Soft Servo
SYSTEMS, INC

LadderWorks PLC Programming Language (Basic Commands and Function Blocks)

Revision 1.31

Warning

The product described herein has the potential – through misuse, inattention, or lack of understanding – to create conditions that could result in personal injury, damage to equipment, or damage to the product(s) described herein. Machinery in motion and high-power, high-current servo drives can be dangerous; potentially hazardous situations such as runaway motors could result in death; burning or other serious personal injury to personnel; damage to equipment or machinery; or economic loss if procedures aren't followed properly. Soft Servo Systems, Inc. assumes no liability for any personal injury, property damage, losses or claims arising from misapplication of its products. In no event shall Soft Servo Systems, Inc. or its suppliers be liable to you or any other person for any incidental collateral, special or consequential damages to machines or products, including without limitation, property damage, damages for loss of profits, loss of customers, loss of goodwill, work stoppage, data loss, computer failure or malfunction claims by any party other than you, or any and all similar damages or loss even if Soft Servo Systems, Inc., its suppliers, or its agent has been advised of the possibility of such damages.

It is therefore necessary for any and all personnel involved in the installation, maintenance, or use of these products to thoroughly read this pamphlet and related manuals and understand their contents. Soft Servo Systems, Inc. stands ready to answer any questions or clarify any confusion related to these products in as timely a manner as possible.

The selection and application of Soft Servo Systems, Inc.'s products remain the responsibility of the equipment designer or end user. Soft Servo Systems, Inc. accepts no responsibility for the way its controls are incorporated into a machine tool or factory automation setting. Any documentation and warnings provided by Soft Servo Systems, Inc. must be promptly provided to any end users.

This document is based on information that was available at the time of publication. All efforts have been made to ensure that this document is accurate and complete. However, due to the widely varying uses of this product, and the variety of software and hardware configurations possible in connection with these uses, the information contained in this manual does not purport to cover every possible situation, contingency or variation in hardware or software configuration that could possibly arise in connection with the installation, maintenance, and use of the products described herein. Soft Servo Systems, Inc. assumes no obligations of notice to holders of this document with respect to changes subsequently made. Under no circumstances will Soft Servo Systems, Inc. be liable for any damages or injuries resulting from any defect or omission in this manual.

Soft Servo Systems, Inc. makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. NO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS OF PURPOSE SHALL APPLY.

Important Notice

The information contained in this manual is intended to be used only for the purposes agreed upon in the related contract with Soft Servo Systems, Inc. All material contained herein is subject to restricted rights and restrictions set forth in the contract between the parties.

These manuals contain confidential and proprietary information that is not to be shared with, nor distributed to, third parties by any means without the prior express, written permission of Soft Servo Systems, Inc. No materials contained herein are to be duplicated or reproduced in whole or in part without the express, written permission of Soft Servo Systems, Inc.

Although every effort and precaution has been taken in preparing this manual, the information contained herein is subject to change without notice. This is because Soft Servo Systems, Inc. is constantly striving to improve its products. Soft Servo Systems, Inc. assumes no responsibility for errors or omissions.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview

The LadderWorks Console is a soft PLC application for creating PLC sequence programs that are executed by the LadderWorks PLC Engine, providing machine control that is totally integrated with motion control. The basic building blocks of a PLC sequence program are written using PLC basic commands and LadderWorks function blocks, which are detailed in this manual.

You can code a sequence program in either Ladder Diagram (LD) format or Instruction List (IL) format. You can create, edit and verify sequence programs with the LadderWorks Console. The Instruction List form consists of only text. The ladder diagram format consists of relay junctions and functional command symbols described later in this document. When you are satisfied with your code, compile your IL or LD sequence program into machine code with the LadderWorks Console application. This machine code is then fed into the LadderWorks PLC Engine as the sequence program, and you can verify that the logic represented by your sequence program is what you intended.

You will be 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 need a thorough understanding of PLC basic commands to understand the details of the functional commands presented later. Therefore, you should read carefully through this entire document before coding your sequence program for your Soft Servo system.

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, also known as “function blocks,” are commands that make the programming of the complex controls of machinery much easier than just using basic commands.

For basic commands and function blocks, we must concern ourselves also with signal addresses, and storing the results of logical operations.

1.3 Signal Addresses

An address can be assigned to all signals, relay coils and junctions, drawn in the ladder diagram shown in the following figure. An address consists of a letter, an address number and a bit number. The leading zero can optionally be suppressed. There are addresses for X, Y, F, G, R, C, K, T, D and A signals. For more details about signal addresses, see the *LadderWorks PLC Reference Manual*.

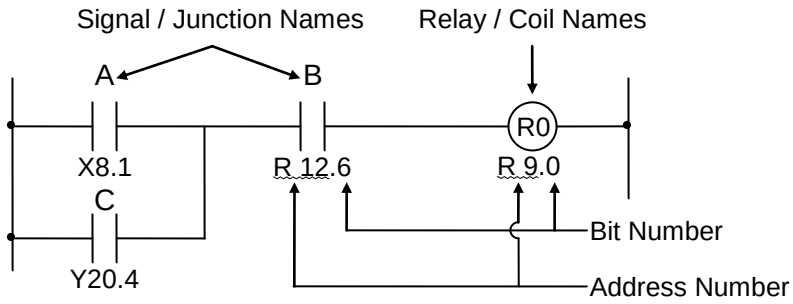


Figure 1-1: Signal Addresses

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

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

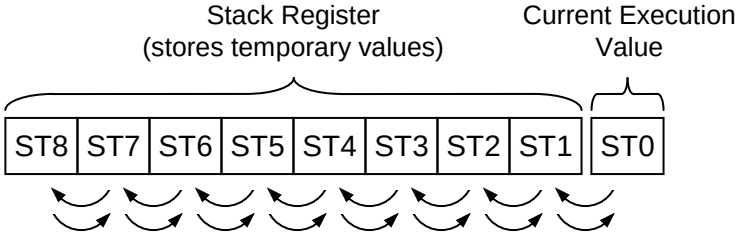


Figure 1-2: Structure of the Result History Register

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 the above figure. 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 manual.

Both the PLC basic commands and the PLC functional commands use the Result History Register. [NOTE: Functional commands also use the Functional Command Register.]

Chapter 2: PLC Basic Commands

This chapter concerns itself with basic PLC commands.

2.1 Summary of Basic Commands

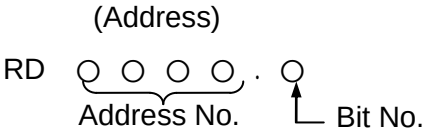
The following table shows the kinds of basic commands and their functions. Detailed descriptions follow.

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 2-1: PLC Basic Commands and Their Functions

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 2-1 and the coding sheet entry example in Table 2-2.

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 2-2 and 2-3 show two alternative ways to code the ladder diagram in Figure 2-1, in different orders. Both orders lead to the same result.

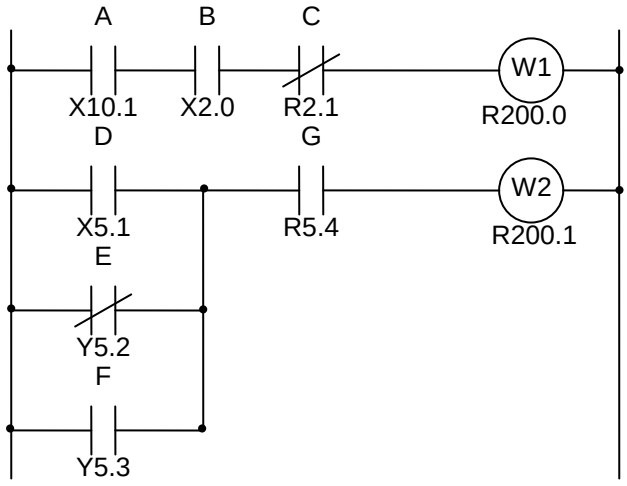


Figure 2-1: Ladder Diagram Example for the RD Command

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				$A \cdot B \cdot \overline{C}$
4	WRT	R200 . 0	W1 out			$A \cdot B \cdot \overline{C}$
5	RD	X5 . 1				D
6	OR.NOT	Y5 . 2				$D + \overline{E}$
7	OR	Y5 . 3				$D + \overline{E} + F$
8	AND	R5 . 4				$(D + \overline{E} + F) \cdot G$
9	WRT	R200 . 1	W2 out			$(D + \overline{E} + F) \cdot G$

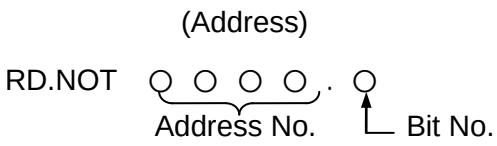
Table 2-2: Coding of the RD Command Example (Alternative #1)

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				$B \cdot A \cdot \overline{C}$
4	WRT	R200 . 0	W1 out			$B \cdot A \cdot \overline{C}$
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				$(F + \overline{E} + D) \cdot G$
9	WRT	R200 . 1	W2 out			$(F + \overline{E} + D) \cdot G$

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

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 2-2 and the coding sheet entry example in Table 2-4.

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 2-4 and 2-5 show two alternative ways to code the ladder diagram in Figure 2-2, in different orders. Both orders lead to the same result.

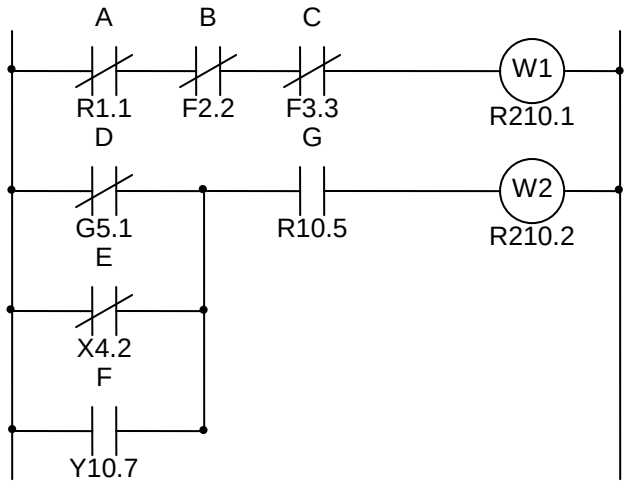


Figure 2-2: Ladder Diagram Example for the RD.NOT Command

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			$(\overline{D} + \overline{E} + F) \cdot G$

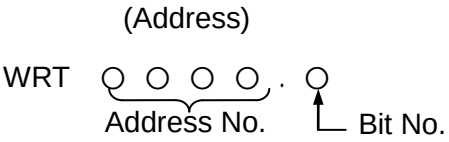
Table 2-4: Coding of the RD.NOT Command Example (Alternative #1)

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

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 2-2, 2-3, 2-4 and 2-5 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 2-3 and Table 2-6 for examples of how to use the WRT command.

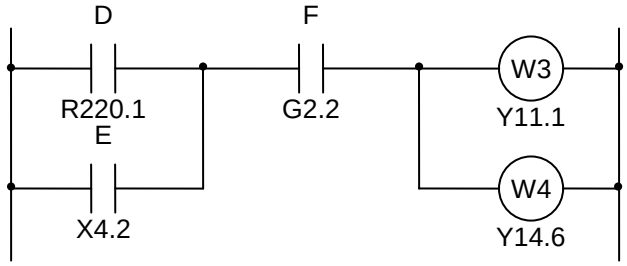


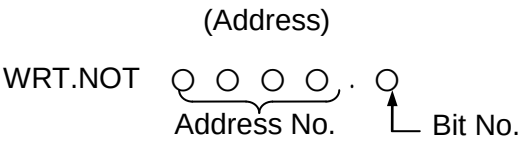
Figure 2-3: Ladder Diagram Example for the WRT Command

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 2-6: Coding of the WRT Command

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 2-4 and Table 2-7 for an example of how to use the WRT.NOT command.

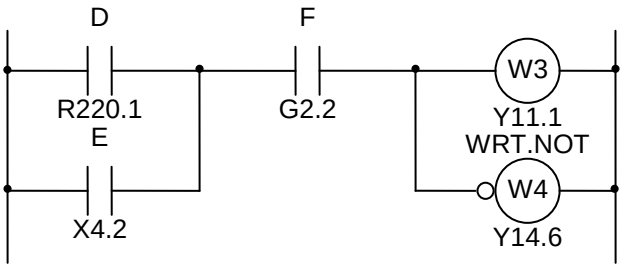


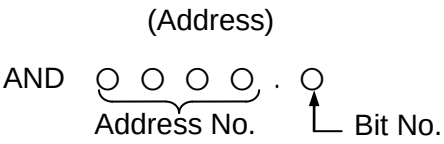
Figure 2-4: Ladder Diagram Example for the WRT.NOT Command

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.NOT	Y14 . 6	W4 out			$\overline{(D + E) • F}$

Table 2-7: Coding of the WRT.NOT Command

2.6 AND Command

Format



Function

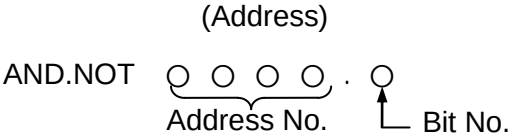
This command takes the value of a specified signal and executes the logical AND (product) with the existing value.

Example of AND Command Usage

See Figure 2-1 and Table 2-2 for an example using the AND command.

2.7 AND.NOT Command

Format



Function

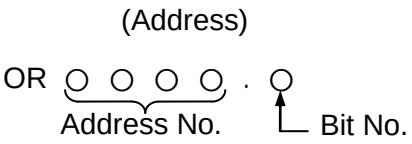
This command inverts the value of a specified signal and executes the logical AND.NOT (inverse product) with the existing value.

Example of AND.NOT Command Usage

See Figure 2-1 and Table 2-2 for an example of how to use the AND.NOT command.

2.8 OR Command

Format



Function

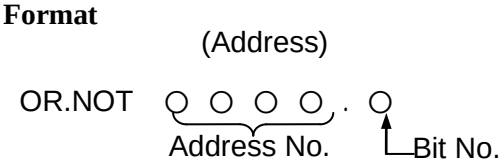
This command takes the value of a specified signal and executes the logical OR (sum) with the existing value.

Example of OR Command Usage

See Figure 2-1 and Table 2-2 for an example using the OR command.

2.9 OR.NOT Command

Format



Function

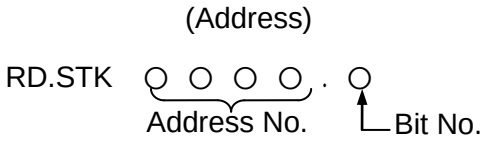
This command inverts the value of a specified signal and executes the logical OR.NOT (inverse sum) with the existing value.

Example of OR.NOT Command Usage

See Figure 2-1 and Table 2-2 for an example of how to use the OR.NOT command.

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 2-5 and Table 2-8 for examples of how to use the RD.STK command.

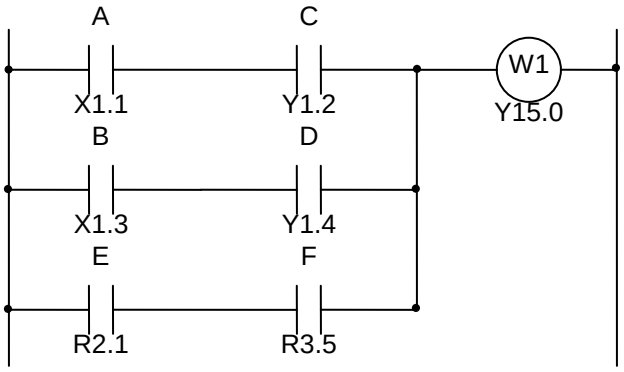


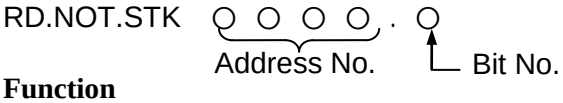
Figure 2-5: Ladder Diagram Example for the RD.STK Command

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 2-8: Coding of the RD.STK Command

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 2-6 and Table 2-9 for examples of how to use the RD.NOT.STK command.

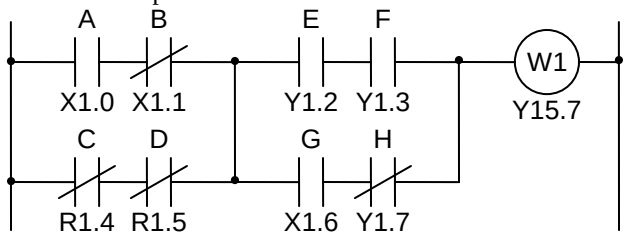


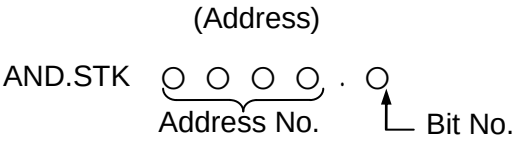
Figure 2-6: Ladder Diagram Example for the RD.NOT.STK Command

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 2-9: Coding of the RD.NOT.STK Command

2.12 AND.STK Command

Format



Function

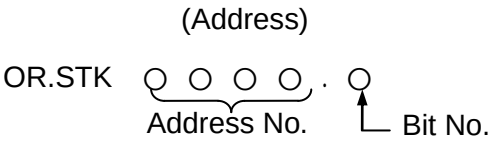
This command sets the logical product of ST0 and ST1 into ST1, and shifts the bits of the register one bit to the right to put the result into ST0.

Example of AND.STK Command Usage

See Figure 2-6 and Table 2-9 for examples on how to use the AND.STK command.

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 2-5 and Table 2-8, and Figure 2-6 and Table 2-9 for examples using the OR.STK command.

Note

In the example shown in Table 2-8, the results are the same even if the OR.STK in step number 5 is moved between steps number 7 and number 8.

Chapter 3: PLC Functional Commands

3.1 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 "function blocks" or "machine commands"). The different types of functional commands and their descriptions are shown in Tables 3-1 and 3-2.

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	DCNV	SUB 14	Data Conversion
13	DCNVB	SUB 31	Extended Data Conversion
14	MOVE	SUB 8	Masked Data Transfer
15	MOVOR	SUB 28	Bit-Wise Sum Data Transfer
16	SFT	SUB 33	Shift Register
17	JMP	SUB 10	Jump
18	JMPE	SUB 30	Jump Termination
19	COM	SUB 9	Common Line Control
20	COME	SUB 29	Common Line Control Termination
21	PARI	SUB 11	Parity Check
22	COMP	SUB 15	Comparison
23	COMPB	SUB 32	Binary Comparison

Table 3-1: Summary of Functional Commands (1 of 2)

No.	Command		Description
	Ladder Format	Code Format	
24	COIN	SUB 16	Equality Check
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 3-2: Summary of Functional Commands (2 of 2)

⚠ CAUTION

The command format and the general usage are described at the beginning of each functional command description in this document. Important information, such as the functional command specifications, is included in this document, and should be reviewed carefully for each functional command.

3.2 Functional Command Format

Functional commands cannot be described by relay symbols, so they are written in a different format (see Figure 3-1). 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.

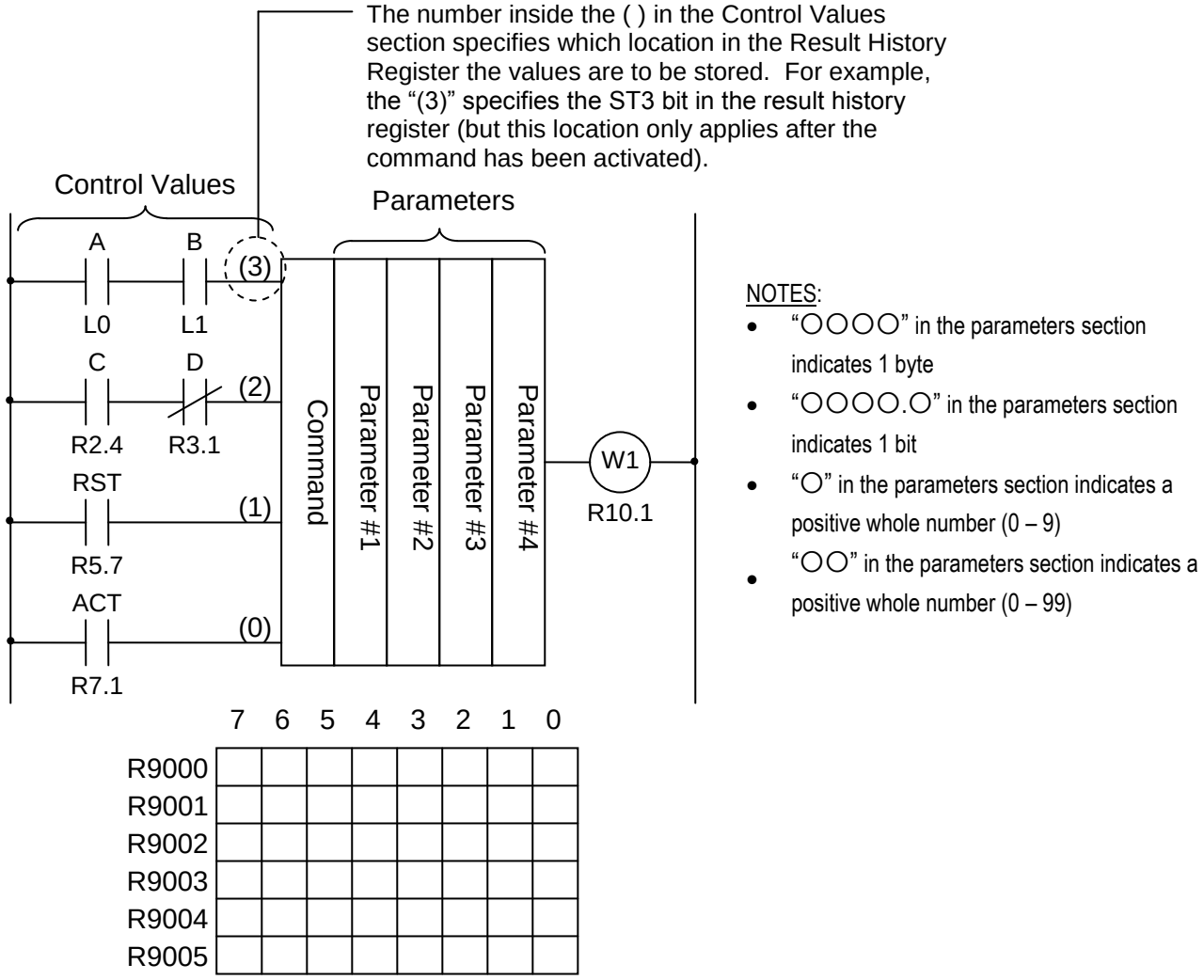


Figure 3-1: Functional Command Format – Ladder Diagram and Functional Command Register

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	○○	Command	A • B	C • $\overline{D}$	RST	ACT
8		○○○○	Param 1	A • B	C • $\overline{D}$	RST	ACT
9		○○○○	Param 2	A • B	C • $\overline{D}$	RST	ACT
10		○○○○	Param 3	A • B	C • $\overline{D}$	RST	ACT
11		○○○○	Param 4	A • B	C • $\overline{D}$	RST	ACT
12	WRT	R10 . 1	W1 Out	A • B	C • $\overline{D}$	RST	W1

Table 3-3: Functional Command Format – Coding

3.3 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 3-3, 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.

3.4 Command

The different kinds of commands are as shown in Tables 3-1 and 3-2.

3.5 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.

3.6 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.

3.7 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 Soft Servo Systems products, the format of the data transferred (codes M, S, T and B) between the ServoWorks CNC Engine/SMP Motion Engine ↔ LadderWorks 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 *Section 3.8: Numerical Data Examples*.

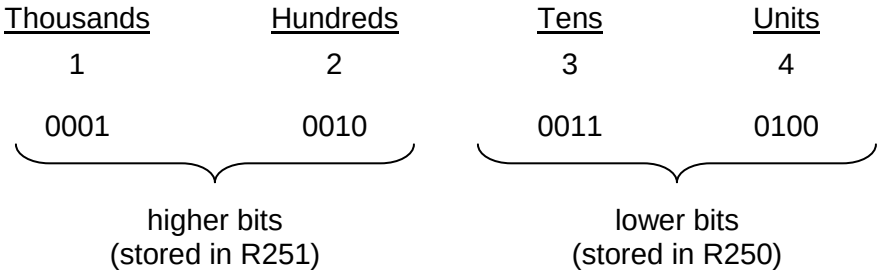
For these reasons, functional commands usually use binary data.

3.8 Numerical Data Examples

3.8.1 BCD Format Data

The BCD (binary coded decimal) number system converts decimal numbers to BCD as follows: each digit of the decimal number is converted to the binary numbers 0000 through 1001. [The binary patterns 1010 through 1111 are not valid in this instance.]

For example, the number 1,234 is converted as follows:



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. [It is four digits so it requires 2 bytes of storage.] The lower bits get stored inside R250, and the higher bits get stored inside R251:

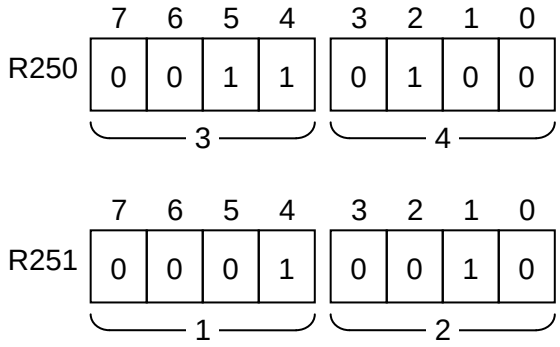


Figure 3-2: Example of 4-Digit BCD Format Data

3.8.2 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 the following figure:

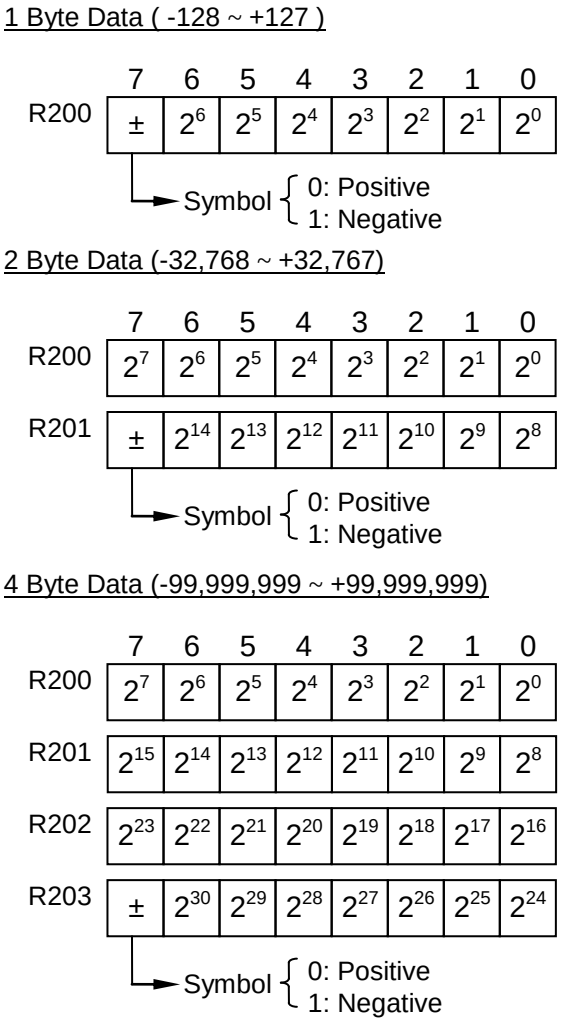


Figure 3-3: Memory Storage of Binary Format Data

Functional commands using binary format data use the starting address R200 to specify 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.

Here is an 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.

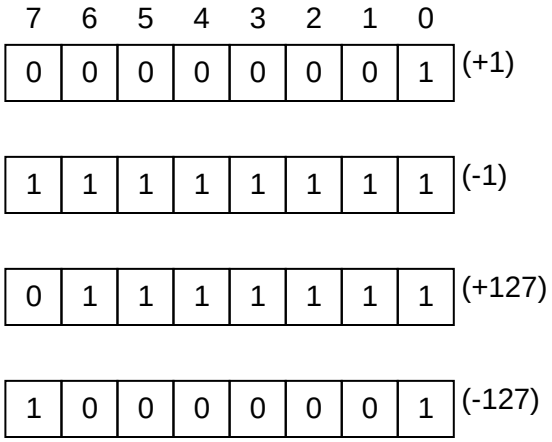


Figure 3-4: Examples of Binary Format Data for 1 Byte Data

3.9 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.

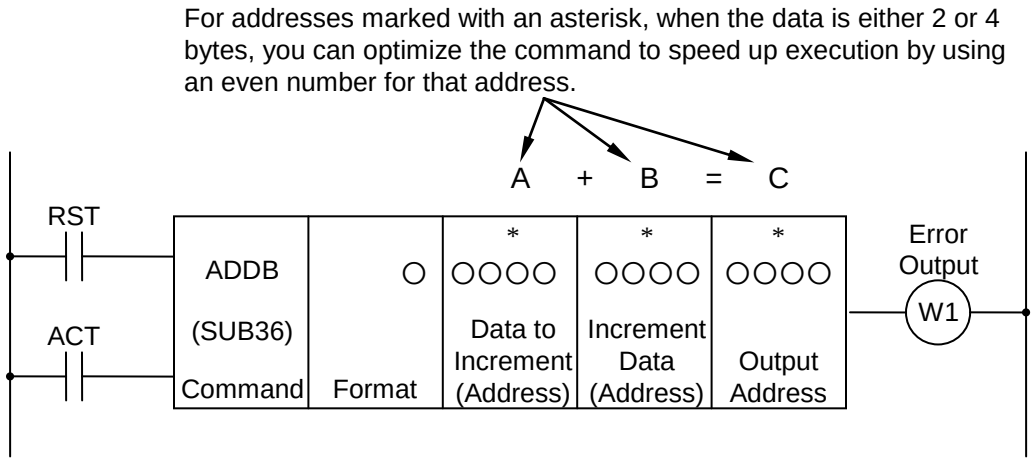


Figure 3-5: Addresses of Numeric Data

3.10 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.

The register is shared among programs, 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 (unlike the Result History Register, which stores values that can be read or written to).

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

Figure 3-6: 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 R9000.0.

Chapter 4: Timer Function Blocks

4.1 TMR (Timer)

Function

This timer is an on-delay timer.

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

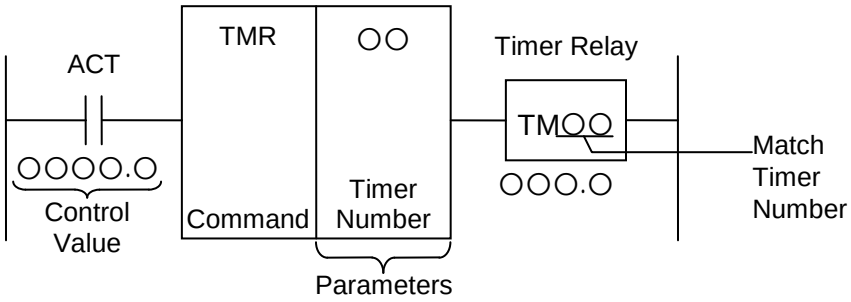


Figure 4-1: Format for the TMR Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description	
1	RD	○○○○	○	ACT	control value
2	TMR	○○			command
3	WRT	○○○○	○	TM○○	timer relay

Table 4-1: 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 the following figure, when ACT = 1 for a specified time period, the timer relay turns on. You can freely choose the timer relay address.

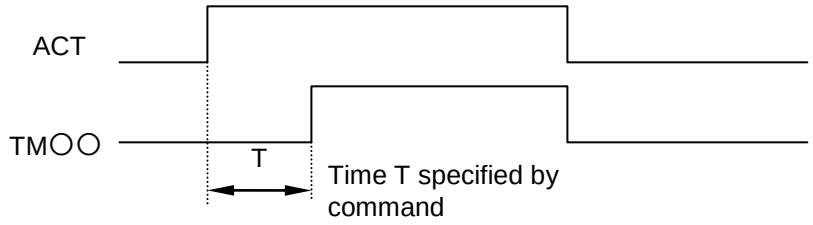


Figure 4-2: Timer Behavior for the TMR Command

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.

Code Example

Refer to the code example in *Section 9.3: SFT (Shift Register)*.

4.2 TMRB (Fixed Timer)

Function

This timer is an on-delay timer where the time is fixed. Because the timer described by *Section 4.1: 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

The following figure shows the format for describing the command.

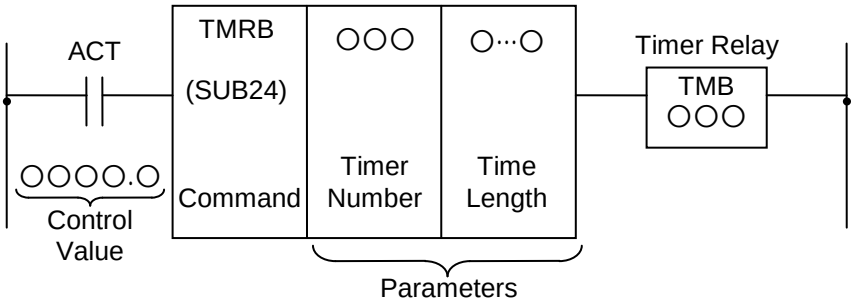


Figure 4-3: Format for the TMRB Command

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 the following figure, when ACT = 1 for a specified time period, the timer relay turns on. You can freely choose the timer relay address.

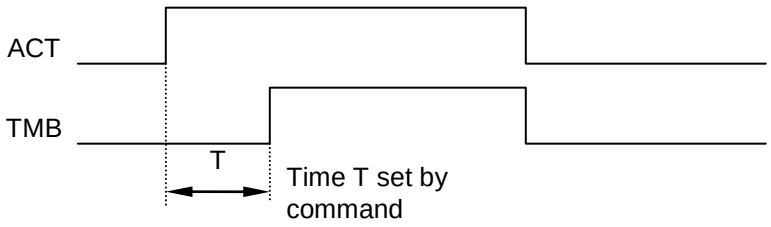


Figure 4-4: Timer Behavior for the TMRB Command

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, are not included.

Code Example

```
%@3
RD  X0.1
SUB 24
01
5000
WRT  R1.0

%
```

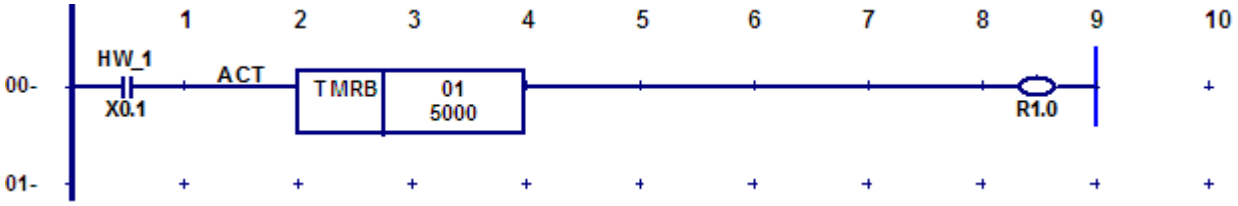


Figure 4-5: TMRB Code Example, Ladder View

4.3 TMRC (Timer)

Function

This timer is an on-delay timer.

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

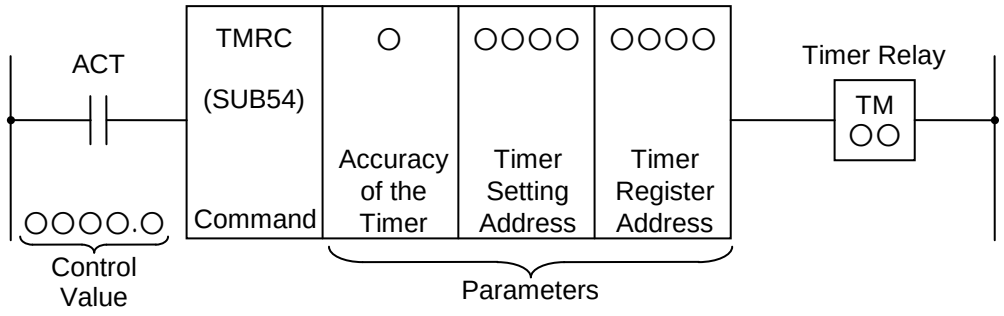


Figure 4-6: Format for the TMRC Command

Step No.	Command	Address No. Bit No.	Description	
1	RD	○○○○ . ○	ACT	control value
2	SUB	54	TMRC Command	command
3	(PRM)	○	Accuracy of the Timer	parameters
4	(PRM)	○○○○	Timer Setting Address	
5	(PRM)	○○○○	Timer Register Address	
6	WRT	○○○○ . ○	TM○○	timer relay

Table 4-2: Coding Format of the TMRC Command

Control Values

Action Command (ACT)

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

Parameters

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

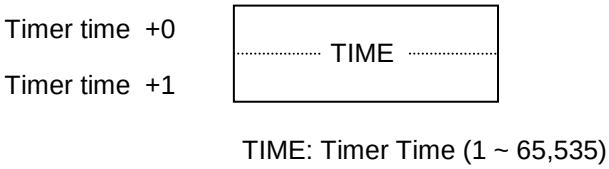


Figure 4-7: TMRC Address of the Time Set of the Timer

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

1 ms: 1 ~ 65,535 ms
8 ms: 8 ~ 524,280 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 4-8: Timer Register Address for the TMRC Command

Timer Relay (TM○○)

As shown in the following figure, after ACT = 1, the timer turns on, and after the specified time, the timer relay turns on.

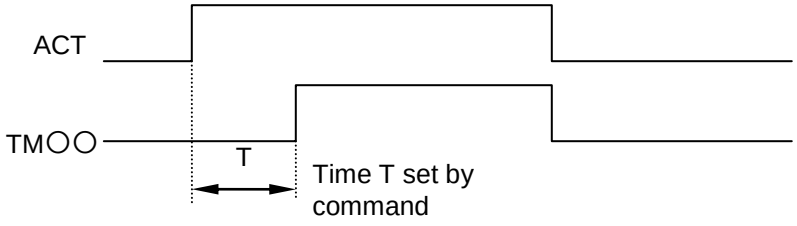


Figure 4-9: Timer Behavior for the TMRC Command

Code Example

```
%@3
RD  X0.1
SUB 54
0
D0
R2
WRT  R10.1

%
```

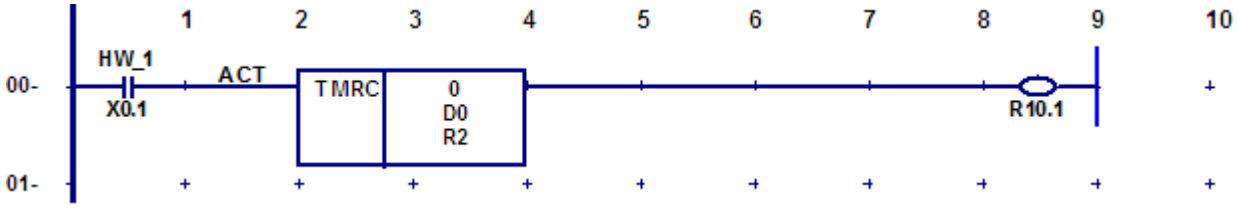


Figure 4-10: TMCR Code Example, Ladder View

Chapter 5: Decoding Function Blocks

5.1 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

The following figure shows the format for describing the command, and the following table shows the coding format.

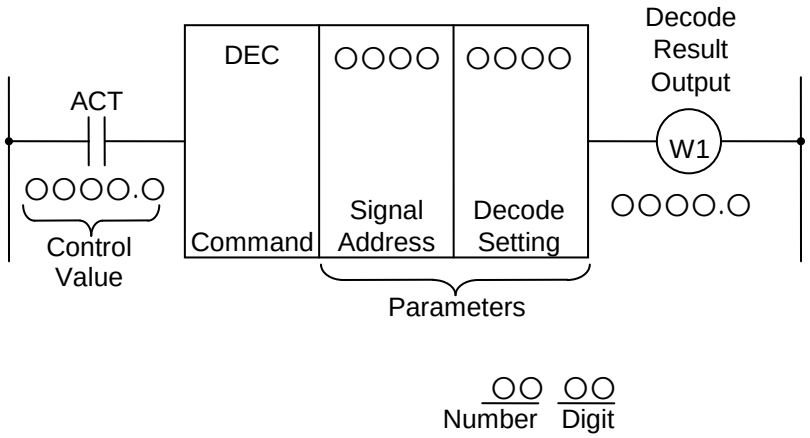


Figure 5-1: Format for the DEC Command

Decoding Sheet

Step No.	Command	Address No.	Bit No.	Description	
1	RD	OOO	O	ACT	control value
2	DEC				command
3	(PRM)	OOOO		Signal Address	parameters
4	(PRM)	OOOO		Decode Setting	
5	WRT	OOOO	O	W1, Decode Result Output	result

Table 5-1: 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 the following figure and the following table for an example of DEC command usage.

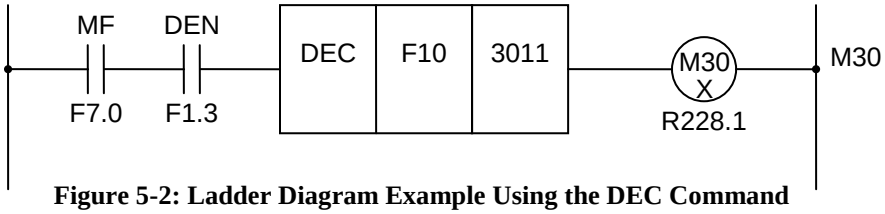


Figure 5-2: Ladder Diagram Example Using the DEC Command

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 5-2: Coding Example of the DEC Command

Code Example

```
%@3
RD     R0.0  // ACT=1, START DEC
DEC    D0    // 1 BYTE BCD
1501         // NUMBER + DIGIT
WRT    R1.0  // ERROR OUTPUT
%
```

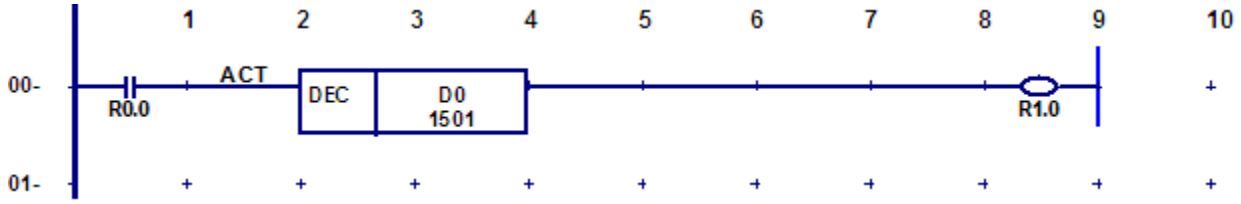


Figure 5-3: DEC Code Example, Ladder View

5.2 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.

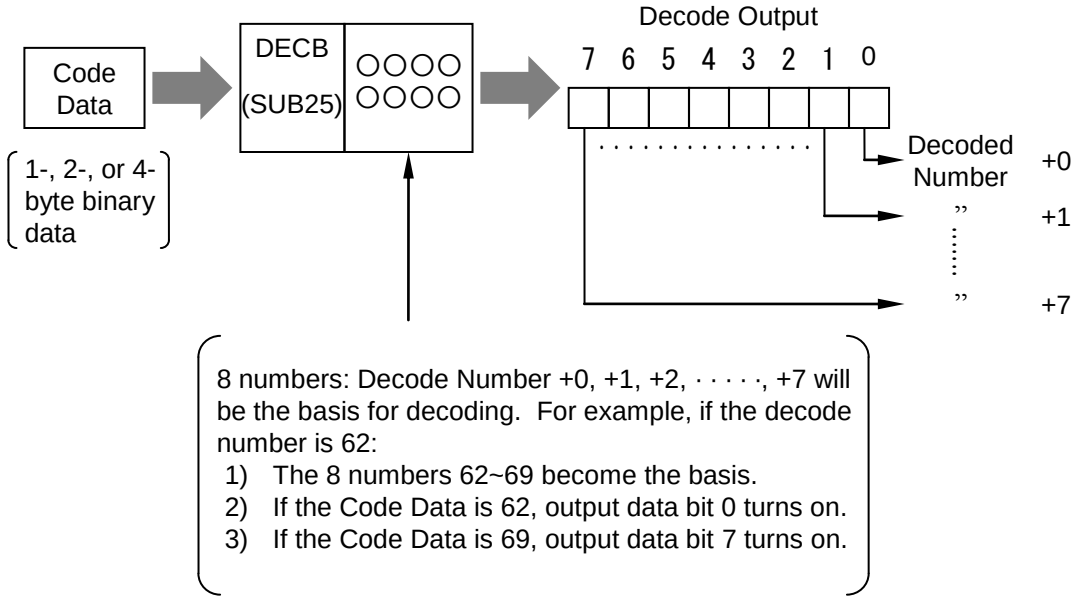
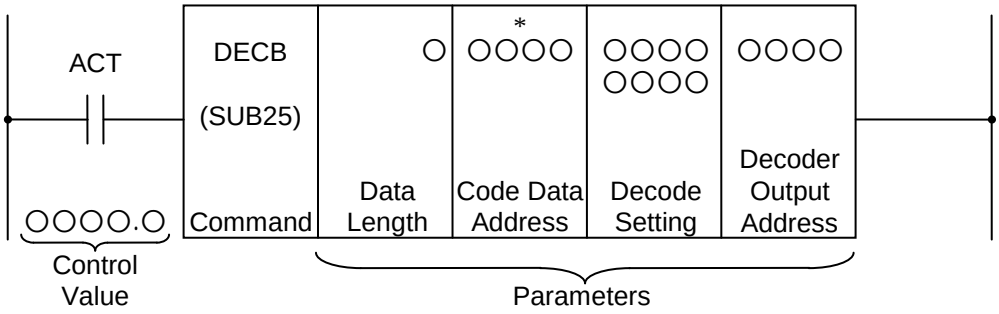


Figure 5-4: Function for the DECB Command

Format

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 5-5: 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.

Code Example

```
%@3

RD    R0.0  //read K0.0 -- the default value is "1"
SUB   25    //DECB command
1           //Using 1 byte data
D0          //Input data address
62          //Starting number of 8 decode numbers
R0

%
```

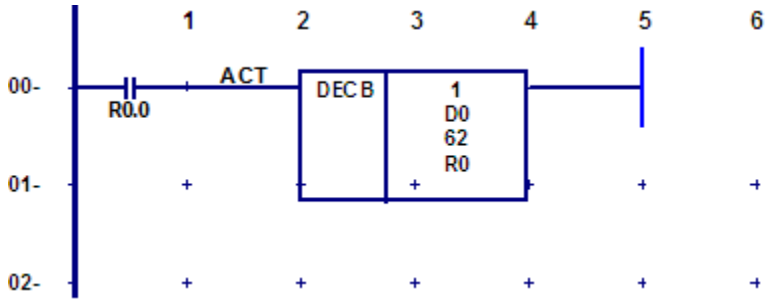


Figure 5-6: DECB Code Example, Ladder View

Chapter 6: Counter Function Blocks

6.1 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 the following figure. If you use the counter in this way, you can determine the location of a rotation object.

.

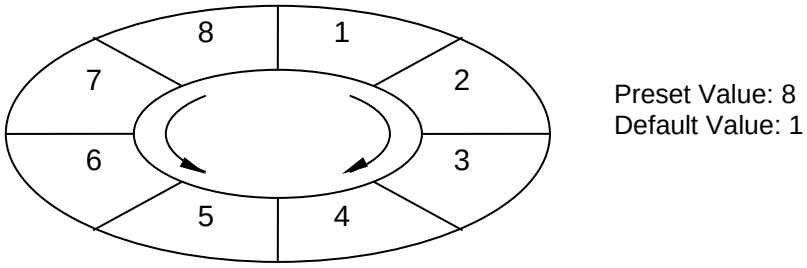


Figure 6-1: Ring Counter Created Using the CTR Command

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

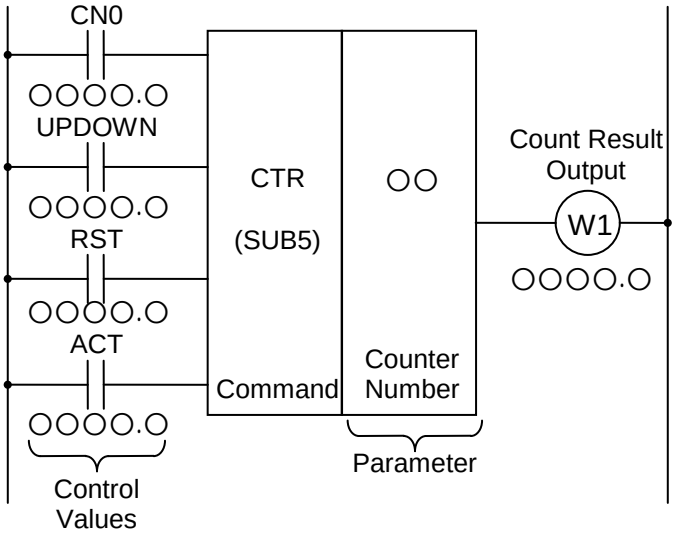


Figure 6-2: Format for the CTR Command

Coding Sheet

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

Result History Register

ST3	ST2	ST1	ST0	
			CN0	control values
		CN0	UPDOWN	control values
	CN0	UPDOWN	RST	control values
CN0	UPDOWN	RST	ACT	control values
CN0	UPDOWN	RST	ACT	command
CN0	UPDOWN	RST	ACT	parameter
CN0	UPDOWN	RST	W1	result

Table 6-1: 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.

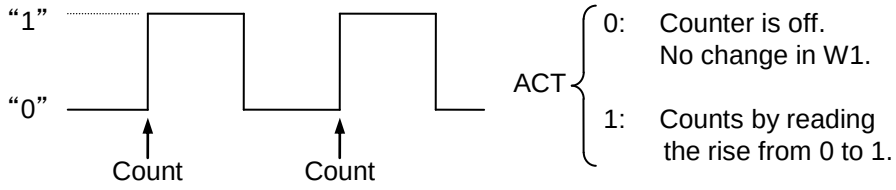


Figure 6-3: Count Signal (Action Command) for the CTR Command

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 6-4.)

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).

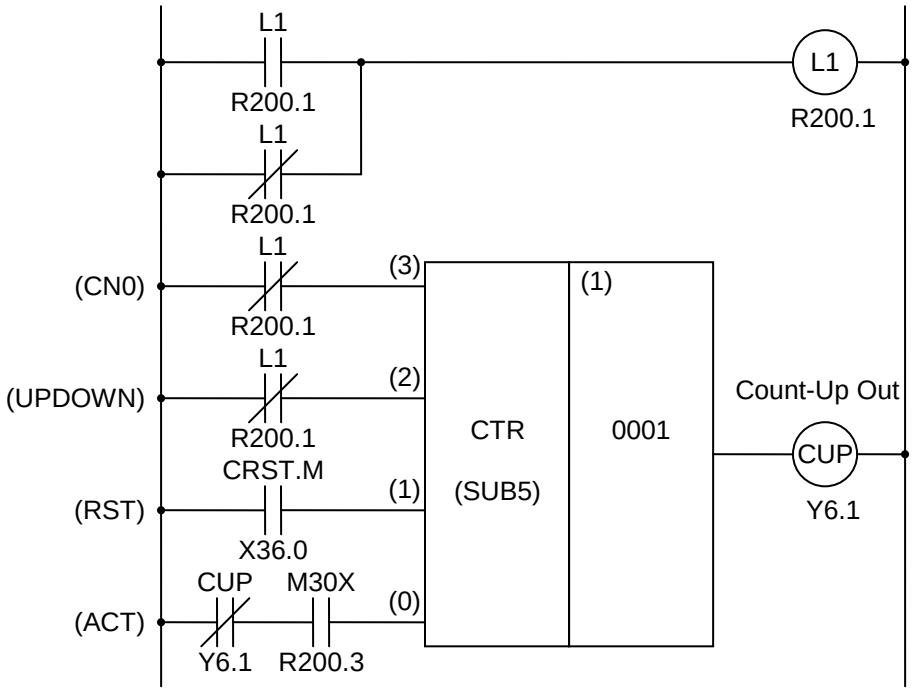


Figure 6-4: Ladder Diagram For Counter Example #1

Counter Example #2: Using a Counter In Order To Record The Location Of The Rotational Object (See Figures 6-5 and 6-6.)

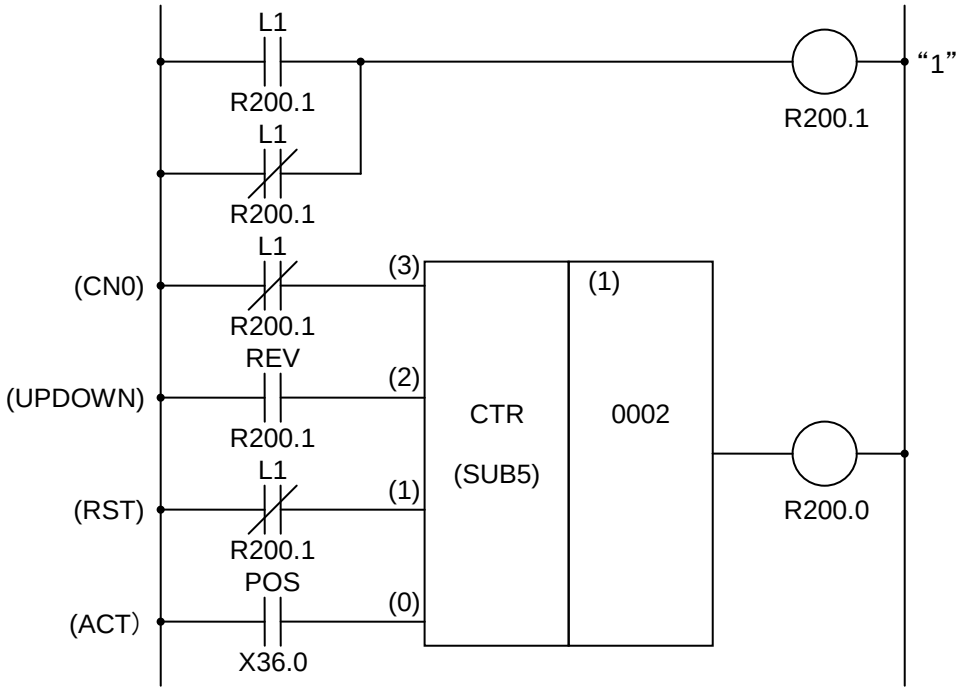


Figure 6-5: Ladder Diagram For Counter Example #2

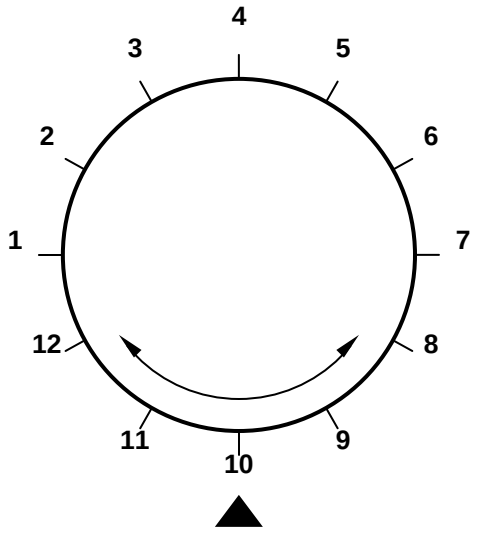


Figure 6-6: Division of a Rotational Body for Counter Example #2

Figure 6-5 is the ladder diagram that stores the location of the rotational body in Figure 6-6.

1) CONTROL VALUES

   a) Counter Starting Number: If you think about the rotational body with twelve angles in Figure 6-6, 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 clockwise direction, and "1" when turning in the counterclockwise direction. So, when the rotation is in the clockwise direction, the counter works as an up-counter, and when the rotation is in the counterclockwise 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 and OFF 12 times when the rotational body turns once.

2) COUNTER NUMBER AND W1

   As shown in this example, the result of W1 is not used elsewhere, but you must still designate an address for W1.

3) ACTION

   a) Specifying the Preset Value: The rotational body controlled here has 12 edges (Figure 6-6), so you must set the preset value of the counter as 12. To set the preset value to 12, use the LadderWorks PLC Setup Console:

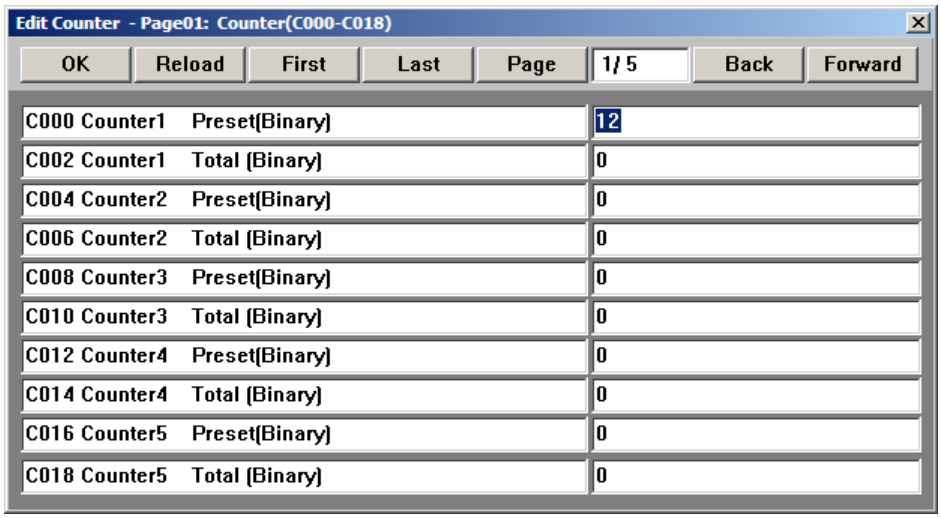


Figure 6-7: Counter Screen of the LadderWorks PLC Setup Console

NOTE: "4" indicates the preset value of Counter 4, not the C004 value (which is the preset value for Counter 2).

b) Specifying the Current Value: You must first synchronize the rotational body location with the current counter value. You can set this value in the LadderWorks PLC Setup Console. 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, · · · · · · · · · · ·

Code Example

```
%@3
RD    K0.0  //k0.0=0 1,2,3,4,5...
RD.STK      K0.1  //k0.1=0
RD.STK      X0.1  //reset
RD.STK      X1.0  //ACT
SUB 5
2
WRT R10.1

%
```

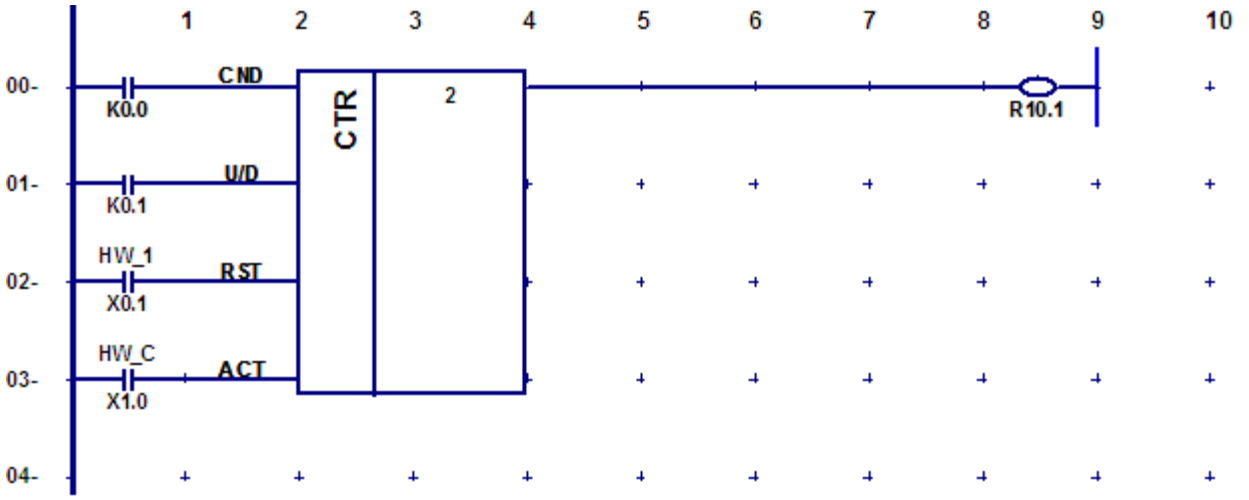


Figure 6-8: CTR Code Example, Ladder View

6.2 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

The following figure shows the format for describing the command, and the following table shows the coding format.

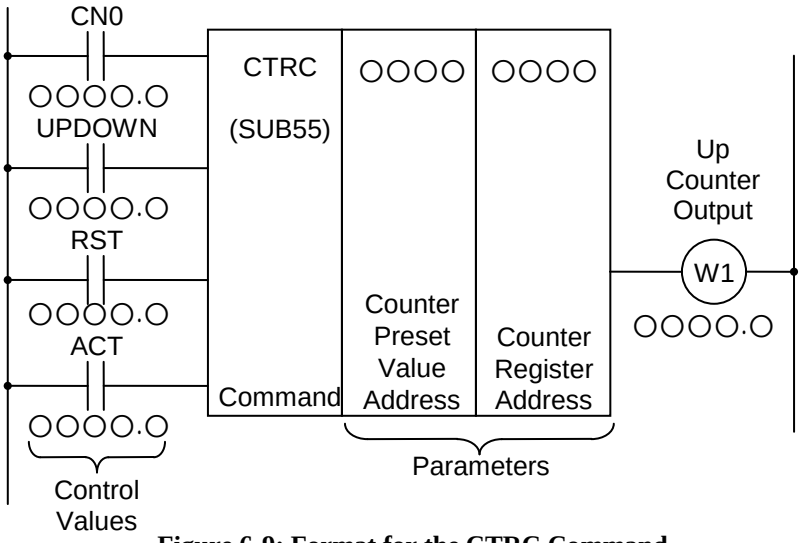


Figure 6-9: Format for the CTRC Command

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

Table 6-2: 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.

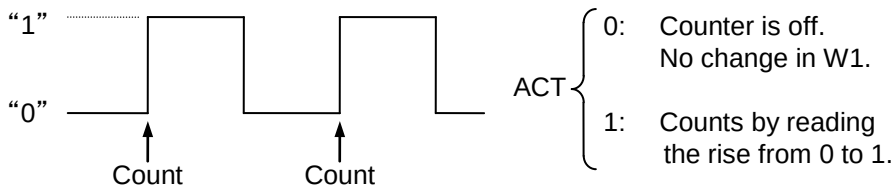


Figure 6-10: Count Signal (Action Command) for the CTRC Command

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.

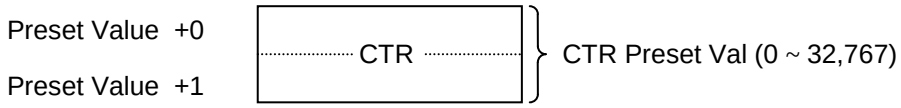


Figure 6-11: Address of the 2 Byte Counter Preset Value for the CTRC Command

The counter preset value is given in binary format, so you can specify values 0 ~ 32,767. The counter preset value corresponds to the counter table values in the Counter Screen of the LadderWorks PLC Setup Console. (See Figure 6-7.) NOTE: "4" indicates the preset value of Counter 4, not the C004 value (which is the preset value of Counter 2).

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.

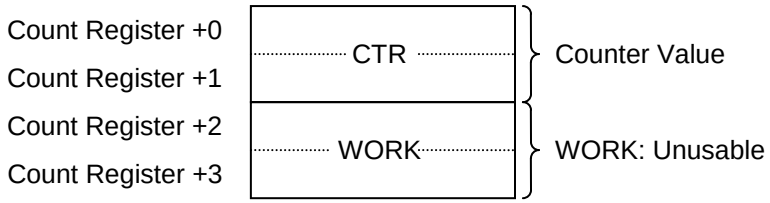


Figure 6-12: Address of the Up Counter Output for the CTRC Command

Code Example

```
%@3
RD R200.1
OR.NOT R200.1
WRT R200.1

RD.NOT R200.1
RD.NOT.STK R200.1
RD.STK X0.1
RD.STK X1.1
SUB 55
1
D5
WRT R10.1

%
```

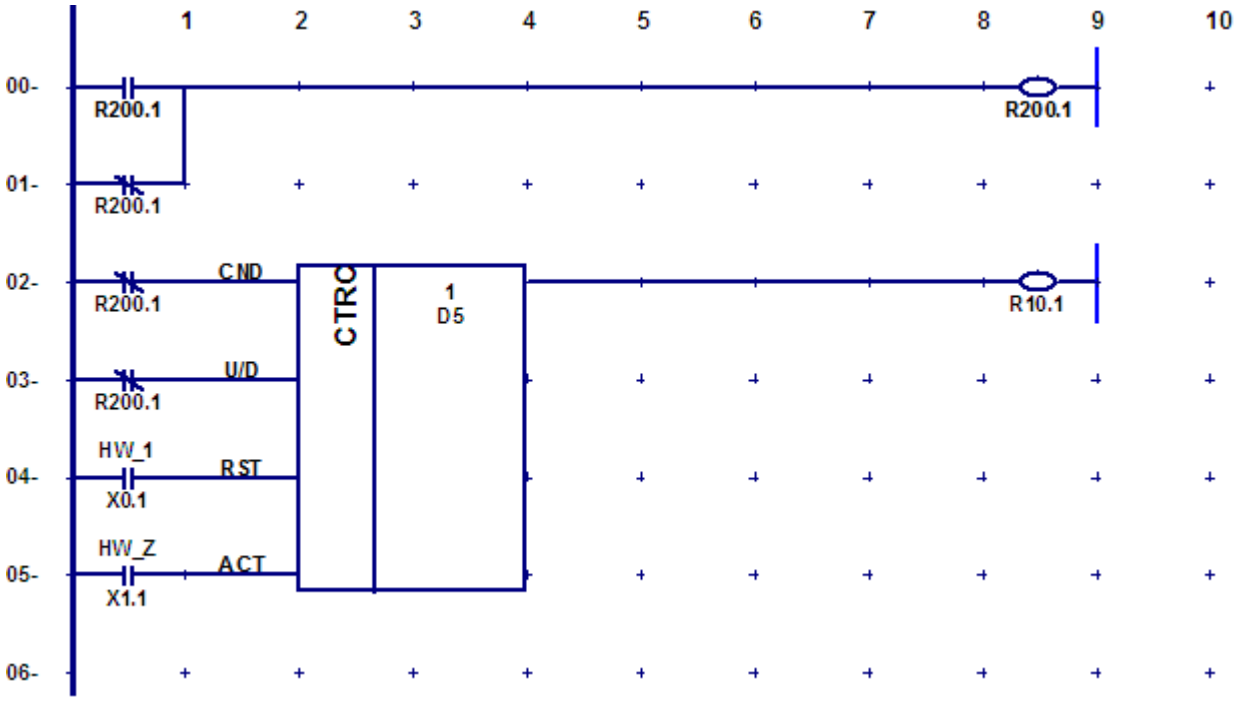


Figure 6-13: CTRC Code Example, Ladder View

Chapter 7: Rotational Control Function Blocks

7.1 ROT (Rotational Control)

Function

This command is used for controlling a rotational system like a rotational knife sharpener, auto tool changer (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

The following figure shows the format for describing the command, and the following table shows the coding format.

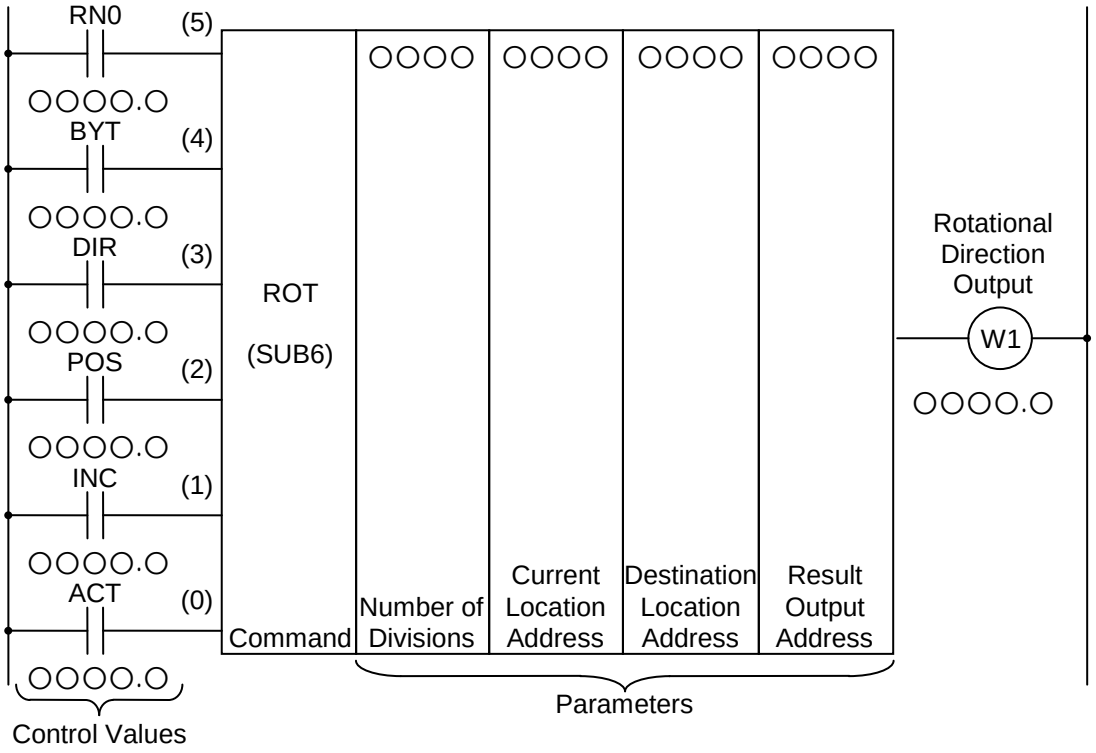


Figure 7-1: Format for the ROT Command

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	control values
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	command
8	(PRM)	OOOO	Number of Divisions	RN0	BYT	DIR	POS	INC	ACT	parameters
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	OOOO . O	W1, Rotational Direction Output	RN0	BYT	DIR	POS	INC	W1	result

Table 7-1: 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 below 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 the following figure for definitions of 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.

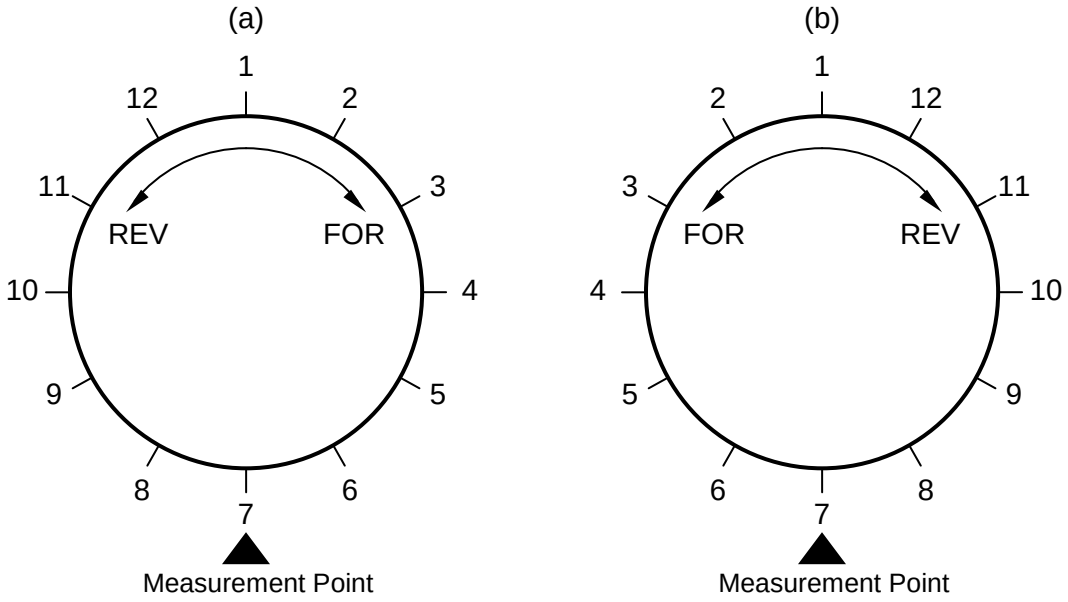


Figure 7-2: Rotation Direction Rule – 12-Division Example

Code Example

```
%@3
RD    R0.0  // RN0=0, START FROM 0
RD.STK      R0.0  // BYT=0, DATA IS 1 BYTE BCD
RD.STK      R0.0  // DIR=0, NO DIRECTION
RD.STK      R0.0  // POS=0
RD.STK      R0.0  // INC=0
RD.STK      R0.1  // ACT=1, RUN ROT COMMAND
SUB 6
4
X41
F26
R230
WRT  R228.1

%
```

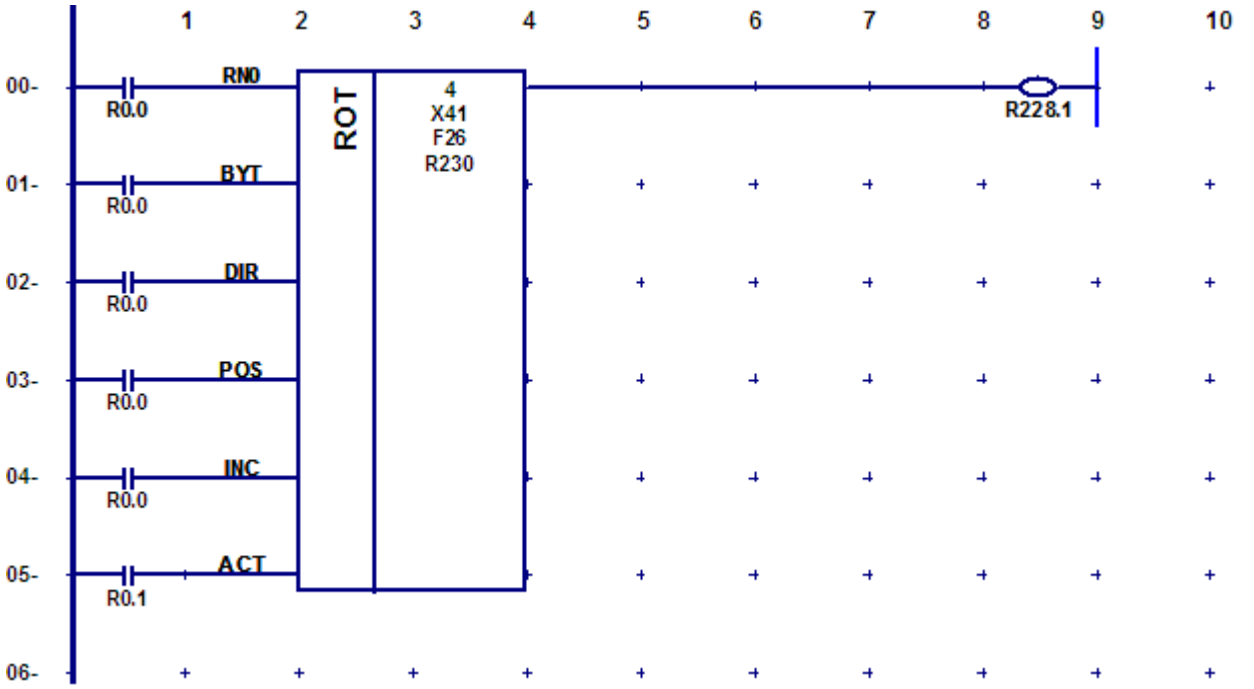


Figure 7-3: ROT Code Example, Ladder View

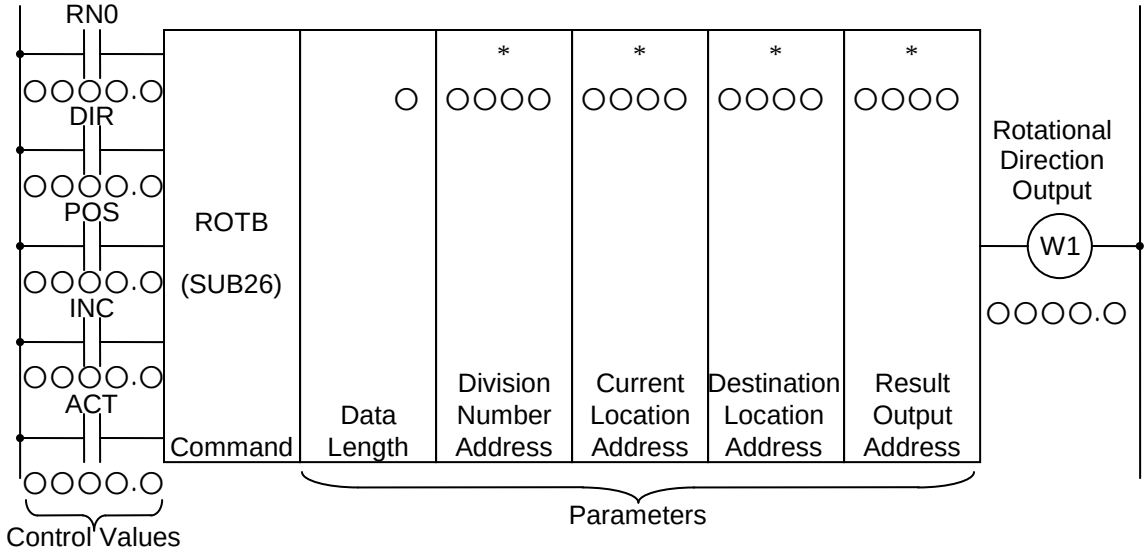
7.2 ROTB (Binary Rotational Control)

Function

This command is used for controlling a rotational body such as an auto tool changer (ATC) or a rotating table. In contrast to the ROT command (Section 7.1) 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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 7-4: 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 7-2 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 7-5.

- The destination is specified by the LadderWorks PLC 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 CNC Engine or the SMP Motion Engine.
- A comparator (COIN) is used to detect the speed reduction position and the stopping position.

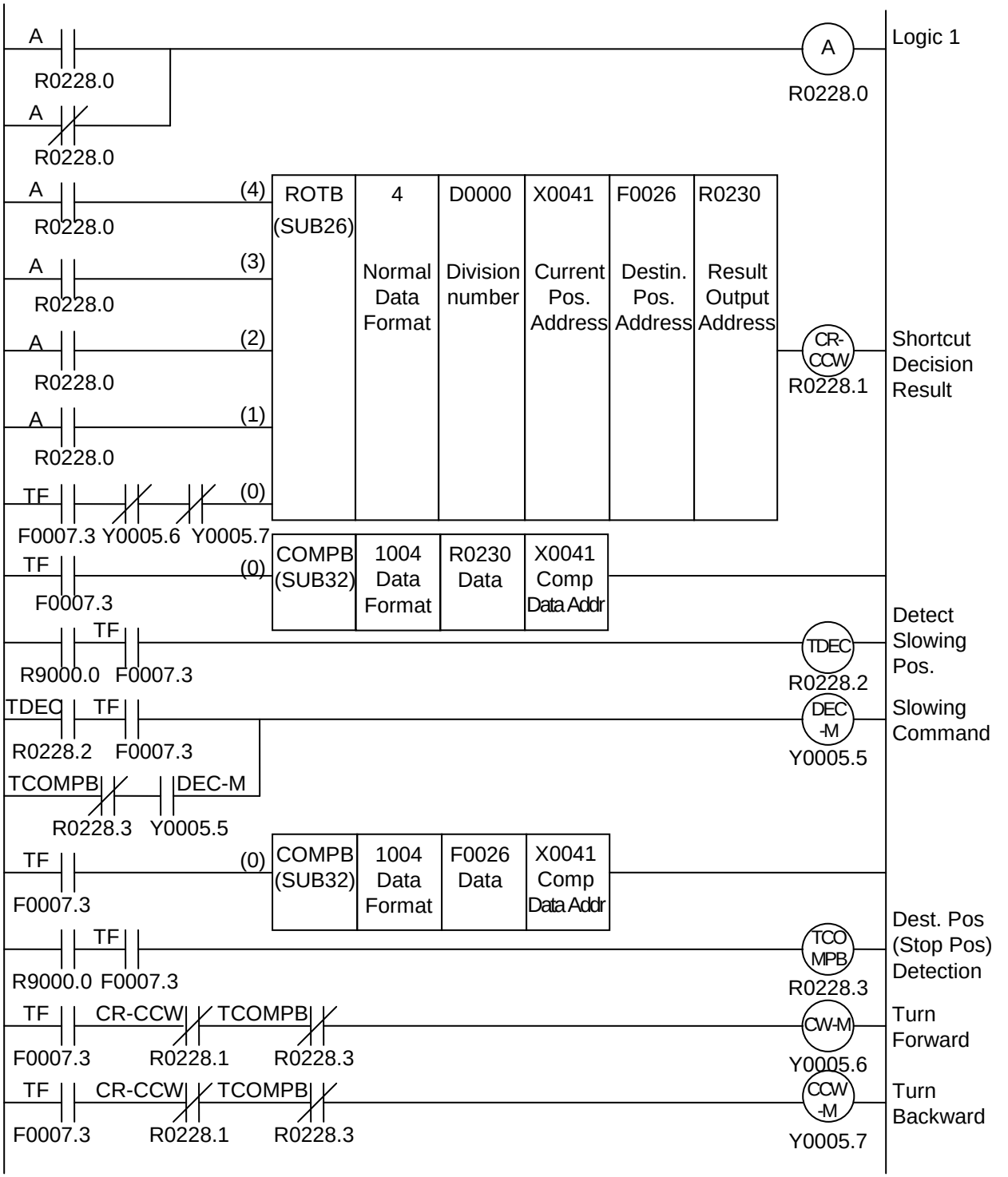


Figure 7-5: Ladder Diagram Example Using the ROTB Command

Code Example

```
%@3
RD  R228.0
RD.STK  R228.0
RD.STK  R228.0
RD.STK  R228.0
RD.STK  F7.3
SUB 26
1
0
X41
F26
R230
WRT  R228.1

%
```

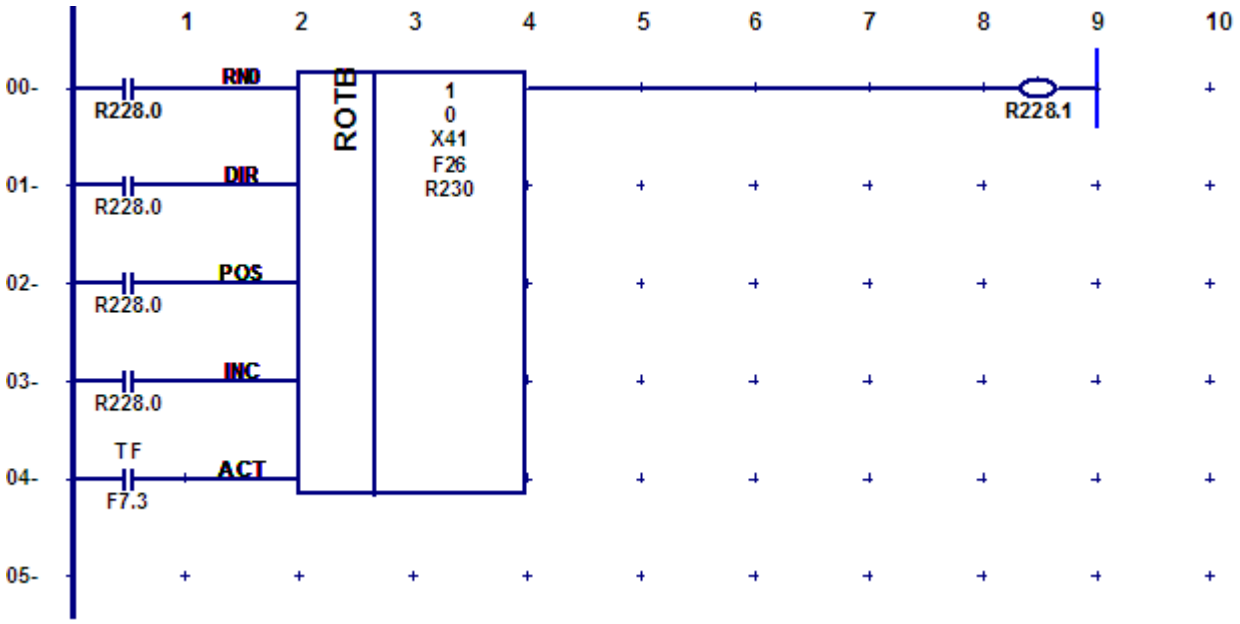


Figure 7-6: ROTB Code Example, Ladder View

Chapter 8: Transformation and Conversion Function Blocks

8.1 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 the following figure. 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 the following figure, 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.

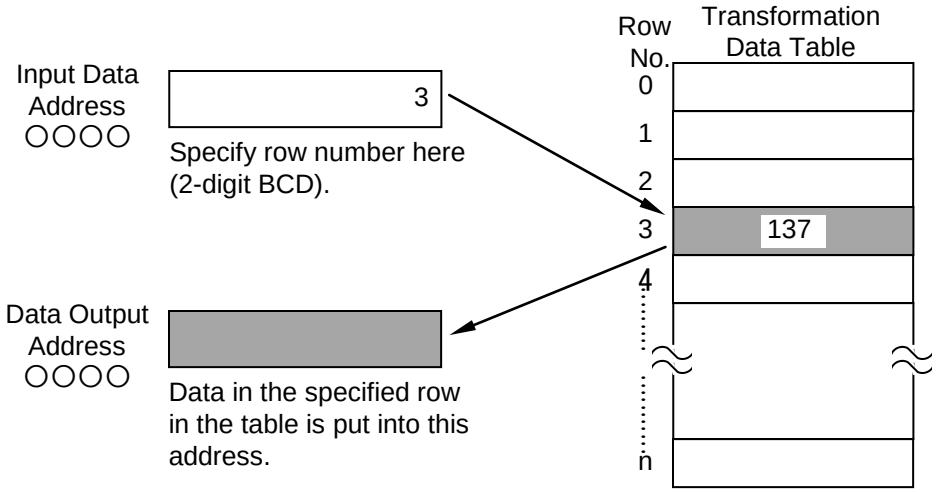


Figure 8-1: Code Transformation Using the COD Command

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

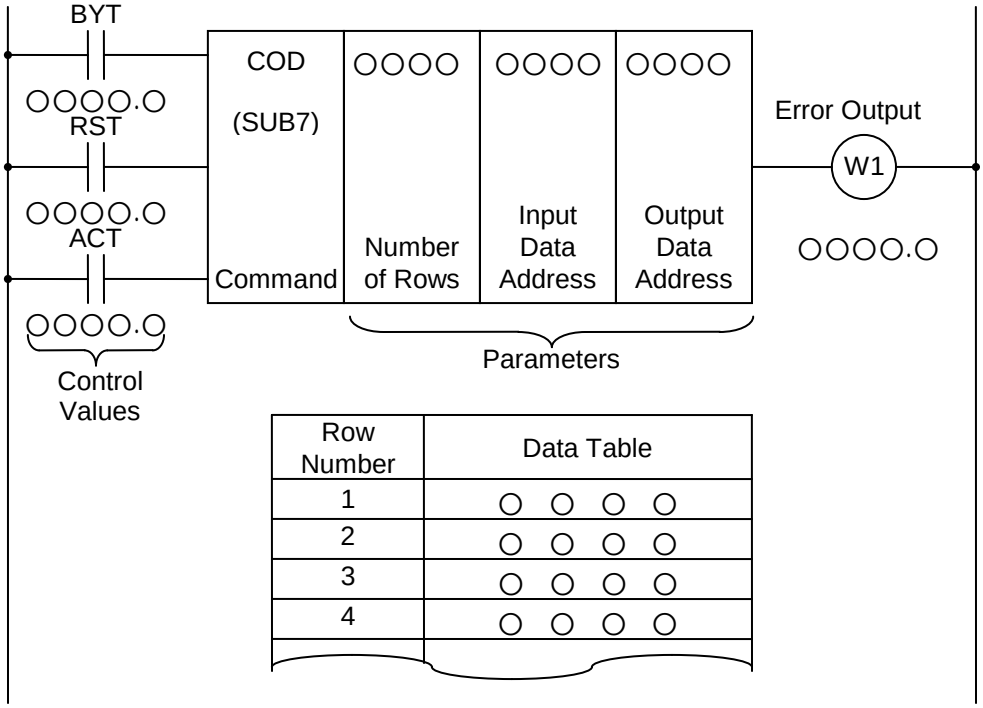


Figure 8-2: Format for the COD Command

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

Table 8-1: 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.

Code Example

```
%@3
RD  R120.1
RD.STK  R120.1
RD.STK  X0.1
SUB 7
5
X1.1
D0
9
8
7
6
5
WRT  R128.1
%
```

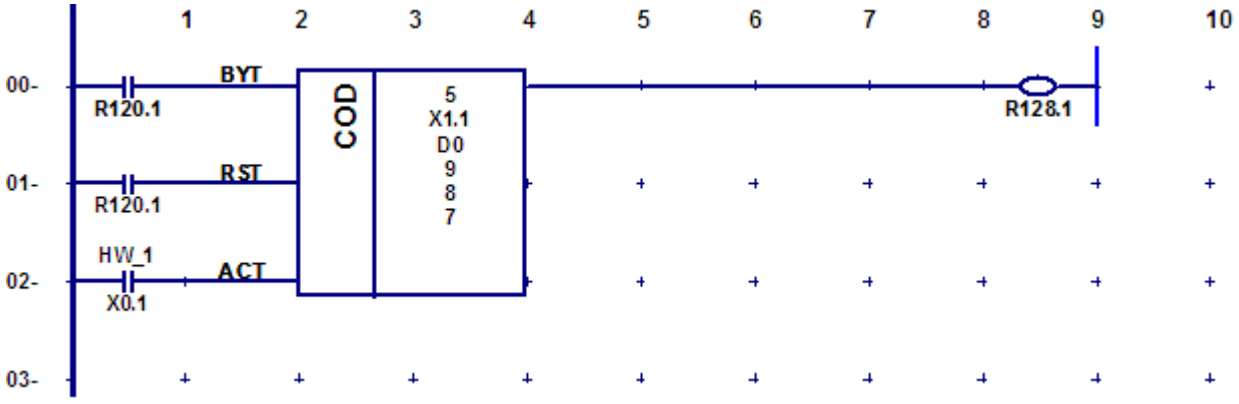


Figure 8-3: COD Code Example, Ladder View

8.2 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 the following figure. CODB function differs from the COD function (*Section 8.1: COD (Code Transformation)*), in that it utilizes numeric data that is in 1-, 2-, or 4-byte size binary style and the transformation data table (D table) size can be expanded to a maximum of 256 rows.

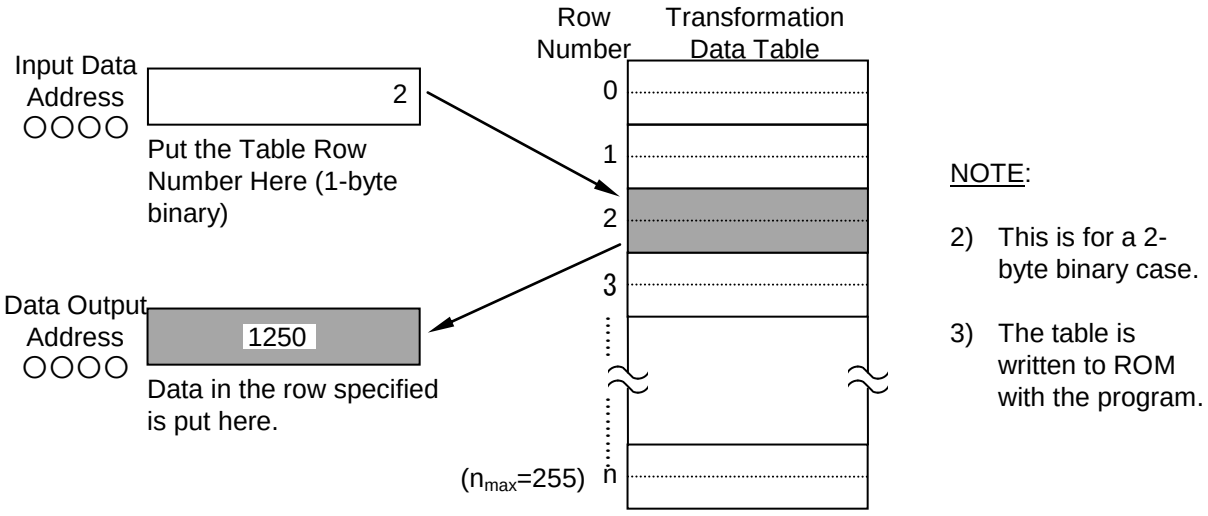
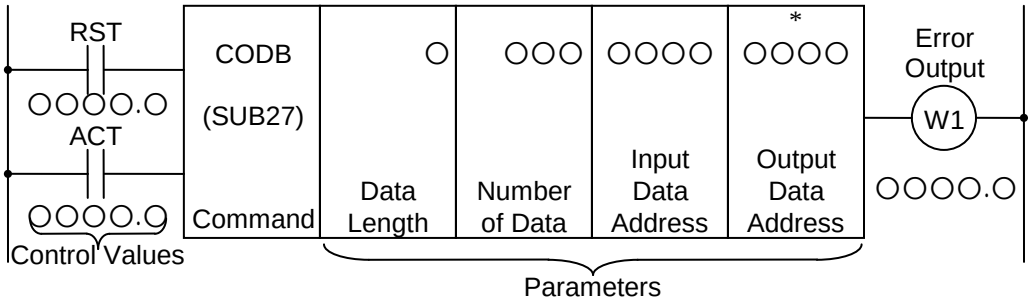


Figure 8-4: Code Transformation Using the CODB Command

Format

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 8-5: 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.

Code Example

```
%@3
// ******************************************
// * Please create D data space in the data table as follows:
// * D0 - D1 AS 1 BYTE BCD
// ******************************************

RD    K0.1  //RST=0, no reset
RD.STK      R0.0  //ACT=1, run CODB command
SUB 27
2           //Data Length: 2 byte binary
5           //Number of data table's row: 0-256
D0          //Input data address, 1 byte
D1          //Output data address, 1, 2 or 4 bytes
120         //Row 0
121         //Row 1
122         //Row 2
123         //Row 3
124         //Row 4
WRT   R0.1  //error output

%
```

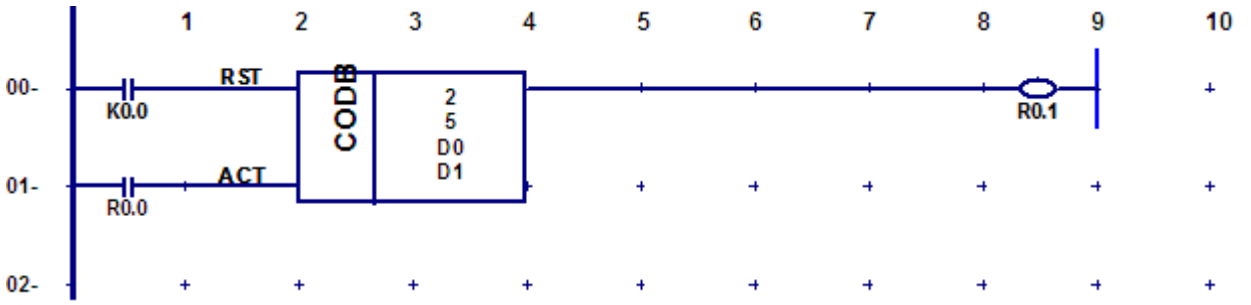


Figure 8-6: CODB Code Example, Ladder View

8.3 DCNV (Data Conversion)

Function

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

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

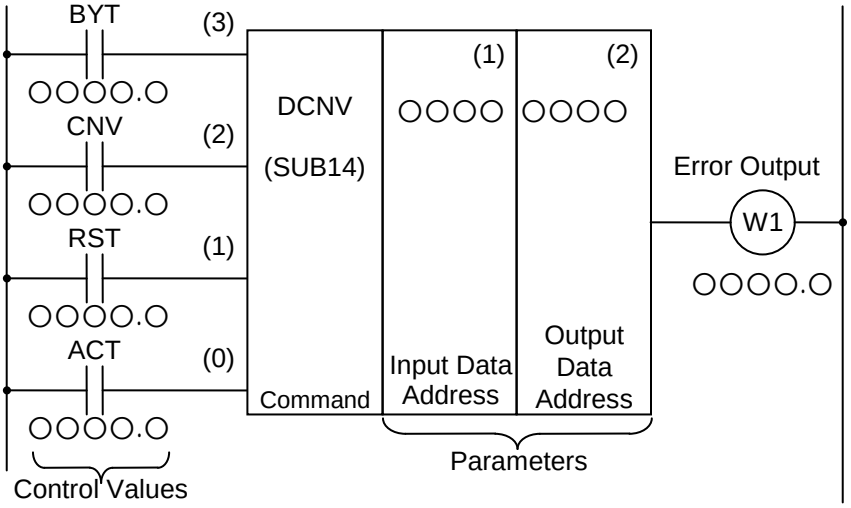


Figure 8-7: Format for the DCNV Command

Coding Sheet

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

Result History Register

ST3	ST2	ST1	ST0	
			BYT	control values
		BYT	CNV	
	BYT	CNV	RST	
BYT	CNV	RST	ACT	
BYT	CNV	RST	ACT	command
BYT	CNV	RST	ACT	parameters
BYT	CNV	RST	ACT	
BYT	CNV	RST	W1	result

Table 8-2: 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.

Parameters

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

2) 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 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.)

Code Example

```
%@3

RD      R0.0      //BYT=0, 1 BYTE DATA
RD.STK  R0.1      //CNV=1, BCD TO BINARY
RD.STK  R0.2      //RST=0, NO RESET
RD.STK  R0.3      //ACT=1, RUN COMMAND
SUB 14
D0          //INPUT
D10         //OUTPUT
WRT   R1.0  //ERROR OUTPUT

%
```

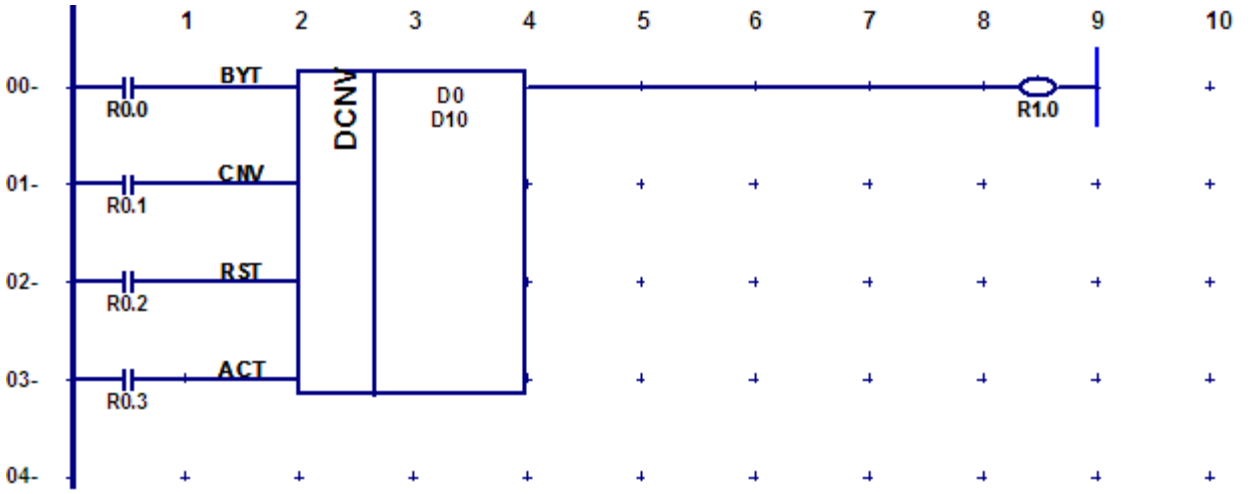


Figure 8-8: DCNV Code Example, Ladder View

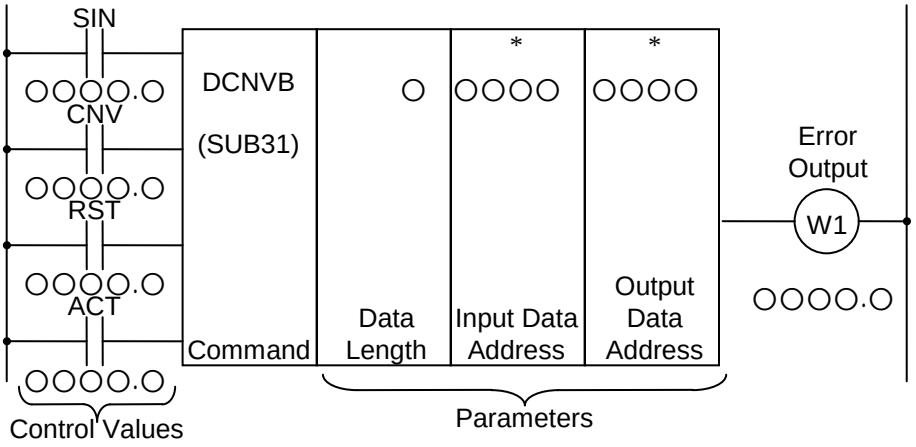
8.4 DCNVB (Extended Data Conversion)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 8-9: 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

3) 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

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

5) 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 the following figure for a description of positive or negative conversion of binary data into BCD code.

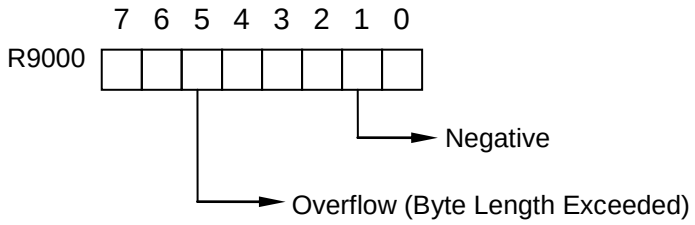


Figure 8-10: Calculation Result Register for the DCNVB Command

Code Example

```
%@3

// ************************************************************************
// * Please create D data space in the data table as follows:
// * D0 - D1 AS 1 BYTE BCD
// ************************************************************************

RD    R0.0        // ONLY BE USED WHEN BCD -> BIN SIN=0(+)
RD.STK  R0.1      // CNV=1, BCD -> BIN
RD.STK  R0.2      // RST=0, NO RESET
RD.STK  R0.3      // ACT=1, RUN DATA CONVERT
SUB 31
1           // DATA LENGTH 1 BYTE DATA
D1          // INPUT DATA ADDRESS
D2          // OUTUT DATA ADDRESS
WRT   R1.0  // ERROR OUTPUT WHEN WRT==1, ERROR.

%
```

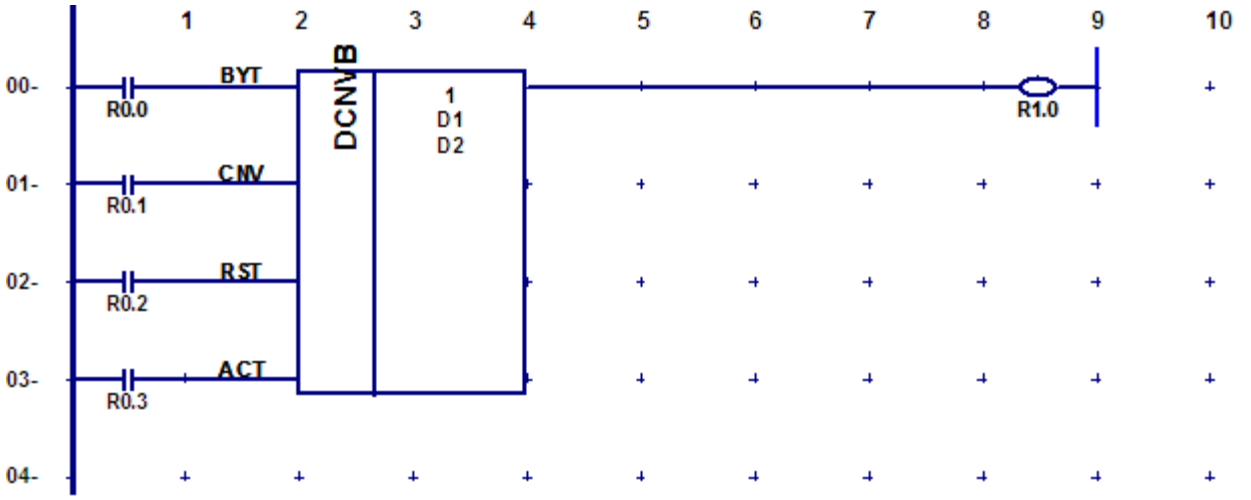


Figure 8-11: DCNVB Code Example, Ladder View

Chapter 9: Data Transfer and Data Shift Function Blocks

9.1 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).

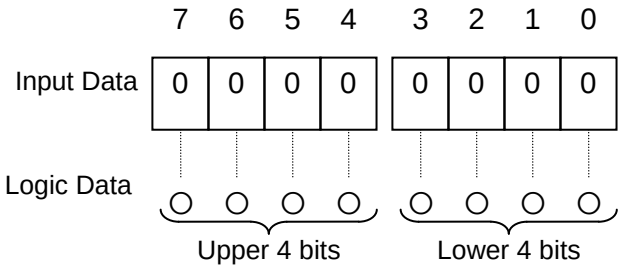


Figure 9-1: Input Data and Logic Data for the MOVE Command

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

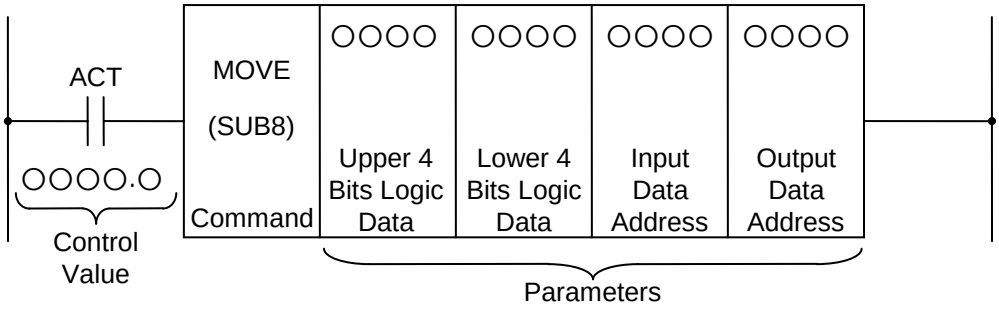


Figure 9-2: Format for the MOVE Command

Coding Sheet				Result History Register				
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**	
1	RD	○○○○ . ○	ACT				ACT	control value
2	SUB	8	MOVE Command				ACT	command
3	(PRM)	○○○○	Upper 4 Bits Logic Data				ACT	parameters
4	(PRM)	○○○○	Lower 4 Bits Logic Data				ACT	
5	(PRM)	○○○○	Input Data Address				ACT	
6	(PRM)	○○○○	Output Data Address				ACT	

Table 9-1: 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.

Parameter

1) Upper 4 Bits Logic Data
The upper 4 bits of 1-byte logic data. (See Figure 9-1.)

2) Lower 4 Bits Logic Data
The lower 4 bits of 1-byte logic data. (See Figure 9-1.)

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

4) Output Data Address
The address for the output data.

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.

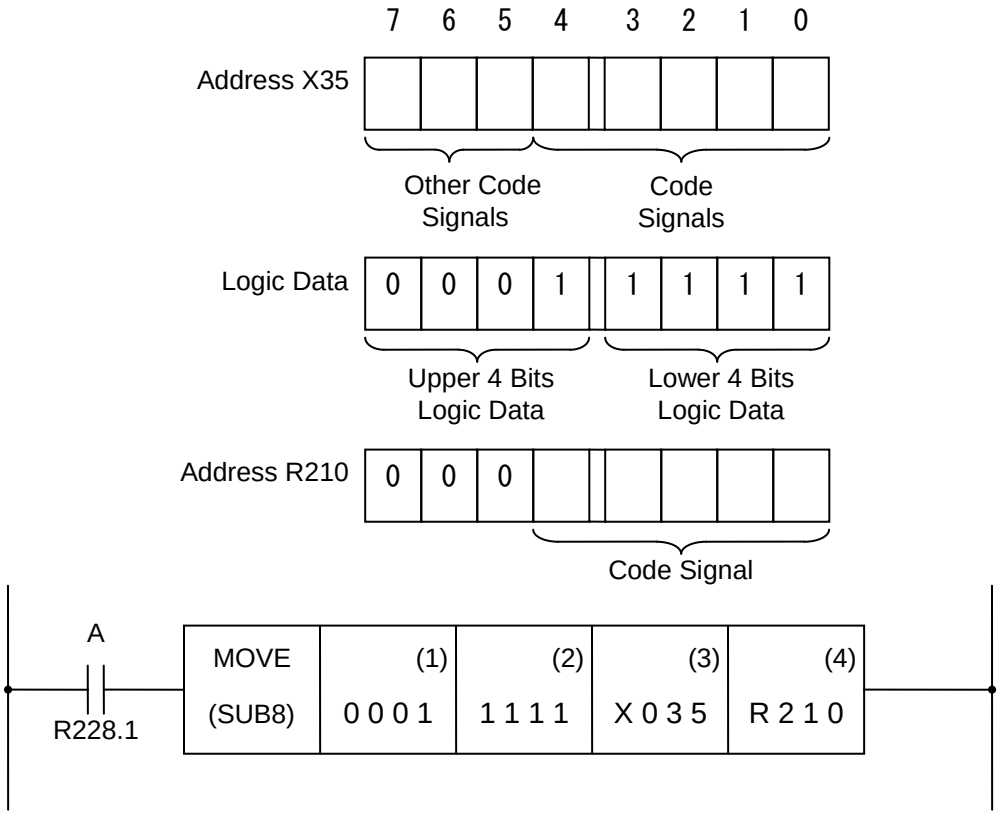


Figure 9-3: Ladder Diagram Example Using the MOVE Command

Code Example

```
%@3
RD  X0.1
SUB 8
0000
0001
D0
D1
%
```

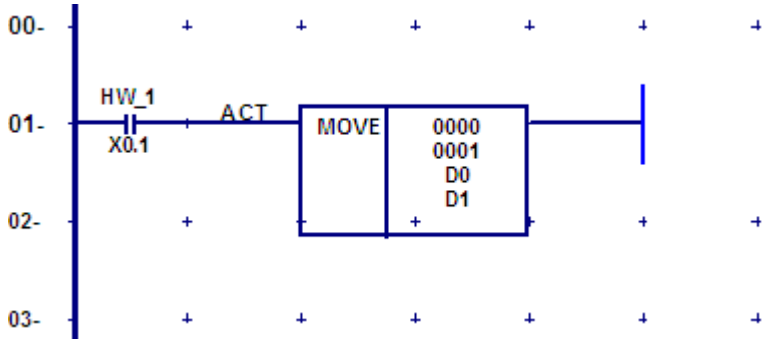


Figure 9-4: MOVE Code Example, Ladder View

9.2 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.

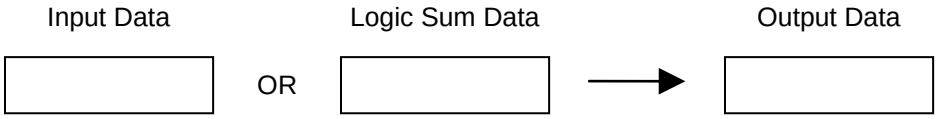


Figure 9-5: Function of the MOVOR Command

Format

The following figure shows the format for describing the command.

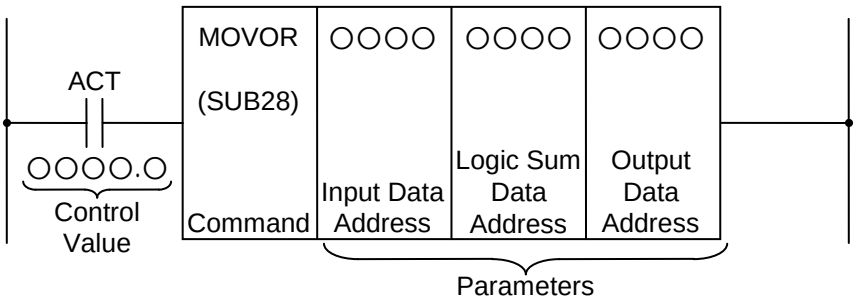


Figure 9-6: Format for the MOVOR Command

Control Values

Action Command (ACT)

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

Parameter

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

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

7) 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.

Code Example

```
%@3

RD  R0.0
SUB 28
D1
D2
D3

%
```

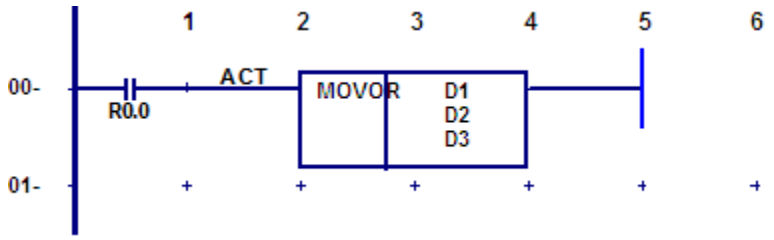


Figure 9-7: MOVOR Code Example, Ladder View

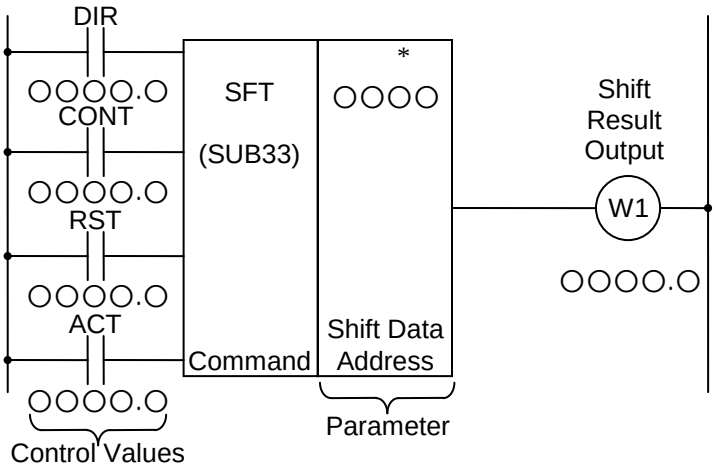
9.3 SFT (Shift Register)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 9-8: 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).

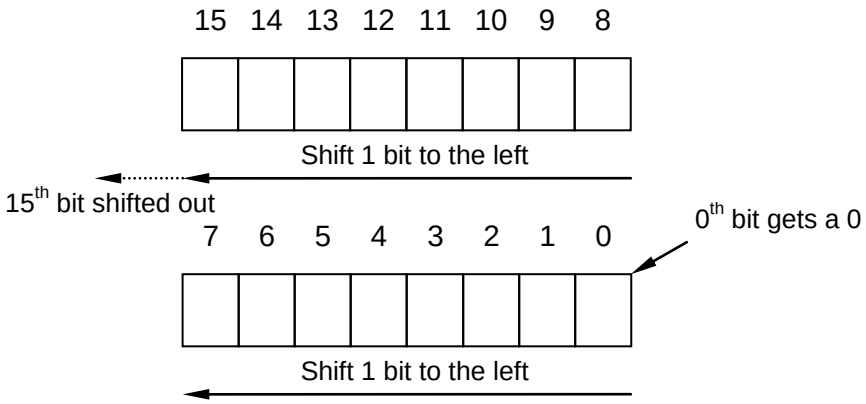


Figure 9-9: Condition Specification CONT = 0 for the SFT Command – Shift Left Example

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.

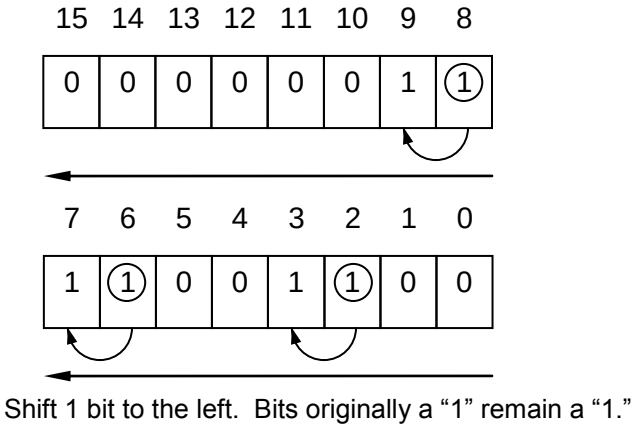


Figure 9-10: Condition Specification CONT = 1 for the SFT Command – Shift Left Example

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.

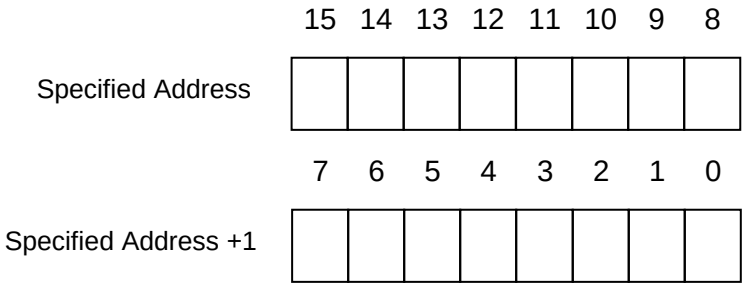


Figure 9-11: Shift Data Address for the SFT Command

Shift Result Output (W1)

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

Code Example

```
%@3

// ------------------------------------------------------------------------
// * Default value of K0 : K0.0=0 K0.1=1  K0.3=1
// * D data table set to : D250: 2 byte BIN, D251: 2 byte BIN
// *                       D250 = 1
// * Timer               : TRM 1=2000 (2sec)
// ------------------------------------------------------------------------
RD       K0.3           // K0.3 = 1
AND.NOT  R0.1
WRT      R0.0

RD       R0.0
TMR 1
WRT      R0.1           // It will be turned on every 2 sec

RD       K0.0           // DIR  = 0 shift to left
RD.STK   K0.1           // CONT = 0 keep previous value
RD.STK   K0.2           // RST  = 0 No reset
RD.STK   R0.1           // ACT  = 1 run SFT command
SUB 33
D250                    // Data address 2 bytes
WRT      R0.2           // result output
// ------------------------------------------------------------------------

%
```

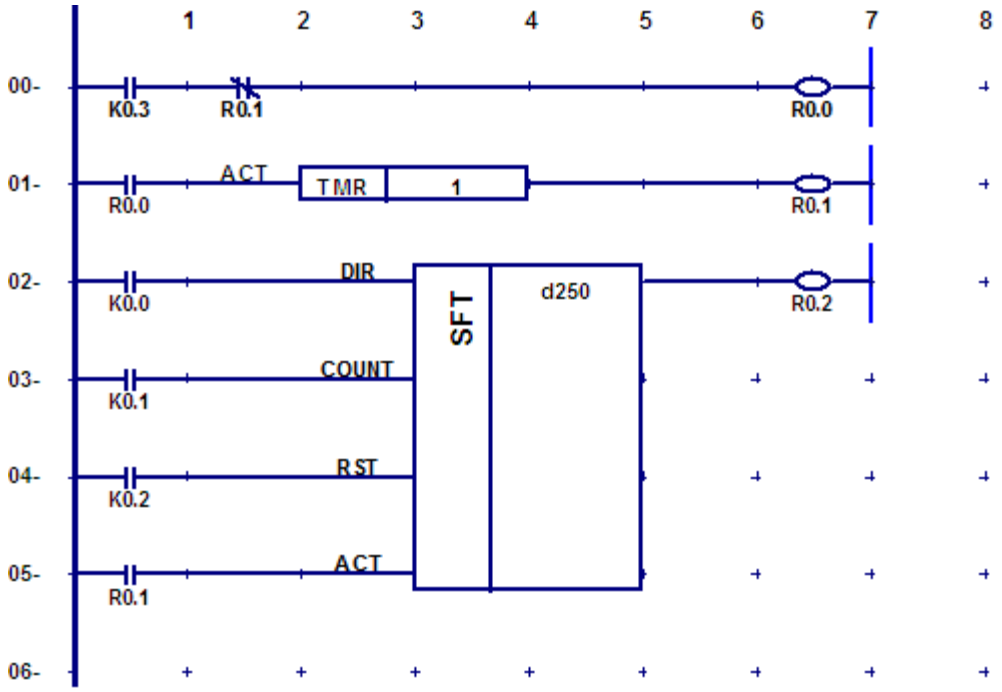


Figure 9-12: TMR and SFT Code Example, Ladder View

Chapter 10: Jump and Common Line Control Function Blocks

10.1 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.

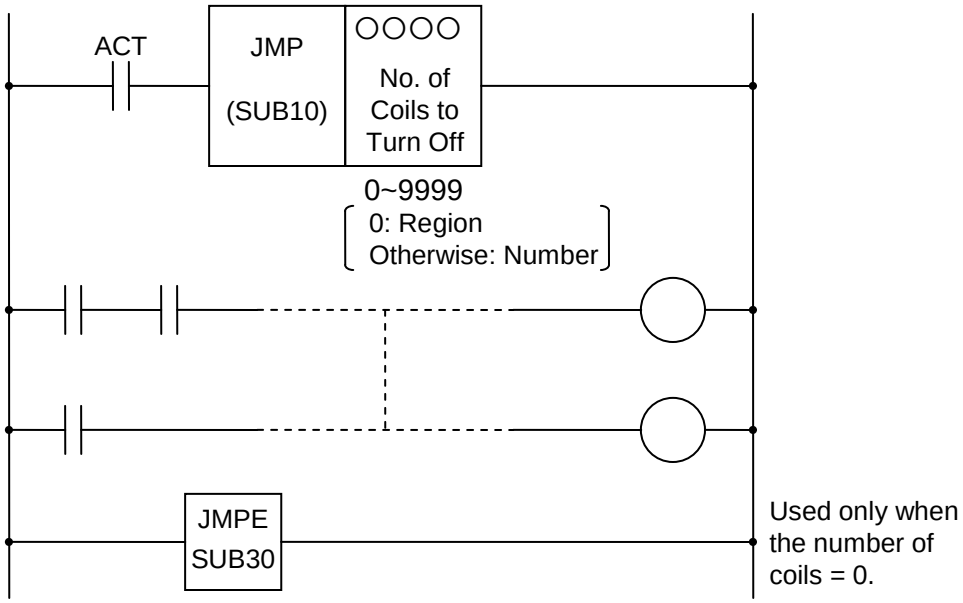


Figure 10-1: Function of the JMP Command

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

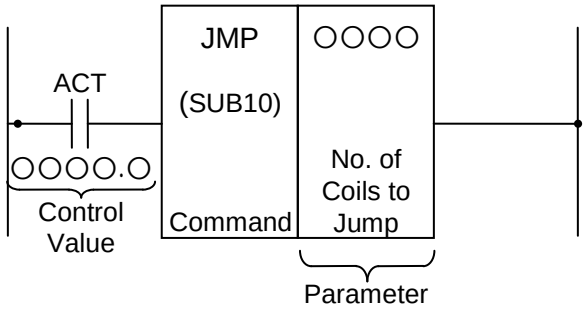


Figure 10-2: Format for the JMP Command

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

Table 10-1: 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 Skip

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 the following figure for a ladder diagram example utilizing the JMP command.

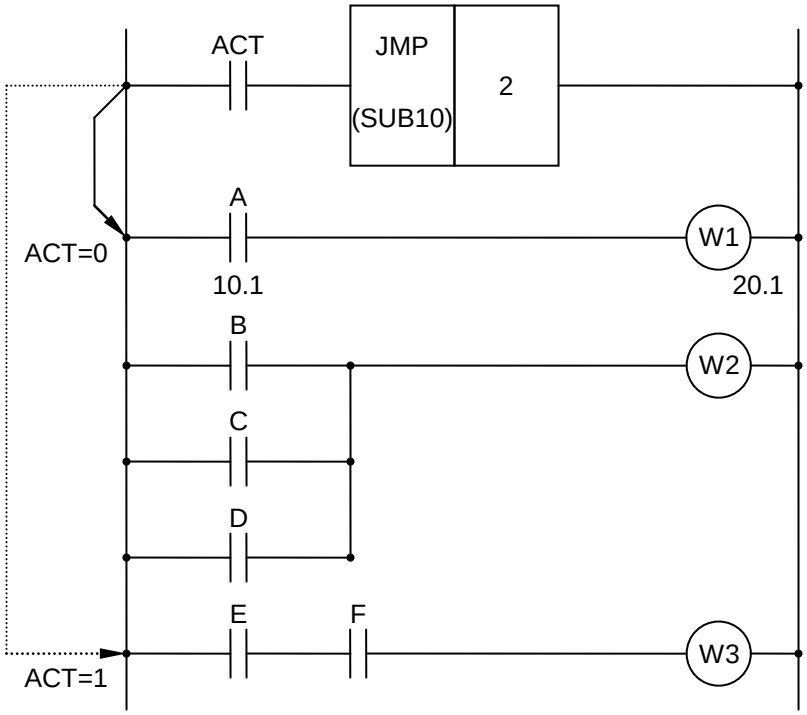


Figure 10-3: Ladder Diagram Example Using the JMP Command

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 10-3, 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.

Code Example

```
%@3

RD X0.1
SUB 10
2

RD X1.1
WRT R10.1

RD X1.1
WRT R10.2

%
```

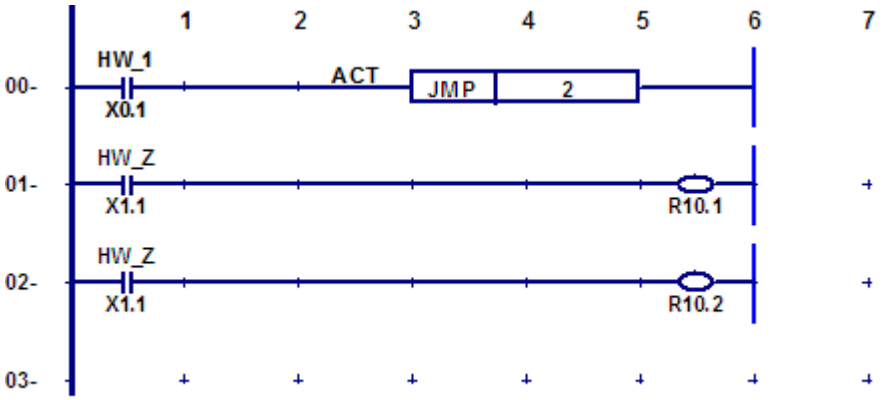


Figure 10-4: JMP Code Example, Ladder View

10.2 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

The following figure shows the format for describing the command.

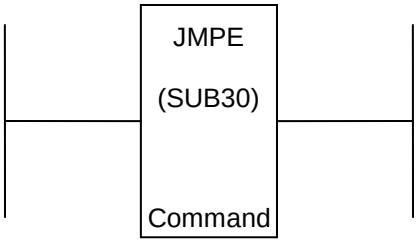


Figure 10-5: Format for the JMPE Command

Code Example

```
%@3
RD      R0.1
SUB 10
0

RD      R1.1
WRT     R10.1

SUB 30
```

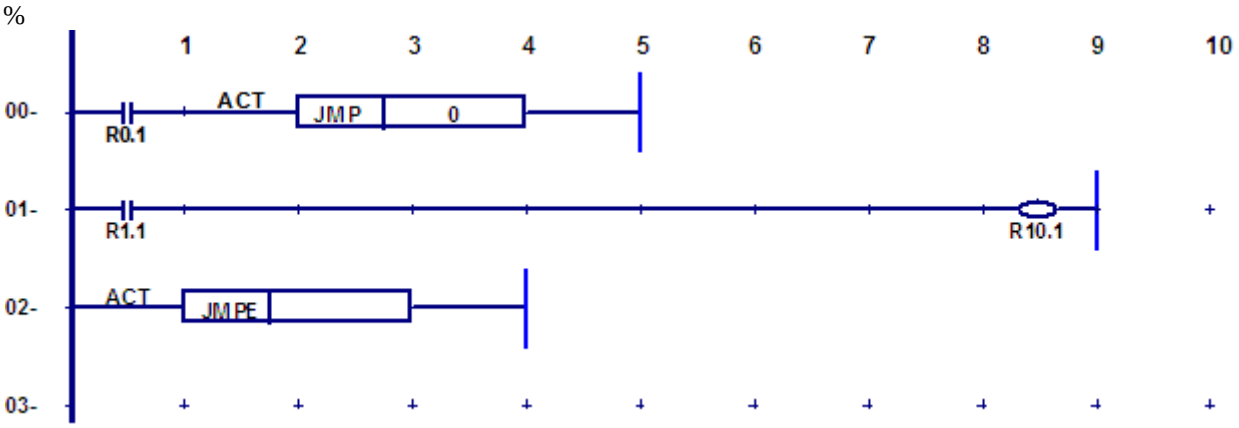


Figure 10-6: JMPE Code Example, Ladder View

10.3 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 the following figure. 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.

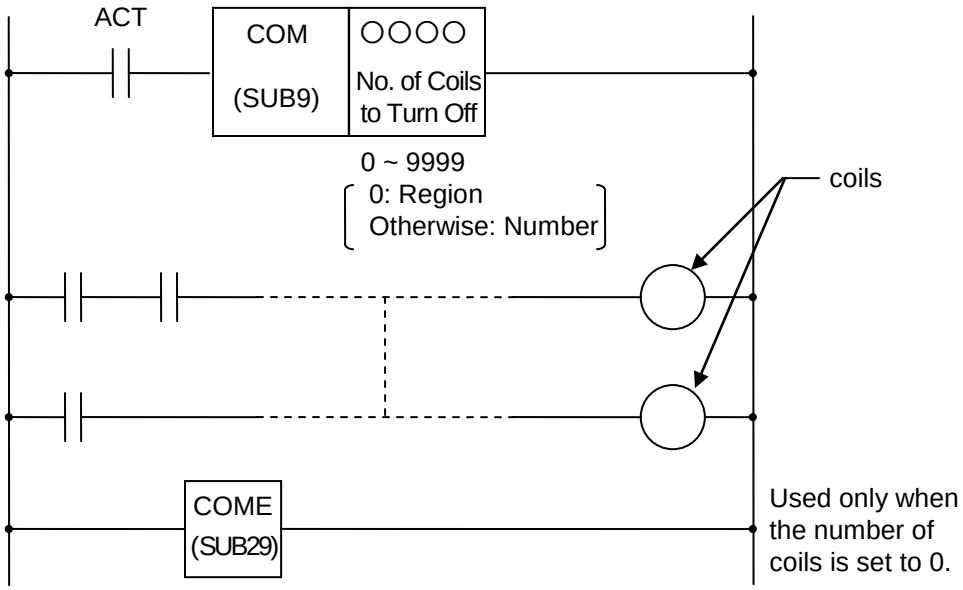


Figure 10-7: Function of the COM Command

Format

The following figure shows the format for describing the command.

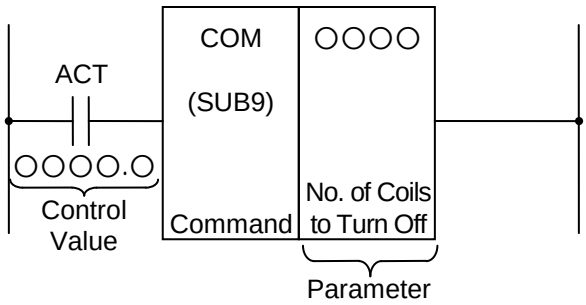


Figure 10-8: Format for the COM Command

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: All coils are turned off until the COME command is reached.
When a number other than 0 is specified: Becomes coil number specific.

The following kinds of coils can be skipped:

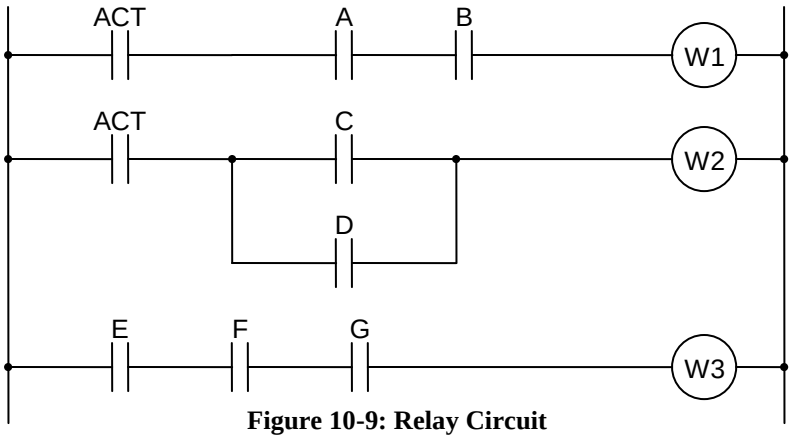


Figure 10-9: Relay Circuit

CAUTION

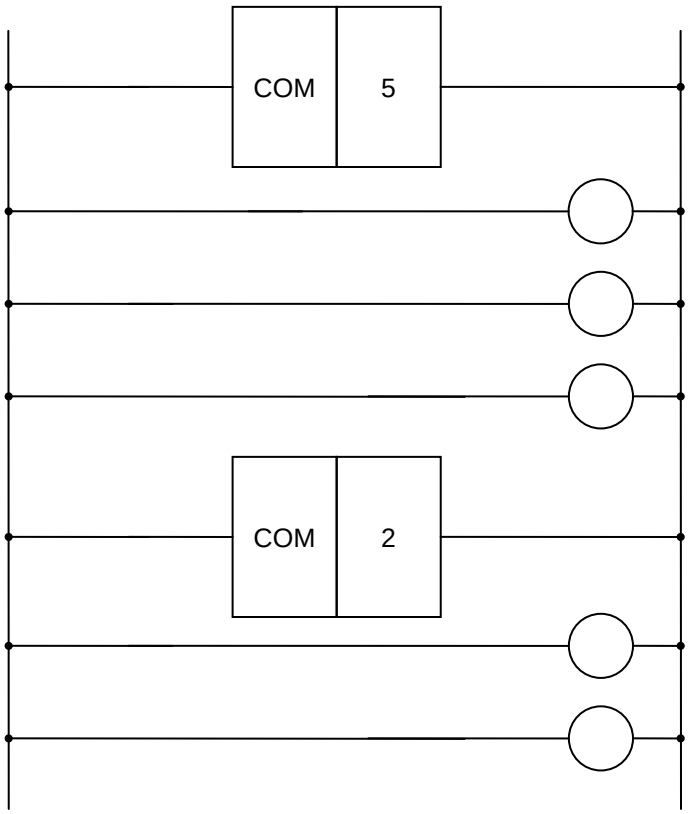
The commands within the specified area are executed regardless of the ACT value. However, when ACT=0, the result coil unconditionally becomes 0.

CAUTION

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

⚠ CAUTION

Nested COM calls are not allowed. In other words, there can't be a second (or third) COM command inside the COM range because COM coils can't be turned off by another COM command. In other words, the following ladder logic will produce an error:



Examples of COM Command Usage

See the following two figures for examples of COM command usage.

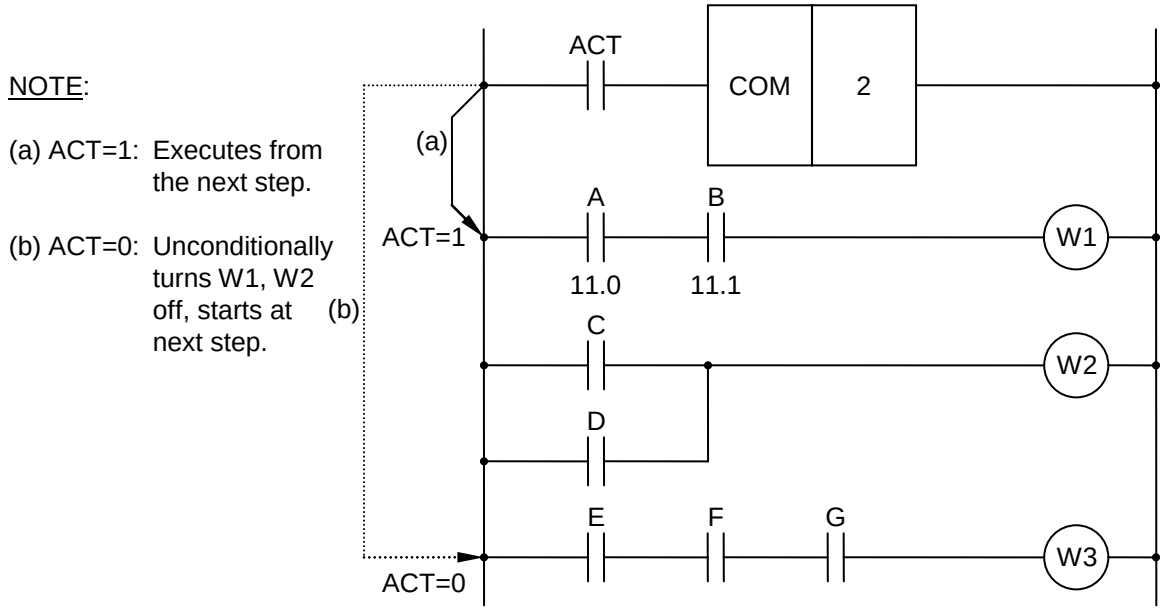


Figure 10-10: Ladder Diagram Using the COM Command

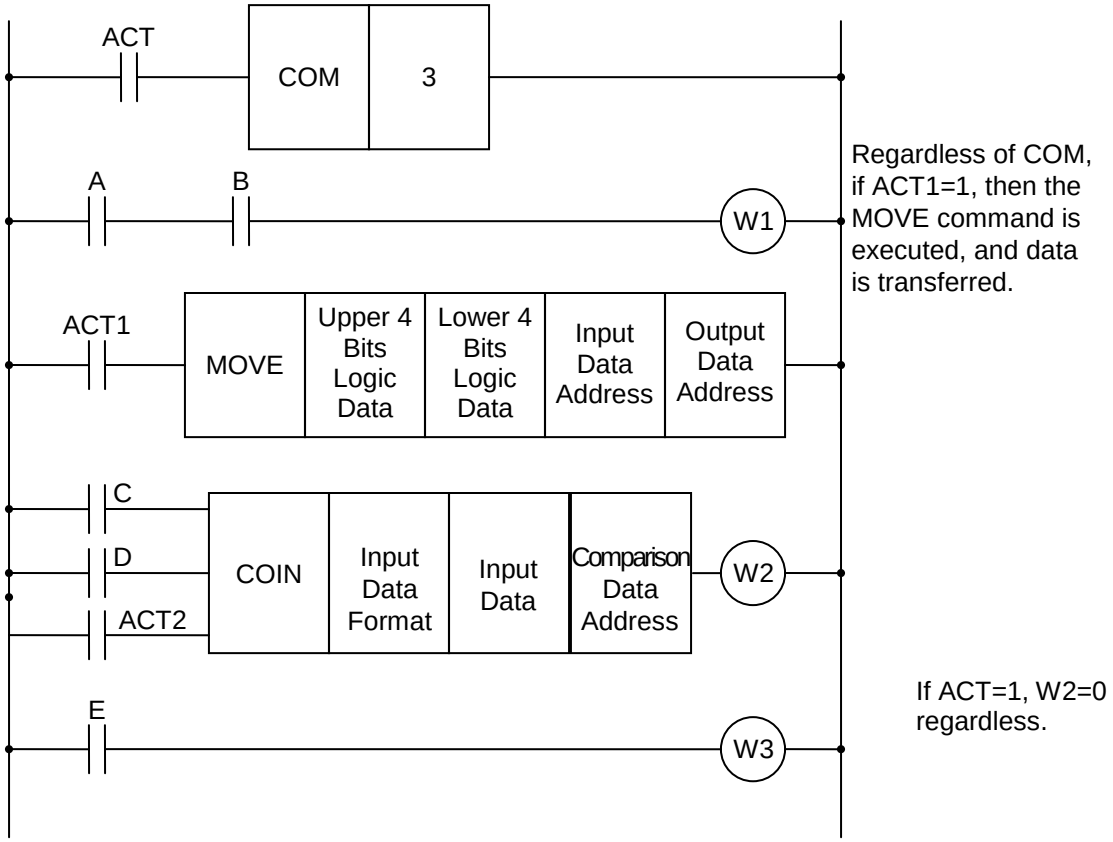


Figure 10-11: Ladder Diagram Example Using COM, MOVE and COIN Commands

Code Example

```
%@3
RD    R0.0  //ACT
SUB 9
2           //NUMBER OF COILS THAT WILL BE TURNED OFF

RD    R0.0        //COIL No. 1
WRT   R1.0

RD    R0.0        //COIL No. 2
WRT   R1.1

RD    R0.0        //COIL No. 3
WRT   R1.2

RD    R0.0        //COIL No. 4
WRT   R1.3

RD    R0.0        //COIL No. 5
WRT   R1.4

RD    R0.0        //COIL No. 6
WRT   R1.5
WRT   R1.6

%
```

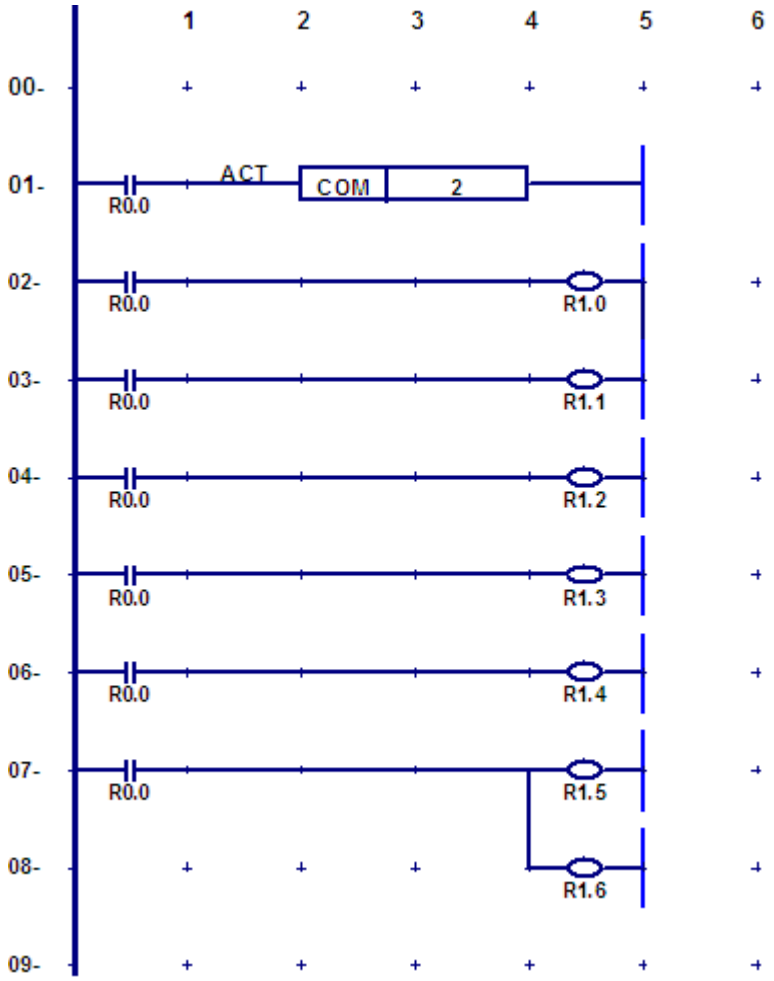


Figure 10-12: COM Code Example, Ladder View

10.4 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

The following figure shows the format for describing the command.

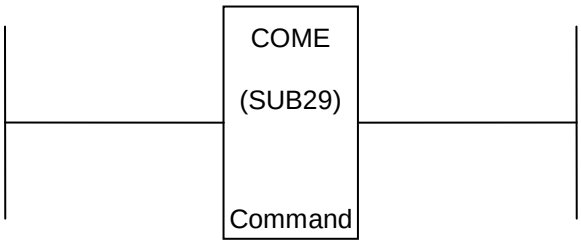


Figure 10-13: Format for the COME Command

Code Example

```
%@3
RD    R0.0
SUB 9
0

RD    R0.1
WRT R10.1

SUB 29

RD    R0.1
WRT R10.2

%
```

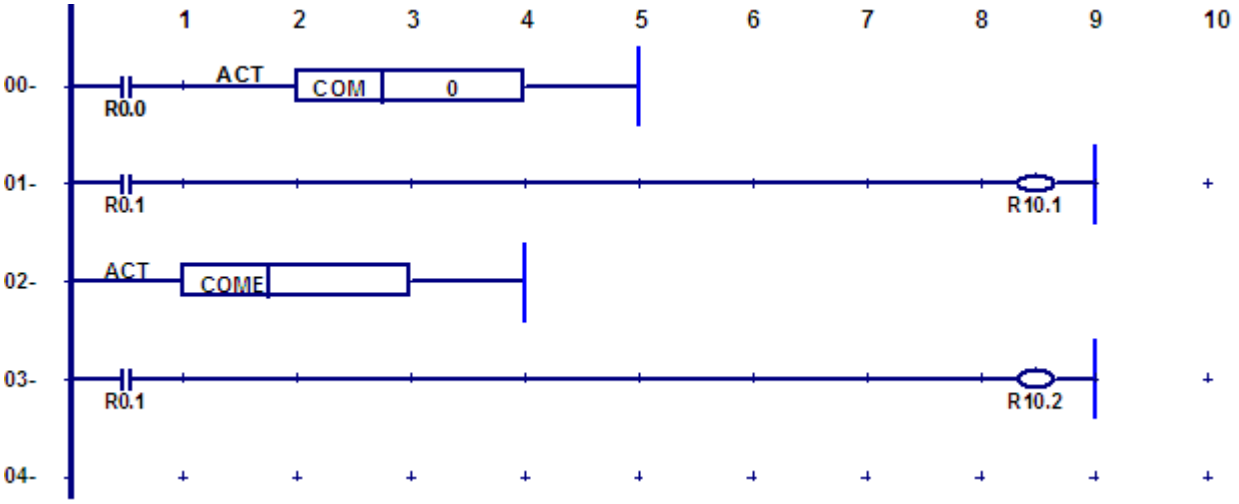


Figure 10-14: COME Code Example, Ladder View

Chapter 11: Function Blocks for Data Checking, Data Comparison and Data Manipulation

11.1 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

The following figure shows the format for describing the command, and the following table shows the coding format.

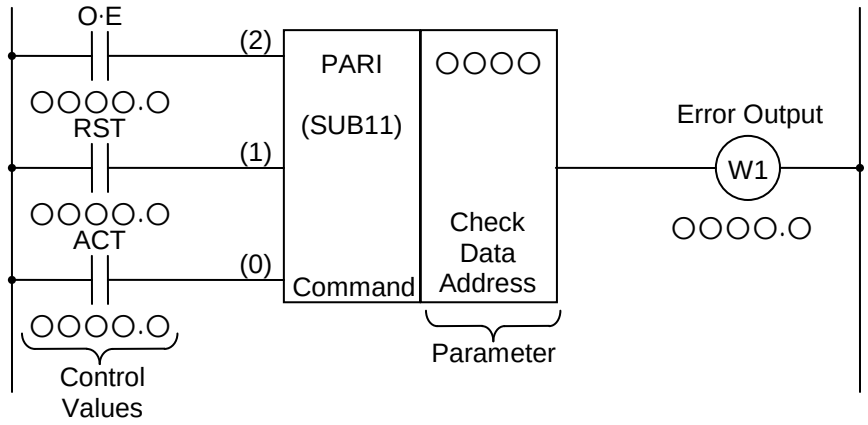


Figure 11-1: Format for the PARI Command

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

Table 11-1: 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.

Parameters

Check Data Address
The address of the data to be checked.

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

The following figure shows the odd number parity check of a code signal that gets input into address X036.

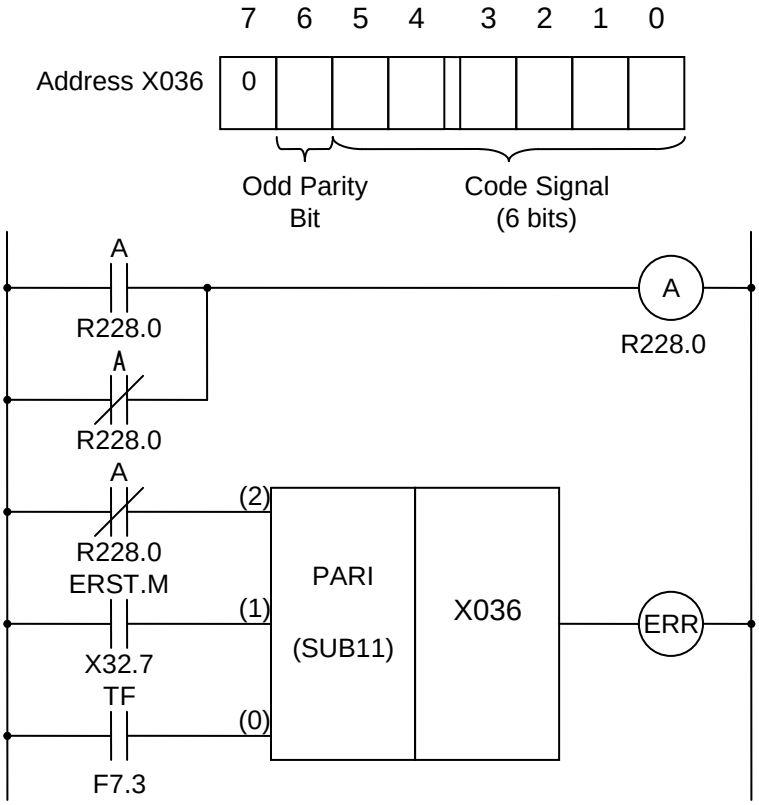


Figure 11-2: Ladder Diagram Example Using the PARI Command

CAUTION

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

Code Example

```
%@3
RD      R0.0      // O.E = 0 : check even
RD.STK  R0.0    // RST = 0 : Not reset
RD.STK  R0.1      // ACT = 1 : Run PARI command
SUB 11
D0
WRT   R0.2  // W1=1, ERROR

%
```

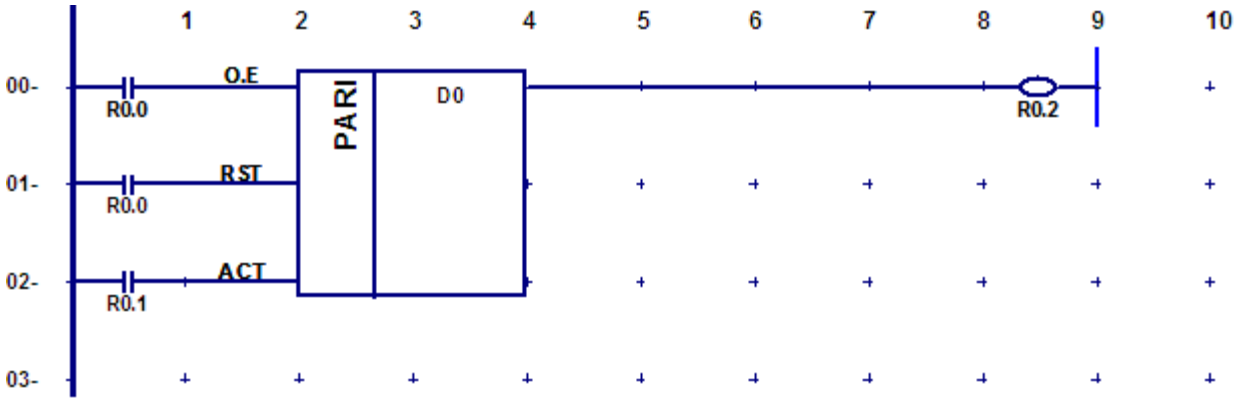


Figure 11-3: PARI Code Example, Ladder View

11.2 COMP (Compare)

Function

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

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

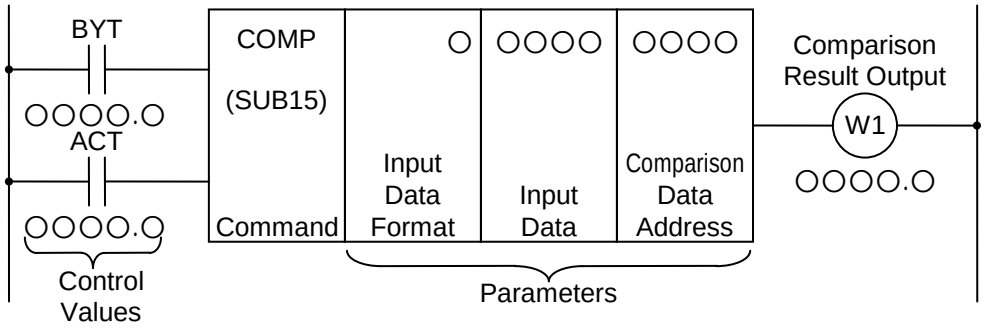


Figure 11-4: Format for the COMP Command

Coding Sheet

Step No.	Command	Address No.	Bit no.	Description
1	RD	OOOO	O	BYT
2	RD.STK	OOOO	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	OOOO	O	W1, Comparison Result Output

Result History Register

ST3	ST2	ST1	ST0	
			BYT	control values
		BYT	ACT	control values
		BYT	ACT	command
		BYT	ACT	parameters
		BYT	ACT	parameters
		BYT	ACT	parameters
		BYT	W1	result

Table 11-2: 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 $\geq$ Comparison Value.
W1 = 1: Input Value $\leq$ Comparison Value.

Code Example

```
%@3

RD  X0.1
WRT R208.1

RD  X0.1
RD.STK  X1.1
SUB 15
1
D0
D1
WRT  R208.1

%
```

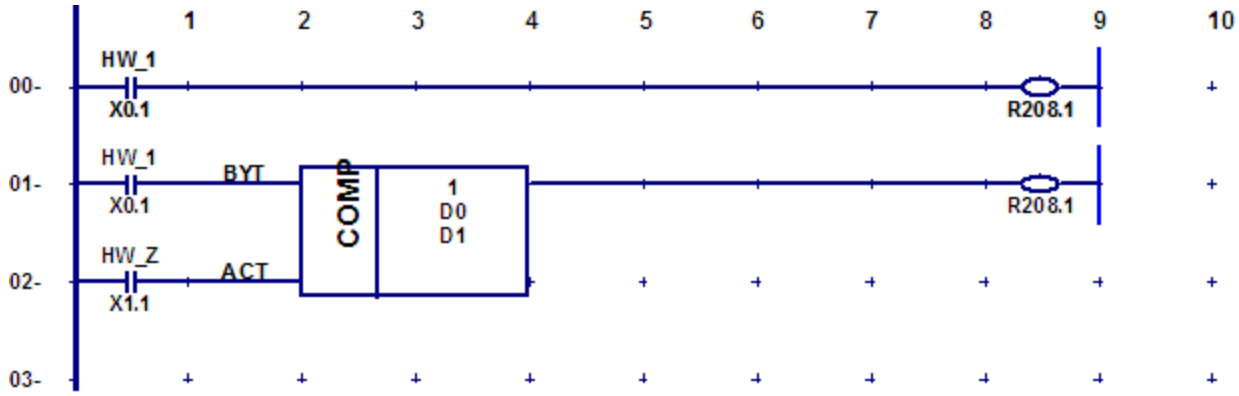


Figure 11-5: COMP Code Example, Ladder View

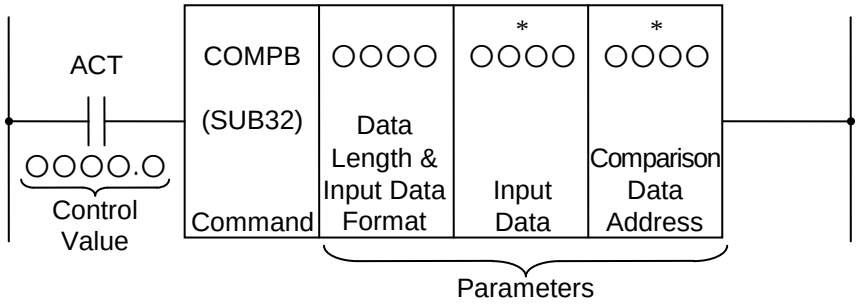
11.3 COMPB (Binary Compare)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 11-6: 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).

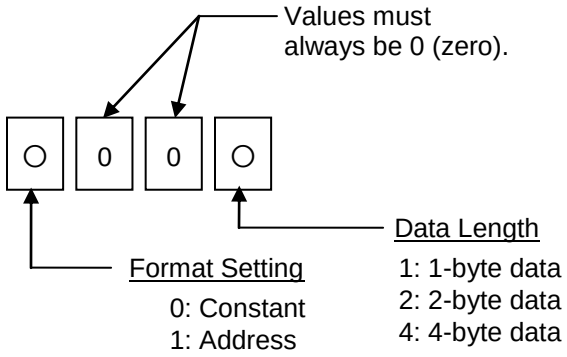


Figure 11-7: Parameters Format Specification for the COMPB Command

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:

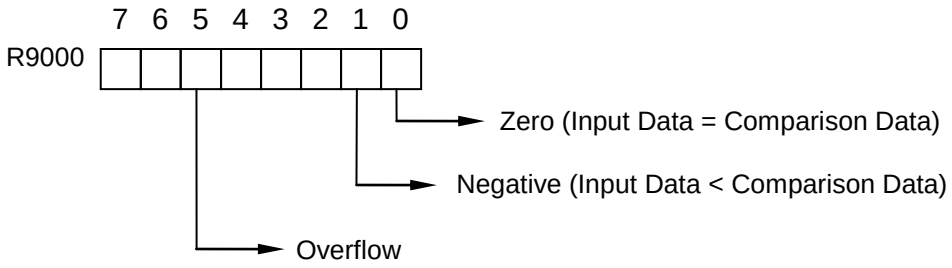


Figure 11-8: Calculation Result Register for the COMPB Command

Code Example

```
%@3

// ************************************************************************
// * Please create D data space in the data table as follows:
// * D0 - D1 1 BYTE BCD
// ************************************************************************

RD  R0.0
SUB 32
1001
D0
D1

%
```

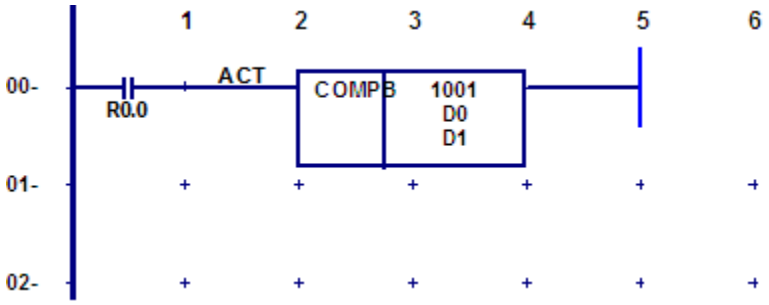


Figure 11-9: COMPB Code Example, Ladder View

11.4 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

The following figure shows the format for describing the command, and the following table shows the coding format.

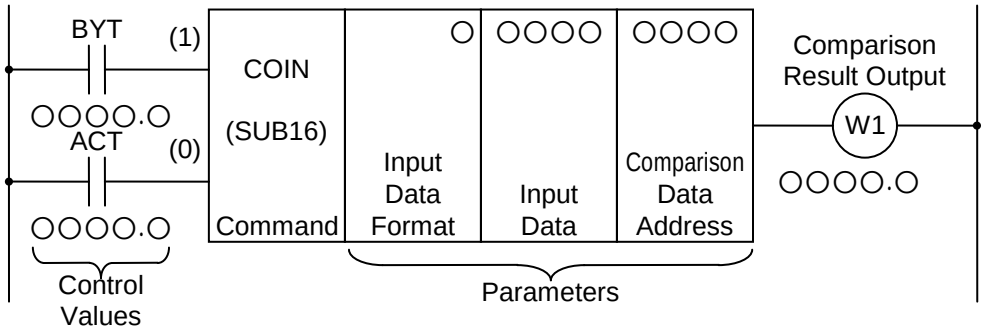


Figure 11-10: Format for the COIN Command

Coding Sheet / **Result History Register**

Step No.	Command	Address No.	Bit No.	Description	ST3	ST2	ST1	ST0	
1	RD	OOOO	. O	BYT				BYT	control values
2	RD.STK	OOOO	. O	ACT			BYT	ACT	
3	SUB	16		COIN Command			BYT	ACT	command
4	(PRM)	O		Input Data Format			BYT	ACT	parameters
5	(PRM)	OOOO		Input data			BYT	ACT	
6	(PRM)	OOOO		Comparison Data Address			BYT	ACT	
7	WRT	OOOO	. O	W1, Comparison Result Output			BYT	W1	result

Table 11-3: 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.

Code Example

```
%@3
// ************************************************************************
// * Please create D data space in the data table as follows:
// * D0 - D1 AS 1 BYTE BCD
// ************************************************************************

RD      K0.0   // 2-digit BCD
RD.STK  R0.0   // Activate coin command
SUB16
1              // Data Format =1, which specifies input address as address
D0             // Data address
D1             // Compare to this data
WRT     R0.1   // Write the result
%
```

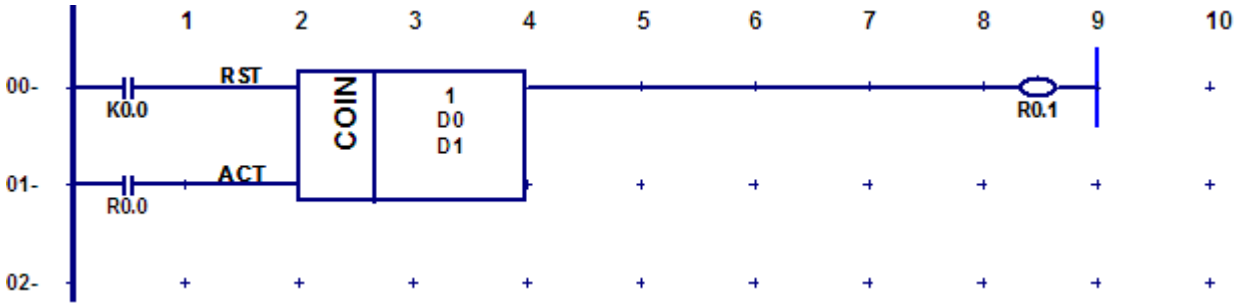


Figure 11-11: COIN Code Example, Ladder View

Chapter 12: Data Search and Data Transfer Function Blocks

12.1 DSCH (Data Search)

Function

With LadderWorks PLC, you can use something called a data table, which is explained in *Section 3.3: PLC Data Table(s)* and *Section 2.9 Data Addresses (D)* in the *LadderWorks PLC Reference Manual*. This command is related to this data table (D 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.

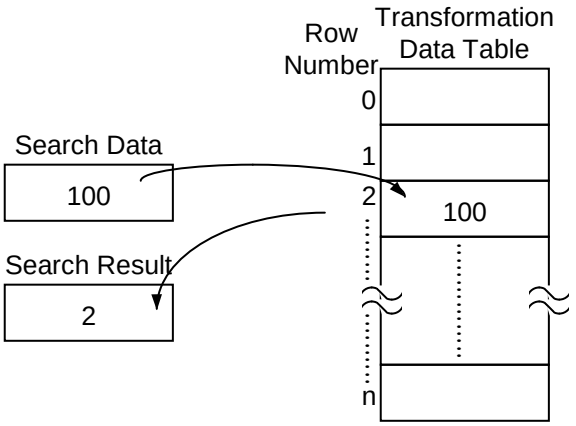


Figure 12-1: Function of the DSCH Command

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 3.3: PLC Data Table(s)* and *Section 2.9 Data Addresses (D)* in the *LadderWorks PLC Reference Manual*.

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

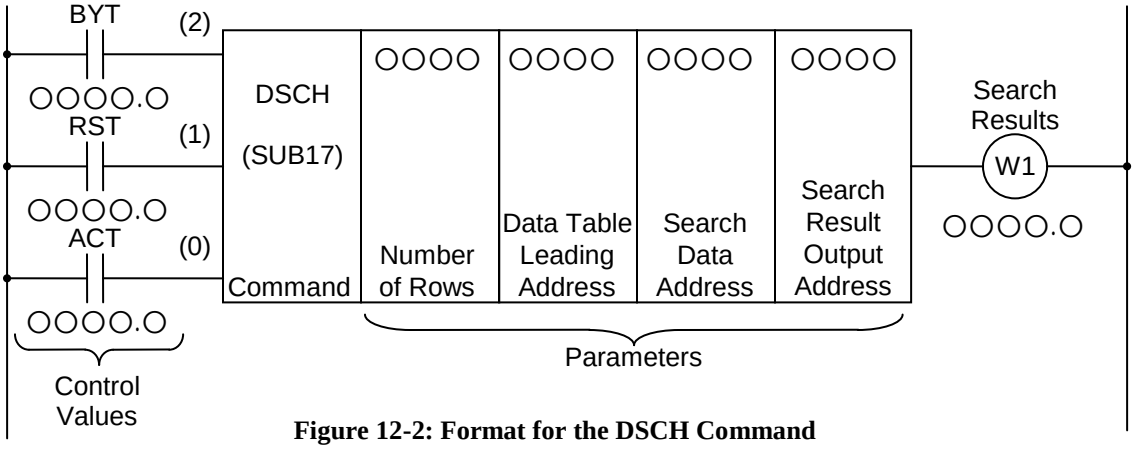


Figure 12-2: Format for the DSCH Command

Coding Sheet

Step No.	Command	Address No.	Bit No.	Description
1	RD	OOOO	O	BYT
2	RD.STK	OOOO	O	RST
3	RD.STK	OOOO	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	OOOO	O	W1, Search Results

Result History Register

ST3	ST2	ST1	ST0	
			BYT	control values
		BYT	RST	
	BYT	RST	ACT	
	BYT	RST	ACT	command
	BYT	RST	ACT	parameters
	BYT	RST	ACT	
	BYT	RST	ACT	
	BYT	RST	ACT	
	BYT	RST	W1	result

Table 12-1: 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
   The size of the table – 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. [NOTE: You must specify a number, not an address.]

2) Data Table Leading Address
   This is the starting address for a specified range of addresses that are to be used for the data. This leading address doesn't have to be the beginning of the table.

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.

Code Example

```
%@3
// ************************************************************************
// * Please create D data space on the data table as the following:
// * D0 - D3 AS 1 BYTE BCD
// * D10, D20 AS 1 BYTE BCD
// ************************************************************************

// Constant Declaration, Declare D0 as 1 byte BCD
RD          R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
1           //CONSTANT
D0          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D1 as 1 byte BCD
RD          R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
2           //CONSTANT
D1          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D2 as 1 byte BCD
RD          R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
3           //CONSTANT
D2          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D3 as 1 byte BCD
RD          R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
4           //CONSTANT
D3          //CONSTANT OUTPUT ADDRESS


//DSCH      (Data search)
RD R0.0      //BYT = 0, 2-digit BCD
RD.STK R0.0  //RW  = 0, not reset
RD.STK R0.1  //ACT = 1, Execution of the DSCH command
SUB 17       //COMMAND
4            //Data Number Storage Address
D0           //Data Table Leading Address
D10          //the address containing the data you want to search for
D20          //Output data address
WRT R0.2     //Error Output W1=1: NOT FOUND


%
```

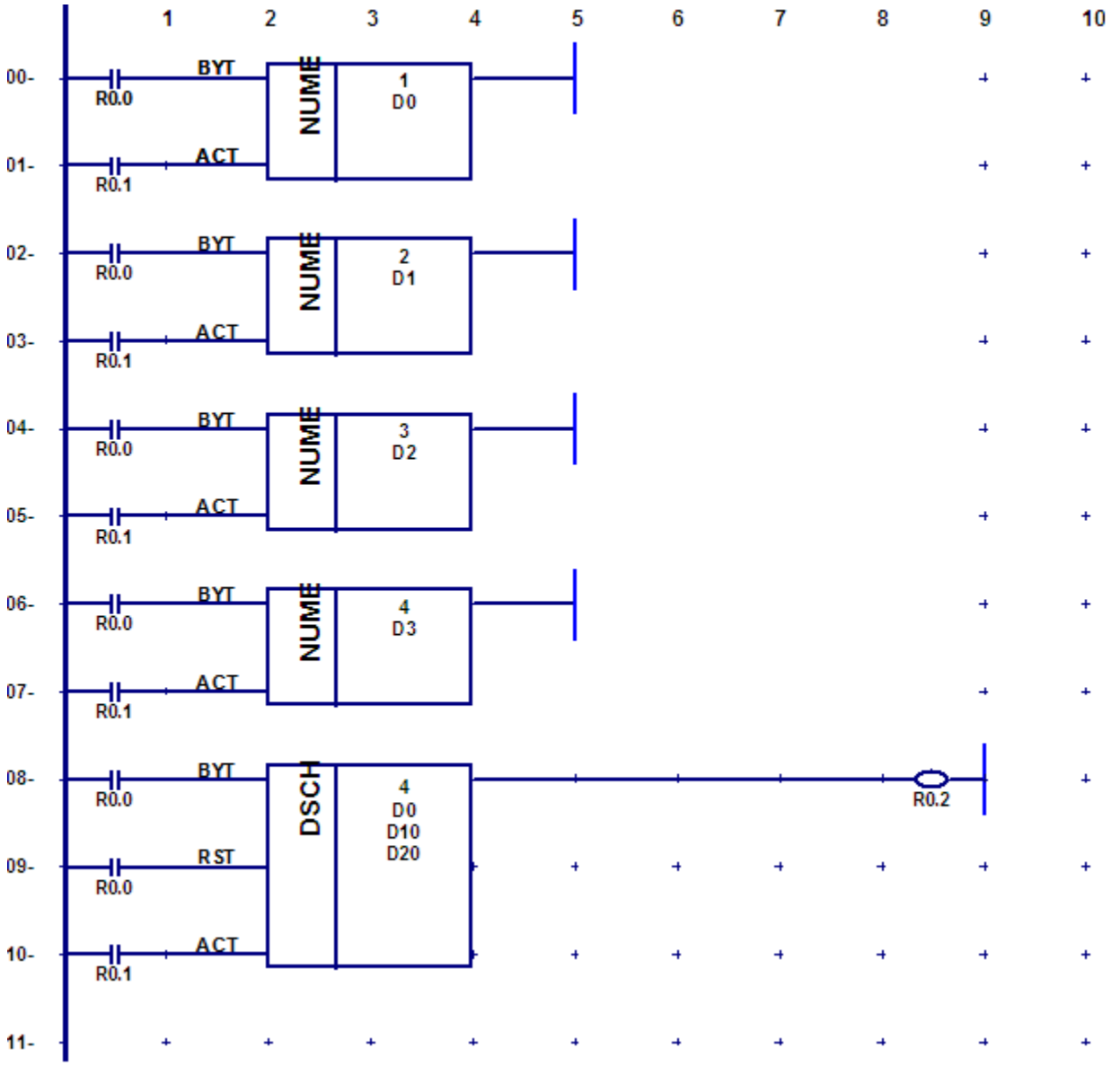


Figure 12-3: DSCH Code Example, Ladder View

12.2 DSCHB (Binary Data Search)

Function

This command searches the data inside the data table like the DSCH command (*Section 12.1: DSCH (Data Search)*). 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.

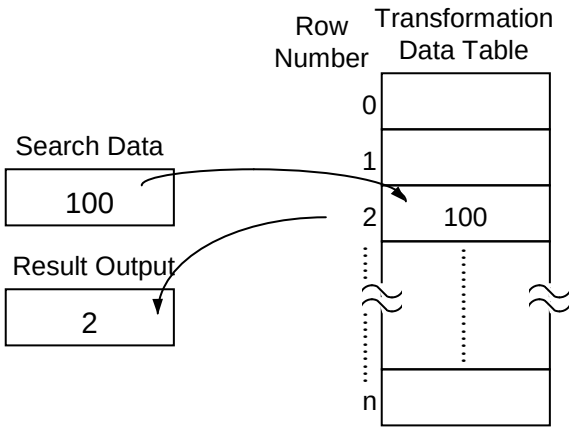
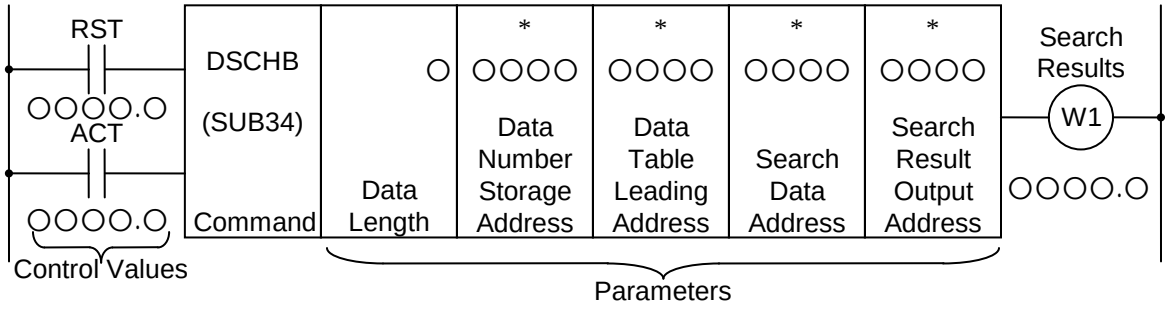


Figure 12-4: Function of the DSCHB Command

Format

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 12-5: 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.

Code Example

```
%@3

// ************************************************************************
// * Please create D data space on the data table as the following:
// * D0  - D3 : 1 byte BCD
// ************************************************************************

// BIN Constant Declaration
RD    R0.0  //ACT=1, RUN NUMB
SUB 40
1           //DATA LENTH
1           //DATA
D0          //OUT PUT DATA ADDRESS

RD    R0.0  //ACT=1, RUN NUMB
SUB 40
1           //DATA LENTH
2           //DATA
D1          //OUT PUT DATA ADDRESS

RD    R0.0  //ACT=1, RUN NUMB
SUB 40
1           //DATA LENTH
3           //DATA
D2          //OUT PUT DATA ADDRESS

RD    R0.0  //ACT=1, RUN NUMB
SUB 40
1           //DATA LENTH
4           //DATA
D3          //OUTPUT DATA ADDRESS


RD.STK      R1.0  //RST = 0, no reset
RD.STK      R0.0  //ACT = 1,DSCHB start
SUB 34
1           //DATA LENGTH
D0          //NUMBER OF ROW
D1          //Data Table Leading Address
D10         //Output Data Storage Address
D20         //Table Internal Number Storage Address
WRT   R0.2  //Error Output (W1)=0: No error

%
```

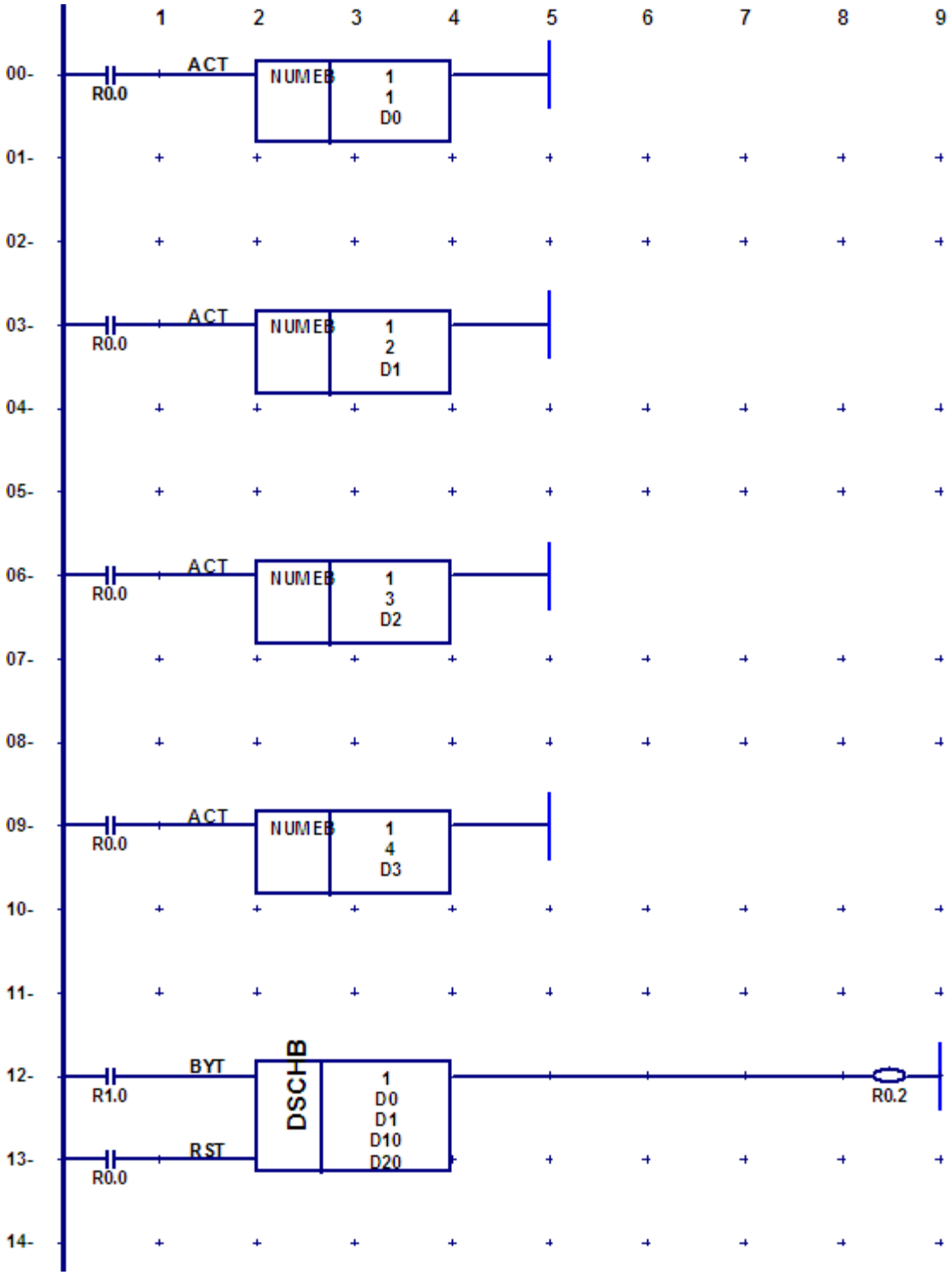


Figure 12-6: DSCHB Code Example, Ladder View

12.3 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.

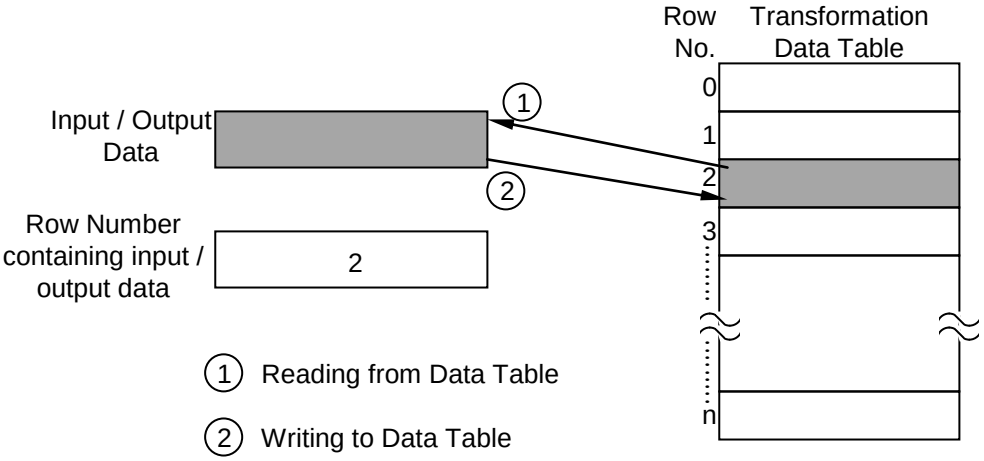


Figure 12-7: Reading from and Writing to the Data Table for the XMOV Command

CAUTION

The leading data table address selected in the XMOV command parameters becomes the 0th table internal number (row number). This table internal number differs from the table internal number mentioned in *Section 3.3 PLC Data Table(s)* in the *LadderWorks PLC Reference Manual*.

Format

The following figure shows the format for describing the command, and Table 4-16 shows the coding format.

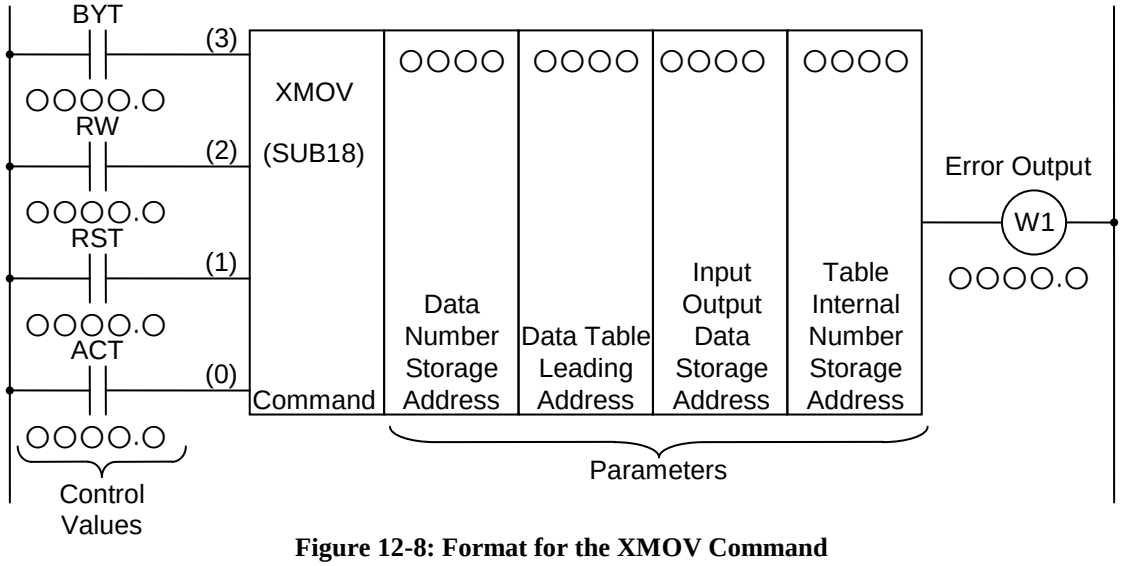


Figure 12-8: Format for the XMOV Command

Coding Sheet				Result History Register				
Step No.	Command	Address No. Bit No.	Description	ST3	ST2	ST1	ST0	
1	RD	○○○○ . ○	BYT				BYT	control values
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	command
6	(PRM)	○○○○	Data Number Storage Address	BYT	RW	RST	ACT	parameters
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	result

Table 12-2: 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.)

Code Example

```
%@3
// ************************************************************************
// * Please set K0.0=0 K0.1=0 K0.2=1
// * Please create D data space on the data table as the following:
// * D7  - D11 : 1 byte BCD
// ************************************************************************

RD R0.0
AND.NOT R0.0
WRT R0.0

// Constant Declaration, Declare D7 as 1 byte BCD
RD R0.0
RD.NOT.STK R0.1
SUB 23
1
D7

// Constant Declaration, Declare D8 as 1 byte BCD
RD R0.0
RD.NOT.STK R0.1
SUB 23
4
D8

// Constant Declaration, Declare D9 as 1 byte BCD
RD R0.0
RD.NOT.STK R0.1
SUB 23
7
D9
```

```
//Constant Declaration, Declare D10 as 1 byte BCD
RD R0.0
RD.NOT.STK R0.1
SUB 23
55
D10

//Constant Declaration, Declare D11 as 1 byte BCD
RD R0.0
RD.NOT.STK R0.1
SUB 23
99
D11

//XMOV(Index Modify Data Transfer)
RD R0.0      //Data Size (BYT)= 0: 2-digit BCD.
RD.STK K0.0  //Read or Write (RW)= 0: Reads the data from the data table.
RD.STK K0.1  //Reset (RST) = 1: Resets. In other words, W1 becomes "0".
RD.STK K0.2  //Action Command (ACT) = 1: Execution of the XMOV command.
SUB 18       //COMMAND
D4           //Data Number Storage Address
D7           //Data Table Leading Address
D100         //Input Output Data Storage Address
D15          //Table Internal Number Storage Address
WRT R0.2     //Error Output (W1)=0: No error

//From this example, you should see the following actions:
//When D15 = 0, the data in D7  will be Transferred to D100 and R0.2=0
//When D15 = 1, the data in D8  will be Transferred to D100 and R0.2=0
//When D15 = 2, the data in D9  will be Transferred to D100 and R0.2=0
//When D15 = 3, the data in D10 will be Transferred to D100 and R0.2=0
//When D15 = 4, the data in D11 will be Transferred to D100 and R0.2=0
//When D15 = 5, the data in D11 will be Transferred to D100 and R0.2=1

// ------------------------------------------------------------------------
// 4 digit BCD sample XMOV
// ------------------------------------------------------------------------
// * Please set K0.4=0 K0.5=0 K0.6=1
// * Please create D data space on the data table as the following:
// * D20  - D28 : 2 byte BCD
// ------------------------------------------------------------------------

// Constant Declaration, Declare D20 as 2 byte BCD
// * D20 - D28 : 2 byte BCD
RD.NOT R0.0
RD.NOT.STK R0.1
SUB 23
0
D20

// Constant Declaration, Declare D22 as 2 byte BCD
RD.NOT R0.0
RD.NOT.STK R0.1
SUB 23
4
D22
```

```
// Constant Declaration, Declare D24 as 2 byte BCD
RD.NOT R0.0
RD.NOT.STK R0.1
SUB 23
7
D24

// Constant Declaration, Declare D26 as 2 byte BCD
RD.NOT R0.0
RD.NOT.STK R0.1
SUB 23
55
D26

// Constant Declaration, Declare D28 as 2 byte BCD
RD.NOT R0.0
RD.NOT.STK R0.1
SUB 23
99
D28

//XMOV(Index Modify Data Transfer)
RD.NOT R0.0  //Data Size (BYT)= 0: 2-digit BCD.
RD.STK K0.4  //Read or Write (RW)= 0: Reads the data from the data table.
RD.STK K0.5  //Reset (RST) = 1: Resets. In other words, W1 becomes "0".
RD.STK K0.6  //Action Command (ACT) = 1: Execution of the XMOV command.
SUB 18       //Command
5            //Data Number Storage Address
D20          //Data Table Leading Address
D200         //Input Output Data Storage Address
D30          //Table Internal Number Storage Address
WRT R0.3     //Error Output (W1)=0: No error

//From this example, you should see the following actions:
//When D30 = 0, the data in D20  will be Transferred to D100 and R0.2=0
//When D30 = 1, the data in D22  will be Transferred to D100 and R0.2=0
//When D30 = 2, the data in D24  will be Transferred to D100 and R0.2=0
//When D30 = 3, the data in D26 will be Transferred to D100 and R0.2=0
//When D30 = 4, the data in D28 will be Transferred to D100 and R0.2=0
//When D30 = 5, the data in D30 will be Transferred to D100 and R0.2=0

RD R0.1
OR.NOT R0.1
WRT R0.1

%
```

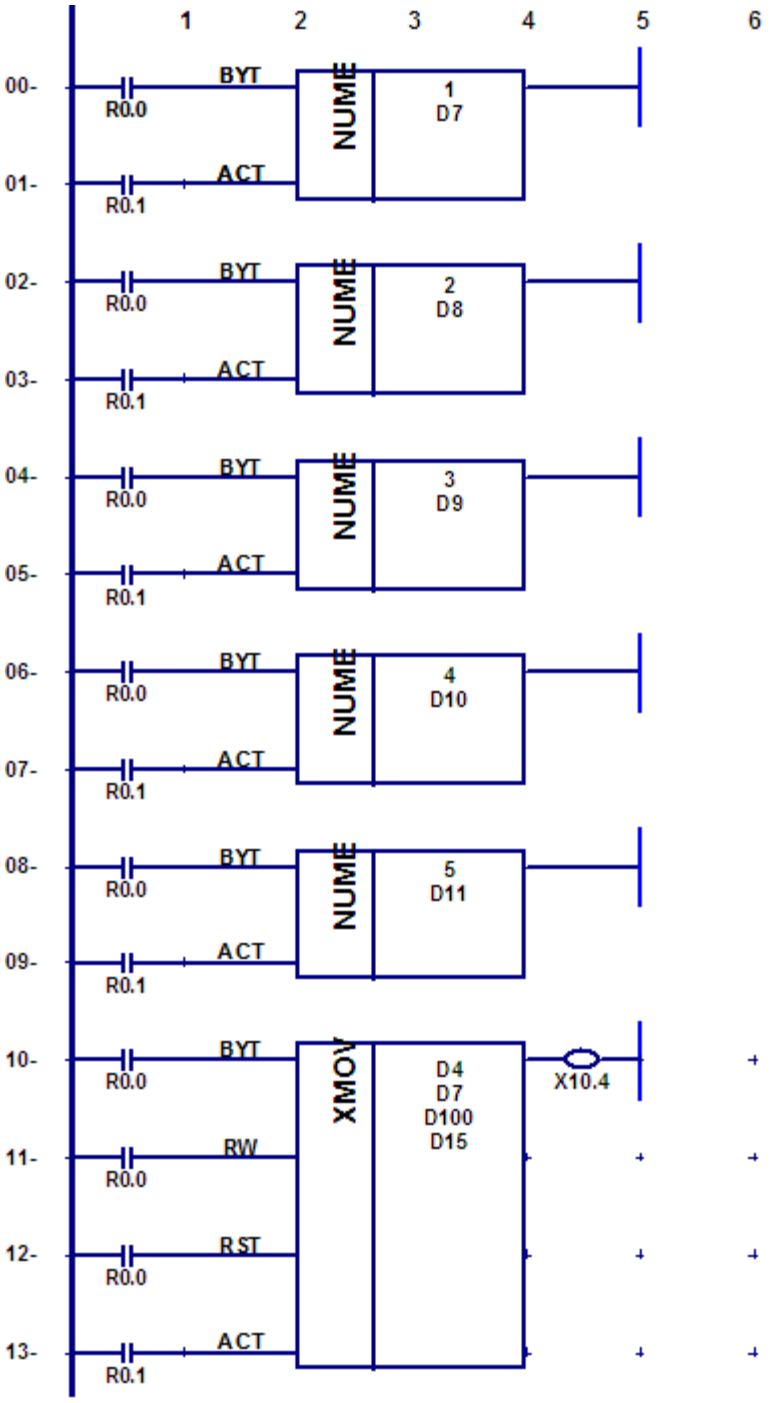


Figure 12-9: XMOV Code Example, Ladder View

12.4 XMOVB (Binary Index Modification Data Transfer)

Function

This command is same as the XMOV Command in *Section 12.3: XMOV (Index Modification Data Transfer)* 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.

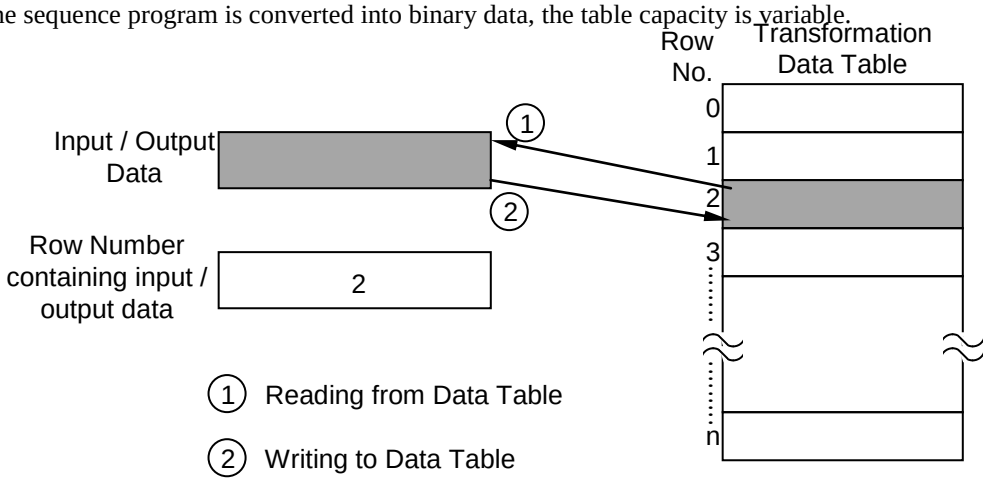
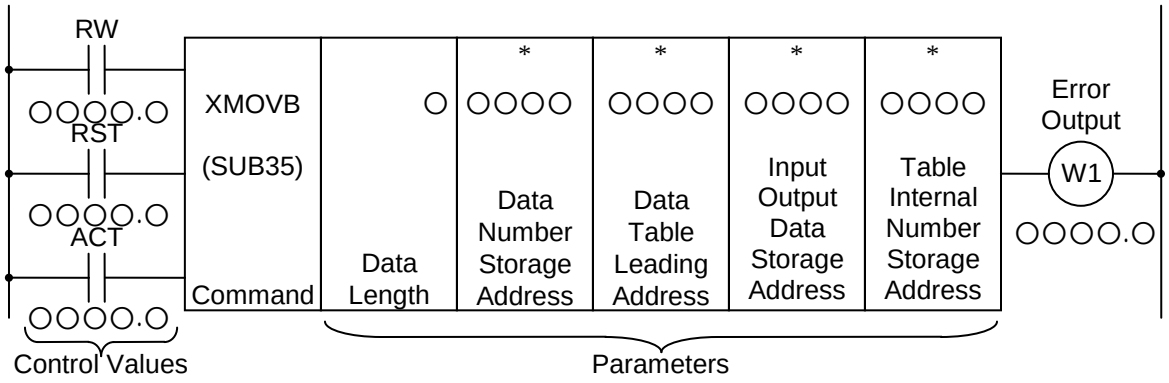


Figure 12-10: Reading from and Writing to the Data Table for the XMOVB Command

Format

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 12-11: 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
Specifies 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.

Code Example

```
%@3
// ************************************************************************
// * Please create D data space on the data table as the following:
// * D0  - D3 : 1 byte BIN


RD K0.0
SUB 40
1
4
D0

RD K0.0
SUB 40
1
5
D1

RD K0.0
SUB 40
1
6
D2

RD K0.0
SUB 40
1
7
D3

RD.STK R0.1  //Read or Write (RW)= 0: Reads the data from the data table.
RD.STK R0.2  //Reset (RST) = 0: No reset.
RD.STK R0.3  //Action Command (ACT) = 1: Execution of the XMOVB command.
SUB 35       //COMMAND
1            //Data Number Storage Address
4            //Number of rows in the data table
D0           //Data Table Leading Address
D200         //Output Data Storage Address
D15          //Table Internal Number Storage Address
WRT R0.2     //Error Output (W1)=0: No error

//From this example, you should see the following actions:
//When D15 = 0, the data in D7  will be Transferred to D200 and R0.2=0
//When D15 = 1, the data in D8  will be Transferred to D200 R0.2=0
//When D15 = 2, the data in D9  will be Transferred to D200 R0.2=0
//When D15 = 3, the data in D10 will be Transferred to D200 R0.2=0

%
```

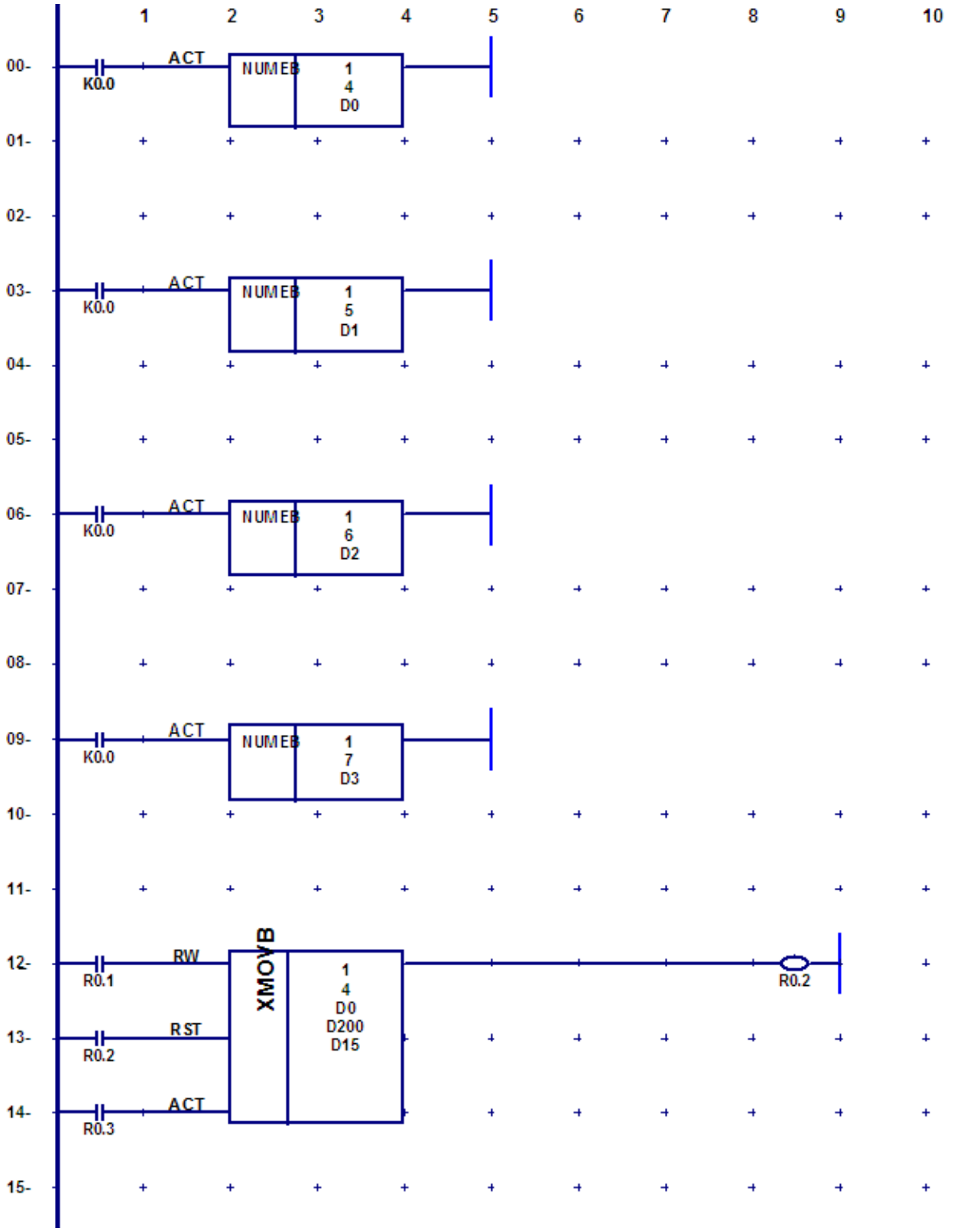


Figure 12-12: XMOVB Code Example, Ladder View

Chapter 13: Function Blocks for Mathematical Operations

13.1 ADD (Addition)

Function

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

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

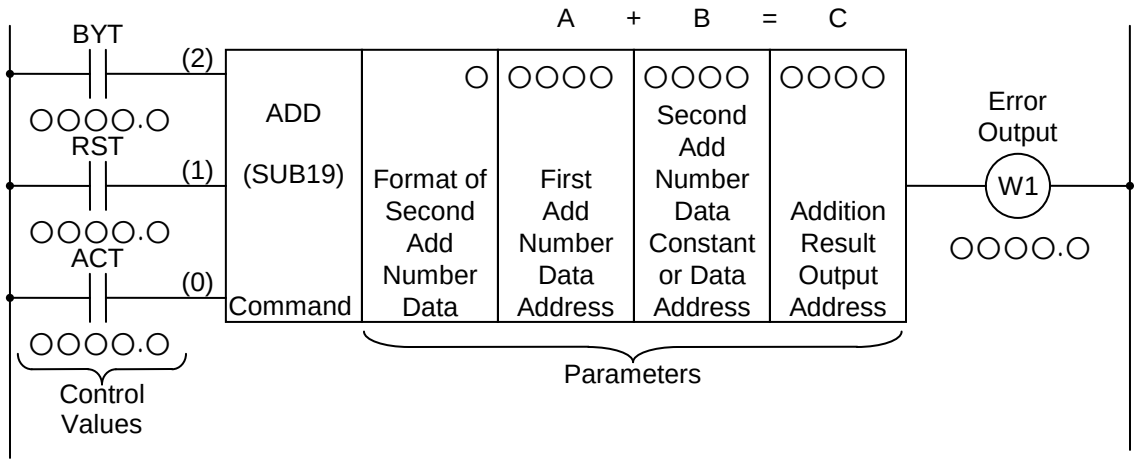


Figure 13-1: Format for the ADD Command

Coding Sheet				Result History Register				
Step No.	**Command**	**Address No. Bit No.**	**Description**	**ST3**	**ST2**	**ST1**	**ST0**	
1	RD	○○○○ . ○	BYT				BYT	control values
2	RD.STK	○○○○ . ○	RST			BYT	RST	
3	RD.STK	○○○○ . ○	ACT		BYT	RST	ACT	
4	SUB	19	ADD Command		BYT	RST	ACT	command
5	(PRM)	○	Format of Second Add Number Data		BYT	RST	ACT	parameters
6	(PRM)	○○○○	First Add Number Data Address		BYT	RST	ACT	
7	(PRM)	○○○○	Second Add Number Data Constant or Data Address		BYT	RST	ACT	
8	(PRM)	○○○○	Addition Result Output Address		BYT	RST	ACT	
9	WRT	○○○○ . ○	W1, Error Output		BYT	RST	W1	result

Table 13-1: 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 Second Add Number Data (B)
   0: Specifies add data as a constant.
   1: Specifies add data as an address.

2) First Add Number Data Address (A)
This is the address where the add number data is stored. Must be an address, not a constant.

3) Second Add Number Data Constant or Data Address (B)
The format for the specification of the add number data is determined by Parameter #1: Format of Second Add Number Data.

4) Addition Result Output Address (for A + B = C)
The address where the addition result is output. Must be an address, not a constant.

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.)

Code Example

```
%@3

RD    K0.0
RD.STK      X0.1
RD.STK      X1.1

SUB 19

1           //USE ADDRESS FORMAT
D0          // A
D1          // B
D2          // A+B=C
WRT  R0.0   //ERR OUTPUT

%
```

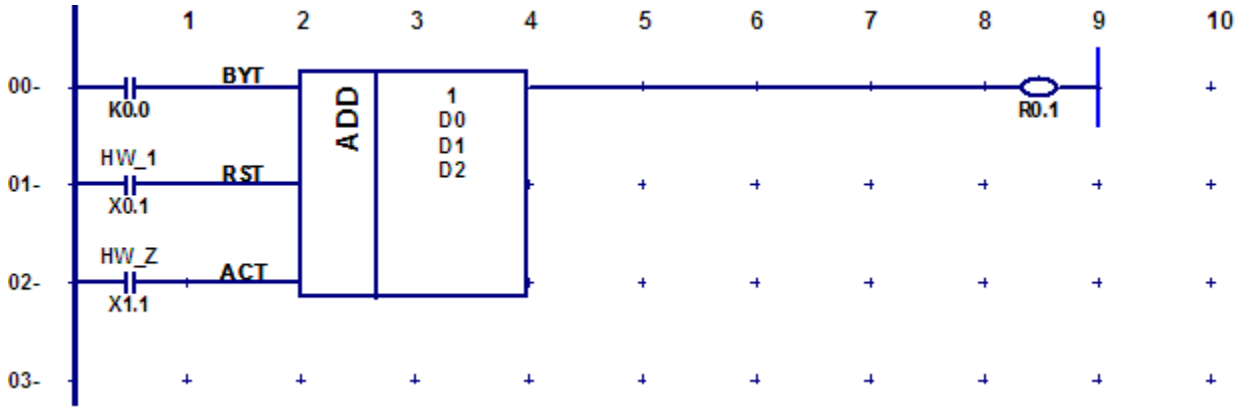


Figure 13-2: ADD Code Example, Ladder View

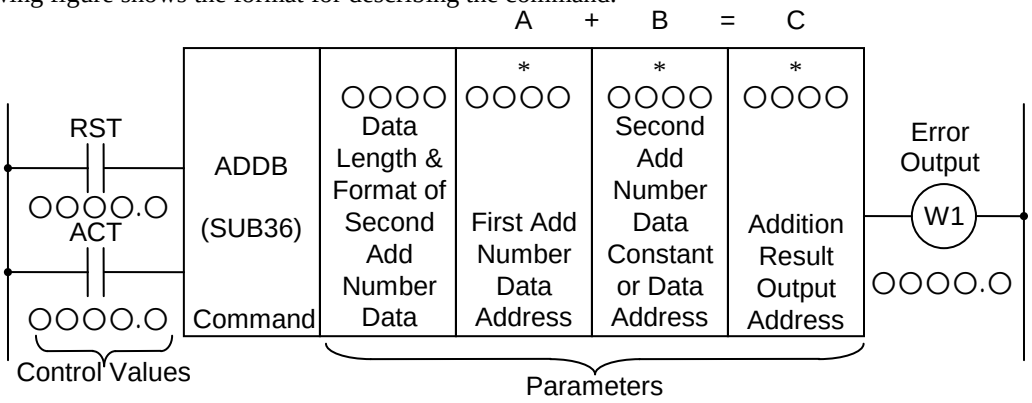
13.2 ADDB (Binary Addition)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 13-3: 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 Second Add Number Data (B)
   The data size (1, 2, or 4 bytes) and the format of the add number data (constant data or address data).

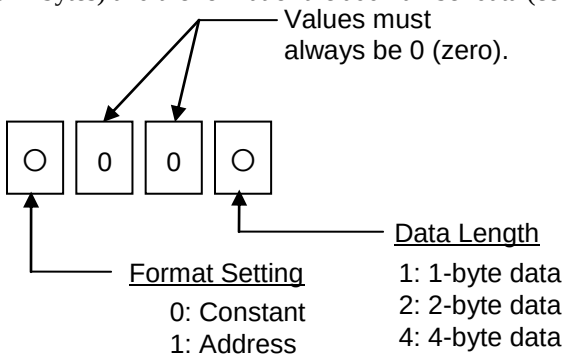


Figure 13-4: Parameters Format Specification for the ADDB Command

2) First Add Number Data Address (A)
Address where the add number data is stored. Must be an address, not a constant.

3) Second Add Number Data Constant or Data Address (B)
The format of the add number data (constant or address) is determined by parameter #1: Data Length and Format of Second Add Number Data.

4) Addition Result Output Address (A + B = C)
The address where the addition result is output. Must be an address, not a constant.

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:

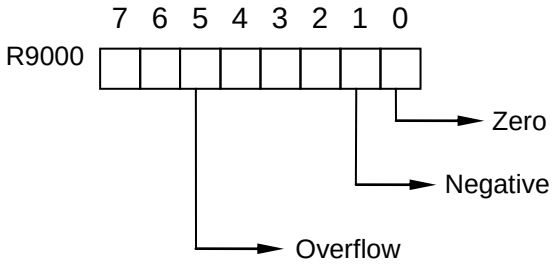


Figure 13-5: Calculation Result Register for the ADDB Command

Code Example

```
%@3
// ******************************************
// * Please create D data space in the data table as follows:
// * D0 - D1 AS 1 BYTE BCD
// ******************************************

RD.STK      R0.0
RD.STK      R0.1
SUB 36
1001
D0    //A
D1    //B
D2    //A+B=C
WRT R1.1
%
```

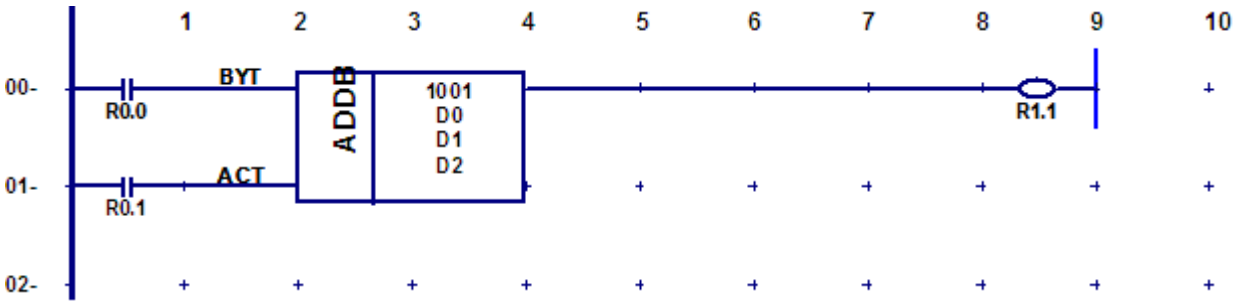


Figure 13-6: ADDB Code Example, Ladder View

13.3 SUB (Subtraction)

Function

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

Format

The following figure shows the format for describing the command, and the following table shows the coding format.

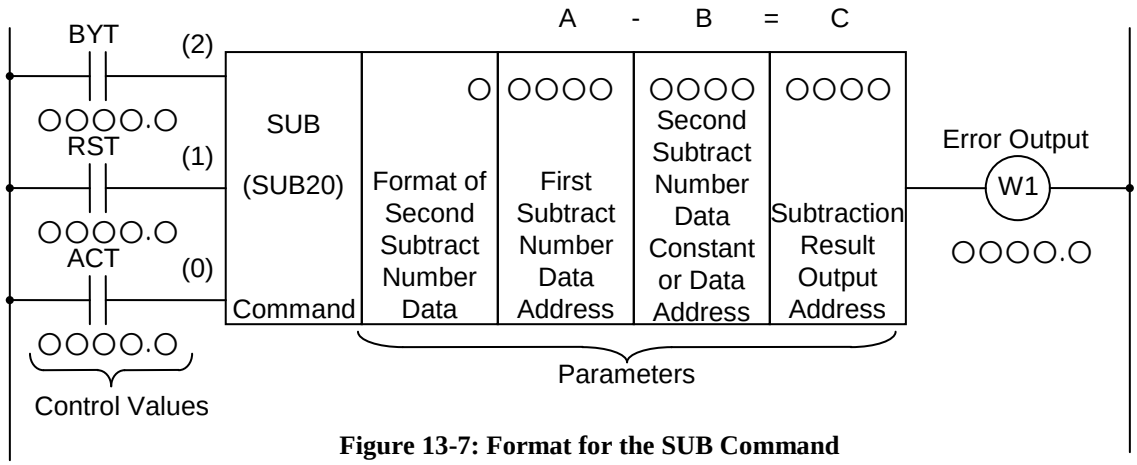


Figure 13-7: Format for the SUB Command

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

Table 13-2: 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 Second Subtract Number Data (B)
   0: Specifies subtract number data as a constant.
   1: Specifies subtract number data as an address.

2) First Subtract Number Data Address (A)
   The address where the subtract number is stored. Must be an address, not a constant.

3) Second Subtract Number Data Constant or Data Address (B)
   Format specification of the subtract number data is determined by Parameter #1: Format of Second Subtract Number Data.

4) Subtraction Result Output Address (A – B = C)
   The address where the subtraction result is output. Must be an address, not a constant.

Error Output (W1)

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

Code Example

```
%@3
RD    R0.0  //BYT=0, 1BYTE BCD
RD.STK      R0.0  //RST=0, NO RESET
RD.STK      R0.1  //ACT=1, RUN SUB COMMAND
SUB 20
1           //O:NUMBER, 1:ADDRESS
D1          //A
D2          //B
D3          //A - B = C
WRT   R0.2  //W1=1, ERROR
%
```

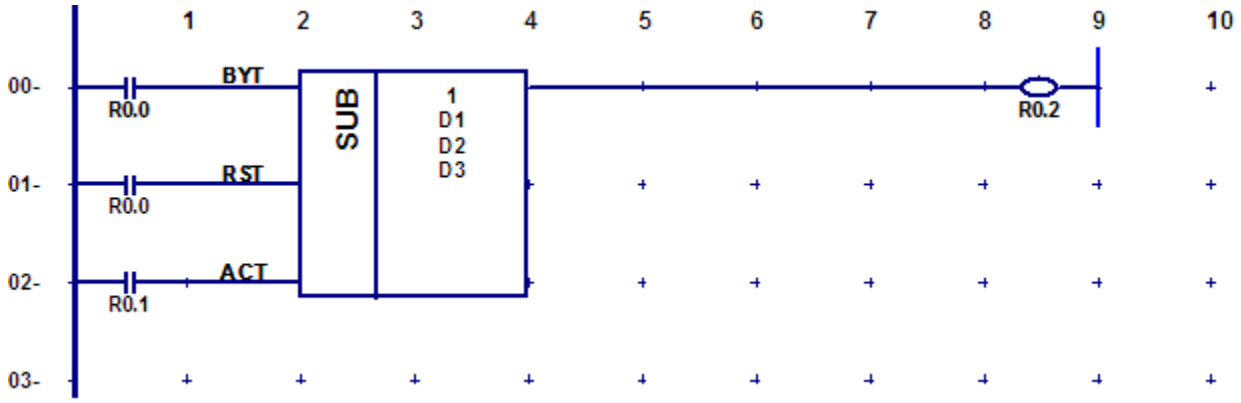


Figure 13-8: SUB Code Example, Ladder View

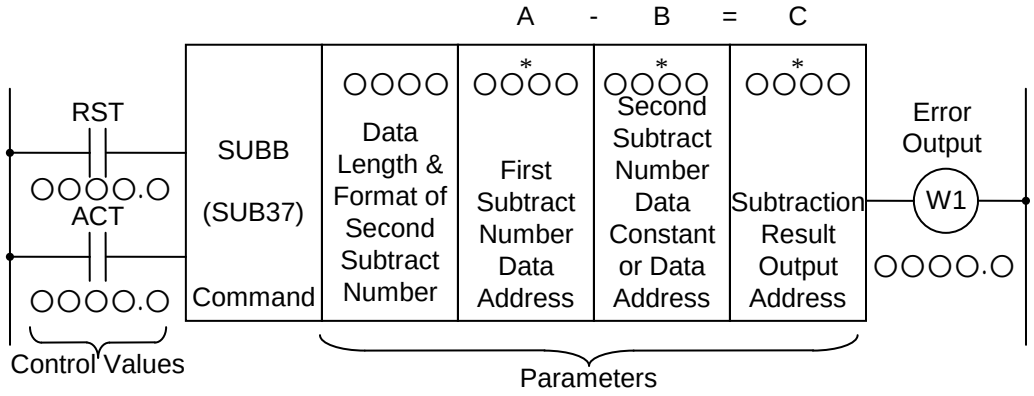
13.4 SUBB (Binary Subtraction)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 13-9: 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 Second Subtract Number Data (B)
   The data size (1, 2, or 4 bytes) and format of the subtract number data (constant data or address data).

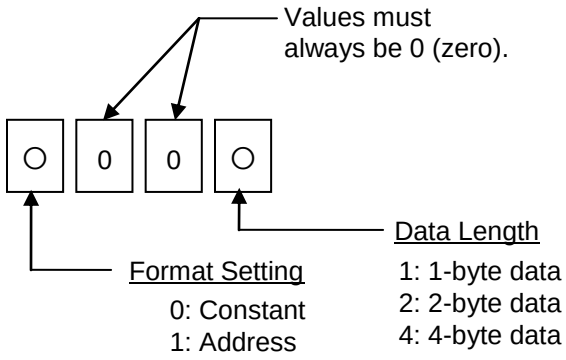


Figure 13-10: Parameters Format Specification for the SUBB Command

2) First Subtract Number Data Address (A)
The address where the subtract number data is stored. Must be an address, not a constant.

3) Second Subtract Number Data Constant or Data Address (B)
The format of the subtract number data (constant or address) is determined by parameter #1: Data Length and Format of Second Subtract Number Data.

4) Subtraction Result Output Address (A – B = C)
The address where the subtraction result is to be output. Must be an address, not a constant.

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:

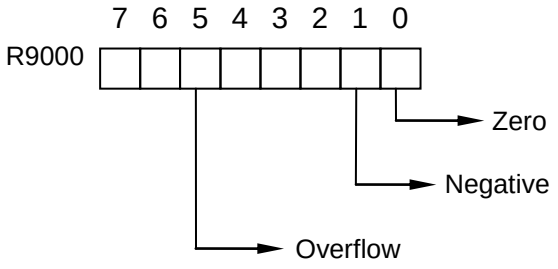


Figure 13-11: Calculation Result Register for the SUBB Command

Code Example

```
%@3

RD.STK R0.0 //RST=0, NO RESET
RD.STK R0.1 //ACT=1, RUN SUBB COMMAND
SUB 37
1001        //SEE BELOW FOR DETAILS
D0          //A
D1          //B
D2          //A - B = C
WRT   R0.1  //W1=1, ERROR

//X00X
//|||+-->1:1 BYTE, 2:2 BYTES, 4:4 BYTES DATA
//||+--->ALWAYS 0
//|+---->ALWAYS 0
//+----->0:NUMBER, 1:ADDRESS

%
```

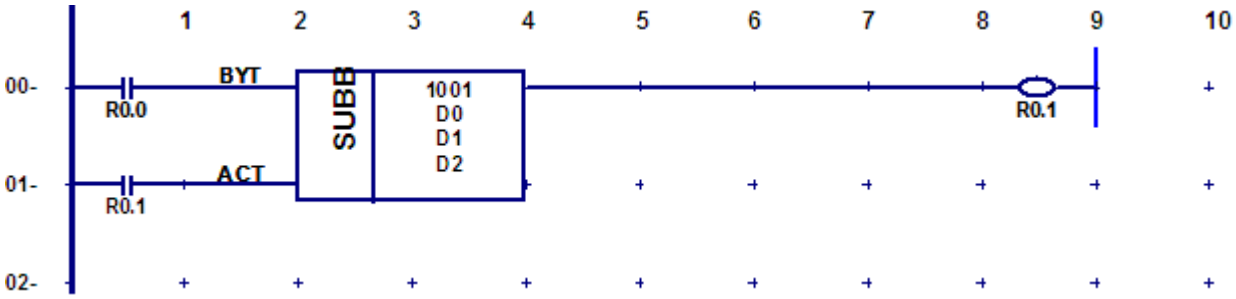


Figure 13-12: SUBB Code Example, Ladder View

13.5 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

The following figure shows the format for describing the command, and the following table shows the coding format.

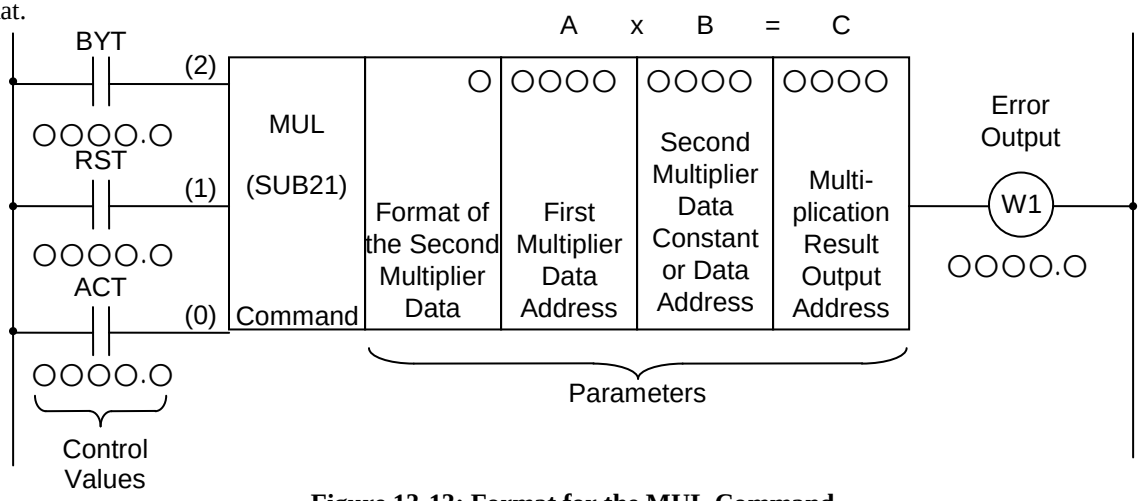


Figure 13-13: Format for the MUL Command

Coding Sheet				Result History Register				
Step No.	Command	Address No. Bit No.	Description	ST3	ST2	ST1	ST0	
1	RD	OOOO . O	BYT				BYT	control values
2	RD.STK	OOOO . O	RST			BYT	RST	
3	RD.STK	OOOO . O	ACT		BYT	RST	ACT	
4	SUB	21	MUL Command		BYT	RST	ACT	command
5	(PRM)	O	Format of Second Multiplier Data		BYT	RST	ACT	parameters
6	(PRM)	OOOO	First Multiplier Data Address		BYT	RST	ACT	
7	(PRM)	OOOO	Second Multiplier Data Constant or Data Address		BYT	RST	ACT	
8	(PRM)	OOOO	Multiplication Result Output Address		BYT	RST	ACT	
9	WRT	OOOO . O	W1, Error Output		BYT	RST	W1	result

Table 13-3: 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 Second Multiplier Data (B)
   0: Specifies multiplier data as a constant.
   1: Specifies multiplier data as an address.

2) First Multiplier Data Address (A)
The address where the multiplier is stored. Must be an address, not a constant.

3) Second Multiplier Data Constant or Data Address (B)
The specification format of the multiplier data is determined by Parameter #1: Format of Second Multiplier Data.

4) Multiplication Result Output Address (A x B = C)
The address where the multiplication result is to be output. Must be an address, not a constant.

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.)

Code Example

```
%@3

RD    R0.0  // BYT=0, DATA IS 1 BYTE BCD
RD.STK      R0.0  // RST=0, NO RESET
RD.STK      R0.1  // ACT=1, RUN MUL COMMOND
SUB 21
1           // 0:NUMBER, 1:ADDRESS
D1          // A
D3          // B
D5          // A * B = C
WRT R1.0    // ERROR OUTPUT
```

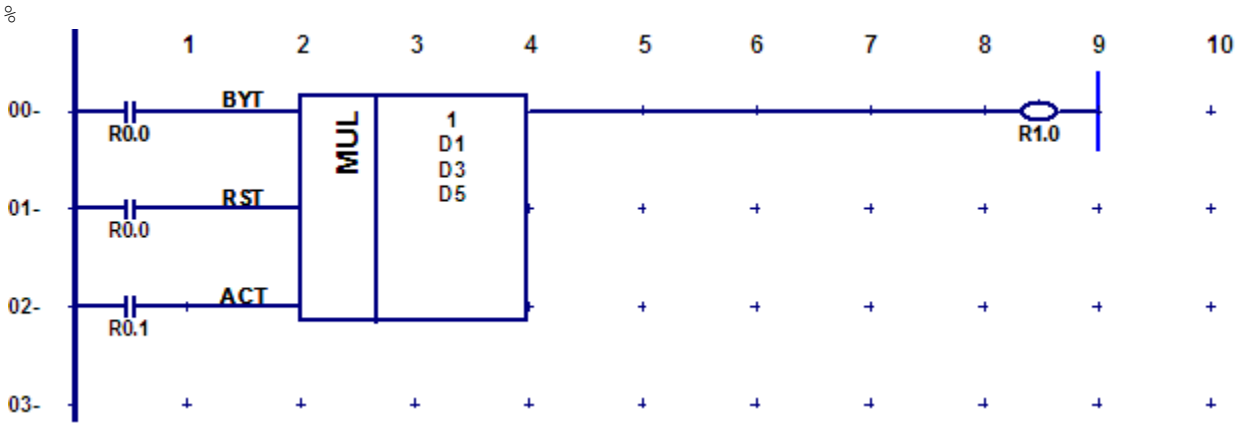


Figure 13-14: MUL Code Example, Ladder View

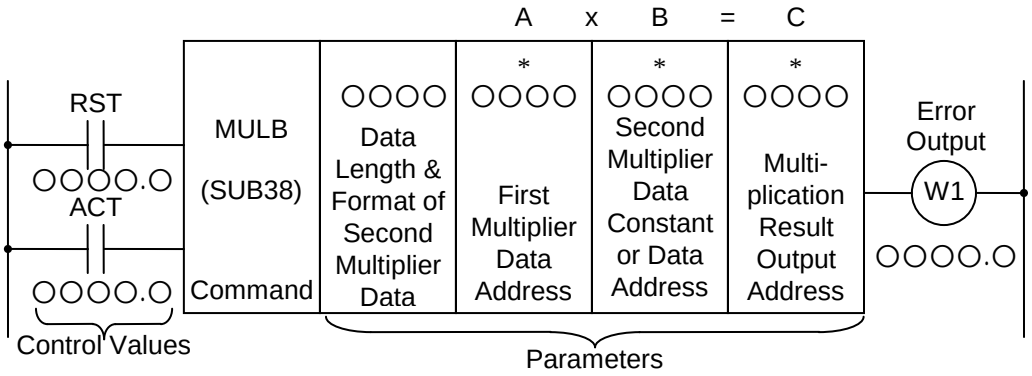
13.6 MULB (Binary Multiplication)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 13-15: 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 Second Multiplier Data (B)
   The data size (1, 2, or 4 bytes) and the format of the multiplier data (constant data or address data).

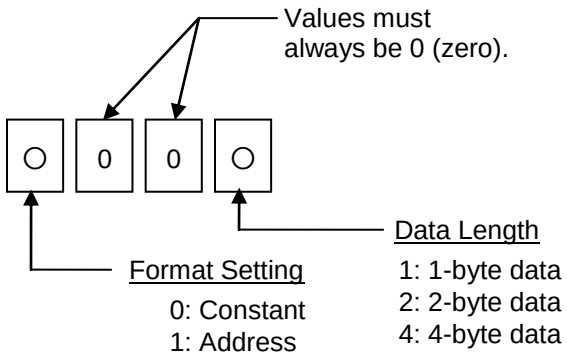


Figure 13-16: Parameters Format Specification for the MULB Command

2) First Multiplier Data Address (A)
The address where the multiplier data is stored. Must be an address, not a constant.

3) Second Multiplier Data Constant or Data Address (B)
The format of the multiplier data (constant or address) is determined by parameter #1: Data Length and Format of Second Multiplier Data.

4) Multiplication Result Output Address (A x B = C)
The address where the multiplication result is to be output. Must be an address, not a constant.

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:

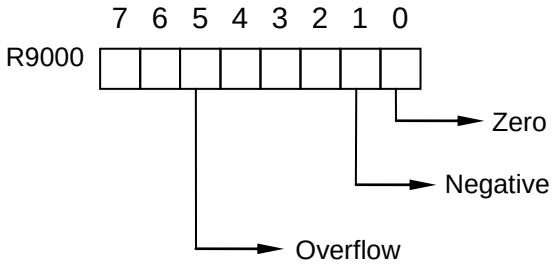


Figure 13-17: Calculation Result Register for the MULB Command

Code Example

```
%@3

RD.STK R0.0 //RST=0, NOT RESET
RD.STK R0.1 //ACT=1, RUN MULB COMMAND
SUB 38
1001        //SEE THE END OF THIS FILE FOR DETAILS
D0          //A
D1          //B
D2          //A * B = C
WRT R1.1    //ERROR OUTPUT W1=1, ERROR.

//X00X
//|||+-->1:1 BYTE, 2:2 BYTES, 4:4 BYTES DATA
//||+--->ALWAYS 0
//|+---->ALWAYS 0
//+----->0:NUMBER, 1:ADDRESS

%
```

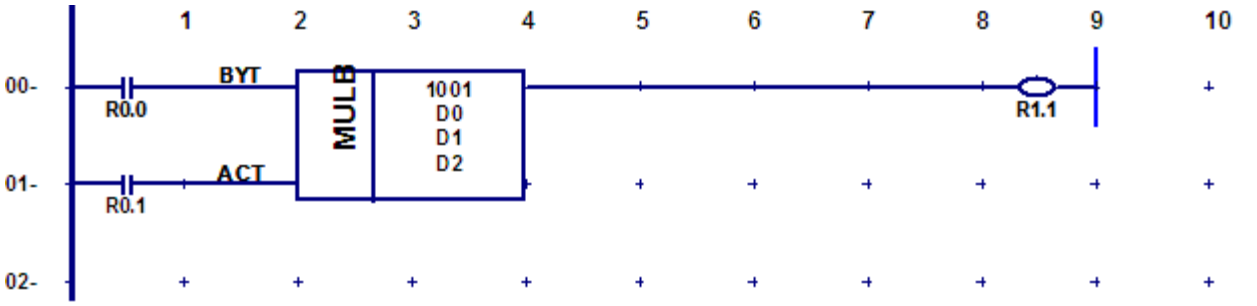


Figure 13-18: MULB Code Example, Ladder View

13.7 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

The following figure shows the format for describing the command, and the following table shows the coding format.

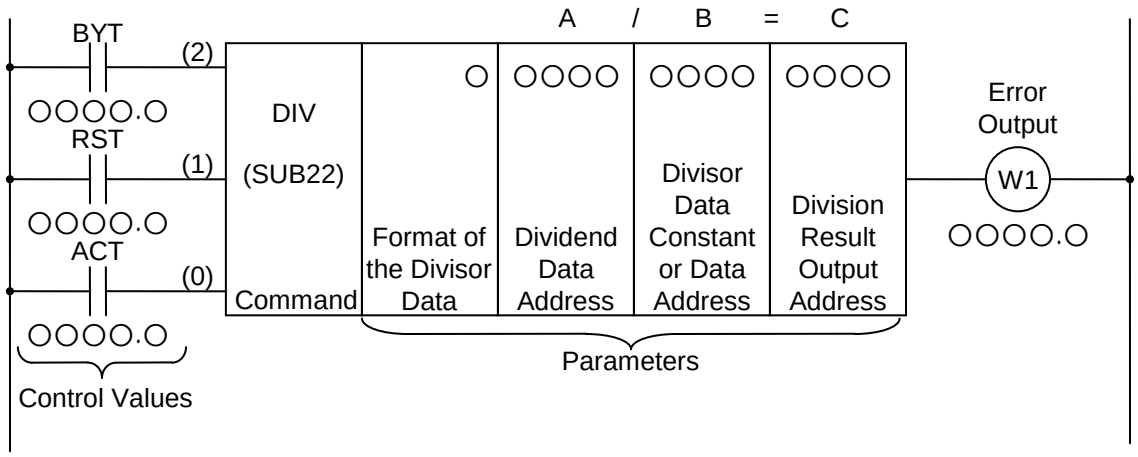


Figure 13-19: Format for the DIV Command

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

Table 13-4: 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 (B)
   0: Specifies divisor data as a constant.
   1: Specifies divisor data as an address.

2) Dividend Data Address (A)
The address where the dividend is stored. Must be an address, not a constant.

3) Divisor Data Constant or Data Address (B)
The specification format of the divisor data is determined by Parameter #1: Format of the Divisor Data.

4) Division Result Output Address (A / B = C)
The address where the division result is to be output. Must be an address, not a constant.

Error Output (W1)

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

Code Example

```
%@3
// ************************************************************************
// * Please create D data space on the data table as the following:
// * D1 AS 2 BYTE BCD
// * D3 AS 2 BYTE BCD
// * D5 AS 2 BYTE BCD
// ************************************************************************

RD    R0.0  //BYT=0, 2 BYTE BCD
RD.STK      R0.1  //RST=0, NO RESET
RD.STK      R0.2  //ACT=1, RUN DIV COMMAND
SUB 22
1           //DATA ADDRESS
D1          //A
D3          //B
D5          //C A/B=C
WRT R1.0    //ERROR OUTPUT
```

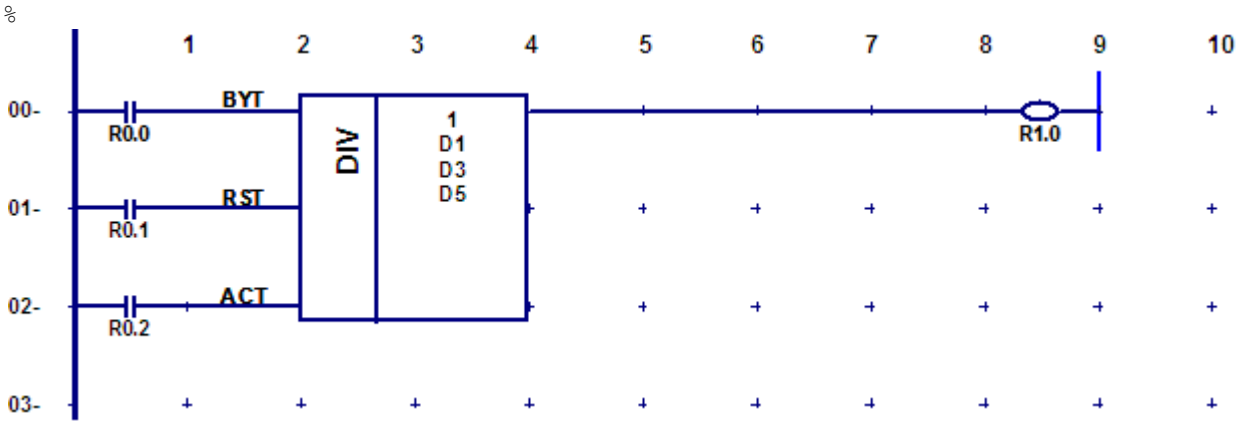


Figure 13-20: DIV Code Example, Ladder View

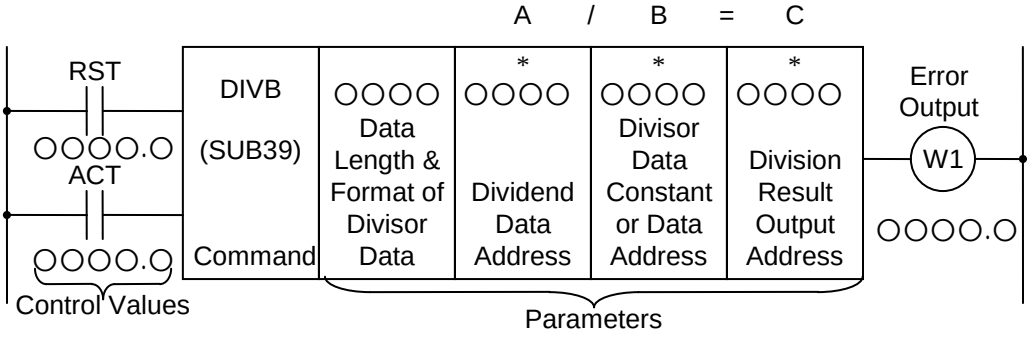
13.8 DIVB (Binary Division)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 13-21: 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 (B)
   The data length (1, 2, or 4 bytes) and the format of the divisor data (constant data or address data).

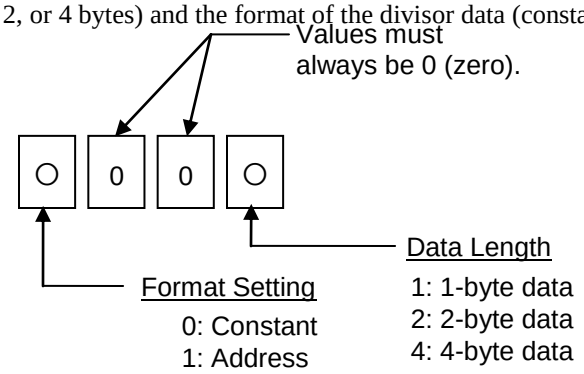


Figure 13-22: Parameters Format Specification for the DIVB Command

2) Dividend Data Address (A)
The address where the dividend data is to be stored. Must be an address, not a constant.

3) Divisor Data Constant or Data Address (B)
The format of the divisor data (constant or address) is determined by parameter #1: Data Length and Format of Divisor Data.

4) Division Result (Quotient) Output Address (A / B = C)
The address where the division result data is to be output. Must be an address, not a constant.

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:

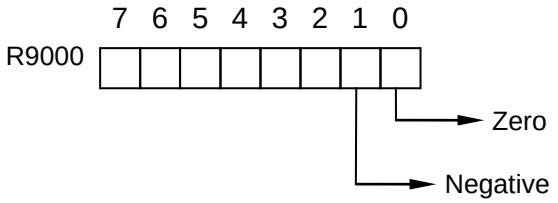


Figure 13-23: Calculation Result Register for the DIVB Command

Remainder Data Output Address

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

Code Example

```
%@3
// ************************************************************************
// * Please create D data space in the data table as follows:
// * D0 - D2 AS 1 BYTE BCD
// ************************************************************************


RD.STK      R0.0  //RST=0, NO RESET
RD.STK      R0.1  //ACT=1, RUN DIVB COMMAND
SUB 39
1001        //SEE THE END FOR DETAIL
D0          //DATA ADDRESS
D1          //DATA ADDRESS/NUMBER
D2
WRT R1.1    //ERROR OUTPUT, R1.1=1, ERROR

//1001
//|||+->1:1 BYTE, 2:2 BYTES, 4:4 BYTES DATA
//||+-->ALWAYS 0
//|+--->ALWAYS 0
//+---->0:NUMBER, 1:ADDRESS
```

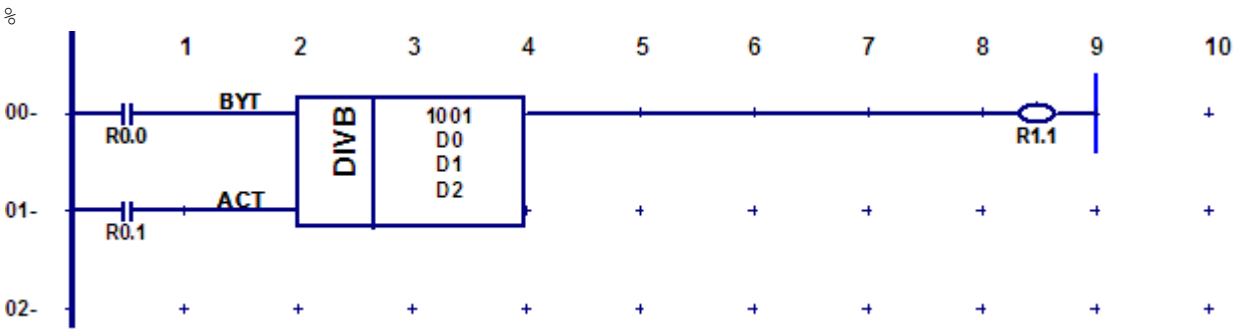


Figure 13-24: DIVB Code Example, Ladder View

Chapter 14: Constant Declaration Function Blocks

14.1 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

The following figure shows the format for describing the command, and the following table shows the coding format.

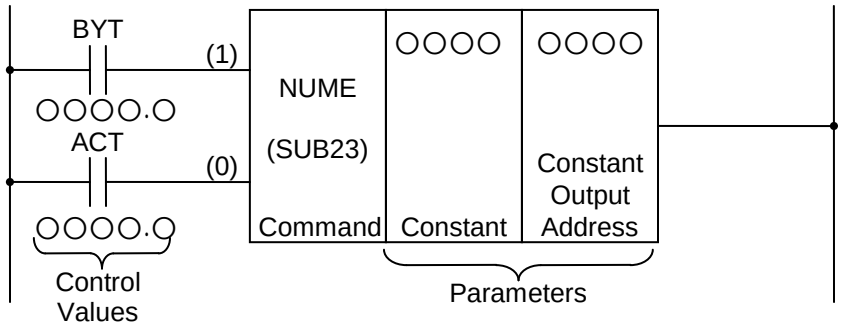


Figure 14-1: Format for the NUME Command

Coding Sheet / **Result History Register**

Step No.	Command	Address No.	Bit No.	Description	ST3	ST2	ST1	ST0	
1	RD	OOOO	O	BYT				BYT	control values
2	RD.STK	OOOO	O	ACT			BYT	ACT	
3	SUB	23		NUME Command			BYT	ACT	command
4	(PRM)	OOOO		Constant			BYT	ACT	parameters
5	(PRM)	OOOO		Constant Output Address			BYT	ACT	

Table 14-1: 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.

Code Example

```
%@3

// Constant Declaration, Declare D0 as 1 byte BCD
RD    R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
1           //CONSTANT
D0          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D1 as 1 byte BCD
RD    R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
2           //CONSTANT
D1          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D2 as 1 byte BCD
RD    R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
3           //CONSTANT
D2          //CONSTANT OUTPUT ADDRESS

// Constant Declaration, Declare D3 as 1 byte BCD
RD    R0.0  //BYT=0, 1 BYTE BCD
RD.STK      R0.1  //ACT=1, RUN NUME CONSTANT DECLARATION
SUB 23
4           //CONSTANT
D3          //CONSTANT OUTPUT ADDRESS


%
```

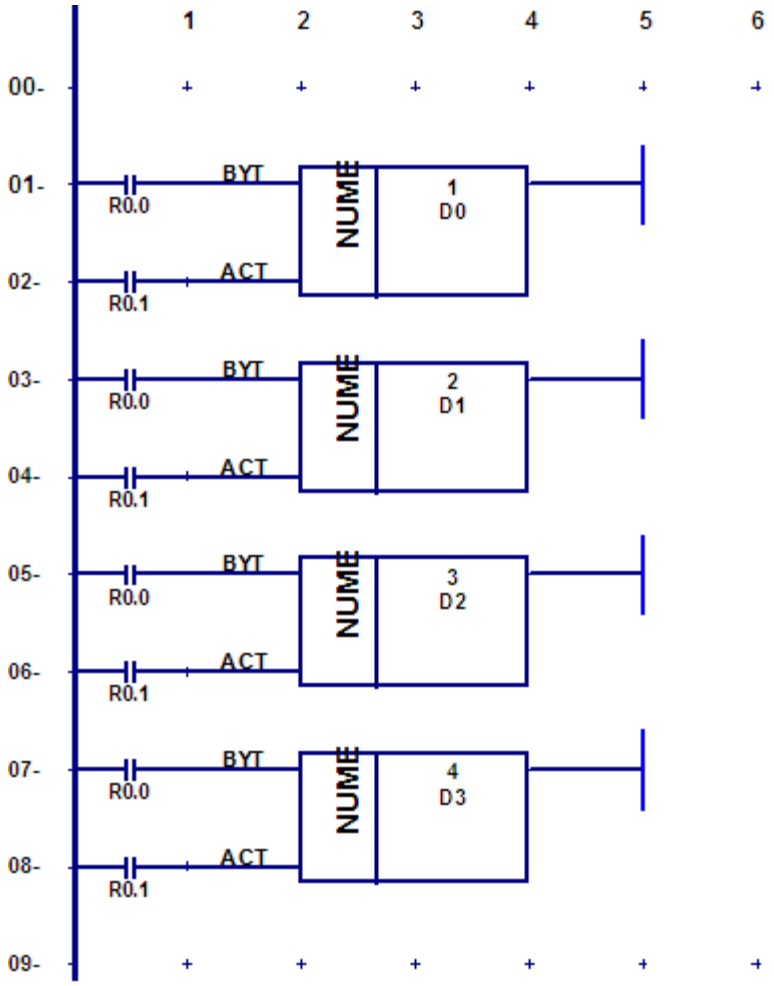


Figure 14-2: NUME Code Example, Ladder View

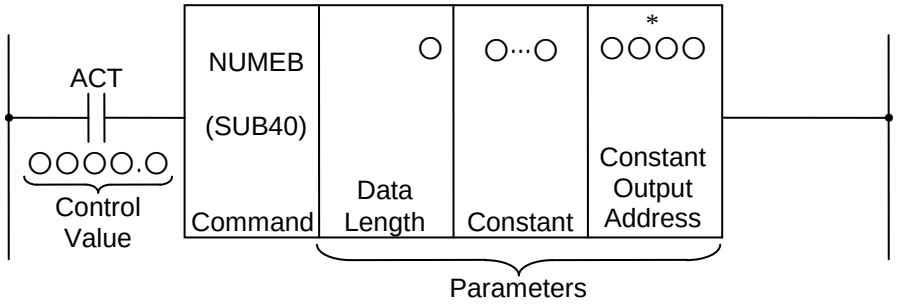
14.2 NUMEB (Binary Constant Declaration)

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

The following figure shows the format for describing the command.



* When the data for this address is either 2 or 4 bytes, you can optimize the command to speed up execution by using an even number for that address.

Figure 14-3: 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.

Code Example

```
%@3

RD.STK      R0.0  //ACT=1, RUN NUMEB BIN CONSTANT DECLARATION
SUB 40
1           //DATA LENGTH
4           //DATA
D0          //CONSTANT OUTPUT ADDRESS

%
```

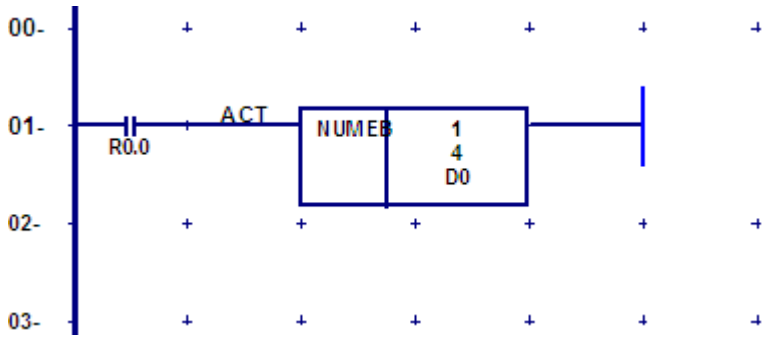


Figure 14-4: NUMEB Code Example, Ladder View

Index