	Binary Data Search
27	XMOV	SUB 18	Index Modify Data Transfer
28	XMOVB	SUB 35	Binary Index Modify Data Transfer
29	ADD	SUB 19	Addition
30	ADDB	SUB 36	Binary Addition
31	SUB	SUB 20	Subtraction
32	SUBB	SUB 37	Binary Subtraction
33	MUL	SUB 21	Multiplication
34	MULB	SUB 38	Binary Multiplication
35	DIV	SUB 22	Division
36	DIVB	SUB 39	Binary Division
37	NUME	SUB 23	Constant
38	NUMEB	SUB 40	Binary Constant

Table 9-2: Summary of Functional Commands (2 of 2)

The command format and the general usage are described at the beginning of each functional command description. Important information, such as the functional command specifications, is included in this chapter, so you should review it carefully for each functional command.

9.1.1 Functional Command Format

Functional commands cannot be described by relay symbols, so they are written in a different format (see Figure 9-1). This format is composed of control values, commands, parameters, address W1, and addresses R9000~R9005 (the functional command register). The functional commands also use the result history register.

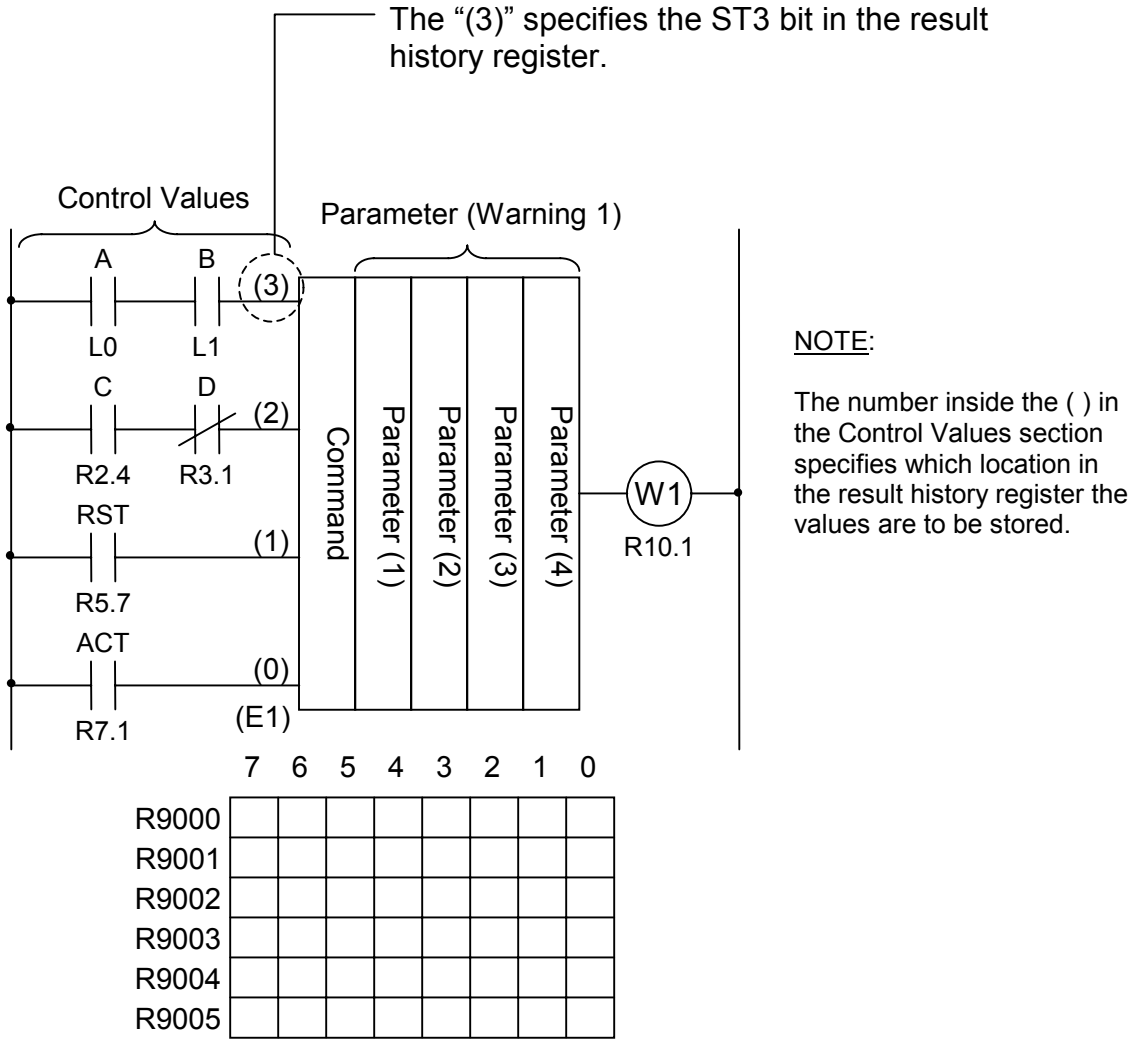


Figure 9-1: Functional Command Format – Ladder Diagram and Functional Command Register

Coding Sheet				Result History Register			
Step No.	Command	Address No. Bit No.	Description	ST3	ST2	ST1	ST0
1	RD	R1 . 0	A				A
2	AND	R1 . 1	B				A • B
3	RD.STK	R2 . 4	C			A • B	C
4	AND.NOT	R3 . 1	D			A • B	C • $\overline{D}$
5	RD.STK	R5 . 7	RST		A • B	C • $\overline{D}$	RST
6	RD.STK	R7 . 1	ACT	A • B	C • $\overline{D}$	RST	ACT
7	SUB	OO	Command	A • B	C • $\overline{D}$	RST	ACT
8		OOOO	Param 1	A • B	C • $\overline{D}$	RST	ACT
9		OOOO	Param 2	A • B	C • $\overline{D}$	RST	ACT
10		OOOO	Param 3	A • B	C • $\overline{D}$	RST	ACT
11		OOOO	Param 4	A • B	C • $\overline{D}$	RST	ACT
12	WRT	R10 . 1	W1 Out	A • B	C • $\overline{D}$	RST	W1

Table 9-3: Functional Command Format – Coding

9.1.2 Control Values

Depending on the functional command, the number of control values and their significances differ. Since the control values have specified locations in the Result History Register to be put into, as shown in Table 9-3, there is a unique ordering of the instructions. You cannot change the instruction order.

⚠ CAUTION

Functional commands that have RST in their control inputs are all RST prioritized. Therefore, even if ACT=0, the command executes the RST operation if RST=1.

9.1.3 Command

The different kinds of commands are as shown in Tables 9-1 and 9-2.

9.1.4 Parameters

Functional commands differ from basic commands in that they use numbers. These numbers, such as basic data or an address for data, go into the system as parameters. The number of parameters and their roles differ for each functional command.

Parameters are indicated as "(PRM)" in the coding sheets.

9.1.5 W1

When the functional command results in an output of one bit ("1" or "0"), then the result can be stored in the address specified by W1. You can arbitrarily choose the address. The logic behind W1 depends on the functional command; some functional commands do not output a value.

9.1.6 Operation Data – Binary Coded Decimal or Binary Format

The data used in functional commands are either in BCD (Binary Coded Decimal) format or binary format. The usual PLC sequence program uses BCD format for numerical data, but for this PLC system, we recommend that you use binary format for the following reasons:

1) For ServoWorks, the format of the data transferred (codes M, S, T and B) between the ServoWorks Motion Engine ↔ PLC Engine is binary.

2) Since the CPU processes all numeric data in binary format, if the data is already in binary format, it is not necessary to convert it, and it will increase the processing speed.

3) Binary format allows for a larger range of numbers. Numbers previously out of range become valid, and the range of functional commands also increases. A binary format can use different numbers of bytes: 1 byte (-128 ~ +127), 2 bytes (-32,768 ~ +32,767), or 4 bytes (-99,999,999 ~ +99,999,999).

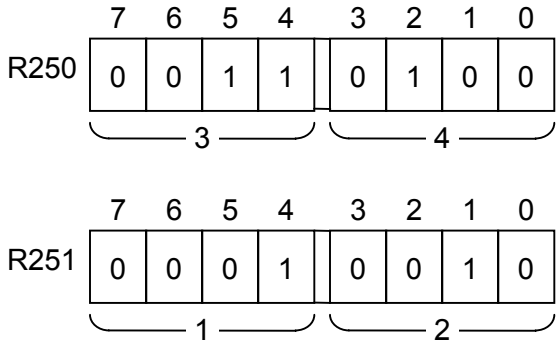
4) When importing different types of numeric data, or when displaying numerical data, there is no inconvenience to using the BCD format. Although the data stored in the internal memory is in binary digits, the number itself is encoded in decimal format. Therefore, conversion back to decimal (for transfers and display) is very simple. The only thing to be cautious of is when the program looks at memory contents. See *9.1.7 Numerical Data Examples*.

For these reasons, all functional commands can and usually will use binary data.

9.1.7 Numerical Data Examples

9.1.7.1 BCD Format Data

The basic data that uses the BCD format is either 1 byte (0 ~ 99) or 2 bytes (0 ~ 9999) long. A 4-digit BCD data goes into 2 successive bytes, as in the following example, which shows when BCD data 1234 gets stored inside addresses R250 and R251.



When specifying this data in a functional command, use the earlier address R200.

(NOTE: The earlier address gets the lower two digits.)

Figure 9-2: Example of BCD Format Data

9.1.7.2 Binary Format Data

The data that uses the binary format is either 1 byte (-128 ~ +127), 2 bytes (-32,768 ~ +32,767), or 4 bytes (-99,999,999 ~ +99,999,999) long, and it is stored into addresses R200, R201, R202 and R203 as shown in Figure 9-3:

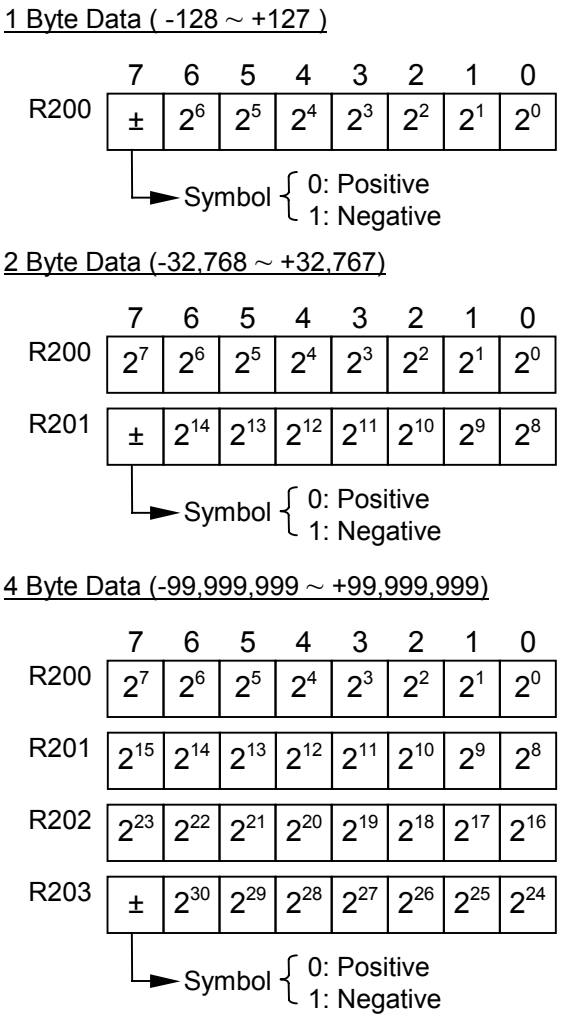


Figure 9-3: Memory Storage of Binary Format Data

Functional commands use the earlier address R200 to specify the data. Furthermore, negative numbers are represented using Two's Complement notation.

To represent a negative number with Two's Complement notation:

1) Write the binary format of the absolute value of the number.

2) Write the One's Complement of that number. That is, take the number from Step #1 and convert the ones into zeros and the zeros into ones.

3) Take the One's Complement number from Step #2, and add one (1) to the result.

An example of Two's Complement notation follows:

Example of the number -17 represented in Two's Complement notation:

	0	0	0	1	0	0	0	1	(Binary format of +17)
	1	1	1	0	1	1	1	0	(One's Complement of +17)
+	0	0	0	0	0	0	0	1	(Add 1)
	1	1	1	0	1	1	1	1	(Two's Complement Form of -17)

Several examples of data represented in binary format follow. The negative numbers are represented with Two's Complement notation.

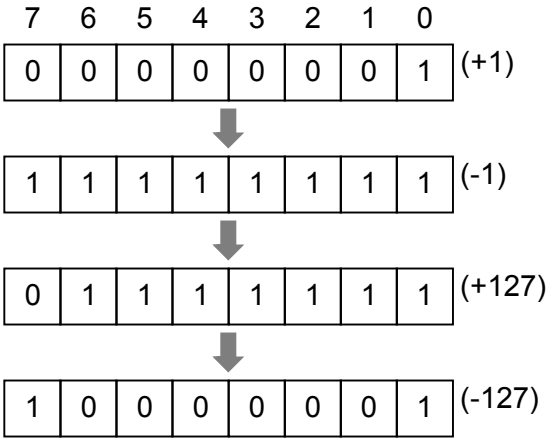


Figure 9-4: Examples of Binary Format Data for 1 Byte Data

9.1.8 Addresses for the Numerical Data Handled by Functional Commands

When the numerical data handled by a functional command is 2 bytes or 4 bytes long, we recommend using an even address. By using an address that is even, you will slightly decrease the execution time of the functional command.

The parameters of the functional commands with this feature, mainly on functional commands that use binary data, are marked with an * on the parameter section of the functional command format description. For an address to be even in an internal relay, the numbers following the R are even, and for an address to be even in a data table, the numbers following D are even.

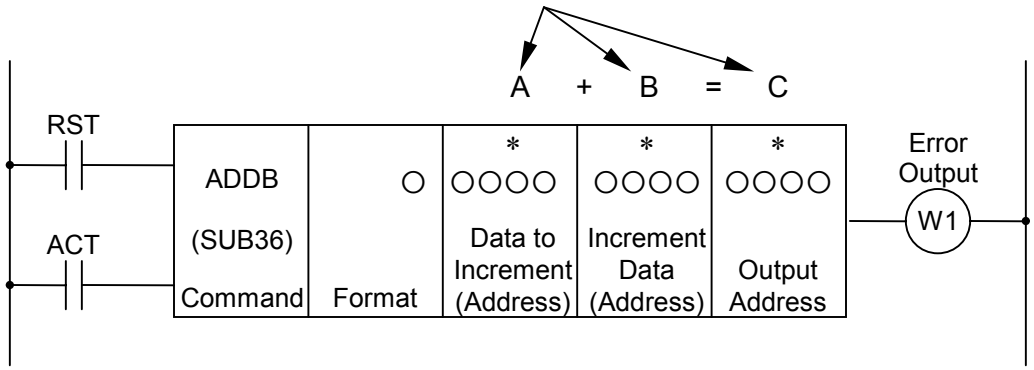


Figure 9-5: Addresses of Numeric Data

9.1.9 Functional Command Register (R9000 ~ R9005)

This register holds the results of the functional commands. This register is shared by all commands, so you must read the data immediately after the functional command finishes executing; otherwise, it will be overwritten by the next command.

Furthermore, you cannot transfer the information in this register between the different levels within the sequence program. For example, you cannot execute a subtraction command (SUBB) in a level 1 program and then try to read this result within a level 2 program by looking at the address R9000.

The register is shared among programs in the same level, and it is stored until right before the next command executes. The sequence program is able to read the value, but you cannot write to it directly.

	7	6	5	4	3	2	1	0
R9000								
R9001								
R9002								
R9003								
R9004								
R9005								

Figure 9-6: Functional Command Register

This register is a 6-byte register R9000 ~ R9005, and data can be entered 1 bit or 1 byte at a time. In order to read the 1st bit of R9000, you use the command RD R9000.1.

9.2 Descriptions of Functional Commands

9.2.1 TMR (Timer)

Function

This timer is an on-delay timer.

Format

Figure 9-7 shows the format for describing the command, and Table 9-4 shows the coding format.

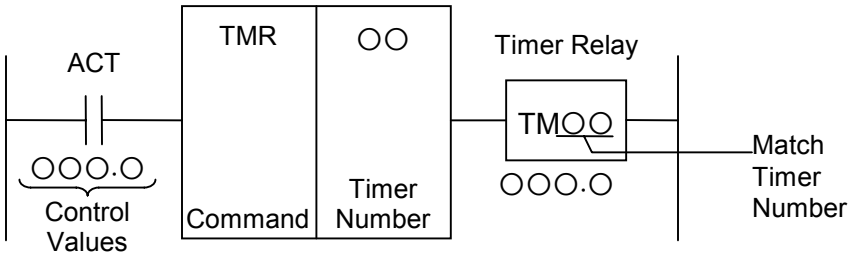


Figure 9-7: Format for the TMR Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○○	○	ACT
2	TMR	○○		
3	WRT	○○○○	○	TM○○

Table 9-4: Coding Format of the TMR Command

Control Values

Action Command (ACT)

ACT = 0: Turns off (makes "0") the Timer Relay (TM○○).
ACT = 1: Timer turns on.

Timer Relay (TM○○)

As shown in Figure 9-8, when ACT = 1 for a specified time period, the timer relay turns on. You can freely choose the timer relay address.

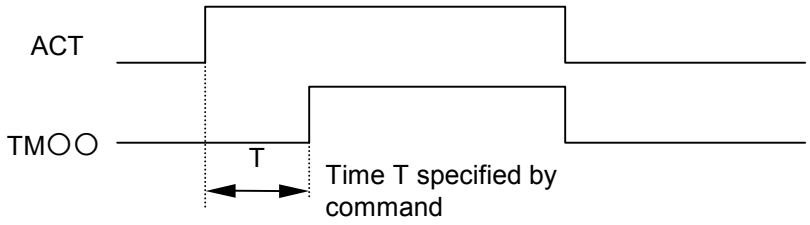


Figure 9-8: Timer Behavior for the TMR Command

Parameters

Timer Number

The time on the timer is set in the PLC Control Console application and is in units of ms. Timers 1 ~ 8 are checked every 48 ms, and timers 9 ~ 100 are checked every 8 ms. Therefore, on timers 1 ~ 8, the time intervals are evaluated on 48 ms intervals, and remaining values less than 48 are disregarded. The timers 9 ~ 100 are set with times that are integer multiples of 8, and the remainders are disregarded. For example, if you set the timer to 38 ms, 38 = 8 × 4 + 6, the remainder 6 is disregarded and the timer is set as 32 ms.

Accuracy of the Timer

Timers 1 ~ 8 have times within the range 48 ms ~ 99,999,999 ms and are spread out every 0 ~ +48 ms. Timers 9 ~ 40 have times within the range 8 ms ~ 99,999,999 ms and are spread out every 0 ~ +8 ms.

The spread of the timer times only includes the timing error that occurs when executing the timer command. Other delays are not included. For example, if the timer command is used in sequence section 2, the time delay observed after startup and before the sequence is evaluated (worst case: 1 complete cycle time of sequence section 2) is not included.

9.2.2 TMRB (Fixed Timer)

Function

This timer is an on-delay timer where the time is fixed. Because the timer described by *Section 10.2.1: TMR* stores the time in memory, it is a dynamic timer that can change the specified time through the PLC control screen when necessary. The fixed timer permanently writes the time when the program is written, so once the time is set, it cannot be changed unless the sequence program itself is changed.

Format

Figure 9-9 shows the format for describing the command.

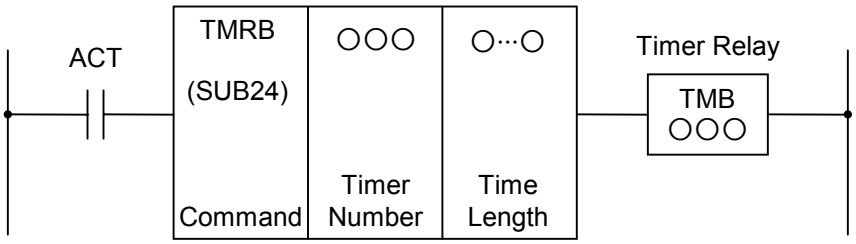


Figure 9-9: Format for the TMRB Command

Control Values

Action Command (ACT)

ACT = 0: Turns off (makes "0") the timer relay (TMB○○○).
ACT = 1: Timer turns on.

Timer Relay (TMB○○○)

As shown in Figure 9-10, when ACT = 1 for a specified time period, the timer relay turns on. You can freely choose the timer relay address.

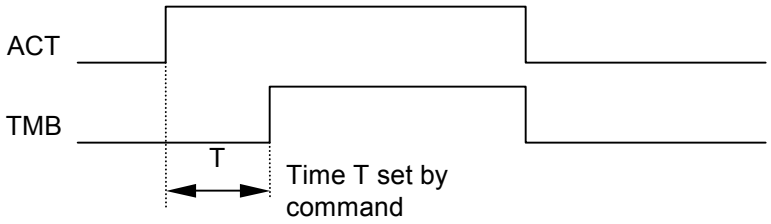


Figure 9-10: Timer Behavior for the TMRB Command

Parameters

1) Timer Number
   A number (1 ~ 100) that identifies the fixed timer.

2) Time Length
This fixed timer is processed every 8 ms. Therefore, the time specified should be an integral multiple of 8 ms, as any remainders are disregarded. For example, if the timer value is specified to 38 ms, 38 = 8 × 4 + 6 and the remainder of 6 is disregarded, resulting in a final value of 32 ms on the timer. The time ranges from 8 ~ 262, 136 ms.

Accuracy of the Timer

The time delay on a fixed timer is between 0 ~ +8 ms. This delay only includes the delay caused by the execution of the timer command. Other delays, such as execution of the sequence program (1 cycle of level 2), are not included.

9.2.3 TMRC (Timer)

Function

This timer is an on-delay timer.

Format

Figure 9-11 shows the format for describing the command, and Table 9-5 shows the coding format.

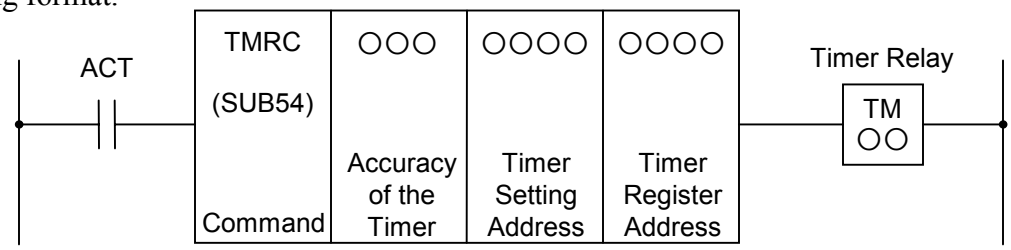


Figure 9-11: Format for the TMRC Command

Step No.	Command	Address No. Bit No.	Description
1	RD	OOOO . O	
2	SUB	54	TMRC Command
3	(PRM)	O	Accuracy of the Timer
4	(PRM)	OOO	Timer Setting Address
5	(PRM)	OOOO	Timer Register Address
6	WRT	OOOO . O	TMOO

Table 9-5: Coding Format of the TMRC Command

Control Values

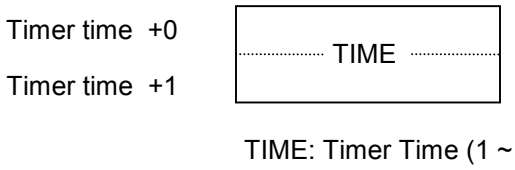
Action Command (ACT)

ACT = 0: Turns off (makes "0") the timer relay (TMOO).
ACT = 1: Timer turns on.

Parameters

1) Accuracy of the Timer
   The timer's accuracy can be specified as one of the following two choices:
   0: 8 ms
   1: 48 ms

2) Timer Setting Address
This setting specifies the leading address of the timer's time setting. The 2 bytes starting with this address will contain the timer setting time. Usually, a D domain address is used.



TIME: Timer Time (1 ~ 32,767)

Figure 9-12: TMRC Address of the Time Set of the Timer

The time setting of the timer is in 8 ms or 48 ms intervals, and this value is stored in binary format. The timer setting time becomes as follows:
8 ms: 8 ~ 262,136 ms
48 ms: 48 ~ 1,572,816 ms

3) Timer Register Address
This setting specifies the leading address of the timer register address. The 4 bytes starting with this address will contain the timer register value. Usually, an R domain address is used. The PC uses this region, so the sequence program should not use it.

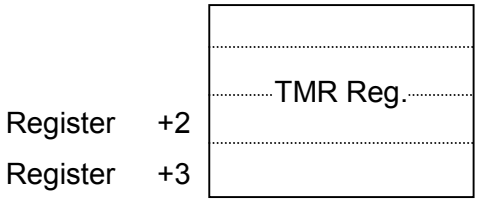


Figure 9-13: Timer Register Address for the TMRC Command

Timer Relay (TM○○)

As shown in Figure 9-14, after ACT = 1, the timer turns on, and after the specified time, the timer relay turns on.

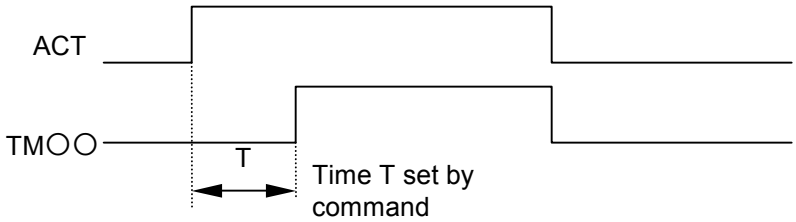


Figure 9-14: Timer Behavior for the TMRC Command

9.2.4 DEC (Decoding)

Function

This command compares a two-line BCD coded signal with a specified BCD coded signal. This command outputs "1" if they match, "0" otherwise. This command is mainly used in the decoding of M functions and T functions.

Format

Figure 9-15 shows the format for describing the command, and Table 9-6 shows the coding format.

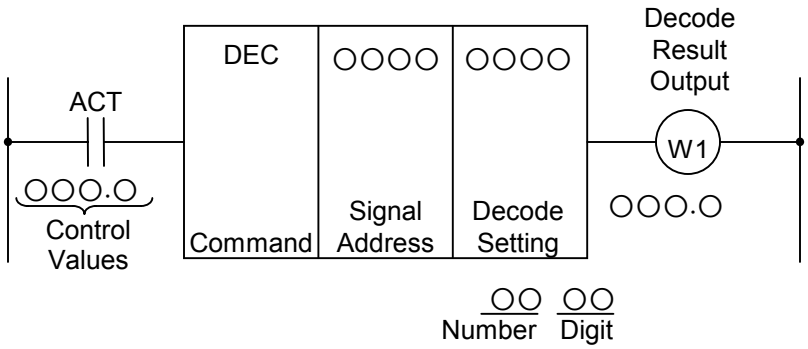


Figure 9-15: Format for the DEC Command

Decoding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOO	O	ACT
2	DEC	OOOO		
3	(PRM)	OOOO		
4	WRT	OOO	O	W1, Decode Result Output

Table 9-6: Coding Format of the DEC Command

Control Values

Action Command (ACT)

ACT = 0: Turns off the output data (all 8) of the decode result output (W1).

ACT = 1: Starts the decoding, and outputs the decoded result to the decode output address.

Parameters

1) Signal Address

The initial address of the two-line BCD coded signal.

2) Decode Setting
There are two meanings to a decode setting: the number and the digit.

Decode setting: ○○,○○
Number Digit

a) Number Setting: The two-word number that is going to be used to decode.

b) Digit Setting: From the two digits in the decimal representation:
01: The top digit is set to 0 and only the bottom digit is decoded.
10: The bottom digit is set as 0 and the top digit is decoded.
11: Both digits are decoded.

Decode Result Output (W1)

W1 = 0: The value of the signal does not match the given number.
W1 = 1: The value of the signal matches the given number.
You can freely choose the address of W1.

Example of DEC Command Usage

See Figure 9-16 and Table 9-7 for an example of DEC command usage.

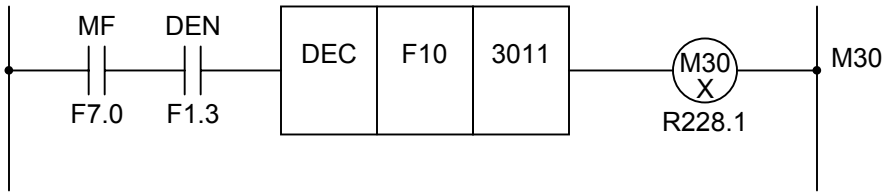


Figure 9-16: Ladder Diagram Example Using the DEC Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	F7	. 0	
2	AND	F1	. 3	
3	DEC	F10		
4	(PRM)	3011		
5	WRT	R228	. 1	M30X

Table 9-7: Coding Example of the DEC Command

9.2.5 DECB (Binary Decoding Processing)

Function

This command decodes a 1-, 2-, or 4-byte binary format code data. If the code data matches one of the pre-specified 8 numbers, it outputs a "1" to the output bit corresponding to the number it matched. If it does not match, it outputs "0". One of the parameters is the address of the leading number (out of 8). This command is mainly used in decoding M functions and T functions.

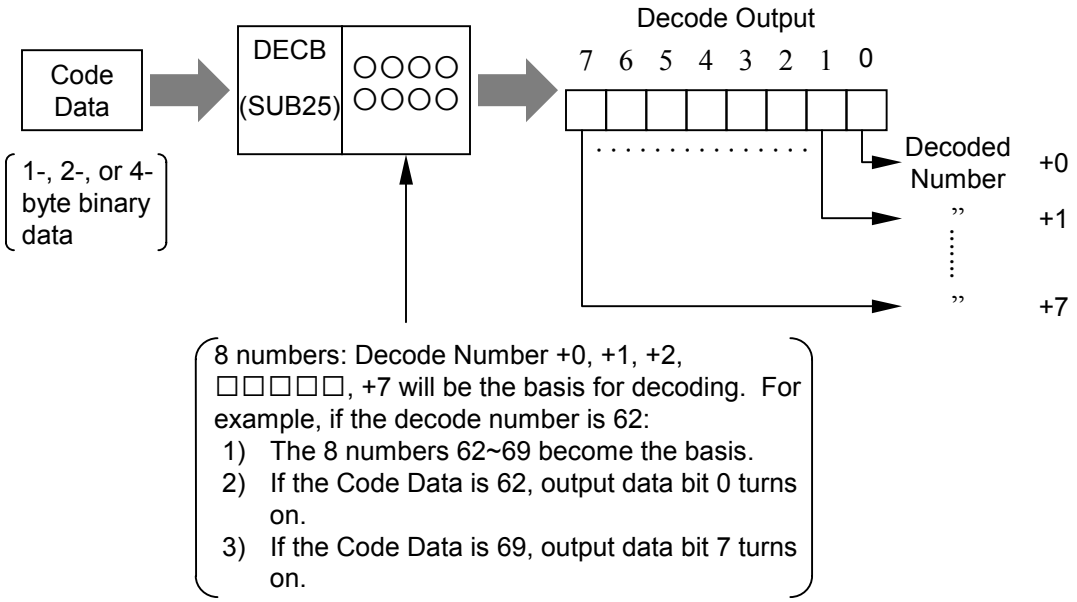


Figure 9-17: Function for the DECB Command

Format

Figure 9-18 shows the format for describing the command.

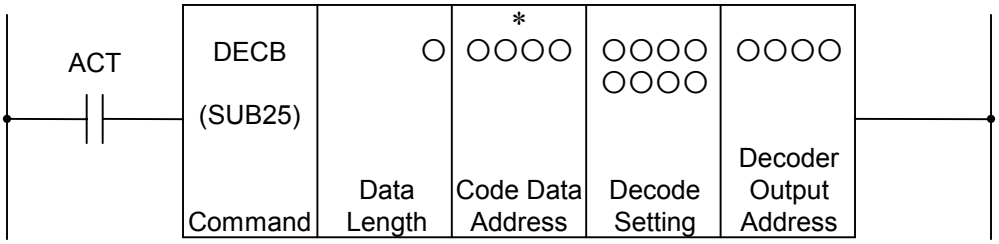


Figure 9-18: Format for the DECB Command

Control Values

Action Command (ACT)

ACT = 0: Turns off the output data (all 8) of the decoding result (W1).

ACT = 1: Starts the decoding, and outputs the decoded result to the decode output address.

Parameters

1) Data Length
The byte length of the code data.
When 1: Code data is 1-byte binary data.
When 2: Code data is 2-byte binary data.
When 4: Code data is 4-byte binary data.

2) Code Data Address
The address of the code data.

3) Decode Setting
The lead number of the eight numbers that are used to decode.

4) Decoder Output Address
The address that receives the output of the decoding result. Output requires one byte.

9.2.6 CTR (COUNTER)

Function

The most common use of the counter is to carry out addition. However, in the context of machine control, the counter is used in various ways. The numbers in the counter (preset value, addition value) can be specified to be in either BCD format or binary format, depending on the system parameters of the PLC.

The counter has the following functionalities, which you can use appropriately depending on the situation:

1) Preset Counter
   This counter starts at a value, then when the counter exceeds the specified value, it signals this event in the output. The preset value is set on the PLC control screen. You can also specify the preset value within the sequence program.

2) Ring Counter
   After passing the specified time, this counter continues and eventually loops back to the starting number.

3) Up/Down Counter
   This counter is a reversible counter; it can count both upwards and downwards.

4) Starting Value
   Specifies the starting value of a counter as either "0" or "1."

By combining the above functions, you can create not just an addition counter, but also a two-way ring counter like the one shown in Figure 9-19. If you use the counter in this way, you can determine the location of a rotation object.

.

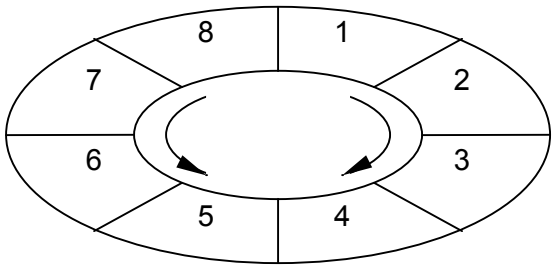


Figure 9-19: Ring Counter Created Using the CTR Command

Format

Figure 9-20 shows the format for describing the command, and Table 9-8 shows the coding format.

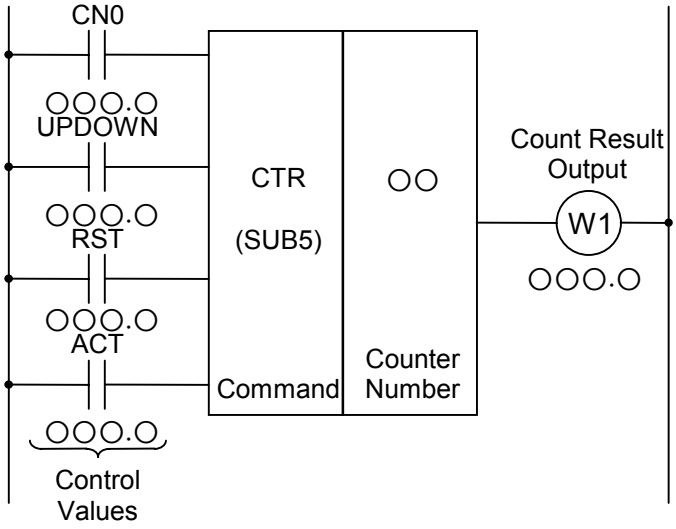


Figure 9-20: Format for the CTR Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○	. ○	CN0
2	RD.STK	○○○	. ○	UPDOWN
3	RD.STK	○○○	. ○	RST
4	RD.STK	○○○	. ○	ACT
5	SUB	5		CTR Command
6	(PRM)	○○		Counter Number
7	WRT	○○○	. ○	W1, Count Result Output

Result History Register

ST3	ST2	ST1	ST0
			CN0
		CN0	UPDOWN
	CN0	UPDOWN	RST
CN0	UPDOWN	RST	ACT
CN0	UPDOWN	RST	ACT
CN0	UPDOWN	RST	ACT
CN0	UPDOWN	RST	W1

Table 9-8: Coding Format of the CTR Command

Control Values

1) Starting Value (CN0)
   CN0 = 0: The counter starts from 0. (0, 1, 2, 3, ……., n_0)
   CN0 = 1: The counter starts from 1. (1, 2, 3, ……, n_0)

2) Up/Down (UPDOWN)
   UPDOWN = 0: Specifies an Up-Counter. The starting value is chosen by CN0.
   UPDOWN = 1: Specifies a Down-Counter. A preset value becomes the starting value.

3) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. On a reset, W1 becomes 0 and the counter value is restored to the starting value.

⚠ CAUTION

You should set RST as "0" normally and change it to "1" only when resetting is necessary. If not, the memory may not be read properly.

4) Action Command (ACT)
The counter detects a rising edge of the ACT signal, and counts at that point.

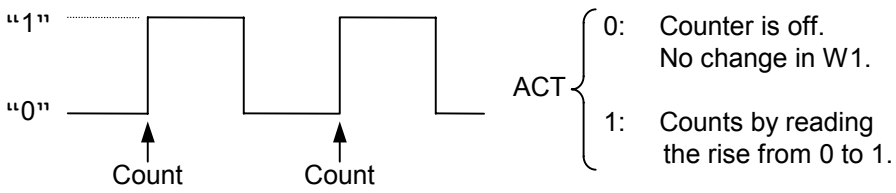


Figure 9-21: Count Signal (Action Command) for the CTR Command

Parameters

Counter Number
There are 20 counters, each with 2 available bytes to use for the Preset Value and the Addition Value, so the counter numbers that you can use are from 1~20.

Count Result Output (W1)

W1 = 0: When not counted until the preset value.
W1 = 1: When counted until the preset value.
You can assign the address of W1 arbitrarily.

Examples of Counter Usage

Counter Example #1: A Preset Counter (See Figure 9-22.)

The preset counter counts, and when it reaches the specified number it outputs a signal.

- L1 is a circuit to create logic " l."
- Since the count range is 0~9999, CN0 needs to be 0, so the B junction of signal L is used.
- To get an Up-Counter, a B junction of signal L1 is used to make UPDOWN = 0.
- The input signal from the machine side, CRST · M, is used as a counter reset signal.
- The counter signal is M30X, a decoded NC output M code. The B junction of the CUP is in series with the M30X so that, after the counter reaches the designated value, it does not count any more (unless it receives a reset signal).

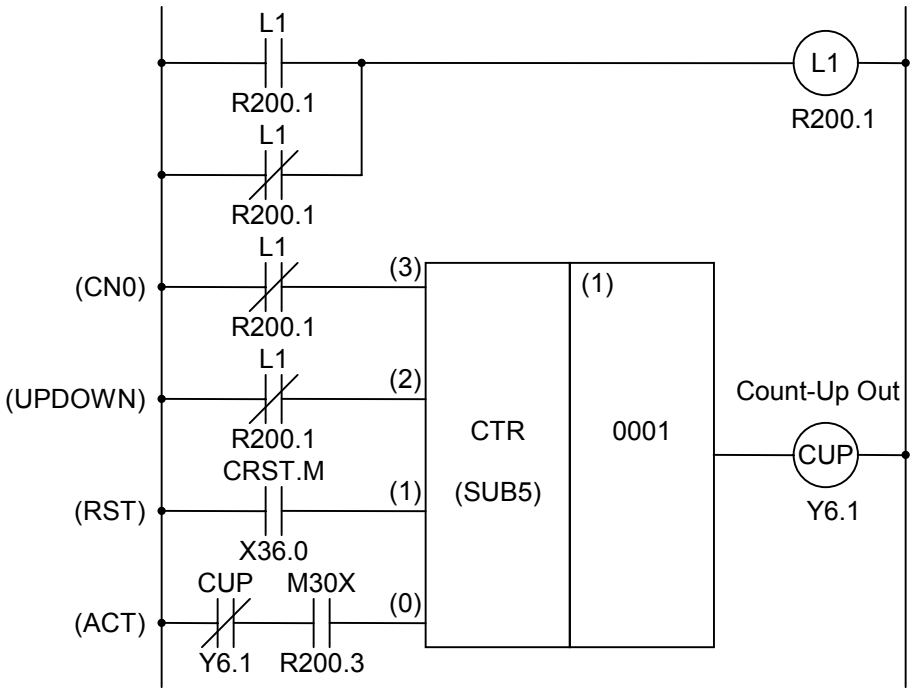


Figure 9-22: Ladder Diagram For Counter Example #1

Counter Example #2: Using a Counter In Order To Record The Location Of The Rotational Object (See Figures 9-23 and 9-24.)

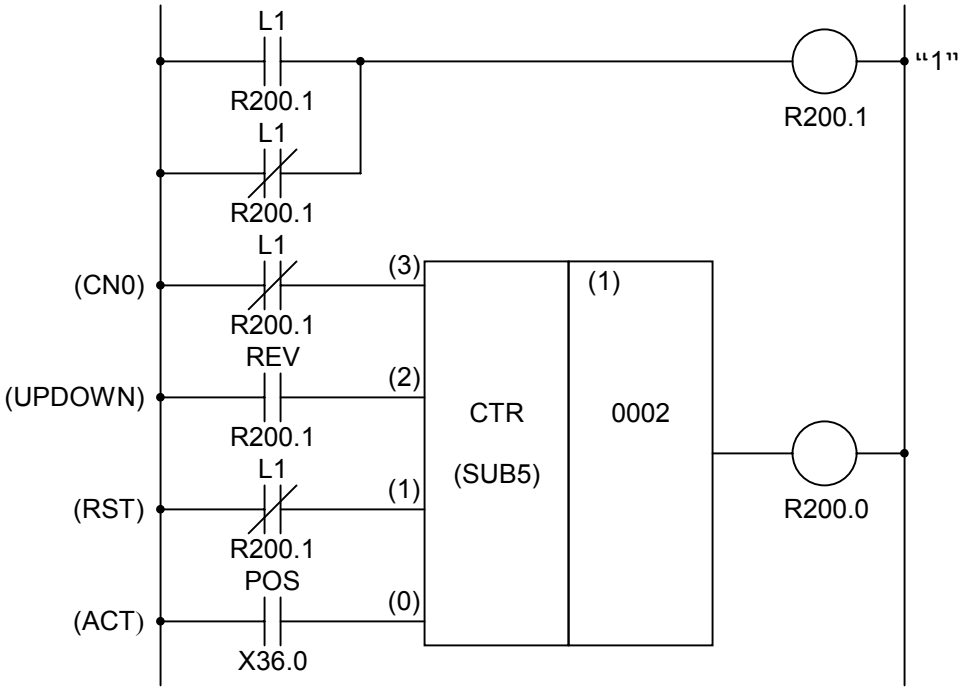


Figure 9-23: Ladder Diagram For Counter Example #2

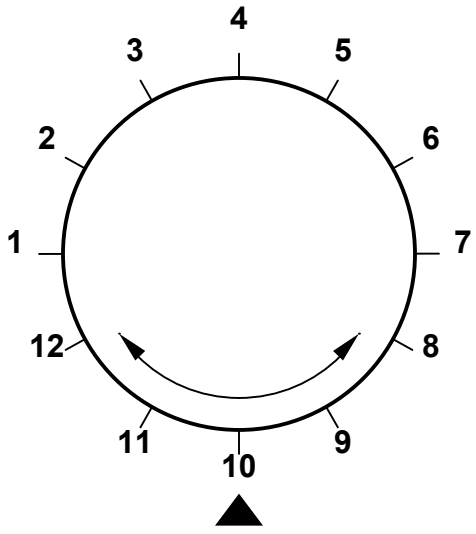


Figure 9-24: Division of a Rotational Body for Counter Example #2

Figure 9-23 is the ladder diagram that stores the location of the rotational body in Figure 9-24.

Counter Example #2 (con't)

1) CONTROL VALUES

   a) Counter Starting Number: If you think about the rotational body with twelve angles in Figure 9-24, the count starting number is 1. With that in mind, in order to make CN0 = 1, use the A junction of the signal L1.

   b) Specifying the Up-Down Signal: REV is a signal that changes with the rotational direction over time; REV changes into "0" when turning in the right direction, and "1" when turning in the reverse direction. So, when the rotation is in the right direction, the counter works as an up-counter, and when the rotation is in the reverse direction, it works as a down-counter.

   c) Reset: In the example, the W1 is not used. Therefore, RST = 0 all the time, and the B junction of the signal L1 is used.

   d) Count Signal: The count signal POS is a signal that alternates between ON-OFF 12 times when the rotational body turns once.

2) COUNTER NUMBER AND W1

   In this example, counter 2 was used. Although the result of W1 is not used elsewhere, you must designate an address for W1.

3) ACTION

   a) Specifying the Preset Value: The rotational body controlled here has 12 edges (Figure 9-24), so you must set the preset value of the counter as 12. To set the preset value to 12, the PLC control screen is used.

   b) Specifying the Current Value: You must first synchronize the rotational body location with the current counter value. This value is set in the PLC control screen. After this initialization is done once, the counter will correctly represent the current location.

   c) After the above a) and b) are done, then every time the rotational body rotates, the POS turns ON and OFF and the counter 2 counts the number of rotations. The counter counts in the following way:

   - When the rotation is in the forward direction:
     1, 2, 3, · · · · · · · · · · · , 11, 12, 1, 2, · · · · · · · · · · ·
   - When the rotation is in the reverse direction:
     1, 12, 11, · · · · · · · · · · · , 3, 2, 1, 12, · · · · · · · · · · ·

9.2.7 CTRC (Counter)

Function

This counter uses numeric data in binary format. The following types can be used according to the situation:

1) Preset Counter
   This counter is given a number to count to, previously specified, and when the counter exceeds this value, the counter outputs a signal.

2) Ring Counter
   After this counter reaches the preset value, it cycles to the starting value and starts again.

3) Up/Down Counter
   This counter is a reversible counter; it can count both upwards and downwards.

4) Starting Value
   Specifies the starting value of a counter as either "0" or "1."

Format

Figure 9-25 shows the format for describing the command, and Table 9-9 shows the coding format.

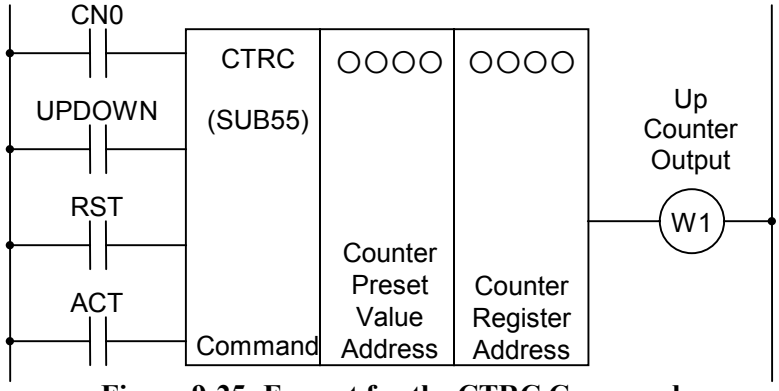


Figure 9-25: Format for the CTRC Command

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○○	. ○	CN0
2	RD.STK	○○○○	. ○	UPDOWN
3	RD.STK	○○○○	. ○	RST
4	RD.STK	○○○○	. ○	ACT
5	SUB	55		CTRC Command
6	(PRM)	○○○○		Count Preset Value Address
7	(PRM)	○○○○		Counter Register Address
8	WRT	○○○○	. ○	W1, Up Counter Output

Table 9-9: Coding Format of the CTRC Command

Control Values

1) Starting Value (CN0)
   CN0 = 0: The counter starts from 0. (0, 1, 2, 3, ..., n_0.)
   CN0 = 1: The counter starts from 1. (1, 2, 3, ..., n_0.)

2) Up/Down (UPDOWN)
   UPDOWN = 0: Up Counter. The starting values are:
   0 when CN0 = 0
   1 when CN0 = 1
   UPDOWN = 1: Down Counter. A preset value becomes the starting value.

3) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. On a reset, W1 becomes 0 and the counter value is restored to the starting value.

4) Action Command (ACT)
   The counter detects a rising edge of the ACT signal, and counts at that point.

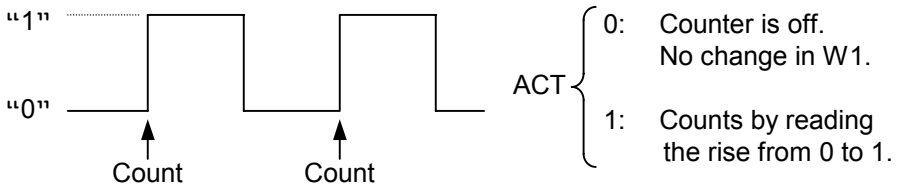


Figure 9-26: Count Signal (Action Command) for the CTRC Command

Parameters

1) Counter Preset Value Address
This parameter is the leading address of the counter preset value in memory. The two bytes starting from the lead address store the counter preset value. Usually, an address in the D range is used.

Figure 9-27: Address of the Counter Preset Value for the CTRC Command

The counter preset value is given in binary format, so you can specify values 0 ~ 32,767.

2) Counter Register Address
This parameter is the leading address for the counter register. The four bytes starting from the lead address store the values for the counter register. Usually, an address in the D range is used.

⚠ CAUTION

If an address in the R region is used for a counter register, the value is reset to 0 when the power is turned on.

Up Counter Output (W1)

W1 = 0: When the counter does not exceed the preset value.
W1 = 1: When the counter exceeds the preset value.
The address of W1 can be assigned arbitrarily.

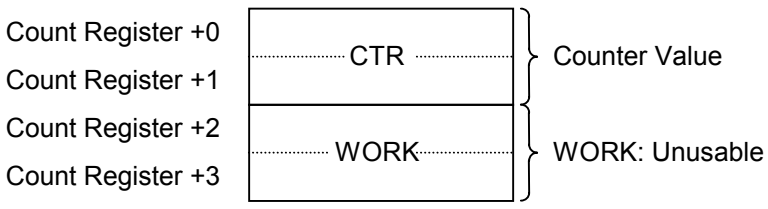


Figure 9-28: Address of the Up Counter Output for the CTRC Command

9.2.8 ROT (Rotational Control)

Function

This command is used for controlling a rotational system like a rotational knife sharpener, ATC, or a rotating table with the following functions:

1) Provides a shortcut for distinguishing the direction of rotation.

2) Calculates the number of steps between the current location and the destination.

3) Calculates the location or the number of steps to the location one before the destination.

Format

Figure 9-29 shows the format for describing the command, and Table 9-10 shows the coding format.

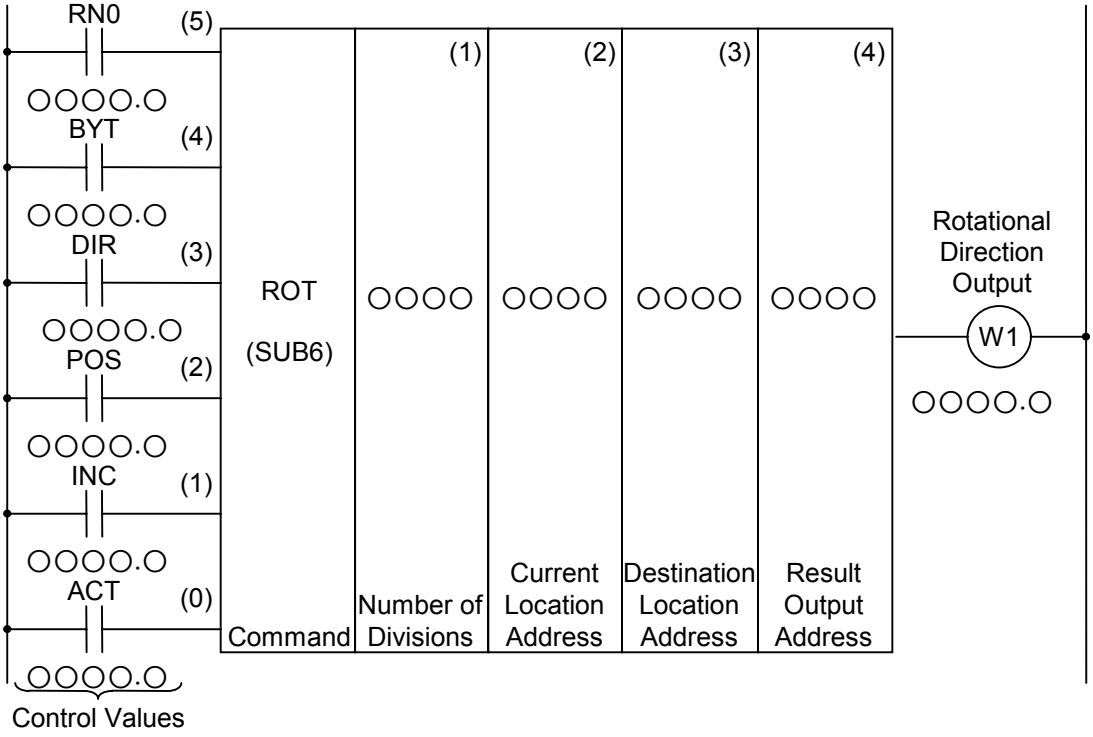


Figure 9-29: Format for the ROT Command

Coding Sheet **Result History Register**

Step No.	Command	Address No.	Bit No.	Description	ST5	ST4	ST3	ST2	ST1	ST0
1	RD	OOOO	O	RN0						RN0
2	RD.STK	OOOO	O	BYT					RN0	BYT
3	RD.STK	OOOO	O	DIR				RN0	BYT	DIR
4	RD.STK	OOOO	O	POS			RN0	BYT	DIR	POS
5	RD.STK	OOOO	O	INC		RN0	BYT	DIR	POS	INC
6	RD.STK	OOOO	O	ACT	RN0	BYT	DIR	POS	INC	ACT
7	SUB	6		ROT Command	RN0	BYT	DIR	POS	INC	ACT
8	(PRM)	OOOO		Number of Divisions	RN0	BYT	DIR	POS	INC	ACT
9	(PRM)	OOOO		Current Location Address	RN0	BYT	DIR	POS	INC	ACT
10	(PRM)	OOOO		Destination Location Address	RN0	BYT	DIR	POS	INC	ACT
11	(PRM)	OOOO		Result Output Address	RN0	BYT	DIR	POS	INC	ACT
12	WRT	OOO	O	W1, Rotational Direction Output	RN0	BYT	DIR	POS	INC	W1

Table 9-10: Coding Format of the ROT Command

Control Values

1) Rotational Body Starting Number (RN0)
   RN0 = 0: The rotational body location number starts from 0.
   RN0 = 1: The rotational body location number starts from 1.

2) Data Size (BYT)
   BYT = 0: The data is 2-digit BCD.
   BYT = 1: The data is 4-digit BCD.

3) Whether Or Not To Distinguish the Direction of Rotation (DIR)
   DIR = 0: No distinction of direction of rotation; it will always be FOR direction.
   DIR = 1: Distinguishes the direction of rotation. See the *Rotational Direction Output (W1)* section for more information.

4) Type of Calculation (POS)
   POS = 0: Calculates to the destination.
   POS = 1: Calculates to the location one before the destination.

5) Calculate Location or the Number of Steps (INC)
   INC = 0: Calculates the number of a location. (To calculate the location one before the destination, specify INC = 0 and POS = 1.)
   INC = 1: Calculates the number of steps. (To calculate the difference between the current location and the destination, specify INC = 1 and POS = 0.)

6) Action Command (ACT)
   ACT = 0: No execution of the ROT command. No change in W1.
   ACT = 1: Execution of the ROT command.
   Usually ACT = 0, and when a calculation result is necessary, ACT = 1.

Parameters

1) Number of Divisions
   Specifies the rotational body's calculation number.

2) Current Location Address
   This is the address that contains the current location of the machine.

3) Destination Location Address
   This is the address where the destination location (or the command location) is stored. For example, this is the output of the NC, stored among the T-code address region.

4) Result Output Address
   This is the address where the result of the unit is stored. The result can be the number of steps to turn to the destination, to one before the destination, or the location of the step one before the destination. When using this value, make sure that ACT = 1.

Rotational Direction Output (W1)

When using the shortcut controls, the direction of rotation is output to W1.

W1 = 0: Specifies the FOR direction.
W1 = 1: Specifies the REV direction.

See Figure 9-30 for definitions for FOR (forward) and REV (reverse). When looking from a specific point, the direction that increases the location number is FOR, the direction that decreases the location number is REV. The address of W1 can be assigned arbitrarily. In order to use the results from W1 you must make sure that ACT = 1.

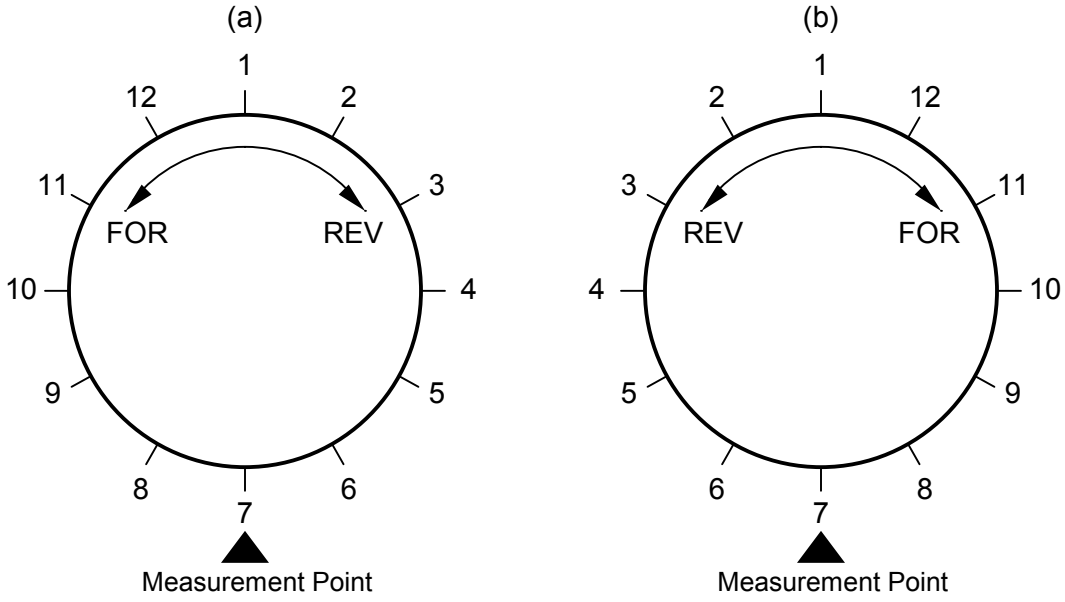


Figure 9-30: Rotation Direction Rule – 12-Division Example

9.2.9 ROTB (Binary Rotational Control)

Function

This command is used for controlling a rotational body such as ATC and a rotational table. In contrast to the ROT command (Section 9.2.8) where the parameter of the rotational calculation number is fixed, this command specifies an address to the data, so the data can change while execution occurs. Also, all data used is in binary format. The rest of the specifications are the same as the ROT command.

Format

Figure 9-31 shows the format for describing the command.

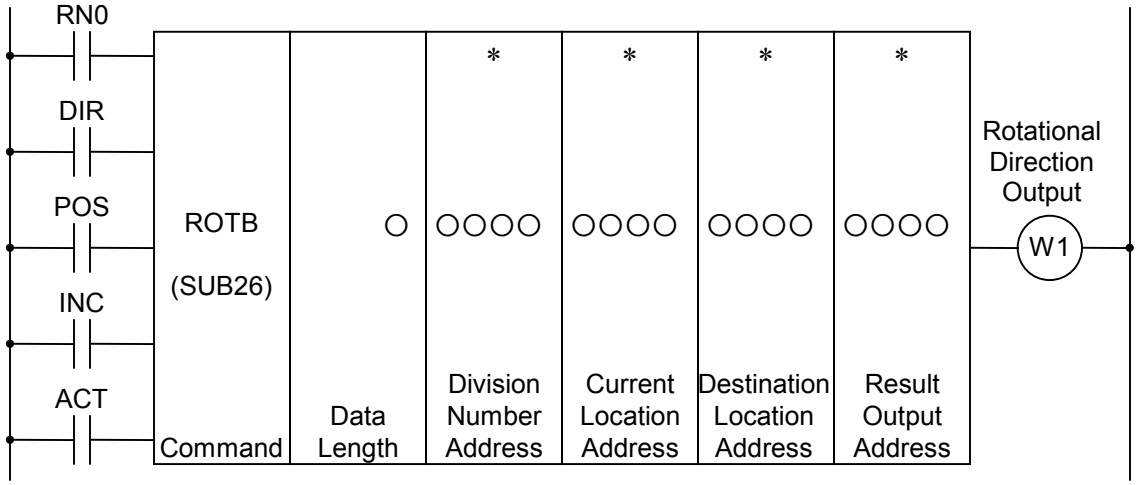


Figure 9-31: Format for the ROTB Command

Control Values

1) Rotational Body Starting Number (RN0)
   RN0 = 0: The rotational body location number starts from 0.
   RN0 = 1: The rotational body location number starts from 1.

2) Whether Or Not To Distinguish the Direction of Rotation (DIR)
   DIR = 0: No distinction of direction of rotation; it will always be FOR direction.
   DIR = 1: Distinguishes the direction of rotation. See the *Rotational Direction Output (W1)* section for the specifications.

3) Type of Calculation (POS)
   POS = 0: Calculates to the destination.
   POS = 1: Calculates to the location one before the destination.

4) Calculate Location or the Number of Steps (INC)
   INC = 0: Calculates the number of a location. (To calculate the location one before the destination, specify INC = 0 and POS = 1.)
   INC = 1: Calculates the number of steps. (To calculate the difference between the current location and the destination, specify INC = 1 and POS = 0.)

5) Action Command (ACT)
   ACT = 0: No execution of the ROT command. No change in W1.
   ACT = 1: Execution of the ROT command.
   Usually ACT = 0, and when a calculation result is necessary, ACT = 1.

Parameters

1) Data Length
   Specifies the byte length of the data in the first digit of the parameters.
   When 1: Data is 1-byte binary data.
   When 2: Data is 2-byte binary data.
   When 4: Data is 4-byte binary data.
   The numeric data (rotational body divisions and the current location data) is in binary format, and the specified number of bytes is necessary in memory.

2) Division Number Address
   The address where the number of rotational body divisions.

3) Current Location Address
   This is the address that contains the current location of the machine.

4) Destination Location Address
   This is the address where the destination location (or the command location) is stored. For example, this is the output of the NC, stored among the T-code address region.

5) Result Output Address
   This is the address where the result of the unit is stored. The result can be the number of steps to turn to the destination, to one before the destination, or the location of the step one before the destination. When using this value, make sure that ACT = 1.

Rotational Direction Output (W1)

When using the shortcut controls, the direction of rotation is output to W1.

W1 = 0: Specifies the FOR direction.
W1 = 1: Specifies the REV direction.

See Figure 9-30 for definitions for FOR (forward) and REV (reverse). When looking from a specific point, the direction that increases the location number is FOR, the direction that decreases the location number is REV. The address of W1 can be assigned arbitrarily. In order to use the results from W1 you must make sure that ACT = 1.

Example of ROTB Command Usage

The ladder diagram showing the shortcut rotation control of the twelve-position rotational body, where the speed reduces one position prior to the destination, is shown in Figure 9-32.

- The destination is specified by the ServoWorks mapping tables as the 32 bits of binary in addresses F26-F29.
- The current location is specified from the machine tool in the signal (address X41) in binary.
- The location of the position one prior to the destination is output into address R230.
- Execution start is designated by the signal TF (address F7.3), an output from the ServoWorks Motion Engine.
- A comparator (COIN) is used to detect the speed reduction position and the stopping position.

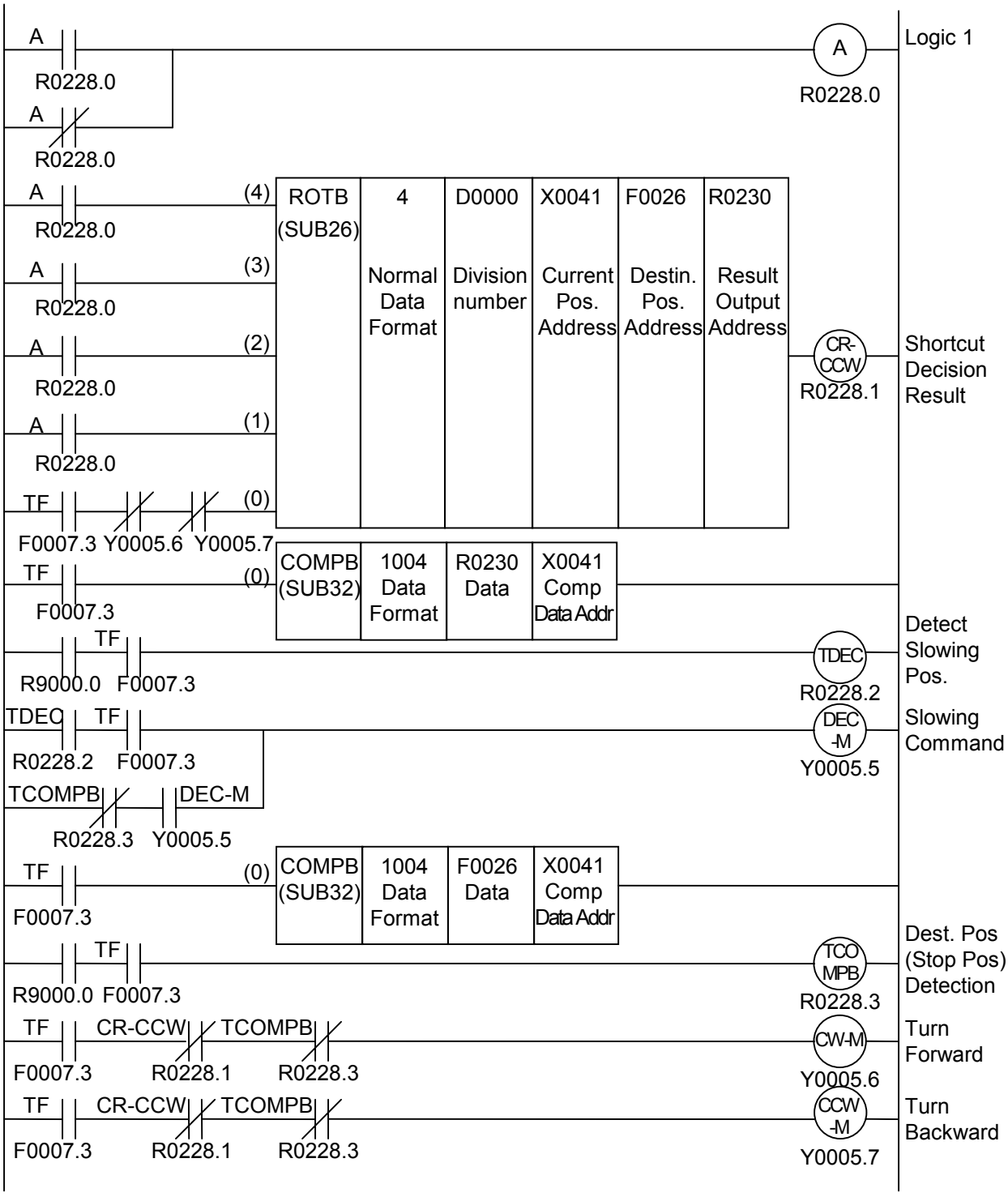


Figure 9-32: Ladder Diagram Example Using the ROTB Command

9.2.10 COD (Code Transformation)

Function

This command transforms BCD code to 2- or 4-digit BCD numbers. The input and output address as well as the transformation data table are required, as shown in Figure 9-33. The input address specifies, in 2-digit BCD, the row number of the table to access. The values are entered into the transformation data table in 2-digit or 4-digit BCD. The value at the output address will contain the value stored in the specified row in the transformation data table. For example, as in Figure 9-33, the input contains the value 3, so the contents of the transformation data table at row 3, 137, is output to the data output address.

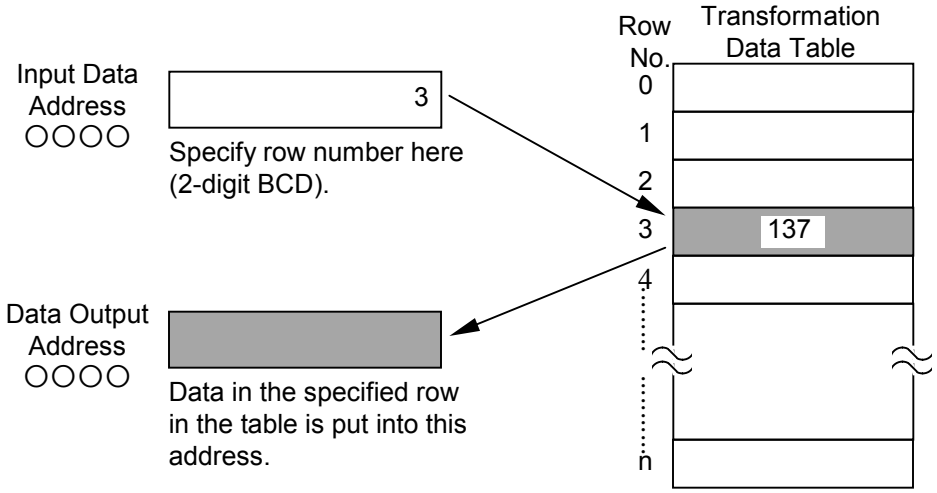


Figure 9-33: Code Transformation Using the COD Command

Format

Figure 9-34 shows the format for describing the command, and Table 9-11 shows the coding format.

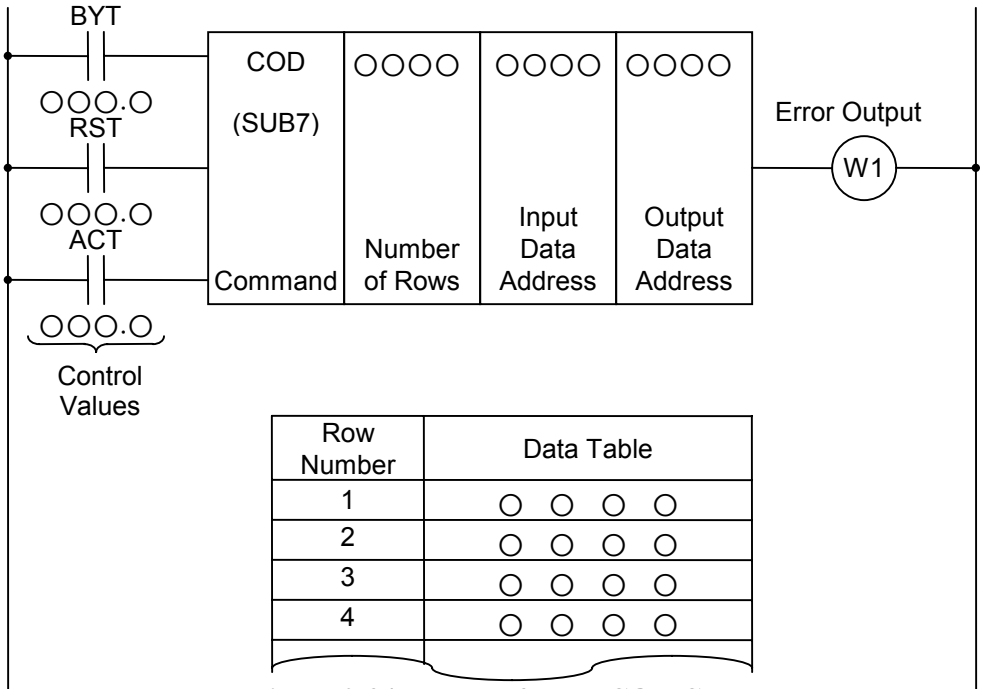


Figure 9-34: Format for the COD Command

Coding Sheet				Result History Register			
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**
1	RD	○○○ . ○	BYT				BYT
2	RD.STK	○○○ . ○	RST			BYT	RST
3	RD.STK	○○○ . ○	ACT		BYT	RST	ACT
4	SUB	7	COD Command		BYT	RST	ACT
5	(PRM)	○○○○	Number of Rows (1)		BYT	RST	ACT
6	(PRM)	○○○○	Input Data Address (2)		BYT	RST	ACT
7	(PRM)	○○○○	Output Data Address (3)		BYT	RST	ACT
8	(PRM)	○○○○	Row 0 Data in Table (4)		BYT	RST	ACT
9	(PRM)	○○○○	Row 1 Data (5)		BYT	RST	ACT
10	. . .	. . .	. . .		BYT	RST	ACT
11	WRT	○○○ . ○	W1, Error Output		BYT	RST	W1

Table 9-11: Coding Format of the COD Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data inside the transformation data table is 2-digit BCD.
   BYT = 1: The data inside the transformation data table is 4-digit BCD.

2) Error Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the COD Command. No change in W1.
   ACT = 1: Execution of the COD Command.

Parameters

1) Number of Rows
   The size of the transformation data table (the number of rows). Possible values lie within the range 00~99. If n is the last row number, then value n+1 should be the table size.

2) Input Data Address
   The data from the transformation data table is specified using a row number. The input address is the location where the row number can be found. The row number is 1 byte (2-digit BCD).

3) Output Data Address
   The output address is the location where the data from the transformation data table is to be transferred to. If the conversion data is 2-digit BCD, then the memory location specified is 1 byte. If the conversion data is 4-digit BCD, then the memory location is 2 bytes.

Error Output (W1)

W1 = 0: No error.
W1 = 1: Error.

For example, if the command tries to access a row number greater than the table size, then W1 becomes "1." When W1 = 1, we recommend handling the error with an interlock appropriate to each application, such as making the error lamp in the machine control panel light on and off, or stopping the rotation of the axis.

Transformation Data Table

The size of the transformation data table is 100, rows 00 to 99. The data in the table can either be in 2- or 4-digit BCD, as specified in the control values.

9.2.11 CODB (Binary Code Conversion)

Functions

This command converts binary style data into 1-, 2- or 4-byte size binary. To carry out the data conversion, the command requires the input and output data address and the transformation data table, as shown in Figure 9-35. CODB function differs from the COD function (Section 9.2.10), in that it utilizes numeric data that is in 1-, 2-, or 4-byte size binary style and the transformation data table size can be expanded to a maximum of 256 rows.

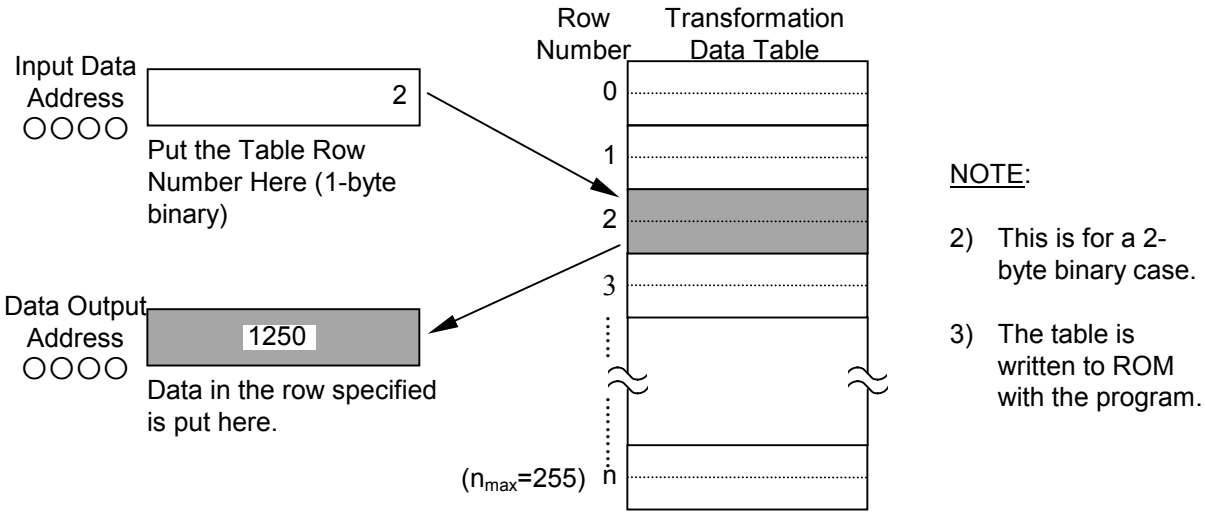


Figure 9-35: Code Transformation Using the CODB Command

Format

Figure 9-36 shows the format for describing the command.

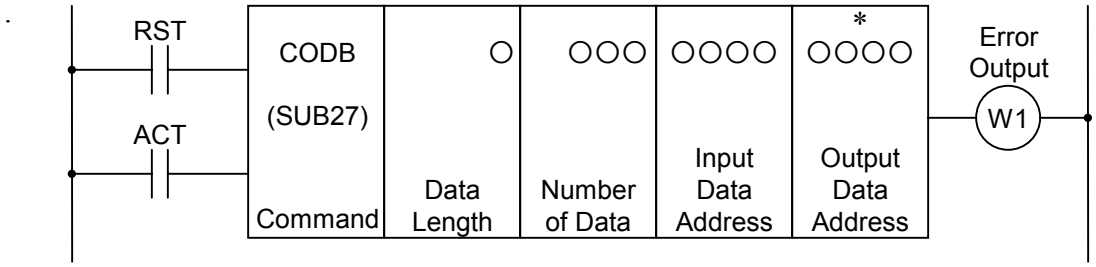


Figure 9-36: Format for the CODB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the CODB command.
   ACT = 1: Execution of the CODB command.

Parameters

1) Data Length
   The byte length of the numeric binary data inside the transformation data table.
   When 1: Numeric data is 1-byte binary data.
   When 2: Numeric data is 2-byte binary data.
   When 4: Numeric data is 4-byte binary data.

2) Number of Data
   The size of the transformation data table. Size ranges from 0~255; the maximum number of rows is 256.

3) Input Data Address
   The data from the transformation data table is specified using a row number. The input address is the location where the row number can be found. The row number is 1 byte (eight bits).

4) Output Data Address
   The location to where the data from the transformation data table is to be transferred. The amount of memory necessary is the byte size specified in the command.

Transformation Data Table

The size of the conversion data table ranges from 0~255 for a maximum of 256 rows.

Error Output (W1)

W1 = 0: No error.
W1 = 1: Error.

9.2.12 MOVE (Masked Data Transfer)

Function

This command performs a bit-wise product (AND) of the specified data and the input data, the value output to the output address. Use this command when you are specifying only certain bits of the 8-bit signal within an address, also called "masking the value."

(Data) * (Input Data)→ Output into Output Data Address

The input data is 1 byte (8 bits).

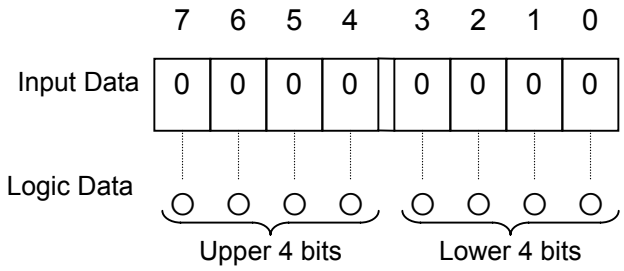


Figure 9-37: Input Data and Logic Data for the MOVE Command

Format

Figure 9-38 shows the format for describing the command, and Table 9-12 shows the coding format.

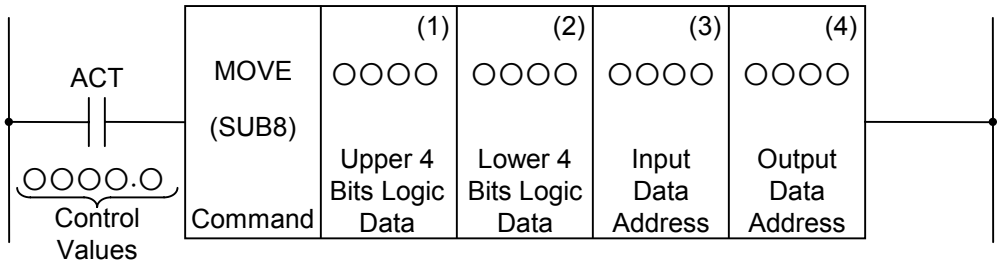


Figure 9-38: Format for the MOVE Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOO	O	ACT
2	SUB	8		MOVE Command
3	(PRM)	OOOO		Upper 4 Bits Logic Data (1)
4	(PRM)	OOOO		Lower 4 Bits Logic Data (2)
5	(PRM)	OOOO		Input Data Address (3)
6	(PRM)	OOOO		Output Data Address (4)

Result History Register

ST3	ST2	ST1	ST0
			ACT
			ACT
			ACT
			ACT
			ACT
			ACT

Table 9-12: Coding Format of the MOVE Command

Control Values

Action Command (ACT)

ACT = 0: No execution of the MOVE command.
ACT = 1: Execution of the MOVE command.

Example of MOVE Command Usage

When the code signal and a different signal are mixed inside the input signal address X35 of the machine, the bits not corresponding to the address X35 code signal are an obstruction in comparing the code signal of address X35 to a code signal of a different address. Therefore, using the MOVE command, only the code signal in address X35 is output into address R210.

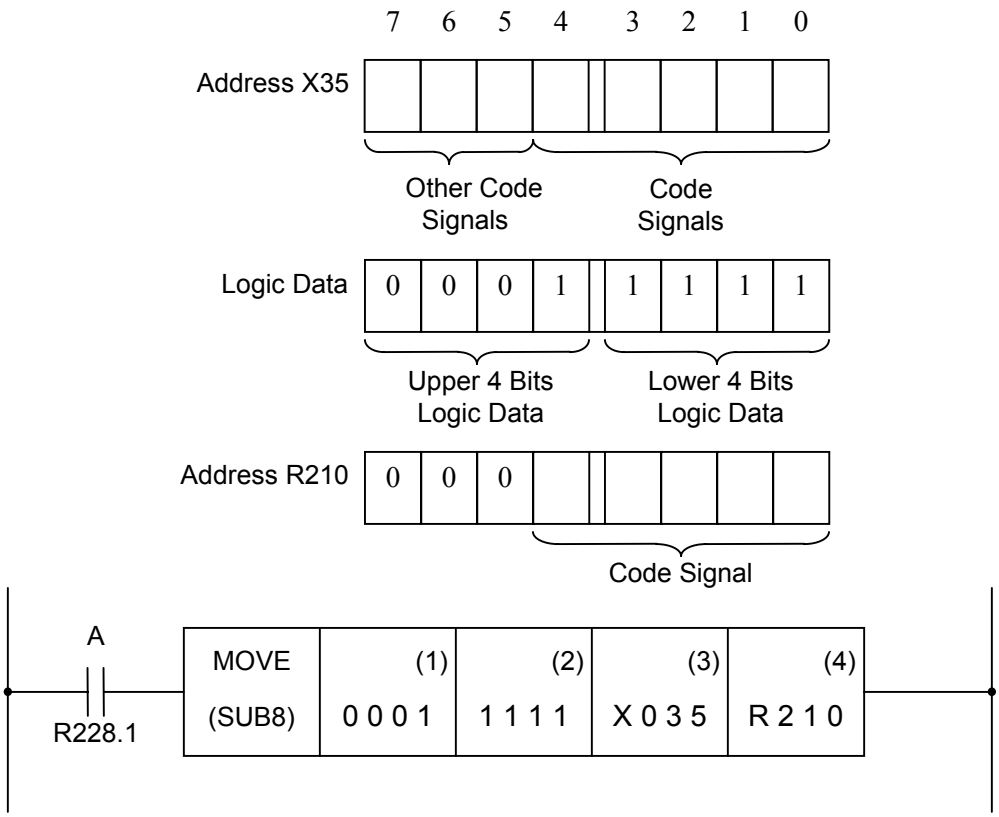


Figure 9-39: Ladder Diagram Example Using the MOVE Command

9.2.13 MOVOR (Bit-Wise Sum Data Transfer)

Function

This command carries out the bit-wise sum (OR) of the input data and the logic sum data (each is 1 byte) and transfers the data into the output address.

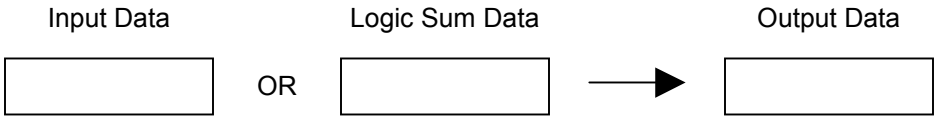


Figure 9-40: Function of the MOVOR Command

Format

Figure 9-41 shows the format for describing the command.

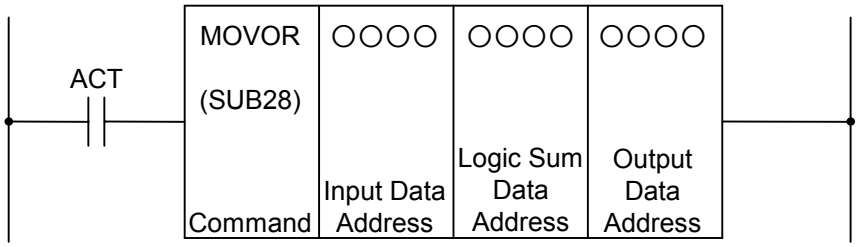


Figure 9-41: Format for the MOVOR Command

Control Values

Action Command (ACT)

ACT = 0: No execution of the MOVOR command.
ACT = 1: Execution of the MOVOR command.

Parameter

1) Input Data Address
   The address for the input data.

2) Logic Sum Data Address
   The address where the logic sum is carried out.

3) Output Data Address
   Address for output after the sum. You can output the result back into the logic sum data location, by specifying the output address to be the same address as the logic sum data address.

9.2.14 COM (Common Line Control)

Function

This command can force a region of coils to be in the off state. The region can be specified by the number of successive coils, or until the common line control termination command (COME). See Figure 9-42. If the number of coils is specified to be a non-zero number, then that many coils are automatically turned off. If the number of coils is zero, then the region up to the COME command is turned off. If the coil number is non-zero, but a COME command is there, then an error is displayed when the program ends.

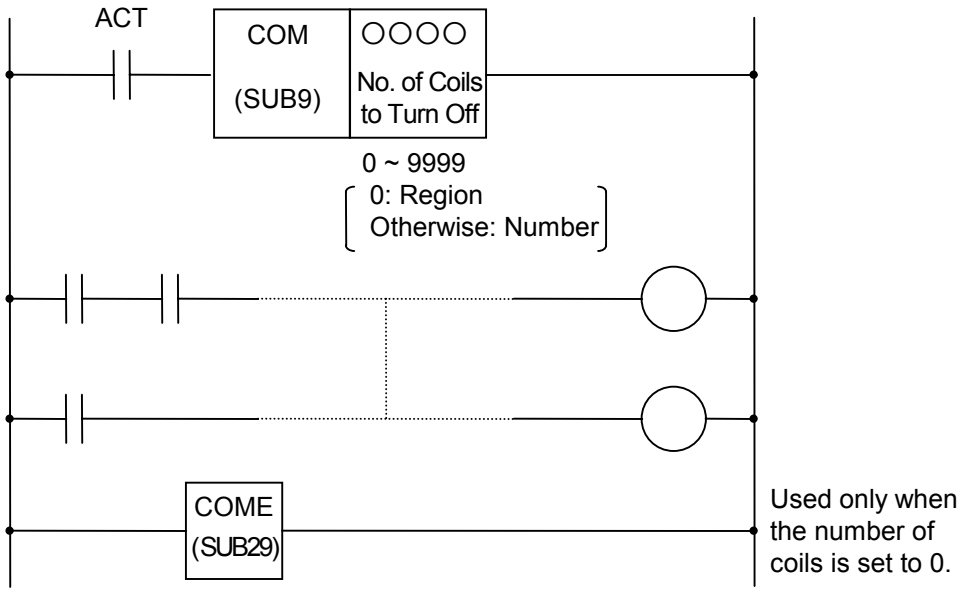


Figure 9-42: Function of the COM Command

Format

Figure 9-43 shows the format for describing the command.

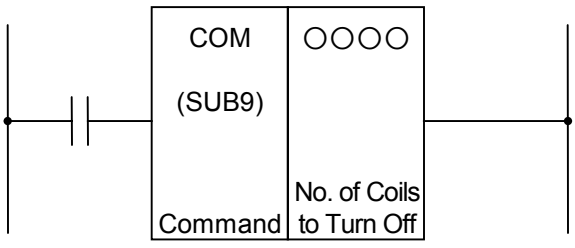


Figure 9-43: Format for the COM Command

Control Values

Action Command (ACT)

ACT = 0: Sets the coils in the specified region to "0" unconditionally.
ACT = 1: No action.

The command takes effect starting at the step after the COM command.

Parameters

Number of Coils to Turn Off

Must be a number between 0-9999.

When 0 is specified: Becomes area-specific as explained above.
When a number other than 0 is specified: Becomes coil number specific.

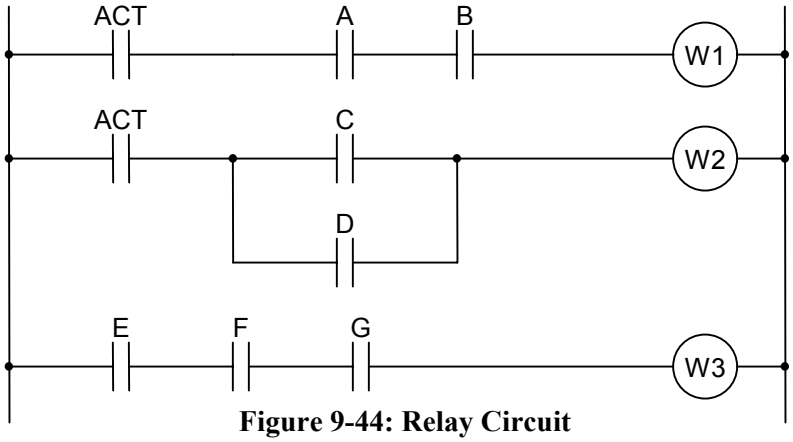


Figure 9-44: Relay Circuit

CAUTION

The commands within the specified area are executed regardless of the ACT value. However, when ACT=0, the result coil unconditionally becomes 0.

CAUTION

Nested COM calls are not allowed.

CAUTION

Coils written with the command WRT.NOT are unconditionally set to 1 when ACT=0 for the COM.

Examples of COM Command Usage

See Figures 9-45 and 9-46 for examples of COM command usage.

NOTE:

(a) ACT=1: Executes from the next step.

(b) ACT=0: Unconditionally turns W1, W2 off, starts at next step. (b)

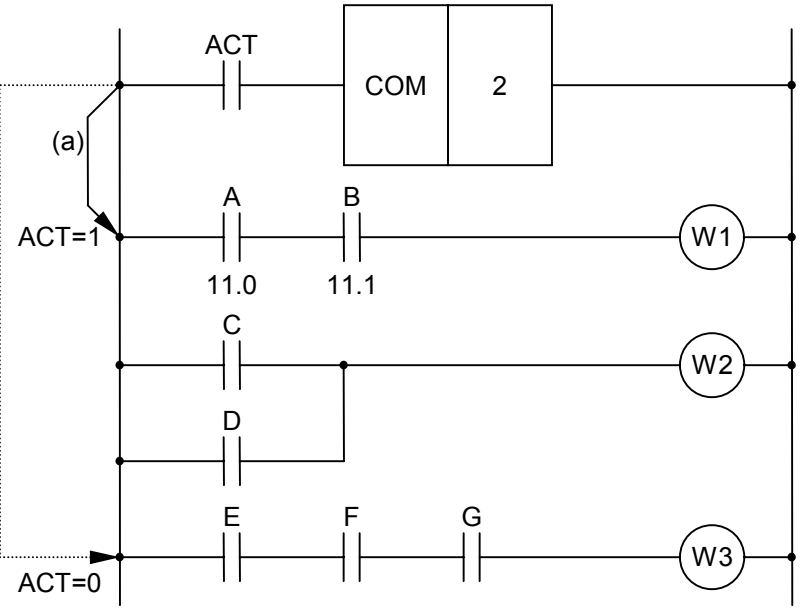


Figure 9-45: Ladder Diagram Using the COM Command

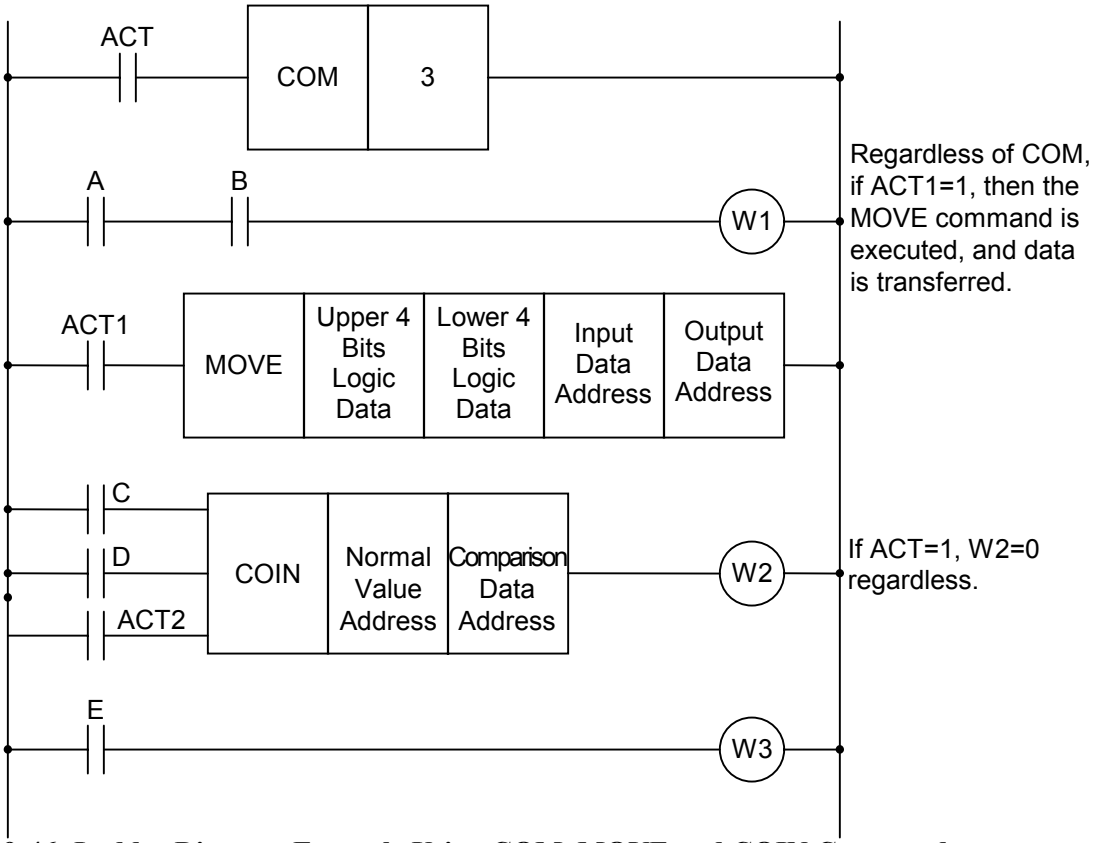


Figure 9-46: Ladder Diagram Example Using COM, MOVE and COIN Commands

9.2.15 COME (Common Line Control Termination)

Function

This command ends the region specified by a COM command. You cannot use this command by itself; you must always pair it with the COM command.

Format

Figure 9-47 shows the format for describing the command.

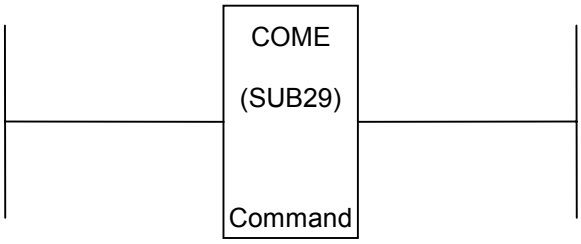


Figure 9-47: Format for the COME Command

9.2.16 JMP (Jump)

Function

This command skips calculations for a region of coils. The region can be specified by the number of successive coils, or until the Jump termination command (JMPE). If the number of coils is specified to be a non-zero number, then that many coils are automatically skipped. If the number of coils is zero, then the region up to the JMPE command is turned off (skipped). If the coil number is non-zero, but a JMPE command is there, then an error is displayed when the program ends.

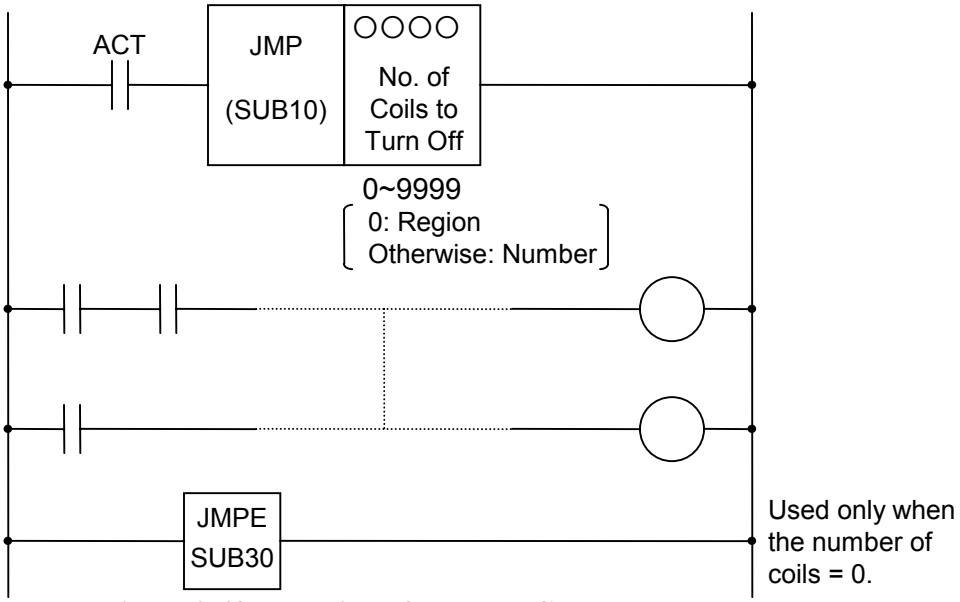


Figure 9-48: Function of the JMP Command

Format

Figure 9-49 shows the format for describing the command, and Table 9-13 shows the coding format.

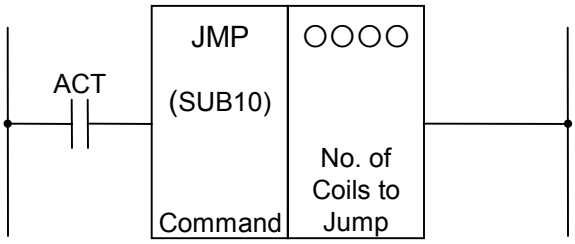


Figure 9-49: Format for the JMP Command

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○	. ○	ACT
2	SUB	10		JMP Command
3	(PRM)	○○○○		No. of Coils to Jump

Table 9-13: Coding Format of the JMP Command

Control Values

Action Command (ACT)

ACT = 0: No jump. Program continues execution after the JMP command.

ACT = 1: Skips calculations for the specified area, restarting calculations from the following step.

Parameter

Number of Coils to Jump

Must be a number between 0-9999.

When 0 is specified: Becomes an area specific jump.

When a value other than 0 is specified: Becomes a coil number specific jump.

Example of JMP Command Usage

See Figure 9-50 for a ladder diagram example utilizing the JMP command.

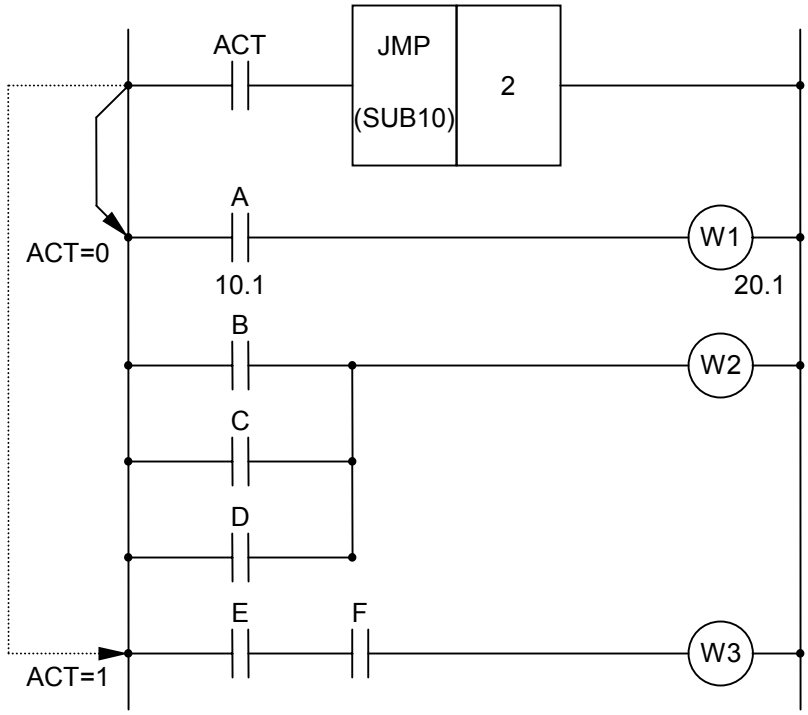


Figure 9-50: Ladder Diagram Example Using the JMP Command

When ACT = 0, the program continues execution after the JMP command (i.e. no jump).

When ACT = 1, the sequences for the region delimited by the number of coils are "jumped" and the results of the logic calculations do not affect the coil value. In other words, when ACT = 1, in Figure 9-50, when the signal changes from "1"↔ "0", W1 remains under the same condition as before ACT = 1. Similarly, even if signals B, C, and D change, W2 is not changed. However, JMP commands do not decrease the execution time of the program.

9.2.17 JMPE (Jump Termination)

Function

This command marks the end of the region specified for the Jump Command (JMP). You cannot use this command by itself; you must always pair it with the JMP command.

Format

Figure 9-51 shows the format for describing the command.

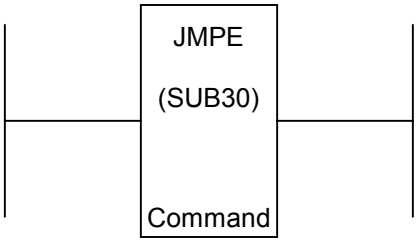


Figure 9-51: Format for the JMPE Command

9.2.18 PARI (Parity Check)

Function

This command carries out the parity check for the code signal and outputs an error if the parity check fails. The check looks at 1 byte of data (8 bits) and performs odd or even parity check.

A parity check is used to reveal errors in storage or transmission. An extra bit (called the "parity bit") is added to a byte or word. In even parity, that bit is set to either "0" or "1," whichever would make an even number of "1" bits in the byte. In odd parity, that bit is also set to either "0" or "1," whichever would make an odd number of "1" bits in the byte.

For example, for even parity, if the first seven bits of a byte are 0 1 1 0 1 0 0, then the parity bit needs to be "1" to make four (an even number) "1" bits. The entire byte would be as follows: 1 0 1 1 0 1 0 0.

A single parity bit can only reveal odd bit errors, since if an even number of bits is wrong, then the parity bit will not change. Also, if there is an error, the parity check won't be able to pinpoint which bit is wrong. However, parity checks are still quite useful.

Format

Figure 9-52 shows the format for describing the command, and Table 9-14 shows the coding format.

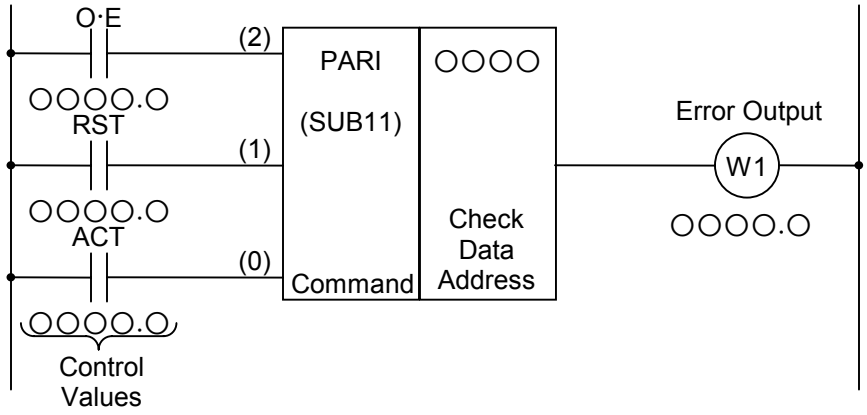


Figure 9-52: Format for the PARI Command

Coding Sheet				Result History Register			
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**
1	RD	○○○ . ○	ACT				O·E
2	RD.STK	○○○ . ○	ACT			O·E	RST
3	RD.STK	○○○ . ○	ACT		O·E	RST	ACT
4	SUB	11	PARI Command		O·E	RST	ACT
5	(PRM)	○○○○	Check Data Address		O·E	RST	ACT
6	WRT	○○○○ . ○	W1, Error Output		O·E	RST	W1

Table 9-14: Coding Format of the PARI Command

Control Values

1) Odd or Even Parity (O·E)
   O·E = 0: Even parity check.
   O·E = 1: Odd parity check.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0". If there is a parity error and RST=1, then the program is reset.

3) Action Command (ACT)
   ACT = 0: No execution of the PARI command. No parity check is carried out. No change in W1.
   ACT = 1: Execution of the PARI command. A parity check is carried out.

Error Output (W1)

W1 = 0: No error.
W1 = 1: Error. (i.e. when ACT=1 and the parity check fails.)

Example of PARI Command Usage

Figure 9-53 shows the odd number parity check of a code signal that gets input into address X036.

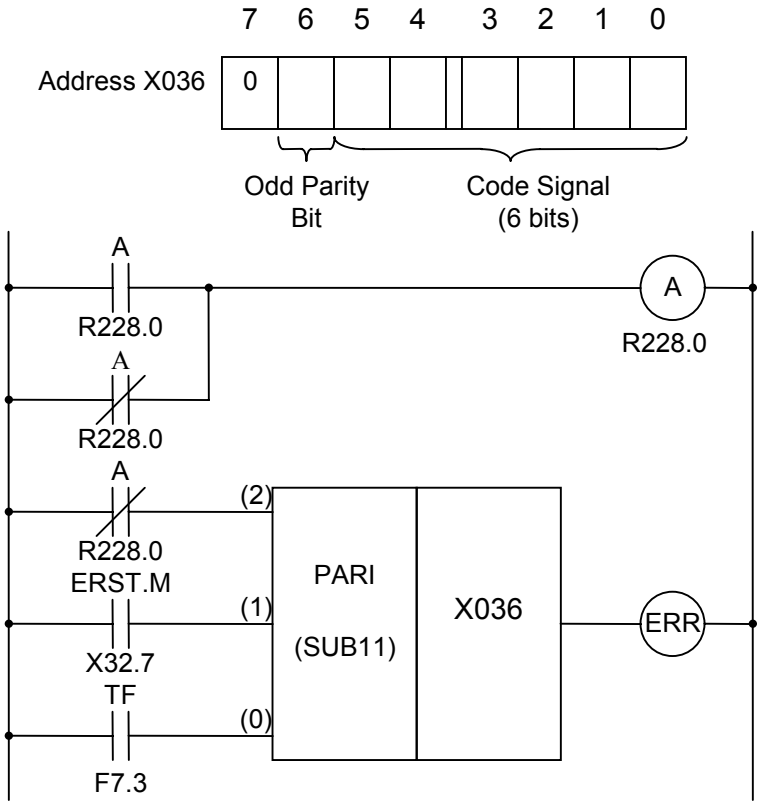


Figure 9-53: Ladder Diagram Example Using the PARI Command

⚠ CAUTION

Out of bits 0~7, the bits that do not correspond to parity check have to be "0".

9.2.19 DCNV (Data Conversion)

Function

This command converts binary code into BCD code, and vice versa.

Format

Figure 9-54 shows the format for describing the command, and Table 9-15 shows the coding format.

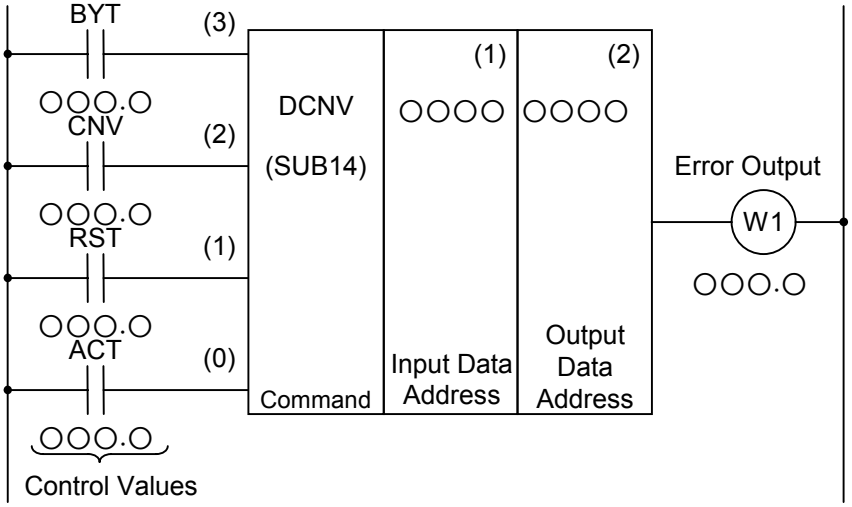


Figure 9-54: Format for the DCNV Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○	.○	BYT
2	RD.STK	○○○	.○	CNV
3	RD.STK	○○○	.○	RST
4	RD.STK	○○○	.○	ACT
5	SUB	14		DCNV Command
6	(PRM)	○○○○	.○	Input Data Address (1)
7	(PRM)	○○○○		Output Data Address (2)
8	WRT	○○○	.○	W1, Error Output

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	CNV
	BYT	CNV	RST
BYT	CNV	RST	ACT
BYT	CNV	RST	ACT
BYT	CNV	RST	ACT
BYT	CNV	RST	ACT
BYT	CNV	RST	W1

Table 9-15: Coding Format of the DCNV Command

Control Values

1) Data Size (BYT)
   BYT = 0: Processing 1-byte data (8 bits).
   BYT = 1: Processing 2-byte data (16 bits).

2) Conversion Format (CNV)
   CNV = 0: Converts binary code into BCD code.
   CNV = 1: Converts BCD code into binary code.

3) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. When W1 = 1 and RST = 1, then W1 becomes "0."

4) Action Command (ACT)
   ACT = 0: No conversion of data. No change in W1.
   ACT = 1: Carries out data conversion.

Error Output (W1)

W1 = 0: Normal (no conversion error).
W1 = 1: Conversion error. (W1 = 1 when the input data that needs to be BCD data is in binary, or when converting binary data into BCD data, the data size (byte size) goes over the previously specified value.)

Example of DCNV Command Usage

See Figure 9-10(c)

9.2.20 DCNVB (Extended Data Conversion)

Function

This command converts binary code of 1, 2, or 4 bytes into BCD code, or BCD code into binary code. Memory with specified byte length is necessary for the conversion result output data.

Format

Figure 9-55 shows the format for describing the command.

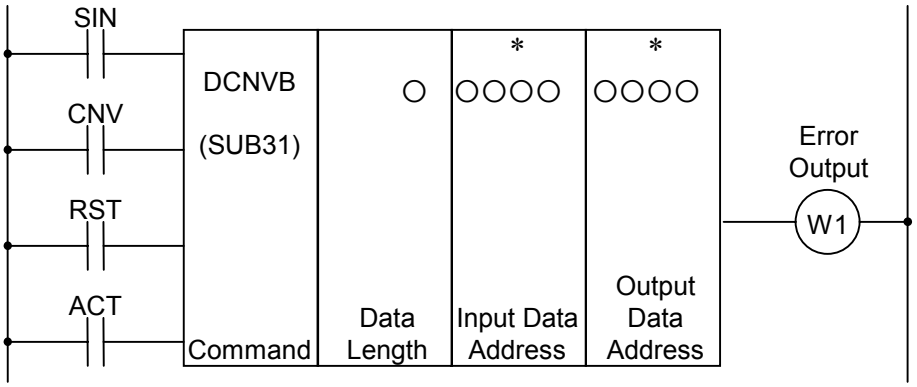


Figure 9-55: Format for the DCNVB Command

Control Values

1) Sign for BCD → Binary (SIN)
   SIN has meaning only when converting BCD code data into binary code data, and represents the sign for BCD code data. SIN has no meaning when converting binary into BCD; however, you cannot truncate this operand.
   SIN = 0: Positive input data (BCD code).
   SIN = 1: Negative input data (BCD code).

2) Conversion Format (CNV)
   CNV = 0: Converts binary code data into BCD code data.
   CNV = 1: Converts BCD code data into binary code data.

3) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

4) Action Command (ACT)
   ACT = 0: No conversion of data. No change in W1.
   ACT = 1: Carries out data conversion.

Parameters

1) Data Length
   Specifies the byte length of the data in the first digit of the parameters.
   When 1: Numeric data is 1-byte binary data.
   When 2: Numeric data is 2-byte binary data.
   When 4: Numeric data is 4-byte binary data

2) Input Data Address
   The address where the input data is stored.

3) Output Data Address
   The address where the converted BCD or binary code result is output.

Error Output (W1)

W1 = 0: Normal (no conversion error).
W1 = 1: Conversion error. (W1=1 when the input data is binary data that should be BCD data, or when the byte size goes over the specified byte size when converting binary data into BCD code).

Calculation Result Register (R9000)

The calculation information gets set, and when each bit is "1", it means the following as shown below. See Figure 9-56 for a description of positive or negative conversion of binary data into BCD code.

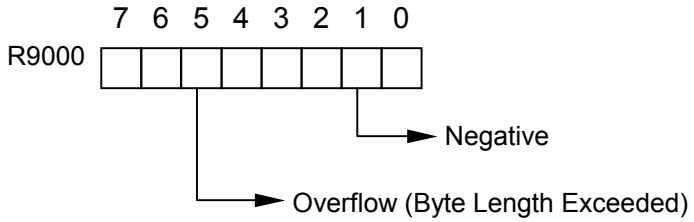


Figure 9-56: Calculation Result Register for the DCNVB Command

9.2.21 COMP (Compare)

Function

This command compares the sizes of the input value and the comparison value.

Format

Figure 9-57 shows the format for describing the command, and Table 9-16 shows the coding format.

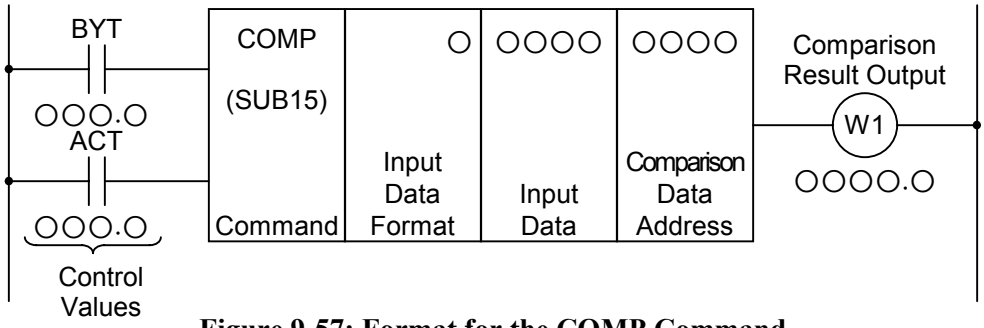


Figure 9-57: Format for the COMP Command

Coding Sheet

Step No.	Command	Address No.	Bit no.	Description
1	RD	OOO	O	BYT
2	RD.STK	OOO	O	ACT
3	SUB	15		COMP Command
4	(PRM)	O		Input Data Format
5	(PRM)	OOOO		Input Data
6	(PRM)	OOOO		Comparison Data Address
7	WRT	OOO	O	W1, Comparison Result Output

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	W1

Table 9-16: Coding Format of the COMP Command

Control Values

1) Data Size (BYT)
   BYT = 0: The processed data (input and comparison value) is 2-digit BCD.
   BYT = 1: The processed data (input and comparison value) is 4-digit BCD.

2) Action Command (ACT)
   ACT = 0: No execution of the COMP command. No change in W1.
   ACT = 1: Execution of the COMP command. The comparison result is output to W1.

Parameters

1) Input Data Format
   0: Specifies the input data as a constant.
   1: Specifies the input data as an address – indirection. (Specifies the address where the input data is stored instead of directly specifying the data.)

2) Input Data
   Input data can either be a constant or an address of a constant. The different types can be distinguished by specifying the "Input Data Format" parameter.

3) Comparison Data Address
   The address of the data to be compared.

Comparison Result Output (W1)

W1 = 0: Input Value > Comparison Value.
W1 = 1: Input Value ≤ Comparison Value.

9.2.22 COMPB (Binary Compare)

Function

This command compares 1-, 2-, or 4-byte size binary data, putting the result to the calculation result register (R500). The input data and the comparison data must be the same length.

Format

Figure 9-58 shows the format for describing the command.

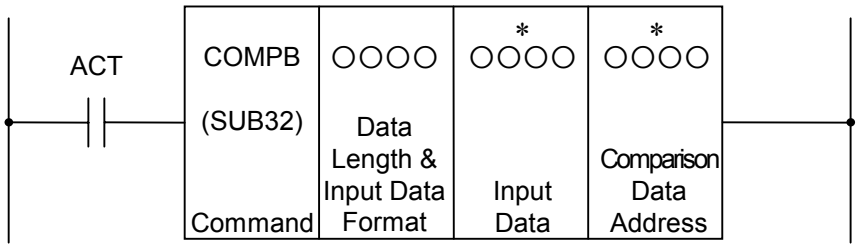


Figure 9-58: Format for the COMPB Command

Control Values

Action Command (ACT)

ACT = 0: No execution of the COMPB command. No change in W1.

ACT = 1: Execution of the COMPB command. The comparison result is output to W1.

Parameters

1) Data Length and Input Data Format
The data length (1, 2, or 4 bytes) and the input data format (constant data or data address).

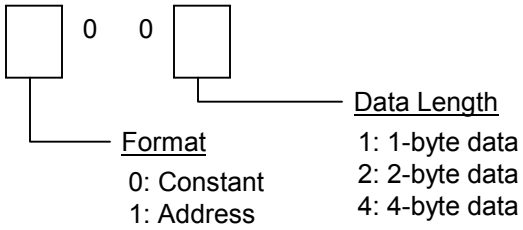


Figure 9-59: Parameters Format Specification for the COMPB Command

2) Input Data
The format of the input data is determined by the specification of the "Input Data Format" parameter. Depending on what format was specified, this parameter could be an input data address or an input data constant.

3) Comparison Data Address
The address where the comparison data will be stored.

Calculation Result Register (R9000)

A "1" in the following bit locations signifies the following:

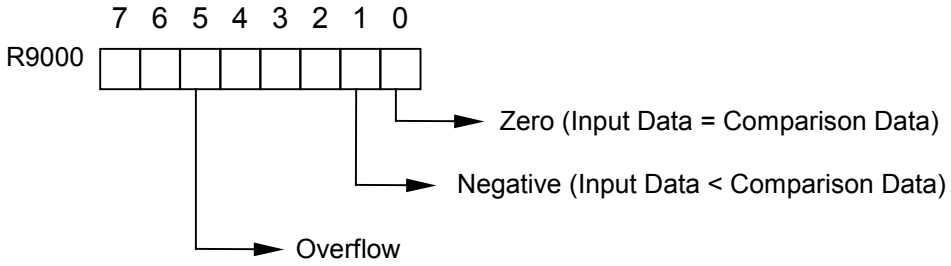


Figure 9-60: Calculation Result Register for the COMPB Command

9.2.23 COIN (Equality Check)

Function

This command determines if the input value and the comparison value are the same. This command can only be used when the data is in BCD format.

Format

Figure 9-61 shows the format for describing the command, and Table 9-17 shows the coding format.

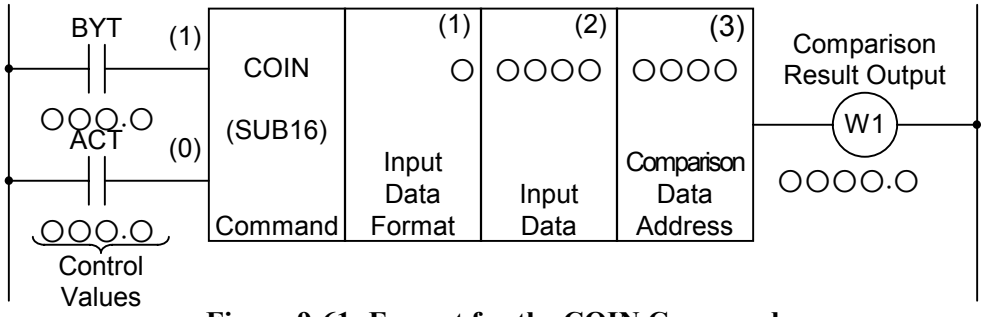


Figure 9-61: Format for the COIN Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOO	O	BYT
2	RD.STK	OOO	O	ACT
3	SUB	16		COIN Command
4	(PRM)	O		Input Data Format
5	(PRM)	OOOO		Input data
6	(PRM)	OOOO		Comparison Data Address
7	WRT	OOO	O	W1, Comparison Result Output

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	W1

Table 9-17: Coding Format of the COIN Command

Control Values

1) Data Size (BYT)
   BYT = 0: Data processed (input and comparison data) is 2-digit BCD.
   BYT = 1: Data processed (input and comparison data) is 4-digit BCD.

2) Action Command (ACT)
   ACT = 0: No execution of the COIN command. No change in W1.
   ACT = 1: Execution of the COIN command. Outputs the result to W1.

Parameters

1) Input Data Format
   0: Specifies the input data as a constant.
   1: Specifies the input data as an address.

2) Input Data
   Input data can either be an input data address or an input data constant. The different types can be distinguished by the "Input Data Format" parameter.

3) Comparison Data Address
   The address where the comparison data will be stored.

Comparison Result Output (W1)

W1 = 0: Input Data ≠ Comparison Data.
W1 = 1: Input Data = Comparison Data.

9.2.24 SFT (Shift Register)

Function

This command shifts 2 contiguous bytes (16 bits) of data 1 bit to the right or to the left. W1=1 when the data "1" shifts out from the left margin (bit 15 was "1") on a shift-left, or the right margin (bit 0 was "1") on a shift-right. W1 is basically the overflow value.

Format

Figure 9-62 shows the format for describing the command.

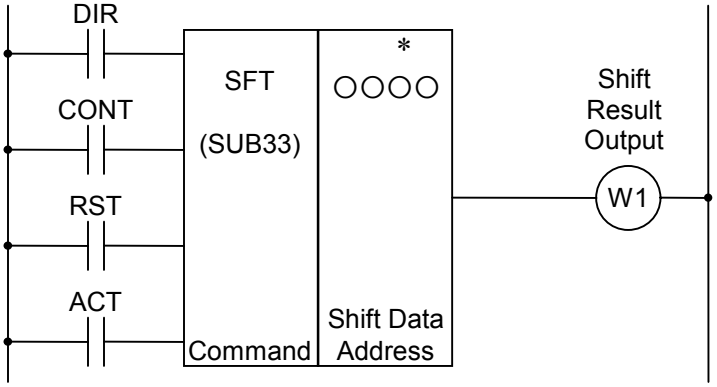


Figure 9-62: Format for the SFT Command

Control Values

1) Shift Direction (DIR)
   DIR = 0: Shifts to the left.
   DIR = 1: Shifts to the right.

2) Condition Specification (CONT)

CONT = 0: This is the normal shift, where every bit after the shift holds the value of the bit next to it (left or right depending on DIR) before the shift. If shifting left, bit 15 will get the value previously in bit 14, etc. Furthermore, a 0 is inserted into the first bit of the shift (bit 0 for shift left, 15 for shift right).

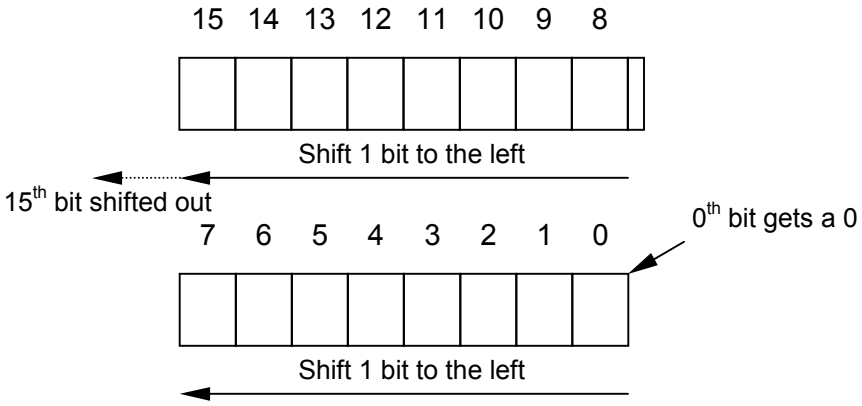


Figure 9-63: Condition Specification CONT = 0 for the SFT Command – Shift Left Example

CONT = 1: Shifting is the same as when CONT = 0, however, when the original bit was a "1", it remains a "1". Hence, all bits with a 1 propagate in the appropriate direction.

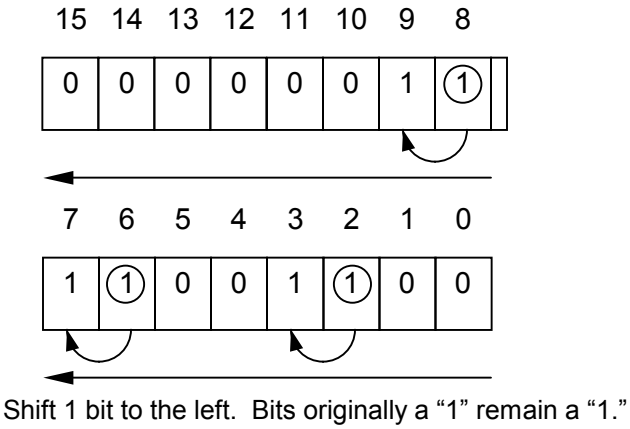


Figure 9-64: Condition Specification CONT = 1 for the SFT Command – Shift Left Example

3) Reset (RST)

Resets (W1 = 0) the result from the shifting process, the value that got shifted out.

RST = 0: No reset.

RST = 1: Resets. In other words, W1 becomes "0."

4) Action Command (ACT)
ACT = 0: Shift process is not carried out.
ACT = 1: Carries out the shift process. In order to just shift 1 bit, after executing ACT = 1, immediately change to ACT = 0.

Parameters

Shift Data Address
The address of the data to be shifted. The address needs to be 2 contiguous bytes of data. The bit numbers below are described as bits 0~15, but when specifying for the program, there is an address number for each byte (8 bits) and one can only specify from bit numbers 0~7.

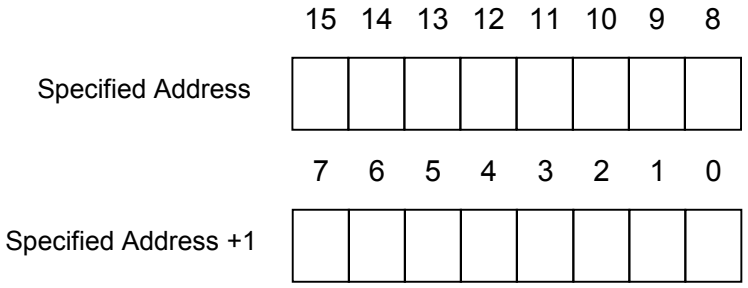


Figure 9-65: Shift Data Address for the SFT Command

Shift Result Output (W1)

W1 = 0: Shows that a "0" was shifted out.
W1 = 1: Shows that a "1" was shifted out.

9.2.25 DSCH (Data Search)

Function

In the ServoWorksPLC, you can use something called a data table, which will be explained later (see *Section 9.3: PLC Data Table*). This command is related to this data table. This command checks whether certain data is inside the data table, and if it is it outputs the row number where the data resides in the data table. It also replies accordingly if the data does not exist in the table.

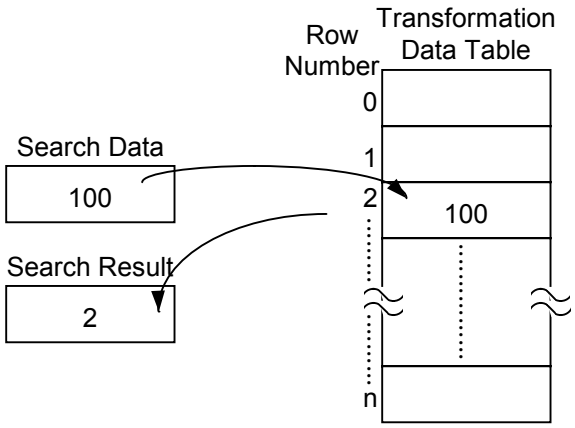


Figure 9-66: Function of the DSCH Command

CAUTION

The leading data table address selected in the DSCH command parameters becomes the 0^{th} table internal number (row number). This table internal number differs from the table internal number mentioned in Section 9.3 for the data table for the PLC.

Format

Figure 9-67 shows the format for describing the command, and Table 9-18 shows the coding format.

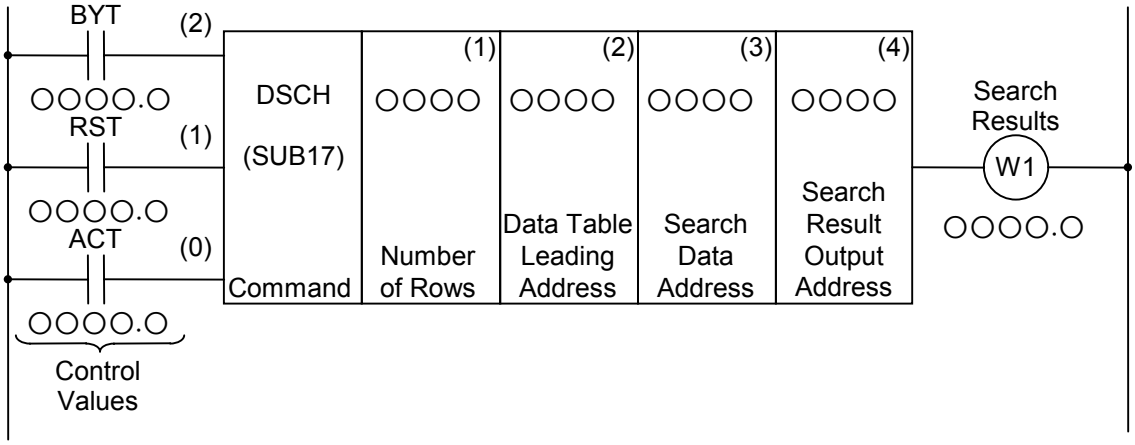


Figure 9-67: Format for the DSCH Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOO	O	BYT
2	RD.STK	OOO	O	RST
3	RD.STK	OOO	O	ACT
4	SUB	17		DSCH Command
5	(PRM)	OOOO		Number of Rows
6	(PRM)	OOOO		Data Table Leading Address
7	(PRM)	OOOO		Search Data Address
8	(PRM)	OOOO		Search Result Output Address
9	WRT	OOO	O	W1, Search Results

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	RST
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	W1

Table 9-18: Coding Format of the DSCH Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data stored in the data table is 2-digit BCD.
   BYT = 1: The data stored in the data table is 4-digit BCD.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the DSCH Command. No change in W1.
   ACT = 1: Execution of the DSCH Command. After executing, if the search data is found, then it outputs the table internal number where the data is stored. However, if the search data is not found, W1 = 1.

Parameters

1) Number of Rows
   This address holds the number of rows of the data table (table capacity). There must be enough valid memory for the number of rows specified. If the first data table number is Number 0 and the last is number "n," "n+1" is specified as the data number (number of rows) of the data table.

2) Data Table Leading Address
   This is the starting address for a specified range of addresses that are to be used for the data.

3) Search Data Address
   This is the address where the data to be searched for is stored.

4) Search Result Output Address
   This is the address where the row number of the table that contains the data is output when the search is successful. This address needs to have memory corresponding to the specified size of the data (BYT).

Search Results (W1)

W1 = 0: Search data found.
W1 = 1: Search data not found.

9.2.26 DSCHB (Binary Data Search)

Function

This command searches the data inside the data table like the DSCH command (Section 9.2.25). This command differs in that the data is in binary format and the number of rows for the data table (table capacity) changes to an address that contains the data, so even after the sequence program is compiled into binary data, the table capacity can still change.

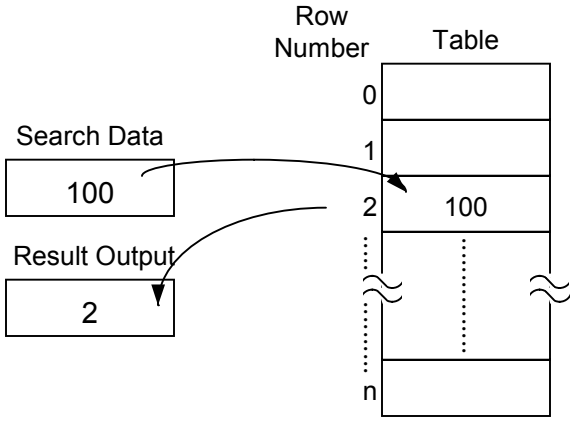


Figure 9-68: Function of the DSCHB Command

Format

Figure 9-69 shows the format for describing the command.

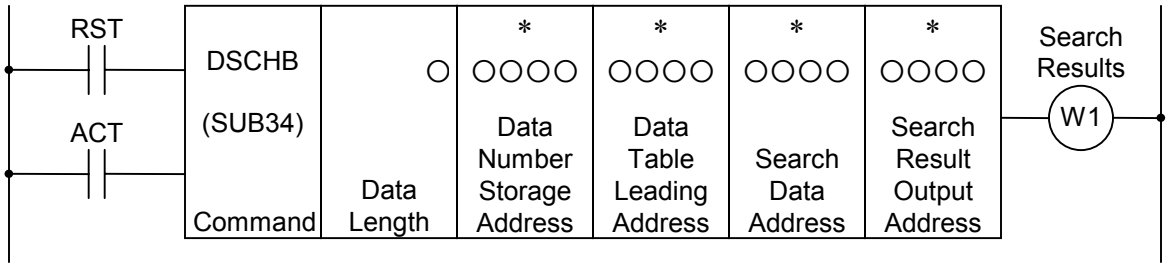


Figure 9-69: Format for the DSCHB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the DSCHB Command. No change in W1.
   ACT = 1: Execution of the DSCHB command. If the search data is found, then outputs the row number where the data is stored. W = 1 if not found.

Parameters

1) Data Length
   Specifies the byte length of the data in the first digit of the parameters.
   When 1: Data is 1-byte binary data.
   When 2: Data is 2-byte binary data.
   When 4: Data is 4-byte binary data.

2) Data Number Storage Address
   This address holds the number of rows in the data table (table capacity). There must be enough valid memory for the number of rows specified. If the first data table number is Number 0 and the last is number "n," "n+1" is specified as the data number (number of rows) of the data table.

3) Data Table Leading Address
   This is the starting address for is a specified range of addresses that can be used for the data.

4) Search Data Address
   This is the address where the data to be searched will be stored.

5) Search Result Output Address
   The row number of a successful search is put into the Search Result Output Address. It requires enough memory to hold the data specified.

Search Results (W1)

W1 = 0: Search data found.
W1 = 1: Search data not found.

9.2.27 XMOV (Index Modification Data Transfer)

Function

Similar to the DSCH command, this command operates on the data table; it reads to or writes from the data table.

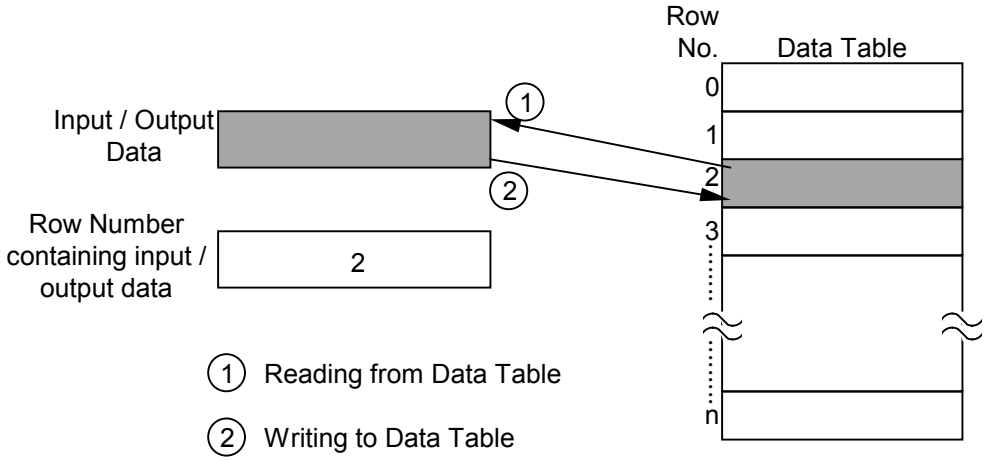


Figure 9-70: Reading from and Writing to the Data Table for the XMOV Command

CAUTION

The leading data table address selected in the XMOV command parameters becomes the 0^{th} table internal number (row number). This table internal number differs from the table internal number mentioned in *Section 9.3: PLC Data Table*.

Format

Figure 9-71 shows the format for describing the command, and Table 9-19 shows the coding format.

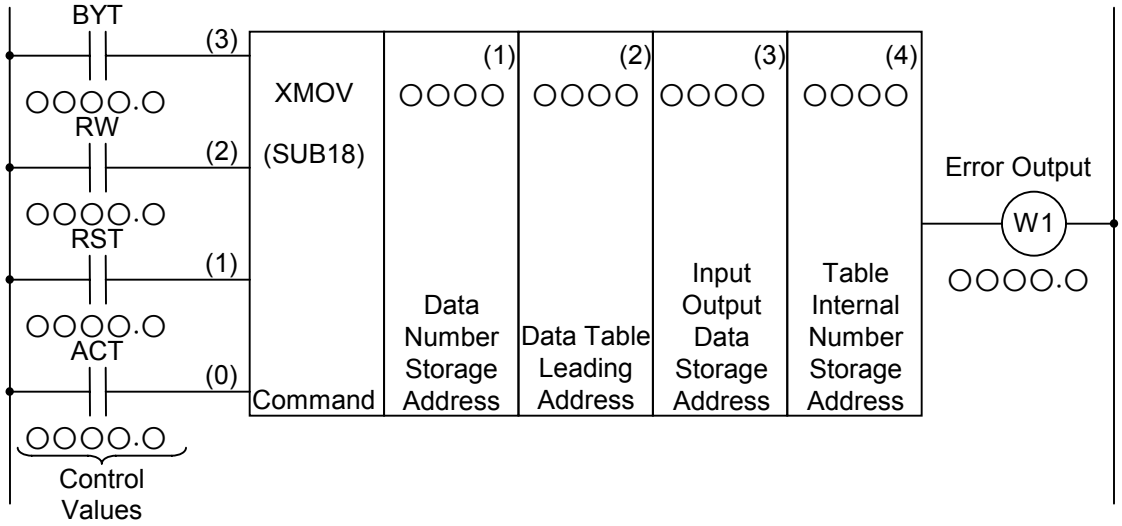


Figure 9-71: Format for the XMOV Command

Coding Sheet				Result History Register			
Step No.	Command	Address No. Bit No.	Description	ST3	ST2	ST1	ST0
1	RD	○○○ . ○	BYT				BYT
2	RD.STK	○○○ . ○	RW			BYT	RW
3	RD.STK	○○○ . ○	RST		BYT	RW	RST
4	RD.STK	○○○ . ○	ACT	BYT	RW	RST	ACT
5	SUB	18	XMOV Command	BYT	RW	RST	ACT
6	(PRM)	○○○○	Data Number Storage Address	BYT	RW	RST	ACT
7	(PRM)	○○○○	Data Table Leading Address	BYT	RW	RST	ACT
8	(PRM)	○○○○	Input Output Data Storage Address	BYT	RW	RST	ACT
9	(PRM)	○○○○	Table Internal Number Storage Address	BYT	RW	RST	ACT
10	WRT	○○○ . ○	W1, Error Output	BYT	RW	RST	W1

Table 9-19: Coding Format of the XMOV Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data stored in the data table is 2-digit BCD.
   BYT = 1: The data stored in the data table is 4-digit BCD.

2) Read or Write (RW)
   RW = 0: Reads the data from the data table.
   RW = 1: Writes new data into the data table.

3) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0".

4) Action Command (ACT)
   ACT = 0: No execution of the XMOV Command. No change in W1.
   ACT = 1: Execution of the XMOV command.

Parameters

1) Data Number Storage Address
   This address holds the number of rows of the data table (table capacity). There must be enough valid memory for the number of rows specified. If the first data table number is Number 0 and the last is number "n," "n+1" is specified as the data number of the data table.

2) Data Table Leading Address
   This is the starting address for a specified range of addresses that can be used for the data.

3) Input Output Data Storage Address
   This is the address where the data to read from or the address to write to is stored.

4) Table Internal Number Storage Address
   The Table Internal Number indexes the table to choose which data to access in the data table. The table internal number storage address is the address where this table internal number is stored, where the amount of memory specified in BYT is allocated.

Error Output (W1)

W1 = 0: No error.
W1 = 1: Error. (Accessing a table internal number that is too large for the data table will throw an error.)

9.2.28 XMOVB (Binary Index Modification Data Transfer)

Function

This command is same as the XMOV Command in Section 9.2.27 in that it reads from and writes to data in the data table. The only two differences are that all the numeric data used is in binary format and that the data number specification of the data table (table capacity) become address specified and even after the sequence program is converted into binary data, the table capacity is variable.

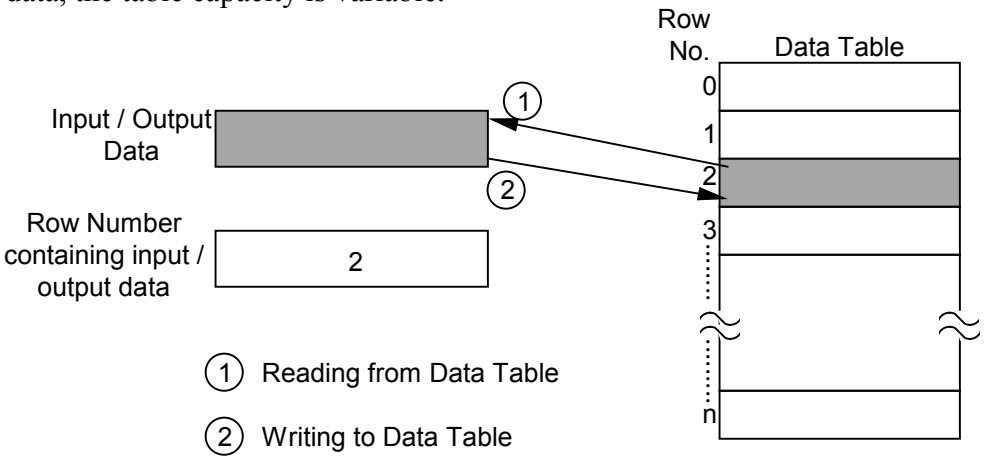


Figure 9-72: Reading from and Writing to the Data Table for the XMOVB Command

Format

Figure 9-73 shows the format for describing the command.

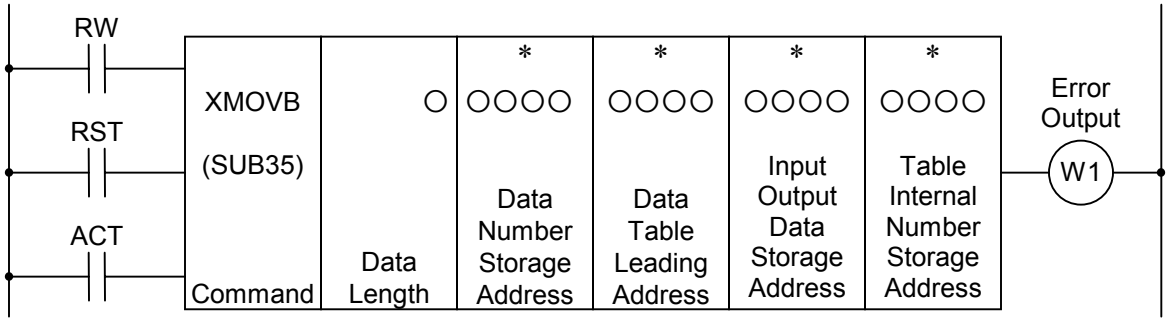


Figure 9-73: Format for the XMOVB Command

Control Values

1) Read or Write (RW)
   RW = 0: Reads data from the data table.
   RW = 1: Writes data into the data table.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the XMOV command. No change in W1.
   ACT = 1: Execution of the XMOV command.

Parameters

1) Data Length
   Specifies the byte length of the data in the first digit of the parameters.
   When 1: Data is 1-byte binary data.
   When 2: Data is 2-byte binary data.
   When 4: Data is 4-byte binary data.

2) Data Number Storage Address
   This address holds the number of rows of the data table. There must be enough valid memory for the number of rows specified. If the first data table number is Number 0 and the last is number "n," "n+1" is specified as the data number of the data table.

3) Data Table Leading Address
   This is the starting address for a specified range of addresses that can be used for the data.

4) Input Output Data Storage Address
   This is the address where the data from reading from and writing to the data table is stored.

5) Table Internal Number Storage Address
   The Table Internal Number indexes the table to choose which data to access in the data table. The table internal number storage address is the address where this table internal number is stored, where the amount of memory specified in BYT is allocated.

Error Output (W1)

W1 = 0: No error.
W1 = 1: Error. An error will show if a table internal number that is larger than the number of rows in the table is specified.

9.2.29 ADD (Addition)

Function

This command adds two 2- or 4-digit BCD numbers.

Format

Figure 9-74 shows the format for describing the command, and Table 9-20 shows the coding format.

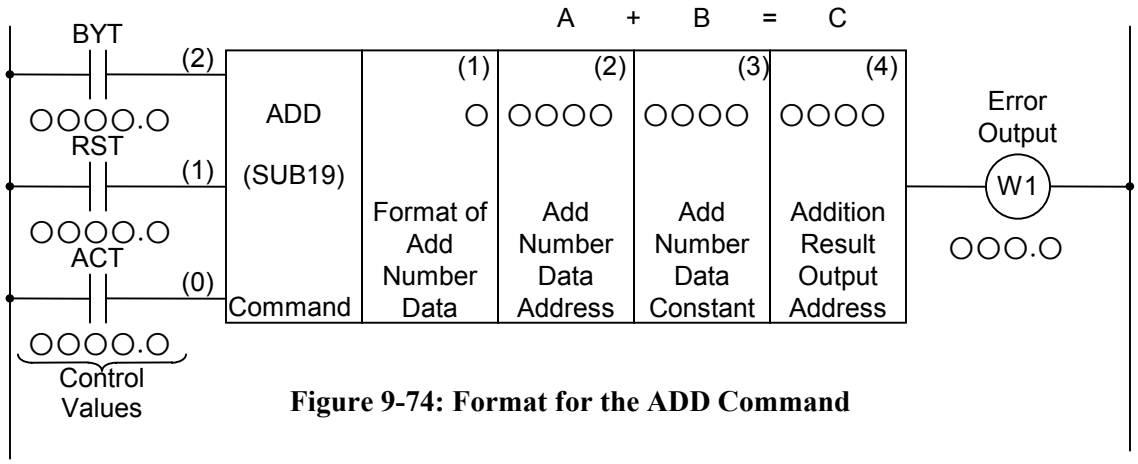


Figure 9-74: Format for the ADD Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	○○○	. ○	BYT
2	RD.STK	○○○	. ○	RST
3	RD.STK	○○○	. ○	ACT
4	SUB	19		ADD Command
5	(PRM)	○		Format of Add Number Data
6	(PRM)	○○○○		Add Number Data Address
7	(PRM)	○○○○		Add Number Data Constant
8	(PRM)	○○○○		Addition Result Output Address
9	WRT	○○○	. ○	W1, Error Output

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	RST
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	ACT
	BYT	RST	W1

Table 9-20: Coding Format of the ADD Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data processed is 2-digit BCD.
   BYT = 1: The data processed is 4-digit BCD.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the ADD command. No change in W1.
   ACT = 1: Execution of the ADD command.

Parameters

1) Format of Add Number Data
   0: Specifies add data as a constant.
   1: Specifies add data as an address.

2) Add Number Data Address
   This is the address where the add number data is stored.

3) Add Number Data Constant
   The format for the specification of the add number data is determined by the "Format of Add Number Data" parameter.

4) Addition Result Output Address
   The address where the addition result is output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the addition result is larger than the data size specified by the BYT control value.)

9.2.30 ADDB (Binary Addition)

Function

This command adds binary data of 1, 2, or 4 bytes. The numeric data from the calculation result and other calculation information is set inside the calculation result register (R9000). The add number data or the addition result output data needs corresponding bytes of memory.

Format

Figure 9-75 shows the format for describing the command.

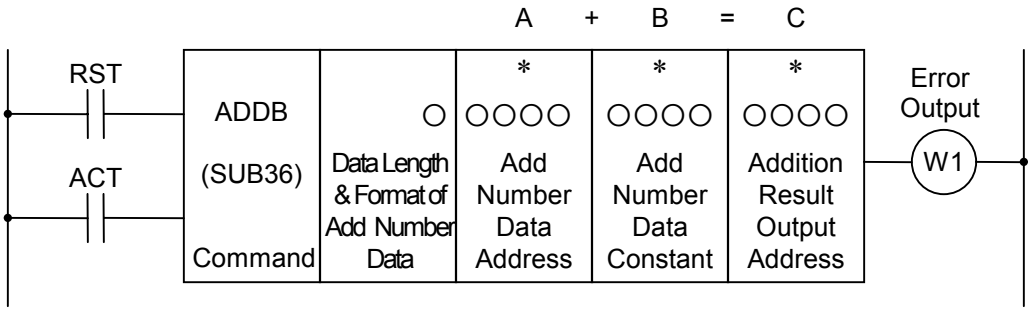


Figure 9-75: Format for the ADDB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the ADDB command. No change in W1.
   ACT = 1: Execution of the ADDB command.

Parameters

1) Data Length and Format of Add Number Data
   The data size (1, 2, or 4 bytes) and the format of the add number data (constant data or address data).

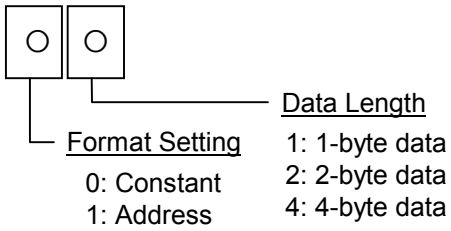


Figure 9-76: Parameters Format Specification for the ADDB Command

2) Add Number Data Address
Address where the add number data is stored.

3) Add Number Data Constant
The format of the add number data is determined by the specifications shown in Figure 9-76.

4) Addition Result Output Address
The address where the addition result is output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the addition result is larger than the data size specified in the "Data Length and Format of Add Number Data" parameter.)

Calculation Result Register (R9000)

After calculation, a "1" in the following bit positions signifies the following:

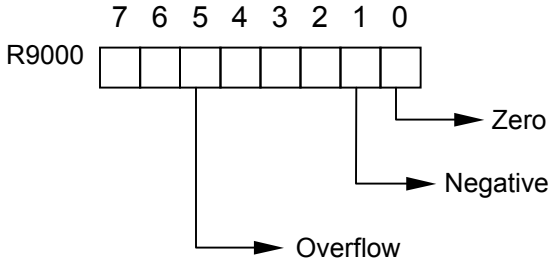


Figure 9-77: Calculation Result Register for the ADDB Command

9.2.31 SUB (Subtraction)

Function

This command subtracts two 2- or 4-digit BCD numbers.

Format

Figure 9-78 shows the format for describing the command, and Table 9-21 shows the coding format.

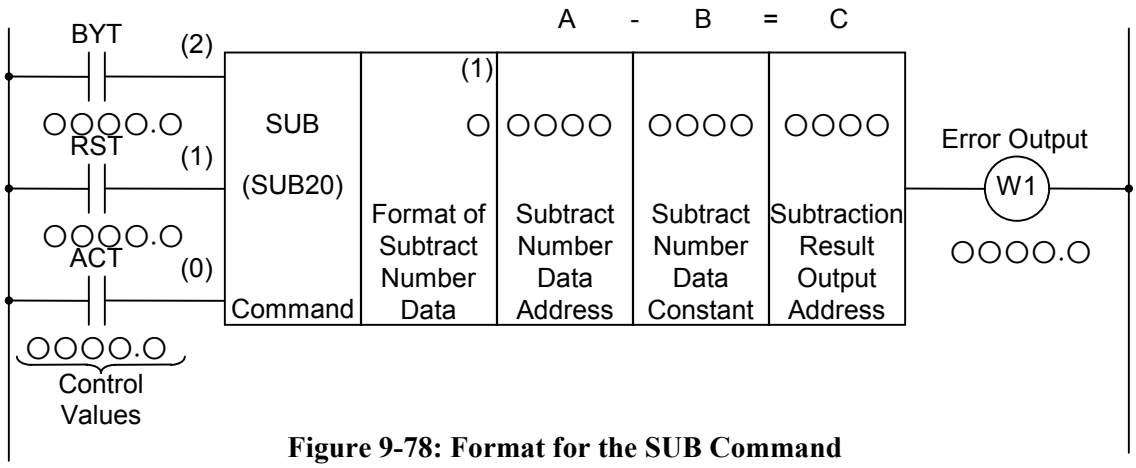


Figure 9-78: Format for the SUB Command

Coding Sheet				Result History Register			
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**
1	RD	○○○ . ○	BYT				BYT
2	RD.STK	○○○ . ○	RST			BYT	RST
3	RD.STK	○○○ . ○	ACT		BYT	RST	ACT
4	SUB	20	SUB Command		BYT	RST	ACT
5	(PRM)	○	Format of Subtract Number Data		BYT	RST	ACT
6	(PRM)	○○○○	Subtract Number Data Address		BYT	RST	ACT
7	(PRM)	○○○○	Subtract Number Data Constant		BYT	RST	ACT
8	(PRM)	○○○○	Subtraction Result Output Address		BYT	RST	ACT
9	WRT	○○○ . ○	W1, Error Output		BYT	RST	W1

Table 9-21: Coding Format of the SUB Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data processed is 2-digit BCD.
   BYT = 1: The data processed is 4-digit BCD.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the SUB command. No change in W1.
   ACT = 1: Execution of the SUB command.

Parameters

1) Format of Subtract Number Data
   0: Specifies subtract number data as a constant.
   1: Specifies subtract number data as an address.

2) Subtract Number Data Address
   The address where the subtract number is stored.

3) Subtract Number Data Constant
   Format specification of the subtract number data is determined by the "Format of Subtract Number Data" parameter.

4) Subtraction Result Output Address
   The address where the subtraction result is output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the subtraction result is negative.)

9.2.32 SUBB (Binary Subtraction)

Function

This command carries out subtraction of 1-, 2-, or 4-byte size binary format data. The numeric data from the calculation result and other calculation information is set inside the calculation result register (R9000). The subtract number data or the subtraction output data needs corresponding bytes of memory.

Format

Figure 9-79 shows the format for describing the command.

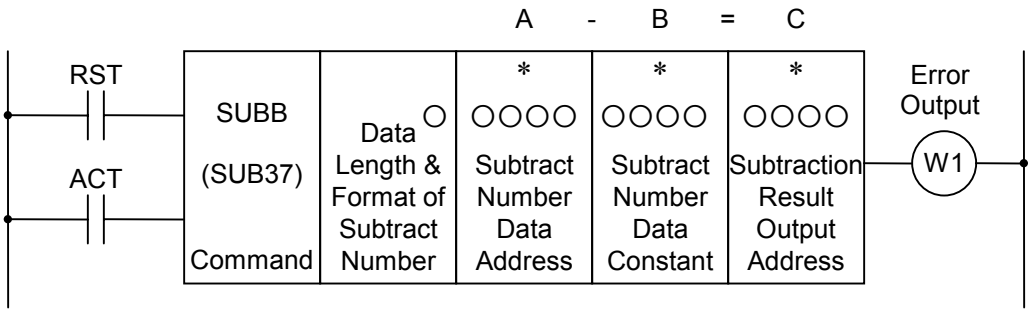


Figure 9-79: Format for the SUBB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the SUBB command. No change in W1.
   ACT = 1: Execution of the SUBB command.

Parameters

1) Data Length and Format of Subtract Number Data
   The data size (1, 2, or 4 bytes) and format of the subtract number data (constant data or address data).

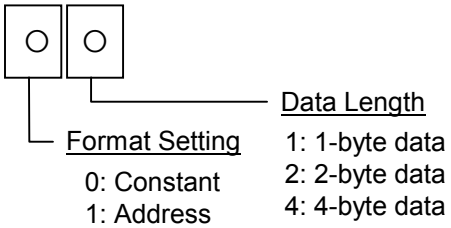


Figure 9-80: Parameters Format Specification for the SUBB Command

2) Subtract Number Data Address
The address where the subtract number data is stored.

3) Subtract Number Data Constant
The format of the subtract number data is determined by the specifications shown in Figure 9-80.

4) Subtraction Result Output Address
The address where the subtraction result is to be output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the subtraction result is larger than the specified data size.)

Calculation Result Register (R9000)

After calculations, a "1" in the following bit positions signifies the following:

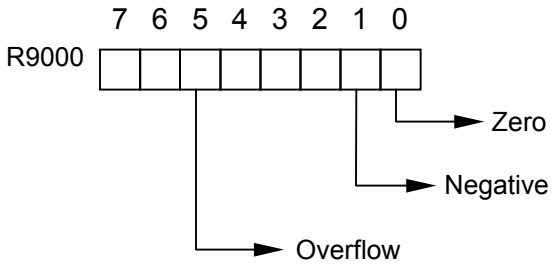


Figure 9-81: Calculation Result Register for the SUBB Command

9.2.33 MUL (Multiplication)

Function

This command carries out multiplication of 2- or 4-digit BCD, but the calculation results have to be within 2- or 4-digit BCD as well.

Format

Figure 9-82 shows the format for describing the command, and Table 9-22 shows the coding format.

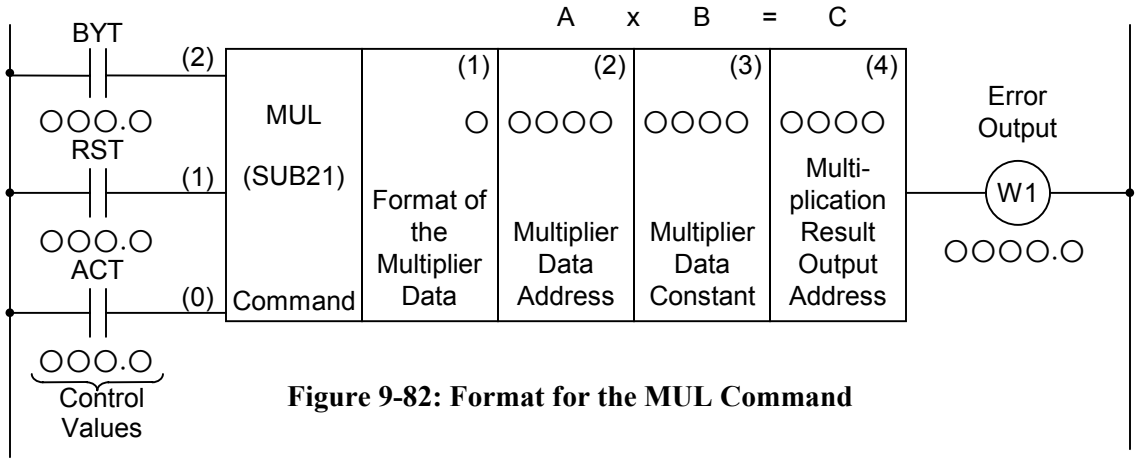


Figure 9-82: Format for the MUL Command

Coding Sheet				Result History Register			
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**
1	RD	○○○ . ○	BYT				BYT
2	RD.STK	○○○ . ○	RST			BYT	RST
3	RD.STK	○○○ . ○	ACT		BYT	RST	ACT
4	SUB	21	MUL Command		BYT	RST	ACT
5	(PRM)	○	Format of Multiplier Data		BYT	RST	ACT
6	(PRM)	○○○○	Multiplier Data Address		BYT	RST	ACT
7	(PRM)	○○○○	Multiplier Data Constant		BYT	RST	ACT
8	(PRM)	○○○○	Multiplication Result Output Address		BYT	RST	ACT
9	WRT	○○○ . ○	W1, Error Output		BYT	RST	W1

Table 9-22: Coding Format of the MUL Command

Control Values

1) Data Size (BYT)
   BYT = 0: The data processed is 2-digit BCD.
   BYT = 1: The data processed is 4-digit BCD.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the MUL command. No change in W1.
   ACT = 1: Execution of the MUL command.

Parameters

1) Format of Multiplier Data
   0: Specifies multiplier data as a constant.
   1: Specifies multiplier data as an address.

2) Multiplier Data Address
The address where the multiplier is stored.

3) Multiplier Data Constant
The specification format of the multiplier data is determined by the "Format of Multiplier Data" parameter.

4) Multiplication Result Output Address
The address where the multiplication result is to be output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the multiplication result exceeds the number specified by the BYT control value.)

9.2.34 MULB (Binary Multiplication)

Function

This command carries out multiplication of 1-, 2-, or 4-byte size binary format data. The numeric data from the calculation result and other calculation information is set inside the calculation result register (R9000). The multiplier data or the multiplication output data needs corresponding bytes of memory.

Format

Figure 9-83 shows the format for describing the command.

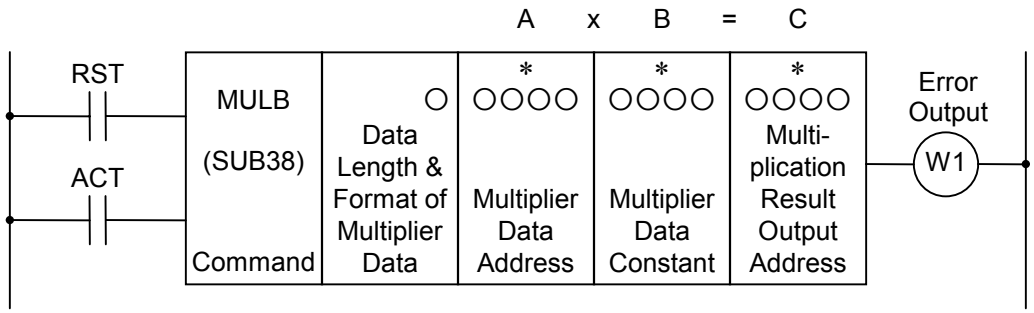


Figure 9-83: Format for the MULB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the MULB command. No change in W1.
   ACT = 1: Execution of the MULB command.

Parameters

1) Data Length and Format of Multiplier Data
The data size (1, 2, or 4 bytes) and the format of the multiplier data (constant data or address data).

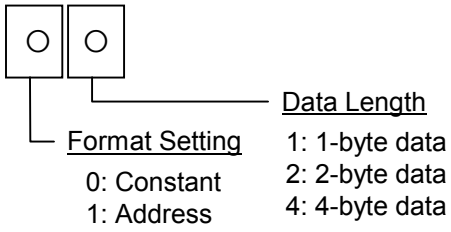


Figure 9-84: Parameters Format Specification for the MULB Command

2) Multiplier Data Address
The address where the multiplier data is stored.

3) Multiplier Data Constant
The format of the multiplier data is determined by the specifications shown in Figure 9-84.

4) Multiplication Result Output Address
The address where the multiplication result is to be output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the multiplication result exceeds the data size specified.)

Calculation Result Register (R9000)

After calculations, a "1" in the bit positions signifies the following:

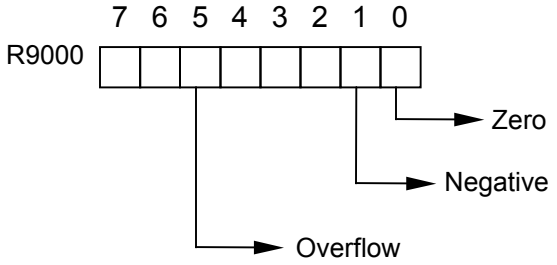


Figure 9-85: Calculation Result Register for the MULB Command

9.2.35 DIV (Division)

Function

This command carries out the division of 2- or 4-digit BCD. However, all decimals are disregarded in the calculation results.

Format

Figure 9-86 shows the format for describing the command, and Table 9-23 shows the coding format.

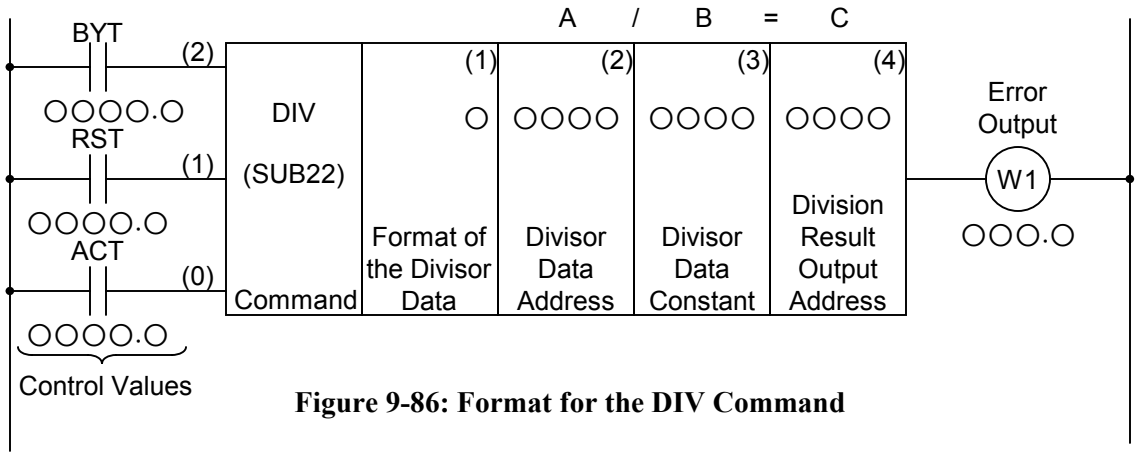


Figure 9-86: Format for the DIV Command

Coding Sheet				Result History Register			
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**
1	RD	○○○ . ○	BYT				BYT
2	RD.STK	○○○ . ○	RST			BYT	RST
3	RD.STK	○○○ . ○	ACT		BYT	RST	ACT
4	SUB	22	DIV Command		BYT	RST	ACT
5	(PRM)	○	Format of Divisor Data		BYT	RST	ACT
6	(PRM)	○○○○	Divisor Data Address		BYT	RST	ACT
7	(PRM)	○○○○	Divisor Data Constant		BYT	RST	ACT
8	(PRM)	○○○○	Division Result Output Address		BYT	RST	ACT
9	WRT	○○○ . ○	W1, Error Output		BYT	RST	W1

Table 9-23: Coding Format of the DIV Command

Control Values

1) Data Size (BYT)
   BYT = 0: The processed data is 2-digit BCD.
   BYT = 1: The processed data is 4-digit BCD.

2) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

3) Action Command (ACT)
   ACT = 0: No execution of the DIV command. No change in W1.
   ACT = 1: Execution of the DIV command.

Parameters

1) Format of the Divisor Data
   0: Specifies divisor data as a constant.
   1: Specifies divisor data as an address.

2) Divisor Data Address
   The address where the divisor is stored.

3) Divisor Data Constant
   The specification format of the divisor data is determined by the "Format of the Divisor Data" parameter.

4) Division Result Output Address
   The address where the division result is to be output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the divisor data is "0".)

9.2.36 DIVB (Binary Division)

Function

This command carries out division of 1-, 2-, or 4-byte size binary format data. The numeric data from the calculation result and other calculation information is set inside the calculation result register (R9000), and the remainder is set inside R9002. The divisor data and the division output (quotient) data need corresponding bytes of memory.

Format

Figure 9-87 shows the format for describing the command.

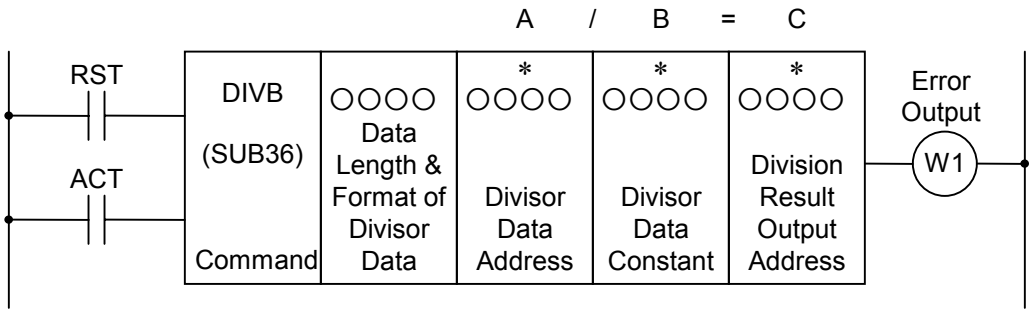


Figure 9-87: Format for the DIVB Command

Control Values

1) Reset (RST)
   RST = 0: No reset.
   RST = 1: Resets. In other words, W1 becomes "0."

2) Action Command (ACT)
   ACT = 0: No execution of the DIVB command. No change in W1.
   ACT = 1: Execution of the DIVB command.

Parameters

1) Data Length and Format of Divisor Data
   The data length (1, 2, or 4 bytes) and the format of the divisor data (constant data or address data).

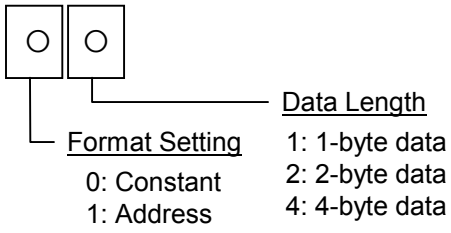


Figure 9-88: Parameters Format Specification for the DIVB Command

2) Divisor Data Address
The address where the divisor data is to be stored.

3) Divisor Data Constant
The format of the divisor data is determined by the specifications shown in Figure 9-88.

4) Division Result (Quotient) Output Address
The address where the division result data is to be output.

Error Output (W1)

W1 = 0: No error (normal calculation).
W1 = 1: Error (abnormal calculation). (W1=1 when the divisor data is "0," because you can't divide any number by zero.)

Calculation Result Register (R9000)

After calculations, a "1" in the bits signifies the following:

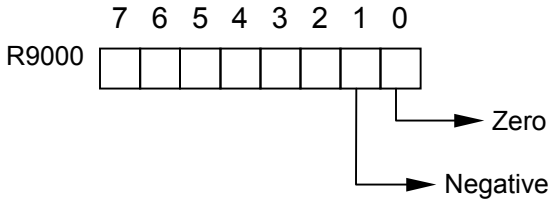


Figure 9-89: Calculation Result Register for the DIVB Command

Remainder Data Output Address

The remainder data, depending on its data size, gets output into R9002~R9005.

9.2.37 NUME (Constant Declaration)

Function

When using a function command, there are cases when constants are necessary. In those cases, this command may be used to declare 2- or 4-digit BCD constant data.

Format

Figure 9-90 shows the format for describing the command, and Table 9-24 shows the coding format.

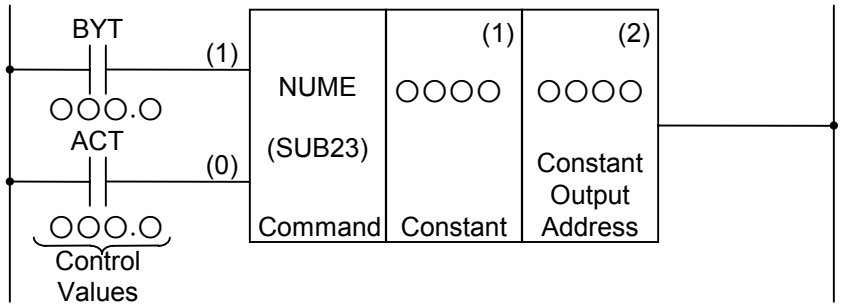


Figure 9-90: Format for the NUME Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOO	O	BYT
2	RD.STK	OOO	O	ACT
3	SUB	23		NUME Command
4	(PRM)	OOOO		Constant
5	(PRM)	OOOO		Constant Output Address

Result History Register

ST3	ST2	ST1	ST0
			BYT
		BYT	ACT
		BYT	ACT
		BYT	ACT
		BYT	ACT

Table 9-24: Coding Format of the NUME Command

Control Values

1) Data Size (BYT)
   BYT = 0: The constant is 2-digit BCD.
   BYT = 1: The constant is 4-digit BCD.

2) Action Command (ACT)
   ACT = 0: No execution of the NUME command.
   ACT = 1: Execution of the NUME command.

Parameters

1) Constant
   The constants are specified with the data size specified by the BYT control value.

2) Constant Output Address
   The address where the constant declared in the “Constant” parameter is output.

9.2.38 NUMEB (Binary Constant Declaration)

Function

This command declares 1-, 2-, or 4-byte size binary constant data. When programming, if data is input in base 10 format, after the execution of the program, it changes into binary format and the binary style constant from a specified address is stored into the byte size memory.

Format

Figure 9-91 shows the format for describing the command.

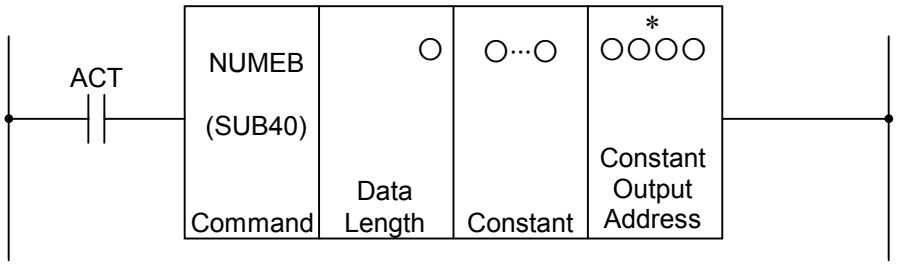


Figure 9-91: Format for the NUMEB Command

Control Values

Action Command (ACT)

ACT = 0: No execution of the NUMEB command.
ACT = 1: Execution of the NUMEB command.

Parameters

1) Data Length
   Specifies the byte length of the data in the first digit in the parameters.
   When 1: Data is 1-byte binary data.
   When 2: Data is 2-byte binary data.
   When 4: Data is 4-byte binary data.

2) Constant
   The constant data is declared in base 10. In this case, the data has to be constant data that can be stored within the capacity specified in the format specification declared in the "Data Length" parameter.

3) Constant Output Address
   The address of the binary format constant data. The memory for the specified data length needs to be contiguous.

Appendix A: S-100T Data Mapping Tables

Overview of Mapping Tables

Each physical input and output device has a specific location associated with it. The binary value ("0" or "1") of that memory location corresponds to the logical state or value of the device.

The following mapping tables should be used in writing the sequence program for your machine tool. For instance, as part of your program, you may want to check to see if a "spindle stop" command has been issued through the ServoWorks Motion Engine. To do so, you would check the F Data Mapping Tables, and find that the "Spindle STOP" signal (called MSPLSTP) is at address F018.1. To read this signal, you might use the command "RD F018.1."

Every possible command or signal either to or from the PLC Engine has a designed signal and address, as follows:

- F Data: Signals from the CNC (ServoWorks Motion Engine) to the PLC Engine.
- G Data: Signals from the PLC Engine to the CNC (ServoWorks Motion Engine).
- X Data: Signals from the machine tool to the PLC Engine.
- Y Data: Signals to the machine tool from the PLC Engine.

F Data Mapping Table

ADDRESS	NAME	DESCRIPTION
F000.0		
F000.1		
F000.2		
F000.3		
F000.4	SPL	Cycle Stop
F000.5	STL	Cycle Start
F000.6	SA	Servo Ready
F000.7	OP	Automatic Operation
F001.0	AL	CNC Alarm
F001.1	RST	Control Reset
F001.2		
F001.3	DEN	Distribution Done
F001.4	ENB	Spindle Enabled
F001.5	TAP	Tapping
F001.6		
F001.7	MA	CNC Ready
F002.0	INCH	Inch Input
F002.1	RPD	Rapid Traverse
F002.2	CSS	Constant Surface Speed
F002.3	THRD	Thread Cutting
F002.4		
F002.5		
F002.6	CUT	Cutting
F002.7	MDRN	Dry Run
F003.0	MINC	Incremental Jog Mode
F003.1	MH	Handwheel Mode
F003.2	MJ	Continuous Jog Mode
F003.3	MMDI	MDI Mode
F003.4		
F003.5	MMEM	Auto Mode

ADDRESS	NAME	DESCRIPTION
F003.6	MEDIT	Edit Mode
F003.7		
F004.0	MBDT1	Optional Block Skip
F004.1	MMLK	All Axes Machine Lock
F004.2	MABSM	Manual Absolute
F004.3	MSBK	Single Block
F004.4	MAFL	M/S/T/B Lock
F004.5	MREF	Reference Position Return Mode
F004.6		
F004.7	MOPSTP	Optional Stop
F006.0	MRP	Rapid Positioning Mode
F006.1	MSPDL	Spindle Mode
F006.2		
F006.3		
F006.4		
F006.5		
F006.6		
F006.7		
F007.0	MF	M Strobe
F007.1		
F007.2	SF	S Strobe
F007.3	TF	T Strobe
F007.4	BF	B Strobe
F007.5		
F007.6		
F007.7		
F009.0		
F009.1		
F009.2		
F009.3		
F009.4	DM30	M30
F009.5	DM02	M02
F009.6	DM01	M01

ADDRESS	NAME	DESCRIPTION
F009.7	DM00	M00
F010.0	M01	M Code
F010.1	M02	M Code
F010.2	M04	M Code
F010.3	M08	M Code
F010.4	M10	M Code
F010.5	M20	M Code
F010.6	M40	M Code
F010.7	M80	M Code
F017.0		
F017.1		
F017.2		
F017.3		
F017.4		
F017.5	SPARL	Spindle Speed Arrival
F017.6	SPZERO	Spindle Zero Speed (Actual)
F017.7	MSPLPOS	Spindle Position Control Mode
F018.0	MSPLCW	Spindle CW
F018.1	MSPLSTP	Spindle STOP
F018.2	MSPLCCW	Spindle CCW
F018.3		
F018.4		
F018.5		
F018.6		
F018.7		
F019.0		
F019.1		
F019.2	MHNDLIN	HandWheel Interrupt Mode
F019.3		
F019.4		
F019.5		
F019.6		
F019.7		

ADDRESS	NAME	DESCRIPTION
F020.0	MILK1P	Interlock 1+
F020.1		
F020.2	MILK3P	Interlock 3+
F020.3		
F020.4		
F020.5		
F020.6		
F020.7		
F021.0	MILK1M	Interlock 1–
F021.1		
F021.2	MILK3M	Interlock 3–
F021.3		
F021.4		
F021.5		
F021.6		
F021.7		
F022.0	S00001	Spindle Speed Code (RPM)
F022.1	S00002	Spindle Speed Code (RPM)
F022.2	S00004	Spindle Speed Code (RPM)
F022.3	S00008	Spindle Speed Code (RPM)
F022.4	S00010	Spindle Speed Code (RPM)
F022.5	S00020	Spindle Speed Code (RPM)
F022.6	S00040	Spindle Speed Code (RPM)
F022.7	S00080	Spindle Speed Code (RPM)
F023.0	S00100	Spindle Speed Code (RPM)
F023.1	S00200	Spindle Speed Code (RPM)
F023.2	S00400	Spindle Speed Code (RPM)
F023.3	S00800	Spindle Speed Code (RPM)
F023.4	S01000	Spindle Speed Code (RPM)
F023.5	S02000	Spindle Speed Code (RPM)
F023.6	S04000	Spindle Speed Code (RPM)
F023.7	S08000	Spindle Speed Code (RPM)
F24.0	S10000	Spindle Speed Code (RPM)

ADDRESS	NAME	DESCRIPTION
F24.1	S20000	Spindle Speed Code (RPM)
F24.2	S40000	Spindle Speed Code (RPM)
F24.3	S80000	Spindle Speed Code (RPM)
F24.4		
F24.5		
F24.6		
F24.7		
F026.0	T01	Tool Number Code
F026.1	T02	Tool Number Code
F026.2	T04	Tool Number Code
F026.3	T08	Tool Number Code
F026.4	T10	Tool Number Code
F026.5	T20	Tool Number Code
F026.6	T40	Tool Number Code
F026.7	T80	Tool Number Code
F030.0	B01	B Code
F030.1	B02	B Code
F030.2	B04	B Code
F030.3	B08	B Code
F030.4	B10	B Code
F030.5	B20	B Code
F030.6	B40	B Code
F030.7	B80	B Code
F040.0	AR00001	Actual Spindle Speed (RPM)
F040.1	AR00002	Actual Spindle Speed (RPM)
F040.2	AR00004	Actual Spindle Speed (RPM)
F040.3	AR00008	Actual Spindle Speed (RPM)
F040.4	AR00010	Actual Spindle Speed (RPM)
F040.5	AR00020	Actual Spindle Speed (RPM)
F040.6	AR00040	Actual Spindle Speed (RPM)
F040.7	AR00080	Actual Spindle Speed (RPM)
F041.0	AR00100	Actual Spindle Speed (RPM)
F041.1	AR00200	Actual Spindle Speed (RPM)

ADDRESS	NAME	DESCRIPTION
F041.2	AR00400	Actual Spindle Speed (RPM)
F041.3	AR00800	Actual Spindle Speed (RPM)
F041.4	AR01000	Actual Spindle Speed (RPM)
F041.5	AR02000	Actual Spindle Speed (RPM)
F041.6	AR04000	Actual Spindle Speed (RPM)
F041.7	AR08000	Actual Spindle Speed (RPM)
F042.0	AR10000	Actual Spindle Speed (RPM)
F042.1	AR20000	Actual Spindle Speed (RPM)
F042.2	AR40000	Actual Spindle Speed (RPM)
F042.3	AR80000	Actual Spindle Speed (RPM)
F042.4		
F042.5		
F042.6		
F042.7		
F068.0	DWELL	G04
F068.1	FPM	Feed Per Minute
F068.2	TNRP	Tool Nose Radius Compensation
F068.3		
F068.4		
F068.5		
F068.6		
F068.7		
F072.0	MFO_H	MFO Rotary Switch
F072.1	SSO_H	SSO Rotary Switch
F072.2	RPO_H	RPO Rotary Switch
F072.3		
F072.4		
F072.5		
F072.6		
F072.7		
F085.0		
F085.1		
F085.2		

ADDRESS	NAME	DESCRIPTION
F085.3		
F085.4	MP1	HandWheel Multiple Selection
F085.5	MP2	HandWheel Multiple Selection
F085.6		
F085.7		
F086.0	HS1A	HandWheel Axis 1
F086.1		
F086.2	HS1C	HandWheel Axis 3
F086.3		
F086.4		
F086.5		
F086.6		
F086.7		
F094.0	ZP1	At 1st Reference Point 1
F094.1		
F094.2	ZP3	At 1st Reference Point 3
F094.3	ZP4	At 1st Reference Point 4 (C-Axis)
F094.4		
F094.5		
F094.6		
F094.7		
F096.0	ZP21	At 2nd Reference Point 1
F096.1		
F096.2	ZP23	At 2nd Reference Point 3
F096.3		
F096.4		
F096.5		
F096.6		
F096.7		
F098.0	ZP31	At 3rd Reference Point 1
F098.1		
F098.2	ZP33	At 3rd Reference Point 3
F098.3		

ADDRESS	NAME	DESCRIPTION
F098.4		
F098.5		
F098.6		
F098.7		
F100.0	ZP41	At 4th Reference Point 1
F100.1		
F100.2	ZP43	At 4th Reference Point 3
F100.3		
F100.4		
F100.5		
F100.6		
F100.7		
F102.0	MV1	Axis In Motion 1
F102.1		
F102.2	MV3	Axis In Motion 3
F102.3	MV4	Axis In Motion 4 (C-Axis)
F102.4		
F102.5		
F102.6		
F102.7		
F104.0	INP1	In Position 1
F104.1		
F104.2	INP3	In Position 3
F104.3	INP4	In Position 4 (C-Axis)
F104.4		
F104.5		
F104.6		
F104.7		
F106.0	MVD1	Axis Motion In Negative Direction 1
F106.1		
F106.2	MVD3	Axis Motion In Negative Direction 3
F106.3	MVD4	Axis Motion In Negative Direction 4 (C-Axis)
F106.4		

ADDRESS	NAME	DESCRIPTION
F106.5		
F106.6		
F106.7		
F120.0	ZRF1	1st Reference Point Established 1
F120.1		
F120.2	ZRF3	1st Reference Point Established 3
F120.3	ZRF4	1st Reference Point Established 4 (C-Axis)
F120.4		
F120.5		
F120.6		
F120.7		
F124.0	SL_1P	Soft Limit 1+
F124.1		
F124.2	SL_3P	Soft Limit 3+
F124.3	SL_4P	Soft Limit 4+ (C-Axis)
F124.4		
F124.5		
F124.6		
F124.7		
F125.0	SL_1M	Soft Limit 1–
F125.1		
F125.2	SL_3M	Soft Limit 3–
F125.3	SL_4M	Soft Limit 4– (C-Axis)
F125.4		
F125.5		
F125.6		
F125.7		
F126.0	HL_1P	Hard Limit 1+
F126.1		
F126.2	HL_3P	Hard Limit 3+
F126.3		
F126.4		
F126.5		

ADDRESS	NAME	DESCRIPTION
F126.6		
F126.7		
F127.0	HL_1M	Hard Limit 1–
F127.1		
F127.2	HL_3M	Hard Limit 3–
F127.3		
F127.4		
F127.5		
F127.6		
F127.7		
F128.0	SVAL_1	Servo Alarm 1
F128.1		
F128.2	SVAL_3	Servo Alarm 3
F128.3	SVAL_4	Servo Alarm 4 (C-Axis)
F128.4		
F128.5		
F128.6		
F128.7		
F129.0	OVPOSER1	Over Position Error 1
F129.1		
F129.2	OVPOSER3	Over Position Error 3
F129.3	OVPOSER4	Over Position Error 4 (C-Axis)
F129.4		
F129.5		
F129.6		
F129.7		

G Data Mapping Table

ADDRESS	NAME	DESCRIPTION
G004.3	FIN	Done Signal
G005.0	MFIN	M Done
G005.2	SFIN	S Done
G005.3	TFIN	T Done
G005.4	BFIN	B Done
G005.6	AFL	M/S/T/B Lock
G006.2	*ABSM	Manual Absolute
G006.4	OVC	Feed Override Cancel
G007.2	ST	Cycle Start
G008.0	*IT	All Axes Interlock
G008.4	*ESP	Emergency Stop
G008.5	*SP	Cycle Stop
G008.6	RRW	Reset NC and Rewind Program
G008.7	ERS	External Reset
G020.0	MFO_H01	MFO Value Selection
G020.1	MFO_H02	MFO Value Selection
G020.2	MFO_H04	MFO Value Selection
G020.3	MFO_H08	MFO Value Selection
G020.4	MFO_H16	MFO Value Selection
G021.0	SSO_H01	SSO Value Selection
G021.1	SSO_H02	SSO Value Selection
G021.2	SSO_H04	SSO Value Selection
G022.0	RPO_H01	RPO Value Selection
G022.1	RPO_H02	RPO Value Selection
G043.0	MD1	NC Mode Selection
G043.1	MD2	NC Mode Selection
G043.2	MD4	NC Mode Selection
G043.3	MD8	NC Mode Selection
G044.0	BDT1	Optional Block Skip
G046.1	SBK	Single Block

ADDRESS	NAME	DESCRIPTION
G046.2	OPSTP	Optional Stop
G046.7	DRN	Dry Run
G100.0	+J1	Axis Selection 1+
G100.2	+J3	Axis Selection 3+
G102.0	–J1	Axis Selection 1–
G102.2	–J3	Axis Selection 3–
G105.0	JCS_1	Continuous Jog Speed Selection
G105.1	JCS_2	Continuous Jog Speed Selection
G105.2	JIA_1	Incremental Jog Amount Selection
G105.3	JIA_2	Incremental Jog Amount Selection
G105.4	JIA_4	Incremental Jog Amount Selection
G105.5	REF_SR	Set/Return Mode for Reference Point
G105.6	REF_1	Reference Point Selection
G105.7	REF_2	Reference Point Selection
G108.0	MLK1	Machine Lock Axis 1
G108.2	MLK3	Machine Lock Axis 3
G126.0	SVF1	Servo Control 1
G126.2	SVF3	Servo Control 3
G126.3	SVF4	Spindle Control
G130.0	*IT1	Interlock Axis 1
G130.2	*IT3	Interlock Axis 3
G132.0	MIT1P	Interlock Axis 1+
G132.2	MIT3P	Interlock Axis 3+
G134.0	MIT1M	Interlock Axis 1–
G134.2	MIT3M	Interlock Axis 3–
G141.0	M-TO	M Timeout
G141.1	S-TO	S Timeout
G141.2	T-TO	T Timeout
G141.3	B-TO	B Timeout

X Data Mapping Tables

HandWheel I/P (FP-60)

ADDRESS	NAME	DESCRIPTION
X00.0	HW_ESTP	
X00.1	HW_1	HandWheel Axis 1
X00.2	HW_2	
X00.3	HW_3	HandWheel Axis 3
X00.4	HW_4	
X00.7	HW_X1	
X01.0	HW_X10	
X01.1	HW_X100	

Home & Limit Switches (DC-120)

ADDRESS	NAME	DESCRIPTION
X12.1	HS_1	Home Switch 1
X12.2	LS_1M	Limit Switch 1–
X12.3	LS_1P	Limit Switch+
X13.1	HS_3	Home Switch 3
X13.2	LS_3M	Limit Switch 3–
X13.3	LS_3P	Limit Switch 3+

Appendix B: S-100M and General Motion Applications (MC-Quad, MotionPro and SWSDK) Data Mapping Tables

Overview of Mapping Tables

Each physical input and output device has a specific location associated with it. The binary value ("0" or "1") of that memory location corresponds to the logical state or value of the device.

The following mapping tables should be used in writing the sequence program for your machine tool. For instance, as part of your program, you may want to check to see if a "cycle stop" command has been issued through the ServoWorks Motion Engine. To do so, you would check the F Data Mapping Tables, and find that the "Cycle Stop" signal (called SPL) is at address F000.4. To read this signal, you might use the command "RD F000.4."

Every possible command or signal either to or from the PLC Engine has a designed signal and address, as follows:

- F Data: Signals from the ServoWorks Motion Engine to the PLC Engine.
- G Data: Signals to the ServoWorks Motion Engine from the PLC Engine.
- X Data: Signals from the machine tool to the PLC Engine (machine input).
- Y Data: Signals to the machine tool from the PLC Engine (machine output).

Tables B-1 through B-8 are located at the end of this appendix.

F Data Mapping Table

ADDRESS	NAME	DESCRIPTION	NOTE
F000.4	SPL	Cycle Stop	0: Program started or reset. 1: Program stopped (feed hold).
F000.5	STL	Cycle Start	0: Program stopped or reset. 1: Program started.
F000.6	SA	Servo Ready	0: Not all the used axes are enabled. 1: All the used axes are enabled.
F000.7	OP	Auto Mode Running	0: Reset. 1: Program started or stopped or ended.
F001.0	AL	CNC Alarm	0: CNC is not in (reset from) the Alarm state. 1: CNC is in the Alarm state.
F001.1	RST	Control Reset	0: CNC is not in the Reset state. 1: CNC is in the Reset state. When CNC has been reset, it will be in the Reset state for 1 second.
F001.3	DEN	Distribution Done	0: Block pulse distribution is not done. 1: Block pulse distribution is done.
F001.7	MA	CNC Ready	0: CNC is not ready (communication between host and remote devices is not on). 1: CNC is ready (communication between host and remote devices is on).
F002.5	OPSTP	Optional Stop	0: Optional stop signal is not on. 1: Optional stop signal is on.
F002.7	MDRN	Dry Run	0: Dry run mode is not on. 1: Dry run mode is on.
F003.0	MINC	Incremental Feed Select	Refer to Table B-1.
F003.1	MH	Manual Handwheel Mode	Refer to Table B-1.
F003.2	MJ	Jog Mode	Refer to Table B-1.
F003.3	MMDI	MDI Mode	Refer to Table B-1.
F003.4	MRMT	DNC Mode	Refer to Table B-1.
F003.5	MMEM	Auto Mode	Refer to Table B-1.
F004.0	MBDT1	Optional Block Skip	0: Optional block skip signal is not on. 1: Optional block skip signal is on.

ADDRESS	NAME	DESCRIPTION	NOTE
F004.2	MASBM	Manual Absolute Mode	0: Manual Absolute Mode is not on. 1: Manual Absolute Mode is on.
F004.3	MSBK	Single Block	0: Single block mode is not on. 1: Single block mode is on.
F004.5	MREF	Manual Reference Point Return	Refer to Table B-1.
F004.6	MREF1	Manual Reference Point Return	Refer to Table B-1.
F007.0	MF	M Strobe	0: Miscellaneous function strobe is not on. 1: Miscellaneous function strobe is on.
F007.2	SF	S Strobe	0: Spindle function strobe is not on. 1: Spindle function strobe is on.
F007.3	TF	T Strobe	0: Tool function strobe is not on. 1: Tool function strobe is on.
F009.4	DM30	M30 Decode	0: M30 function is not on. 1: M30 function is on.
F009.5	DM02	M02 Decode	0: M02 function is not on. 1: M02 function is on.
F009.6	DM01	M01 Decode	0: M01 function is not on. 1: M01 function is on.
F009.7	DM00	M00 Decode	0: M00 function is not on. 1: M00 function is on.
F010 – F013		M Code Signal	Miscellaneous function code in 4-byte binary format. F10 is the least significant byte. F13 is the most significant byte.
F20	MILK1p to MILK8p	Interlock 1+ to 8+	0: Axis plus direction interlock is off 1: Axis plus direction interlock is on.
F21	MILK1m to MILK8m	Interlock 1- to 8-	0: Axis minus direction interlock is off. 1: Axis minus direction interlock is on.
F85.4	MP1	Handwheel Multiple Selection 1	Refer to Table B-2.
F85.5	MP2	Handwheel Multiple Selection 2	Refer to Table B-2.
F86.0	HS1A	Handwheel Manual Axis Selection A	Refer to Table B-3.
F86.1	HS1B	Handwheel Manual Axis Selection B	Refer to Table B-3.
F86.2	HS1C	Handwheel Manual Axis Selection C	Refer to Table B-3.
F86.3	HS1D	Handwheel Manual Axis Selection D	Refer to Table B-3.
F88.0	HS1IA	Handwheel Interrupt Axis Selection A	Refer to Table B-3.
F88.1	HS1IB	Handwheel Interrupt Axis Selection B	Refer to Table B-3.

ADDRESS	NAME	DESCRIPTION	NOTE
F88.2	HS1IC	Handwheel Interrupt Axis Selection C	Refer to Table B-3.
F88.3	HS1ID	Handwheel Interrupt Axis Selection D	Refer to Table B-3.
F94	ZP1 to ZP8	Zero Reference Point Return 1 to 8	0: Axis is not at home (reference point). 1: Axis is at home (reference point).
F102	MV1 to MV8	In Motion 1 to 8	0: Axis is not in motion. 1: Axis is in motion.
F104	INP1 to INP8	In Position 1 to 8	0: Axis is not in position. 1: Axis is in position.
F106	MVD1 to MVD8	Motion In Negative Direction 1 to 8	0: Axis motion is not in negative direction. 1: Axis motion is in negative direction.
F120	ZRF1 to ZRF8	Zero Reference Point Finding 1 to 8	0: Axis home (reference point) has not been established. 1: Axis home (reference point) has been established.
F300 – F399	USRD1 to USRD100	User defined CNC status data addresses.	Typically used for custom user interface, etc.

G Data Mapping Table

ADDRESS	NAME	DESCRIPTION	NOTE
G4.3	FIN	Done Signal	0: Auxiliary (M/S/T) function not finished. 1: Auxiliary (M/S/T) function finished. Edge triggered.
G5.0	MFIN	M Done	0: Miscellaneous (M/S/T) function not finished. 1: Miscellaneous (M/S/T) function finished. Edge triggered.
G5.2	SFIN	S Done	0: Spindle (M/S/T) function not finished. 1: Spindle (M/S/T) function finished. Edge triggered.
G5.3	TFIN	T Done	0: Tool (M/S/T) function not finished. 1: Tool (M/S/T) function finished. Edge triggered.
G6.2	ABSM	Manual Absolute Mode	0: Manual Absolute Mode off. 1: Manual Absolute Mode on. Edge triggered.
G6.6	SKIPP	Skip Signal	G31 block is skipped when this signal value changes. Edge triggered.
G7.2	ST	Cycle Start	Set NC program cycle start on signal rising edge (0→1)
G8.4	*ESP	Emergency Stop	Set CNC Emergency Stop on signal falling edge, and keep CNC in the Emergency Stop state when the signal is 0 (can not be reset by software)
G8.5	*SP	Cycle Stop	Set NC program cycle stop on signal falling edge (1→0)
G8.6	RRW	CNC Reset	Reset CNC on signal rising edge (0→1)
G8.7	ERS	External Reset	Reset CNC on signal rising edge (0→1)
G10 ~ G11	*JV0 to *JV15	Manual Feedrate Override Bit 0 to 15	Refer to Table B-4.
G12	*FV0 to FV7	Feedrate Override Bit 0 to 7	Refer to Table B-5.
G14.0	ROV1	Rapid Override 1	Refer to Table B-6.
G14.1	ROV2	Rapid Override 2	Refer to Table B-6.
G18.0	HS1A	Handwheel Manual Axis Selection A	Refer to Table B-3.

ADDRESS	NAME	DESCRIPTION	NOTE
G18.1	HS1B	Handwheel Manual Axis Selection B	Refer to Table B-3.
G18.2	HS1C	Handwheel Manual Axis Selection C	Refer to Table B-3.
G18.3	HS1D	Handwheel Manual Axis Selection D	Refer to Table B-3.
G19.0	MP1	Handwheel Multiple Selection 1	Refer to Table B-2.
G19.1	MP2	Handwheel Multiple Selection 2	Refer to Table B-2.
G41.0	HS1IA	Handwheel Interrupt Axis Selection A	Refer to Table B-3.
G41.1	HS1IB	Handwheel Interrupt Axis Selection B	Refer to Table B-3.
G41.2	HS1IC	Handwheel Interrupt Axis Selection C	Refer to Table B-3.
G41.3	HS1ID	Handwheel Interrupt Axis Selection D	Refer to Table B-3.
G43.0	MD1	NC Mode Selection	Refer to Table B-7.
G43.1	MD2	NC Mode Selection	Refer to Table B-7.
G43.2	MD4	NC Mode Selection	Refer to Table B-7.
G43.3	DNC1	Set DNC Mode	Refer to Table B-7.
G43.4	ZRN	Zero Reference Point Return	Refer to Table B-7.
G43.5	RT	Rapid Move	Refer to Table B-7.
G44.0	BDT1	Optional Block Skip	0: Set optional block skip mode off. 1: Set optional block skip mode on. Edge triggered
G46.0	OPSTP	Optional Stop	0: Set optional stop mode off. 1: Set optional stop mode on. Edge triggered.
G46.1	SBK	Single Block	0: Set single block mode off. 1: Set single block mode on. Edge triggered.
G46.7	DRN	Dry Run	0: Set dry run mode off. 1: Set dry run mode on. Edge triggered.
G100	+J1 to +J8	Jog Axis 1+ to 8+	Refer to Table B-8.
G102	–J1 to –J8	Jog Axis 1– to 8–	Refer to Table B-8.
G108	MLK1 to MLK8	Machine Lock Axis 1 to 8	0: Axis 1 to 8 machine unlock. 1: Axis 1 to 8 machine lock. Edge triggered.
G126	SVF1 to SVF8	Servo Control Axis 1 to 8	0: Axis 1 to 8 servo off. 1: Axis 1 to 8 servo on. Edge triggered.

ADDRESS	NAME	DESCRIPTION	NOTE
G132	MIT1p to MIT8p	Interlock Axis 1+ to 8+	0: Axis 1 to 8 plus direction interlock off. 1: Axis 1 to 8 plus direction interlock on. Edge triggered.
G134	MIT1m to MIT8m	Interlock Axis 1– to 8–	0: Axis 1 to 8 minus direction interlock off. 1: Axis 1 to 8 minus direction interlock on. Edge triggered.

X Data Mapping Table

ADDRESS	NAME	DESCRIPTION	NOTE
X00.0	HW_ESTP	Handwheel E-Stop	0: HandWheel E-Stop button is active (pushed down). 1: HandWheel E-Stop button is not active (released).
X00.1	HW_X	Handwheel Axis X	0: HandWheel axis X is not selected. 1: HandWheel axis X is selected.
X00.2	HW_Y	Handwheel Axis Y	0: HandWheel axis Y is not selected. 1: HandWheel axis Y is selected.
X00.3	HW_Z	Handwheel Axis Z	0: HandWheel axis Z is not selected. 1: HandWheel axis Z is selected.
X00.4	HW_4	Handwheel Axis Selection 4	0: HandWheel axis 4 is not selected. 1: HandWheel axis 4 is selected.
X00.5	HW_5	Handwheel Axis Selection 5	0: HandWheel axis 5 is not selected. 1: HandWheel axis 5 is selected.
X00.6	HW_6	Handwheel Axis Selection 6	0: HandWheel axis 6 is not selected. 1: HandWheel axis 6 is selected.
X00.7	HW_X1	Handwheel Multiple X1	0: HandWheel multiple X1 is not selected. 1: HandWheel multiple X1 is selected.
X01.0	HW_X10	Handwheel Multiple X10	0: HandWheel multiple X10 is not selected. 1: HandWheel multiple X10 is selected.
X01.1	HW_X100	Handwheel Multiple X100	0: HandWheel multiple X100 is not selected. 1: HandWheel multiple X100 is selected.
X01.2	N/A	N/A	Reserved
X01.3	N/A	N/A	Reserved
X01.4	N/A	N/A	Reserved
X01.5	N/A	N/A	Reserved

ADDRESS	NAME	DESCRIPTION	NOTE
X01.6	N/A	N/A	Reserved
X01.7	N/A	N/A	Reserved
X02	Local_Din_Low	Local (FP Board) Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X03	Local_Din_High	Local (FP Board) Digital Input Higher 8 Bits (8–15)	Bit to bit map.
X04	Local_Dout_Low	Local (FP Board) Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X05	Local_Dout_High	Local (FP Board) Digital Output Higher 8 Bits (8–15)	Bit to bit map.
X06	N/A	N/A	Reserved
X07	N/A	N/A	Reserved
X08	N/A	N/A	Reserved
X09	N/A	N/A	Reserved
X10	N/A	N/A	Reserved
X11	N/A	N/A	Reserved
X12.0	N/A	N/A	Reserved
X12.1	HS_1	Home Switch Axis 1	0: Home switch axis 1 is not active. 1: Home switch axis 1 is active.
X12.2	NLS_1	Negative Limit Switch Axis 1	0: Negative limit switch axis 1 is not active. 1: Negative limit switch axis 1 is active.
X12.3	PLS_1	Positive Limit Switch Axis 1	0: Positive limit switch axis 1 is not active. 1: Positive limit switch axis 1 is active.
X12.4	N/A	N/A	Reserved
X12.5	HS_2	Home Switch Axis 2	0: Home switch axis 2 is not active. 1: Home switch axis 2 is active.
X12.6	NLS_2	Negative Limit Switch Axis 2	0: Negative limit switch axis 2 is not active. 1: Negative limit switch axis 2 is active.
X12.7	PLS_2	Positive Limit Switch Axis 2	0: Positive limit switch axis 2 is not active. 1: Positive limit switch axis 2 is active.
X13.0	N/A	N/A	Reserved

ADDRESS	NAME	DESCRIPTION	NOTE
X13.1	HS_3	Home Switch Axis 3	0: Home switch axis 3 is not active. 1: Home switch axis 3 is active.
X13.2	NLS_3	Negative Limit Switch Axis 3	0: Negative limit switch axis 3 is not active. 1: Negative limit switch axis 3 is active.
X13.3	PLS_3	Positive Limit Switch Axis 3	0: Positive limit switch axis 3 is not active. 1: Positive limit switch axis 3 is active.
X13.4	N/A	N/A	Reserved
X13.5	HS_4	Home Switch Axis 4	0: Home switch axis 4 is not active. 1: Home switch axis 4 is active.
X13.6	NLS_4	Negative Limit Switch Axis 4	0: Negative limit switch axis 4 is not active. 1: Negative limit switch axis 4 is active.
X13.7	PLS_4	Positive Limit Switch Axis 4	0: Positive limit switch axis 4 is not active. 1: Positive limit switch axis 4 is active.
X14.0	AF_1	Amplifier Fault Axis 1	0: Amplifier fault axis 1 is not active. 1: Amplifier fault axis 1 is active.
X14.1	AC0_1	Alarm Code Bit 0 Axis 1	0: Alarm code bit 0 axis 1 is not active. 1: Alarm code bit 0 axis 1 is active.
X14.2	AC1_1	Alarm Code Bit 1 Axis 1	0: Alarm code bit 1 axis 1 is not active. 1: Alarm code bit 1 axis 1 is active.
X14.3	AC2_1	Alarm Code Bit 2 Axis 1	0: Alarm code bit 2 axis 1 is not active. 1: Alarm code bit 2 axis 1 is active.
X14.4	AF_2	Amplifier Fault Axis 2	0: Amplifier fault axis 2 is not active. 1: Amplifier fault axis 2 is active.
X14.5	AC0_2	Alarm Code Bit 0 Axis 2	0: Alarm code bit 0 axis 2 is not active. 1: Alarm code bit 0 axis 2 is active.
X14.6	AC1_2	Alarm Code Bit 1 Axis 2	0: Alarm code bit 1 axis 2 is not active. 1: Alarm code bit 1 axis 2 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X14.7	AC2_2	Alarm Code Bit 2 Axis 2	0: Alarm code bit 2 axis 2 is not active. 1: Alarm code bit 2 axis 2 is active.
X15.0	AF_3	Amplifier Fault Axis 3	0: Amplifier fault axis 3 is not active. 1: Amplifier fault axis 3 is active.
X15.1	AC0_3	Alarm Code Bit 0 Axis 3	0: Alarm code bit 0 axis 3 is not active. 1: Alarm code bit 0 axis 3 is active.
X15.2	AC1_3	Alarm Code Bit 1 Axis 3	0: Alarm code bit 1 axis 3 is not active. 1: Alarm code bit 1 axis 3 is active.
X15.3	AC2_3	Alarm Code Bit 2 Axis 3	0: Alarm code bit 2 axis 3 is not active. 1: Alarm code bit 2 axis 3 is active.
X15.4	AF_4	Amplifier Fault Axis 4	0: Amplifier fault axis 4 is not active. 1: Amplifier fault axis 4 is active.
X15.5	AC0_4	Alarm Code Bit 0 Axis 4	0: Alarm code bit 0 axis 4 is not active. 1: Alarm code bit 0 axis 4 is active.
X15.6	AC1_4	Alarm Code Bit 1 Axis 4	0: Alarm code bit 1 axis 4 is not active. 1: Alarm code bit 1 axis 4 is active.
X15.7	AC2_4	Alarm Code Bit 2 Axis 4	0: Alarm code bit 2 axis 4 is not active. 1: Alarm code bit 2 axis 4 is active.
X16	DC_Din_Low [0]	DC Module Address-0 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X17	DC_Din_High [0]	DC Module Address-0 Digital Input Higher 8 Bits (8–15)	Bit to bit map.
X18	DC_Dout_Low [0]	DC Module Address-0 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X19	DC_Dout_High [0]	DC Module Address-0 Digital Output Higher 8 Bits (8–15)	Bit to bit map.
X20	IM_Din_Low_0 [0]	IM Module Address-0 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X21	IM_Din_Low_1 [0]	IM Module Address-0 Digital Input Lower 8 Bits (8–15)	Bit to bit map.
X22	IM_Din_High_0 [0]	IM Module Address-0 Digital Input Higher 8 Bits (16–23)	Bit to bit map.
X23	IM_Din_High_1 [0]	IM Module Address-0 Digital Input Higher 8 Bits (24–31)	Bit to bit map

ADDRESS	NAME	DESCRIPTION	NOTE
X24	IM_Dout_Low_0 [0]	IM Module Address-0 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X25	IM_Dout_Low_1 [0]	IM Module Address-0 Digital Output Lower 8 Bits (8–15)	Bit to bit map.
X26	IM_Dout_High_0 [0]	IM Module Address-0 Digital Output Higher 8 Bits (16–23)	Bit to bit map.
X27	IM_Dout_High_1 [0]	IM Module Address-0 Digital Output Higher 8 Bits (24–31)	Bit to bit map.
X28.0	N/A	N/A	Reserved
X28.1	HS_5	Home Switch Axis 5	0: Home switch axis 5 is not active. 1: Home switch axis 5 is active.
X28.2	NLS_5	Negative Limit Switch Axis 5	0: Negative limit switch axis 5 is not active. 1: Negative limit switch axis 5 is active.
X28.3	PLS_5	Positive Limit Switch Axis 5	0: Positive limit switch axis 5 is not active. 1: Positive limit switch axis 5 is active.
X28.4	N/A	N/A	Reserved
X28.5	HS_6	Home Switch Axis 6	0: Home switch axis 6 is not active. 1: Home switch axis 6 is active.
X28.6	NLS_6	Negative Limit Switch Axis 6	0: Negative limit switch axis 6 is not active. 1: Negative limit switch axis 6 is active.
X28.7	PLS_6	Positive Limit Switch Axis 6	0: Positive limit switch axis 6 is not active. 1: Positive limit switch axis 6 is active.
X29.0	N/A	N/A	Reserved
X29.1	HS_7	Home Switch Axis 7	0: Home switch axis 7 is not active. 1: Home switch axis 7 is active.
X29.2	NLS_7	Negative Limit Switch Axis 7	0: Negative limit switch axis 7 is not active. 1: Negative limit switch axis 7 is active.
X29.3	PLS_7	Positive Limit Switch Axis 7	0: Positive limit switch axis 7 is not active. 1: Positive limit switch axis 7 is active.
X29.4	N/A	N/A	Reserved

ADDRESS	NAME	DESCRIPTION	NOTE
X29.5	HS_8	Home Switch Axis 8	0: Home switch axis 8 is not active. 1: Home switch axis 8 is active.
X29.6	NLS_8	Negative Limit Switch Axis 8	0: Negative limit switch axis 8 is not active. 1: Negative limit switch axis 8 is active.
X29.7	PLS_8	Positive Limit Switch Axis 8	0: Positive limit switch axis 8 is not active. 1: Positive limit switch axis 8 is active.
X30.0	AF_5	Amplifier Fault Axis 5	0: Amplifier fault axis 5 is not active. 1: Amplifier fault axis 5 is active.
X30.1	AC0_5	Alarm Code Bit 0 Axis 5	0: Alarm code bit 0 axis 5 is not active. 1: Alarm code bit 0 axis 5 is active.
X30.2	AC1_5	Alarm Code Bit 1 Axis 5	0: Alarm code bit 1 axis 5 is not active. 1: Alarm code bit 1 axis 5 is active.
X30.3	AC2_5	Alarm Code Bit 2 Axis 5	0: Alarm code bit 2 axis 5 is not active. 1: Alarm code bit 2 axis 5 is active.
X30.4	AF_6	Amplifier Fault Axis 6	0: Amplifier fault axis 6 is not active. 1: Amplifier fault axis 6 is active.
X30.5	AC0_6	Alarm Code Bit 0 Axis 6	0: Alarm code bit 0 axis 6 is not active. 1: Alarm code bit 0 axis 6 is active.
X30.6	AC1_6	Alarm Code Bit 1 Axis 6	0: Alarm code bit 1 axis 6 is not active. 1: Alarm code bit 1 axis 6 is active.
X30.7	AC2_6	Alarm Code Bit 2 Axis 6	0: Alarm code bit 2 axis 6 is not active. 1: Alarm code bit 2 axis 6 is active.
X31.0	AF_7	Amplifier Fault Axis 7	0: Amplifier fault axis 7 is not active. 1: Amplifier fault axis 7 is active.
X31.1	AC0_7	Alarm Code Bit 0 Axis 7	0: Alarm code bit 0 axis 7 is not active. 1: Alarm code bit 0 axis 7 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X31.2	AC1_7	Alarm Code Bit 1 Axis 7	0: Alarm code bit 1 axis 7 is not active. 1: Alarm code bit 1 axis 7 is active.
X31.3	AC2_7	Alarm Code Bit 2 Axis 7	0: Alarm code bit 2 axis 7 is not active. 1: Alarm code bit 2 axis 7 is active.
X31.4	AF_8	Amplifier Fault Axis 8	0: Amplifier fault axis 8 is not active. 1: Amplifier fault axis 8 is active.
X31.5	AC0_8	Alarm Code Bit 0 Axis 8	0: Alarm code bit 0 axis 8 is not active. 1: Alarm code bit 0 axis 8 is active.
X31.6	AC1_8	Alarm Code Bit 1 Axis 8	0: Alarm code bit 1 axis 8 is not active. 1: Alarm code bit 1 axis 8 is active.
X31.7	AC2_8	Alarm Code Bit 2 Axis 8	0: Alarm code bit 2 axis 8 is not active. 1: Alarm code bit 2 axis 8 is active.
X32	DC_Din_Low [1]	DC Module Address-1 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X33	DC_Din_High [1]	DC Module Address-1 Digital Input Higher 8 Bits (8–15)	Bit to bit map.
X34	DC_Dout_Low [1]	DC Module Address-1 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X35	DC_Dout_High [1]	DC Module Address-1 Digital Output Higher 8 Bits (8–15)	Bit to bit map.
X36	IM_Din_Low_0 [1]	IM Module Address-1 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X37	IM_Din_Low_1 [1]	IM Module Address-1 Digital Input Lower 8 Bits (8–15)	Bit to bit map.
X38	IM_Din_High_0 [1]	IM Module Address-1 Digital Input Higher 8 Bits (16–23)	Bit to bit map.
X39	IM_Din_High_1 [1]	IM Module Address-1 Digital Input Higher 8 Bits (24–31)	Bit to bit map.
X40	IM_Dout_Low_0 [1]	IM Module Address-1 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X41	IM_Dout_Low_1 [1]	IM Module Address-1 Digital Output Lower 8 Bits (8–15)	Bit to bit map.
X42	IM_Dout_High_0 [1]	IM Module Address-1 Digital Output Higher 8 Bits (16–23)	Bit to bit map.
X43	IM_Dout_High_1 [1]	IM Module Address-1 Digital Output Higher 8 Bits (23–31)	Bit to bit map.
X44.0	N/A	N/A	Reserved

ADDRESS	NAME	DESCRIPTION	NOTE
X44.1	HS_9	Home Switch Axis 9	0: Home switch axis 9 is not active. 1: Home switch axis 9 is active.
X44.2	NLS_9	Negative Limit Switch Axis 9	0: Negative limit switch axis 9 is not active. 1: Negative limit switch axis 9 is active.
X44.3	PLS_9	Positive Limit Switch Axis 9	0: Positive limit switch axis 9 is not active. 1: Positive limit switch axis 9 is active.
X44.4	N/A	N/A	Reserved
X44.5	HS_10	Home Switch Axis 10	0: Home switch axis 10 is not active. 1: Home switch axis 10 is active.
X44.6	NLS_10	Negative Limit Switch Axis 10	0: Negative limit switch axis 10 is not active. 1: Negative limit switch axis 10 is active.
X44.7	PLS_10	Positive Limit Switch Axis 10	0: Positive limit switch axis 10 is not active. 1: Positive limit switch axis 10 is active.
X45.0	N/A	N/A	Reserved
X45.1	HS_11	Home Switch Axis 11	0: Home switch axis 11 is not active. 1: Home switch axis 11 is active.
X45.2	NLS_11	Negative Limit Switch Axis 11	0: Negative limit switch axis 11 is not active. 1: Negative limit switch axis 11 is active.
X45.3	PLS_11	Positive Limit Switch Axis 11	0: Positive limit switch axis 11 is not active. 1: Positive limit switch axis 11 is active.
X45.4	N/A	N/A	Reserved
X45.5	HS_12	Home Switch Axis 12	0: Home switch axis 12 is not active. 1: Home switch axis 12 is active.
X45.6	NLS_12	Negative Limit Switch Axis 12	0: Negative limit switch axis 12 is not active. 1: Negative limit switch axis 12 is active.
X45.7	PLS_12	Positive Limit Switch Axis 12	0: Positive limit switch axis 12 is not active. 1: Positive limit switch axis 12 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X46.0	AF_9	Amplifier Fault Axis 9	0: Amplifier fault axis 9 is not active. 1: Amplifier fault axis 9 is active.
X46.1	AC0_9	Alarm Code Bit 0 Axis 9	0: Alarm code bit 0 axis 9 is not active. 1: Alarm code bit 0 axis 9 is active.
X46.2	AC1_9	Alarm Code Bit 1 Axis 9	0: Alarm code bit 1 axis 9 is not active. 1: Alarm code bit 1 axis 9 is active.
X46.3	AC2_9	Alarm Code Bit 2 Axis 9	0: Alarm code bit 2 axis 1 is not active. 1: Alarm code bit 2 axis 1 is active.
X46.4	AF_10	Amplifier Fault Axis 10	0: Amplifier fault axis 10 is not active. 1: Amplifier fault axis 10 is active.
X46.5	AC0_10	Alarm Code Bit 0 Axis 10	0: Alarm code bit 0 axis 10 is not active. 1: Alarm code bit 0 axis 10 is active.
X46.6	AC1_10	Alarm Code Bit 1 Axis 10	0: Alarm code bit 1 axis 10 is not active. 1: Alarm code bit 1 axis 10 is active.
X46.7	AC2_10	Alarm Code Bit 2 Axis 10	0: Alarm code bit 2 axis 10 is not active. 1: Alarm code bit 2 axis 10 is active.
X47.0	AF_11	Amplifier Fault Axis 11	0: Amplifier fault axis 3 is not active. 1: Amplifier fault axis 3 is active.
X47.1	AC0_11	Alarm Code Bit 0 Axis 11	0: Alarm code bit 0 axis 11 is not active. 1: Alarm code bit 0 axis 11 is active.
X47.2	AC1_11	Alarm Code Bit 1 Axis 11	0: Alarm code bit 1 axis 11 is not active. 1: Alarm code bit 1 axis 11 is active.
X47.3	AC2_11	Alarm Code Bit 2 Axis 11	0: Alarm code bit 2 axis 11 is not active. 1: Alarm code bit 2 axis 11 is active.
X47.4	AF_12	Amplifier Fault Axis 12	0: Amplifier fault axis 12 is not active. 1: Amplifier fault axis 12 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X47.5	AC0_12	Alarm Code Bit 0 Axis 12	0: Alarm code bit 0 axis 12 is not active. 1: Alarm code bit 0 axis 12 is active.
X47.6	AC1_12	Alarm Code Bit 1 Axis 12	0: Alarm code bit 1 axis 12 is not active. 1: Alarm code bit 1 axis 12 is active.
X47.7	AC2_12	Alarm Code Bit 2 Axis 12	0: Alarm code bit 2 axis 12 is not active. 1: Alarm code bit 2 axis 12 is active.
X48	DC_Din_Low [2]	DC Module Address-2 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X49	DC_Din_High [2]	DC Module Address-2 Digital Input Higher 8 Bits (8–15)	Bit to bit map.
X50	DC_Dout_Low [2]	DC Module Address-2 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X51	DC_Dout_High [2]	DC Module Address-2 Digital Output Higher 8 Bits (8–15)	Bit to bit map.
X52	IM_Din_Low_0 [2]	IM Module Address-2 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X53	IM_Din_Low_1 [2]	IM Module Address-2 Digital Input Lower 8 Bits (8–15)	Bit to bit map.
X54	IM_Din_High_0 [2]	IM Module Address-2 Digital Input Higher 8 Bits (16–23)	Bit to bit map.
X55	IM_Din_High_1 [2]	IM Module Address-2 Digital Input Higher 8 Bits (24–31)	Bit to bit map
X56	IM_Dout_Low_0 [2]	IM Module Address-2 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X57	IM_Dout_Low_1 [2]	IM Module Address-2 Digital Output Lower 8 Bits (8–15)	Bit to bit map.
X58	IM_Dout_High_0 [2]	IM Module Address-2 Digital Output Higher 8 Bits (16–23)	Bit to bit map.
X59	IM_Dout_High_1 [2]	IM Module Address-2 Digital Output Higher 8 Bits (24–31)	Bit to bit map.
X60.0	N/A	N/A	Reserved
X60.1	HS_13	Home Switch Axis 13	0: Home switch axis 13 is not active. 1: Home switch axis 13 is active.
X60.2	NLS_13	Negative Limit Switch Axis 13	0: Negative limit switch axis 13 is not active. 1: Negative limit switch axis 13 is active.
X60.3	PLS_13	Positive Limit Switch Axis 13	0: Positive limit switch axis 9 is not active. 1: Positive limit switch axis 9 is active.
X60.4	N/A	N/A	Reserved

ADDRESS	NAME	DESCRIPTION	NOTE
X60.5	HS_14	Home Switch Axis 14	0: Home switch axis 14 is not active. 1: Home switch axis 14 is active.
X60.6	NLS_14	Negative Limit Switch Axis 14	0: Negative limit switch axis 14 is not active. 1: Negative limit switch axis 14 is active.
X60.7	PLS_14	Positive Limit Switch Axis 14	0: Positive limit switch axis 14 is not active. 1: Positive limit switch axis 14 is active.
X61.0	N/A	N/A	Reserved
X61.1	HS_15	Home Switch Axis 15	0: Home switch axis 15 is not active. 1: Home switch axis 15 is active.
X61.2	NLS_15	Negative Limit Switch Axis 15	0: Negative limit switch axis 15 is not active. 1: Negative limit switch axis 15 is active.
X61.3	PLS_15	Positive Limit Switch Axis 15	0: Positive limit switch axis 15 is not active. 1: Positive limit switch axis 15 is active.
X61.4	N/A	N/A	Reserved
X61.5	HS_16	Home Switch Axis 16	0: Home switch axis 16 is not active. 1: Home switch axis 16 is active.
X61.6	NLS_16	Negative Limit Switch Axis 16	0: Negative limit switch axis 16 is not active. 1: Negative limit switch axis 16 is active.
X61.7	PLS_16	Positive Limit Switch Axis 16	0: Positive limit switch axis 16 is not active. 1: Positive limit switch axis 16 is active.
X62.0	AF_13	Amplifier Fault Axis 13	0: Amplifier fault axis 13 is not active. 1: Amplifier fault axis 13 is active.
X62.1	AC0_13	Alarm Code Bit 0 Axis 13	0: Alarm code bit 0 axis 13 is not active. 1: Alarm code bit 0 axis 13 is active.
X62.2	AC1_13	Alarm Code Bit 1 Axis 13	0: Alarm code bit 1 axis 13 is not active. 1: Alarm code bit 1 axis 13 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X62.3	AC2_13	Alarm Code Bit 2 Axis 13	0: Alarm code bit 2 axis 13 is not active. 1: Alarm code bit 2 axis 13 is active.
X62.4	AF_14	Amplifier Fault Axis 14	0: Amplifier fault axis 14 is not active. 1: Amplifier fault axis 14 is active.
X62.5	AC0_14	Alarm Code Bit 0 Axis 14	0: Alarm code bit 0 axis 14 is not active. 1: Alarm code bit 0 axis 14 is active.
X62.6	AC1_14	Alarm Code Bit 1 Axis 14	0: Alarm code bit 1 axis 14 is not active. 1: Alarm code bit 1 axis 14 is active.
X62.7	AC2_14	Alarm Code Bit 2 Axis 14	0: Alarm code bit 2 axis 14 is not active. 1: Alarm code bit 2 axis 14 is active.
X63.0	AF_15	Amplifier Fault Axis 15	0: Amplifier fault axis 15 is not active. 1: Amplifier fault axis 15 is active.
X63.1	AC0_15	Alarm Code Bit 0 Axis 15	0: Alarm code bit 0 axis 15 is not active. 1: Alarm code bit 0 axis 15 is active.
X63.2	AC1_15	Alarm Code Bit 1 Axis 15	0: Alarm code bit 1 axis 15 is not active. 1: Alarm code bit 1 axis 15 is active.
X63.3	AC2_15	Alarm Code Bit 2 Axis 15	0: Alarm code bit 2 axis 15 is not active. 1: Alarm code bit 2 axis 15 is active.
X63.4	AF_4	Amplifier Fault Axis 4	0: Amplifier fault axis 4 is not active. 1: Amplifier fault axis 4 is active.
X63.5	AC0_4	Alarm Code Bit 0 Axis 4	0: Alarm code bit 0 axis 4 is not active. 1: Alarm code bit 0 axis 4 is active.
X63.6	AC1_4	Alarm Code Bit 1 Axis 4	0: Alarm code bit 1 axis 4 is not active. 1: Alarm code bit 1 axis 4 is active.
X63.7	AC2_4	Alarm Code Bit 2 Axis 4	0: Alarm code bit 2 axis 4 is not active. 1: Alarm code bit 2 axis 4 is active.

ADDRESS	NAME	DESCRIPTION	NOTE
X64	DC_Din_Low [3]	DC Module Address-3 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X65	DC_Din_High [3]	DC Module Address-3 Digital Input Higher 8 Bits (8–15)	Bit to bit map.
X66	DC_Dout_Low [3]	DC Module Address-3 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X67	DC_Dout_High [3]	DC Module Address-3 Digital Output Higher 8 Bits (8–15)	Bit to bit map.
X68	IM_Din_Low_0 [3]	IM Module Address-3 Digital Input Lower 8 Bits (0–7)	Bit to bit map.
X69	IM_Din_Low_1 [3]	IM Module Address-3 Digital Input Lower 8 Bits (8–15)	Bit to bit map.
X70	IM_Din_High_0 [3]	IM Module Address-3 Digital Input Higher 8 Bits (16–23)	Bit to bit map.
X71	IM_Din_High_1 [3]	IM Module Address-3 Digital Input Higher 8 Bits (24–31)	Bit to bit map
X72	IM_Dout_Low_0 [3]	IM Module Address-3 Digital Output Lower 8 Bits (0–7)	Bit to bit map.
X73	IM_Dout_Low_1 [3]	IM Module Address-3 Digital Output Lower 8 Bits (8–15)	Bit to bit map.
X74	IM_Dout_High_0 [3]	IM Module Address-3 Digital Output Higher 8 Bits (16–23)	Bit to bit map.
X75	IM_Dout_High_1 [3]	IM Module Address-3 Digital Output Higher 8 Bits (24–31)	Bit to bit map.
X76	N/A	N/A	Reserved
X77	N/A	N/A	Reserved
X78	N/A	N/A	Reserved
X79	N/A	N/A	Reserved
X80.0	ServoWire_Din_1_1	Digital Input on ServoWire Drive #1 – In1	
X80.1	ServoWire_Din_1_2	Digital Input on ServoWire Drive #1 – In2	
X80.2	ServoWire_Din_1_A	Digital Input on ServoWire Drive #1 – Asen	
X80.3	ServoWire_Din_1_B	Digital Input on ServoWire Drive #1 – Bsen	
X80.4	ServoWire_Din_1_C	Digital Input on ServoWire Drive #1 – Csen	
X81.0	ServoWire_Din_2_1	Digital Input on ServoWire Drive #2 – In1	
X81.1	ServoWire_Din_2_2	Digital Input on ServoWire Drive #2 – In2	
X81.2	ServoWire_Din_2_A	Digital Input on ServoWire Drive #2 – Asen	
X81.3	ServoWire_Din_2_B	Digital Input on ServoWire Drive #2 – Bsen	
X81.4	ServoWire_Din_2_C	Digital Input on ServoWire Drive #2 – Csen	
X82.0	ServoWire_Din_3_1	Digital Input on ServoWire Drive #3 – In1	

ADDRESS	NAME	DESCRIPTION	NOTE
X82.1	ServoWire_Din_3_2	Digital Input on ServoWire Drive #3 – In2	
X82.2	ServoWire_Din_3_A	Digital Input on ServoWire Drive #3 – Asen	
X82.3	ServoWire_Din_3_B	Digital Input on ServoWire Drive #3 – Bsen	
X82.4	ServoWire_Din_3_C	Digital Input on ServoWire Drive #3 – Csen	
X83.0	ServoWire_Din_4_1	Digital Input on ServoWire Drive #4 – In1	
X83.1	ServoWire_Din_4_2	Digital Input on ServoWire Drive #4 – In2	
X83.2	ServoWire_Din_4_A	Digital Input on ServoWire Drive #4 – Asen	
X83.3	ServoWire_Din_4_B	Digital Input on ServoWire Drive #4 – Bsen	
X83.4	ServoWire_Din_4_C	Digital Input on ServoWire Drive #4 – Csen	
X84.0	ServoWire_Din_5_1	Digital Input on ServoWire Drive #5 – In1	
X84.1	ServoWire_Din_5_2	Digital Input on ServoWire Drive #5 – In2	
X84.2	ServoWire_Din_5_A	Digital Input on ServoWire Drive #5 – Asen	
X84.3	ServoWire_Din_5_B	Digital Input on ServoWire Drive #5 – Bsen	
X84.4	ServoWire_Din_5_C	Digital Input on ServoWire Drive #5 – Csen	
X85.0	ServoWire_Din_6_1	Digital Input on ServoWire Drive #6 – In1	
X85.1	ServoWire_Din_6_2	Digital Input on ServoWire Drive #6 – In2	
X85.2	ServoWire_Din_6_A	Digital Input on ServoWire Drive #6 – Asen	
X85.3	ServoWire_Din_6_B	Digital Input on ServoWire Drive #6 – Bsen	
X85.4	ServoWire_Din_6_C	Digital Input on ServoWire Drive #6 – Csen	
X86.0	ServoWire_Din_7_1	Digital Input on ServoWire Drive #7 – In1	
X86.1	ServoWire_Din_7_2	Digital Input on ServoWire Drive #7 – In2	
X86.2	ServoWire_Din_7_A	Digital Input on ServoWire Drive #7 – Asen	
X86.3	ServoWire_Din_7_B	Digital Input on ServoWire Drive #7 – Bsen	
X86.4	ServoWire_Din_7_C	Digital Input on ServoWire Drive #7 – Csen	
X87.0	ServoWire_Din_8_1	Digital Input on ServoWire Drive #8 – In1	
X87.1	ServoWire_Din_8_2	Digital Input on ServoWire Drive #8 – In2	

ADDRESS	NAME	DESCRIPTION	NOTE
X87.2	ServoWire_Din_8_A	Digital Input on ServoWire Drive #8 – Asen	
X87.3	ServoWire_Din_8_B	Digital Input on ServoWire Drive #8 – Bsen	
X87.4	ServoWire_Din_8_C	Digital Input on ServoWire Drive #8 – Csen	
X88.0	ServoWire_Din_9_1	Digital Input on ServoWire Drive #9 – In1	
X88.1	ServoWire_Din_9_2	Digital Input on ServoWire Drive #9 – In2	
X88.2	ServoWire_Din_9_A	Digital Input on ServoWire Drive #9 – Asen	
X88.3	ServoWire_Din_9_B	Digital Input on ServoWire Drive #9 – Bsen	
X88.4	ServoWire_Din_9_C	Digital Input on ServoWire Drive #9 – Csen	
X89.0	ServoWire_Din_10_1	Digital Input on ServoWire Drive #10 – In1	
X89.1	ServoWire_Din_10_2	Digital Input on ServoWire Drive #10 – In2	
X89.2	ServoWire_Din_10_A	Digital Input on ServoWire Drive #10 – Asen	
X89.3	ServoWire_Din_10_B	Digital Input on ServoWire Drive #10 – Bsen	
X89.4	ServoWire_Din_10_C	Digital Input on ServoWire Drive #10 – Csen	
X90.0	ServoWire_Din_11_1	Digital Input on ServoWire Drive #11 – In1	
X90.1	ServoWire_Din_11_2	Digital Input on ServoWire Drive #11 – In2	
X90.2	ServoWire_Din_11_A	Digital Input on ServoWire Drive #11 – Asen	
X90.3	ServoWire_Din_11_B	Digital Input on ServoWire Drive #11 – Bsen	
X90.4	ServoWire_Din_11_C	Digital Input on ServoWire Drive #11 – Csen	
X91.0	ServoWire_Din_12_1	Digital Input on ServoWire Drive #12 – In1	
X91.1	ServoWire_Din_12_2	Digital Input on ServoWire Drive #12 – In2	
X91.2	ServoWire_Din_12_A	Digital Input on ServoWire Drive #12 – Asen	
X91.3	ServoWire_Din_12_B	Digital Input on ServoWire Drive #12 – Bsen	
X91.4	ServoWire_Din_12_C	Digital Input on ServoWire Drive #12 – Csen	
X92.0	ServoWire_Din_13_1	Digital Input on ServoWire Drive #13 – In1	
X92.1	ServoWire_Din_13_2	Digital Input on ServoWire Drive #13 – In2	
X92.2	ServoWire_Din_13_A	Digital Input on ServoWire Drive #13 – Asen	

ADDRESS	NAME	DESCRIPTION	NOTE
X92.3	ServoWire_Din_13_B	Digital Input on ServoWire Drive #13 – Bsen	
X92.4	ServoWire_Din_13_C	Digital Input on ServoWire Drive #13 – Csen	
X93.0	ServoWire_Din_14_1	Digital Input on ServoWire Drive #14 – In1	
X93.1	ServoWire_Din_14_2	Digital Input on ServoWire Drive #14 – In2	
X93.2	ServoWire_Din_14_A	Digital Input on ServoWire Drive #14 – Asen	
X93.3	ServoWire_Din_14_B	Digital Input on ServoWire Drive #14 – Bsen	
X93.4	ServoWire_Din_14_C	Digital Input on ServoWire Drive #14 – Csen	
X94.0	ServoWire_Din_15_1	Digital Input on ServoWire Drive #15 – In1	
X94.1	ServoWire_Din_15_2	Digital Input on ServoWire Drive #15 – In2	
X94.2	ServoWire_Din_15_A	Digital Input on ServoWire Drive #15 – Asen	
X94.3	ServoWire_Din_15_B	Digital Input on ServoWire Drive #15 – Bsen	
X94.4	ServoWire_Din_15_C	Digital Input on ServoWire Drive #15 – Csen	
X95.0	ServoWire_Din_16_1	Digital Input on ServoWire Drive #16 – In1	
X95.1	ServoWire_Din_16_2	Digital Input on ServoWire Drive #16 – In2	
X95.2	ServoWire_Din_16_A	Digital Input on ServoWire Drive #16 – Asen	
X95.3	ServoWire_Din_16_B	Digital Input on ServoWire Drive #16 – Bsen	
X95.4	ServoWire_Din_16_C	Digital Input on ServoWire Drive #16 – Csen	
X96	N/A	N/A	Reserved
X97	N/A	N/A	Reserved
X98	N/A	N/A	Reserved
X99	N/A	N/A	Reserved

Y Data Mapping Table

ADDRESS	NAME	DESCRIPTION	NOTE
Y00	DC_Dout_Low [0]	DC module Address-0 digital output lower 8 bits (0–7)	Bit to bit map.
Y01	DC_Dout_High [0]	DC module Address-0 digital output higher 8 bits (8–15)	Bit to bit map.
Y02	DC_Dout_Low [1]	DC module Address-1 digital output lower 8 bits (0–7)	Bit to bit map.
Y03	DC_Dout_High [1]	DC module Address-1 digital output higher 8 bits (8–15)	Bit to bit map.
Y04	DC_Dout_Low [2]	DC module Address-2 digital output lower 8 bits (0–7)	Bit to bit map.
Y05	DC_Dout_High [2]	DC module Address-2 digital output higher 8 bits (8–15)	Bit to bit map.
Y06	DC_Dout_Low [3]	DC module Address-3 digital output lower 8 bits (0–7)	Bit to bit map.
Y07	DC_Dout_High [3]	DC module Address-3 digital output higher 8 bits (8–15)	Bit to bit map.
Y08	IM_Dout_Low_0 [0]	IM module Address-0 digital output lower 8 bits (0–7)	Bit to bit map.
Y09	IM_Dout_Low_1 [0]	IM module Address-0 digital output lower 8 bits (8–15)	Bit to bit map.
Y10	IM_Dout_High_0 [0]	IM module Address-0 digital output higher 8 bits (16–23)	Bit to bit map.
Y11	IM_Dout_High_1 [0]	IM module Address-0 digital output higher 8 bits (24–31)	Bit to bit map.
Y12	IM_Dout_Low_0 [1]	IM module Address-1 digital output lower 8 bits (0–7)	Bit to bit map.
Y13	IM_Dout_Low_1 [1]	IM module Address-1 digital output lower 8 bits (8–15)	Bit to bit map.
Y14	IM_Dout_High_0 [1]	IM module Address-1 digital output higher 8 bits (16–23)	Bit to bit map.
Y15	IM_Dout_High_1 [1]	IM module Address-1 digital output higher 8 bits (24–31)	Bit to bit map.
Y16	IM_Dout_Low_0 [2]	IM module Address-2 digital output lower 8 bits (0–7)	Bit to bit map.
Y17	IM_Dout_Low_1 [2]	IM module Address-2 digital output lower 8 bits (8–15)	Bit to bit map.
Y18	IM_Dout_High_0 [2]	IM module Address-2 digital output higher 8 bits (16–23)	Bit to bit map.
Y19	IM_Dout_High_1 [2]	IM module Address-2 digital output higher 8 bits (24–31)	Bit to bit map.
Y20	IM_Dout_Low_0 [3]	IM module Address-3 digital output lower 8 bits (0–7)	Bit to bit map.
Y21	IM_Dout_Low_1 [3]	IM module Address-3 digital output lower 8 bits (8–15)	Bit to bit map.
Y22	IM_Dout_High_0 [3]	IM module Address-3 digital output higher 8 bits (16–23)	Bit to bit map.
Y23	IM_Dout_High_1 [3]	IM module Address-3 digital output higher 8 bits (24–31)	Bit to bit map.

ADDRESS	NAME	DESCRIPTION	NOTE
Y24	Local_Dout_Low	Local (FP board) digital output lower 8 bits (0–7)	Bit to bit map.
Y25	Local_Dout_High	Local (FP board) digital output higher 8 bits (8–15)	Bit to bit map.
Y26.0	ServoWire_Dout_1_1	Digital Output on ServoWire Drive #1 – Out1	
Y26.1	ServoWire_Dout_1_2	Digital Output on ServoWire Drive #1 – Out2	
Y26.2	ServoWire_Dout_1_3	Digital Output on ServoWire Drive #1 – Out3	
Y26.3	ServoWire_Dout_1_4	Digital Output on ServoWire Drive #1 – Out4	
Y26.4	ServoWire_Dout_1_5	Digital Output on ServoWire Drive #1 – Out5	
Y26.5	ServoWire_Dout_1_6	Digital Output on ServoWire Drive #1 – Out6	
Y27	N/A	N/A	Reserved
Y28.0	ServoWire_Dout_2_1	Digital Output on ServoWire Drive #2 – Out1	
Y28.1	ServoWire_Dout_2_2	Digital Output on ServoWire Drive #2 – Out2	
Y28.2	ServoWire_Dout_2_3	Digital Output on ServoWire Drive #2 – Out3	
Y28.3	ServoWire_Dout_2_4	Digital Output on ServoWire Drive #2 – Out4	
Y28.4	ServoWire_Dout_2_5	Digital Output on ServoWire Drive #2 – Out5	
Y28.5	ServoWire_Dout_2_6	Digital Output on ServoWire Drive #2 – Out6	
Y29	N/A	N/A	Reserved
Y30.0	ServoWire_Dout_3_1	Digital Output on ServoWire Drive #3 – Out1	
Y30.1	ServoWire_Dout_3_2	Digital Output on ServoWire Drive #3 – Out2	
Y30.2	ServoWire_Dout_3_3	Digital Output on ServoWire Drive #3 – Out3	
Y30.3	ServoWire_Dout_3_4	Digital Output on ServoWire Drive #3 – Out4	
Y30.4	ServoWire_Dout_3_5	Digital Output on ServoWire Drive #3 – Out5	
Y30.5	ServoWire_Dout_3_6	Digital Output on ServoWire Drive #3 – Out6	
Y31	N/A	N/A	Reserved
Y32.0	ServoWire_Dout_4_1	Digital Output on ServoWire Drive #4 – Out1	
Y32.1	ServoWire_Dout_4_2	Digital Output on ServoWire Drive #4 – Out2	
Y32.2	ServoWire_Dout_4_3	Digital Output on ServoWire Drive #4 – Out3	
Y32.3	ServoWire_Dout_4_4	Digital Output on ServoWire Drive #4 – Out4	

ADDRESS	NAME	DESCRIPTION	NOTE
Y32.4	ServoWire_Dout_4_5	Digital Output on ServoWire Drive #4 – Out5	
Y32.5	ServoWire_Dout_4_6	Digital Output on ServoWire Drive #4 – Out6	
Y33	N/A	N/A	Reserved
Y34.0	ServoWire_Dout_5_1	Digital Output on ServoWire Drive #5 – Out1	
Y34.1	ServoWire_Dout_5_2	Digital Output on ServoWire Drive #5 – Out2	
Y34.2	ServoWire_Dout_5_3	Digital Output on ServoWire Drive #5 – Out3	
Y34.3	ServoWire_Dout_5_4	Digital Output on ServoWire Drive #5 – Out4	
Y34.4	ServoWire_Dout_5_5	Digital Output on ServoWire Drive #5 – Out5	
Y34.5	ServoWire_Dout_5_6	Digital Output on ServoWire Drive #5 – Out6	
Y35	N/A	N/A	Reserved
Y36.0	ServoWire_Dout_6_1	Digital Output on ServoWire Drive #6 – Out1	
Y36.1	ServoWire_Dout_6_2	Digital Output on ServoWire Drive #6 – Out2	
Y36.2	ServoWire_Dout_6_3	Digital Output on ServoWire Drive #6 – Out3	
Y36.3	ServoWire_Dout_6_4	Digital Output on ServoWire Drive #6 – Out4	
Y36.4	ServoWire_Dout_6_5	Digital Output on ServoWire Drive #6 – Out5	
Y36.5	ServoWire_Dout_6_6	Digital Output on ServoWire Drive #6 – Out6	
Y37	N/A	N/A	Reserved
Y38.0	ServoWire_Dout_7_1	Digital Output on ServoWire Drive #7 – Out1	
Y38.1	ServoWire_Dout_7_2	Digital Output on ServoWire Drive #7 – Out2	
Y38.2	ServoWire_Dout_7_3	Digital Output on ServoWire Drive #7 – Out3	
Y38.3	ServoWire_Dout_7_4	Digital Output on ServoWire Drive #7 – Out4	
Y38.4	ServoWire_Dout_7_5	Digital Output on ServoWire Drive #7 – Out5	
Y38.5	ServoWire_Dout_7_6	Digital Output on ServoWire Drive #7 – Out6	
Y39	N/A	N/A	Reserved
Y40.0	ServoWire_Dout_8_1	Digital Output on ServoWire Drive #8 – Out1	
Y40.1	ServoWire_Dout_8_2	Digital Output on ServoWire Drive #8 – Out2	
Y40.2	ServoWire_Dout_8_3	Digital Output on ServoWire Drive #8 – Out3	

ADDRESS	NAME	DESCRIPTION	NOTE
Y40.3	ServoWire_Dout_8_4	Digital Output on ServoWire Drive #8 – Out4	
Y40.4	ServoWire_Dout_8_5	Digital Output on ServoWire Drive #8 – Out5	
Y40.5	ServoWire_Dout_8_6	Digital Output on ServoWire Drive #8 – Out6	
Y41	N/A	N/A	Reserved
Y42.0	ServoWire_Dout_9_1	Digital Output on ServoWire Drive #9 – Out1	
Y42.1	ServoWire_Dout_9_2	Digital Output on ServoWire Drive #9 – Out2	
Y42.2	ServoWire_Dout_9_3	Digital Output on ServoWire Drive #9 – Out3	
Y42.3	ServoWire_Dout_9_4	Digital Output on ServoWire Drive #9 – Out4	
Y42.4	ServoWire_Dout_9_5	Digital Output on ServoWire Drive #9 – Out5	
Y42.5	ServoWire_Dout_9_6	Digital Output on ServoWire Drive #9 – Out6	
Y43	N/A	N/A	Reserved
Y44.0	ServoWire_Dout_10_1	Digital Output on ServoWire Drive #10 – Out1	
Y44.1	ServoWire_Dout_10_2	Digital Output on ServoWire Drive #10 – Out2	
Y44.2	ServoWire_Dout_10_3	Digital Output on ServoWire Drive #10 – Out3	
Y44.3	ServoWire_Dout_10_4	Digital Output on ServoWire Drive #10 – Out4	
Y44.4	ServoWire_Dout_10_5	Digital Output on ServoWire Drive #10 – Out5	
Y44.5	ServoWire_Dout_10_6	Digital Output on ServoWire Drive #10 – Out6	
Y45	N/A	N/A	Reserved
Y46.0	ServoWire_Dout_11_1	Digital Output on ServoWire Drive #11 – Out1	
Y46.1	ServoWire_Dout_11_2	Digital Output on ServoWire Drive #11 – Out2	
Y46.2	ServoWire_Dout_11_3	Digital Output on ServoWire Drive #11 – Out3	
Y46.3	ServoWire_Dout_11_4	Digital Output on ServoWire Drive #11 – Out4	
Y46.4	ServoWire_Dout_11_5	Digital Output on ServoWire Drive #11 – Out5	
Y46.5	ServoWire_Dout_11_6	Digital Output on ServoWire Drive #11 – Out6	
Y47	N/A	N/A	Reserved
Y48.0	ServoWire_Dout_12_1	Digital Output on ServoWire Drive #12 – Out1	
Y48.1	ServoWire_Dout_12_2	Digital Output on ServoWire Drive #12 – Out2	

ADDRESS	NAME	DESCRIPTION	NOTE
Y48.2	ServoWire_Dout_12_3	Digital Output on ServoWire Drive #12 – Out3	
Y48.3	ServoWire_Dout_12_4	Digital Output on ServoWire Drive #12 – Out4	
Y48.4	ServoWire_Dout_12_5	Digital Output on ServoWire Drive #12 – Out5	
Y48.5	ServoWire_Dout_12_6	Digital Output on ServoWire Drive #12 – Out6	
Y49	N/A	N/A	Reserved
Y50.0	ServoWire_Dout_13_1	Digital Output on ServoWire Drive #13 – Out1	
Y50.1	ServoWire_Dout_13_2	Digital Output on ServoWire Drive #13 – Out2	
Y50.2	ServoWire_Dout_13_3	Digital Output on ServoWire Drive #13 – Out3	
Y50.3	ServoWire_Dout_13_4	Digital Output on ServoWire Drive #13 – Out4	
Y50.4	ServoWire_Dout_13_5	Digital Output on ServoWire Drive #13 – Out5	
Y50.5	ServoWire_Dout_13_6	Digital Output on ServoWire Drive #13 – Out6	
Y51	N/A	N/A	Reserved
Y52.0	ServoWire_Dout_14_1	Digital Output on ServoWire Drive #14 – Out1	
Y52.1	ServoWire_Dout_14_2	Digital Output on ServoWire Drive #14 – Out2	
Y52.2	ServoWire_Dout_14_3	Digital Output on ServoWire Drive #14 – Out3	
Y52.3	ServoWire_Dout_14_4	Digital Output on ServoWire Drive #14 – Out4	
Y52.4	ServoWire_Dout_14_5	Digital Output on ServoWire Drive #14 – Out5	
Y52.5	ServoWire_Dout_14_6	Digital Output on ServoWire Drive #14 – Out6	
Y53	N/A	N/A	Reserved
Y54.0	ServoWire_Dout_15_1	Digital Output on ServoWire Drive #15 – Out1	
Y54.1	ServoWire_Dout_15_2	Digital Output on ServoWire Drive #15 – Out2	
Y54.2	ServoWire_Dout_15_3	Digital Output on ServoWire Drive #15 – Out3	
Y54.3	ServoWire_Dout_15_4	Digital Output on ServoWire Drive #15 – Out4	
Y54.4	ServoWire_Dout_15_5	Digital Output on ServoWire Drive #15 – Out5	
Y54.5	ServoWire_Dout_15_6	Digital Output on ServoWire Drive #15 – Out6	
Y55	N/A	N/A	Reserved
Y56.0	ServoWire_Dout_16_1	Digital Output on ServoWire Drive #16 – Out1	

ADDRESS	NAME	DESCRIPTION	NOTE
Y56.1	ServoWire_Dout_16_2	Digital Output on ServoWire Drive #16 – Out2	
Y56.2	ServoWire_Dout_16_3	Digital Output on ServoWire Drive #16 – Out3	
Y56.3	ServoWire_Dout_16_4	Digital Output on ServoWire Drive #16 – Out4	
Y56.4	ServoWire_Dout_16_5	Digital Output on ServoWire Drive #16 – Out5	
Y56.5	ServoWire_Dout_16_6	Digital Output on ServoWire Drive #16 – Out6	
Y57	N/A	N/A	Reserved

NC MODE	MINC (F3.0)	MH (F3.1)	MJ (F3.2)	MMDI (F3.3)	MRMT (F3.4)	MMEM (F3.5)	MREF (F4.5)	MREF1 (F4.6)
MDI	0	0	0	1	0	0	0	0
HOME	0	0	0	0	0	0	1	1
AUTO	0	0	0	0	0	1	0	0
JOG	0	0	1	0	0	0	0	0
HANDWHEEL	1	1	0	0	0	0	0	0
RAPID	0	0	1	0	0	0	0	0
DNC	1	1	0	0	1	1	0	0
POS	0	0	0	1	0	0	0	0

Table B-1: SwPLC F Address Map for NC Mode Settings

MULTIPLE	MP1 (F85.4/G19.0)	MP2 (F85.5/G19.1)
X1	0	0
X10	1	0
X100	0	1
X1000	1	1

Table B-2: SwPLC F/G Address Map for HandWheel Multiple Selection

AXIS	HS1A (F86.0, G18.0) HS1IA (F88.0, G41.0)	HS1B (F86.1, G18.1) HS1IB (F88.1, G41.1)	HS1C (F86.2, G18.2) HS1IC (F88.2, G41.2)	HS1D (F86.3, G18.3) HS1ID (F88.3, G41.3)
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1

Table B-3: SwPLC F/G Address Map for HandWheel and HandWheel Interrupt Axis Selection

ADDRESS	SIGNAL	WEIGHT
G10.0	0	1
	1	0
G10.1	0	2
	1	0
G10.2	0	4
	1	0
G10.3	0	8
	1	0
G10.4	0	16
	1	0
G10.5	0	32
	1	0
G10.6	0	64
	1	0
G10.7	0	128
	1	0
G11.0	0	256
	1	0
G11.1	0	512
	1	0
G11.2	0	1024
	1	0
G11.3	0	2048
	1	0
G11.4	0	4096
	1	0
G11.5	0	8192
	1	0
G11.6	0	16384
	1	0
G11.7	0	32768
	1	0

NOTE: Manual Feedrate Override = Sum of the Weights from G10.0 to G11.7 (percent)

Table B-4: SwPLC G Address Map for Manual Feedrate Override

ADDRESS	SIGNAL	WEIGHT
G12.0	0	1
	1	0
G12.1	0	2
	1	0
G12.2	0	4
	1	0
G12.3	0	8
	1	0
G12.4	0	16
	1	0
G12.5	0	32
	1	0
G12.6	0	64
	1	0
G12.7	0	128
	1	0

NOTE: Feedrate Override = Sum of the Weights from G12.0 to G12.7 (percent)

Table B-5: SwPLC G Address Map for Feedrate Override

RAPID OVERRIDE	ROV1 (G14.0)	ROV2 (G14.1)
100	0	0
50	1	0
25	0	1
0	1	1

Table B-6: SwPLC G Address Map for Rapid Override

NC MODE	MD1 (G43.0)	MD2 (G43.1)	MD4 (G43.2)	DNCI (G43.3)	ZRN (G43.4)	RT (G43.5)
MDI	0	0	0	0	0	N/A
HOME	1	0	1	0	1	N/A
AUTO	1	0	0	0	0	N/A
JOG	1	0	1	0	0	0
HandWheel	0	0	1	0	0	N/A
RAPID	1	0	1	0	0	1
DNC	1	0	0	1	0	N/A
POS	0	1	1	0	0	N/A

Table B-7: SwPLC G Address Map for NC Mode Settings

ADDRESS	NAME	IN JOG MODE	IN RAPID MODE	IN HOME MODE
G100	+J1 to +J8	Move axis in plus direction at jog feedrate	Move axis in plus direction at rapid feedrate	Start searching for Z pulse in the plus direction if the home direction parameter is set to plus
G102	-J1 to -J8	Move axis in minus direction at jog feedrate	Move axis in minus direction at rapid feedrate	Start searching for Z pulse in the minus direction if the home direction parameter is set to minus

Table B-8: SwPLC G Address Map for Jog Axis Control

Index

.

.bin files 4-4
.div files 4-4
.fig files 4-4
.lad file, default 4-3
.lad files 4-1
.lst files 4-4
.mod file, default 4-4
.mod files 4-4

A

ADD command 9-82
ADDB command 9-84
address range
  F0~F399 6-3
  G0~G399 6-3
addresses ..... *See* memory addresses
  A0~A99 6-8
  C0~C79 ..... 6-5, 7-1
  counter 7-1, 7-2
  D0~D1999 ..... 6-6, 7-2
  data table 7-2
  format in a ladder diagram 3-11
  K0~K99 ..... 6-6, 7-2
  keep relay 7-2
  R9000~R9099 6-4
  signal letters in a ladder diagram . 3-11
  T0~T399 6-7
  X0~X99 6-3
  Y0~Y99 6-3
addresses, numerical data 9-8
alarm relay addresses 6-8
AND command 8-10
AND.NOT command 8-10
AND.STK command 8-15
architecture, ServoWorksPLC
  Application Suite 3-2

B

basic commands 8-1
  AND 8-10
  AND.NOT 8-10
  AND.STK 8-15
  difference from functional commands 8-2
  OR 8-11
  OR.NOT 8-11
  OR.STK 8-15
  RD 8-4
  RD.NOT 8-6
  RD.NOT.STK 8-13
  RD.STK 8-11
  signal addresses 8-2
  sumamry 8-3
  WRT 8-8
  WRT.NOT 8-9
BCD ..... *See* binary coded decimal
binary coded decimal (BCD) 9-5, 9-6
binary files 1-3
binary format 4-1, 9-5
  advantages of 9-5
  example 9-6
  memory storage of 9-7
  Two's Complement notation 9-7
bit pattern display 1-4, 3-1
Bit Pattern Window 3-16
bit patterns 3-16

C

COD command 9-37
CODB command 9-41
coding convention 4-6
COIN command 9-66
COM command 9-47
COME command 9-50
commands
  basic 8-1
  types of (basic and functional) 8-2
comments in a ladder diagram 3-12
COMP command 9-62
COMPB command 9-64
Compile Finish Dialog Box 3-5
compiling sequence programs 1-3, 3-4
Control Console Application 1-3, 3-1
control values 9-4

counter
preset 9-20, 9-23, 9-26
ring 9-20, 9-26
up/down 9-20, 9-26
counter addresses 6-5, 7-1, 7-2
counters, setting up 3-8
CTR command 9-20
CTRC command 9-26
cycle, PLC 1-2

D

data table 7-3, 7-4
data table addresses 6-6, 7-2
data, addresses for numbers 9-8
DCNV command 9-58
DCNVB command 9-60
debugging sequence programs 1-4, 3-9
DEC command 9-16
DECB command 9-18
default .lad file 4-3
default .mod file 4-4
description of PLC 1-1
Diagnose Search Window 3-16
display of bit patterns 1-4, 3-1
DIV command 9-96
DIVB command 9-99
DSCH command 9-71
DSCHB 7-3
DSCHB command 9-74

E

Edit Counter Window 3-8
Edit Keep Relay Window 3-6
Edit Timer Window 3-7
editing sequence programs 1-3
E-STOP, HandWheel 4-1
executable binary files 1-3
executable binary format 4-1
execution of a sequence program 5-1
execution time of a sequence program. 5-2

F

F data mapping tables A-2, B-2
Fanuc-compatible ladder logic 1-4
files, sequence program 4-1
functional command register 9-9
functional commands 9-1
ADD 9-82
ADDB 9-84
COD 9-37
CODB 9-41
COIN 9-66
COM 9-47
COME 9-50
COMP 9-62
COMPB 9-64
control values 9-4
CTR 9-20
CTRC 9-26
DCNV 9-58
DCNVB 9-60
DEC 9-16
DECB 9-18
difference from basic commands ... 8-2
DIV 9-96
DIVB 9-99
DSCH 9-71
DSCHB 7-3, 9-74
format 9-3
JMP 9-51
JMPE 9-54
MOVE 9-43
MOVOR 9-46
MUL 9-91
MULB 9-94
NUME 9-101
NUMEB 9-103
numerical data addresses 9-8
operation data 9-5
parameters 9-5
PARI 9-55
ROT 9-29
ROTB 9-33
SFT 9-68
SUB 9-86
SUBB 9-89
summary 9-1, 9-2
TMR 9-10
TMRB 9-12
TMRC 9-14
W1 9-5

XMOV 9-76
XMOVB 7-3, 9-80

G

G data mapping tables A-12, B-5

H

HandWheel E-STOP 4-1

I

I/O addresses ..... *See* memory addresses
I/O declaration 3-5
I/O signals 5-3, 6-2
IL format 1-3, 1-4, 4-1
infinite number of relays 3-14
initializing
ServoWorksPLC Application Suite . 2-1, 2-2
input signals to the PLC Engine 5-3
input/output ..... *See* I/O
installing
ServoWorksPLC Application Suite . 2-1, 2-2
Instruction List format 1-3, 1-4
Instruction List Format 4-1
instruction sequence execution 5-4
interface settings 3-5
interface table 3-5
intermediate results 8-2
internal relay 6-4

J

JMP command 9-51
JMPE command 9-54

K

keep relay addresses 6-6, 7-2
keep relays, setting up 3-6

L

ladder diagram 1-3
address format 3-11
address signal letters 3-11
commenting 3-12
format 4-1
relay junctions 3-14
rows 3-13
signal names 3-12
symbols 3-13
ladder files 4-1
ladder logic – Fanuc-compatible 1-4

M

machine commands ..... *See* functional commands
mapping tables
F data A-2, B-2
G data A-12, B-5
overview A-1, B-1
X data A-14, B-8
Y data B-24
MC-Quad 2-1
memory addresses 6-1
alarm relay 6-8
counter addresses 6-5
data table 6-6
description 6-1
internal relay 6-4
keep relay and static memory control 6-6
related to the machine tool 6-3
related to the ServoWorks Motion Engine 6-3
signal types 6-2
specifications 6-2
timer 6-7
types of 6-1
memory,static 7-1
module files 4-4
Module Selection Window 3-10
Monitor/Debugger 1-3, 3-1
Motion Engine 1-1
MotionPro 2-1
MOVE command 9-43
MOVOR command 9-46
MUL command 9-91
MULB command 9-94

N

NUME command 9-101
NUMEB command 9-103

numerical data addresses..... 9-8

O

operation data..... 9-5
OR command..... 8-11
OR.NOT command..... 8-11
OR.STK command..... 8-15
order of execution..... 5-1
output signals from the PLC Engine.. 5-3

P

parameters..... 9-5
PARI command..... 9-55
PLC
  cycle..... 1-2
  data table..... 7-3
  description of..... 1-1
  instruction sequence execution..... 5-4
  scan..... 1-2
  scan time..... 1-3
PLC Bit Pattern Utility..... 3-16
PLC Control Screen..... 2-4, 3-3
PLC data table..... 7-4
PLC Diagnose Window .. 3-9, 3-10, 3-15
PLC Engine..... 1-1
  description..... 1-3
  input signals..... 5-3
  integration with Motion Engine..... 1-3
  integration within the ServoWorks system..... 1-2
  output signals..... 5-3
  schematic representation of sequence program execution..... 5-5
PLC Ladder Compiler Screen..... 2-4, 3-4
PLC Ladder Monitor/Debugger search function..... 3-15
PLC sequence programs .... *See* sequence programs
PLC Table Setting Screen..... 3-5
PLC Table Setting Screen Dialog Box 3-6
PLC Time Chart Utility..... 3-17
preset counter..... 9-20, 9-23, 9-26
PRM..... 9-5
programmable logic control..... *See* PC

R

RD command..... 8-4
RD.NOT command..... 8-6
RD.NOT.STK command..... 8-13
RD.STK command..... 8-11
relay junctions..... 3-14
relays,infinite number of..... 3-14
repetitive sampling..... 5-2
Result History Register..... 8-2
  structure..... 8-2
ring counter..... 9-20, 9-26
ROT command..... 9-29
ROTB command..... 9-33
rows in a ladder diagram..... 3-13
RUN status..... 3-2

S

S-100M..... 2-1
S-100T..... 2-1
sampling, repetitive..... 5-2
scan time..... 1-3
scan, PLC..... 1-2
search function, PLC Ladder Monitor/Debugger..... 3-15
Select Module Definition Window.... 2-5
sequence programs
  coding..... 4-6
  compiling..... 1-3
  debugging..... 3-9
  description..... 1-4, 4-1
  editing..... 1-3
  execution of..... 1-2
  execution time..... 5-2
  execution, schematic representation of..... 5-5
  format..... 1-4, 4-1
  repetitive sampling..... 5-2
  sequence of processing..... 5-1
  setting up..... 4-1, 4-5, 4-6
  verifying..... 1-3, 3-9
sequence programs, compiling..... 3-4
sequential processing of a sequence program..... 5-1
ServoWorks Motion Engine..... 1-1
ServoWorks PLC Engine..... 1-1
  description..... 1-3

input signals 5-3
integration with Motion Engine 1-3
integration within the ServoWorks system 1-2
output signals 5-3
schematic representation of sequence program execution 5-5
ServoWorksPLC application suite 1-1
composition of 1-3
ServoWorksPLC Application Suite ... 3-1
ServoWorksPLC Control Console Application 1-3, 3-1
ServoWorksPLC installation 2-1, 2-2
ServoWorksPLC Monitor/Debugger 1-3, 3-1
ServoWorksPLC Utility Tools 1-3, 1-4
setting up
counters 3-8
keep relays 3-6
tables 3-5
timers 3-7
setting up a sequence program ... 4-1, 4-5, 4-6
SFT command 9-68
signal addresses 8-2
signal names in a ladder diagram 3-12
signals ..... *See* I/O signals
stack register *See* Result History Register
starting the PLC Control Console application 3-2
static memory 6-5, 6-7
counter 7-1, 7-2
data table 7-2
keep relay 7-2
overview 7-1
reading and writing 7-3
timer 7-1
static memory control addresses 6-6
STOPPED status 3-3
storing the results of logic operations in the result history register 8-2
SUB command 9-86
SUBB command 9-89
symbols in a ladder diagram 3-13

T

tables, setting up 3-5
Time Chart Window 3-17
time charts 1-4, 3-1, 3-17
timer 7-1
timer addresses 6-7
timers, setting up 3-7
TMR command 9-10
TMRB command 9-12
TMRC command 9-14
Two's Complement notation 9-7

U

uninstalling the ServoWorksPLC Application Suite 2-6
up/down counter 9-20, 9-26
using the PLC Bit Pattern Utility 3-16
using the PLC Time Chart Utility 3-17
Utility Tools 1-3, 1-4

V

verifying sequence programs 1-3, 3-9

W

W1 9-5
WRT command 8-8
WRT.NOT command 8-9

X

X data mapping tables A-14, B-8
XMOV command 9-76
XMOVB 7-3
XMOVB command 9-80

Y

Y data mapping tables B-24