

FANUC Robot series

R-30iB/ R-30iB Mate CONTROLLER

Integrated PMC

OPERATOR'S MANUAL

MAROBIPMC04121E REV. B

This publication contains proprietary information
of FANUC America Corporation furnished for
customer use only. No other uses are authorized
without the express written permission of
FANUC America Corporation.

FANUC America Corporation
3900 W. Hamlin Road
Rochester Hills, Michigan 48309-3253

B-83254EN/02

Copyrights and Trademarks

This new publication contains proprietary information of FANUC America Corporation furnished for customer use only. No other uses are authorized without the express written permission of FANUC America Corporation.

The descriptions and specifications contained in this manual were in effect at the time this manual was approved for printing. FANUC America Corporation, hereinafter referred to as FANUC, reserves the right to discontinue models at any time or to change specifications or design without notice and without incurring obligations.

FANUC manuals present descriptions, specifications, drawings, schematics, bills of material, parts, connections and/or procedures for installing, disassembling, connecting, operating and programming FANUC products and/or systems. Such systems consist of robots, extended axes, robot controllers, application software, the KAREL® programming language, INSIGHT® vision equipment, and special tools.

FANUC recommends that only persons who have been trained in one or more approved FANUC Training Course(s) be permitted to install, operate, use, perform procedures on, repair, and/or maintain FANUC products and/or systems and their respective components. Approved training necessitates that the courses selected be relevant to the type of system installed and application performed at the customer site.

⚠ WARNING

This equipment generates, uses, and can radiate radiofrequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for Class A computing devices pursuant to subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of the equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measure may be required to correct the interference.

FANUC conducts courses on its systems and products on a regularly scheduled basis at the company's world headquarters in Rochester Hills, Michigan. For additional information contact

FANUC America Corporation
Training Department
3900 W. Hamlin Road
Rochester Hills, Michigan 48309-3253
www.fanucrobotics.com

For customer assistance, including Technical Support, Service, Parts & Part Repair, and Marketing Requests, contact the Customer Resource Center, 24 hours a day, at 1-800-47-ROBOT (1-800-477-6268). International customers should call 011-1-248-377-7159.

Send your comments and suggestions about this manual to:
product.documentation@fanucrobotics.com

**Copyright © 2013 by FANUC America Corporation
All Rights Reserved**

The information illustrated or contained herein is not to be reproduced, copied, downloaded, translated into another language, published in any physical or electronic format, including internet, or transmitted in whole or in part in any way without the prior written consent of FANUC America Corporation.

**AccuStat®, ArcTool®, iRVision®, KAREL®, PaintTool®, PalletTool®,
SOCKETS®, SpotTool®, SpotWorks®, and TorchMate® are Registered
Trademarks of FANUC.**

FANUC reserves all proprietary rights, including but not limited to trademark and trade name rights, in the following names:

AccuAir™, AccuCal™, AccuChop™, AccuFlow™, AccuPath™,
AccuSeal™, ARC Mate™, ARC Mate Sr.™, ARC Mate System 1™,
ARC Mate System 2™, ARC Mate System 3™, ARC Mate System 4™,
ARC Mate System 5™, ARCWorks Pro™, AssistTool™, AutoNormal™,
AutoTCP™, BellTool™, BODYWorks™, Cal Mate™, Cell Finder™,
Center Finder™, Clean Wall™, DualARM™, LR Tool™,
MIG Eye™, MotionParts™, MultiARM™, NoBots™, Paint
Stick™, PaintPro™, PaintTool 100™, PAINTWorks™, PAINTWorks
II™, PAINTWorks III™, PalletMate™, PalletMate PC™,
PalletTool PC™, PayloadID™, RecipTool™, RemovalTool™,
Robo Chop™, Robo Spray™, S-420i™, S-430i™, ShapeGen™,
SoftFloat™, SOFT PARTS™, SpotTool+™, SR Mate™, SR
ShotTool™, SureWeld™, SYSTEM R-J2 Controller™, SYSTEM R-J3
Controller™, SYSTEM R-J3iB Controller™, SYSTEM R-J3iC Controller™,
SYSTEM R-30iA Controller™, SYSTEM R-30iA Mate Controller™, SYSTEM
R-30iB Controller™, SYSTEM R-30iB Mate Controller™, TCP Mate™,
TorchMate™, TripleARM™, TurboMove™, visLOC™, visPRO-3D™,
visTRACT™, WebServer™, WebTP™, and YagTool™.

©FANUC CORPORATION 2013

- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

Patents

One or more of the following U.S. patents might be related to the FANUC products described in this manual.

FANUC America Corporation Patent List

4,630,567 4,639,878 4,707,647 4,708,175 4,708,580 4,942,539 4,984,745
5,238,029 5,239,739 5,272,805 5,293,107 5,293,911 5,331,264 5,367,944
5,373,221 5,421,218 5,434,489 5,644,898 5,670,202 5,696,687 5,737,218
5,823,389 5,853,027 5,887,800 5,941,679 5,959,425 5,987,726 6,059,092
6,064,168 6,070,109 6,086,294 6,122,062 6,147,323 6,204,620 6,243,621
6,253,799 6,285,920 6,313,595 6,325,302 6,345,818 6,356,807 6,360,143
6,378,190 6,385,508 6,425,177 6,477,913 6,490,369 6,518,980 6,540,104
6,541,757 6,560,513 6,569,258 6,612,449 6,703,079 6,705,361 6,726,773
6,768,078 6,845,295 6,945,483 7,149,606 7,149,606 7,211,978 7,266,422
7,399,363

FANUC CORPORATION Patent List

4,571,694 4,626,756 4,700,118 4,706,001 4,728,872 4,732,526 4,742,207
4,835,362 4,894,596 4,899,095 4,920,248 4,931,617 4,934,504 4,956,594
4,967,125 4,969,109 4,970,370 4,970,448 4,979,127 5,004,968 5,006,035
5,008,834 5,063,281 5,066,847 5,066,902 5,093,552 5,107,716 5,111,019
5,130,515 5,136,223 5,151,608 5,170,109 5,189,351 5,267,483 5,274,360
5,292,066 5,300,868 5,304,906 5,313,563 5,319,443 5,325,467 5,327,057
5,329,469 5,333,242 5,337,148 5,371,452 5,375,480 5,418,441 5,432,316
5,440,213 5,442,155 5,444,612 5,449,875 5,451,850 5,461,478 5,463,297
5,467,003 5,471,312 5,479,078 5,485,389 5,485,552 5,486,679 5,489,758
5,493,192 5,504,766 5,511,007 5,520,062 5,528,013 5,532,924 5,548,194
5,552,687 5,558,196 5,561,742 5,570,187 5,570,190 5,572,103 5,581,167
5,582,750 5,587,635 5,600,759 5,608,299 5,608,618 5,624,588 5,630,955
5,637,969 5,639,204 5,641,415 5,650,078 5,658,121 5,668,628 5,687,295
5,691,615 5,698,121 5,708,342 5,715,375 5,719,479 5,727,132 5,742,138
5,742,144 5,748,854 5,749,058 5,760,560 5,773,950 5,783,922 5,799,135
5,812,408 5,841,257 5,845,053 5,872,894 5,887,122 5,911,892 5,912,540
5,920,678 5,937,143 5,980,082 5,983,744 5,987,591 5,988,850 6,023,044
6,032,086 6,040,554 6,059,169 6,088,628 6,097,169 6,114,824 6,124,693
6,140,788 6,141,863 6,157,155 6,160,324 6,163,124 6,177,650 6,180,898
6,181,096 6,188,194 6,208,105 6,212,444 6,219,583 6,226,181 6,236,011
6,236,896 6,250,174 6,278,902 6,279,413 6,285,921 6,298,283 6,321,139
6,324,443 6,328,523 6,330,493 6,340,875 6,356,671 6,377,869 6,382,012
6,384,371 6,396,030 6,414,711 6,424,883 6,431,018 6,434,448 6,445,979
6,459,958 6,463,358 6,484,067 6,486,629 6,507,165 6,654,666 6,665,588
6,680,461 6,696,810 6,728,417 6,763,284 6,772,493 6,845,296 6,853,881
6,888,089 6,898,486 6,917,837 6,928,337 6,965,091 6,970,802 7,038,165
7,069,808 7,084,900 7,092,791 7,133,747 7,143,100 7,149,602 7,131,848
7,161,321 7,171,041 7,174,234 7,173,213 7,177,722 7,177,439 7,181,294
7,181,313 7,280,687 7,283,661 7,291,806 7,299,713 7,315,650 7,324,873
7,328,083 7,330,777 7,333,879 7,355,725 7,359,817 7,373,220 7,376,488
7,386,367 7,464,623 7,447,615 7,445,260 7,474,939 7,486,816 7,495,192
7,501,778 7,502,504 7,508,155 7,512,459 7,525,273 7,526,121

Conventions

WARNING

Information appearing under the "WARNING" caption concerns the protection of personnel. It is boxed and bolded to set it apart from the surrounding text.

CAUTION

Information appearing under the "CAUTION" caption concerns the protection of equipment, software, and data. It is boxed and bolded to set it apart from the surrounding text.

Note Information appearing next to NOTE concerns related information or useful hints.

• Original Instructions

Before using the Robot, be sure to read the "FANUC Robot Safety Manual (B-80687EN)" and understand the content.

- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

The products in this manual are controlled based on Japan's "Foreign Exchange and Foreign Trade Law". The export from Japan may be subject to an export license by the government of Japan.

Further, re-export to another country may be subject to the license of the government of the country from where the product is re-exported. Furthermore, the product may also be controlled by re-export regulations of the United States government.

Should you wish to export or re-export these products, please contact FANUC for advice.

In this manual we have tried as much as possible to describe all the various matters.

However, we cannot describe all the matters which must not be done, or which cannot be done, because there are so many possibilities.

Therefore, matters which are not especially described as possible in this manual should be regarded as "impossible".

Safety

FANUC America Corporation is not and does not represent itself as an expert in safety systems, safety equipment, or the specific safety aspects of your company and/or its work force. It is the responsibility of the owner, employer, or user to take all necessary steps to guarantee the safety of all personnel in the workplace.

The appropriate level of safety for your application and installation can be best determined by safety system professionals. FANUC America Corporation therefore, recommends that each customer consult with such professionals in order to provide a workplace that allows for the safe application, use, and operation of FANUC America Corporation systems.

According to the industry standard ANSI/RIA R15-06, the owner or user is advised to consult the standards to ensure compliance with its requests for Robotics System design, usability, operation, maintenance, and service. Additionally, as the owner, employer, or user of a robotic system, it is your responsibility to arrange for the training of the operator of a robot system to recognize and respond to known hazards associated with your robotic system and to be aware of the recommended operating procedures for your particular application and robot installation.

Ensure that the robot being used is appropriate for the application. Robots used in classified (hazardous) locations must be certified for this use.

FANUC America Corporation therefore, recommends that all personnel who intend to operate, program, repair, or otherwise use the robotics system be trained in an approved FANUC America Corporation training course and become familiar with the proper operation of the system. Persons responsible for programming the system—including the design, implementation, and debugging of application programs—must be familiar with the recommended programming procedures for your application and robot installation.

The following guidelines are provided to emphasize the importance of safety in the workplace.

CONSIDERING SAFETY FOR YOUR ROBOT INSTALLATION

Safety is essential whenever robots are used. Keep in mind the following factors with regard to safety:

- The safety of people and equipment
- Use of safety enhancing devices
- Techniques for safe teaching and manual operation of the robot(s)
- Techniques for safe automatic operation of the robot(s)
- Regular scheduled inspection of the robot and workcell
- Proper maintenance of the robot

Keeping People Safe

The safety of people is always of primary importance in any situation. When applying safety measures to your robotic system, consider the following:

- External devices
- Robot(s)
- Tooling
- Workpiece

Using Safety Enhancing Devices

Always give appropriate attention to the work area that surrounds the robot. The safety of the work area can be enhanced by the installation of some or all of the following devices:

- Safety fences, barriers, or chains
- Light curtains
- Interlocks
- Pressure mats
- Floor markings
- Warning lights
- Mechanical stops
- EMERGENCY STOP buttons
- DEADMAN switches

Setting Up a Safe Workcell

A safe workcell is essential to protect people and equipment. Observe the following guidelines to ensure that the workcell is set up safely. These suggestions are intended to supplement and not replace existing federal, state, and local laws, regulations, and guidelines that pertain to safety.

- Sponsor your personnel for training in approved FANUC America Corporation training course(s) related to your application. Never permit untrained personnel to operate the robots.
- Install a lockout device that uses an access code to prevent unauthorized persons from operating the robot.
- Use anti-tie-down logic to prevent the operator from bypassing safety measures.
- Arrange the workcell so the operator faces the workcell and can see what is going on inside the cell.
- Clearly identify the work envelope of each robot in the system with floor markings, signs, and special barriers. The work envelope is the area defined by the maximum motion range of the robot, including any tooling attached to the wrist flange that extend this range.

Safety

- Position all controllers outside the robot work envelope.
- Never rely on software or firmware based controllers as the primary safety element unless they comply with applicable current robot safety standards.
- Mount an adequate number of EMERGENCY STOP buttons or switches within easy reach of the operator and at critical points inside and around the outside of the workcell.
- Install flashing lights and/or audible warning devices that activate whenever the robot is operating, that is, whenever power is applied to the servo drive system. Audible warning devices shall exceed the ambient noise level at the end-use application.
- Wherever possible, install safety fences to protect against unauthorized entry by personnel into the work envelope.
- Install special guarding that prevents the operator from reaching into restricted areas of the work envelope.
- Use interlocks.
- Use presence or proximity sensing devices such as light curtains, mats, and capacitance and vision systems to enhance safety.
- Periodically check the safety joints or safety clutches that can be optionally installed between the robot wrist flange and tooling. If the tooling strikes an object, these devices dislodge, remove power from the system, and help to minimize damage to the tooling and robot.
- Make sure all external devices are properly filtered, grounded, shielded, and suppressed to prevent hazardous motion due to the effects of electro-magnetic interference (EMI), radio frequency interference (RFI), and electro-static discharge (ESD).
- Make provisions for power lockout/tagout at the controller.
- Eliminate *pinch points*. Pinch points are areas where personnel could get trapped between a moving robot and other equipment.
- Provide enough room inside the workcell to permit personnel to teach the robot and perform maintenance safely.
- Program the robot to load and unload material safely.
- If high voltage electrostatics are present, be sure to provide appropriate interlocks, warning, and beacons.
- If materials are being applied at dangerously high pressure, provide electrical interlocks for lockout of material flow and pressure.

Staying Safe While Teaching or Manually Operating the Robot

Advise all personnel who must teach the robot or otherwise manually operate the robot to observe the following rules:

- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Know whether or not you are using an intrinsically safe teach pendant if you are working in a hazardous environment.

- Before teaching, visually inspect the robot and work envelope to make sure that no potentially hazardous conditions exist. The work envelope is the area defined by the maximum motion range of the robot. These include tooling attached to the wrist flange that extends this range.
- The area near the robot must be clean and free of oil, water, or debris. Immediately report unsafe working conditions to the supervisor or safety department.
- FANUC America Corporation recommends that no one enter the work envelope of a robot that is on, except for robot teaching operations. However, if you must enter the work envelope, be sure all safeguards are in place, check the teach pendant DEADMAN switch for proper operation, and place the robot in teach mode. Take the teach pendant with you, turn it on, and be prepared to release the DEADMAN switch. Only the person with the teach pendant should be in the work envelope.

⚠️WARNING

Never bypass, strap, or otherwise deactivate a safety device, such as a limit switch, for any operational convenience. Deactivating a safety device is known to have resulted in serious injury and death.

- Know the path that can be used to escape from a moving robot; make sure the escape path is never blocked.
- Isolate the robot from all remote control signals that can cause motion while data is being taught.
- Test any program being run for the first time in the following manner:

⚠️WARNING

Stay outside the robot work envelope whenever a program is being run. Failure to do so can result in injury.

- Using a low motion speed, single step the program for at least one full cycle.
 - Using a low motion speed, test run the program continuously for at least one full cycle.
 - Using the programmed speed, test run the program continuously for at least one full cycle.
- Make sure all personnel are outside the work envelope before running production.

Staying Safe During Automatic Operation

Advise all personnel who operate the robot during production to observe the following rules:

- Make sure all safety provisions are present and active.

Safety

- Know the entire workcell area. The workcell includes the robot and its work envelope, plus the area occupied by all external devices and other equipment with which the robot interacts.
- Understand the complete task the robot is programmed to perform before initiating automatic operation.
- Make sure all personnel are outside the work envelope before operating the robot.
- Never enter or allow others to enter the work envelope during automatic operation of the robot.
- Know the location and status of all switches, sensors, and control signals that could cause the robot to move.
- Know where the EMERGENCY STOP buttons are located on both the robot control and external control devices. Be prepared to press these buttons in an emergency.
- Never assume that a program is complete if the robot is not moving. The robot could be waiting for an input signal that will permit it to continue its activity.
- If the robot is running in a pattern, do not assume it will continue to run in the same pattern.
- Never try to stop the robot, or break its motion, with your body. The only way to stop robot motion immediately is to press an EMERGENCY STOP button located on the controller panel, teach pendant, or emergency stop stations around the workcell.

Staying Safe During Inspection

When inspecting the robot, be sure to

- Turn off power at the controller.
- Lock out and tag out the power source at the controller according to the policies of your plant.
- Turn off the compressed air source and relieve the air pressure.
- If robot motion is not needed for inspecting the electrical circuits, press the EMERGENCY STOP button on the operator panel.
- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- If power is needed to check the robot motion or electrical circuits, be prepared to press the EMERGENCY STOP button, in an emergency.
- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.

Staying Safe During Maintenance

When performing maintenance on your robot system, observe the following rules:

- Never enter the work envelope while the robot or a program is in operation.
- Before entering the work envelope, visually inspect the workcell to make sure no potentially hazardous conditions exist.

- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Consider all or any overlapping work envelopes of adjoining robots when standing in a work envelope.
- Test the teach pendant for proper operation before entering the work envelope.
- If it is necessary for you to enter the robot work envelope while power is turned on, you must be sure that you are in control of the robot. Be sure to take the teach pendant with you, press the DEADMAN switch, and turn the teach pendant on. Be prepared to release the DEADMAN switch to turn off servo power to the robot immediately.
- Whenever possible, perform maintenance with the power turned off. Before you open the controller front panel or enter the work envelope, turn off and lock out the 3-phase power source at the controller.
- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.

AWARNING

Lethal voltage is present in the controller WHENEVER IT IS CONNECTED to a power source. Be extremely careful to avoid electrical shock. HIGH VOLTAGE IS PRESENT at the input side whenever the controller is connected to a power source. Turning the disconnect or circuit breaker to the OFF position removes power from the output side of the device only.

- Release or block all stored energy. Before working on the pneumatic system, shut off the system air supply and purge the air lines.
- Isolate the robot from all remote control signals. If maintenance must be done when the power is on, make sure the person inside the work envelope has sole control of the robot. The teach pendant must be held by this person.
- Make sure personnel cannot get trapped between the moving robot and other equipment. Know the path that can be used to escape from a moving robot. Make sure the escape route is never blocked.
- Use blocks, mechanical stops, and pins to prevent hazardous movement by the robot. Make sure that such devices do not create pinch points that could trap personnel.

AWARNING

Do not try to remove any mechanical component from the robot before thoroughly reading and understanding the procedures in the appropriate manual. Doing so can result in serious personal injury and component destruction.

- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.
- When replacing or installing components, make sure dirt and debris do not enter the system.
- Use only specified parts for replacement. To avoid fires and damage to parts in the controller, never use nonspecified fuses.
- Before restarting a robot, make sure no one is inside the work envelope; be sure that the robot and all external devices are operating normally.

KEEPING MACHINE TOOLS AND EXTERNAL DEVICES SAFE

Certain programming and mechanical measures are useful in keeping the machine tools and other external devices safe. Some of these measures are outlined below. Make sure you know all associated measures for safe use of such devices.

Programming Safety Precautions

Implement the following programming safety measures to prevent damage to machine tools and other external devices.

- Back-check limit switches in the workcell to make sure they do not fail.
- Implement “failure routines” in programs that will provide appropriate robot actions if an external device or another robot in the workcell fails.
- Use *handshaking* protocol to synchronize robot and external device operations.
- Program the robot to check the condition of all external devices during an operating cycle.

Mechanical Safety Precautions

Implement the following mechanical safety measures to prevent damage to machine tools and other external devices.

- Make sure the workcell is clean and free of oil, water, and debris.
- Use DCS (Dual Check Safety), software limits, limit switches, and mechanical hardstops to prevent undesired movement of the robot into the work area of machine tools and external devices.

KEEPING THE ROBOT SAFE

Observe the following operating and programming guidelines to prevent damage to the robot.

Operating Safety Precautions

The following measures are designed to prevent damage to the robot during operation.

- Use a low override speed to increase your control over the robot when jogging the robot.
- Visualize the movement the robot will make before you press the jog keys on the teach pendant.
- Make sure the work envelope is clean and free of oil, water, or debris.
- Use circuit breakers to guard against electrical overload.

Programming Safety Precautions

The following safety measures are designed to prevent damage to the robot during programming:

- Establish *interference zones* to prevent collisions when two or more robots share a work area.
- Make sure that the program ends with the robot near or at the home position.
- Be aware of signals or other operations that could trigger operation of tooling resulting in personal injury or equipment damage.
- In dispensing applications, be aware of all safety guidelines with respect to the dispensing materials.

NOTE: Any deviation from the methods and safety practices described in this manual must conform to the approved standards of your company. If you have questions, see your supervisor.

ADDITIONAL SAFETY CONSIDERATIONS FOR PAINT ROBOT INSTALLATIONS

Process technicians are sometimes required to enter the paint booth, for example, during daily or routine calibration or while teaching new paths to a robot. Maintenance personnel also must work inside the paint booth periodically.

Whenever personnel are working inside the paint booth, ventilation equipment must be used. Instruction on the proper use of ventilating equipment usually is provided by the paint shop supervisor.

Although paint booth hazards have been minimized, potential dangers still exist. Therefore, today's highly automated paint booth requires that process and maintenance personnel have full awareness of the system and its capabilities. They must understand the interaction that occurs between the vehicle moving along the conveyor and the robot(s), hood/deck and door opening devices, and high-voltage electrostatic tools.



CAUTION

Ensure that all ground cables remain connected. Never operate the paint robot with ground provisions disconnected. Otherwise, you could injure personnel or damage equipment.

Paint robots are operated in three modes:

- Teach or manual mode
- Automatic mode, including automatic and exercise operation
- Diagnostic mode

During both teach and automatic modes, the robots in the paint booth will follow a predetermined pattern of movements. In teach mode, the process technician teaches (programs) paint paths using the teach pendant.

In automatic mode, robot operation is initiated at the System Operator Console (SOC) or Manual Control Panel (MCP), if available, and can be monitored from outside the paint booth. All personnel must remain outside of the booth or in a designated safe area within the booth whenever automatic mode is initiated at the SOC or MCP.

In automatic mode, the robots will execute the path movements they were taught during teach mode, but generally at production speeds.

When process and maintenance personnel run diagnostic routines that require them to remain in the paint booth, they must stay in a designated safe area.

Paint System Safety Features

Process technicians and maintenance personnel must become totally familiar with the equipment and its capabilities. To minimize the risk of injury when working near robots and related equipment, personnel must comply strictly with the procedures in the manuals.

This section provides information about the safety features that are included in the paint system and also explains the way the robot interacts with other equipment in the system.

The paint system includes the following safety features:

- Most paint booths have red warning beacons that illuminate when the robots are armed and ready to paint. Your booth might have other kinds of indicators. Learn what these are.

- Some paint booths have a blue beacon that, when illuminated, indicates that the electrostatic devices are enabled. Your booth might have other kinds of indicators. Learn what these are.
- EMERGENCY STOP buttons are located on the robot controller and teach pendant. Become familiar with the locations of all E-STOP buttons.
- An intrinsically safe teach pendant is used when teaching in hazardous paint atmospheres.
- A DEADMAN switch is located on each teach pendant. When this switch is held in, and the teach pendant is on, power is applied to the robot servo system. If the engaged DEADMAN switch is released or pressed harder during robot operation, power is removed from the servo system, all axis brakes are applied, and the robot comes to an EMERGENCY STOP. Safety interlocks within the system might also E-STOP other robots.

WARNING

An EMERGENCY STOP will occur if the DEADMAN switch is released on a bypassed robot.

- Overtravel by robot axes is prevented by software limits. All of the major and minor axes are governed by software limits. DCS (Dual Check Safety), limit switches and hardstops also limit travel by the major axes.
- EMERGENCY STOP limit switches and photoelectric eyes might be part of your system. Limit switches, located on the entrance/exit doors of each booth, will EMERGENCY STOP all equipment in the booth if a door is opened while the system is operating in automatic or manual mode. For some systems, signals to these switches are inactive when the switch on the SOC is in teach mode.
- When present, photoelectric eyes are sometimes used to monitor unauthorized intrusion through the entrance/exit silhouette openings.
- System status is monitored by computer. Severe conditions result in automatic system shutdown.

Staying Safe While Operating the Paint Robot

When you work in or near the paint booth, observe the following rules, in addition to all rules for safe operation that apply to all robot systems.

WARNING

Observe all safety rules and guidelines to avoid injury.

⚠️ WARNING

Never bypass, strap, or otherwise deactivate a safety device, such as a limit switch, for any operational convenience. Deactivating a safety device is known to have resulted in serious injury and death.

⚠️ WARNING

Enclosures shall not be opened unless the area is known to be nonhazardous or all power has been removed from devices within the enclosure. Power shall not be restored after the enclosure has been opened until all combustible dusts have been removed from the interior of the enclosure and the enclosure purged. Refer to the Purge chapter for the required purge time.

- Know the work area of the entire paint station (workcell).
- Know the work envelope of the robot and hood/deck and door opening devices.
- Be aware of overlapping work envelopes of adjacent robots.
- Know where all red, mushroom-shaped EMERGENCY STOP buttons are located.
- Know the location and status of all switches, sensors, and/or control signals that might cause the robot, conveyor, and opening devices to move.
- Make sure that the work area near the robot is clean and free of water, oil, and debris. Report unsafe conditions to your supervisor.
- Become familiar with the complete task the robot will perform BEFORE starting automatic mode.
- Make sure all personnel are outside the paint booth before you turn on power to the robot servo system.
- Never enter the work envelope or paint booth before you turn off power to the robot servo system.
- Never enter the work envelope during automatic operation unless a safe area has been designated.
- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Remove all metallic objects, such as rings, watches, and belts, before entering a booth when the electrostatic devices are enabled.
- Stay out of areas where you might get trapped between a moving robot, conveyor, or opening device and another object.
- Be aware of signals and/or operations that could result in the triggering of guns or bells.
- Be aware of all safety precautions when dispensing of paint is required.
- Follow the procedures described in this manual.

Special Precautions for Combustible Dusts (Powder Paint)

When the robot is used in a location where combustible dusts are found, such as the application of powder paint, the following special precautions are required to insure that there are no combustible dusts inside the robot.

- Purge maintenance air should be maintained at all times, even when the robot power is off. This will insure that dust can not enter the robot.
 - A purge cycle will not remove accumulated dusts. Therefore, if the robot is exposed to dust when maintenance air is not present, it will be necessary to remove the covers and clean out any accumulated dust. Do not energize the robot until you have performed the following steps.
1. Before covers are removed, the exterior of the robot should be cleaned to remove accumulated dust.
 2. When cleaning and removing accumulated dust, either on the outside or inside of the robot, be sure to use methods appropriate for the type of dust that exists. Usually lint free rags dampened with water are acceptable. Do not use a vacuum cleaner to remove dust as it can generate static electricity and cause an explosion unless special precautions are taken.
 3. Thoroughly clean the interior of the robot with a lint free rag to remove any accumulated dust.
 4. When the dust has been removed, the covers must be replaced immediately.
 5. Immediately after the covers are replaced, run a complete purge cycle. The robot can now be energized.

Staying Safe While Operating Paint Application Equipment

When you work with paint application equipment, observe the following rules, in addition to all rules for safe operation that apply to all robot systems.

WARNING

When working with electrostatic paint equipment, follow all national and local codes as well as all safety guidelines within your organization. Also reference the following standards: NFPA 33 Standards for Spray Application Using Flammable or Combustible Materials, and NFPA 70 National Electrical Code.

- **Grounding:** All electrically conductive objects in the spray area must be grounded. This includes the spray booth, robots, conveyors, workstations, part carriers, hooks, paint pressure pots, as well as solvent containers. Grounding is defined as the object or objects shall be electrically connected to ground with a resistance of not more than 1 megohms.
- **High Voltage:** High voltage should only be on during actual spray operations. Voltage should be off when the painting process is completed. Never leave high voltage on during a cap cleaning process.
- Avoid any accumulation of combustible vapors or coating matter.
- Follow all manufacturer recommended cleaning procedures.
- Make sure all interlocks are operational.

- No smoking.
- Post all warning signs regarding the electrostatic equipment and operation of electrostatic equipment according to NFPA 33 Standard for Spray Application Using Flammable or Combustible Material.
- Disable all air and paint pressure to bell.
- Verify that the lines are not under pressure.

Staying Safe During Maintenance

When you perform maintenance on the painter system, observe the following rules, and all other maintenance safety rules that apply to all robot installations. Only qualified, trained service or maintenance personnel should perform repair work on a robot.

- Paint robots operate in a potentially explosive environment. Use caution when working with electric tools.
- When a maintenance technician is repairing or adjusting a robot, the work area is under the control of that technician. All personnel not participating in the maintenance must stay out of the area.
- For some maintenance procedures, station a second person at the control panel within reach of the EMERGENCY STOP button. This person must understand the robot and associated potential hazards.
- Be sure all covers and inspection plates are in good repair and in place.
- Always return the robot to the “home” position before you disarm it.
- Never use machine power to aid in removing any component from the robot.
- During robot operations, be aware of the robot’s movements. Excess vibration, unusual sounds, and so forth, can alert you to potential problems.
- Whenever possible, turn off the main electrical disconnect before you clean the robot.
- When using vinyl resin observe the following:
 - Wear eye protection and protective gloves during application and removal.
 - Adequate ventilation is required. Overexposure could cause drowsiness or skin and eye irritation.
 - If there is contact with the skin, wash with water.
 - Follow the Original Equipment Manufacturer’s Material Safety Data Sheets.
- When using paint remover observe the following:
 - Eye protection, protective rubber gloves, boots, and apron are required during booth cleaning.
 - Adequate ventilation is required. Overexposure could cause drowsiness.
 - If there is contact with the skin or eyes, rinse with water for at least 15 minutes. Then seek medical attention as soon as possible.
 - Follow the Original Equipment Manufacturer’s Material Safety Data Sheets.

SAFETY PRECAUTIONS

Thank you for purchasing FANUC Robot.

This chapter describes the precautions which must be observed to ensure the safe use of the robot.

Before attempting to use the robot, be sure to read this chapter thoroughly.

Before using the functions related to robot operation, read the relevant operator's manual to become familiar with those functions.

If any description in this chapter differs from that in the other part of this manual, the description given in this chapter shall take precedence.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral devices installed in a work cell.

In addition, refer to the "FANUC Robot SAFETY HANDBOOK (B-80687EN)".

1 WORKING PERSON

The personnel can be classified as follows.

Operator:

- Turns robot controller power ON/OFF
- Starts robot program from operator's panel

Programmer or teaching operator:

- Operates the robot
- Teaches robot inside the safety fence

Maintenance engineer:

- Operates the robot
- Teaches robot inside the safety fence
- Maintenance (adjustment, replacement)

- An operator cannot work inside the safety fence.
- A programmer, teaching operator, and maintenance engineer can work inside the safety fence. The working activities inside the safety fence include lifting, setting, teaching, adjusting, maintenance, etc.
- To work inside the fence, the person must be trained on proper robot operation.

During the operation, programming, and maintenance of your robotic system, the programmer, teaching operator, and maintenance engineer should take additional care of their safety by using the following safety precautions.

- Use adequate clothing or uniforms during system operation
- Wear safety shoes
- Use helmet

2 DEFINITION OF WARNING, CAUTION AND NOTE

To ensure the safety of users and prevent damage to the machine, this manual indicates each precaution on safety with "Warning" or "Caution" according to its severity. Supplementary information is indicated by "Note". Read the contents of each "Warning", "Caution" and "Note" before attempting to use the robots.

⚠ WARNING

Applied when there is a danger of the user being injured or when there is a danger of both the user being injured and the equipment being damaged if the approved procedure is not observed.

⚠ CAUTION

Applied when there is a danger of the equipment being damaged, if the approved procedure is not observed.

NOTE

Notes are used to indicate supplementary information other than Warnings and Cautions.

- Read this manual carefully, and store it in a sales place.

3 WORKING PERSON SAFETY

Working person safety is the primary safety consideration. Because it is very dangerous to enter the operating space of the robot during automatic operation, adequate safety precautions must be observed.

The following lists the general safety precautions. Careful consideration must be made to ensure working person safety.

- (1) Have the robot system working persons attend the training courses held by FANUC.

FANUC provides various training courses. Contact our sales office for details.

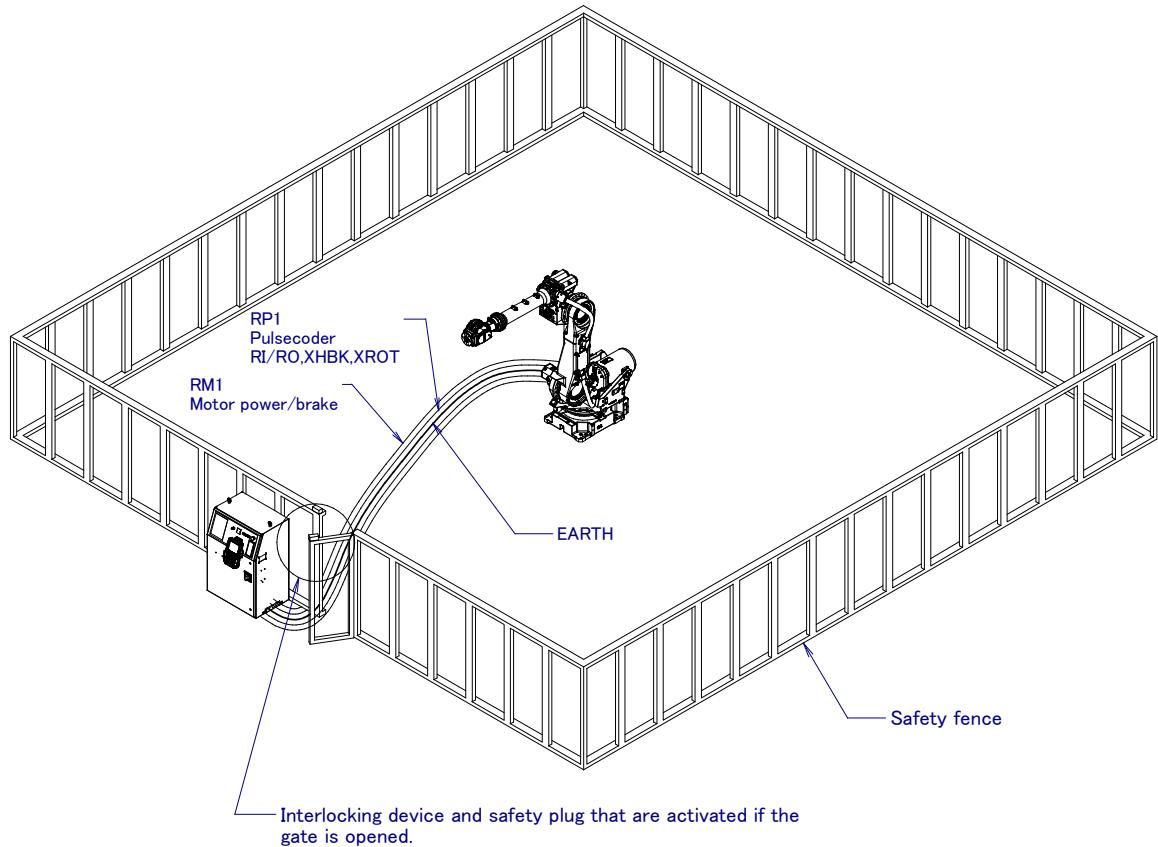
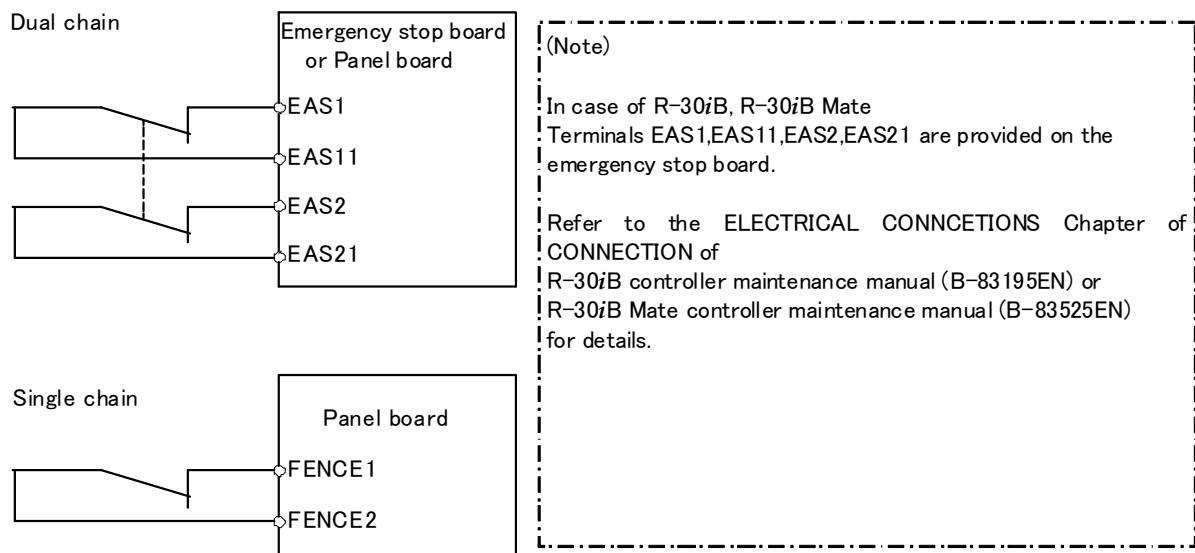
- (2) Even when the robot is stationary, it is possible that the robot is still in a ready to move state, and is waiting for a signal. In this state, the robot is regarded as still in motion. To ensure working person safety, provide the system with an alarm to indicate visually or aurally that the robot is in motion.
- (3) Install a safety fence with a gate so that no working person can enter the work area without passing through the gate. Install an interlocking device, a safety plug, and so forth in the safety gate so that the robot is stopped as the safety gate is opened.

The controller is designed to receive this interlocking signal of the door switch. When the gate is opened and this signal received, the controller stops the robot (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type). For connection, see Fig.3 (a) and Fig.3 (b).

- (4) Provide the peripheral devices with appropriate grounding (Class A, Class B, Class C, and Class D).
- (5) Try to install the peripheral devices outside the work area.
- (6) Draw an outline on the floor, clearly indicating the range of the robot motion, including the tools such as a hand.
- (7) Install a mat switch or photoelectric switch on the floor with an interlock to a visual or aural alarm that stops the robot when a working person enters the work area.
- (8) If necessary, install a safety lock so that no one except the working person in charge can turn on the power of the robot.

The circuit breaker installed in the controller is designed to disable anyone from turning it on when it is locked with a padlock.

- (9) When adjusting each peripheral device independently, be sure to turn off the power of the robot
- (10) Operators should be ungloved while manipulating the operator's panel or teach pendant. Operation with gloved fingers could cause an operation error.
- (11) Programs, system variables, and other information can be saved on memory card or USB memories. Be sure to save the data periodically in case the data is lost in an accident.
- (12) The robot should be transported and installed by accurately following the procedures recommended by FANUC. Wrong transportation or installation may cause the robot to fall, resulting in severe injury to workers.
- (13) In the first operation of the robot after installation, the operation should be restricted to low speeds. Then, the speed should be gradually increased to check the operation of the robot.
- (14) Before the robot is started, it should be checked that no one is in the area of the safety fence. At the same time, a check must be made to ensure that there is no risk of hazardous situations. If detected, such a situation should be eliminated before the operation.
- (15) When the robot is used, the following precautions should be taken. Otherwise, the robot and peripheral equipment can be adversely affected, or workers can be severely injured.
 - Avoid using the robot in a flammable environment.
 - Avoid using the robot in an explosive environment.
 - Avoid using the robot in an environment full of radiation.
 - Avoid using the robot under water or at high humidity.
 - Avoid using the robot to carry a person or animal.
 - Avoid using the robot as a stepladder. (Never climb up on or hang from the robot.)
- (16) When connecting the peripheral devices related to stop(safety fence etc.) and each signal (external emergency , fence etc.) of robot. be sure to confirm the stop movement and do not take the wrong connection.
- (17) When preparing trestle, please consider security for installation and maintenance work in high place according to Fig.3 (c). Please consider footstep and safety bolt mounting position.

**Fig. 3 (a) Safety fence and safety gate****Fig. 3 (b) Limit switch circuit diagram of the safety fence**

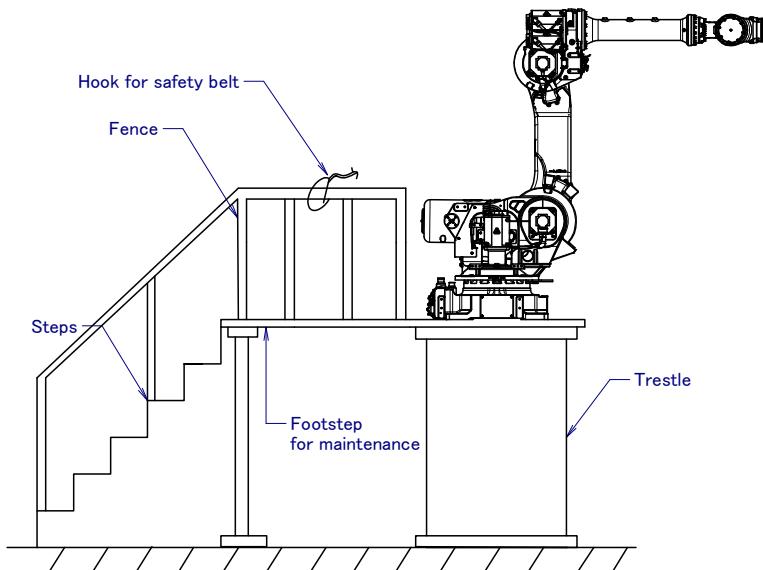


Fig.3 (c) Footstep for maintenance

3.1 OPERATOR SAFETY

The operator is a person who operates the robot system. In this sense, a worker who operates the teach pendant is also an operator. However, this section does not apply to teach pendant operators.

- (1) If you do not have to operate the robot, turn off the power of the robot controller or press the EMERGENCY STOP button, and then proceed with necessary work.
- (2) Operate the robot system at a location outside of the safety fence
- (3) Install a safety fence with a safety gate to prevent any worker other than the operator from entering the work area unexpectedly and to prevent the worker from entering a dangerous area.
- (4) Install an EMERGENCY STOP button within the operator's reach.

The robot controller is designed to be connected to an external EMERGENCY STOP button. With this connection, the controller stops the robot operation (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type), when the external EMERGENCY STOP button is pressed. See the diagram below for connection.

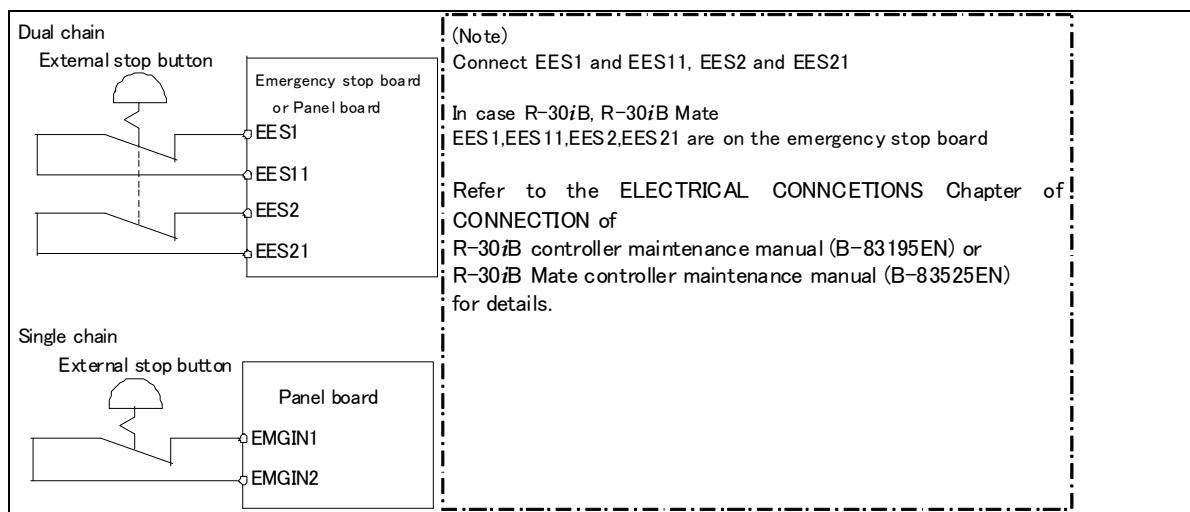


Fig.3.1 Connection diagram for external emergency stop button

3.2 SAFETY OF THE PROGRAMMER

While teaching the robot, the operator must enter the work area of the robot. The operator must ensure the safety of the teach pendant operator especially.

- (1) Unless it is specifically necessary to enter the robot work area, carry out all tasks outside the area.
- (2) Before teaching the robot, check that the robot and its peripheral devices are all in the normal operating condition.
- (3) If it is inevitable to enter the robot work area to teach the robot, check the locations, settings, and other conditions of the safety devices (such as the EMERGENCY STOP button, the DEADMAN switch on the teach pendant) before entering the area.
- (4) The programmer must be extremely careful not to let anyone else enter the robot work area.
- (5) Programming should be done outside the area of the safety fence as far as possible. If programming needs to be done in the area of the safety fence, the programmer should take the following precautions:
 - Before entering the area of the safety fence, ensure that there is no risk of dangerous situations in the area.
 - Be prepared to press the emergency stop button whenever necessary.
 - Robot motions should be made at low speeds.
 - Before starting programming, check the entire system status to ensure that no remote instruction to the peripheral equipment or motion would be dangerous to the user.

Our operator panel is provided with an emergency stop button and a key switch (mode switch) for selecting the automatic operation mode (AUTO) and the teach modes (T1 and T2). Before entering the inside of the safety fence for the purpose of teaching, set the switch to a teach mode, remove the key from the mode switch to prevent other people from changing the operation mode carelessly, then open the safety gate. If the safety gate is opened with the automatic operation mode set, the robot stops (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type). After the switch is set to a teach mode, the safety gate is disabled. The programmer should understand that the safety gate is disabled and is responsible for keeping other people from entering the inside of the safety fence.

Our teach pendant is provided with a DEADMAN switch as well as an emergency stop button. These button and switch function as follows:

- (1) Emergency stop button: Causes an emergency stop (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type) when pressed.
 - (2) DEADMAN switch: Functions differently depending on the teach pendant enable/disable switch setting status.
 - (a) Disable: The DEADMAN switch is disabled.
 - (b) Enable: Servo power is turned off when the operator releases the DEADMAN switch or when the operator presses the switch strongly.
- Note) The DEADMAN switch is provided to stop the robot when the operator releases the teach pendant or presses the pendant strongly in case of emergency. The R-30iB/R-30iB Mate employs a 3-position DEADMAN switch, which allows the robot to operate when the 3-position DEADMAN switch is pressed to its intermediate point. When the operator releases the DEADMAN switch or presses the switch strongly, the robot stops immediately.

The operator's intention of starting teaching is determined by the controller through the dual operation of setting the teach pendant enable/disable switch to the enable position and pressing the DEADMAN switch. The operator should make sure that the robot could operate in such conditions and be responsible in carrying out tasks safely.

Based on the risk assessment by FANUC, number of operation of DEADMAN SW should not exceed about 10000 times per year.

The teach pendant, operator panel, and peripheral device interface send each robot start signal. However the validity of each signal changes as follows depending on the mode switch and the DEADMAN switch of the operator panel, the teach pendant enable switch and the remote condition on the software.

In case of R-30iB Controller

Mode	Teach pendant enable switch	Software remote condition	Teach pendant	Operator panel	Peripheral device
AUTO mode	On	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed
	Off	Local	Not allowed	Allowed to start	Not allowed
		Remote	Not allowed	Not allowed	Allowed to start
T1, T2 mode	On	Local	Allowed to start	Not allowed	Not allowed
		Remote	Allowed to start	Not allowed	Not allowed
	Off	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed

T1,T2 mode:DEADMAN switch is effective.

- (6) To start the system using the operator's panel, make certain that nobody is the robot work area and that there are no abnormal conditions in the robot work area.
- (7) When a program is completed, be sure to carry out a test operation according to the procedure below.
 - (a) Run the program for at least one operation cycle in the single step mode at low speed.
 - (b) Run the program for at least one operation cycle in the continuous operation mode at low speed.
 - (c) Run the program for one operation cycle in the continuous operation mode at the intermediate speed and check that no abnormalities occur due to a delay in timing.
 - (d) Run the program for one operation cycle in the continuous operation mode at the normal operating speed and check that the system operates automatically without trouble.
 - (e) After checking the completeness of the program through the test operation above, execute it in the automatic operation mode.
- (8) While operating the system in the automatic operation mode, the teach pendant operator should leave the robot work area.

3.3 SAFETY OF THE MAINTENANCE ENGINEER

For the safety of maintenance engineer personnel, pay utmost attention to the following.

- (1) During operation, never enter the robot work area.
- (2) A hazardous situation may arise when the robot or the system, are kept with their power-on during maintenance operations. Therefore, for any maintenance operation, the robot and the system should be put into the power-off state. If necessary, a lock should be in place in order to prevent any other person from turning on the robot and/or the system. In case maintenance needs to be executed in the power-on state, the emergency stop button must be pressed.
- (3) If it becomes necessary to enter the robot operation range while the power is on, press the emergency stop button on the operator panel, or the teach pendant before entering the range. The maintenance personnel must indicate that maintenance work is in progress and be careful not to allow other people to operate the robot carelessly.
- (4) When entering the area enclosed by the safety fence, the maintenance worker must check the entire system in order to make sure no dangerous situations exist. In case the worker needs to enter the safety area whilst a dangerous situation exists, extreme care must be taken, and entire system status must be carefully monitored.
- (5) Before the maintenance of the pneumatic system is started, the supply pressure should be shut off and the pressure in the piping should be reduced to zero.

- (6) Before the start of teaching, check that the robot and its peripheral devices are all in the normal operating condition.
- (7) Do not operate the robot in the automatic mode while anybody is in the robot work area.
- (8) When you maintain the robot alongside a wall or instrument, or when multiple workers are working nearby, make certain that their escape path is not obstructed.
- (9) When a tool is mounted on the robot, or when any moving device other than the robot is installed, such as belt conveyor, pay careful attention to its motion.
- (10) If necessary, have a worker who is familiar with the robot system stand beside the operator panel and observe the work being performed. If any danger arises, the worker should be ready to press the EMERGENCY STOP button at any time.
- (11) When replacing a part, please contact FANUC service center. If a wrong procedure is followed, an accident may occur, causing damage to the robot and injury to the worker.
- (12) When replacing or reinstalling components, take care to prevent foreign material from entering the system.
- (13) When handling each unit or printed circuit board in the controller during inspection, turn off the circuit breaker to protect against electric shock.
If there are two cabinets, turn off the both circuit breaker.
- (14) A part should be replaced with a part recommended by FANUC. If other parts are used, malfunction or damage would occur. Especially, a fuse that is not recommended by FANUC should not be used. Such a fuse may cause a fire.
- (15) When restarting the robot system after completing maintenance work, make sure in advance that there is no person in the work area and that the robot and the peripheral devices are not abnormal.
- (16) When a motor or brake is removed, the robot arm should be supported with a crane or other equipment beforehand so that the arm would not fall during the removal.
- (17) Whenever grease is spilled on the floor, it should be removed as quickly as possible to prevent dangerous falls.
- (18) The following parts are heated. If a maintenance worker needs to touch such a part in the heated state, the worker should wear heat-resistant gloves or use other protective tools.
 - Servo motor
 - Inside the controller
 - Reducer
 - Gearbox
 - Wrist unit
- (19) Maintenance should be done under suitable light. Care must be taken that the light would not cause any danger.
- (20) When a motor, reducer, or other heavy load is handled, a crane or other equipment should be used to protect maintenance workers from excessive load. Otherwise, the maintenance workers would be severely injured.
- (21) The robot should not be stepped on or climbed up during maintenance. If it is attempted, the robot would be adversely affected. In addition, a misstep can cause injury to the worker.
- (22) When performing maintenance work in high place, secure a footstep and wear safety belt.
- (23) After the maintenance is completed, spilled oil or water and metal chips should be removed from the floor around the robot and within the safety fence.
- (24) When a part is replaced, all bolts and other related components should put back into their original places. A careful check must be given to ensure that no components are missing or left not mounted.
- (25) In case robot motion is required during maintenance, the following precautions should be taken :
 - Foresee an escape route. And during the maintenance motion itself, monitor continuously the whole system so that your escape route will not become blocked by the robot, or by peripheral equipment.
 - Always pay attention to potentially dangerous situations, and be prepared to press the emergency stop button whenever necessary.
- (26) The robot should be periodically inspected. (Refer to the robot mechanical manual and controller maintenance manual.) A failure to do the periodical inspection can adversely affect the performance or service life of the robot and may cause an accident

- (27) After a part is replaced, a test operation should be given for the robot according to a predetermined method. (See TESTING section of "R-30iB/R-30iB Mate Controller operator's manual (Basic Operation)".) During the test operation, the maintenance staff should work outside the safety fence.

4 SAFETY OF THE TOOLS AND PERIPHERAL DEVICES

4.1 PRECAUTIONS IN PROGRAMMING

- (1) Use a limit switch or other sensor to detect a dangerous condition and, if necessary, design the program to stop the robot when the sensor signal is received.
- (2) Design the program to stop the robot when an abnormal condition occurs in any other robots or peripheral devices, even though the robot itself is normal.
- (3) For a system in which the robot and its peripheral devices are in synchronous motion, particular care must be taken in programming so that they do not interfere with each other.
- (4) Provide a suitable interface between the robot and its peripheral devices so that the robot can detect the states of all devices in the system and can be stopped according to the states.

4.2 PRECAUTIONS FOR MECHANISM

- (1) Keep the component cells of the robot system clean, and operate the robot in an environment free of grease, water, and dust.
- (2) Don't use unconfirmed liquid for cutting fluid and cleaning fluid.
- (3) Employ a limit switch or mechanical stopper to limit the robot motion so that the robot or cable does not strike against its peripheral devices or tools.
- (4) Observe the following precautions about the mechanical unit cables. When these attentions are not kept, unexpected troubles might occur.
 - Use mechanical unit cable that have required user interface.
 - Don't add user cable or hose to inside of mechanical unit.
 - Please do not obstruct the movement of the mechanical unit cable when cables are added to outside of mechanical unit.
 - In the case of the model that a cable is exposed, Please do not perform remodeling (Adding a protective cover and fix an outside cable more) obstructing the behavior of the outcrop of the cable.
 - Please do not interfere with the other parts of mechanical unit when install equipments in the robot.
- (5) The frequent power-off stop for the robot during operation causes the trouble of the robot. Please avoid the system construction that power-off stop would be operated routinely. (Refer to bad case example.) Please execute power-off stop after reducing the speed of the robot and stopping it by hold stop or cycle stop when it is not urgent. (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type.)
(Bad case example)
 - Whenever poor product is generated, a line stops by emergency stop.
 - When alteration was necessary, safety switch is operated by opening safety fence and power-off stop is executed for the robot during operation.
 - An operator pushes the emergency stop button frequently, and a line stops.
 - An area sensor or a mat switch connected to safety signal operate routinely and power-off stop is executed for the robot.
- (6) Robot stops urgently when collision detection alarm (SRVO-050) etc. occurs. The frequent urgent stop by alarm causes the trouble of the robot, too. So remove the causes of the alarm.

5 SAFETY OF THE ROBOT MECHANISM

5.1 PRECAUTIONS IN OPERATION

- (1) When operating the robot in the jog mode, set it at an appropriate speed so that the operator can manage the robot in any eventuality.
- (2) Before pressing the jog key, be sure you know in advance what motion the robot will perform in the jog mode.

5.2 PRECAUTIONS IN PROGRAMMING

- (1) When the work areas of robots overlap, make certain that the motions of the robots do not interfere with each other.
- (2) Be sure to specify the predetermined work origin in a motion program for the robot and program the motion so that it starts from the origin and terminates at the origin.
Make it possible for the operator to easily distinguish at a glance that the robot motion has terminated.

5.3 PRECAUTIONS FOR MECHANISMS

- (1) Keep the work areas of the robot clean, and operate the robot in an environment free of grease, water, and dust.

5.4 PROCEDURE TO MOVE ARM WITHOUT DRIVE POWER IN EMERGENCY OR ABNORMAL SITUATIONS

For emergency or abnormal situations (e.g. persons trapped in or by the robot), brake release unit can be used to move the robot axes without drive power.

Please refer to controller maintenance manual and mechanical unit operator's manual for using method of brake release unit and method of supporting robot.

6 SAFETY OF THE END EFFECTOR

6.1 PRECAUTIONS IN PROGRAMMING

- (1) To control the pneumatic, hydraulic and electric actuators, carefully consider the necessary time delay after issuing each control command up to actual motion and ensure safe control.
- (2) Provide the end effector with a limit switch, and control the robot system by monitoring the state of the end effector.

7

STOP TYPE OF ROBOT

The following three robot stop types exist:

Power-Off Stop (Category 0 following IEC 60204-1)

Servo power is turned off and the robot stops immediately. Servo power is turned off when the robot is moving, and the motion path of the deceleration is uncontrolled.

The following processing is performed at Power-Off stop.

- An alarm is generated and servo power is turned off.
- The robot operation is stopped immediately. Execution of the program is paused.

Controlled stop (Category 1 following IEC 60204-1)

The robot is decelerated until it stops, and servo power is turned off.

The following processing is performed at Controlled stop.

- The alarm "SRVO-199 Controlled stop" occurs along with a decelerated stop. Execution of the program is paused.
- An alarm is generated and servo power is turned off.

Hold (Category 2 following IEC 60204-1)

The robot is decelerated until it stops, and servo power remains on.

The following processing is performed at Hold.

- The robot operation is decelerated until it stops. Execution of the program is paused.

WARNING

The stopping distance and stopping time of Controlled stop are longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the increased stopping distance and stopping time, is necessary when Controlled stop is used.

When the emergency stop button is pressed or the FENCE is open, the stop type of robot is Power-Off stop or Controlled stop. The configuration of stop type for each situation is called *stop pattern*. The stop pattern is different according to the controller type or option configuration.

There are the following 3 Stop patterns.

Stop pattern	Mode	Emergency stop button	External Emergency stop	FENCE open	SVOFF input	Servo disconnect
A	AUTO	P-Stop	P-Stop	C-Stop	C-Stop	P-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop
B	AUTO	P-Stop	P-Stop	P-Stop	P-Stop	P-Stop
	T1	P-Stop	P-Stop	-	P-Stop	P-Stop
	T2	P-Stop	P-Stop	-	P-Stop	P-Stop
C	AUTO	C-Stop	C-Stop	C-Stop	C-Stop	C-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop

P-Stop: Power-Off stop

C-Stop: Controlled stop

-: Disable

The following table indicates the Stop pattern according to the controller type or option configuration.

Option	R-30iB/ R-30iB Mate
Standard	A (*)
Controlled stop by E-Stop (A05B-2600-J570)	C (*)

(*) R-30iB / R-30iB Mate does not have servo disconnect. / R-30iB Mate does not have SVOFF input.

The stop pattern of the controller is displayed in "Stop pattern" line in software version screen. Please refer to "Software version" in operator's manual of controller for the detail of software version screen.

"Controlled stop by E-Stop" option

When "Controlled stop by E-Stop" (A05B-2600-J570) option is specified, the stop type of the following alarms becomes

Controlled stop but only in AUTO mode. In T1 or T2 mode, the stop type is Power-Off stop which is the normal operation of the system.

Alarm	Condition
SRVO-001 Operator panel E-stop	Operator panel emergency stop is pressed.
SRVO-002 Teach pendant E-stop	Teach pendant emergency stop is pressed.
SRVO-007 External emergency stops	External emergency stop input (EES1-EES11, EES2-EES21) is open.
SRVO-408 DCS SSO Ext Emergency Stop	In DCS Safe I/O connect function, SSO[3] is OFF.
SRVO-409 DCS SSO Servo Disconnect	In DCS Safe I/O connect function, SSO[4] is OFF.

Controlled stop is different from Power-Off stop as follows:

- In Controlled stop, the robot is stopped on the program path. This function is effective for a system where the robot can interfere with other devices if it deviates from the program path.
- In Controlled stop, physical impact is less than Power-Off stop. This function is effective for systems where the physical impact to the mechanical unit or EOAT (End Of Arm Tool) should be minimized.
- The stopping distance and stopping time of Controlled stop is longer than the stopping distance and stopping time of Power-Off stop, depending on the robot model and axis. Please refer to the operator's manual of a particular robot model for the data of stopping distance and stopping time.

When this option is loaded, this function cannot be disabled.

The stop type of DCS Position and Speed Check functions is not affected by the loading of this option.

WARNING

The stopping distance and stopping time of Controlled stop are longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the increased stopping distance and stopping time, is necessary when this option is loaded.

TABLE OF CONTENTS

SAFETY PRECAUTIONS.....	s-1
1 OVERVIEW OF PMC	1
1.1 WHAT IS PMC?.....	1
1.1.1 Basic Configuration of PMC.....	1
1.1.2 I/O Signals of PMC	2
1.1.3 PMC Signal Addresses.....	2
1.2 WHAT IS LADDER LANGUAGE?	4
1.2.1 Ladder Diagram Format	4
1.2.2 Signal Name (Symbol Name).....	5
1.2.3 Comment	5
1.2.4 Graphic Symbols of Relays and Coils.....	5
1.2.5 Line Number and Net Number	5
1.2.6 Difference Between Relay Sequence Circuit and Ladder Sequence Program	6
1.2.7 Specification of Extended Symbol and Comment.....	7
1.3 SEQUENCE PROGRAM CREATION PROCEDURE	9
1.3.1 Determining Specification.....	9
1.3.2 Creating Ladder Diagram	9
1.3.3 Editing Sequence Program	10
1.3.4 Transferring and Writing Sequence Program to PMC	11
1.3.5 Checking Sequence Program.....	11
1.3.6 Storage and Management of Sequence Program.....	11
1.4 EXECUTION OF SEQUENCE PROGRAM	11
1.4.1 Execution Procedure of Sequence Program	12
1.4.2 Repetitive Operation.....	13
1.4.3 Processing Priority (1st Level, 2nd Level, and 3rd Level).....	13
1.4.4 Structured Sequence Program	15
1.4.4.1 Implementation	15
1.4.4.2 Subprogramming and nesting	19
1.4.5 Synchronization Processing of I/O Signals	23
1.4.6 Interlock	26
1.5 MULTI-PATH PMC FUNCTION.....	26
1.5.1 Execution Order and Execution Time Percentage.....	27
1.5.2 Multi-Path PMC Interface	28
1.5.3 Common PMC Memory Mode of Multi-Path PMC.....	29
1.6 SAVE/LOAD PMC RELATED DATA	31

2	PMC SPECIFICATIONS.....	33
2.1	SPECIFICATIONS.....	33
2.1.1	Basic Specifications	33
2.1.2	Determination of PMC Memory Type	35
2.1.3	Program Capacity	35
2.1.4	Used Memory Size of Sequence Program.....	36
2.1.5	PMC Addresses	37
2.1.6	Basic Instructions	38
2.1.7	Functional Instructions (Arranged in Sequence of Instruction Group)	39
2.1.8	Functional Instructions (Arranged in Sequence of SUB No.)	46
2.2	PMC SIGNAL ADDRESSES.....	52
2.2.1	Addresses for Signals Between the PMC and ROBOT (F, G).....	52
2.2.2	Addresses of Signals Between the PMC and External I/O Device (X, Y)	52
2.2.3	Internal Relay Addresses (R).....	52
2.2.4	System Relay Addresses (R9000, Z0).....	53
2.2.5	Extra Relay Addresses (E).....	57
2.2.6	Message Display Addresses (A).....	58
2.2.7	Timer Addresses (T).....	58
2.2.8	Counter Addresses (C)	59
2.2.9	Keep Relay Addresses (K)	61
2.2.10	System Keep Relay Addresses (K).....	62
2.2.11	Data Table Addresses (D)	63
2.2.12	Addresses for Multi-path PMC Interface (M, N)	67
2.2.13	Subprogram Number Addresses (P)	68
2.2.14	Label Number Addresses (L)	68
2.3	PMC PARAMETERS	68
2.3.1	Cautions for Reading from/Writing to Nonvolatile Memory	70
2.3.2	PMC Parameter Format	70
2.4	PARAMETERS FOR THE PMC SYSTEM.....	75
2.4.1	Setting Parameters	75
2.5	COMPATIBILITY BETWEEN PMC MEMORY TYPE	76
2.5.1	Compatibility between PMC Memory-A and PMC Memory-B	76
2.5.2	Compatibility between PMC Memory-B and PMC Memory-C/D.....	76
2.5.3	Compatibility with PMC Memory-C and PMC Memory-D	77
3	PMC I/O ASSIGNMENT	78
3.1	I/O ASSIGNMENT ON ROBOT CONTROLLER.....	78
3.2	PMC EXTERNAL I/O ASSIGNMENT	78

3.2.1	Relationship with I/O Assignment (Robot)	82
3.2.2	Override Function.....	83
3.2.3	Occupancy Setting.....	85
3.3	PMC INTERNAL I/O ASSIGNMENT.....	87
3.3.1	Robot : DO → PMC : F.....	89
3.3.2	Robot : DI ← PMC : G	90
3.3.3	Robot : GO → PMC : F.....	91
3.3.4	Robot : GI ← PMC : G	91
3.3.5	Robot : UO → PMC : F.....	92
3.3.6	Robot : UI ← PMC : G	93
3.3.7	Robot : INFO → PMC : F	94
3.3.8	Robot : UALM ← PMC : G	95
3.3.9	Robot : Register → PMC : F	96
3.3.10	Robot : Register ← PMC : G	97
3.3.11	Robot : DI,RI,WI,WSI,SI,*UI → PMC : X	98
3.3.12	Robot : DO,RO,WO,WSO,SO,*UO ← PMC : Y	99
3.3.13	Robot : GI,AI → PMC : F	101
3.3.14	Robot : GO,AO ← PMC : G	102
3.3.15	Robot : DO[10001-10136] → PMC1: K0-17 Robot : DO[10137-10160] → PMC1: K900-902 Robot : DO[11001-23000] → PMC1: R0-1499 Robot : GO[10001-11500] → PMC1: D0-2999.....	103
3.4	EXAMPLE OF PMC I/O ASSIGNMENT	104
3.4.1	Example 1 : UOP is Controlled by External I/O Device.....	104
3.4.2	Example 2 : UOP is controlled by PMC	108
3.5	COMPATIBILITY WITH CONVENTIONAL MODELS	113
3.5.1	The Convert Method of Source Program Using FANUC LADDER-III for Robot	114
3.5.2	Compatibility Setup Function	115
4	LADDER LANGUAGE	119
4.1	BASIC INSTRUCTIONS	119
4.1.1	Details of the Basic Instructions.....	120
4.1.2	RD Instruction	122
4.1.3	RD.NOT Instruction.....	123
4.1.4	WRT Instruction.....	124
4.1.5	WRT.NOT Instruction.....	125
4.1.6	AND Instruction	126

4.1.7	AND.NOT Instruction.....	127
4.1.8	OR Instruction	128
4.1.9	OR.NOT Instruction.....	129
4.1.10	RD.STK Instruction.....	130
4.1.11	RD.NOT.STK Instruction	131
4.1.12	AND.STK Instruction	132
4.1.13	OR.STK Instruction.....	133
4.1.14	SET Instruction.....	134
4.1.15	RST Instruction	135
4.1.16	RDPT Instruction	136
4.1.17	ANDPT Instruction	137
4.1.18	ORPT Instruction	138
4.1.19	RDPT.STK Instruction.....	139
4.1.20	RDNT Instruction.....	140
4.1.21	ANDNT Instruction.....	141
4.1.22	ORNt Instruction.....	142
4.1.23	RDNT.STK Instruction	143
4.1.24	PUSH Instruction / POP Instruction.....	144
4.2	FUNCTIONAL INSTRUCTIONS	145
4.2.1	Format of the Functional Instructions	145
4.3	TIMER	150
4.3.1	TMR (On-delay Timer: SUB 3)	150
4.3.2	TMRB (Fixed On-delay Timer: SUB 24)	151
4.3.3	TMRBF (Fixed Off-delay Timer: SUB 77).....	153
4.3.4	TMRC (On-delay Timer: SUB 54).....	154
4.3.5	TMRST (Stop Watch Timer (1ms Accuracy) : SUB 221) TMRSS (Stop Watch Timer (1sec Accuracy) : SUB 222).....	156
4.4	COUNTER	160
4.4.1	CTR (Counter: SUB 5).....	160
4.4.2	CTRB (Fixed Counter: SUB 56)	164
4.4.3	CTRC (Counter: SUB 55)	166
4.4.4	CTRD (Counter (4 Bytes Length) : SUB 223).....	168
4.5	DATA TRANSFER	171
4.5.1	MOVB (Transfer of 1 Byte: SUB 43)	171
4.5.2	MOVW (Transfer of 2 Bytes: SUB 44)	172
4.5.3	MOVD (Transfer of 4 Bytes: SUB 47)	173
4.5.4	MOVN (Transfer of an Arbitrary Number of Bytes: SUB 45)	174

4.5.5	MOVE (Logical Product Transfer: SUB 8).....	175
4.5.6	MOVOR (Data Transfer After Logical Sum: SUB 28).....	176
4.5.7	XMOVB (Binary Index Modifier Data Transfer: SUB 35)	177
4.5.8	XMOV (Indexed Data Transfer: SUB 18)	184
4.5.9	MOVBT (Bit Transfer: SUB 224).....	186
4.5.10	SETNB (Data Setting (1 Byte Length) : SUB 225) SETNW (Data Setting (2 Bytes Length) : SUB 226) SETND (Data Setting (4 Bytes Length) : SUB 227).....	189
4.5.11	XCHGB (Data Exchange (1 Byte Length) : SUB 228) XCHGW (Data Exchange (2 Bytes Length) : SUB 229) XCHGD (Data Exchange (4 Bytes Length) : SUB 230).....	191
4.5.12	SWAPW (Data Swap (2 Bytes Length) : SUB 231) SWAPD (Data Swap (4 Bytes Length) : SUB 232).....	193
4.5.13	DSCHB (Binary Data Search: SUB 34).....	195
4.5.14	DSCH (Data Search: SUB 17)	198
4.6	TABLE DATA.....	200
4.6.1	TBLRB (Reading Data from Table (1 Byte Length) : SUB 233) TBLRW (Reading Data from Table (2 Bytes Length) : SUB 234) TBLRD (Reading Data from Table (4 Bytes Length) : SUB 235).....	201
4.6.2	TBLRN (Reading Data from Table (Arbitrary Bytes Length) : SUB 236).....	203
4.6.3	TBLWB (Writing Data to Table (1 Byte Length) : SUB 237) TBLWW (Writing Data to Table (2 Bytes Length) : SUB 238) TBLWD (Writing Data to Table (4 Bytes Length) : SUB 239)	206
4.6.4	TBLWN (Writing Data to Table (Arbitrary Bytes Length) : SUB 240)	208
4.6.5	DSEQB (Searching Data from Table(=)(1 Byte Length):SUB 241) DSEQW (Searching Data from Table(=)(2 Bytes Length):SUB 242) DSEQD (Searching Data from Table(=)(4 Bytes Length):SUB 243) DSNEB (Searching Data from Table(≠)(1 Byte Length):SUB 244) DSNEW (Searching Data from Table(≠)(2 Bytes Length):SUB 245) DSNED (Searching Data from Table(≠)(4 Bytes Length):SUB 246) DSGTB (Searching Data from Table(>)(1 Byte Length):SUB 247) DSGTW (Searching Data from Table(>)(2 Bytes Length):SUB 248) DSGTD (Searching Data from Table(>)(4 Bytes Length):SUB 249) DSLTB (Searching Data from Table(<)(1 Byte Length):SUB 250) DSLTw (Searching Data from Table(<)(2 Bytes Length):SUB 251) DSLTD (Searching Data from Table(<)(4 Bytes Length):SUB 252) DSGEB (Searching Data from Table(≥)(1 Byte Length):SUB 253)	

DSGEW (Searching Data from Table(\geq))(2 Bytes Length):SUB 254)	
DSGED (Searching Data from Table(\geq))(4 Bytes Length) :SUB 255)	
DSLEB (Searching Data from Table(\leq))(1 Byte Length) :SUB 256)	
DSLEW (Searching Data from Table(\leq))(2 Bytes Length) :SUB 257)	
DSLED (Searching Data from Table(\leq))(4 Bytes Length) :SUB 258).....	211
4.6.6 DMAXB (Maximum Data (1 Byte Length): SUB 259)	
DMAXW (Maximum Data (2 Bytes Length) : SUB 260)	
DMAXD (Maximum Data (4 Bytes Length) : SUB 261)	215
4.6.7 DMINB (Minimum Data (1 Byte Length): SUB 262)	
DMINW (Minimum Data (2 Bytes Length): SUB 263)	
DMIND (Minimum Data (4 Bytes Length): SUB 264)	218
4.7 COMPARISON	221
4.7.1 Signed Binary Comparison (=)	
EQB (1 Byte Length: SUB 200)	
EQW (2 Bytes Length: SUB 201)	
EQD (4 Bytes Length: SUB 202).....	222
4.7.2 Signed Binary Comparison (\neq)	
NEB (1 Byte Length: SUB 203)	
NEW (2 Bytes Length: SUB 204)	
NED (4 Bytes Length: SUB 205).....	224
4.7.3 Signed Binary Comparison ($>$)	
GTB (1 Byte Length: SUB 206)	
GTW (2 Bytes Length: SUB 207)	
GTD (4 Bytes Length: SUB 208).....	225
4.7.4 Signed Binary Comparison ($<$)	
LTB (1 Byte Length: SUB 209)	
LTW (2 Bytes Length: SUB 210)	
LTD (4 Bytes Length: SUB 211)	226
4.7.5 Signed Binary Comparison (\geq)	
GEB (1 Byte Length: SUB 212)	
GEW (2 Bytes Length: SUB 213)	
GED (4 Bytes Length: SUB 214).....	228
4.7.6 Signed Binary Comparison (\leq)	
LEB (1 Byte Length: SUB 215)	
LEW (2 Bytes Length: SUB 216)	
LED (4 Bytes Length: SUB 217)	229

4.7.7	Signed Binary Comparison (Range) RNGB (1 Byte Length: SUB 218) RNGW (2 Bytes Length: SUB 219) RNGD (4 Bytes Length: SUB 220).....	230
4.7.8	COMPB (Comparison Between Binary Data: SUB 32).....	232
4.7.9	COMP (Comparison: SUB 15).....	234
4.7.10	COIN (Coincidence Check: SUB 16).....	235
4.8	BIT OPERATION	237
4.8.1	DIFU (Rising Edge Detection: SUB 57)	238
4.8.2	DIFD (Falling Edge Detection: SUB 58)	239
4.8.3	EOR (Exclusive OR: SUB 59)	240
4.8.4	AND (Logical AND: SUB 60)	241
4.8.5	OR (Logical OR: SUB 61)	243
4.8.6	NOT (Logical NOT: SUB 62).....	244
4.8.7	PARI (Parity Check: SUB 11).....	246
4.8.8	SFT (Shift Register: SUB 33)	247
4.8.9	EORB (Exclusive OR (1 Byte Length) : SUB 265) EORW (Exclusive OR (2 Bytes Length) : SUB 266) EORD (Exclusive OR (4 Bytes Length) : SUB 267)	249
4.8.10	ANDB (Logical AND (1 Byte Length) : SUB 268) ANDW (Logical AND (2 Bytes Length) : SUB 269) ANDD (Logical AND (4 Bytes Length) : SUB 270)	251
4.8.11	ORB (Logical OR (1 Byte Length) : SUB 271) ORW (Logical OR (2 Bytes Length) : SUB 272) ORD (Logical OR (4 Bytes Length) : SUB 273)	253
4.8.12	NOTB (Logical NOT (1 Byte Length) : SUB 274) NOTW (Logical NOT (2 Bytes Length) : SUB 275) NOTD (Logical NOT (4 Bytes Length) : SUB 276)	255
4.8.13	SHLB (Bit Shift Left (1 Byte Length) : SUB 277) SHLW (Bit Shift Left (2 Bytes Length) : SUB 278) SHLD (Bit Shift Left (4 Bytes Length) : SUB 279).....	257
4.8.14	SHLN (Bit Shift Left (Arbitrary Bytes Length) : SUB 280).....	260
4.8.15	SHRB (Bit Shift Right (1 Byte Length) : SUB 281) SHRW (Bit Shift Right (2 Bytes Length) : SUB 282) SHRD (Bit Shift Right (4 Bytes Length) : SUB 283)	262
4.8.16	SHRN (Bit Shift Right (Arbitrary Bytes Length) : SUB 284)	265

4.8.17	ROLB (Bit Rotation Left (1 Byte Length) : SUB 285) ROLW (Bit Rotation Left (2 Bytes Length) : SUB 286) ROLD (Bit Rotation Left (4 Bytes Length) : SUB 287)	267
4.8.18	ROLN (Bit Rotation Left (Arbitrary Bytes Length) : SUB 288)	270
4.8.19	RORB (Bit Rotation Right (1 Byte Length) : SUB 289) RORW (Bit Rotation Right (2 Bytes Length) : SUB 290) RORD (Bit Rotation Right (4 Bytes Length) : SUB 291).....	272
4.8.20	RORN (Bit Rotation Right (Arbitrary Bytes Length) : SUB 292).....	275
4.8.21	BSETB (Bit Set (1 Byte Length) : SUB 293) BSETW (Bit Set (2 Bytes Length) : SUB 294) BSETD (Bit Set (4 Bytes Length) : SUB 295).....	277
4.8.22	BSETN (Bit Set (Arbitrary Bytes Length) : SUB 296).....	279
4.8.23	BRSTB (Bit Reset (1 Byte Length) : SUB 297) BRSTW (Bit Reset (2 Bytes Length) : SUB 298) BRSTD (Bit Reset (4 Bytes Length) : SUB 299).....	281
4.8.24	BRSTN (Bit Reset (Arbitrary Bytes Length) : SUB 300).....	283
4.8.25	BTSTB (Bit Test (1 Byte Length) : SUB 301) BTSTW (Bit Test (2 Bytes Length) : SUB 302) BTSTD (Bit Test (4 Bytes Length) : SUB 303).....	285
4.8.26	BTSTN (Bit Test (Arbitrary Bytes Length) : SUB 304).....	286
4.8.27	BPOSB (Bit Search (1 Byte Length) : SUB 305) BPOSW (Bit Search (2 Bytes Length) : SUB 306) BPOSD (Bit Search (4 Bytes Length) : SUB 307).....	288
4.8.28	BPOSN (Bit Search (Arbitrary Bytes Length) : SUB 308).....	290
4.8.29	BCNTB (Bit Count (1 Byte Length) : SUB 309) BCNTW (Bit Count (2 Bytes Length) : SUB 310) BCNTD (Bit Count (4 Bytes Length) : SUB 311)	292
4.8.30	BCNTN (Bit Count (Arbitrary Bytes Length) : SUB 312)	293
4.9	CODE CONVERSION	295
4.9.1	COD (Code Conversion: SUB 7)	295
4.9.2	CODB (Binary Code Conversion: SUB 27).....	298
4.9.3	DCNV (Data Conversion: SUB 14)	300
4.9.4	DCNVB (Extended Data Conversion: SUB 31).....	301
4.9.5	DEC (Decode: SUB 4)	303
4.9.6	DECB (Binary Decoding: SUB 25)	305

4.9.7	TBCDB (Binary to BCD Conversion (1 Byte Length) : SUB 313)	
	TBCDW (Binary to BCD Conversion (2 Bytes Length) : SUB 314)	
	TBCDD (Binary to BCD Conversion (4 Bytes Length) : SUB 315)	308
4.9.8	FBCDB (BCD to Binary Conversion (1 Byte Length) : SUB 313)	
	FBCDW (BCD to Binary Conversion (2 Bytes Length) : SUB 314)	
	FBCDD (BCD to Binary Conversion (4 Bytes Length) : SUB 315).....	309
4.10	OPERATION INSTRUCTION	312
4.10.1	ADDB (Binary Addition: SUB 36)	312
4.10.2	SUBB (Binary Subtraction: SUB 37).....	314
4.10.3	MULB (Binary Multiplication: SUB 38)	316
4.10.4	DIVB (Binary Division: SUB 39)	318
4.10.5	ADD (BCD Addition: SUB 19)	319
4.10.6	SUB (BCD Subtraction: SUB 20)	321
4.10.7	MUL (BCD Multiplication: SUB 21).....	322
4.10.8	DIV (BCD Division: SUB 22)	324
4.10.9	NUMEB (Definition of Binary Constants: SUB 40).....	325
4.10.10	NUME (BCD Definition of Constant: SUB 23).....	327
4.10.11	ADDSB (Addition (1 Byte Length) : SUB 319)	
	ADDSW (Addition (2 Bytes Length) : SUB 320)	
	ADDSD (Addition (4 Bytes Length) : SUB 321)	328
4.10.12	SUBSB (Subtraction (1 Byte Length) : SUB 322)	
	SUBSW (Subtraction (2 Bytes Length) : SUB 323)	
	SUBSD (Subtraction (4 Bytes Length) : SUB 324)	330
4.10.13	MULSB (Multiplication (1 Byte Length) : SUB 325)	
	MULSW (Multiplication (2 Bytes Length) : SUB 326)	
	MULSD (Multiplication (4 Bytes Length) : SUB 327).....	332
4.10.14	DIVSB (Division (1 Byte Length) : SUB 328)	
	DIVSW (Division (2 Bytes Length) : SUB 329)	
	DIVSD (Division (4 Bytes Length) : SUB 330)	334
4.10.15	MODSB (Remainder (1 Byte Length) : SUB 331)	
	MODSW (Remainder (2 Bytes Length) : SUB 332)	
	MODSD (Remainder (4 Bytes Length) : SUB 333).....	336
4.10.16	INCSB (Increment (1 Byte Length) : SUB 334)	
	INCSW (Increment (2 Bytes Length) : SUB 335)	
	INCSD (Increment (4 Bytes Length) : SUB 336)	338

4.10.17	DECSB (Decrement (1 Byte Length) : SUB 337) DECSW (Decrement (2 Bytes Length) : SUB 338) DECSD (Decrement (4 Bytes Length) : SUB 339).....	340
4.10.18	ABSSB (Absolute Value (1 Byte Length) : SUB 340) ABSSW (Absolute Value (2 Bytes Length) : SUB 341) ABSSD (Absolute Value (4 Bytes Length) : SUB 342).....	341
4.10.19	NEGSB (Sign Inversion (1 Byte Length) : SUB 343) NEGSW (Sign Inversion (2 Bytes Length) : SUB 344) NEGSD (Sign Inversion (4 Bytes Length) : SUB 345).....	343
4.11	PROGRAM CONTROL	344
4.11.1	COM (Common Line Control: SUB 9)	345
4.11.2	COME (Common Line Control End: SUB 29)	347
4.11.3	JMP (Jump: SUB 10)	347
4.11.4	JMPE (Jump End: SUB 30).....	349
4.11.5	JMPB (Label Jump 1: SUB 68).....	349
4.11.6	JMPC (Label Jump 2: SUB 73).....	351
4.11.7	LBL (Label: SUB 69).....	352
4.11.8	CALL (Conditional Subprogram Call: SUB 65).....	353
4.11.9	CALLU (Unconditional Subprogram Call: SUB 66).....	354
4.11.10	SP (Subprogram: SUB 71)	354
4.11.11	SPE (End of a Subprogram: SUB 72)	355
4.11.12	END1 (1st Level Sequence Program End: SUB 1)	356
4.11.13	END2 (2nd Level Sequence Program End: SUB 2).....	356
4.11.14	END3 (3rd Level Sequence Program End: SUB 48)	356
4.11.15	END (End of a Ladder Program: SUB 64).....	357
4.11.16	NOP (No Operation: SUB 70).....	357
4.11.17	CS (Case Call: SUB 74)	357
4.11.18	CM (Sub Program Call in Case Call: SUB 75)	359
4.11.19	CE (End of Case Call: SUB 76)	360
4.12	ROTATION CONTROL	360
4.12.1	ROT (Rotation Control: SUB 6)	360
4.12.2	ROTB (Binary Rotation Control: SUB 26)	363
4.13	INVALID INSTRUCTIONS	365
5	FANUC LADDER-III FOR ROBOT PROGRAMMING.....	366
5.1	CONNECTING FANUC LADDER-III for Robot TO ROBOT CONTROLLER	367
5.1.1	RS-232-C Connection	367

5.1.2	Ethernet Connection	371
5.1.3	Connection with ROBOGUIDE	375
5.2	CREATING PMC PROGRAM.....	379
5.3	EDITTING A PMC PROGRAM.....	381
5.4	COMPILEING A PMC PROGRAM.....	382
5.5	TRANSFERRING PMC PROGRAM	383
5.6	PMC PROGRAM MONITORING (ONLINE MONITOR)	387
5.7	RUNNING OR STOPPING PMC PROGRAM.....	399
5.8	WRITING PMC PROGRAM TO F-ROM	400
5.9	MODIFYING THE PMC PROGRAM IN THE ROBOT CONTROLLER	401
5.10	STORE/LOAD PMC PROGRAM	404
5.10.1	Exporting LADDER*.PMC using FANUC LADDER-III for Robot	404
5.10.2	Importing LADDER*.PMC using FANUC LADDER-III for Robot	406
6	TEACH PENDANT OPERATION.....	409
6.1	PMC Menus	409
6.1.1	Byte Menu	410
6.1.2	Bit Menu.....	412
6.1.3	Timer Menu.....	413
6.1.4	Counter Menu.....	414
6.1.5	Data Table Control Data Menu	415
6.1.6	Data Table Menu	416
6.1.7	Parameters Menu	418
6.1.8	Status Menu.....	419
6.1.9	Step Status Menu.....	419
6.1.10	Title Menu	420
6.1.11	PMC Setting Menu.....	420
6.1.12	External I/O Assignment Menu.....	422
6.1.13	Internal I/O Assignment Menu	424
6.1.14	Search	425
6.1.15	Run/Stop PMC	426
6.1.16	Change PMC Path	427
6.1.17	Save LADDER*.PMC and PARAM*.PMC	428
6.2	FILE MENU OPERATIONS	429
6.2.1	Save LADDER*.PMC, PARAM*.PMC, PMCCFG.SV.....	430
6.2.2	Load LADDER*.PMC, PARAM*.PMC, PMCCFG.SV	431
6.3	PMC MONITOR FUNCTION	433
6.4	PMC EDIT FUNCTION	436

7	STEP SEQUENCE FUNCTION.....	440
7.1	OVERVIEW	440
7.1.1	Step Sequence Method	440
7.1.2	Graphical Symbols	442
7.1.3	Editing and Debugging Step Sequence Programs	443
7.2	STEP SEQUENCE BASICS	444
7.2.1	Terminology	444
7.2.2	Execution of Step Sequence	452
7.3	CONFIGURATION AND OPERATION OF STEP-SEQUENCE PROGRAMS	454
7.3.1	Step.....	454
7.3.2	Initial Step	456
7.3.3	Transition	457
7.3.4	Divergence of Selective Sequence	457
7.3.5	Convergence of Selective Sequence.....	458
7.3.6	Divergence of Simultaneous Sequence	458
7.3.7	Convergence of Simultaneous Sequence.....	459
7.3.8	Jump	460
7.3.9	Label.....	460
7.3.10	Block Step	461
7.3.11	Initial Block Step.....	461
7.3.12	End Of Block Step.....	462
7.4	EXTENDED LADDER INSTRUCTIONS.....	462
7.4.1	Functional Instruction TRSET	462
7.4.2	PMC Address (S Address).....	463
7.5	SPECIFICATION OF STEP SEQUENCE	464
7.5.1	Specification.....	464
7.5.2	General Rules	465
7.5.3	Exclusive Control for Functional Instructions	469
8	FUNCTION BLOCK FUNCTION.....	471
8.1	OVERVIEW	471
8.1.1	Item Names.....	472
8.1.2	Overview of Specifications	472
8.1.3	Memory Usage Related to Function Blocks.....	475
8.1.4	Assignment of FB Variable	476
8.2	FUNCTION BLOCK DEFINITION.....	477
8.2.1	Function Block Name.....	477

8.2.2	Variable Information	477
8.2.3	FB Body Program.....	484
8.2.4	Other Information.....	486
8.3	FUNCTION BLOCK CALL	487
8.3.1	Function Block Call Positions	487
8.3.2	Creating a Function Block Call Section	487
8.4	EXECUTING A FUNCTION BLOCK.....	489

1 OVERVIEW OF PMC

1.1 WHAT IS PMC?

The programmable machine controller (PMC) is a programmable controller (PC) built into a Robot controller to perform sequence control for a robot system.

Sequence control is to perform control steps successively in a predetermined sequence or according to the logic operation.

Programs for performing sequence control for machine tools are called sequence programs. Generally, sequence programs coded in the Ladder language are used.

To use PMC, the option "Integrated PMC(J760)" is necessary.

In addition, the following functions are available if the option "Multi-path integrated PMC(J763)" is ordered.

- Multi-path PMC function
- Step sequence function
- Function block function
- Sequence program memory size expansion (Total 24,000 → 300,000 steps, memory size 256KB → 3MByte)
- PMC memory expansion (PMC memory type C and D are available for 1st PMC)
- Allow to use 4msec for level 1 execution period

To create PMC sequence program, FANUC LADDER-III for Robot is needed.

To use PMC in R-30iB Mate controller, the main board that has PMC function is necessary.

PMC display menu on teach pendant can display ladder diagram and the sequence program can be changed on teach pendant if "PMC change mode(R652)" is ordered.

In addition, the sequence control of safety I/O can be performed if the option "DCS Safety PMC(J764)" is ordered with "Integrated PMC(J760)". Please refer to "FANUC Robot Series R-30iB/ R-30iB Mate Controller Dual Check Safety Operator's Manual (B-83184EN)" for DCS Safety PMC function.

1.1.1 Basic Configuration of PMC

The following is the basic configuration of the PMC:

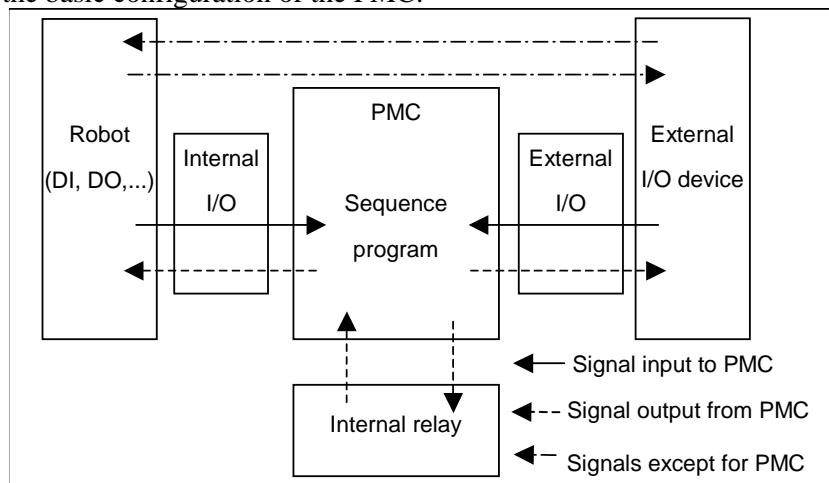


Fig. 1.1.1 Basic configuration of PMC

The sequence program reads input signals, performs operations, and outputs results in a predetermined sequence.

1.1.2 I/O Signals of PMC

Input signals of the PMC include signals input from the Robot and signals input from the external I/O device. Output signals of the PMC include signals output to the Robot and signals output to the external I/O device. The PMC controls these I/O signals by executing a sequence program to control the robot system.

1.1.3 PMC Signal Addresses

PMC signal addresses indicate the locations of I/O signals exchanged with the external I/O device, I/O signals exchanged with the Robot, and signals for internal relays and data (PMC parameters) in nonvolatile memory.

PMC addresses are roughly classified as shown in Fig. 1.1.3 (a).

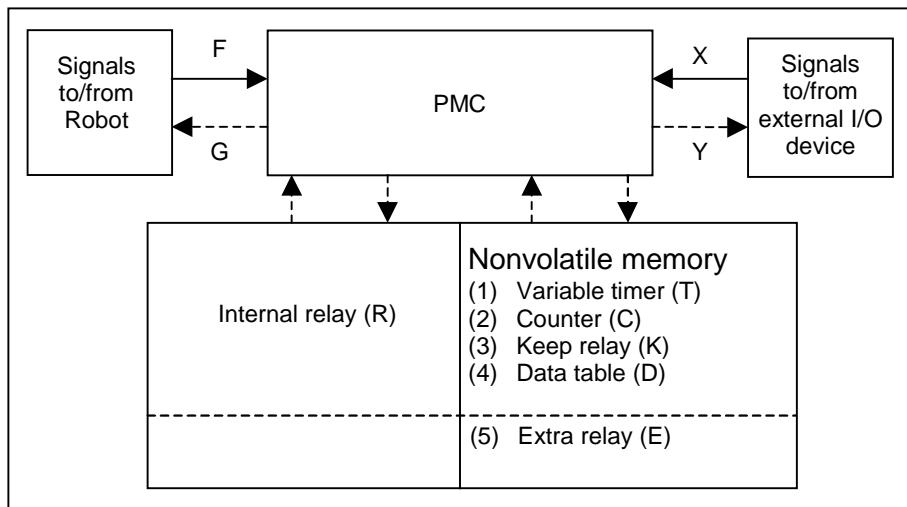


Fig. 1.1.3 (a) PMC-related addresses

The PMC signal address format consists of an address number and bit number (0 to 7) as follows:

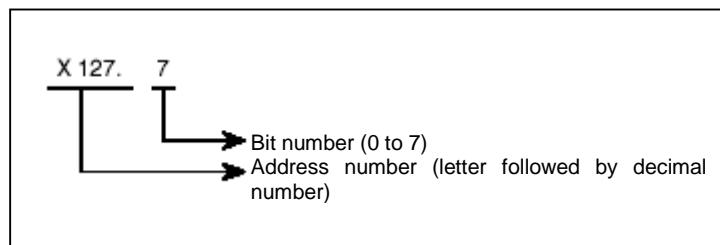


Fig. 1.1.3 (b) PMC address format

The first letter of an address number represents the type of the signal.

In sequence programs, an address of a byte may be specified. In the above example, specify X127 to specify a byte address. In this case, the period "." and bit number are unnecessary.

Table 1.1.3 lists the address symbols and corresponding signals.

Table 1.1.3 Address Symbols and signal types

Symbol	Signal type
F	Input signal from Robot to PMC
G	Output signal from PMC to Robot
X	Input signal from external I/O device to PMC
Y	Output signal from PMC to external I/O device
R	Internal relay
E	Extra relay
A	Message display
T	Variable timer
C	Counter
K	Keep relay
D	Data table
M	Input signal from another PMC path
N	Output signal to another PMC path
L	Label number
P	Subprogram number

(1) Addresses of signals between the PMC and Robot (F and G)

These addresses are assigned to interface signals between the Robot and PMC. The relationships between the signals and addresses are defined by the PMC internal I/O assignment.

F indicates an input signal from the Robot to PMC.

G indicates an output signal from the PMC to Robot.

(2) Addresses of signals between the PMC and external I/O device (X and Y)

I/O signals exchanged with an external I/O device can be assigned to any addresses within an available range to control the external I/O device.

X indicates an input signal from the external I/O device to PMC.

Y indicates an output signal from the PMC to external I/O device.

(3) Addresses of internal relays (R)

These addresses are used to temporarily store operation results during sequence program execution processing.

The address locations of internal relays also include a reserved area used by the PMC system software. The signals in the reserved area cannot be written by sequence programs.

(4) Signal addresses for message display (A)

This address is not used.

(5) Nonvolatile memory addresses

The contents of these address locations are not erased even when the power is turned off.

These addresses are used for management of the data items listed below. These data items are called PMC parameters.

(a) Variable timer (T)

(b) Counter (C)

(c) Keep relay (K)

A reserved area used by the PMC system software is partly included.

(d) Data table (D)

(e) Extra relay (E)

These addresses are used to temporarily store operation results during sequence program execution processing.

(6) Multi-path PMC interface address (M, N)

These addresses are used for the Multi-path PMC interface.

M indicates an input signal from another PMC path.

N indicates an output signal to another PMC path.

(7) Other addresses

(a) Label number (L)

Sequence program instructions include an instruction to cause a jump to a specified position in the middle of processing. This address indicates the jump destination used by this instruction. The contents of L address cannot be read/written in sequence program.

(b) Subprogram number (P)

In sequence programs, a main program can call subprograms. P addresses indicate the numbers of these subprograms. The contents of P address cannot be read/written in sequence program.

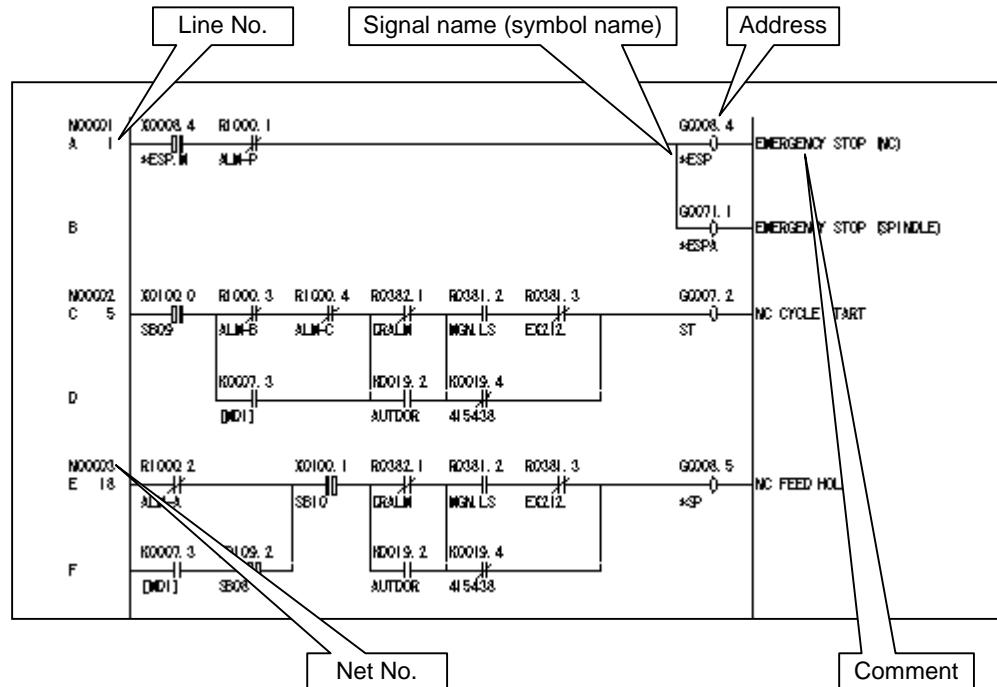
1.2 WHAT IS LADDER LANGUAGE?

The Ladder language is one of sequence programming languages. This programming language, which represents the sequence and logic operations of I/O signals by ladder diagrams, is widely used by sequence control engineers. This language is mainly used for PMCs.

1.2.1 Ladder Diagram Format

Designers develop and see ladder diagrams in the design stage. However, other people (for example, many maintenance engineers) have much more chances to see ladder diagrams than the designers of the ladder diagrams have. Therefore, the designers should create ladder diagrams so that these diagrams are intelligible to any one.

The following is the format of ladder diagrams:



The meanings of ladder diagram contents will be described later.

1.2.2 Signal Name (Symbol Name)

Symbol names representing I/O signal names can be assigned to PMC addresses. It is recommended that signal names (symbol names) suitable for I/O signals be assigned as explained below.

- (1) Signal names may consist of any alphanumeric characters and the special symbols. For the allowable number of characters, see the table in Subsection 2.1.1.
- (2) As the names between Robot and PMC, use the signal names corresponded to robot signal name like DO15, or corresponded to the I/O device name like IMSTP.
- (3) The same signal name (symbol name) cannot be assigned to more than one signal address.

1.2.3 Comment

A comment can be added to each symbol in the symbol table so that it can be indicated as a comment on a relay or coil in the sequence program. For the number of characters that can be entered, see the table in Subsection 2.1.1.

For all relays and coils that are output signals to the external I/O device, add a comment to provide a detailed signal explanation. For other auxiliary relays, provide explanations of the signals if these relays have significant meanings in sequence control.

In particular for external I/O device input signals, be sure to provide a detailed signal explanation as a comment in the symbol table.

Add detailed comments to signals dedicated to the external I/O device so that one can guess the meanings of these signals just from the symbol names.

1.2.4 Graphic Symbols of Relays and Coils

Ladder diagrams use the following relay symbols:

Relays (contacts)

Instruction representation	Function
- -	Normally open contact (contact A)
- / -	Normally closed contact (contact B)

Coils

Instruction representation	Function
-○-	Coil
-○○-	Negated coil
-(S)-	Set coil
-(R)-	Reset coil

These instructions perform a 1-bit operation and are called basic instructions.

In addition, there are functional instructions that enable easy programming of complicated operations for processing byte, word, and double-word data, which are difficult to program just using basic instructions. The symbol formats of the functional instructions are slightly different from instruction to instruction. For details, see the description of each functional instruction in Chapter 4.

1.2.5 Line Number and Net Number

A line number is indicated in every line of ladder diagrams.

A continuous ladder circuit from a contact to a coil is called a net. A net number is also indicated for each net.

1.2.6 Difference Between Relay Sequence Circuit and Ladder Sequence Program

In general relay sequence circuits, because of a limited number of relay contacts, one contact may be shared by several relays to minimize the number of contacts used. Fig. 1.2.6 (a) gives an example.

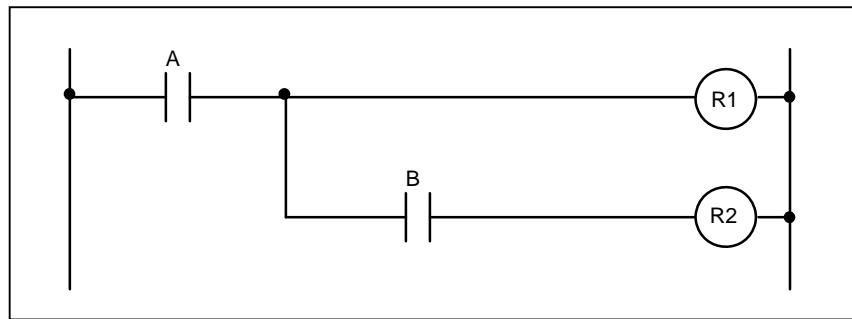


Fig. 1.2.6 (a)

With the PMC, relay contacts are considered to be unlimited, so ladder diagrams are created as shown in Fig. 1.2.6 (b).

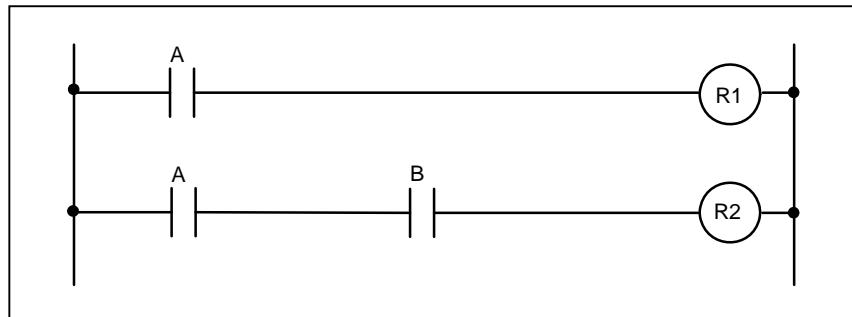


Fig. 1.2.6 (b)

In a relay sequence circuit, having no contact between a branch point and a coil as shown in Fig. 1.2.6 (c), a similar ladder diagram can be created even for the PMC.

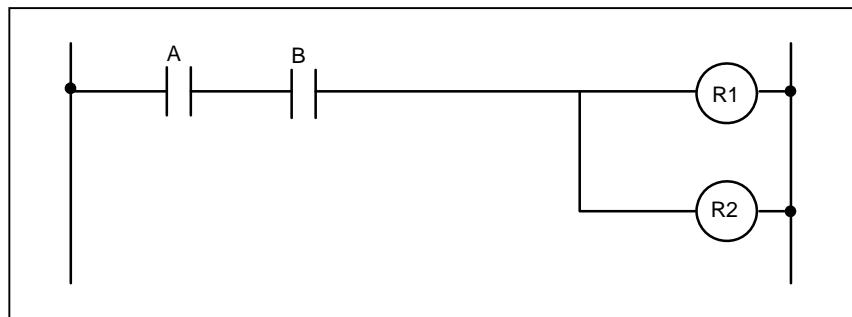


Fig. 1.2.6 (c)

NOTE

The extended PMC ladder instruction function allows the sequence circuits like Fig. 1.2.6(a).

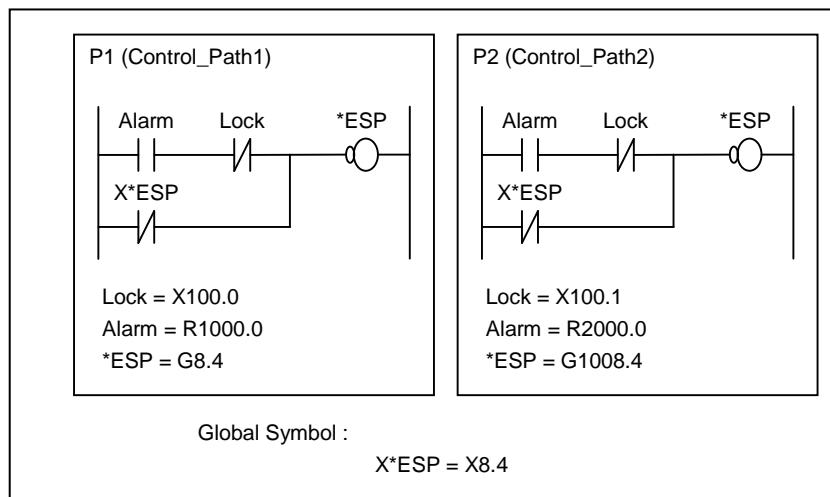
1.2.7 Specification of Extended Symbol and Comment

Using extended symbol and comment, you can use following functions.

- Local symbols effective in sub programs
- Extension of maximum character length of symbol and comment
- Multi comment support
- Multiple definitions of symbol and comment to one signal
- Data type definition
- Automatic address assignment at compiling on FANUC LADDER-III for Robot

(1) Local symbols effective in sub programs

You can define local symbols effective only in a sub program. So you can define local symbols having same string in other sub programs. Local symbols defined in different sub programs do not conflict.



Using local symbols, symbol conflict does not occur. Therefore, it is easy to develop ladder in modular programming technique. In addition, it is easy to reuse sequence programs. When you have to program a similar program in some sub programs, copy the logic to another sub program, redefine the local symbols, and compile on FANUC LADDER-III for Robot.

NOTE

- 1 Same local symbol names are not allowed in the same sub program.
- 2 Same symbol name of global symbol and local symbol are not allowed.
- 3 Local symbol cannot be defined to address P. Symbol definition to address P must be global symbol.
- 4 When you use the function block option, extended symbol and comment option is automatically selected.

(2) Extension of maximum character length of symbol and comment

Maximum character length of a symbol and comment is extended as follows. So you can describe in details.

Kind	Extended type	Former
symbol	40 characters in maximum	16 characters in maximum
comment	4 set 255 characters in maximum	1 set 30 characters in maximum

(3) Multi comment support

One symbol entry has four comments set in maximum but only first comment is actually used.

(4) Multiple definitions of symbol and comment to one signal

You can define multiple symbols and comments to the same signal.

NOTE

When multiple symbol and comment are defined to the same signal, you can search the names by each symbol. On the other hand the symbol on PMC screen is displayed one of these symbol names. So if you search symbols, displayed symbol name on searched position may be different from searched word.

(5) Data type definition

You can define symbol and comment with data type definition.

Data type	Meaning
BOOL	Boolean
BYTE	8 bits integer
WORD	16 bits integer
DWORD	32 bits integer
LABEL	Label (Address L)
PROG	Sub program (Address P)

NOTE

- 1 In ladder editing screen, for example, BYTE type symbol can be set to the WORD type parameter of a function. But it is recommended that data type of the symbol should be consistent with the data type of the parameter that it is assigned to.
- 2 When two or more symbols are defined with a signal and these symbols have different data types the symbol name of largest data type is displayed on PMC screens.

(6) Automatic address assignment at compiling on FANUC LADDER-III for Robot

On FANUC LADDER-III for Robot, when programming by symbol names without actual addresses, this function assigns addresses to them automatically.

⚠ CAUTION

The assignment of address may change by modifying symbol / comment data.

NOTE

By setting 1 to K903.5 of system keep relay, the signal state of the symbols whose addresses are assigned automatically can be initialized when updating sequence program to the one of different symbol / comment data.

In this case, changing the symbol / comment data whose address is not assigned automatically will also initialize all signal states of the address range for automatic assignment to 0.

(7) Available characters

These characters can be used.

- Available characters for symbols :

Kind	Extended type	Former
Characters that can be used for symbols	A to Z, a to z, 0 to 9, _ ! " # & ' () * + , - < = > ? @ [/] ^ ` { } ~ (Note)	A to Z, a to z, 0 to 9, _ Space, ! " # \$ % & ' () * + , - < = > ? @ [/] ^ ` { } ~ ; :
Characters that cannot be used for the first character of the symbols	% \$	
Characters that cannot be used for the symbols	Space, ; : .	

NOTE

Although it is allowed to use special characters in symbols, using only alphabets, digits and _(underscore) to comply with the variable name defined in IEC61131-3 is recommended.

- Available characters for comments:

Kind	Extended type	Former
Characters that can be used for comments	A to Z, a to z, 0 to 9, Space ! " # & ' () * + , - < = > ? @ [/] ^ ` { } ~ ; :	A to Z, a to z, 0 to 9, Space, ! " # \$ % & ' () * + , - < = > ? @ [/] ^ ` { } ~ ; :

1.3 SEQUENCE PROGRAM CREATION PROCEDURE

This section briefly explains how to create a program for providing sequence control by using the Ladder language as an example. When creating a sequence program, see the necessary manual for editing after understanding the contents of this chapter thoroughly.

1.3.1 Determining Specification

First, determine the specifications of the control target. Calculate the number of I/O signals, and determine the interfaces of the I/O signals.

In this step, creation of interface specifications is recommended.

1.3.2 Creating Ladder Diagram

After determining specifications, represent control operations with a ladder diagram. Timer, counter, and other functions that cannot be represented by relay symbols are called functional instructions. Represent these functional instructions with corresponding symbols.

When using offline programmer, you can enter a sequence program in a ladder diagram form. At the time of sequence program editing, you can make entry while creating a ladder diagram on the display screen, so you need not prepare a ladder diagram in advance.

If you want to create a sequence program efficiently, however, it is recommended that you should create a ladder diagram in advance.

Ladder diagrams are referenced as maintenance drawings by maintenance engineers. Therefore, try to create as intelligible ladder diagrams as possible.

1.3.3 Editing Sequence Program

A sequence program in the Ladder language is edited with one of the following two methods:

(1) PC programmer

FANUC supplies FANUC LADDER-III for Robot as sequence program development software for FANUC PMC. Use of FANUC LADDER-III for Robot allows you to edit a program in the Ladder language on a personal computer.

(2) PMC change mode (Option)

This is the function to change an existing sequence program on teach pendant of robot controller. This function can change existing elements of a sequence program of ladder language.

By using FANUC LADDER-III for Robot, a sequence program can be entered in a ladder diagram form from the EDITOR screen. FANUC LADDER-III for Robot can also output an entered sequence program to a printer in a ladder diagram form.

Furthermore, FANUC LADDER-III for Robot provides a function for converting a program in a ladder diagram form to mnemonic form or vice versa. By using this function, you can edit the program in mnemonic form with a text editor.

Fig. 1.3.3 shows an example of a ladder diagram, and Table 1.3.3 shows an example of a mnemonic form.

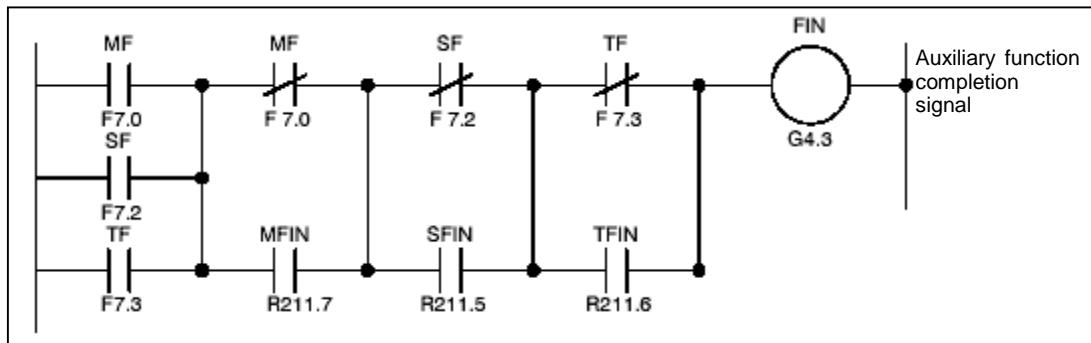


Fig. 1.3.3

Table 1.3.3

Step No.	Instruction	Address No. & bit No.	Remarks
1	RD	F7.0	MF
2	OR	F7.2	SF
3	OR	F7.3	TF
4	RD.NOT.STK	F7.0	MF
5	OR	R211.7	MFIN
6	AND.STK		
7	RD.NOT.STK	F7.2	SF
8	OR	R211.5	SFIN
9	AND.STK		
10	RD.NOT.STK	F7.3	TF
11	OR	R211.6	TFIN
12	AND.STK		
13	WRT	G4.3	FIN

During sequence program editing, signal names (symbols) and comments can be entered for I/O signals, relays, and coils. Easy-to-understand signal names and comments should be entered to improve program maintainability.

1.3.4 Transferring and Writing Sequence Program to PMC

After completing editing for the sequence program, input (transfer) the program to the PMC. This operation is unnecessary when you have changed the program by using the PMC change mode.

When you have edited the sequence program by using the PC programmer, input the sequence program from the editing environment (the personal computer (PC)) to the PMC. The following input methods can be used:

(1) Input from the file screen

The sequence program on the PC is input to the PMC via a memory card or a USB memory.

(2) Input from the online monitor screen

For data input, connect the PC containing the sequence program to the Robot controller via Ethernet or RS-232C. After inputting the sequence program, write it in the flash ROM. This operation can be done with the PC programmer.

1.3.5 Checking Sequence Program

After writing the sequence program in the flash ROM, check the sequence program.

The sequence program can be checked in the following two ways:

(1) Checking with a simulator (ROBOGUIDE)

Instead of using input signals from the external I/O device, turn the signals on and off to input signals, and confirm output signals by checking the on/off states in PMC menu.

(2) Checking by system operation

Connect the external I/O device to make checks. Before starting the operation, take safety measures because when the sequence program is executed for the first time, an unpredictable motion can occur.

1.3.6 Storage and Management of Sequence Program

When the sequence program is completed after checking, it should be stored and managed carefully.

The sequence program can be output to the printer in a ladder diagram form by using the PC programmer. The output ladder diagram should be managed as a maintenance drawing.

1.4 EXECUTION OF SEQUENCE PROGRAM

Sequence programs in the Ladder language are executed in the order of instructions coded in the ladder diagrams.

Fig. 1.4 shows how a sequence program is executed.

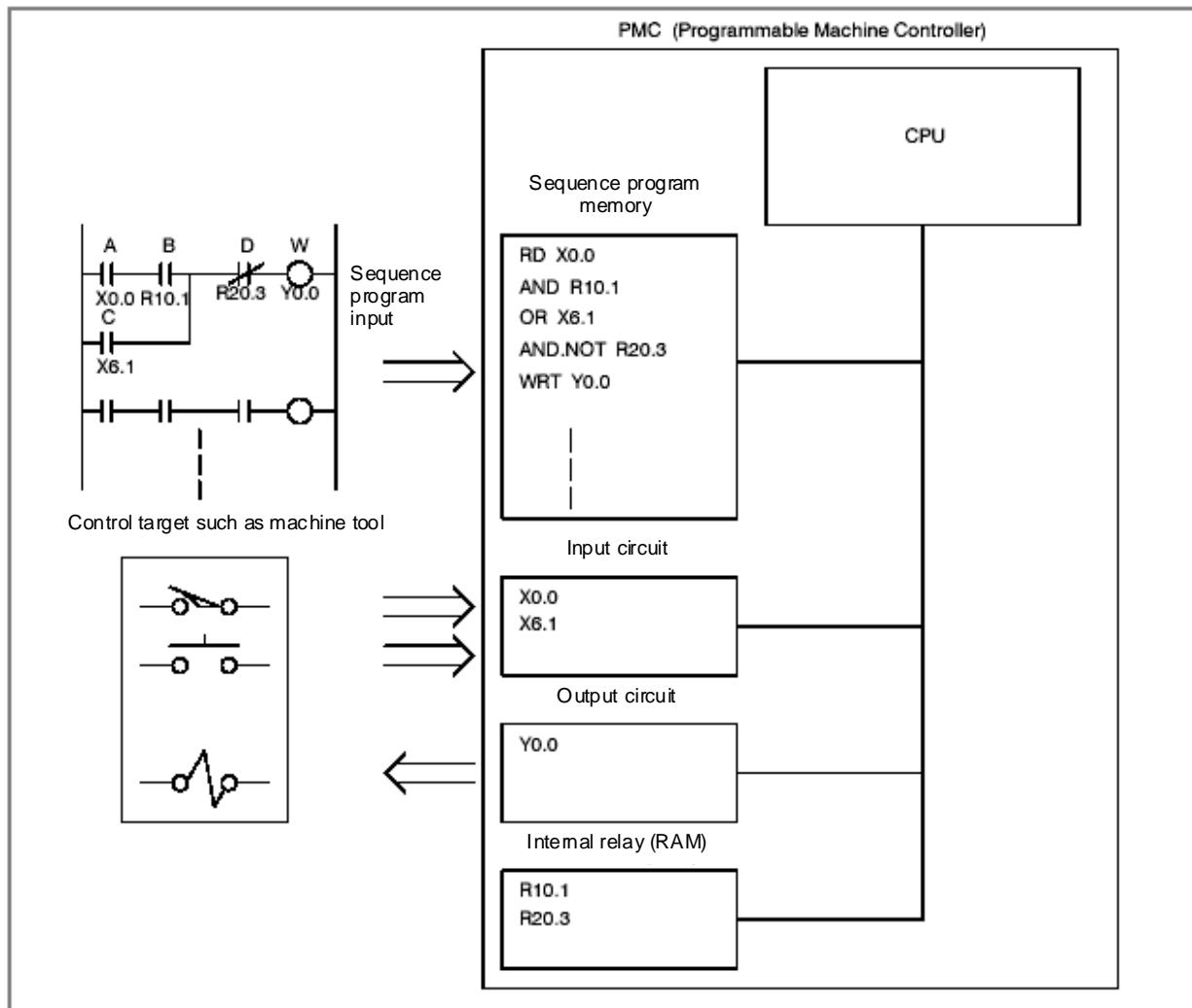


Fig. 1.4 Sequence program execution by PMC

The RD instruction causes the CPU to read the signal of the input circuit at address X0.0 and set the read data in the operation register. Next, the AND instruction causes the CPU to AND the set data with the internal relay state at address R10.1 and set the result in the operation register. The CPU then executes the subsequent instructions at high speed, and the operation result is output to the output circuit at address Y0.0.

1.4.1 Execution Procedure of Sequence Program

In general relay sequence circuits, relays operate at exactly the same time. This means that when relay A operates in the following figure, relays D and E operate at exactly the same time (when contacts B and C are both off).

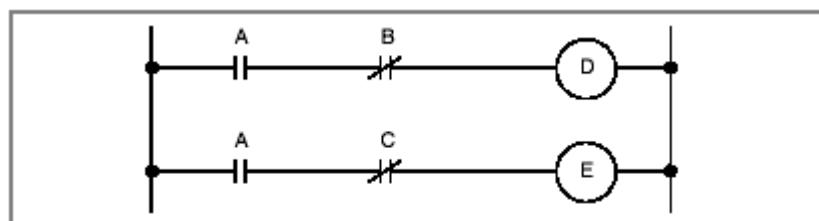


Fig 1.4.1 (a)

In PMC sequence control, on the other hand, relays in the circuit operate sequentially. When relay A in Fig. 1.4.1 (a) operates, relay D operates, then relay E operates.

Therefore, in PMC sequence control, relays operate in the order coded in the ladder diagram (the order of programming). The sequential operations in this sequence are performed at high speed, but some instructions are affected by the execution order.

Accordingly, in the ladder diagrams shown in Fig. 1.4.1 (b), there is a distinctive difference in operation between the PMC sequence and the sequence of the relay circuit.

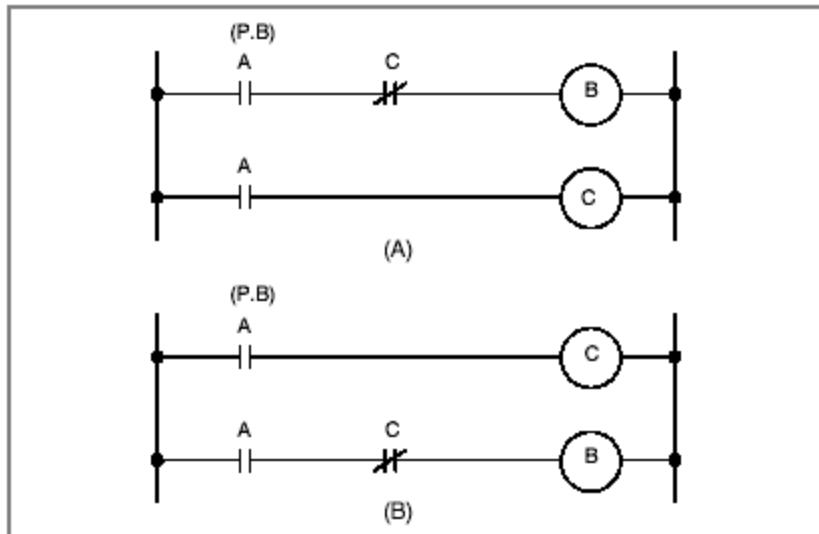


Fig. 1.4.1 (b) Circuit examples

(1) For relay sequence circuit

(A) and (B) in Fig. 1.4.1 (b) operate in the same manner. When A (P.B) is turned on, current flows through coils B and C, turning on B and C simultaneously. After C is turned on (after relay operation time), B is turned off.

(2) For PMC programming

In (A) in Fig. 1.4.1 (b), as with the relay sequence circuit, when A (P.B) is turned on, B and C are turned on, then B is turned off after a certain time elapses (after a time required for one cycle of the PMC sequence). In (B) in Fig. 1.4.1 (b), turning on A (P.B) turns on C but does not turn on B even momentarily.

1.4.2 Repetitive Operation

A sequence program is executed until the end of the ladder diagram (the end of the program) is reached, then program execution is repeated from the beginning of the ladder diagram (the beginning of the program).

The execution time from the beginning to the end of the ladder diagram (the time required for one cycle) is a time for processing the sequence program once and is called a scan.

This processing time depends on the sequence control scale (the number of steps) and the size of the 1st level sequence described below. A shorter processing time results in a better signal response in the sequence.

1.4.3 Processing Priority (1st Level, 2nd Level, and 3rd Level)

A sequence program consists of two operation parts: a high-speed sequence part called the 1st level, which is executed every several msec, and a normal sequence part called the 2nd level. When the sequence program used allows use of the 3rd level, the 3rd level sequence part is added. (See Fig. 1.4.3 (a).)

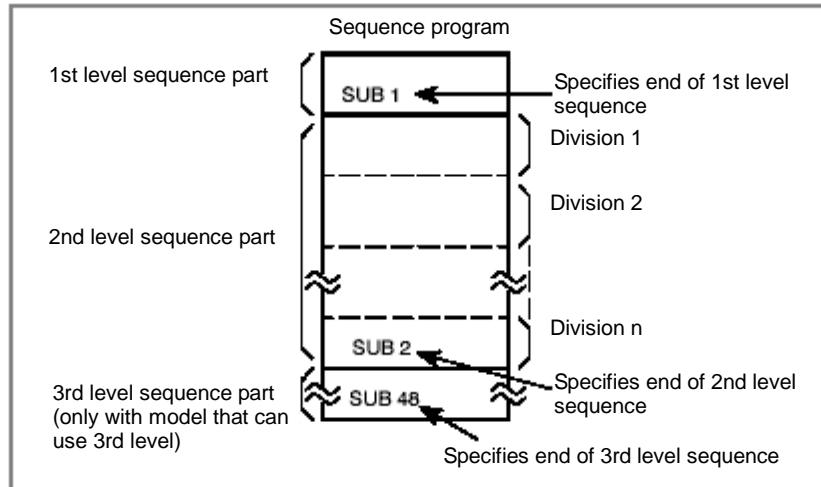


Fig. 1.4.3 (a) Sequence program structure

The 1st level sequence part is a high-speed sequence part that is executed every ladder execution cycle. The ladder execution cycle is 4 or 8 msec, which is set in a Robot parameter. If the execution of the 1st level program requires a long time, the overall execution time including the 2nd level (sequence processing time) is extended. So, the 1st level sequence part should be created so that it can be processed in a short time where possible. The 2nd level sequence part is executed every (ladder execution cycle \times n) msec (where n is the number by which the 2nd level is divided). The 3rd level sequence part is executed when the PMC is idle.

(1) Division of the 2nd level program

The 2nd level program must be divided to execute the 1st level program. The order of sequence program execution is illustrated in Fig. 1.4.3 (b), where the number of divisions is assumed to be n.

After the last division (division n) of the 2nd level program is executed, the sequence program is executed from the beginning. Therefore, when the number of divisions is n, the execution cycle of the overall sequence program is expressed as the ladder execution cycle \times n msec.

As the amount of the 1st level sequence part increases, the amount of the 2nd level sequence portion executed within the ladder execution cycle decreases. As a result, the number of divisions n increases, which increases the overall execution time including the 2nd level (sequence processing time). Therefore, the 1st level sequence program part should be minimized where possible. The division number of 2nd level may be indefinite because of changing of the working condition of functional instructions in 1st level and 2nd level.

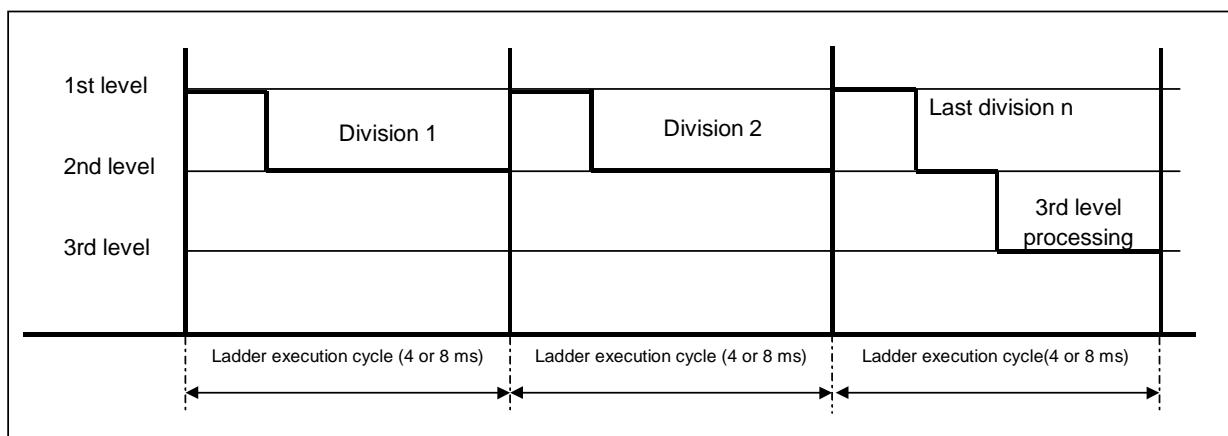


Fig. 1.4.3 (b) Sequence program execution order

(2) 1st level sequence part

High-speed sequence operation. Only high-speed sequence processing such as processing of a pulse signal with a short signal width in time is performed.

These signals include IMSTP.

(3) 3rd level sequence part

The 3rd level sequence processing is performed during the remaining time from the end of the last division (n) of the 2nd level until the 1st level processing restarts (see Fig. 1.4.3 (b)).

It is possible to program the 3rd level, but the execution cycle period of time for processing the 3rd level sequence part is not guaranteed. Therefore, the 1st and 2nd level sequence parts should be programmed without using the 3rd level sequence part.

1.4.4 Structured Sequence Program

Structured ladder coding has the following advantages:

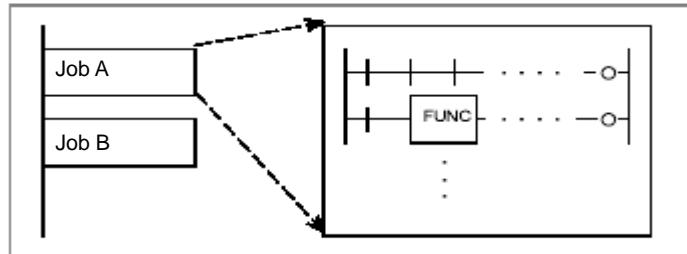
- Programming is easy to understand, therefore programming becomes easier.
- Program errors can be found easily.
- Troubleshooting can be done easily.

1.4.4.1 Implementation

Three major implementation techniques are supported.

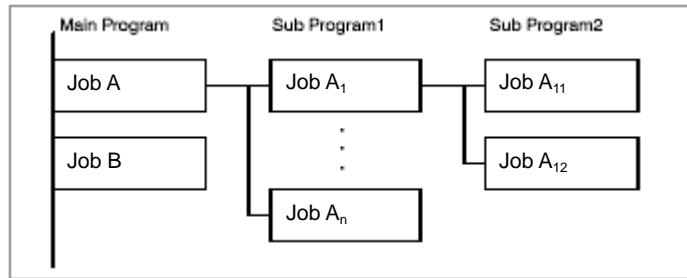
(1) Use of routines

Ladder sequence processing units are created so that they can be treated as routines.



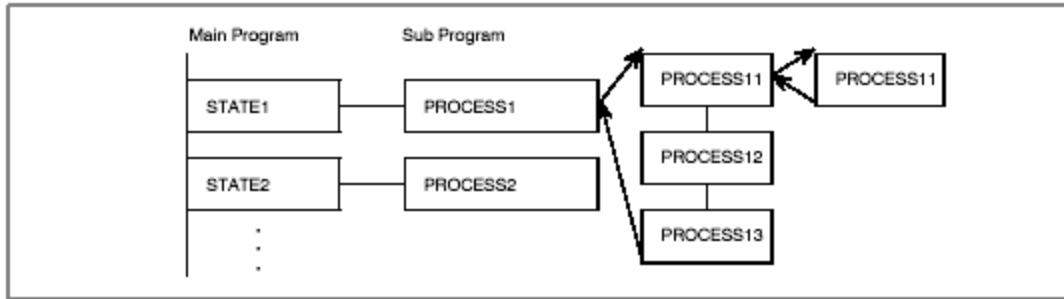
(2) Nesting

Ladder routines created in (1) are connected to configure a ladder sequence.



(3) Conditional branch

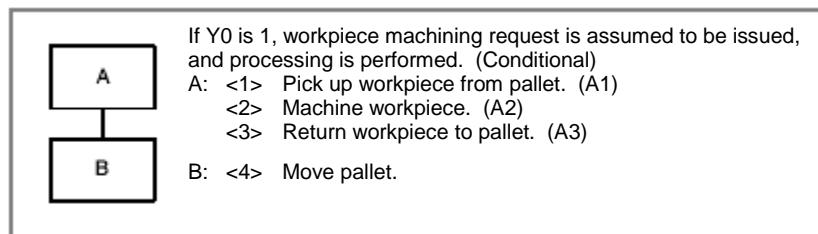
The main program loops and determines conditions. If conditions are satisfied, a subprogram process is executed. If the conditions are not satisfied, the subprogram process is skipped.



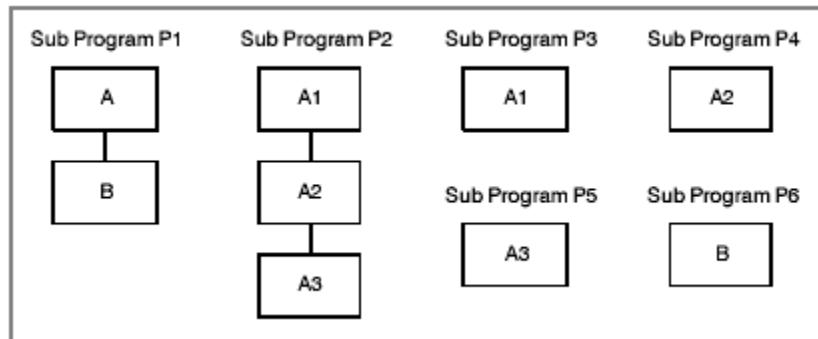
Application example

(1) Example

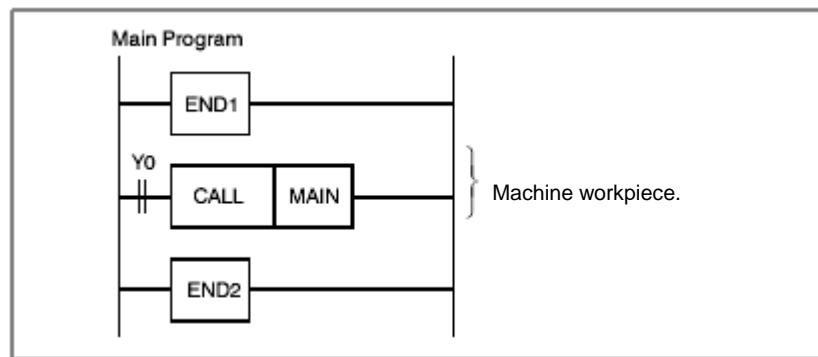
Suppose that there are four major jobs.

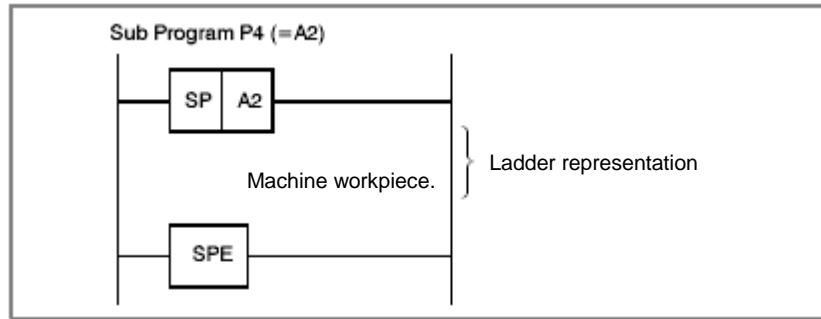
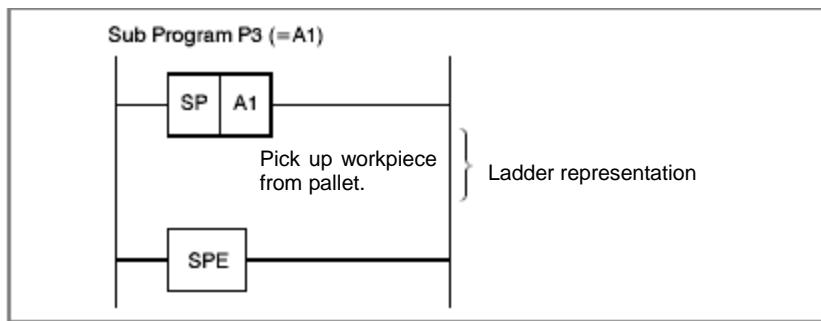
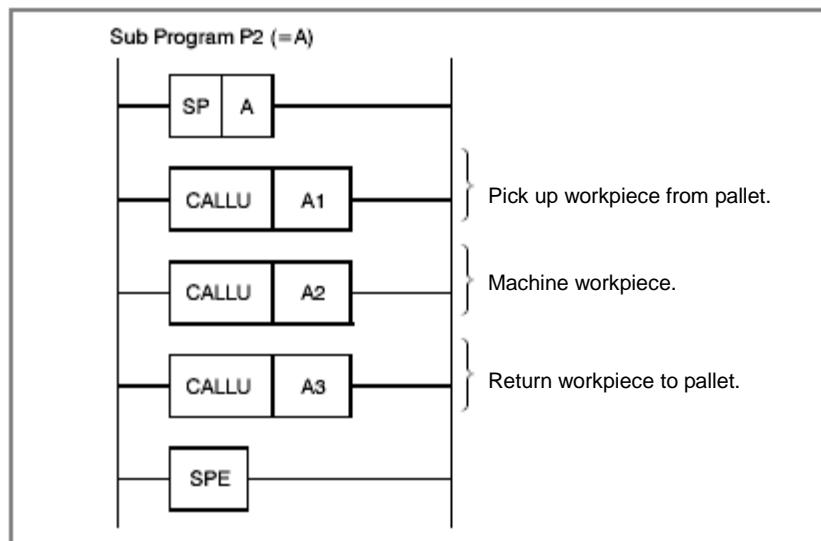
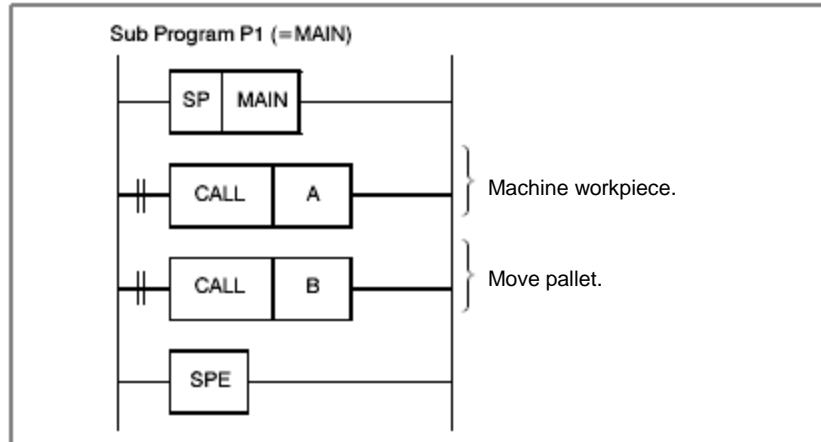


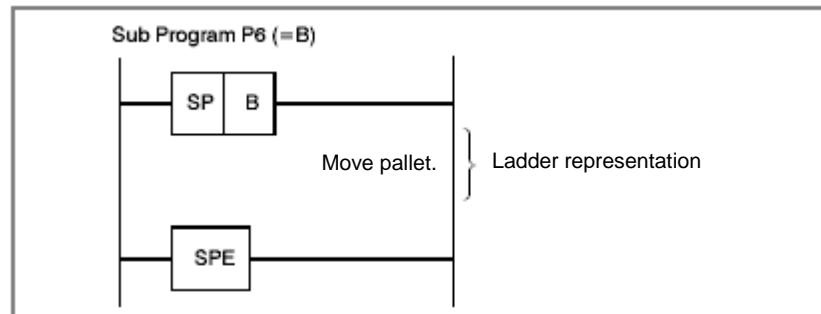
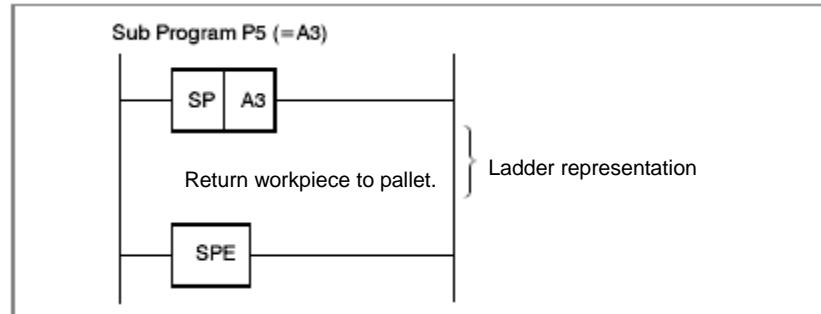
(2) Program configuration



(3) Program coding







Specifications

(1) Main program

A ladder program consisting of the 1st ladder level and 2nd ladder level is called a main program. You can create just one main program. Subprogram calls from the 1st ladder level are not allowed. Any number of subprogram calls from the 2nd ladder level may be made. Functional instructions JMP and COM must be closed within the main program and each subprogram.

(2) Subprogram

Programs called from the 2nd ladder level are referred to as subprograms. A subprogram is a program unit enclosed by functional instructions SP and SPE. Up to 512 or 5000 subprograms can be created for one PMC.

(3) Nesting

A subprogram can call another subprogram.

Up to eight levels of subprograms can be nested.

Recursive calls are not permitted.

(4) Programming order when subprograms are used

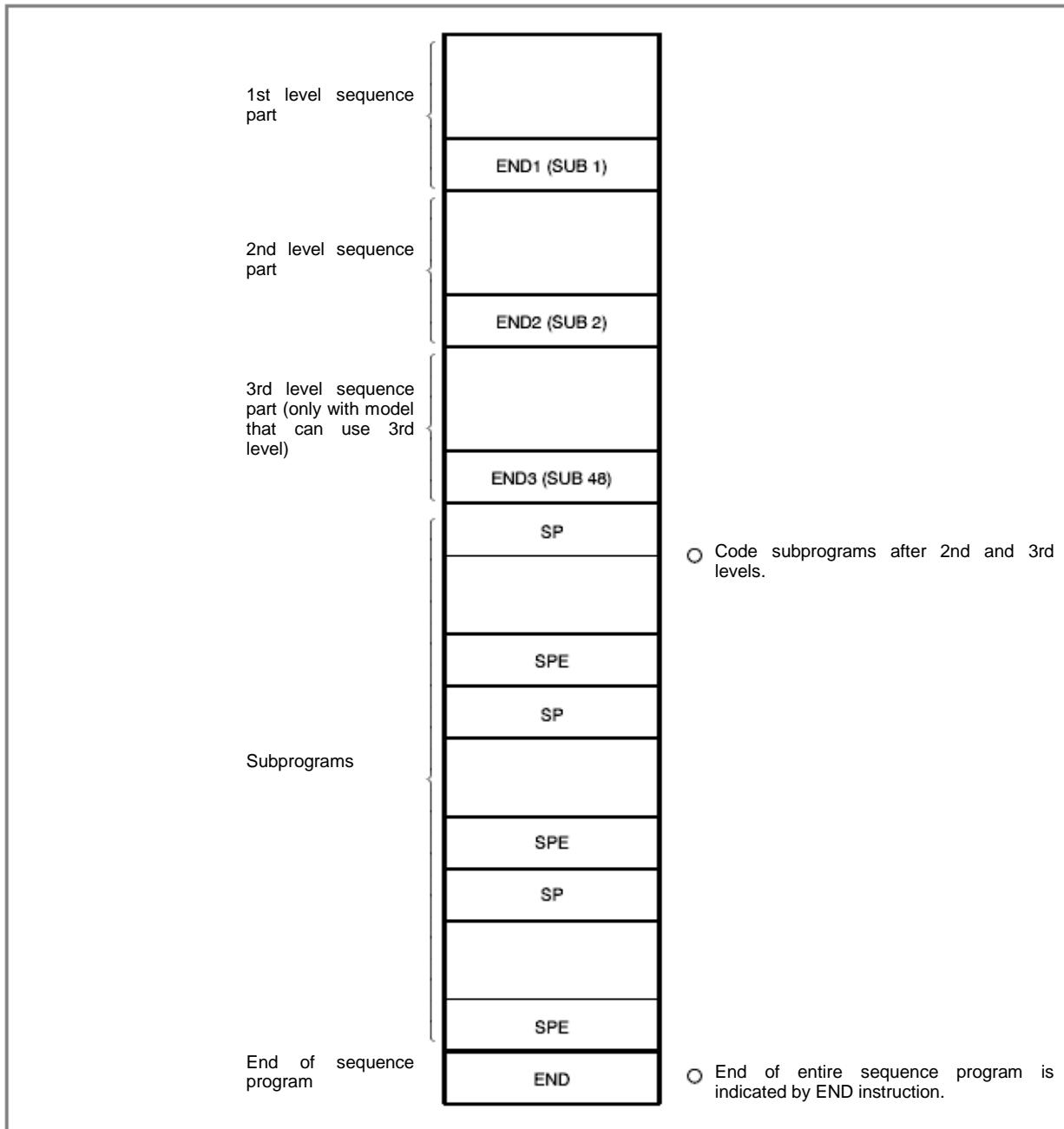


Fig. 1.4.4.1

1.4.4.2 Subprogramming and nesting**Function**

A conditional call (or unconditional call) is coded in the main program, and the name of a subprogram to be executed is specified. In the subprogram, the subprogram name and a ladder sequence to be executed are coded.

When a conditional call specifying Pn (representing a program name) is made, a subprogram named Pn is called and executed.

A subroutine name can be assigned by adding a symbol or comment to Pn.

In the example shown in Fig. 1.4.4.2, the main program calls three subprograms. These calls are all conditional calls. Subprogram P1 is named SUBPRO. Subprogram P1 calls subprogram PROCS1 unconditionally.

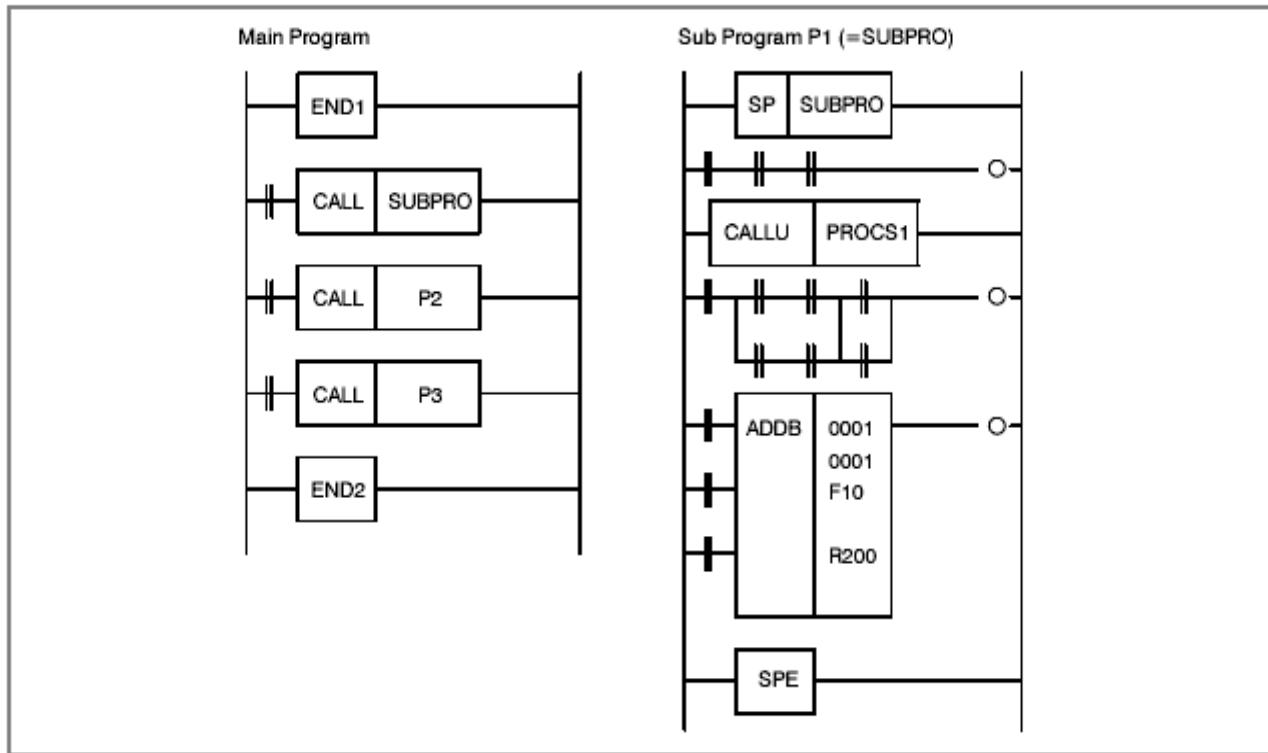
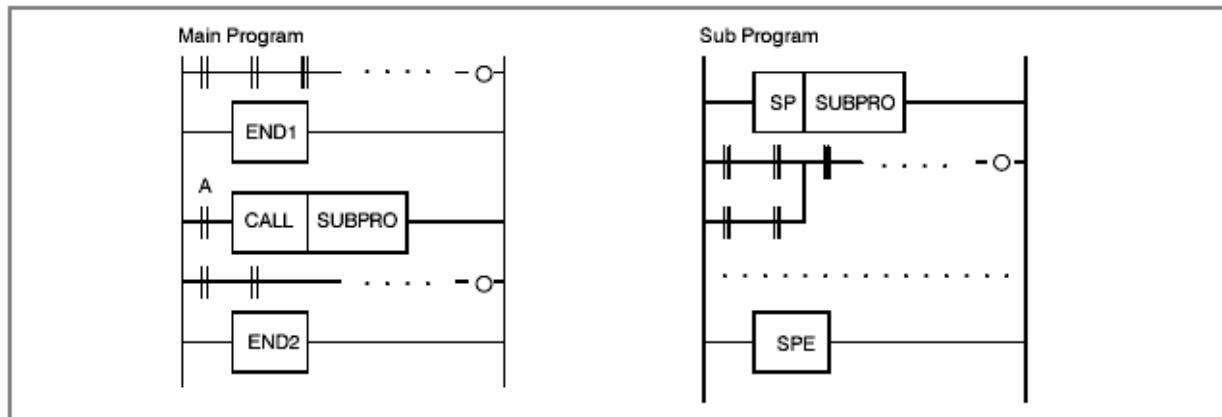
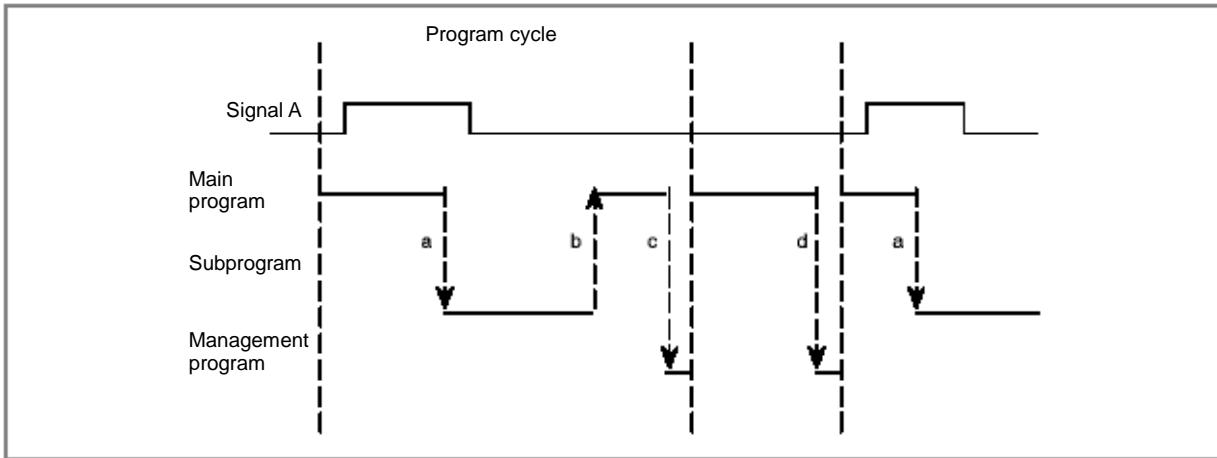


Fig. 1.4.4.2 Example of subprogramming and nesting

Execution method

The main program is always active. Subprograms are active only when called by another program. In the following example, subprogram SUBPRO is called by signal A.





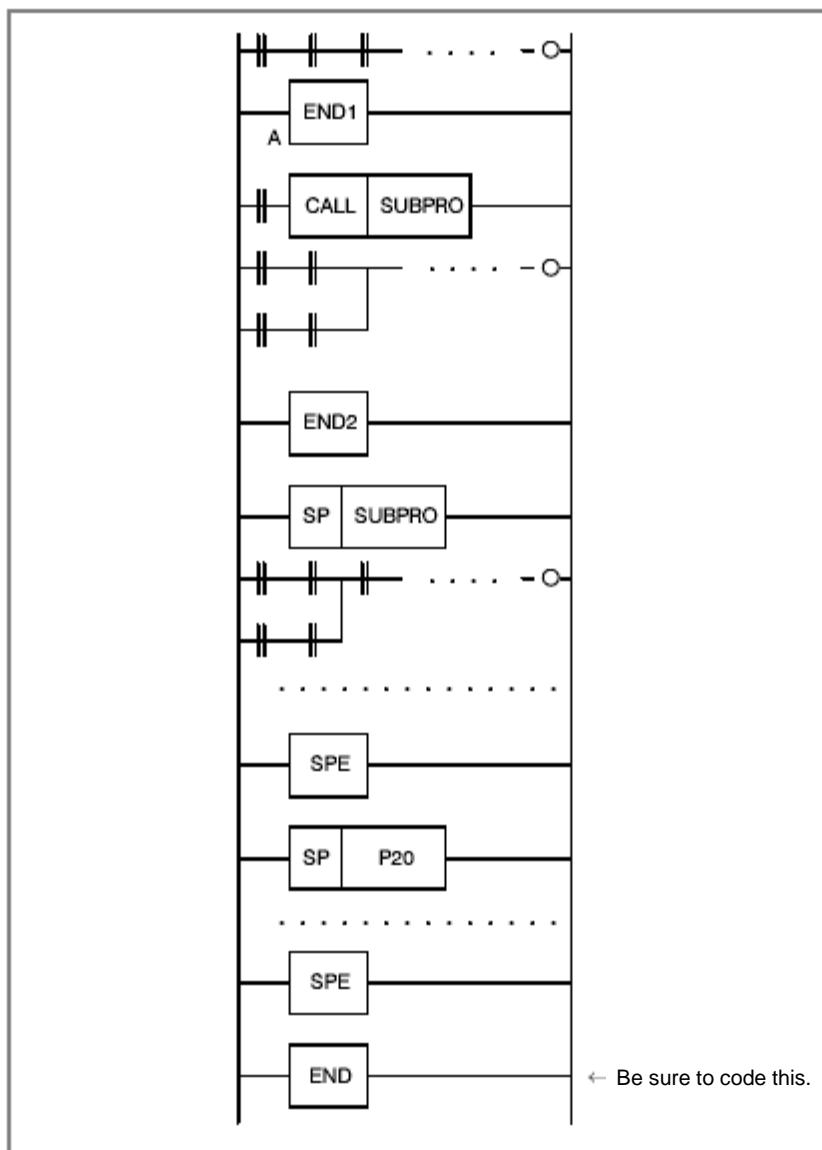
Execution flow

- (1) A subprogram call by functional instruction CALL transfers control to the subprogram.
- (2) When the execution of the subprogram is completed, control is returned to the main program.
- (3) When the execution of the main program is completed, the ladder program postprocessing is performed.

Creating a program

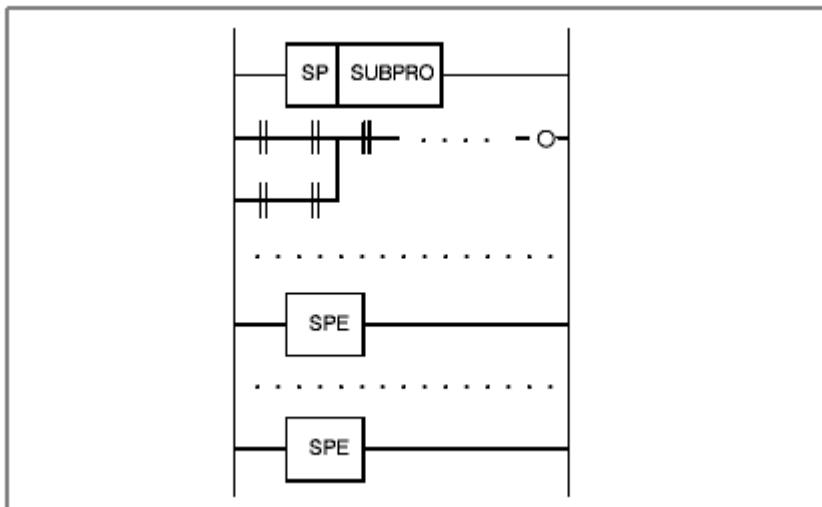
After the 1st, 2nd, and 3rd level ladder programs, create subprograms in the similar manner.

Creation example

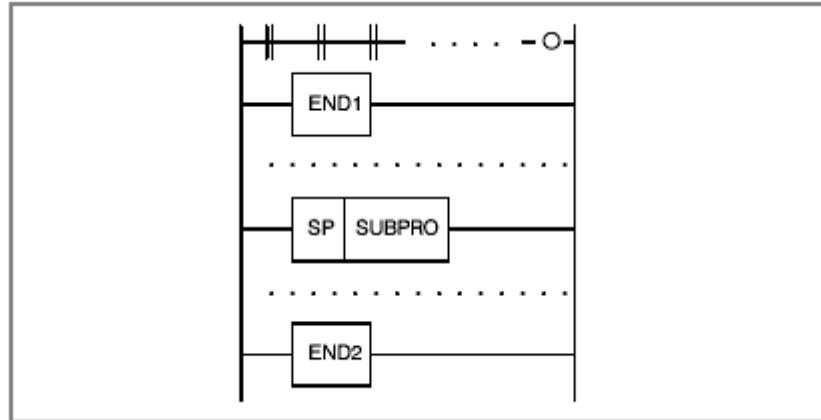


Inhibit items

- (1) Subprograms are nested.



- (2) A subprogram is created within the 1st, 2nd, or 3rd level ladder program.



1.4.5 Synchronization Processing of I/O Signals

Signals input to the PMC include input signals from the Robot and input signals from the external I/O device. Signals output from the PMC include output signals to the Robot and output signals to the external I/O device.

The relationships between these signals and the PMC are shown in Fig. 1.4.5 (a), in which input signals are input to the input memory of the PMC, and output signals are issued from the PMC.

As shown in Fig. 1.4.5 (a), the input signals are synchronized during 1 scan of the 2nd level sequence part.

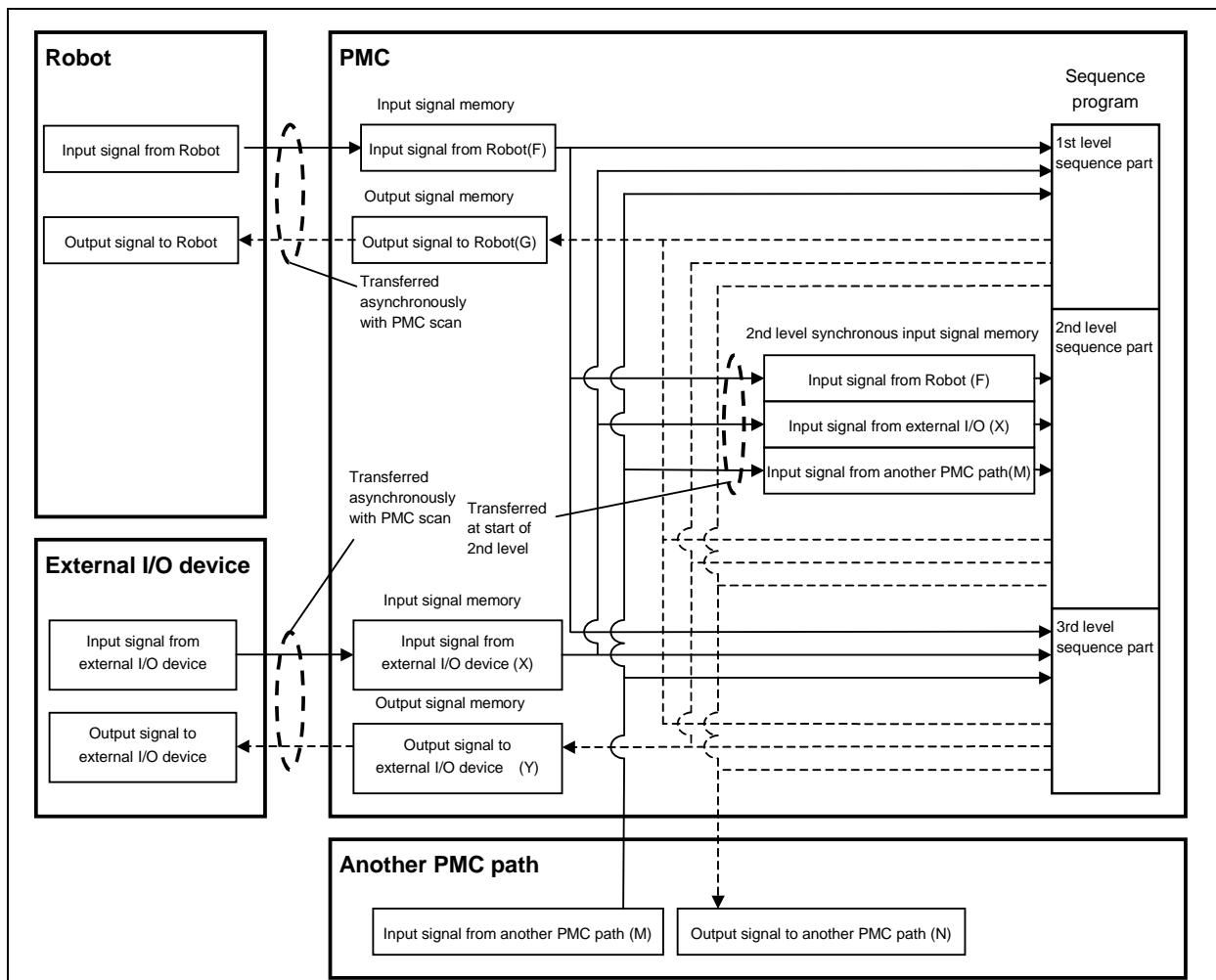


Fig. 1.4.5 (a) I/O signals of PMC

Input signal processing

(1) Input memory of the Robot

Signals input from the Robot to PMC are transferred to the input signal memory of the PMC. Since the 1st and 3rd level sequence parts directly reference and process these signals, these signals are not synchronized with input signals from the Robot.

(2) Input signals from the external I/O device

Signals input from the external I/O device are transferred to the input signal memory via the input circuit. The 1st and 3rd level sequence parts read the input signals from the input signal memory and process them.

(3) Input signal memory

The input signal memory stores signals transferred from the Robot or external I/O device.

The 1st and 3rd level sequence parts of the PMC read and process signals stored in this memory. In this case, the signal set in the input signal memory is not synchronized with the 1st and 3rd level sequence parts. For notes on asynchronous processing, see the description of following “Notes on programming asynchronous I/O signals”.

(4) 2nd level synchronous input signal memory

The 2nd level synchronous input signal memory stores signals processed by the 2nd level sequence part of the PMC. Signals synchronized with the 2nd level sequence part are set in this memory.

Input signals in the input signal memory and input signals from the Robot are automatically transferred to the 2nd level synchronous input signal memory at the beginning of the 2nd level sequence part. Therefore, the status of the 2nd level synchronous input signal memory is kept unchanged during the time from the beginning of the 2nd level sequence part until the end of the sequence part.

The programmer function automatically performs processing so that the 1st and 3rd level sequence parts use input signals in the input signal memory and input signals from the Robot while the 2nd level sequence part uses the 2nd level synchronous input signal memory. (This need not be considered during programming.)

NOTE

The 2nd level synchronous input signal memories are F, X, and M address. Other addresses are not synchronous input signals.

Output signal processing

(1) Output memory to the Robot

Signals output from the PMC to Robot are transferred from the output signal memory of the PMC to Robot.

(2) Output signals to the external I/O device

Signals output to the external I/O device are transferred from the output signal memory of the PMC to the output circuit.

(3) Output signal memory

The output signal memory is set by the sequence program of the PMC. Signals set in the output signal memory are transferred to the Robot or external I/O device asynchronous with PMC scan.

Notes on programming asynchronous I/O signals

Normal input signals from the Robot are transferred to the PMC asynchronously with PMC scan. Normal output signals to the Robot are transferred from the PMC asynchronously with PMC scan. When creating a sequence program, note that the input signals from the Robot are not synchronized with the 1st and 3rd level sequence program parts. Because the input signals from the Robot are asynchronous, the status of an input

signal from the Robot may change during execution of the 1st level sequence program part, which can lead to a problem as shown in Fig. 1.4.5 (b). To prevent such a problem, write the TF signal to an internal relay at the beginning of the 1st level sequence part so that the subsequent operation of the 1st level sequence program part references the internal relay. Then, the TF signal can be treated as a synchronous signal. See Fig. 1.4.5 (c).

Signals input from the external I/O device are also asynchronous, so these signals should be treated in a similar manner.

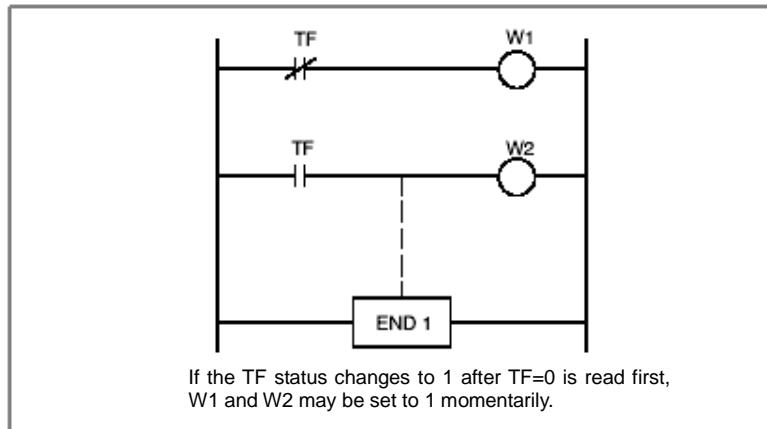


Fig. 1.4.5 (b)

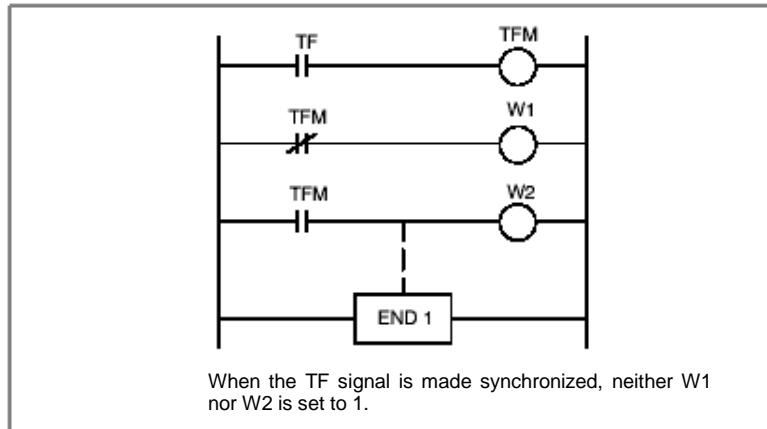


Fig. 1.4.5 (c)

Difference in signal status between 1st level and 2nd level sequence parts

The status of the same input signal may become different between the 1st and 2nd level sequence parts. The 1st level sequence part uses the input signal memory for signal processing while the 2nd level sequence part uses the 2nd level synchronous input signal memory. Therefore, it is possible that an input signal for the 2nd level sequence part lags behind the input signal for the 1st level sequence part by a cycle of the 2nd level sequence execution at the worst.

When creating a sequence program, note the following:

Signal status

- A.M On (pulse signal with short pulse width in time)
- B Off
- C On

When the 1st level is executed, the following difference can occur between Fig. 1.4.5 (d) and Fig. 1.4.5 (e):

- (1) For Fig. 1.4.5 (d)

Even when W1 = 1, W2 may not be 1. (This is because the A.M signal may differ between the 1st level and 2nd level.)

- (2) For Fig. 1.4.5 (e)

If W1 = 1, W2 is always 1.

When performing the sequence shown in Fig. 1.4.5 (d), do the following:

At the 1st level, perform the high-speed sequence processing applied when the A.M signal status changes (operating).

At the 2nd level, perform the sequence processing applied when the A.M signal status does not change (stopped).

NOTE

In the middle of 1st level processing, a signal status change may occur asynchronously with the sequence program processing. For details, see "Notes on programming asynchronous I/O signals".

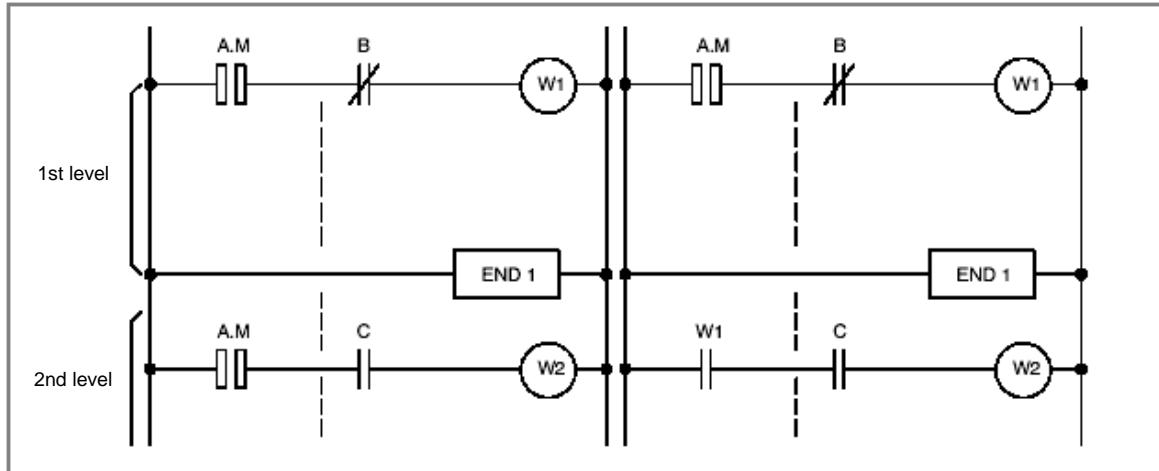


Fig. 1.4.5 (d)

Fig. 1.4.5 (e)

1.4.6 Interlock

In sequence control, considering how to provide an interlock is a key design issue from the safety point of view. Of course, an interlock must be provided by sequence programs. Furthermore, an interlock must also be provided by external emergency stop and safety fence input. Even when an interlock is provided logically by a sequence program (software), the interlock by the sequence program will not work if the hardware for executing the sequence program fails for a certain cause. Therefore, be sure to provide an interlock by external emergency stop and safety fence input to ensure safety of the operator and prevent machine damage.

1.5 MULTI-PATH PMC FUNCTION

The multi-path PMC function allows one PMC system to execute multiple sequence programs at the same time.

PMC memory for each sequence program is basically independent, and the same PMC address can be used for different purposes of the individual PMCs. Extra relays (E addresses) can be shared among PMCs as shared memory. All PMCs can read from and write to this area, so the area can be used for the interface between the PMCs. M,N addresses can be also used for the interface between the PMCs.

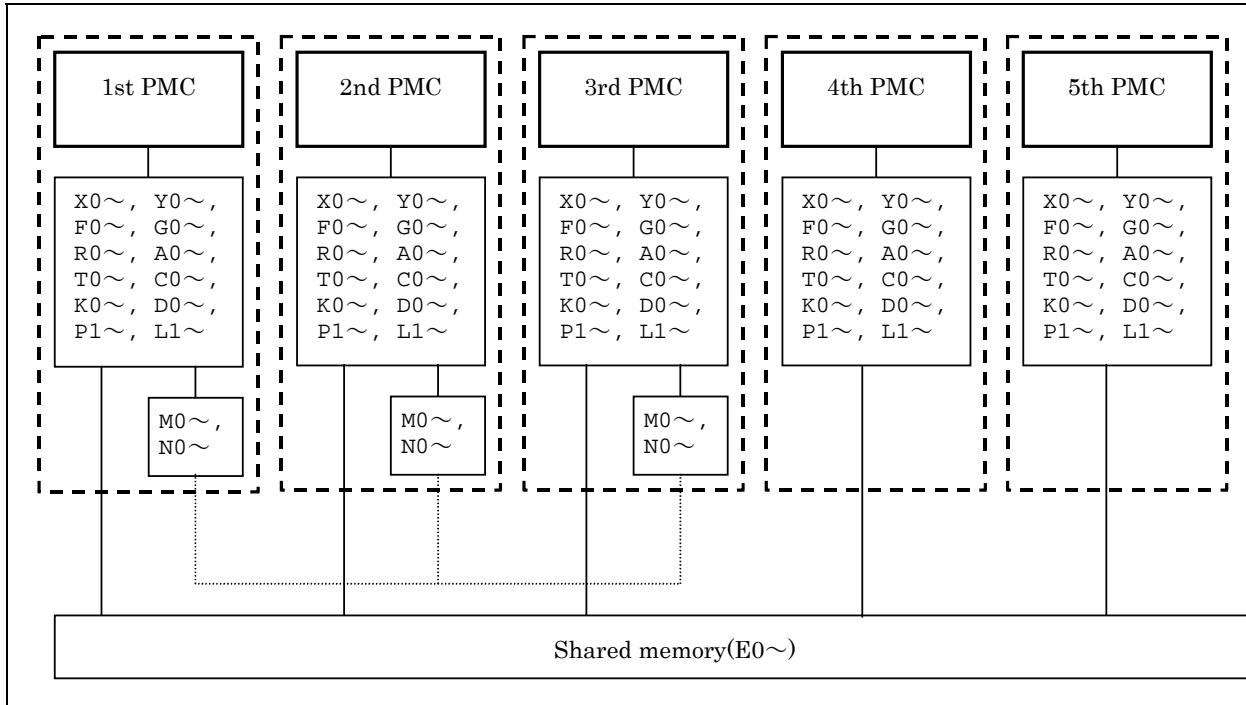


Fig. 1.5 PMC memory of multi-path PMC function

A program for each PMC is saved as an independent file and can be edited, updated, and backed up separately.

1.5.1 Execution Order and Execution Time Percentage

For the multi-path PMC function, the order of PMC execution and execution time percentages of the PMCs can be set with PMC setup menu.

Execution order

If parameters related to the execution order are not set (0 is set), the following order sequence is assumed by default:

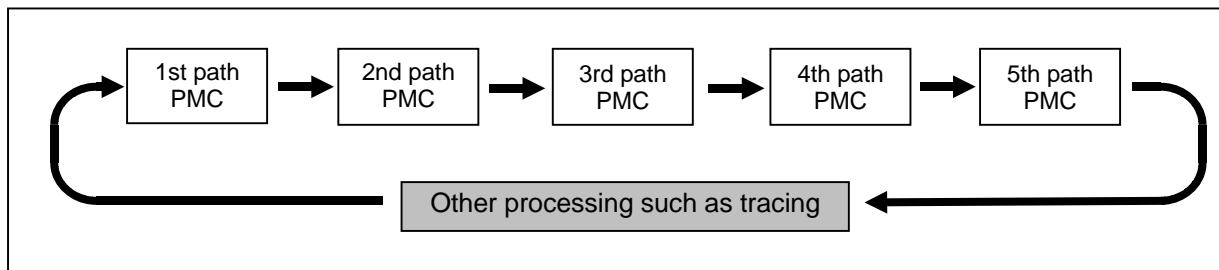


Fig. 1.5.1 (a) Default execution order of multiple PMCs

Execution time percentage

If parameters related to execution time percentages are not set, execution time percentages of first path PMC is assumed 100%. Please change execution time percentages of other path PMC if they are used.

An example of changing the execution order and execution time percentages by setting in PMC setup menu is explained below. In the following, sequence programs are executed in the order from the third PMC to the first PMC to the second PMC with the execution time percentage of the third PMC set to 30%, the percentage of the first PMC to 50%, and the percentage of the second PMC to 20%:

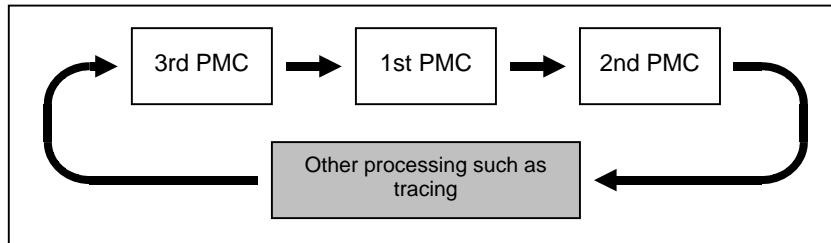


Fig. 1.5.1 (b) Example of setting execution order of multiple PMCs

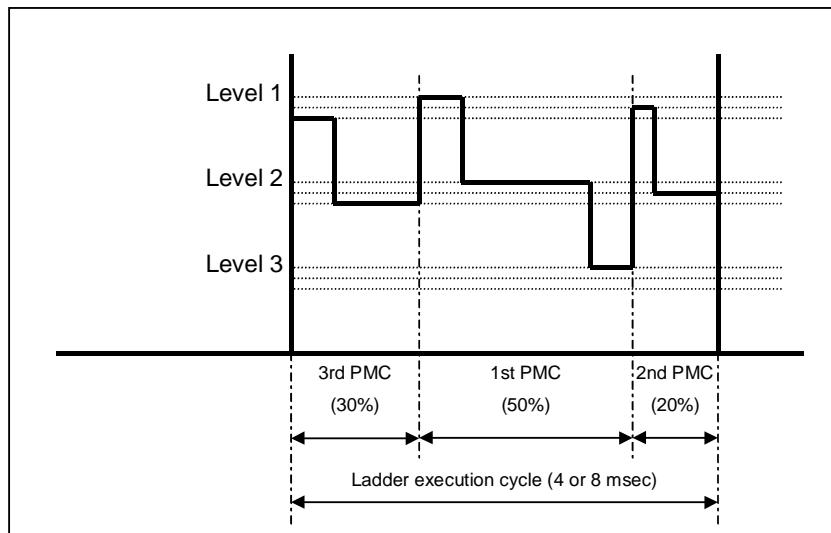


Fig. 1.5.1 (c) Example of setting execution time percentages of multiple PMCs

1.5.2 Multi-Path PMC Interface

The multi-path PMC interface is the communication means between two PMC paths.

Generally, Each path of multi-path PMC system has individual PMC memory space except E address. And, E address can be used to share data of multi-path PMC system. However, this method has a risk that the memory is over written by other PMC path inappropriately.

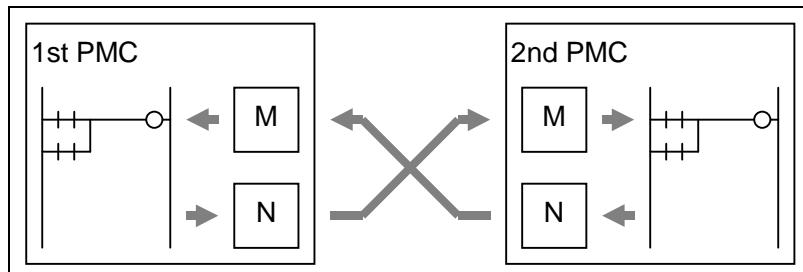
NOTE

This interface cannot be used in 4th-path PMC and 5th-path PMC.

When using this function, the input and output signals of each path become definitely. Therefore, you can send or receive the data between two PMC paths safely.

When you output data to N address in one of PMC paths, it can be referenced by M address in the other PMC path.

Ex.) When using this function with 1st PMC and 2nd PMC :



Moreover, signals of M address are synchronized during 1 scan of 2nd level program. Therefore, you can reference the same signal status on the first step and the last step of level2 program, like as X and F address.

Setting for two PMC paths is done in PMC setup menu.

⚠ WARNING

E address can be used to share data of multi-path PMC system. However, E addresses are not synchronized during 1 scan of 2nd level program. So, the memory may change during execution of 2nd level program. You must take care that the memory is not overwritten by other PMC path in multi-path PMC system.

1.5.3 Common PMC Memory Mode of Multi-Path PMC

When using the Common PMC Memory mode, a program that controls a related process can be divided to multi-path Sequence Programs.

And, those Sequence Programs can be inputted/outputted, edited and saved independently.

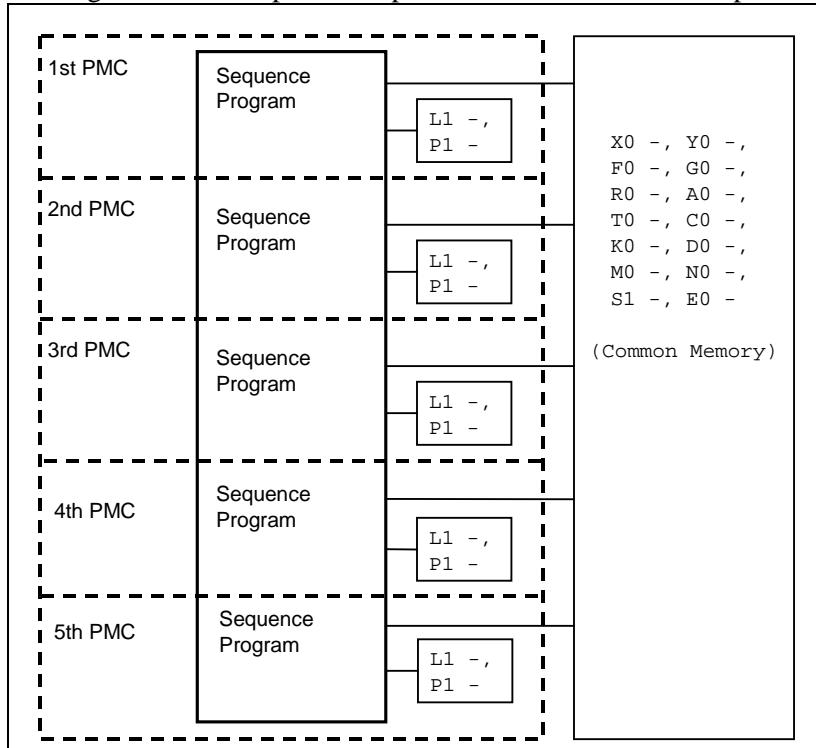


Fig. 1.5.3 (a) Configuration of the Common PMC Memory mode

The Common PMC Memory mode is enabled in PMC setup menu.

There are differences in the following specifications in the Independent PMC Memory mode and the Common PMC Memory mode.

Table 1.5.3 (a) Comparison of Independent PMC Memory mode and Common PMC Memory mode

Data and Functions		Independent PMC Memory mode	Common PMC Memory mode
Sequence program	Ladder (TMR, CTR, CTRB)	each PMC path	each PMC path
	Title	each PMC path	shared by all PMC paths
	Symbol & Comment	each PMC path	each PMC path
	Message data	each PMC path	each PMC path
	System parameter (Counter data type)	each PMC path	each PMC path
	- Inputting/Outputting	each PMC path	1st-path PMC is effective
	- Programmer protection	each PMC path	each PMC path
			1st-path PMC is effective

Data and Functions		Independent PMC Memory mode	Common PMC Memory mode
PMC Parameter			
PMC Parameter	Timer	each PMC path	shared by all PMC paths
	Counter	each PMC path	shared by all PMC paths
	Keep Relay	each PMC path	shared by all PMC paths
	Data Table	each PMC path	shared by all PMC paths
	Data Table control data	each PMC path	shared by all PMC paths
	Setting Parameter	each PMC path	shared by all PMC paths
	- Inputting/Outputting	each PMC path	1st-path PMC is effective
	- Programmer protection	each PMC path	1st-path PMC is effective

When using the Common PMC Memory mode, execution order of PMC sequence programs is different than usual. In addition, settings of execution time parentage are not used for common PMC path. Fig.1.5.3 (b) is example of execution order, when PMC path number is 3 , execution time is 30%, 50% and 20%. Sequence is start from Level1 sequence of all PMC path and settings of execution time parentage are not used.

Execution order:

Independent memory: 1st PMC Level1 -> 1st PMC Level2 -> 1st PMC Level3 ->
-> 2nd PMC Level1 -> 2nd PMC Level2 -> 3rd PMC Level1 -> 3rd PMC Level2

Common memory: 1st PMC Level1 -> 2nd PMC Level1 -> 3rd PMC Level1 ->
-> 1st PMC Level2 -> 2nd PMC Level2 -> 3rd PMC Level2 -> 1st PMC Level3

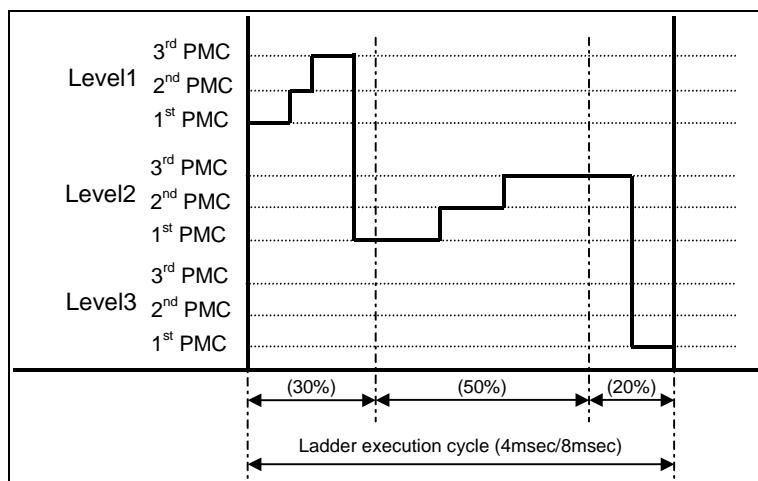


Fig. 1.5.3 (b) Execution order for common PMC memory mode

WARNING

- 1 Please separate the range of PMC Memory that will be written by each PMC path. And, don't write to the same address from other PMC paths because it will often cause a problem. When making such programs it will be difficult to debug and fix problems.

⚠ WARNING

- 2 When using the Common PMC Memory mode, the memory of PMC Parameter is shared by those PMC paths, too. Therefore, don't set any duplicated number of functional instructions that is used for PMC Parameter in those PMC paths.

<Functional instructions using PMC Parameter>

- TMR (Timer : SUB 3)
- CTR (Counter : SUB 5)
- CTRB (Fixed Counter : SUB 56)

But, the instruction number of the following functional instructions can be used for each PMC path, also in the Common PMC Memory mode.

<Functional instructions numbered each PMC path>

- TMRB (Fixed Timer : SUB 24)
- TMRBF (Off Delay Fixed Timer : SUB 77)
- DIFU (Rising Edge Detection : SUB 57)
- DIFD (Falling Edge Detection : SUB 58)

NOTE

- 1 To use the Common PMC Memory mode in the 2nd to 5th-path PMC, select the same PMC type as 1st-path PMC on FANUC LADDER-III for Robot.
- 2 Data Table Control data is shared between PMC paths that use the Common PMC Memory mode.
- 3 L and P addresses, that are used for the labels of jump or sub-program calls, can be used for each PMC path independently in the Common PMC Memory mode.

1.6 SAVE/LOAD PMC RELATED DATA

The sequence program, PMC parameters and the setting of PMC menu in the robot controller can be saved as files to an external file device. PMC parameter is the generic name of internal relays and Data Table Control Data that are retained when the controller is turned off.

The sequence program, PMC parameters and the setting in PMC menu can be recovered by loading the saved files.

The sequence program, PMC parameters and the setting in PMC menu can be copied by loading the files that are saved from another robot.

File names and contents

File names and contents of the files.

File name	Contents	Condition to load
LADDER1.PMC LADDER2.PMC LADDER3.PMC LADDER4.PMC LADDER5.PMC	Sequence program of each PMC path. (The number in the file name is the path number.) Title data Ladder Symbols Coil comments	For 7DC1 or 7DD0 software, In Controlled start menu, or E-STOP on Teach Pendant or Operator Panel is pressed.
LADDERS.PMC	Sequence program of Safety PMC. Title data Ladder Symbols Coil comments	

File name	Contents	Condition to load
PARAM1.PMC PARAM2.PMC PARAM3.PMC PARAM4.PMC PARAM5.PMC	PMC parameters of each PMC path. (The number in the file name is the path number.) Value of internal relays that are retained at power off K(Keep relay) D(Data Table) T(Variable Timer) C(Counter) Data Table Control Data	In Controlled start menu or PMC is stopped.
PARAMS.PMC	PMC parameters of Safety PMC path. K900.0, K900.1, K900.4 and K900.7	
PMCCFG.SV	The setting of PMC menu. PMC external I/O assignment PMC internal I/O assignment Multi-PMC setting	In Controlled start menu.

NOTE

PARAM*.PMC of PMC path over maximum PMC path number are not saved.
When Multi-Path PMC option (A05B-2500-J763) is not loaded, the maximum PMC path number is 1.
LADDER*.PMC of the PMC path is not saved when the sequence program does not exits.

Operation to save files

These files can be saved with the File menu on the teach pendant.

Refer to Section 6.2, "File Menu Operations" for information on saving files from the File menu.

Operation to load file

These files can be loaded with the File menu on Teach Pendant. The "Condition to load" in the above list must be satisfied to load the files.

Refer to Section 6.2, "File Menu Operations" for information on loading files from the File menu.

2 PMC SPECIFICATIONS

The specification of DCS Safety PMC is not specified in this chapter. Please refer to “FANUC Robot Series R-30iB/ R-30iB Mate Controller Dual Check Safety Operator’s Manual (B-83184EN)” about the specification of DCS Safety PMC.

2.1 SPECIFICATIONS

2.1.1 Basic Specifications

Table 2.1.1 (a) Basic specifications of each PMC path

Function	1st ~ 5th- path PMC
PMC Memory Type(Note1)	1st PMC PMC Memory-B PMC Memory-C PMC Memory-D 2nd~5th PMC PMC Memory-A PMC Memory-B PMC Memory-C Common PMC Memory with 1st PMC
Programming language	Ladder Step sequence(Note2) Function block
Number of ladder levels	3
Level 1 execution period (Note3)	4 or 8 msec
Processing power <ul style="list-style-type: none"> • Basic instruction processing speed(transition contact) (Note4) • Basic instruction processing speed(Positive/Negative transition contact) 	9.1 nsec/step 310 nsec/step
Program capacity (Note5) <ul style="list-style-type: none"> • Ladder • Symbol & Comment • Message 	Up to about 300,000 steps At least 1KB At least 8KB
Instructions <ul style="list-style-type: none"> • Basic instructions • Functional instructions (Note6) 	24 211 (230)
Instructions(When the expanded PMC ladder instruction function is invalid) <ul style="list-style-type: none"> • Basic instructions • Functional instructions (Note6) 	14 86 (105)
Internal I/O interface <ul style="list-style-type: none"> • Inputs (F) • Outputs (G) 	768 bytes × 2(Note7) 768 bytes × 2(Note7)
External I/O interface <ul style="list-style-type: none"> • Inputs (X) • Outputs (Y) 	128 bytes × 5 128 bytes × 5
Symbol & Comment (Note8) <ul style="list-style-type: none"> • Number of symbol characters • Number of comment characters (Note9) 	40 255
Program storage area (Flash ROM)	Max. 3MB (total size of sequence program of all PMC paths)

NOTE

- 1 As for the setting the PMC memory type, see subsection 2.1.3.
- 2 The Step Sequence is unavailable in 2nd to 5th PMC.
- 3 PMC setup menu is used to specify a level-1 execution period. Note, however, that it is impossible to specify a level-1 execution period for each PMC separately.
- 4 It is the processing speed of contact other than Positive/Negative transition contact.
- 5 The maximum overall program size (including the maximum number of ladder steps, symbols/ comments, and messages) varies depending on option settings. See subsection 2.1.4 and 2.1.4 for details.
- 6 For the number of functional instructions, each parenthesized number indicates the number of all functional instructions, and each non-parenthesized number, the number of valid functional instructions.
- 7 In 2nd-5th path PMC, 768 bytes × 1.
- 8 These are the number for extended symbol and comment character. The number of basic symbol character is 16 and the number of comment character is 30.
- 9 This number is the number of single-byte characters. When you use double-byte characters as a comment, the number becomes half.

Table 2.1.1 (b) Basic specifications of each PMC Memory Type

Function	1st to 5th PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
PMC Memory				
• Internal relay (R)	1,500 bytes	8,000 bytes	16,000 bytes	60,000 bytes
• System Relay (R9000 or Z)	500 bytes	500 bytes	500 bytes	500 bytes
• Extra relay (E)(Note1)	10,000 bytes	10,000 bytes	10,000 bytes	10,000 bytes
• Message display (A)				
· Display requests	2,000 points	2,000 points	4,000 points	6,000 points
· Status displays	2,000 points	2,000 points	4,000 points	6,000 points
• Nonvolatile memory				
• Timer (T)				
· Variable timer	80 bytes (40 pieces)	500 bytes (250 pieces)	1,000 bytes (500 pieces)	1,000 bytes (500 pieces)
· Variable timer precision	80 bytes (40 pieces)	500 bytes (250 pieces)	1,000 bytes (500 pieces)	1,000 bytes (500 pieces)
• Counter (C)				
· Variable counter	80 bytes (20 pieces)	400 bytes (100 pieces)	800 bytes (200 pieces)	1200 bytes (300 pieces)
· Fixed counter	40 bytes (20 pieces)	200 bytes (100 pieces)	400 bytes (200 pieces)	600 bytes (300 pieces)
• Keep relay (K)				
· User area	20 bytes	100 bytes	200 bytes	300 bytes
· System area	100 bytes	100 bytes	100 bytes	100 bytes
• Data table (D)	3,000 bytes	10,000 bytes	20,000 bytes	60,000 bytes
• Step sequence				
· Step number (S)	(None)	2,000 bytes	2,000 bytes	2,000 bytes
Functional instructions				
• Variable timers (TMR)	40 pieces	250 pieces	500 pieces	500 pieces
• Fixed timers (TMRB/TMRBF)	100 pieces	500 pieces	1,000 pieces	1,500 pieces
• Variable counters (CTR)	20 pieces	100 pieces	200 pieces	300 pieces
• Fixed counters (CTRB)	20 pieces	100 pieces	200 pieces	300 pieces

Function	1st to 5th PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
• Rising/Falling edge detection (DIFU/DIFD)	256 pieces	1,000 pieces	2,000 pieces	3,000 pieces
• Labels (LBL)	9,999 pieces	9,999 pieces	9,999 pieces	9,999 pieces
• Subprograms (SP)	512 pieces	5,000 pieces	5,000 pieces	5,000 pieces

NOTE

- 1 The extra relay is common memory for the multi-PMC function. This means that its size covers all of PMCs.

2.1.2 Determination of PMC Memory Type

PMC memory type

There are four PMC memory types i.e. memory-A, memory-B, memory-C and memory-D. These memory types differ in the size of PMC address. For the 2nd to 5th path PMC, the PMC memory can be also shared with the 1st path PMC.

PMC memory type can be set in PMC setup menu.

When multi path integrated PMC (J763) option is not ordered, PMC memory type can not be changed.

The following is the selectable PMC memory types in each PMC path.

1st path PMC	2nd to 5th path PMC	Remark
PMC-memory B (default) PMC-memory C	PMC-memory A (default) PMC-memory B PMC-memory C Shared with 1st path PMC	You can not specify PMC-memory B and C for 4th and 5th path PMC.
PMC-memory D	Shared with 1st path PMC	

2.1.3 Program Capacity

All of the memory size, to save the sequence programs for all PMC paths, is specified depending on the following options.

Option	Total steps	Memory size
Without "Multi path integrated PMC(J763)"	24,000 steps	256KB
With "Multi path integrated PMC(J763)"	300,000 steps	3MB (3072KB)

Default memory size of each PMC path is 256KB (24,000 steps) even though Multi path integrated PMC option is ordered. To use larger size of sequence program, please change the memory size in PMC setup menu. The unit of memory size is 128KB. Total size of memory size must be 3MB (3072KB) or less.

Example of configuration

- The sequence program of the 1st PMC: Ladder 48,000 steps, Memory size 640KB
- The sequence program of the 2nd PMC: Ladder 32,000 steps, Memory size 384KB
- The sequence program of the 3rd PMC: Ladder 16,000 steps, Memory size 128KB

The "Multi-path integrated PMC" option is required to be above configuration. Please change the memory size of 1st path to 640KB, and the memory size of 2nd path to 384KB in PMC setup menu.

NOTE

- 1 When the total size is exceed specified memory capacity by options, the alarm occurs in the PMC path in which detected the error.
- 2 When free memory of robot controller is not enough, the specified memory size can not be used. In this case, the alarm occurs.

2.1.4 Used Memory Size of Sequence Program

The following table lists the memory capacity used by sequence programs. When creating sequence programs, keep the total size within this memory capacity.

Table 2.1.4 Used memory size for each data

Category	Item	Required memory size (Note 1)
Ladder (Note 2)	Basic instruction	Please refer to table 2.1.6.
	Functional instruction	Please refer to table 2.1.7 and table 2.1.8.
	Functional instruction parameter	4 bytes
Symbol/comment conventional type (Note 2)	One definition of symbol/comment (Including symbol string)	24 bytes
	One comment character	1 byte (Note 3)
Symbol/comment extended type (Note 2)	One definition of symbol/comment	16 - 23 bytes (Note 5)
	One symbol character	1 byte
	One comment character	1 byte (Note 3)
	One sub-program	8 bytes (Note 6)
Message (Note 2)	One message character (alphanumeric characters)	1 byte (Note 4)
Others	Area used by the system	About 16K bytes (PMC Memory-A, B)
		About 24K bytes (Note 7) (PMC Memory-C)
		About 32K bytes (Note 7) (PMC Memory-D)

NOTE

- 1 The total sequence program size (including all items such as ladders, symbols/comments, and messages) cannot exceed the sequence program memory storage capacity. If a ladder, symbol/ comment, or message is large, the size of other categories may be limited.
- 2 The PMC programmer may adjust arrangement of these items in the sequence program memory to improve processing efficiency. As a result, up to 1K byte (1024 bytes) may be added to the sum of the sizes of individual items.
- 3 Each full-size character takes a memory capacity of 2 bytes.
- 4 Message is not used.
- 5 One definition of extended symbol and comment takes 16-23 bytes plus the memory according to the length of symbol and comment.
- 6 8 bytes are taken for a sub-program when local symbols are defined in the sub-program.
- 7 In the PMC Memory-C, the system area is expanded by about 8KB from PMC Memory-A or B. In the PMC Memory-D, the area is expanded by about 16KB from PMC Memory-A or B. Therefore, available memory size for Symbol, Comment and Message data is smaller than PMC Memory-A and B. If the program overflowed by converting PMC Memory Type, please decrease the Symbol, Comment or Message data, or change the memory size to larger size.

2.1.5 PMC Addresses

Table 2.1.5 (a) PMC Address list (1)

Signals	Symbol	1st to 5th path PMC			
		PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Input signal to the PMC from the machine	X	X0 ~ X127 X200 ~ X327 X400 ~ X527 X600 ~ X727 X1000 ~ X1127	X0 ~ X127 X200 ~ X327 X400 ~ X527 X600 ~ X727 X1000 ~ X1127	X0 ~ X127 X200 ~ X327 X400 ~ X527 X600 ~ X727 X1000 ~ X1127	X0 ~ X127 X200 ~ X327 X400 ~ X527 X600 ~ X727 X1000 ~ X1127
Output signal from the PMC to the machine	Y	Y0 ~ Y127 Y200 ~ Y327 Y400 ~ Y527 Y600 ~ Y727 Y1000 ~ Y1127	Y0 ~ Y127 Y200 ~ Y327 Y400 ~ Y527 Y600 ~ Y727 Y1000 ~ Y1127	Y0 ~ Y127 Y200 ~ Y327 Y400 ~ Y527 Y600 ~ Y727 Y1000 ~ Y1127	Y0 ~ Y127 Y200 ~ Y327 Y400 ~ Y527 Y600 ~ Y727 Y1000 ~ Y1127
Input signal to the PMC from the Robot	F	F0 ~ F767 F1000 ~ F1767(Note 2)			
Output signal from the PMC to the Robot	G	G0 ~ G767 G1000 ~ G1767(Note 1)			
Input signal from other PMC path	M	M0 ~ M767	M0 ~ M767	M0 ~ M767	M0 ~ M767
Output signal to other PMC path	N	N0 ~ N767	N0 ~ N767	N0 ~ N767	N0 ~ N767
Internal relay	R	R0 ~ R1499	R0 ~ R7999	R0 ~ R15999	R0 ~ R59999
System relay	R / Z	R9000 ~ R9499	R9000 ~ R9499	Z0 ~ Z499	Z0 ~ Z499
Extra relay	E	E0 ~ E9999 (Note 2)			
Message display · Display request · Display status	A	A0 ~ A249 A9000 ~ A9249	A0 ~ A249 A9000 ~ A9249	A0 ~ A499 A9000 ~ A9499	A0 ~ A749 A9000 ~ A9749

Table 2.1.5 (b) PMC Address list (2)

Signals	Symbol	1st to 5th path PMC			
		PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Timer · Variable timer · Variable timer precision (Note 3)	T	T0 ~ T79 T9000 ~ T9499	T0 ~ T499 T9000 ~ T9499	T0 ~ T999 T9000 ~ T9999	T0 ~ T999 T9000 ~ T9999
Counter · Variable counter · Fixed counter	C	C0 ~ C79 C5000 ~ C5199	C0 ~ C399 C5000 ~ C5199	C0 ~ C799 C5000 ~ C5399	C0 ~ C1199 C5000 ~ C5599
Keep relay · User area · System area	K	K0 ~ K19 K900 ~ K999	K0 ~ K99 K900 ~ K999	K0 ~ K199 K900 ~ K999	K0 ~ K299 K900 ~ K999
Data table	D	D0 ~ D2999	D0 ~ D9999	D0 ~ D19999	D0 ~ D59999
Label	L	L1 ~ L9999	L1 ~ L9999	L1 ~ L9999	L1 ~ L9999
Subprogram	P	P1 ~ P512	P1 ~ P5000	P1 ~ P5000	P1 ~ P5000
Step number (Step sequence)	S	(none)	S1 ~ S2000	S1 ~ S2000	S1 ~ S2000

NOTE

- 1 F1000-F1767 and G1000-G1767 are available only in 1st path PMC.
- 2 This area is common memory for the multi-path PMC function. Each program can write and read the same value in the area.
- 35 This area is used to specify the precision of a variable timer.
 - Don't modify the value of active timer and its precision except for writing same value.
 - Don't set the value other than the following range.
 - If above rules are violated, the behavior of the timer is not guaranteed.

The value of precision

- 0: Default (8msec or 48msec)
- 1: 1msec
- 2: 10msec
- 3: 100msec
- 4: 1sec
- 5: 1min

2.1.6 Basic Instructions

Table 2.1.6 Basic instruction list

Instruction name	Required memory size	1st to 5th path PMC
RD	4 bytes	○
RD.NOT	4 bytes	○
WRT	4 bytes	○
WRT.NOT	4 bytes	○
AND	4 bytes	○
AND.NOT	4 bytes	○
OR	4 bytes	○
OR.NOT	4 bytes	○
RD.STK	4 bytes	○
RD.NOT.STK	4 bytes	○
AND.STK	4 bytes	○
OR.STK	4 bytes	○
SET	4 bytes	○
RST	4 bytes	○
RDPT	12 bytes	●
ANDPT	12 bytes	●
ORPT	12 bytes	●
RDPT.STK	12 bytes	●
RDNT	12 bytes	●
ANDNT	12 bytes	●
ORN	12 bytes	●
RDNT.STK	12 bytes	●
PUSH	4 bytes	●
POP	4 bytes	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function ×: Unusable.)

2.1.7 Functional Instructions (Arranged in Sequence of Instruction Group)

Table 2.1.7 (a) Functional instruction list (arranged in sequence of instruction group) (1)

Instruction group		Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Timer	1	TMR	3	On-delay timer	8	○
	2	TMRB	24	Fixed on-delay timer	12	○
	3	TMRBF	77	Fixed off-delay timer	12	○
	4	TMRC	54	On-delay timer	16	○
	5	TMRST	221	Stop watch timer (1 ms accuracy)	20	●
	6	TMRSS	222	Stop watch timer (1 sec accuracy)	20	●
Counter	1	CTR	5	Counter processing	8	○
	2	CTRB	56	Counter processing	12	○
	3	CTRC	55	Counter processing	12	○
	4	CTRD	223	Counter processing (4 byte length)	12	●
Data transfer	1	MOVB	43	1-byte transfer	12	○
	2	MOVW	44	2-byte transfer	12	○
	3	MOVD	47	4-byte transfer	12	○
	4	MOVN	45	Transfer of arbitrary number of bytes	16	○
	5	MOVE	8	Data transfer after logical product	20	○
	6	MOVOR	28	Data transfer after logical sum	16	○
	7	XMOVB	35	Index modification binary data transfer	24	○
	8	XMOV	18	Index modification data transfer	20	○
	9	MOVBT	224	Bit transfer	24	●
	10	SETNB	225	Data setting (1 byte length)	20	●
	11	SETNW	226	Data setting (2 byte length)	20	●
	12	SETND	227	Data setting (4 byte length)	20	●
	13	XCHGB	228	Data exchange (1 byte length)	12	●
	14	XCHGW	229	Data exchange (2 byte length)	12	●
	15	XCHGD	230	Data exchange (4 byte length)	12	●
	16	SWAPW	231	Data swap (2 byte length)	16	●
	17	SWAPD	232	Data swap (4 byte length)	16	●
	18	DSCHB	34	Binary data search	24	○
	19	DSCH	17	Data search	20	○
Table Data	1	TBLRB	233	Reading data from table (1 byte length)	24	●
	2	TBLRW	234	Reading data from table (2 byte length)	24	●
	3	TBLRD	235	Reading data from table (4 byte length)	24	●
	4	TBLRN	236	Reading data from table (Arbitrary byte length)	28	●
	5	TBLWB	237	Writing data to table (1 byte length)	24	●
	6	TBLWW	238	Writing data to table (2 byte length)	24	●
	7	TBLWD	239	Writing data to table (4 byte length)	24	●
	8	TBLWN	240	Writing data to table (Arbitrary byte length)	28	●
	9	DSEQB	241	Searching data from table (=) (1 byte length)	28	●
	10	DSEQW	242	Searching data from table (=) (2 byte length)	28	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).
×: Unusable.)

Table 2.1.7 (b) Functional instruction list (arranged in sequence of instruction group) (2)

Instruction group	Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Table Data	11 DSEQD	243	Searching data from table (=) (4 byte length)	28	●
	12 DSNEB	244	Searching data from table (\neq) (1 byte length)	28	●
	13 DSNEW	245	Searching data from table (\neq) (2 byte length)	28	●
	14 DSNED	246	Searching data from table (\neq) (4 byte length)	28	●
	15 DSGTB	247	Searching data from table ($>$) (1 byte length)	28	●
	16 DSGTW	248	Searching data from table ($>$) (2 byte length)	28	●
	17 DSGTD	249	Searching data from table ($>$) (4 byte length)	28	●
	18 DSLTB	250	Searching data from table ($<$) (1 byte length)	28	●
	19 DSLTW	251	Searching data from table ($<$) (2 byte length)	28	●
	20 DSLTD	252	Searching data from table ($<$) (4 byte length)	28	●
	21 DSGEB	253	Searching data from table (\geq) (1 byte length)	28	●
	22 DSGEW	254	Searching data from table (\geq) (2 byte length)	28	●
	23 DSGED	255	Searching data from table (\geq) (4 byte length)	28	●
	24 DSLEB	256	Searching data from table (\leq) (1 byte length)	28	●
	25 DSLEW	257	Searching data from table (\leq) (2 byte length)	28	●
	26 DSLED	258	Searching data from table (\leq) (4 byte length)	28	●
	27 DMAXB	259	Maximum data (1 byte length)	28	●
	28 DMAXW	260	Maximum data (2 byte length)	28	●
	29 DMAXD	261	Maximum data (4 byte length)	28	●
	30 DMINB	262	Minimum data (1 byte length)	28	●
	31 DMINW	263	Minimum data (2 byte length)	28	●
	32 DMIND	264	Minimum data (4 byte length)	28	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

x: Unusable.)

Table 2.1.7 (c) Functional instruction list (arranged in sequence of instruction group) (3)

Instruction group	Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Comparison	1 EQB	200	Signed Binary Comparison (=) (1 byte length)	16	○
	2 EQW	201	Signed Binary Comparison (=) (2 byte length)	16	○
	3 EQD	202	Signed Binary Comparison (=) (4 byte length)	16	○
	4 NEB	203	Signed Binary Comparison (\neq) (1 byte length)	16	○
	5 NEW	204	Signed Binary Comparison (\neq) (2 byte length)	16	○
	6 NED	205	Signed Binary Comparison (\neq) (4 byte length)	16	○
	7 GTB	206	Signed Binary Comparison ($>$) (1 byte length)	16	○
	8 GTW	207	Signed Binary Comparison ($>$) (2 byte length)	16	○
	9 GTD	208	Signed Binary Comparison ($>$) (4 byte length)	16	○
	10 LTB	209	Signed Binary Comparison ($<$) (1 byte length)	16	○
	11 LTW	210	Signed Binary Comparison ($<$) (2 byte length)	16	○
	12 LTD	211	Signed Binary Comparison ($<$) (4 byte length)	16	○
	13 GEB	212	Signed Binary Comparison (\geq) (1 byte length)	16	○
	14 GEW	213	Signed Binary Comparison (\geq) (2 byte length)	16	○
	15 GED	214	Signed Binary Comparison (\geq) (4 byte length)	16	○
	16 LEB	215	Signed Binary Comparison (\leq) (1 byte length)	16	○
	17 LEW	216	Signed Binary Comparison (\leq) (2 byte length)	16	○
	18 LED	217	Signed Binary Comparison (\leq) (4 byte length)	16	○
	19 RNGB	218	Signed Binary Comparison (range) (1 byte length)	20	○
	20 RNGW	219	Signed Binary Comparison (range) (2 byte length)	20	○
	21 RNGD	220	Signed Binary Comparison (range) (4 byte length)	20	○
	22 COMPB	32	Comparison between binary data	20	○
	23 COMP	15	Comparison	16	○
	24 COIN	16	Coincidence check	16	○

(○: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

×: Unusable.)

Table 2.1.7 (d) Functional instruction list (arranged in sequence of instruction group) (4)

Instruction group		Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Bit operation	1	DIFU	57	Rising-edge detection	8	○
	2	DIFD	58	Falling-edge detection	8	○
	3	EOR	59	Exclusive OR	20	○
	4	AND	60	Logical AND	20	○
	5	OR	61	Logical OR	20	○
	6	NOT	62	Logical NOT	16	○
	7	PARI	11	Parity check	8	○
	8	SFT	33	Shift register	8	○
	9	EORB	265	Exclusive OR (1 byte length)	20	●
	10	EORW	266	Exclusive OR (2 byte length)	20	●
	11	EORD	267	Exclusive OR (4 byte length)	20	●
	12	ANDB	268	Logical AND (1 byte length)	20	●
	13	ANDW	269	Logical AND (2 byte length)	20	●
	14	ANDD	270	Logical AND (4 byte length)	20	●
	15	ORB	271	Logical OR (1 byte length)	20	●
	16	ORW	272	Logical OR (2 byte length)	20	●
	17	ORD	273	Logical OR (4 byte length)	20	●
	18	NOTB	274	Logical NOT (1 byte length)	16	●
	19	NOTW	275	Logical NOT (2 byte length)	16	●
	20	NOTD	276	Logical NOT (4 byte length)	16	●
	21	SHLB	277	Bit shift left (1 byte length)	20	●
	22	SHLW	278	Bit shift left (2 byte length)	20	●
	23	SHLD	279	Bit shift left (4 byte length)	20	●
	24	SHLN	280	Bit shift left (Arbitrary byte length)	24	●
	25	SHRB	281	Bit shift right (1 byte length)	20	●
	26	SHRW	282	Bit shift right (2 byte length)	20	●
	27	SHRD	283	Bit shift right (4 byte length)	20	●
	28	SHRN	284	Bit shift right (Arbitrary byte length)	24	●
	29	ROLB	285	Bit rotation left (1 byte length)	20	●
	30	ROLW	286	Bit rotation left (2 byte length)	20	●
	31	ROLD	287	Bit rotation left (4 byte length)	20	●
	32	ROLN	288	Bit rotation left (Arbitrary byte length)	24	●
	33	RORB	289	Bit rotation right (1 byte length)	20	●
	34	RORW	290	Bit rotation right (2 byte length)	20	●
	35	RORD	291	Bit rotation right (4 byte length)	20	●
	36	RORN	292	Bit rotation right (Arbitrary byte length)	24	●
	37	BSETB	293	Bit set (1 byte length)	16	●
	38	BSETW	294	Bit set (2 byte length)	16	●
	39	BSETD	295	Bit set (4 byte length)	16	●
	40	BSETN	296	Bit set (Arbitrary byte length)	20	●
	41	BRSTB	297	Bit reset (1 byte length)	16	●
	42	BRSTW	298	Bit reset (2 byte length)	16	●
	43	BRSTD	299	Bit reset (4 byte length)	16	●
	44	BRSTN	300	Bit reset (Arbitrary byte length)	20	●
	45	BTSTB	301	Bit test (1 byte length)	16	●
	46	BTSTW	302	Bit test (2 byte length)	16	●
	47	BTSTD	303	Bit test (4 byte length)	16	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).
 ×: Unusable.)

Table 2.1.7 (e) Functional instruction list (arranged in sequence of instruction group) (5)

Instruction group		Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Bit operation	48	BTSTN	304	Bit test (Arbitrary byte length)	20	●
	49	BPOSB	305	Bit search (1 byte length)	12	●
	50	BPOSW	306	Bit search (2 byte length)	12	●
	51	BPOSD	307	Bit search (4 byte length)	12	●
	52	BPOSN	308	Bit search (Arbitrary byte length)	16	●
	53	BCNTB	309	Bit count (1 byte length)	12	●
	54	BCNTW	310	Bit count (2 byte length)	12	●
	55	BCNTD	311	Bit count (4 byte length)	12	●
	56	BCNTN	312	Bit count (Arbitrary byte length)	16	●
Code conversion	1	COD	7	Code conversion	16+n (Note 3)	○
	2	CODB	27	Binary code conversion	20+n (Note 3)	○
	3	DCNV	14	Data conversion	12	○
	4	DCNVB	31	Extended data conversion	16	○
	5	DEC	4	Decoding	12	○
	6	DECB	25	Binary decoding	20	○
	7	TBCDB	313	Binary to BCD conversion (1 byte length)	16	●
	8	TBCDW	314	Binary to BCD conversion (2 byte length)	16	●
	9	TBCDD	315	Binary to BCD conversion (4 byte length)	16	●
	10	FBCDB	316	BCD to Binary conversion (1 byte length)	16	●
	11	FBCDW	317	BCD to Binary conversion (2 byte length)	16	●
	12	FBCDD	318	BCD to Binary conversion (4 byte length)	16	●
Operation	1	ADDB	36	Binary addition	20	○
	2	SUBB	37	Binary subtraction	20	○
	3	MULB	38	Binary multiplication	20	○
	4	DIVB	39	Binary division	20	○
	5	ADD	19	BCD addition	20	○
	6	SUB	20	BCD subtraction	20	○
	7	MUL	21	BCD multiplication	20	○
	8	DIV	22	BCD division	20	○
	9	NUMEB	40	Binary constant definition	16	○
	10	NUME	23	BCD-constant definition	12	○
	11	ADDSB	319	Addition (1 byte length)	20	●
	12	ADDSW	320	Addition (2 byte length)	20	●
	13	ADDSD	321	Addition (4 byte length)	20	●
	14	SUBSB	322	Subtraction (1 byte length)	20	●
	15	SUBSW	323	Subtraction (2 byte length)	20	●
	16	SUBSD	324	Subtraction (3 byte length)	20	●
	17	MULSB	325	Multiplication (1 byte length)	20	●
	18	MULSW	326	Multiplication (2 byte length)	20	●
	19	MULSD	327	Multiplication (4 byte length)	20	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).)

x: Unusable.)

Table 2.1.7 (f) Functional instruction list (arranged in sequence of instruction group) (6)

Instruction group		Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Operation	20	DIVSB	328	Division (1 byte length)	20	●
	21	DIVSW	329	Division (2 byte length)	20	●
	22	DIVSD	330	Division (4 byte length)	20	●
	23	MODSB	331	Remainder (1 byte length)	20	●
	24	MODSW	332	Remainder (2 byte length)	20	●
	25	MODSD	333	Remainder (4 byte length)	20	●
	26	INCSB	334	Increment (1 byte length)	8	●
	27	INCSW	335	Increment (2 byte length)	8	●
	28	INCSD	336	Increment (4 byte length)	8	●
	29	DECSSB	337	Decrement (1 byte length)	8	●
	30	DECSSW	338	Decrement (2 byte length)	8	●
	31	DECSD	339	Decrement (4 byte length)	8	●
	32	ABSSB	340	Absolute value (1 byte length)	16	●
	33	ABSSW	341	Absolute value (2 byte length)	16	●
	34	ABSSD	342	Absolute value (4 byte length)	16	●
	35	NEGSB	343	Sign inversion (1 byte length)	16	●
	36	NEGSW	344	Sign inversion (2 byte length)	16	●
	37	NEGSD	345	Sign inversion (4 byte length)	16	●
CNC Function	1	DISPB	41	Message display	8	×
	2	EXIN	42	External data input	8	×
	3	WINDR	51	CNC window data read	8	×
	4	WINDW	52	CNC window data write	8	×
	5	AXCTL	53	PMC axis control	12	×
	6	PSGN2	63	Position signal	8	×
	7	PSGNL	50	Position signal	12	×
Program control	1	COM	9	Common line control	8	○
	2	COME	29	End of common line control	4	○
	3	JMP	10	Jump	12	○
	4	JMPE	30	End of jump	4	○
	5	JMPB	68	Label jump 1	16	○
	6	JMPC	73	Label jump 2	16	○
	7	LBL	69	Label	12	○
	8	CALL	65	Conditional subprogram call	12	○
	9	CALLU	66	Unconditional subprogram call	12	○
	10	SP	71	Subprogram	8	○
	11	SPE	72	End of subprogram	4	○
	12	END1	1	End of first-level program	4	○
	13	END2	2	End of second-level program	4	○

(○: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).)

x: Unusable.)

Table 2.1.7 (g) Functional instruction list (arranged in sequence of instruction group) (7)

Instruction group		Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
Program control	14	END3	48	End of third-level program	4	○ (Note 2)
	15	END	64	End of ladder program	4	○
	16	NOP	70	No operation	8	○
	17	CS	74	Case call	8	○
	18	CM	75	Sub program call in case call	12	○
	19	CE	76	End of case call	4	○
Rotation control	1	ROT	6	Rotation control	20	○
	2	ROTB	26	Binary rotation control	24	○
Invalid instruction	1	SPCNT	46	Spindle control	16	Δ
	2	DISP	49	Message display	16+n (Note 3)	Δ
	3	MMCWR	98	MMC window data read	12	Δ
	4	MMCWW	99	MMC window data write	12	Δ
	5	FNC90	90	Arbitrary-function instruction 1	8	Δ
	6	FNC91	91	Arbitrary-function instruction 2	8	Δ
	7	FNC92	92	Arbitrary-function instruction 3	8	Δ
	8	FNC93	93	Arbitrary-function instruction 4	8	Δ
	9	FNC94	94	Arbitrary-function instruction 5	8	Δ
	10	FNC95	95	Arbitrary-function instruction 6	8	Δ
	11	FNC96	96	Arbitrary-function instruction 7	8	Δ
	12	FNC97	97	Arbitrary-function instruction 8	8	Δ

(○: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

×: Unusable.)

NOTE

- 1 These instructions are intended to maintain source-level compatibility with programs for conventional models. They are treated as a NOP instruction (instruction that performs no operation).
- 2 The 3rd level sequence part is available. However, the execution cycle period for processing the 3rd level sequence part is not guaranteed.
- 3 Memory size increases by the number of data tables to be used. In the COD instruction, CODB instruction (1byte length), CODB instruction (2byte length), or DISP instruction, 2 bytes are added for each data. And, when the number of data is odd, 2 bytes are added moreover. In the CODB instruction (4byte length), 4 bytes are added for each data.

2.1.8 Functional Instructions (Arranged in Sequence of SUB No.)

Table 2.1.8 (a) Functional instruction list (arranged in sequence of SUB No.) (1)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
END1	1	End of first-level program	4	○
END2	2	End of second-level program	4	○
TMR	3	Timer processing	8	○
DEC	4	Decoding	12	○
CTR	5	Counter processing	8	○
ROT	6	Rotation control	20	○
COD	7	Code conversion	16+n (Note 3)	○
MOVE	8	Data transfer after logical product	20	○
COM	9	Common line control	8	○
JMP	10	Jump	12	○
PARI	11	Parity check	8	○
DCNV	14	Data conversion	12	○
COMP	15	Comparison	16	○
COIN	16	Coincidence check	16	○
DSCH	17	Data search	20	○
XMOV	18	Index modification data transfer	20	○
ADD	19	Addition	20	○
SUB	20	Subtraction	20	○
MUL	21	Multiplication	20	○
DIV	22	Division	20	○
NUME	23	Constant definition	12	○
TMRB	24	Fixed-timer processing	12	○
DECB	25	Binary decoding	20	○
ROTB	26	Binary rotation control	24	○
CODB	27	Binary code conversion	20+n (Note 3)	○
MOVOR	28	Data transfer after logical sum	16	○
COME	29	End of common line control	4	○
JMPE	30	End of jump	4	○
DCNVB	31	Extended data conversion	16	○
COMPB	32	Binary comparison	20	○
SFT	33	Shift register	8	○
DSCHB	34	Binary data search	24	○
XMOVB	35	Index modification binary data transfer	24	○
ADDB	36	Binary addition	20	○
SUBB	37	Binary subtraction	20	○
MULB	38	Binary multiplication	20	○
DIVB	39	Binary division	20	○
NUMEB	40	Binary constant definition	16	○
DISPB	41	Message display	8	×
EXIN	42	External data input	8	×
MOVB	43	1-byte transfer	12	○
MOVW	44	2-byte transfer	12	○
MOVN	45	Transfer of arbitrary number of bytes	16	○
SPCNT	46	Spindle control	16	△

(O: Usable. ●: The Extended PMC Ladder Instruction Function △: Executed as NOP instruction (Note 1).
x: Unusable.)

Table 2.1.8 (b) Functional instruction list (arranged in sequence of SUB No.) (2)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
MOVD	47	4-byte transfer	12	○
END3	48	End of third-level program	4	○ (Note 2)
DISP	49	Message display	16+n (Note 3)	△
PSGNL	50	Position signal	12	×
WINDR	51	CNC window data read	8	×
WINDW	52	CNC window data write	8	×
AXCTL	53	PMC axis control	12	×
TMRC	54	Timer processing	16	○
CTRC	55	Counter processing	12	○
CTRБ	56	Counter processing	12	○
DIFU	57	Rising-edge detection	8	○
DIFD	58	Falling-edge detection	8	○
EOR	59	Exclusive OR	20	○
AND	60	Logical AND	20	○
OR	61	Logical OR	20	○
NOT	62	Logical NOT	16	○
PSGN2	63	Position signal	8	×
END	64	End of ladder program	4	○
CALL	65	Conditional subprogram call	12	○
CALLU	66	Unconditional subprogram call	12	○
JMPB	68	Label jump 1	16	○
LBL	69	Label	12	○
NOP	70	No operation	8	○
SP	71	Subprogram	8	○
SPE	72	End of subprogram	4	○
JMPC	73	Label jump 2	16	○
CS	74	Case call	8	○
CM	75	Sub program call in case call	12	○
CE	76	End of case call	4	○
TMRBF	77	Fixed off-delay timer	12	○
FNC90	90	Arbitrary-function instruction 1	8	△
FNC91	91	Arbitrary-function instruction 2	8	△
FNC92	92	Arbitrary-function instruction 3	8	△
FNC93	93	Arbitrary-function instruction 4	8	△
FNC94	94	Arbitrary-function instruction 5	8	△
FNC95	95	Arbitrary-function instruction 6	8	△
FNC96	96	Arbitrary-function instruction 7	8	△
FNC97	97	Arbitrary-function instruction 8	8	△
MMCWR	98	MMC window data read	12	△
MMCWW	99	MMC window data write	12	△

(O: Usable. ●: The Extended PMC Ladder Instruction Function △: Executed as NOP instruction (Note 1).
x: Unusable.)

Table 2.1.8 (c) Functional instruction list (arranged in sequence of SUB No.) (3)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
EQB	200	Signed Binary Comparison (=)(1 byte length)	16	○
EQW	201	Signed Binary Comparison (=)(2 byte length)	16	○
EQD	202	Signed Binary Comparison (=)(4 byte length)	16	○
NEB	203	Signed Binary Comparison (≠)(1 byte length)	16	○
NEW	204	Signed Binary Comparison (≠)(2 byte length)	16	○
NED	205	Signed Binary Comparison (≠)(4 byte length)	16	○
GTB	206	Signed Binary Comparison (>)(1 byte length)	16	○
GTW	207	Signed Binary Comparison (>)(2 byte length)	16	○
GTD	208	Signed Binary Comparison (>)(4 byte length)	16	○
LTB	209	Signed Binary Comparison (<)(1 byte length)	16	○
LTW	210	Signed Binary Comparison (<)(2 byte length)	16	○
LTD	211	Signed Binary Comparison (<)(4 byte length)	16	○
GEB	212	Signed Binary Comparison (\geq)(1 byte length)	16	○
GEW	213	Signed Binary Comparison (\geq)(2 byte length)	16	○
GED	214	Signed Binary Comparison (\geq)(4 byte length)	16	○
LEB	215	Signed Binary Comparison (\leq)(1 byte length)	16	○
LEW	216	Signed Binary Comparison (\leq)(2 byte length)	16	○
LED	217	Signed Binary Comparison (\leq)(4 byte length)	16	○
RNGB	218	Signed Binary Comparison (range)(1 byte length)	20	○
RNGW	219	Signed Binary Comparison (range)(2 byte length)	20	○
RNGB	220	Signed Binary Comparison (range)(4 byte length)	20	○
TMRST	221	Stop watch timer (1 ms accuracy)	20	●
TMRSS	222	Stop watch timer (1 sec accuracy)	20	●
CTRD	223	Counter processing (4 byte length)	12	●
MOVBT	224	Bit transfer	24	●
SETNB	225	Data setting (1 byte length)	20	●
SETNW	226	Data setting (2 byte length)	20	●
SETND	227	Data setting (4 byte length)	20	●
XCHGB	228	Data exchange (1 byte length)	12	●
XCHGW	229	Data exchange (2 byte length)	12	●
XCHGD	230	Data exchange (4 byte length)	12	●
SWAPW	231	Data swap (2 byte length)	16	●
SWAPD	232	Data swap (4 byte length)	16	●
TBLRB	233	Reading data from table (1 byte length)	24	●
TBLRW	234	Reading data from table (2 byte length)	24	●
TBLRD	235	Reading data from table (4 byte length)	24	●
TBLRN	236	Reading data from table (Arbitrary byte length)	28	●
TBLWB	237	Writing data to table (1 byte length)	24	●
TBLWW	238	Writing data to table (2 byte length)	24	●
TBLWD	239	Writing data to table (4 byte length)	24	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

x: Unusable.)

Table 2.1.8 (d) Functional instruction list (arranged in sequence of SUB No.) (4)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
TBLWN	240	Writing data to table (Arbitrary byte length)	28	●
DSEQB	241	Searching data from table (=)(1 byte length)	28	●
DSEQW	242	Searching data from table (=)(2 byte length)	28	●
DSEQD	243	Searching data from table (=)(4 byte length)	28	●
DSNEB	244	Searching data from table (≠)(1 byte length)	28	●
DSNEW	245	Searching data from table (≠)(2 byte length)	28	●
DSNED	246	Searching data from table (≠)(4 byte length)	28	●
DSGTB	247	Searching data from table (>)(1 byte length)	28	●
DSGTW	248	Searching data from table (>)(2 byte length)	28	●
DSGTD	249	Searching data from table (>)(4 byte length)	28	●
DSLTB	250	Searching data from table (<)(1 byte length)	28	●
DSLTw	251	Searching data from table (<)(2 byte length)	28	●
DSLTD	252	Searching data from table (<)(4 byte length)	28	●
DSGEB	253	Searching data from table (≥)(1 byte length)	28	●
DSGEW	254	Searching data from table (≥)(2 byte length)	28	●
DSGED	255	Searching data from table (≥)(4 byte length)	28	●
DSLEB	256	Searching data from table (≤)(1 byte length)	28	●
DSLEW	257	Searching data from table (≤)(2 byte length)	28	●
DSLED	258	Searching data from table (≤)(4 byte length)	28	●
DMAXB	259	Maximum data (1 byte length)	28	●
DMAXW	260	Maximum data (2 byte length)	28	●
DMAXD	261	Maximum data (4 byte length)	28	●
DMINB	262	Minimum data (1 byte length)	28	●
DMINW	263	Minimum data (2 byte length)	28	●
DMIND	264	Minimum data (4 byte length)	28	●
EORB	265	Exclusive OR (1 byte length)	20	●
EORW	266	Exclusive OR (2 byte length)	20	●
EORD	267	Exclusive OR (4 byte length)	20	●
ANDB	268	Logical AND (1 byte length)	20	●
ANDW	269	Logical AND (2 byte length)	20	●
ANDD	270	Logical AND (4 byte length)	20	●
ORB	271	Logical OR (1 byte length)	20	●
ORW	272	Logical OR (2 byte length)	20	●
ORD	273	Logical OR (4 byte length)	20	●
NOTB	274	Logical NOT (1 byte length)	16	●
NOTW	275	Logical NOT (2 byte length)	16	●
NOTD	276	Logical NOT (4 byte length)	16	●
SHLB	277	Bit shift left (1 byte length)	20	●
SHLW	278	Bit shift left (2 byte length)	20	●
SHLD	279	Bit shift left (4 byte length)	20	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

x: Unusable.)

Table 2.1.8 (e) Functional instruction list (arranged in sequence of SUB No.) (5)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
SHLN	280	Bit shift left (Arbitrary byte length)	24	●
SHRB	281	Bit shift right (1 byte length)	20	●
SHRW	282	Bit shift right (2 byte length)	20	●
SHRD	283	Bit shift right (4 byte length)	20	●
SHRN	284	Bit shift right (Arbitrary byte length)	24	●
ROLB	285	Bit rotation left (1 byte length)	20	●
ROLW	286	Bit rotation left (2 byte length)	20	●
ROLD	287	Bit rotation left (4 byte length)	20	●
ROLN	288	Bit rotation left (Arbitrary byte length)	24	●
RORB	289	Bit rotation right (1 byte length)	20	●
RORW	290	Bit rotation right (2 byte length)	20	●
RORD	291	Bit rotation right (4 byte length)	20	●
RORN	292	Bit rotation right (Arbitrary byte length)	24	●
BSETB	293	Bit set (1 byte length)	16	●
BSETW	294	Bit set (2 byte length)	16	●
BSETD	295	Bit set (4 byte length)	16	●
BSETN	296	Bit set (Arbitrary byte length)	20	●
BRSTB	297	Bit reset (1 byte length)	16	●
BRSTW	298	Bit reset (2 byte length)	16	●
BRSTD	299	Bit reset (4 byte length)	16	●
BRSTN	300	Bit reset (Arbitrary byte length)	20	●
BTSTB	301	Bit test (1 byte length)	16	●
BTSTW	302	Bit test (2 byte length)	16	●
BTSTD	303	Bit test (4 byte length)	16	●
BTSTN	304	Bit test (Arbitrary byte length)	20	●
BPOSB	305	Bit search (1 byte length)	12	●
BPOSW	306	Bit search (2 byte length)	12	●
BPOSD	307	Bit search (4 byte length)	12	●
BPOSN	308	Bit search (Arbitrary byte length)	16	●
BCNTB	309	Bit count (1 byte length)	12	●
BCNTW	310	Bit count (2 byte length)	12	●
BCNTD	311	Bit count (4 byte length)	12	●
BCNTN	312	Bit count (Arbitrary byte length)	16	●
TBCDB	313	Binary to BCD conversion (1 byte length)	16	●
TBCDW	314	Binary to BCD conversion (2 byte length)	16	●
TBCDD	315	Binary to BCD conversion (4 byte length)	16	●
FBCDB	316	BCD to Binary conversion (1 byte length)	16	●
FBCDW	317	BCD to Binary conversion (2 byte length)	16	●
FBCDD	318	BCD to Binary conversion (4 byte length)	16	●
ADDSB	319	Addition (1 byte length)	20	●

(O: Usable. ●: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

x: Unusable.)

Table 2.1.8 (f) Functional instruction list (arranged in sequence of SUB No.) (6)

Instruction name	SUB No.	Processing	Required memory size (byte)	1st to 5th PMC
ADDSW	320	Addition (2 byte length)	20	•
ADDSD	321	Addition (4 byte length)	20	•
SUBSB	322	Subtraction (1 byte length)	20	•
SUBSW	323	Subtraction (2 byte length)	20	•
SUBSD	324	Subtraction (3 byte length)	20	•
MULSB	325	Multiplication (1 byte length)	20	•
MULSW	326	Multiplication (2 byte length)	20	•
MULSD	327	Multiplication (4 byte length)	20	•
DIVSB	328	Division (1 byte length)	20	•
DIVSW	329	Division (2 byte length)	20	•
DIVSD	330	Division (4 byte length)	20	•
MODSB	331	Remainder (1 byte length)	20	•
MODSW	332	Remainder (2 byte length)	20	•
MODSD	333	Remainder (4 byte length)	20	•
INCSB	334	Increment (1 byte length)	8	•
INCSW	335	Increment (2 byte length)	8	•
INCSD	336	Increment (4 byte length)	8	•
DECSB	337	Decrement (1 byte length)	8	•
DECSD	338	Decrement (2 byte length)	8	•
DECSD	339	Decrement (4 byte length)	8	•
ABSSB	340	Absolute value (1 byte length)	16	•
ABSSW	341	Absolute value (2 byte length)	16	•
ABSSD	342	Absolute value (4 byte length)	16	•
NEGSB	343	Sign inversion (1 byte length)	16	•
NEGSW	344	Sign inversion (2 byte length)	16	•
NEGSD	345	Sign inversion (4 byte length)	16	•

(O: Usable. •: The Extended PMC Ladder Instruction Function Δ: Executed as NOP instruction (Note 1).

x: Unusable.)

NOTE

- 1 These instructions are intended to maintain source-level compatibility with programs for conventional models. They are treated as a NOP instruction (instruction that performs no operation).
- 2 The 3rd level sequence part is available. However, the execution cycle period for processing the 3rd level sequence part is not guaranteed.
- 3 Memory size increases by the number of data tables to be used. In the COD instruction, CODB instruction (1byte length), CODB instruction (2byte length), or DISP instruction, 2 bytes are added for each data. And, when the number of data is odd, 2 bytes are added moreover. In the CODB instruction (4byte length), 4 bytes are added for each data.

2.2 PMC SIGNAL ADDRESSES

This section describes the use of each PMC address. See Subsection 2.1.5 for explanations about all address types and ranges.

2.2.1 Addresses for Signals Between the PMC and ROBOT (F, G)

This subsection briefly describes interface addresses.

(1) Signals from the Robot to the PMC

The following table lists the range of addresses for the signals sent from the Robot to the PMC. Refer to Internal I/O Assignement for details about the signals.

Data kind	1st path PMC	2nd to 5th path PMC (Option)
Input signals from Robot to PMC	F0 ~ F767 F1000 ~ F1767	F0 ~ F767

(2) Signals from the PMC to the Robot

The following table lists the range of addresses for the signals sent from the PMC to the Robot. Refer to Internal I/O Assignement for details about the signals.

Data kind	1st path PMC	2nd to 5th path PMC (Option)
Output signals to Robot from PMC	G0 ~ G767 G1000 ~ G1767	G0 ~ G767

2.2.2 Addresses of Signals Between the PMC and External I/O Device (X, Y)

(1) Assignment of the external I/O device

(a) Input signals from an external I/O device to PMC
1st to 5th path PMC

The addresses for four channels, X0 to X127, X200 to X327, X400 to X527, X600 to X727 and X1000 to X1127, can be used for the input signals to PMCs.

(b) Output Signals from PMC to an external I/O device
1st to 5th path PMC

The addresses for four channels, Y0 to Y127, Y200 to Y327, Y400 to Y527, Y600 to Y727 and Y1000 to Y1127, can be used for output signals from PMCs.

2.2.3 Internal Relay Addresses (R)

The following table lists the number of signals (bytes) that can be used as internal relays.

Signals that interface with robot or external I/O device can be assigned to these bytes.

Turning on the power clears these areas to 0.

NOTE

This address is not synchronized in the 2nd level ladder. A value of a signal in this address may change during the execution of 2nd level ladder from the value in the 1st and 3rd level ladder.

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Internal Relay Address (Size)	R0 ~ R1499 (1500)	R0 ~ R7999 (8000)	R0 ~ R15999 (16000)	R0 ~ R59999 (60000)

2.2.4 System Relay Addresses (R9000, Z0)

The System Relay is used to control a sequence program by PMC System software. Some addresses such as 'Operation results of functional instructions' are used to condition a sequence program.

The System Relay uses the following PMC address by each PMC Memory Type.

Table 2.2.4 Address of System Relay

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
System relay (Size)	R9000 ~ R9499 (500)	R9000 ~ R9499 (500)	Z0 ~ Z499 (500)	Z0 ~ Z499 (500)

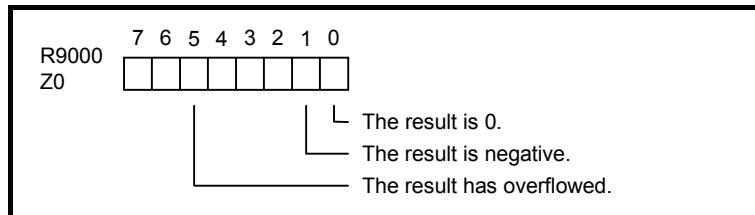
NOTE

- 1 The address conversion of the System Relay is necessary when a Sequence Program is changed between PMC Memory-A/B and PMC Memory-C/D.

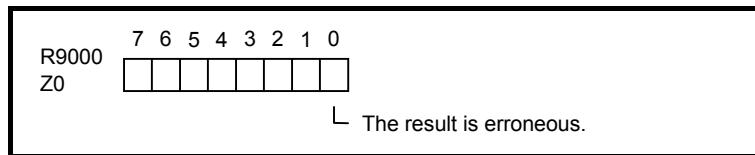
Operation results of functional instructions

This area holds information necessary for individual ladder levels, such as the operation results of functional instructions. This information is saved/restored when the task is switched.

- (1) R9000, Z0 (operation output register for the ADDB, SUBB, MULB, DIVB, and COMPB functional instructions)



- (2) R9000, Z0 (error output for the EXIN, WINDR, and WINDW functional instructions)

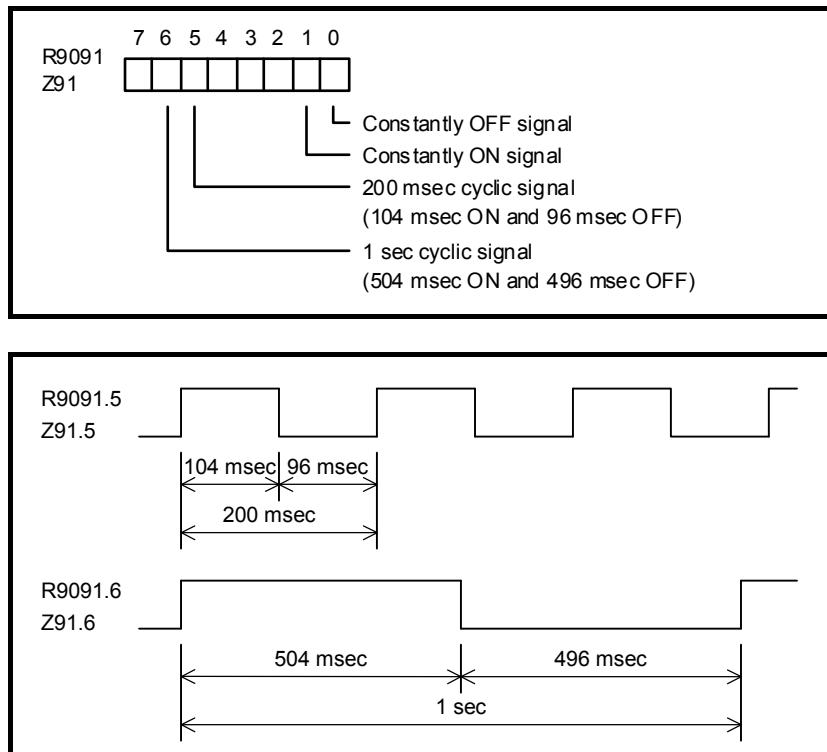


- (3) R9002 to R9005, Z2 to Z5 (operation output registers for the DIVB functional instruction)
The remainder of a division performed with the DIVB functional instruction is output to these addresses.

System timers

Four signals can be used as system timers.

Their specifications are as follows.



CAUTION

- 1 Each signal is initially OFF.
- 2 The signals R9091.0, R9091.1, Z91.0 and Z91.1 are set at the beginning of the first ladder level on every power cycle.
- 3 Each pulse signal (ON-OFF signal) may have an error of ± 8 or 4 msec (ladder execution period).

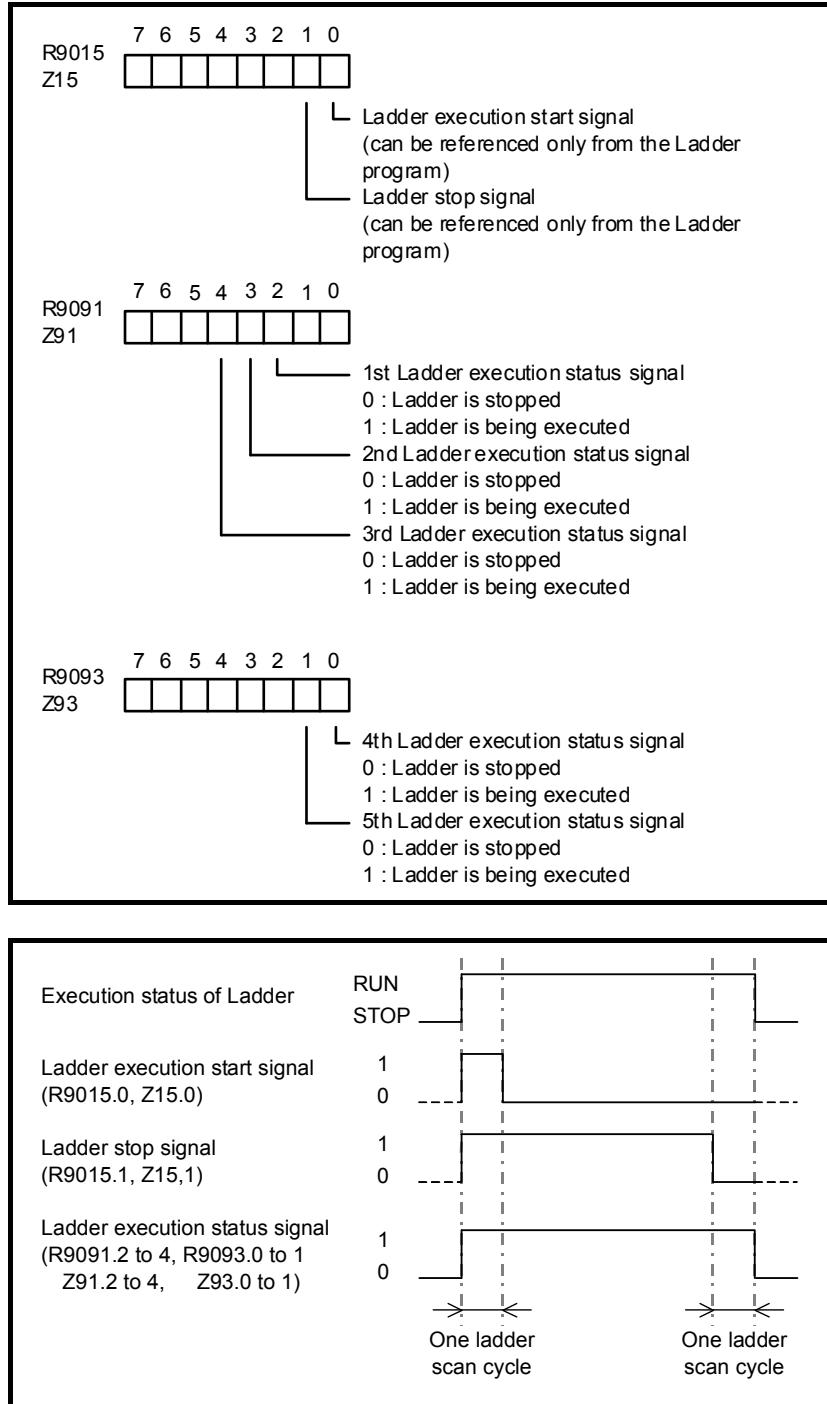
Ladder execution start signal

Ladder stop signal

Ladder execution status signal

Using the ladder execution start and stop signals in a ladder program can detect when the ladder program starts and stops.

Referencing the ladder execution status signal from an external system or program can detect the execution status of the ladder program.



(1) Ladder execution start signal (R9015.0, Z15.0)

When directed to start ladder program execution, the system software starts executing the ladder program, turns on this signal, and keeps it on for the first scan cycle. Like R9000 and Z0, this signal indicates the status of ladder execution corresponding to each ladder execution level. For this reason, this signal is securely turned on for the first scan cycle after the start of execution no matter on what execution level the signal is referenced. This signal is turned on when:

- Ladder execution begins at power turn-on.
- PMC execution is started on the PMC menu.
- FANUC LADDER-III for Robot directs the ladder to start.

Referring to this signal in a ladder program can detect when ladder execution has begun, making it possible to program preprocessing related to ladder execution.

⚠ CAUTION

Refer this signal only from the ladder program. Do not refer it from an external system or program as it indicates the status of ladder execution separately for each ladder execution level.

(2) Ladder stop signal (R9015.1, Z15.1)

When directed to stop ladder program execution, the system software turns off this signal and keeps it off for the last scan cycle before stopping ladder program execution. Like R9000 and Z0, this signal indicates the status of ladder execution corresponding to each ladder execution level. For this reason, this signal is securely turned off for the last scan cycle before the stop of execution no matter on what execution level the signal is referenced. This signal is turned off when:

(a) PMC execution is stopped on the PMC menu.

(b) FANUC LADDER-III for Robot directs the ladder to stop.

Referencing this signal in a ladder program can detect when ladder execution stops, making it possible to program post processing related to ladder execution (that is, preprocessing for ladder execution stop). Before the ladder is stopped, for example, it is possible to put signals in a proper state for safety purposes.

⚠ CAUTION

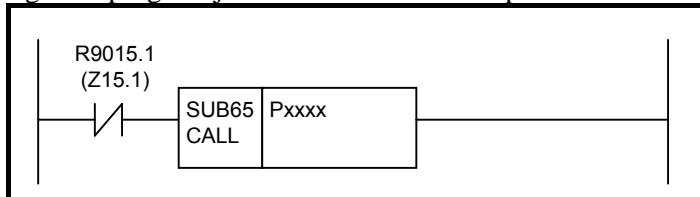
- 1 Reference this signal only from the ladder program. Do not reference it from an external system or program as it indicates the status of ladder execution separately for each ladder execution level.
- 2 If the power is turned off, ladder execution and I/O signal transfer are immediately stopped for safety purposes. In this case, therefore, this signal cannot be used.

(3) Ladder execution status signal (R9091.2 to 4, R9093.0 to 1, Z91.2 to 4, Z93.0 to 1)

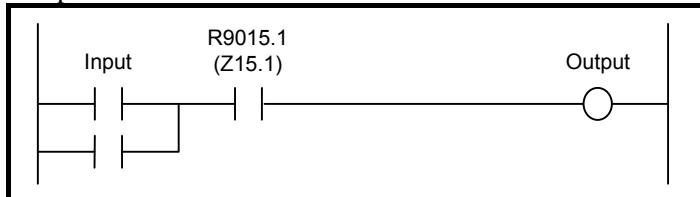
Referencing this signal can detect the execution status of the ladder program.

(4) Example of using the signals

(a) Example of calling a subprogram just before the ladder stops

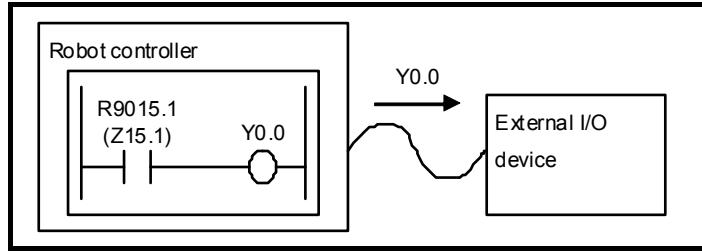


(b) Example of forcibly turning off an output signal programmed on the first ladder level just before the ladder stops



(c) Example of sending an execution-in-progress signal to the outside

Outputting the status of this signal as the DO signal (output address from the PMC) assigned to the external I/O device causes the robot controller to be interlocked with an external system.



2.2.5 Extra Relay Addresses (E)

The following table lists the number of signals (bytes) that can be used as extra relays.

Extra relays can be used in the same manner as for internal relays.

Turning off the power does not cause the memory contents to be erased because these areas are nonvolatile memory.

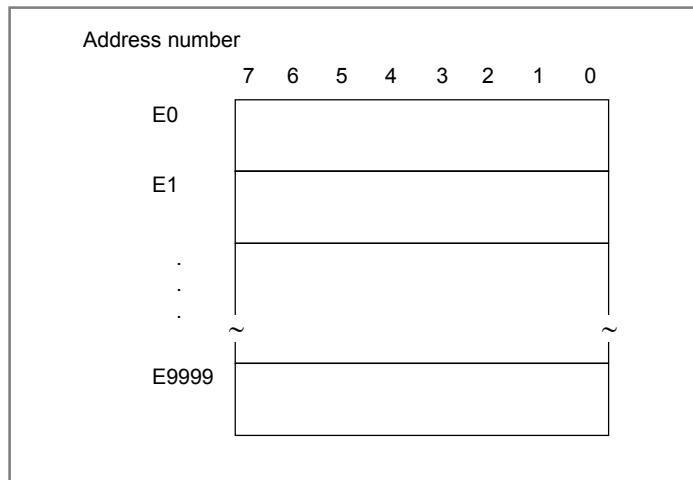
NOTE

- These addresses are not synchronized in the 2nd level ladder. A value of a signal in these addresses may change during the execution of 2nd level ladder from the value in the 1st and 3rd level ladder when it is written in other program.

Table 2.2.5 Address of Extra Relay

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Extra relay (Size)	E0 ~ E9999 (10,000)	E0 ~ E9999 (10,000)	E0 ~ E9999 (10,000)	E0 ~ E9999 (10,000)

For the multi-PMC function, this area is common memory. Each PMC program can write and read the same value from in the area.



System Keep Relays related to Extra Relays

The following system keep relays have influence on managing extra relays.

K906.3 EOUTPUT (Available on 1st PMC only)

- 0: On the I/O screen, the E address is output when PMC parameters are output.
- 1: On the I/O screen, the E address is not output when PMC parameters are output.

2.2.6 Message Display Addresses (A)

This address is not used.

2.2.7 Timer Addresses (T)

These addresses are for variable timers, used with the TMR instruction and for the precision of the variable timers.

The following table lists how many timers can be used (number of timers = number of bytes/2).

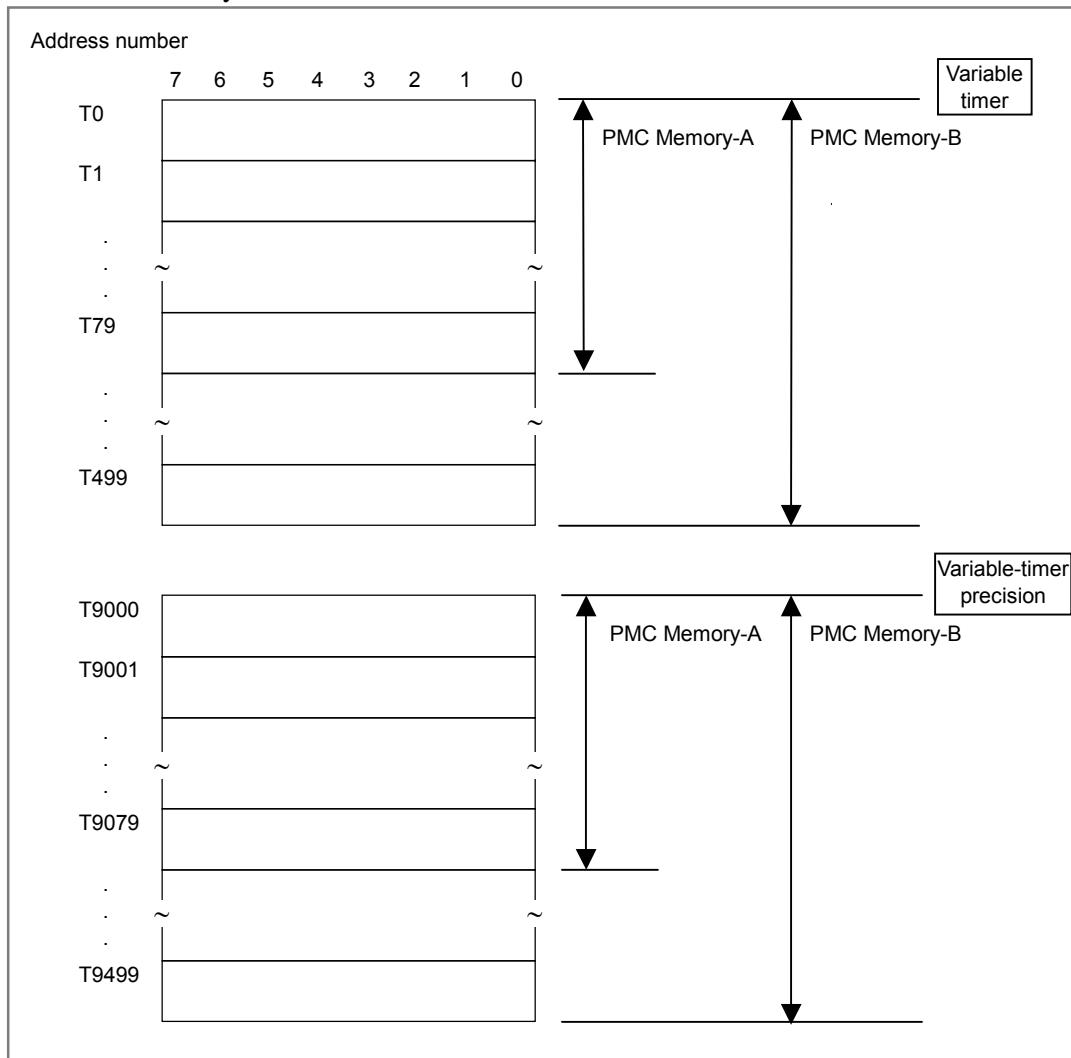
The number of timer precision values matches that of the timers.

Turning off the power does not cause the memory contents to be erased because these areas are nonvolatile memory.

Table 2.2.7 Address of variable timer

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Variable timer	Number of timers	40 pieces	250 pieces	500 pieces
	time setting	T0 ~ T79	T0 ~ 499	T0 ~ T999
	precision	T9000 ~ T9079	T9000 ~ 9499	T0 ~ T9999

Example of PMC Memory-A/B



2.2.8 Counter Addresses (C)

These addresses are for variable counters, used with the CTR instruction and for fixed counters used with the CTRB instruction.

The numbers of the counters that can be used are:

The number of variable counters = number of bytes/4

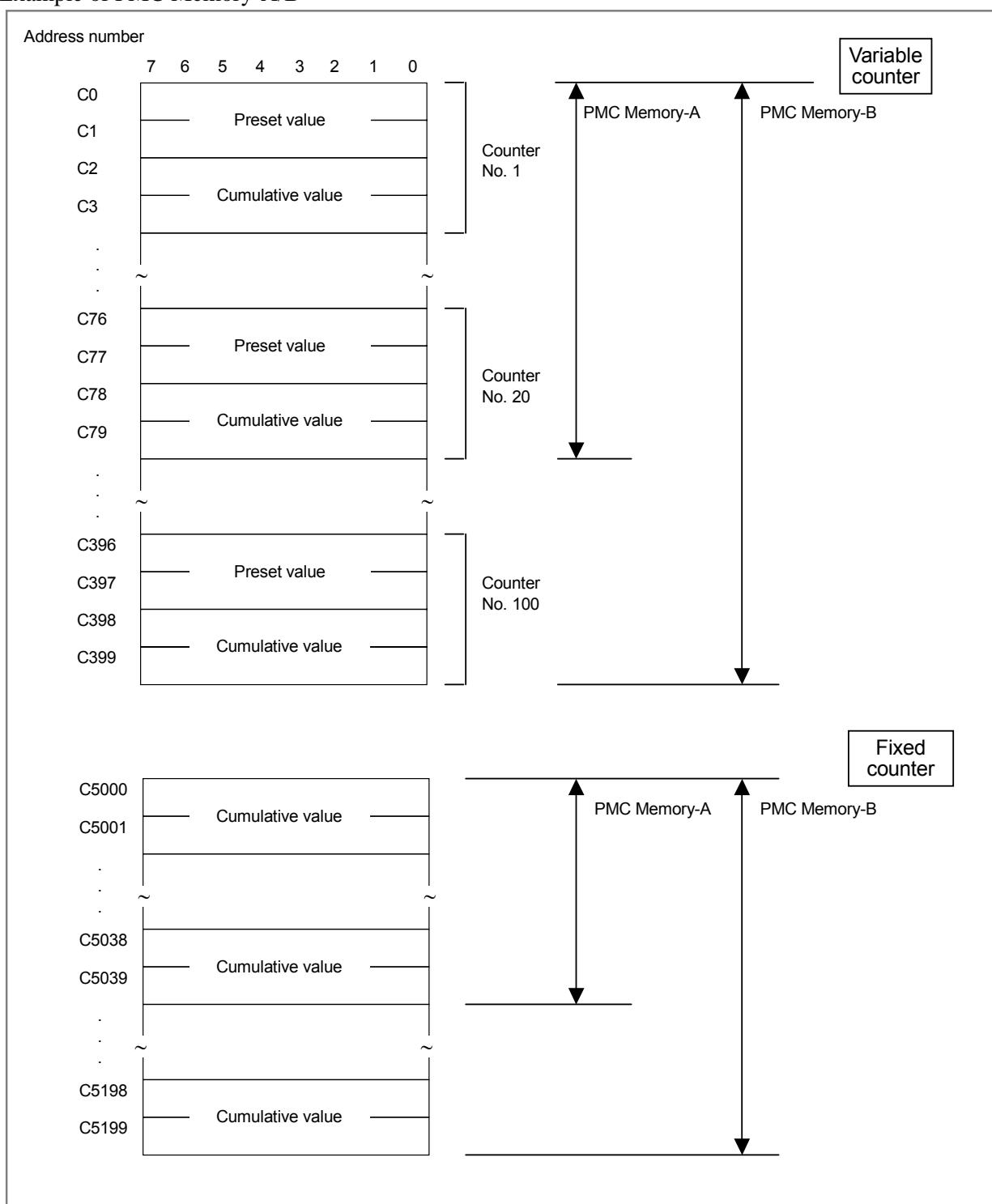
The number of fixed counters = number of bytes/2

Turning off the power does not cause the memory contents to be erased because these areas are nonvolatile memory.

Table 2.2.8 Address of counters

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Variable counter (Number of counters)	C0 ~ C79 (20 pieces)	C0 ~ C399 (100 pieces)	C0 ~ C799 (200 pieces)	C0 ~ C1199 (300 pieces)
Fixed counter (Number of counters)	C5000 ~ C5039 (20 pieces)	C5000 ~ C5199 (100 pieces)	C5000 ~ C5399 (200 pieces)	C5000 ~ C5599 (300 pieces)

Example of PMC Memory-A/B



2.2.9 Keep Relay Addresses (K)

These addresses are for keep relays and PMC parameters.

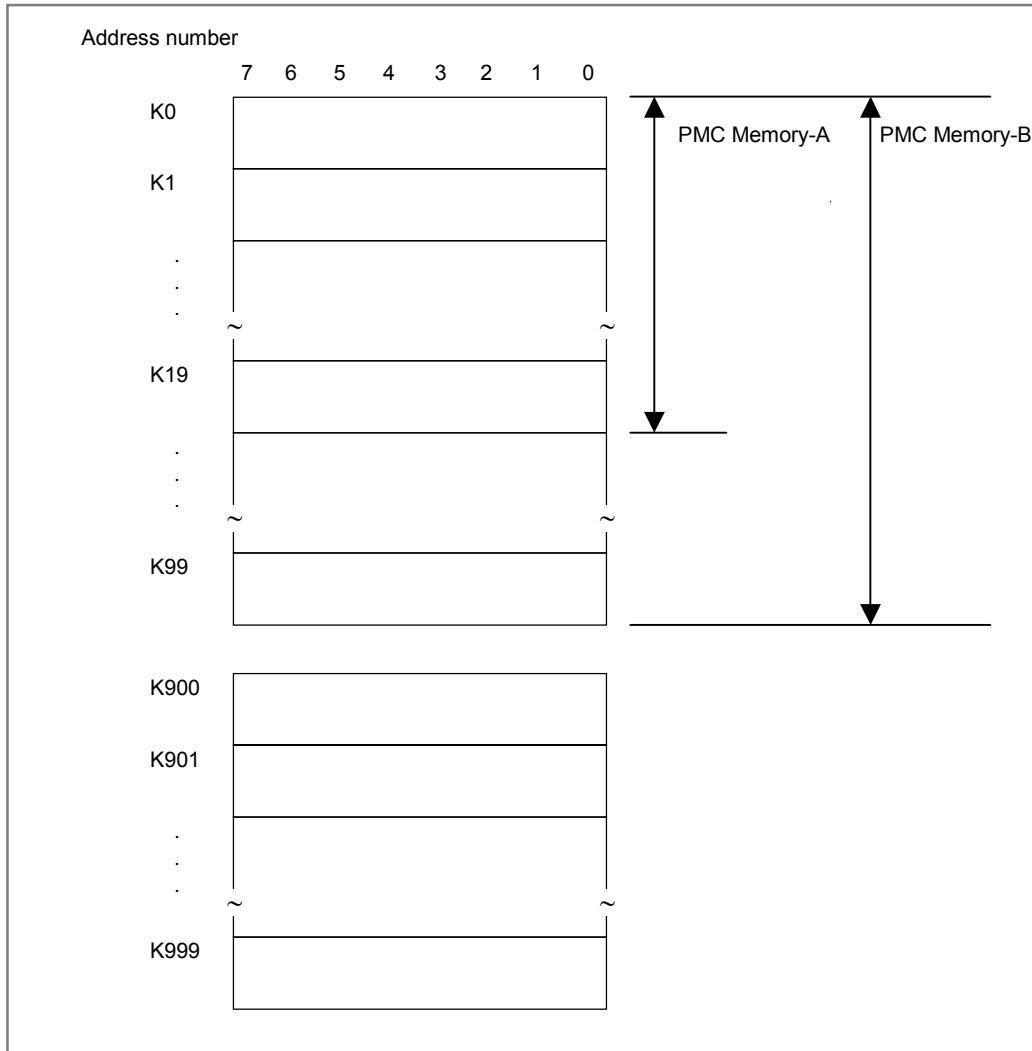
The following table lists the number of bytes that can be used. It also lists information related to the nonvolatile memory control addresses and the area (system area) used by the management software.

Turning off the power does not cause the memory contents to be erased because these areas are nonvolatile memory.

Table 2.2.9 Address of keep relays

Data kind		1st to 5th path PMC			
		PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Keep relays	User area (Size)	K0 ~ K19 (20 bytes)	K0 ~ K99 (100 bytes)	K0 ~ K199 (200 bytes)	K0 ~ K299 (300 bytes)
	System area (Size)	K900 ~ K999 (100 bytes)			
Nonvolatile memory control address		K909	K909	K909	K909

Example of PMC Memory-A/B



2.2.10 System Keep Relay Addresses (K)

The following table lists the keep relays used by the system (PMC management software).

Table 2.2.10 Address of System keep relay

Data kind	1st to 5th path PMC				DCS PMC
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D	
Area used by the management software (Size)	K900 ~ K999 (100 bytes)				

Explained below is the meaning of each bit of the system keep relay address. The bits and addresses left unused are reserved for use by the system.

The system keep relays indicated with an asterisk (*) can be set up, using setting parameters.

K900	#7 DTBLDSP	#6	#5	#4 MEMINP	#3	#2 AUTORUN	#1 PRGRAM	#0 LADMASK
------	------------	----	----	-----------	----	------------	-----------	------------

[Data type] Bit

LADMASK PMC program view inhibit(*)

- 0: The sequence program is allowed to be viewed.
- 1: The sequence program is inhibited from being viewed.

PRGRAM Programmer function enable(*)

- 0: The built-in programmer function is disabled.
- 1: The built-in programmer function is enabled.

AUTORUN PMC program execute(*)

- 0: The sequence program is automatically started when the power is turned on.
- 1: The sequence program is started, using the sequence program execution soft key.

MEMINP Memory write permit(*)

- 0: The forcing and override functions are disabled.
- 1: The forcing and override functions are enabled.

NOTE

Using the override function requires setting "Override enable" (K906.0).

DTBLDSP Data table GRP setting display(*)

- 0: The DATA TABLE CONTROL screen is displayed.
- 1: The DATA TABLE CONTROL screen is not displayed.

K903	#7	#6	#5 CLRATVAR	#4 CLRFBVAR	#3	#2	#1	#0
------	----	----	-------------	-------------	----	----	----	----

[Data type] Bit

CLRFBVAR Initialization of FB variable area at updating sequence program

- 0: Clear FB variable area when FB variable is changed.
- 1: Not clear FB variable area.

NOTE

Refer to "8.1.4 Assignment of FB variable" about address assignment of FB variable.

CLRATVAR Initialization of the memory area for automatic address assignment at updating sequence program

- 0: Not clear the area to which addresses are assigned automatically.

- 1: Clear the area to which addresses are assigned automatically when changing symbol data other than FB variable.

NOTE

Refer to "(6) Automatic address assignment at compiling on FANUC LADDER-III for Robot" of "1.2.7 Extension of a symbol and comment" about automatic address assignment.

K906	#7	#6	#5	#4	#3	#2	#1	#0
								OVRRID

[Data type] Bit

OVRRID Override enable(*)

- 0: The override function is disabled.
1: The override function is enabled.

NOTE

Using the override function requires setting "Memory write permit" (K900.4).

2.2.11 Data Table Addresses (D)

PMC sequence control sometimes requires a sizable amount of numeric data (hereinafter referred to as data table). If the contents of a data table can be set or read freely, they can be used as various PMC sequence control data.

Each table can have an arbitrary size as long as it fits the data table memory, and 1-, 2-, and 4-byte binary and BCD data can be used for each table separately; so it is possible to configure efficient, easy-to-use tables.

Data in a data table can be set in PMC nonvolatile memory or displayed via the DATA TABLE screen. Data set in data tables can also be easily read and written with the sequence program using functional instructions such as data search (DSCHB) and index modification data transfer (XMOVB).

The following table lists the number of bytes that can be used.

NOTE

These addresses are not synchronized in the 2nd level ladder. A value of a signal in these addresses may change during the execution of 2nd level ladder from the value in the 1st and 3rd level ladder when it is written in other program.

Table 2.2.11 Address of Data table

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Data table (Size)	D0 ~ D2999 (3,000 bytes)	D0 ~ D9999 (10,000 bytes)	D0 ~ D19999 (20,000 bytes)	D0 ~ D59999 (60,000 bytes)

(1) Data table configuration

The PMC data table consists of table control data and data tables. The table control data manages the data form (binary or BCD) and size of each table.

Creating a data table requires first setting up table control data from the DATA TABLE CONTROL screen.

The sequence program cannot read or write the table control data. If the USB memory, and so on, are used to read or write the contents of the nonvolatile memory, however, the table control data is read or written together. Fig. 2.2.12 (a) roughly shows the configuration of the data table. Fig. 2.2.12 (b) shows it in detail.

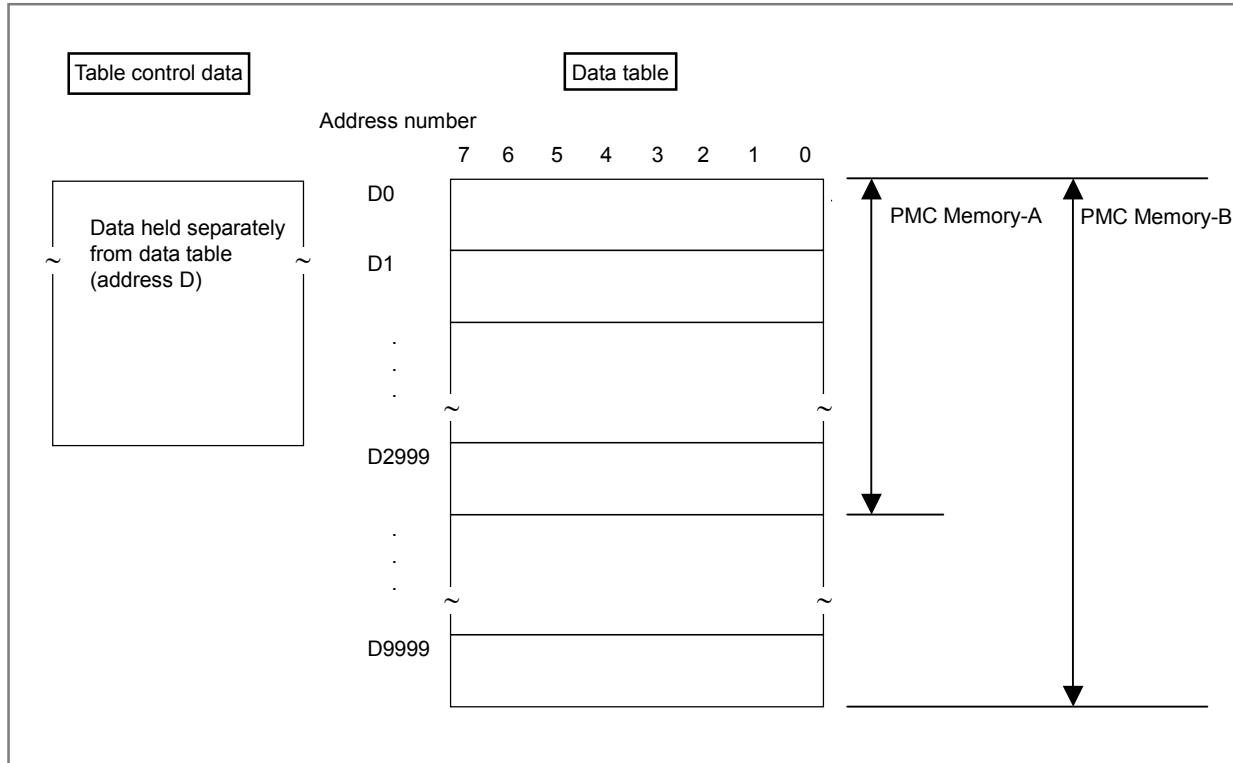
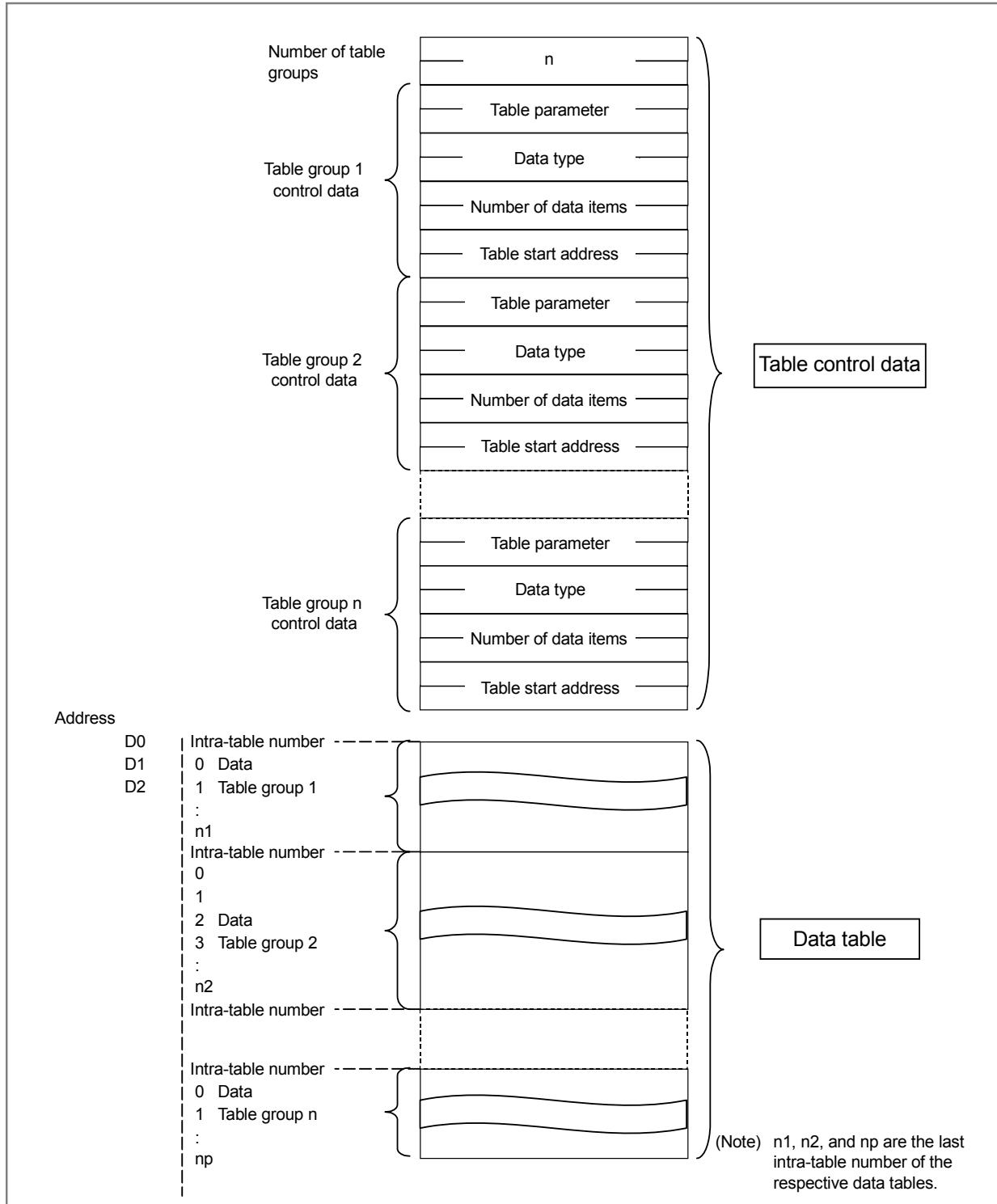


Fig. 2.2.11 (a) General configuration of data table

NOTE

In some cases, the start address of a data table is odd. If an odd number of 1-byte data tables are created, for example, the start address of the next data table may be odd. This setting is acceptable. However, an even start address assures faster operations than an odd start address. We recommend you use even start addresses whenever possible.

**Fig. 2.2.11 (b) Detailed configuration of data tables****(2) Table control data**

The table control data is used to manage data tables.

Unless this data is correctly set up, it is impossible to create data tables, explained in (3), correctly. While referencing the descriptions in this item, first set up table control data and then data tables.

(a) Number of table groups

This item specifies how many groups are to form the data table, using a binary number.

(b) Table group 1 control data to table group n control data

Each data table is provided with table control data. The meaning of data (table start address, table parameter, data type, and the number of data items) set up as table control data is the same for all table groups.

(i) Table start address

This item specifies the start address of a data area used for each data table.

(ii) Table parameter

#7	#6	#5	#4	#3	#2	#1	#0
				SIGN	HEX	MASK	COD

COD

0: Data in the data table is in binary form.

1: Data in the data table is in BCD form.

MASK

0: The contents of the data table is not protected.

1: The contents of the data table is protected.

HEX

0: Data in the data table is in binary or BCD form.

1: Data in the data table is in HEX form.

SIGN

0: Data in the data table is signed.

1: Data in the data table is unsigned.

NOTE

1 The setting of COD (bit 0) is valid if HEX (bit 2) = 0.

2 The setting of SIGN (bit 3) is valid if COD (bit 0) = 0 and HEX (bit 2) = 0.

(iii) Data type

This item specifies the length of data in the data table.

0 : 1 byte long

1 : 2 bytes long

2 : 4 bytes long

3 : 8 bits

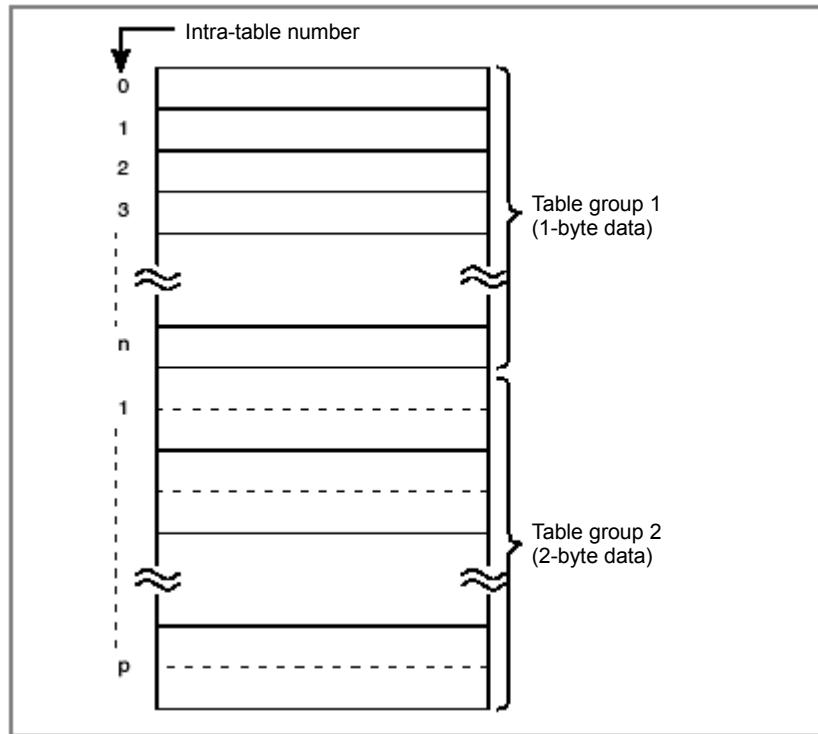
(iv) Number of data items

This item specifies the number of data items in the data table.

(3) Data table

A data table can be divided into several groups, and each group can be created within the memory range (address D) for the data table.

The number of groups is determined according to the number of table control data table groups.



Data in each data table can be 1-, 2, or 4-byte data depending on the data type of the corresponding table control data.

If the table data is 1-byte data, one intra-table number in the corresponding data table is assigned to one byte of data. If the table data is 2-byte data, one intra-table number is assigned to two bytes of data.

(4) Creating data for a data table

Data for a data table is created by specifying an intra-table number for the data table and entering the data into the table from the DATA TABLE screen. A specific method for specifying intra-table numbers is available for individual data table groups separately.

NOTE

The sequence program can also read and write the data table.

2.2.12 Addresses for Multi-path PMC Interface (M, N)

These addresses are used to the Multi-path PMC interface.

- (1) Input signals from another PMC path
The following addresses are available.

Table 2.2.12 Address of input signals from another PMC path

Data kind	1st to 3rd path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Input signals from another PMC path (Size)	M0 ~ M767 (768 bytes)			

- (2) Output signals to another PMC path
The following addresses are available.

Table 2.2.12 Address of output signals to another PMC path

Data kind	1st to 3rd path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Output signals to another PMC path (Size)	N0 ~ N767 (768 bytes)			

NOTE

These interfaces cannot be used in 4th and 5th path PMC.

2.2.13 Subprogram Number Addresses (P)

These addresses are used to specify jump destination subprogram labels in the CALL, CALLU and CM instructions.

Each subprogram number must be unique in the entire sequence program.

The following table lists the number of subprograms that can be used.

Table 2.2.13 Address of Subprogram number

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Subprogram number (Size)	P1 ~ P5000 (5,000)	P1 ~ P5000 (5,000)	P1 ~ P5000 (5,000)	P1 ~ P5000 (5,000)

2.2.14 Label Number Addresses (L)

These addresses are used to specify jump destination labels (positions within the sequence program) in the JMPB and JMPC instructions.

The same label number can be specified for different instructions as long as the instructions are not within the same program unit (main program or subprogram).

The following table lists the number of labels that can be used.

Table 2.2.14 Address of Label

Data kind	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Label number (Size)	L1 ~ L9999 (9,999)	L1 ~ L9999 (9,999)	L1 ~ L9999 (9,999)	L1 ~ L9999 (9,999)

2.3 PMC PARAMETERS

The term "PMC parameter" refers to any of the timer, counter, keep relay parameters, and data table. PMC parameters are held in nonvolatile memory, whose contents are not lost even when the power is turned off.

(1) Timer

This parameter specifies a timer value.

It is possible to set and display the timer value on the TIMER screen.

The sequence program can read and write the timer setting.

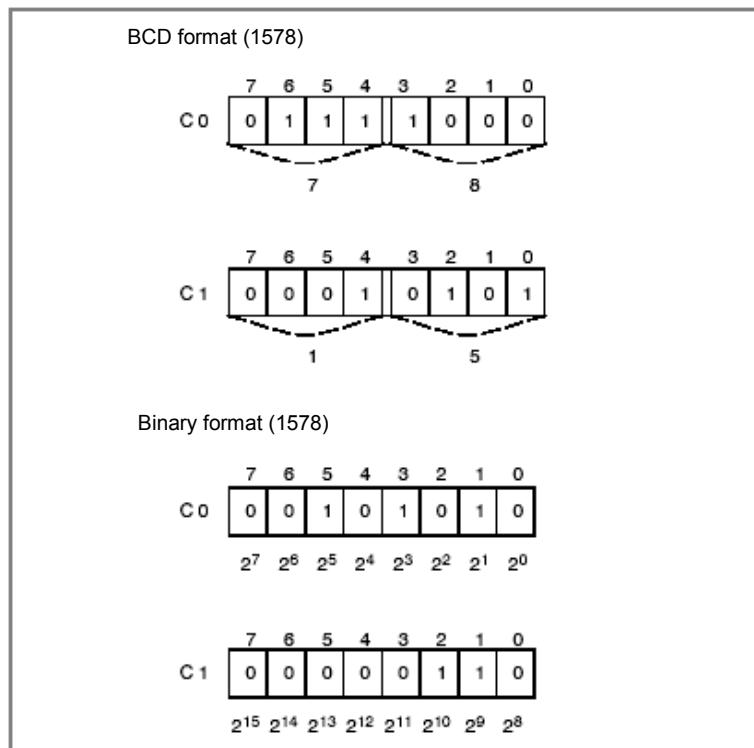
(2) Counter

This parameter is used for a counter preset value and cumulative value. It is possible to set and display these values on the COUNTER screen. Sequence program instructions can also read and write these settings. See Subsection 2.2.8 for details of the counter addresses.

Counter data is two bytes in either BCD or binary form. Higher-order bits are held at higher addresses. Whether the counter address is BCD or binary is determined according to the corresponding PMC system parameter.

The default setting is binary form.

(Example) If the counter addresses of the PMC are C0 and C1, and the preset value is 1578



To change the lower digit of the preset value to a certain value, using a 1-byte processing instruction in the sequence program, write the new data by specifying C0 with an output address in the parameter of a functional instruction.

(3) Keep relay

This parameter is used for parameters for sequence control, keep relays, and others.

It can be set and displayed from the KEEP RELAY screen.

It can also be read and written, using instructions in the sequence program.

The data set up or displayed from the KEEP RELAY screen is 8-bit binary data. On the KEEP RELAY screen, therefore, each of the eighth digits is set or displayed as 0 or 1.

(4) Data table

The data table enables a set of numeric data (data table) to be used for PMC sequence control.

See Subsection “2.2.12”for details.

(5) Extra relay

Extra relays are volatile memory, which can be used as extension of ordinary internal relays, or as common memory for the interface of PMC paths. Because the extra relays is non-volatile, and you can also use them as extension of data table or keep relays.

See “2.2.5 Extra Relay Addresses (E)” for more detail about extra relays.

2.3.1 Cautions for Reading from/Writing to Nonvolatile Memory

All data in the nonvolatile memory can be read and written with the sequence program. The memory from which the sequence program reads and to which it writes is not nonvolatile in effect. It has the same data as in the nonvolatile memory in a form of nonvolatile memory image (RAM). For this reason, turning off the power lets the data of nonvolatile memory image disappear. However, data is sent from the nonvolatile memory as nonvolatile memory image immediately after the power is resumed, thus restoring the previous data correctly.

If the sequence program rewrites the nonvolatile memory image, the changed data is automatically sent to the nonvolatile memory.

Data at more than one address in the nonvolatile memory image can be rewritten at any time. The changed data is automatically sent to the nonvolatile memory.

Therefore, reading from and writing to the nonvolatile memory with the sequence program does not require any special processing. Writing to the nonvolatile memory takes time (about 200 msec), however.

2.3.2 PMC Parameter Format

This subsection describes the format used in outputting the contents of the PMC parameter to an external device. As for the operation of output, refer to section 7 "sequence program and PMC parameter I/O".

(1) Header information

The data begins with header information. Its format is as follows:

[Format]

%

(PMC = xxx, MSID = n)

PMC = xxx "xxx" is the model name of the PMC.

MSID = n "n" is ID information.

The following table lists values that can be set as "xxx" or "n".

PMC Series	"xxx"
R-30iB ROBOT PMC	R-30IB

PMC Path	"n"
1st path PMC	1
2nd path PMC	2
3rd path PMC	3
4th path PMC	4
5th path PMC	5

(2) Timer (T)

[Format]

N60xxxx Pnnnnn;

N600xxxx Pnnnnn;

[Data Contents]

N60xxxx or N600xxxx : parameter number

Specify the sum of the timer address (T) offset and 600000 or 6000000. The number in the following table can be used.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Timer setting value	N600000 to N600078	N600000 to N600498	N600000 to N600998	N6000000 to N6000998
Timer accuracy	N609000 to N609078	N609000 to N609498	N609000 to N609998	N6009000 to N6009998

Pnnnnn The numbers from N600000 mean the timer value

The unit of the value depends on the timer accuracy which are numbers from N609000.

For example, when the timer accuracy is 3(100ms) and this value is 5, the timer value means 500ms. The range of effective value is from 0 to 32767.

The numbers from N609000 mean the timer accuracy. Each value is the following timer accuracy.

Value	Timer accuracy
0	Timer number1~8 : 48ms Timer number9~ : 8ms
1	1ms
2	10ms
3	100ms
4	1 sec.
5	1 min.

(Example)

N600000 P1; (Timer number 1 T0)
N600002 P20; (Timer number 2 T2)

N600498 P32767; (Timer number 250 T498)

N609000 P0; (Timer number 1 T9000)
N609002 P0; (Timer number 2 T9002)

N609498 P0; (Timer number 250 T9498)

(3) Counter (C)

[Format]

N61xxxx Pnnnnn;
N610xxxx Pnnnnn;

[Data Contents]

N61xxxx or N610xxxx; parameter number

Specify the sum of the counter address (C) offset and 610000 or 6100000. The number in the following table can be used.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Variable counter (CTR)	N610000 to N610078	N610000 to N610398	N610000 to N610798	N6100000 to N6101198
Fixed counter (CTRB)	N615000 to N615038	N615000 to N615198	N615000 to N615398	N6105000 to N6105598

Pnnnnn Counter address value in decimal notation

For the variable counter, the preset and current values appear alternately. For the fixed counter, only the current values appear.

It has a size of 2 bytes and can range from 0 to 32767 for a range. The counter addresses are assumed to be binary for input/output no matter whether the counter data type is specified as BCD or binary.

(Example)

N610000 P7;	(Counter number 1	C0)
N610002 P7;	(C2)
N610396 P9999;	(Counter number 100	C396)
N610398 P0;	(C398)
N615000 P7;	(Fixed-counter number 1	C5000)
N615002 P20;	(Fixed-counter number 2	C5002)
N615198 P9999; (Fixed-counter number 100 C5198)		

(4) Keep relay (K)

[Format]

N62xxxx Pnnnnnnnn;
N620xxxx Pnnnnnnnn;

[Data Contents]

N62xxxx or N620xxxx Parameter number

Specify the sum of the keep relay address (K) offset and 620000 or 6200000. The number in the following table can be used.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
User area	N620000 to N620019	N620000 to N620099	N620000 to N620199	N6200000 to N6200299
System area	N620900 to N620999	N620900 to N620999	N620900 to N620999	N6200900 to N6200999

Pnnnnnnnn Keep relay address value in binary notation

It can range from 00000000 to 11111111.

(Example)

N620000 P00000000;	(K0)
N620001 P11111111;	(K1)
N620099 P10101010;	(K99)
N620900 P00000000;	(K900)
N620901 P11111111;	(K901)
N620999 P10101010;	(K999)

(5) Data (D)

(a) Data table control

[Format]

N630xxx Pnnnnn;

N6300xxx Pnnnnn;

[Data Contents]

N630xxx or N6300xxx

Parameter Number

Parameter number		Contents
PMC Memory-A,B,C,	PMC Memory-D	
N630000	N6300000	The group number
N630002	N6300002	Parameter of group1
N630003	N6300003	Data type of group 1
N630004	N6300004	Data size of group 1 (byte)
N630006	N6300006	Start address of group 1
N630010	N6300010	Parameter of group2
N630011	N6300011	Data type of group 2
N630012	N6300012	Data size of group 2 (byte)
N630014	N6300014	Start address of group 2
...
N630002 + ((n-1) × 8)	N6300002 + ((n-1) × 8)	Parameter of group n
N630003 + ((n-1) × 8)	N6300003 + ((n-1) × 8)	Data type of group n
N630004 + ((n-1) × 8)	N6300004 + ((n-1) × 8)	Data size of group n (byte)
N630006 + ((n-1) × 8)	N6300006 + ((n-1) × 8)	Start address of group n
...
N630794	N6300794	Parameter of group 100
N630795	N6300795	Data type of group 100
N630796	N6300796	Data size of group 100 (byte)
N630798	N6300798	Start address of group 100

Pnnnnn Control data table address value.

The range of "The group number" is 1 to 100.

The range of "Table parameter" is 00000000 to 11111111.

The range of "Data type" is as follows.

Data type	Data table output format	Range
0	1 byte signed decimal number	-128 to 127
1	2 byte signed decimal number	-32768 to 32767
2	4 byte signed decimal number	-2147483648 to 2147483647
3	Binary notation	00000000 to 11111111

The range of "Data size" and "Start address" is as follows.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Data Size	1 to 3000	1 to 10000	1 to 20000	1 to 60000
Start address	0 to 2999	0 to 9999	0 to 19999	0 to 59999

(Example)

N630000 P2;
N630002 P00000000;
N630003 P0;
N630004 P10;
N630006 P0;
N630010 P00000001;
N630011 P0;
N630012 P10;
N630014 P10;

(b) Data table

[Format]

N64xxxx Pnnnnn;
N64xxxxx Pnnnnn;

[Data Contents]

N64xxxx to N65xxxx or N64xxxxx Parameter number

Specify the sum of the data table address (D) offset and 640000 or 6400000. The number in the following table can be used.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Data table	N640000 to N642999	N640000 to N649999	N640000 to N659999	N6400000 to N6459999

Pnnnnn Data table address value

Its size depends on the "data type" of data table control data.

Data type	Data table output format	Range
0	1 byte signed decimal number	-128 to 127
1	2 byte signed decimal number	-32768 to 32767
2	4 byte signed decimal number	-2147483648 to 2147483647
3	Binary notation	00000000 to 11111111

When the address is not used for data table control data, data table address value are stored as byte data.

(Example)

N640000 P-128;
N640001 P100;
N640002 P0;

N640010 P1000;
N640012 P-1;

N649992 P50000000;
N649996 P50000000;

(6) Extra memory (E)

All extra relay are stored as byte data.

NOTE

Extra relays (E) can be included only in PMC parameter output from 1st PMC.

The details of these formats are as follows:

[Format]

N69xxxx Pnnnnn;
N690xxxx Pnnnnn;

[Data Contents]

N69xxxx or N690xxxx Parameter number

Specify the sum of the offset number of the extra relay and 690000 or 6900000. The number in the following table can be used.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Extra relay	N690000 to N699999	N690000 to N699999	N690000 to N699999	N6900000 to N6909999

Pnnnnn The value of the extra relay

It is shown in signed decimal number. Its valid range is -128 to 127.

(Example)

N690000 P-128;
N690001 P100;
•
N699998 P127;
N699999 P0;
%

2.4 PARAMETERS FOR THE PMC SYSTEM

2.4.1 Setting Parameters

The parameters set up on the PMC SETTING screen are called the setting parameters. Part of the system keep relays described earlier can be set up using setting parameters. This subsection describes the setting parameters.

- (1) Memory write permit (K900.4 0: No. 1: Yes.)

This item specifies whether to enable the forcing and override functions. The default setting is "Yes" (not to enable).

NOTE

Using the override function requires setting "Override enable" (K906.0).

- (2) Data table GRP setting display (K900.7 0: Yes. 1: No.)

This item specifies whether to display the DATA TABLE CONTROL screen. The default setting is "Yes" (to display).

- (3) PMC stop enable (K902.2 0: No. 1: Yes.)

This item specifies whether to allow the sequence program to start/stop. The default setting is "No" (not to allow).

- (4) Programmer function enable (K900.1 0: No. 1: Yes.)

This item specifies whether to enable the built-in programmer function. The default setting is "Yes" (not to enable).

- (5) Override enable (K906.0 0: No. 1: Yes.)

This item specifies whether to enable the override function. The default setting is "No" (not to enable).

NOTE

Using the override function requires setting "Memory write permit" (K900.4).

2.5 COMPATIBILITY BETWEEN PMC MEMORY TYPE

2.5.1 Compatibility between PMC Memory-A and PMC Memory-B

The sequence program for the PMC Memory-B is compatible with the PMC Memory-A.

You can convert a sequence program for the PMC Memory-A to one for the PMC Memory-B by using the conversion function of FANUC LADDER-III for Robot.

WARNING

A little difference of execution timing may exist between PMC Memory Types. Therefore, you should check again whether the program works correctly after changing the PMC Memory Types even if the program worked fine before changing the PMC Memory Type.

PMC parameter compatibility

PMC parameters outputted from on the PMC Memory-A can be loaded into the PMC Memory-B without any modification.

When loading PMC parameters outputted on the PMC Memory-B to the PMC Memory-A, the data outside the address range will be lost.

2.5.2 Compatibility between PMC Memory-B and PMC Memory-C/D

The sequence program for the PMC Memory-C and PMC Memory-D are compatible with the PMC Memory-B.

A sequence program for the PMC Memory-B can be used on the PMC Memory-C or PMC Memory-D by converting PMC type and changing addresses R9000 - R9499 to Z0 - Z499 with FANUC LADDER-III for Robot.

WARNING

A little difference of execution timing may exist between PMC Memory Types. Therefore, you should check again whether the program works correctly after changing the PMC Memory Types even if the program worked fine before changing the PMC Memory Type.

CAUTION

- 1 In the PMC Memory-C or PMC Memory-D, the System Relay is changed to Z0-Z499 from R9000-R9499 because the size of Internal Relay (R Address) is expanded.
- 2 In the PMC Memory-C or PMC Memory-D, the system used area increases 8KB from PMC Memory-B. Therefore, available memory size for Symbol, Comment and Message data is smaller than the PMC Memory-B. If the program overflowed by converting PMC Memory Type, please decrease the size of Symbol, Comment or Message data, or upgrade the Ladder step option to larger size.

PMC parameter compatibility

PMC parameters outputted from PMC Memory-B can be loaded into the PMC Memory-C or PMC Memory-D without any modification.

When loading PMC parameters outputted from PMC Memory-C or PMC Memory-D to the PMC Memory-B, the data outside the address range will be lost.

2.5.3 Compatibility with PMC Memory-C and PMC Memory-D

The sequence program for the PMC Memory-D are compatible with the PMC Memory-C. You can convert a sequence program for the PMC Memory-C to one for the PMC Memory-D using the conversion function of FANUC LADDER-III for Robot.

WARNING

A little difference of execution timing may exist between PMC Memory Types. Therefore, you should check again whether the program works correctly after changing the PMC Memory Types even if the program worked fine before changing the PMC Memory Types.

PMC parameter compatibility

PMC parameters outputted from the PMC Memory-C can be loaded into the PMC Memory-D without any modification.

When loading PMC parameters outputted from the PMC Memory-D to the PMC Memory-C, the data outside the address range will be lost.

3 PMC I/O ASSIGNMENT

3.1 I/O ASSIGNMENT ON ROBOT CONTROLLER

There are the following I/O assignment related to PMC.

- **PMC external I/O assignment**
Assign input and output signals in the external I/O device to PMC address.
- **PMC internal I/O assignment**
Assign input and output signals in the robot such as DI/O to PMC address.

In addition to them, robot can input and output with the external I/O device independent with PMC by assigning the external I/O device to DI/O. This is the ordinary I/O assignment that is also used when the Integrated PMC option is not ordered. It is called as "I/O assignment (Robot)" to distinguish from PMC related I/O assignment.

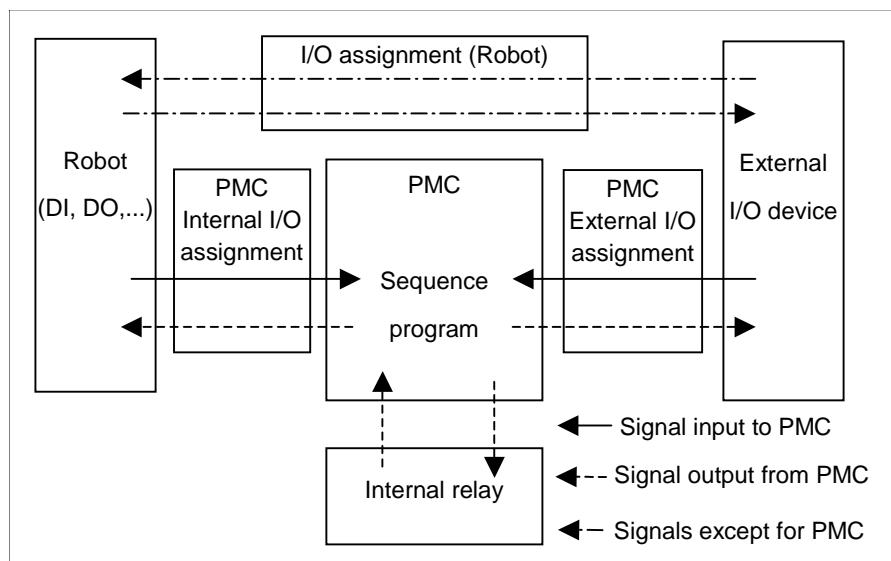


Fig. 3.1 I/O assignment on robot controller

NOTE

The assignment of external I/O device to PMC address is defined by PMC I/O assignment. The setting in "I/O module editing" menu of FANUC LADDER-III for Robot is not used.

3.2 PMC EXTERNAL I/O ASSIGNMENT

PMC can access the external I/O device by the setting of PMC external I/O assignment.

PMC external I/O assignment defines the relationship between signals on the external I/O device (identified by rack and slot) and PMC signal address (X, Y, R).

The PMC external I/O assignment can be setup in the PMC external I/O assignment menu (MENU → "I/O" → F1"[TYPE]" → "PMC" → F2"[DATA]" → "Ext. I/O"). Please refer "6 TEACH PENDANT OPERATION" for detail operation.

PMC1 <RUNNING>					
External I/O assignment					
Type	Rack	Slot	Size	Addr	Occpy
1 DI	0	1	12	1:X00000	NO
2 DO	0	1	12	1:Y00000	YES
3 RI	0	0	1	1:X01000	NO
4 RO	0	0	1	1:Y01000	NO
5 --	0	0	0	1:-00000	NO
6 --	0	0	0	1:-00000	NO

[TYPE] [DATA] [FUNC] CLR_ALL

A signal type is set in the "Type" column. The table 3.2 "Signal type of external I/O assignment" shows the detail of each signal type.

External I/O assignment is defined by the setup of "Type", "Rack", "Slot" and "Addr" column. The "Size" column shows the Byte size of PMC address to be assigned. The "Size" is determined automatically according to the signal type and the number of ports of the external I/O device, and it cannot be changed. When the specified external I/O device is not connected, the "Size" shows 0.

For example, when Process I/O board JA is connected to rack 0, slot 1, digital input signals 96 points (size is 12 byte) are assigned to the PMC signal address X0-X11 of 1st path PMC by the following setting.

Type	Rack	Slot	Size	Addr	Occpy
1 DI	0	1	12	1:X00000	NO

Occupancy setting of external I/O device on this line is enabled when "Occpy" item is "YES". I/O signals of this device are accessed from only PMC sequence programs. Please refer "3.2.3 Occupancy setting" for further information

Table 3.2 Signal type of external I/O assignment

Type	Name	Description
DI	Digital Input	<p>Digital input signal of the external I/O device. The external I/O device is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 digital input signal is assigned to 1 bit of PMC address. - Can be assigned to X or R. - When assigned to X0, the correspondence with the START in I/O assignment is the following. <p>START 1 : X0.0 START 2 : X0.1 : START 8 : X0.7 START 9 : X1.0 :</p>
DO	Digital Output	<p>Digital output signal of the external I/O device. The external I/O device is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 digital output signal is assigned to 1 bit of PMC address. - Can be assigned to Y or R. - When assigned to Y0, the correspondence with the START in I/O assignment is the following. <p>START 1 : Y0.0 START 2 : Y0.1 : START 8 : Y0.7 START 9 : Y1.0 :</p>

Type	Name	Description
AI	Analog Input	<p>Analog input signal of the external I/O device. The external I/O device is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 analog input signal is assigned to 2 byte of PMC address. - Can be assigned to X or R. - When assigned to X0, the correspondence with the CHANNEL in I/O assignment is the following. <p>CHANNEL 1 : X0-1 (X0:Lower BYTE, X1:Upper BYTE) CHANNEL 2 : X2-3 (X2:Lower BYTE, X3:Upper BYTE)</p> <p>:</p>
AO	Analog Output	<p>Analog output signal of the external I/O device. The external I/O device is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 analog output signal is assigned to 2 byte of PMC address. - Can be assigned to Y or R. - When assigned to Y0, the correspondence with the CHANNEL in I/O assignment is the following. <p>CHANNEL 1 : Y0-1 (Y0:Lower BYTE, Y1:Upper BYTE) CHANNEL 2 : Y2-3 (Y2:Lower BYTE, Y3:Upper BYTE)</p> <p>:</p>
RI	Robot Input	<p>Robot input signal. Rack and slot are always 0.</p> <ul style="list-style-type: none"> - 1 input signal is assigned to 1 bit of PMC address. - Can be assigned to X. - When assigned to X0, the correspondence with RI[] is the following. <p>RI[1] : X0.0 RI[2] : X0.1 : RI[8] : X0.7 RI[9] : X1.0</p> <p>:</p>
RO	Robot Output	<p>Robot output signal. Rack and slot are always 0.</p> <ul style="list-style-type: none"> - 1 output signal is assigned to 1 bit of PMC address. - Can be assigned to Y. - When assigned to Y0, the correspondence with RO[] is the following. <p>RO[1] : X0.0 RO[2] : X0.1 : RO[8] : X0.7 RO[9] : X1.0</p> <p>:</p>
SI	SOP Input	<p>SOP input signal. Rack and slot are always 0.</p> <ul style="list-style-type: none"> - 1 input signal is assigned to 1 bit of PMC address. - Can be assigned to X. - When assigned to X0, the correspondence with SI[] is the following. <p>SI[0] : X0.0 SI[1] : X0.1 : SI[7] : X0.7 SI[8] : X1.0</p> <p>:</p>

Type	Name	Description
SO	SOP Output	<p>SOP output signal. Rack and slot are always 0.</p> <ul style="list-style-type: none"> - 1 output signal is assigned to 1 bit of PMC address. - Can be assigned to Y. - When assigned to Y0, the correspondence with SO[] is the following. <p style="margin-left: 20px;">SO[0] : Y0.0 SO[1] : Y0.1 : SO[7] : Y0.7 SO[8] : Y1.0 :</p>
WI	Welding interface digital Input	<p>Welding interface digital input signal of the process I/O board. The process I/O board is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 input signal is assigned to 1 bit of PMC address. - Can be assigned to X. - When assigned to X0, the correspondence with WI number in the Maintenance Manual is the following. <p style="margin-left: 20px;">WI01 : X0.0 WI02 : X0.1 : WI08 : X0.7</p>
WO	Welding interface digital Output	<p>Welding interface digital output signal of the process I/O board. The process I/O board is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 output signal is assigned to 1 bit of PMC address. - Can be assigned to Y. - When assigned to Y0, the correspondence with WO number in the Maintenance Manual is the following. <p style="margin-left: 20px;">WO01 : Y0.0 WO02 : Y0.1 : WO08 : Y0.7</p>
WSI	Wire Soldering detector Input	<p>Wire soldering detector input signal of the process I/O board. The process I/O board is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 input signal is assigned to 1 bit of PMC address. - Can be assigned to X. - When assigned to X0, the wire soldering detector input is assigned to X0.0.
WSO	Wire Soldering detector Output	<p>Wire soldering detector output signal of the process I/O board. The process I/O board is specified by rack and slot.</p> <ul style="list-style-type: none"> - 1 output signal is assigned to 1 bit of PMC address. - Can be assigned to Y. - When assigned to Y0, the wire soldering detector output is assigned to Y0.0.

The specification of PMC external I/O assignment is the following.

- Input signal of the external I/O device can be assigned by X or R signal address. Output signal of the external I/O device can be assigned to Y or R signal address. X and Y signal address can be assigned to the external I/O device that the size is 128 byte (1024 bit) or less. To assign the external I/O device that the size is greater than 128 byte, please use R signal address. R signal address can be assigned to external I/O device that the size is 32 byte (256 bit) or more.
- In the PMC external I/O assignment, all I/O signals of the specified signal type (DI, DO, etc) in the external I/O device are assigned to the PMC signal address. For example, when the external I/O device has 40 points digital input, all of 5 byte (40 bit) signals are assigned. It is not allowed to assign a part of signals (Ex. Only 1 byte in the 5 byte signals). It is allowed to assign one of input signals (DI) or output signals (DO) when the external I/O device has both input signals and output signals.

- The top of the signal address must be multiple of 2 (Ex. X0 and X4 are allowed. X1 and X5 are not allowed). Some I/O devices allows only multiple of 32.
- It is not allowed to assign more than one external I/O devices to the same signal address.

NOTE

- 1 When the external I/O device is assigned to R signal address, it is possible to output (specify as coil) from sequence program. But please do not output to the signal address because it is input signal.
- 2 When the external I/O device is assigned to R signal address, the synchronization processing of I/O signals for 2nd level sequence program does not work for the signals. (Please refer "1.4.5 Synchronization Processing of I/O Signals".)
- 3 When the external I/O device is assigned to R signal address, override function does not work for the signals. (Please refer "3.2.2 Override Function".)
- 4 PMC path number that Memory type is Common cannot be used. In this case, please set the 1st path PMC.

3.2.1 Relationship with I/O Assignment (Robot)

The PMC external I/O assignment and the I/O assignment (Robot) are completely independent settings. Change of the PMC external I/O assignment does not affect to the relationship between DI/O and the external I/O device. On the other hand, the I/O assignment (Robot) does not affect to the relationship between PMC signal address and the external I/O device.

It is possible to setup one external I/O device in either the PMC external I/O assignment or the I/O assignment (Robot), and it is also possible to setup one external I/O device in both.

The following figure is the example that input signals on one external I/O device are assigned to both the PMC external I/O assignment and the I/O assignment (Robot). When the input signal 1 becomes ON, X0.0 becomes 1 and DI[1] is also becomes ON at the same time.

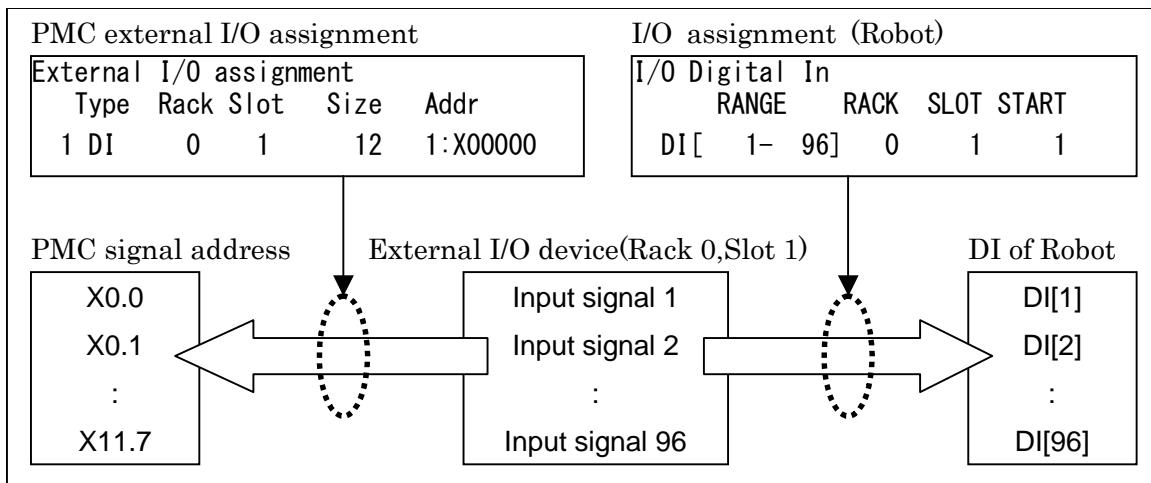


Fig. 3.2.1(a) The PMC external I/O assignment for input signal

The following figure is the example that output signals one external I/O device are assigned to both the PMC external I/O assignment and the I/O assignment (Robot).

When Y0.0 is set to 1(0), output signal 1 becomes ON(OFF). At the same time, DO[1] also become ON(OFF).

In addition, When DO[1] is set to ON(OFF), output signal 1 becomes ON(OFF). At the same time, Y0.0 also becomes 1(0).

If both Y0.0 and DO[1] do output at the same time, the output signal 1 becomes the value that was set by the later process. Because PMC sequence program does the output processing at every scanning period, when the sequence program outputs to Y0.0, the value of output signal 1 is overwritten by the value set by sequence program even though the robot changes DO[1].

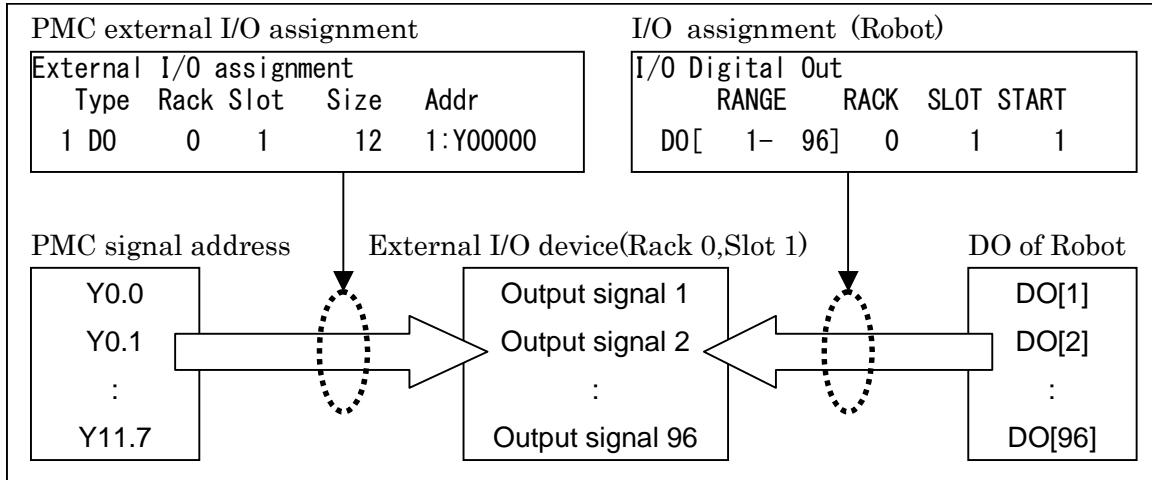


Fig. 3.2.1(b) The PMC external I/O assignment for output signal

⚠ CAUTION

The following functions for I/O of Robot do not affect to the corresponded PMC signal.

- Even though the I/O of Robot is simulated, the corresponded PMC signal accesses actual signal of the external I/O device.
- Even though the I/O of Robot is set to reverse polarity, the corresponded PMC signal status is not reversed.
- Even though the I/O of Robot is set to complementary, the output to the corresponded PMC signal does not work as complementary.
- PMC number that Memory type is Common cannot be used as Internal I/O address. Please set first path PMC as PMC number of Internal I/O.

3.2.2 Override Function

The I/O of Robot such as DI/O can be set as simulation, and the I/O can be used for the test of robot program without input/output to the external I/O device. The simulated DI can have the different value from the assigned external I/O device. And the value of the assigned output signals on external I/O device is not changed even though the simulated DO is changed.

The override function provides the similar functionality in PMC. By using the override function, the PMC signal address can be used for the test of sequence program without input/output to the external I/O device. The overridden PMC signal address can have the different value from the assigned external I/O device. And the value of the assigned output signals on external I/O device is not changed even though the PMC sequence programs change overridden PMC signal. The override function can be output different value between real signals that are assigned to external I/O device and Y address signals that are used by PMC sequence.

- To use the override function, please set 1 to the override enable (K906.0) of the PMC path that the override function is used, then cycle power. After that, please set override for each PMC signal address in PMC menu.
- Override function is available for the signals of external I/O devices that are assigned to X or Y address. It is not available for the signals of external I/O devices that are assigned to R address.

Override setting and value are changed in PMC screen. Overridden value can be changed when value is input on the address item. The '*' is displayed at left side of address value if the address is set override mode. In the following screen example, Y0.1-Y2.7 is set override mode. When value is input on this address area, overridden value is changed.

Please refer “6.1.1 Byte Menu” and “6.1.2 Bit Menu” for detail operation.

PMC 1 <RUNNING>								JOINT 100%			
X	BYTE	7	6	5	4	3	2	1	0	Hex	Dec
X0000		0	0	0	0	0	0	0	:	00	0
X0001	*	0	0	0	0	0	1	*	0*	05	5
X0002	*	0	0	0	0	0	1	*	0*	06	6
X0003		0	0	0	0	0	0	0	:	00	0
X0004		0	0	0	0	0	0	0	:	00	0
X0005		0	0	0	0	0	0	0	:	00	0
X0006		0	0	0	0	0	0	0	:	00	0
X0007		0	0	0	0	0	0	0	:	00	0
X0008		0	0	0	0	0	0	0	:	00	0
X0009		0	0	0	0	0	0	0	:	00	0
Comment:								BIT >			
[TYPE] [DATA] [FUNC]											

Fig. 3.2.2(a) PMC screen to change overridden value

⚠ CAUTION

- 1 The Override function is a special function for debugging the ladder program. Therefore, in the production, please disable the Override function invariably.
- 2 When using the Override function, the transfer of input/output signals with the external I/O device is delayed for the first level execution cycle of ladder program (4ms or 8ms). Do not use the Override function when you debug the ladder program that is affected by the delay of the I/O transfer.
- 3 The I/O setting values of the Override function are cleared when the power is turned off. Therefore, Override settings for all signals are reset when cycling the power.
- 4 The override function can be used in three paths PMC simultaneously. However, the use in single-path is recommended because of the influence of scan time. If you use this function in 4 or more path, the warning “PRIO-520 PMC:Invalid Override setting” occurs and the override mode is invalid in all PMC paths.

When one external I/O device is assigned by both I/O assignment (Robot) and PMC external I/O assignment, the relationship between the I/O simulation and the override function is the following.

- When I/O of Robot is set to simulation, PMC signal is not affected.
- When PMC signal is set to override, I/O of Robot is affected.

As shown in the following Fig. 3.2.2(b), simulation of robot I/O does not affect PMC signal because input data is exchanged when it is transferred to robot DI port. In other hand, overridden value affects to robot DI port, because override function exchange input data when it is written to input signal memory.

In example of Fig. 3.2.2(b), value of input signal memory can be changed without affecting of external device when overridden value of X0.0 is changed in PMC screen. Then, input signal 1 of input signal memory is changed and DI[1] is also changed. When DI[1] is simulation, simulation value of DI[1] is not changed even if overridden value of X0.0 is changed.

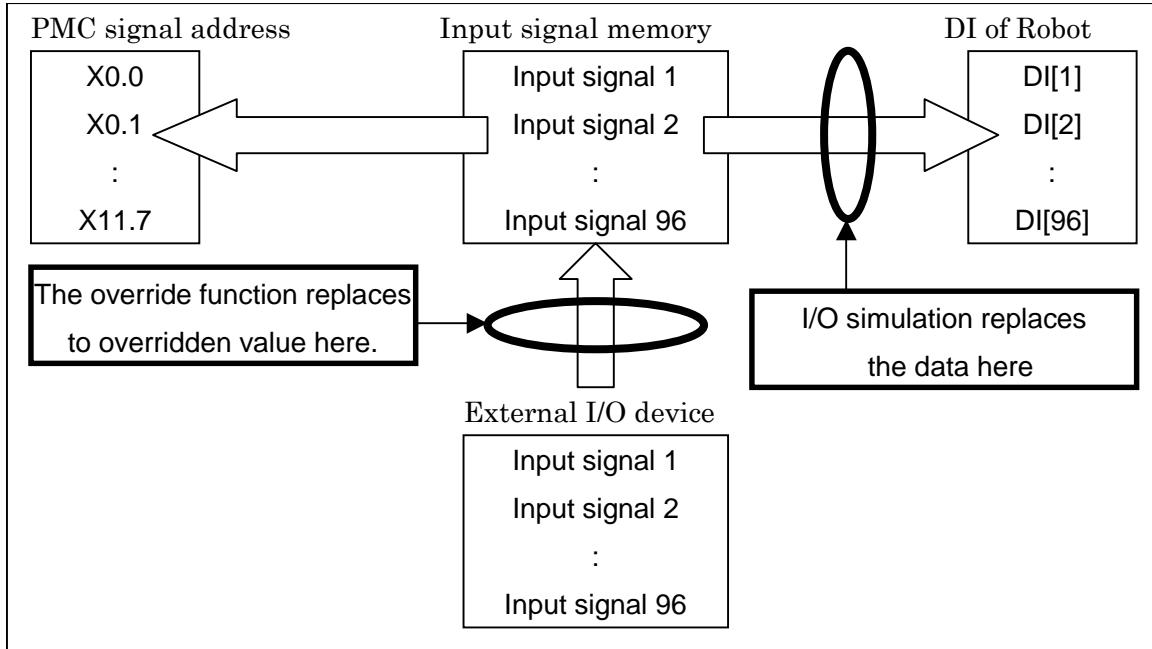


Fig. 3.2.2(b) Override function

In example of Fig. 3.2.2(c), when Y0.0 is set as override, and PMC sequence program sets Y0.0 value, then output signal 1 in the output memory is changed, but the output signal 1 in the external I/O device is not changed. In this case, when DO[1] is not set as simulation, and DO[1] is changed, then output signal 1 in the output memory is changed and Y0.0 value is changed. However, output signal 1 of the external I/O device is not changed because Y0.0 is set as override.

On the other hand, when DO[1] is set as simulation, and DO[1] is changed, then output signal 1 in the output memory is not changed. Therefore, neither Y0.0 nor output signal 1 in external I/O device is not changed.

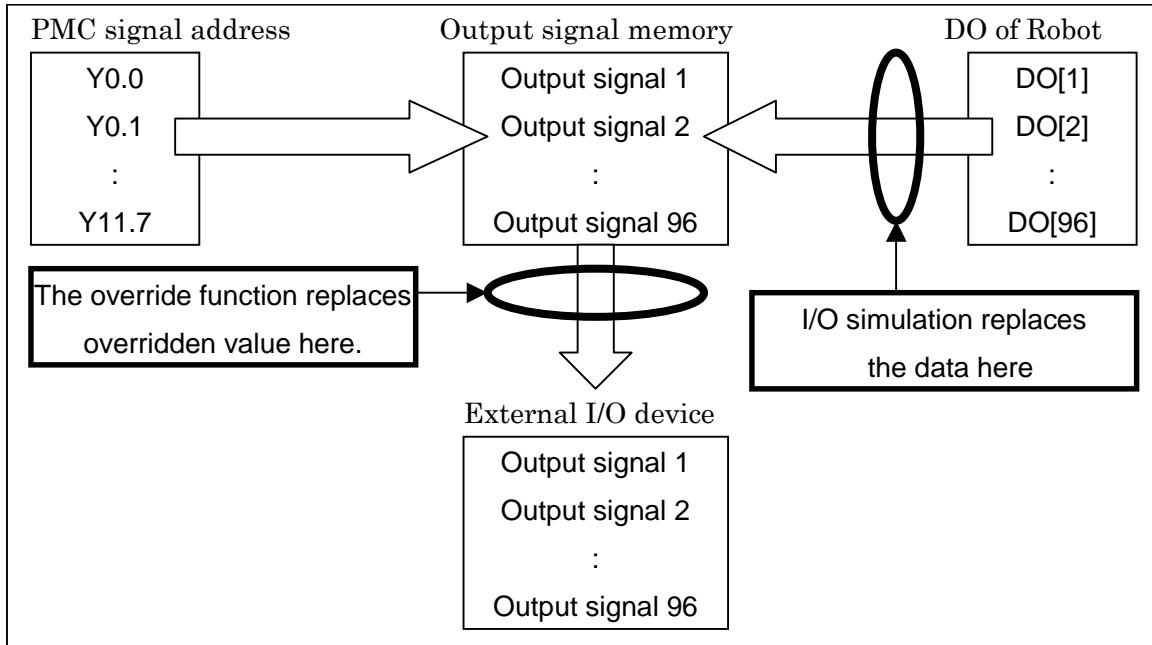


Fig. 3.2.2(c) Override function

3.2.3 Occupancy Setting

There are following advantages to use occupancy setting.

- Great number of I/O signals can be used.

Maximum I/O signals of DI/O are 4098/4098 points. More I/O signals can be used by occupancy setting. If number of I/O signal points is numerous, “PRIO-119 Too many DIGITAL I/O ports” occurs and External I/O devices cannot be recognized. In this case, this alarm does not occur and I/O devices can be used when occupancy setting is used.

- It can be blocked to access signals of the same address from Robot programs.

PMC sequence programs output signals with the scan cycle. Therefore, PMC programs sometimes overwrite signals even if robot program output its signals. Multi-access may causes any problems and it is difficult to investigate causes. For occupancy setting, signals of the device are occupied by PMC sequence. It can be blocked to access from any functions except PMC function.

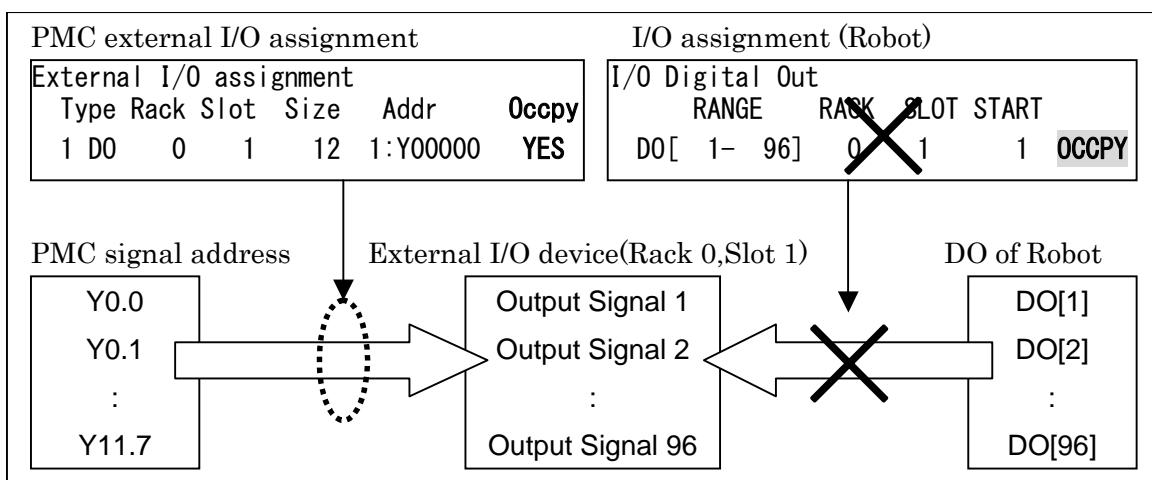


Fig. 3.2.3 Override function

When “Occpy” item is set “YES” in PMC external I/O assignment screen, these signals are set as occupancy for PMC. It can be enabled when IO “Type” in the screen is DI, DO, AI or AO. When “Type” is RI, RI, SI, SO, WI, WO, WSI or WSO, it cannot be enabled. Please refer “6.1.12 External I/O Assignment Screen” for information to set occupancy mode in screen.

Signals of occupancy mode cannot be assigned DI/O, UI/O, GI/O and AI/O ports of robot. To access these signals by robot programs, it is needed to transfer them to other address by PMC sequence programs. For I/O assignment screen, OCCPY is displayed on right side of screen if rack and slot of occupancy mode device is set. After re-power, “PRIO-063 Bad I/O asg” occur and the device is not assigned to I/O ports.

I/O Digital Out				
	RANGE	RACK	SLOT	START
1 DO[1- 96]	0	1	1	OCCPY

NOTE

- Power failure recovery is not performed for output signals of occupancy mode. All output signals of occupancy mode is changed OFF at re-power even if power failure recovery is enabled.
- Power failure recovery is performed for F address value that is assigned to DO, GO and AO ports. When signals of these F address are transferred to other output signals by PMC program, the recovered status of F address are reflected to the output signals.
- Power failure recovery is not performed for all output signals when any occupancy setting is changed.

3.3 PMC INTERNAL I/O ASSIGNMENT

PMC can input/output the data with robot program and robot function by the setting of PMC internal I/O assignment.

PMC internal I/O assignment defines the relationship between the Robot data (DI/O, GI/O, etc) and PMC signal address (X, Y, R).

The PMC internal I/O assignment can be setup in the PMC internal I/O assignment menu (MENU → "I/O" → F1"[TYPE]" → "PMC" → F2"[DATA]" → "Int. I/O"). Please refer "6 TEACH PENDANT OPERATION" for detail operation.

PMC1 <RUNNING>					
Internal I/O assignment			2msec		
	Robot data	Size	Address		
1	U0[1- 20]	8	1:F00400		
2	UI[1- 18]	8	1:G00400		
3	DO[1- 32]	4	1:F00000		
4	DI[1- 32]	4	1:G00000		
5	----[0- 0]	0	1:-00000		
6	----[0- 0]	0	1:-00000		
7	----[0- 0]	0	1:-00000		

[TYPE] [DATA] [FUNC] CLR_ALL

The PMC internal I/O assignment can be setup by setting the Robot data type and number in "Robot data" column, and the PMC path and the signal address in the "Address" column".

For example, when the following setting is done, the 32 points signal of DO[1-32] of Robot are assigned to the 4 byte PMC signal address of F0-F3 in 1st path PMC.

Robot data	Size	Address
3 DO[1- 32]	4	1:F00000

The following table shows the available combination between robot data and PMC signal address for the PMC internal I/O assignment. The detail of each combination is described later.

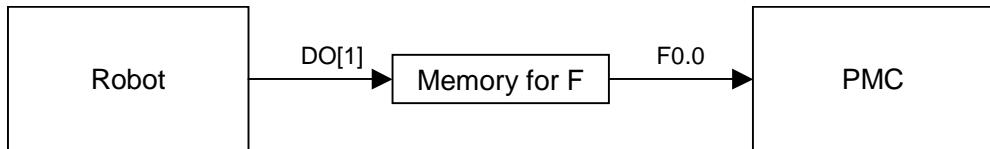
Table 3.3 Combinations between robot data and PMC signal address

Robot	PMC	Description
DO	F	Memory for F of PMC is assigned to DO of Robot. Bit data can be transferred from Robot to PMC.
DI	G	Memory for G of PMC is assigned to DI of Robot. Bit data can be transferred from PMC to Robot.
GO	F	Memory for F of PMC is assigned to GO of Robot. 2 byte data can be transferred from Robot to PMC.
GI	G	Memory for G of PMC is assigned to GI of Robot. 2 byte data can be transferred from PMC to Robot.
UO	F	Memory for F of PMC is assigned to UO of Robot. UOP output such as CMDENBL can be transferred from Robot to PMC.
UI	G	Memory for G of PMC is assigned to UI of Robot. UOP input such as START can be transferred from PMC to Robot.
INFO	F	Robot system information is written in memory for F of PMC. System information such as single step status can be read by F of PMC.
UALM	G	When bit data in G of PMC is 1, the corresponded user alarm occurs. PMC can generate user alarm.
R(Register)	F	Data is transferred from Robot register to F of PMC. PMC can read the robot register value by F.
R(Register)	G	Data is synchronized between G of PMC and Robot register. PMC can read the robot register value that is set by Robot, and Robot can read the robot register value that is set by PMC.
DI, RI, WI, WSI, SI, *UI	X	The X of PMC is assigned to the external I/O device that is assigned to DI, RI, WI, WSI, SI or *UI of Robot. This assignment is for compatibility with conventional models. When this assignment is used, X and Y address cannot be specified in the parameter of functional instructions, and PMC external I/O assignment is not available.
DO, RO, WO, WSO, SO, *UO	Y	The Y of PMC is assigned to the external I/O device that is assigned to DO, RO, WO, WSO, SO or *UO of Robot. This assignment is for compatibility with conventional models. When this assignment is used, X and Y address cannot be specified in the parameter of functional instructions, and PMC external I/O assignment is not available.
GI,AI	F	Data is transferred from GI or AI of Robot to F of PMC. This assignment is for compatibility with conventional models. This assignment is different from the assignment between GI and G, GO and F that are described in the above. The GI or AI needs to be assigned to the external I/O device in I/O assignment (Robot).
GO,AO	G	Data is synchronized between G of PMC and GO or AO of Robot. This assignment is for compatibility with conventional models. This assignment is different from the assignment between GI and G, GO and F that are described in the above. The GO or AO needs to be assigned to the external I/O device in I/O assignment (Robot).
DO[10001-10136] DO[10137-10160] DO[11001-23000] GO[10001-11500]	K0-16 K900-902 R0-1499 D0-2999	Memory for K, R, D of PMC is assigned to DO, GO of Robot. This assignment is for compatibility with conventional models. The assignment for K, R, D is available only for these assignment patterns. Any other assignment pattern (Ex. Assign K0 to DO[1-8]) is not available.

NOTE

PMC path number that Memory type is Common cannot be used. In this case, please set the 1st path PMC.

3.3.1 Robot : DO → PMC : F



Memory for F of PMC is assigned to digital output (DO) of Robot. PMC can read the value (ON/OFF) that the robot outputs as bit value (1/0) in F of PMC.

Example : The following is the correspondence when F0-1 is assigned to DO[1-16].

DO[1]	:	F0.0
DO[2]	:	F0.1
DO[3]	:	F0.2
DO[4]	:	F0.3
DO[5]	:	F0.4
DO[6]	:	F0.5
DO[7]	:	F0.6
DO[8]	:	F0.7
DO[9]	:	F1.0
DO[10]	:	F1.1
DO[11]	:	F1.2
DO[12]	:	F1.3
DO[13]	:	F1.4
DO[14]	:	F1.5
DO[15]	:	F1.6
DO[16]	:	F1.7

When the PMC internal assignment is setup as the above, the digital I/O assignment menu displays as follows. The mark "PMC" is displayed on the right end column, and the rack is always 33. The slot number and the start point number are defined automatically. This line cannot be changed nor deleted in the digital I/O assignment menu. To change this line, please change the setting in PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the digital I/O assignment menu at the next power up.

#	RANGE	RACK	SLOT	START	STAT.
1	DO[1- 16]	33	8	1	PMC

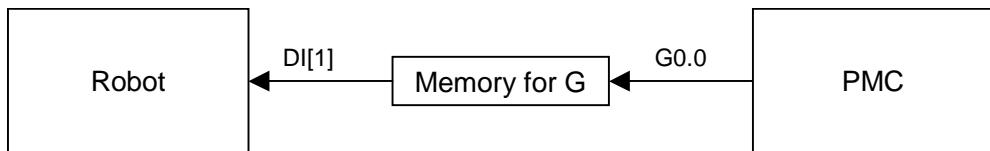
The specification of the assignment of F to DO is the following.

- The minimum unit of the assignment is 16 points for DO, 2 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- DO index can be set any number in the range that is displayed in the digital I/O menu.
- When the power failure recovery is enable, the status of the F signal that is assigned to DO is recovered at power up.
- When the DO is set as simulation, the value of F is not changed even though the value of DO is changed.

NOTE

The DO signal that is already assigned to the external I/O device cannot be assigned to F of PMC. Please delete the assignment in digital I/O assignment menu at first.

3.3.2 Robot : DI \leftarrow PMC : G



Memory for G of PMC is assigned to digital input (DI) of Robot. Robot can read the bit value (1/0) that the PMC outputs as value (ON/OFF) in DI of Robot.

Example : The following is the correspondence when G0-1 is assigned to DI[1-16].

DI[1]	:	G0.0
DI[2]	:	G0.1
DI[3]	:	G0.2
DI[4]	:	G0.3
DI[5]	:	G0.4
DI[6]	:	G0.5
DI[7]	:	G0.6
DI[8]	:	G0.7
DI[9]	:	G1.0
DI[10]	:	G1.1
DI[11]	:	G1.2
DI[12]	:	G1.3
DI[13]	:	G1.4
DI[14]	:	G1.5
DI[15]	:	G1.6
DI[16]	:	G1.7

When the PMC internal assignment is setup as the above, the digital I/O assignment menu displays as follows. The mark "PMC" is displayed on the right end column, and the rack is always 33. The slot number and the start point number are defined automatically. This line cannot be changed nor deleted in the digital I/O assignment menu. To change this line, please change the setting in PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the digital I/O assignment menu at the next power up.

#	RANGE	RACK	SLOT	START	STAT.
1	DI[1- 16]	33	9	1	PMC

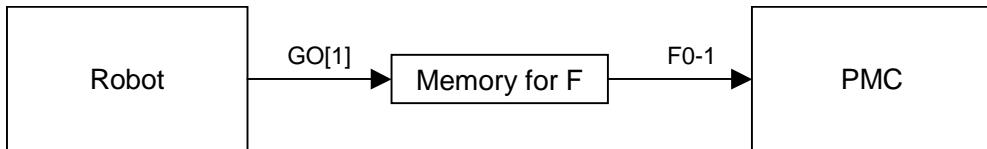
The specification of the assignment of G to DI is the following.

- The minimum unit of the assignment is 16 points for DI, 2 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- DI index can be set any number in the range that is displayed in the digital I/O menu.
- Unrelated to enable or disable of power failure recovery, all G signals that DI is assigned are set to 0 at power up.
- When the DI is set as simulation, the value of DI is not changed even though the value of G is changed.

NOTE

The DI signal that is already assigned to the external I/O device cannot be assigned to G of PMC. Please delete the assignment in digital I/O assignment menu at first.

3.3.3 Robot : GO → PMC : F



Memory for F of PMC is assigned to group output (GO) of Robot. PMC can read the 2 byte value (from -32768 to 32767) as F that the robot outputs as integer value (from 0 to 65535) of GO.

When the value is in the range from 0 to 32767, the value of GO and the value of F are the same. When the value of GO output from Robot is in the range from 32768 to 65535, the 2 byte value of F read from PMC is the value that 65536 is subtracted from the value of GO.

Example : The following is the correspondence when F0-3 is assigned to GO[1-2].

GO[1]	:	F0-1
GO[2]	:	F2-3

When the PMC internal assignment is setup as the above, the group I/O assignment menu displays as follows. The mark "PMC" is displayed on the right end column, and the rack is always 33. The slot number and the start point number are defined automatically. This line cannot be changed nor deleted in the group I/O assignment menu. To change this line, please change the setting in PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the group I/O assignment menu at the next power up.

GO #	RACK	SLOT	START PT	NUM PTS	
1	33	8	1	16	PMC
2	33	8	17	16	PMC

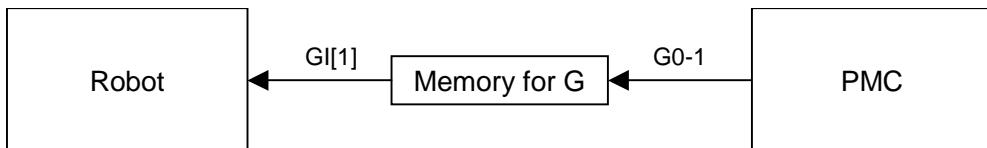
The specification of the assignment of F to GO is the following.

- The minimum unit of the assignment is 1 point for GO, 2 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- GO index can be set any number in the range that is displayed in the group I/O menu.
- When the power failure recovery is enable, the status of the F signal that is assigned to GO is recovered at power up.
- When the GO is set as simulation, the value of F is not changed even though the value of GO is changed.

NOTE

The GO signal that is already assigned to the external I/O device cannot be assigned to F of PMC. Please delete the assignment in group I/O assignment menu at first.

3.3.4 Robot : GI ← PMC : G



Memory for G of PMC is assigned to group input (GI) of Robot. Robot can read integer value (from 0 to 65535) as GI that the PMC outputs as the 2 byte value (from -32768 to 32767) to G.

When the value is in the range from 0 to 32767, the value of GI and the value of G are the same. When the 2 byte value of G is in the range from -32768 to -1, the value of GI is the value that 65536 is added from the 2 byte value of G.

Example : The following is the correspondence when G0-3 is assigned to GI[1-2].

GI[1]	:	G0-1
GI[2]	:	G2-3

When the PMC internal assignment is setup as the above, the group I/O assignment menu displays as follows. The mark "PMC" is displayed on the right end column, and the rack is always 33. The slot number and the start point number are defined automatically. This line cannot be changed nor deleted in the group I/O assignment menu. To change this line, please change the setting in PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the group I/O assignment menu at the next power up.

GI #	RACK	SLOT	START PT	NUM PTS	
1	33	9	1	16	PMC
2	33	9	17	16	PMC

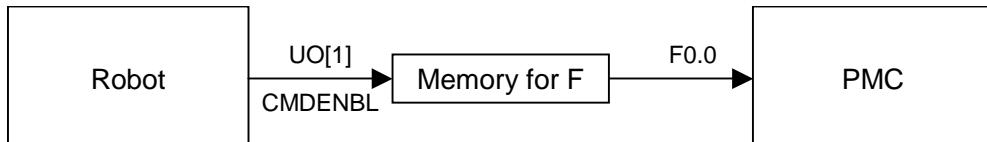
The specification of the assignment of G to GI is the following.

- The minimum unit of the assignment is 1 point for GI, 2 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- GI index can be set any number in the range that is displayed in the group I/O menu.
- Unrelated to enable or disable of power failure recovery, all G signals that GI is assigned are set to 0 at power up.
- When the GI is set as simulation, the value of GI is not changed even though the value of G is changed.

NOTE

The GI signal that is already assigned to the external I/O device cannot be assigned to G of PMC. Please delete the assignment in group I/O assignment menu at first.

3.3.5 Robot : UO → PMC : F



Memory for F of PMC is assigned to UOP output (UO) of Robot. PMC can read the UO value (ON/OFF) such as CMDENBL as bit value (1/0) of F.

Example : The following is the correspondence when F0-3 is assigned to UO[1-32].

UO[1] (CMDENBL)	:	F0.0
UO[2] (SYSRDY)	:	F0.1
UO[3] (PROGRUN)	:	F0.2
UO[4] (PAUSED)	:	F0.3
UO[5] (HELD)	:	F0.4
UO[6] (FAULT)	:	F0.5
UO[7] (ATPERCH)	:	F0.6
UO[8] (TPENBL)	:	F0.7
UO[9] (BATALM)	:	F1.0
UO[10] (BUSY)	:	F1.1
UO[11] (ACK1/SNO1)	:	F1.2

UO[12] (ACK2/SNO2)	:	F1.3
UO[13] (ACK3/SNO3)	:	F1.4
UO[14] (ACK4/SNO4)	:	F1.5
UO[15] (ACK5/SNO5)	:	F1.6
UO[16] (ACK6/SNO6)	:	F1.7
UO[17] (ACK7/SNO7)	:	F2.0
UO[18] (ACK8/SNO8)	:	F2.1
UO[19] (SNACK)	:	F2.2
UO[20] (Reserved)	:	F2.3
Not used	:	F2.4-3.7

When the PMC internal assignment is setup as the above, the UOP I/O assignment menu displays as follows. The mark "*" is displayed on the left of "UI" and "UO". In this case, the setting of rack, slot and start are ignored, and the setting is not used for the assignment of UI and UO. The UI and UO are assigned to F or G according to the setting of the PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the UOP I/O assignment menu at the next power up.

#	RANGE	RACK	SLOT	START	STAT.
1	UI[1- 18]	0	1	1	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	UO[1- 20]	0	1	1	ACTIV

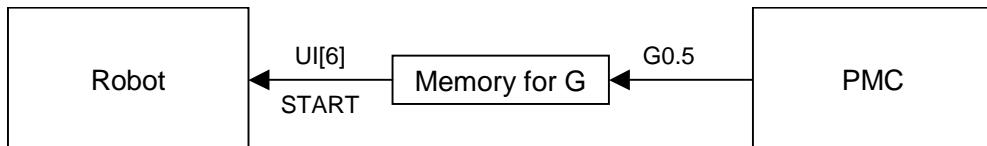
The specification of the assignment of F to UO is the following.

- The minimum unit of the assignment is 16 points for UO, 2 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- UO index can be set any number from 1 to 32766. But the meaningful signals are UO[1-20] in standard system. Please assign 4 byte of F signals to UO[1-32] in general. Please assign the proper number of signals when the number of UO signals is increased by option such as "Multi UOP interface (J964)".

NOTE

When UO is assigned to F in PMC internal I/O assignment menu, the setting of rack, slot and start in UI and UO assignment menu are ignored. The UI and UO are assigned to the memory of F or G according to the setting of the PMC internal I/O assignment menu.

3.3.6 Robot : UI ← PMC : G



Memory for G of PMC is assigned to UOP input (UI) of Robot. The bit value (1/0) that PMC set to G is read as UI value (ON/OFF) such as START by Robot. Therefore, PMC can execute robot program and so on.

Example : The following is the correspondence when G0-3 is assigned to UI[1-32].

UI[1] (IMSTP)	:	G0.0
UI[2] (HOLD)	:	G0.1
UI[3] (SFSPD)	:	G0.2
UI[4] (CSTOP)	:	G0.3
UI[5] (FAULT_RESET)	:	G0.4
UI[6] (START)	:	G0.5
UI[7] (HOME)	:	G0.6

UI[8] (ENBL)	:	G0.7
UI[9] (RSR1/PNS1)	:	G1.0
UI[10] (RSR2/PNS2)	:	G1.1
UI[11] (RSR3/PNS3)	:	G1.2
UI[12] (RSR4/PNS4)	:	G1.3
UI[13] (RSR5/PNS5)	:	G1.4
UI[14] (RSR6/PNS6)	:	G1.5
UI[15] (RSR7/PNS7)	:	G1.6
UI[16] (RSR8/PNS8)	:	G1.7
UI[17] (PNSTROBE)	:	G2.0
UI[18] (PROD_START)	:	G2.1
Not used	:	G2.2-3.7

When the PMC internal assignment is setup as the above, the UOP I/O assignment menu displays as follows. The mark "*" is displayed on the left of "UI" and "UO". In this case, the setting of rack, slot and start are ignored, and the setting is not used for the assignment of UI and UO. The UI and UO are assigned to F or G according to the setting of the PMC internal I/O assignment menu. The setting of the PMC internal I/O assignment is reflected to the UOP I/O assignment menu at the next power up.

#	RANGE	RACK	SLOT	START	STAT.
1	UI[1- 18]	0	1	1	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	UO[1- 20]	0	1	1	ACTIV

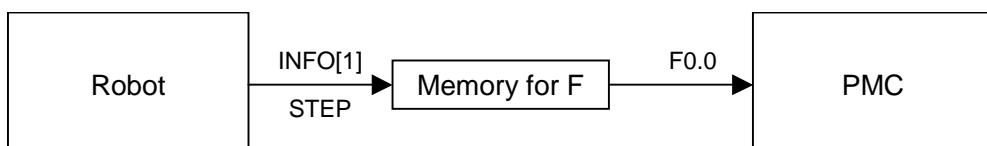
The specification of the assignment of G to UI is the following.

- The minimum unit of the assignment is 16 points for UI, 2 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- UI index can be set any number from 1 to 32766. But the meaningful signals are UI[1-18] in standard system. Please assign 4 byte of G signals to UI[1-32] in general. Please assign the proper number of signals when the number of UI signals is increased by option such as "Multi UOP interface (J964)".

NOTE

When UO is assigned to F in PMC internal I/O assignment menu, the setting of rack, slot and start in UI and UO assignment menu are ignored. The UI and UO are assigned to the memory of F or G according to the setting of the PMC internal I/O assignment menu.

3.3.7 Robot : INFO → PMC : F



The robot system information is set in the memory for F of PMC. The system information such as single step mode status can be read from the bit value (1/0) of F from PMC. The meaning of each bit in INFO is the following.

Signal	Name	Function
INFO[1]	STEP	It becomes 1 in single step mode
INFO[2]	SYSRST	When reset occurs, this becomes 1 during 100 msec. Note : This also becomes 1 by UI[5] (FAULT_RESET). Therefore, if the PMC program connects F signal assigned to INFO[2] to G signal assigned to UO[5], a reset occurs forever.
INFO[3]	TPESTP	When the teach pendant E-STOP button is pressed, this becomes 1. This shows the teach pendant E-STOP button status only. Timing is different from FAULT (UO[6]).
INFO[4]	DEADMN	When the DEADMAN switch is pressed, this becomes 1. This shows the DEADMAN switch status only. Timing is different from FAULT (UO[6]).
INFO[5]		Not used.
INFO[6]	UOPDSB	When "Enable UI signals" in system configuration menu is FALSE, this becomes 1.
INFO[7-16]		Not used.

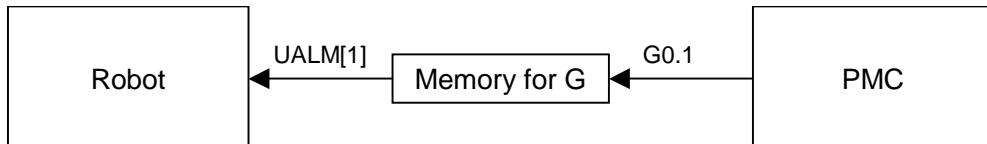
Example : The following is the correspondence when F0-1 is assigned to INFO[1-16].

INFO[1] (STEP)	:	F0.0
INFO[2] (SYSRST)	:	F0.1
INFO[3] (TPESTP)	:	F0.2
INFO[4] (DEADMN)	:	F0.3
INFO[5]	:	F0.4
INFO[6] (UOPDSB)	:	F0.5
INFO[7-16]	:	F0.6-1.7

The specification of the assignment of F to INFO is the following.

- The assignment size is fixed as 16 points for INFO, 2 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).

3.3.8 Robot : UALM \leftarrow PMC : G



PMC can cause user alarm by output bit value (1/0) to G. For example, when the bit of G that is assigned to UALM[1] is set to 1, the user alarm 1 "INTP-213 *** (-PMC-,1) UALM[1]" occurs. This alarm cannot be reset until the bit value of G is set to 0. The message of the user alarm can be setup in user alarm menu.

Example : The following is the correspondence when G0-3 is assigned to UALM[1-32].

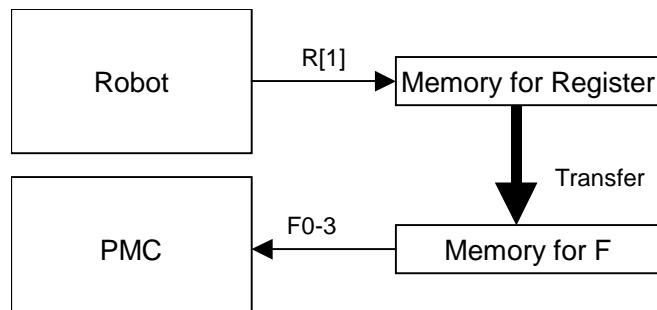
UALM[1] (User alarm 1 occurs)	:	G0.0
UALM[2] (User alarm 2 occurs)	:	G0.1
UALM[3] (User alarm 3 occurs)	:	G0.2
UALM[4] (User alarm 4 occurs)	:	G0.3
UALM[5] (User alarm 5 occurs)	:	G0.4
UALM[6] (User alarm 6 occurs)	:	G0.5
UALM[7] (User alarm 7 occurs)	:	G0.6
UALM[8] (User alarm 8 occurs)	:	G0.7
UALM[9] (User alarm 9 occurs)	:	G1.0
UALM[10] (User alarm 10 occurs)	:	G1.1
UALM[11] (User alarm 11 occurs)	:	G1.2
UALM[12] (User alarm 12 occurs)	:	G1.3
UALM[13] (User alarm 13 occurs)	:	G1.4
UALM[14] (User alarm 14 occurs)	:	G1.5

UALM[15] (User alarm 15 occurs)	:	G1.6
UALM[16] (User alarm 16 occurs)	:	G1.7
UALM[17] (User alarm 17 occurs)	:	G2.0
UALM[18] (User alarm 18 occurs)	:	G2.1
UALM[19] (User alarm 19 occurs)	:	G2.2
UALM[20] (User alarm 20 occurs)	:	G2.3
UALM[21] (User alarm 21 occurs)	:	G2.4
UALM[22] (User alarm 22 occurs)	:	G2.5
UALM[23] (User alarm 23 occurs)	:	G2.6
UALM[24] (User alarm 24 occurs)	:	G2.7
UALM[25] (User alarm 25 occurs)	:	G3.0
UALM[26] (User alarm 26 occurs)	:	G3.1
UALM[27] (User alarm 27 occurs)	:	G3.2
UALM[28] (User alarm 28 occurs)	:	G3.3
UALM[29] (User alarm 29 occurs)	:	G3.4
UALM[30] (User alarm 30 occurs)	:	G3.5
UALM[31] (User alarm 31 occurs)	:	G3.6
UALM[32] (User alarm 32 occurs)	:	G3.7

The specification of the assignment of G to UALM is the following.

- The minimum unit of the assignment is 32 points for UALM, 4 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- The top index of UALM must be the number that the remainder of division by 32 is 1 (Ex. 1, 33, 65).
- The maximum of total number of signals that are assigned to UALM is 128. For example, when UALM[1-64] and UALM[129-192] are assigned, the total number of signals is 128. Therefore, no more signals can be assigned to UALM.

3.3.9 Robot : Register → PMC : F



Data is transferred periodically from Register of Robot to the memory for F of PMC. PMC can read the value of Register as the 4 byte value (from -2147483648 to 2147483647) of F.

The value that the Register value is rounded down to the nearest integer is transferred to the memory for F. When the value of the Register assigned to F is out of the range from -2147483648 to 2147483647, the alarm occurs. The alarm cannot be reset until the value of the Register is changed to the value in the range.

Example : The following is the correspondence when F0-7 is assigned to R[1-2].

R[1]	:	F0-3
R[2]	:	F4-7

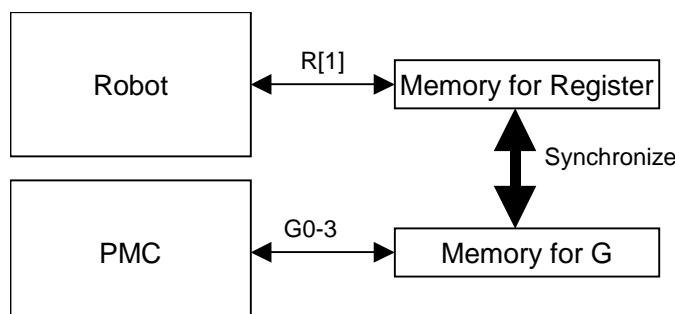
NOTE

The data transfer is performed periodically. Therefore, the data transfer is delayed for the processing period. The processing period becomes long when the number of assigned signals is increased. The current processing period is displayed on the PMC internal I/O assignment menu.

The specification of the assignment of F to GI is the following.

- The minimum unit of the assignment is 1 Register, 4 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- Register index can be set any number in the range that is displayed in the Register menu.

3.3.10 Robot : Register ← PMC : G



Data is synchronized periodically between the memory for G of PMC and Register of Robot.

When Robot changes the value of Register, the data is transferred from the Register to the memory for G of PMC. In this case, the processing is the same as the case "Robot : Register → PMC : F".

When PMC changes the value of G, the data is transferred from the memory for G of PMC to the Register. The 4 byte value (from -2147483648 to 2147483647) set to G by PMC is set to the Register.

When the 4 byte value of G is 2147483647, "*****" is displayed in Register menu.

The synchronization processing is performed periodically. The identification whether Robot changes Register or PMC changes G is made according to which is changed from the previous synchronization processing. If both are changed, the data is transferred from the memory for G of PMC to the Register.

Example : The following is the correspondence when G0-7 is assigned to R[1-2].

R[1]	:	G0-3
R[2]	:	G4-7

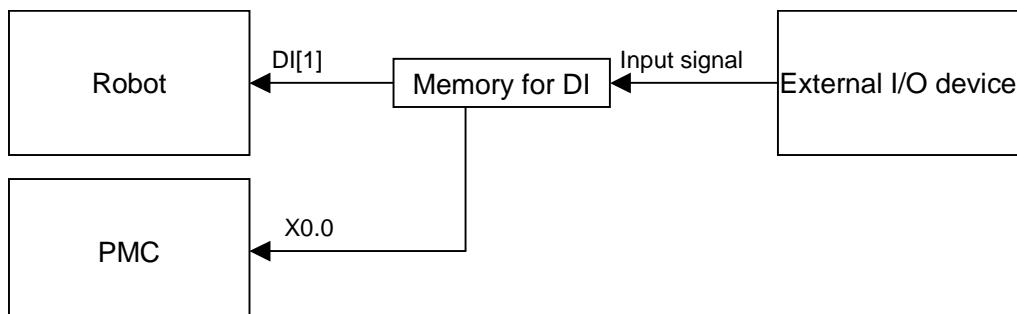
NOTE

The data synchronization is performed periodically. Therefore, the data synchronization is delayed for the processing period. The processing period becomes long when the number of assigned signals is increased. The current processing period is displayed on the PMC internal I/O assignment menu.

The specification of the assignment of G to Register is the following.

- The minimum unit of the assignment is 1 Register, 4 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- Register index can be set any number in the range that is displayed in the Register menu.

3.3.11 Robot : DI,RI,WI,WSI,SI,*UI → PMC : X



The X of PMC is assigned to DI,RI,WI,WSI,SI,*UI(Note) of Robot that is assigned to the external I/O device. This assignment is intended to maintain compatibility with conventional models.

Note : The "*UI" means that the input signals of the external I/O device that is specified in UI assignment menu as rack, slot and start when UI or UO is assigned to G or F in PMC internal I/O assignment menu. When "*UI" is assigned to X, the sequence program can read the status of the input signal of the external I/O device by reading the corresponded X address from the sequence program.

Example : The following is the correspondence when X0.0-0.7 is assigned to DI[1-8].

DI[1]	:	X0.0
DI[2]	:	X0.1
DI[3]	:	X0.2
DI[4]	:	X0.3
DI[5]	:	X0.4
DI[6]	:	X0.5
DI[7]	:	X0.6
DI[8]	:	X0.7

The specification of the assignment of X to DI, RI, WI, WSI, SI or *UI is the following.

- The minimum unit of the assignment is 8 point for DI and so on, 1 byte for X.
- The available address range of X is from 0 to 127, from 200 to 327, from 400 to 527, from 600 to 727, from 1000 to 1127 of the 1st PMC. The address of the PMC that is not 1st PMC is not available. The top address of X can be any address in the range.
- Index of input signal of Robot can be set any number in the range that is displayed in the I/O menu.
- When the input signal of Robot is set as simulation, the simulated status is read by X.
- If the input signal of Robot that is not assigned to the external I/O device is assigned to X of PMC, the value of the X is always 0.

NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, the PMC external I/O assignment is not available. Please delete all setting in PMC external I/O assignment menu.

NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, if X or Y address is specified in the parameter of functional instruction in the sequence program, the alarm "PRIO-136 BYTE access to x" occurs, and the sequence program is not executed. In this case, please copy the X or Y to internal relay by basic instruction, and specify the internal relay address in the parameter of the functional instruction.

NOTE

The override function is not available for the X or Y area that is assigned in the PMC internal I/O assignment menu. In these area, the simulated I/O status is reflected when the corresponded I/O is simulated. Please use I/O simulation instead of the override function.

NOTE

The synchronization processing of the 2nd level sequence part does not work for the X area that is assigned in the PMC internal I/O assignment menu. The signals in this area can be changed during 1 scan of the 2nd level sequence program. When the signal is read from two or more part of the 2nd level sequence part, please copy the signal to an internal relay at the beginning of the 2nd level sequence part so that the subsequent operation of the 2nd level sequence program part references the internal relay.

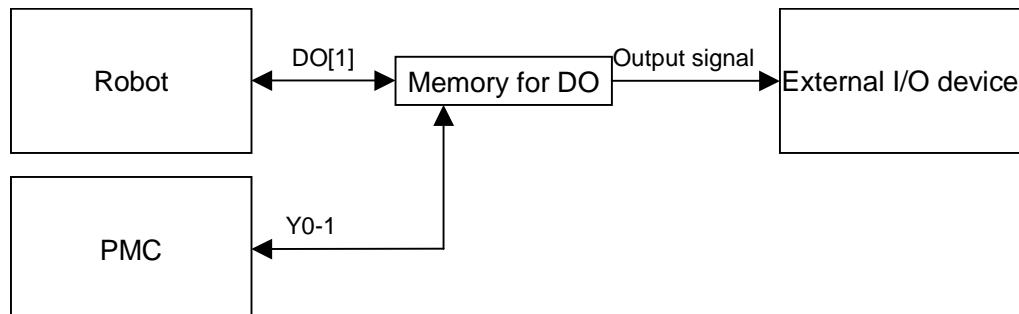
NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, Memory Common Mode is not available. Please do not set PMC type to 'COM' in PMC setting menu.

NOTE

This assignment is intended to maintain compatibility with conventional models. For the purpose of input from the external device, please assign the external I/O device to PMC address by PMC external I/O assignment menu in general.

3.3.12 Robot : DO,RO,WO,WSO,SO,*UO ← PMC : Y



The Y of PMC is assigned to DO,RO,WO,WSO,SO,*UO(Note) of Robot that is assigned to the external I/O device. This assignment is intended to maintain compatibility with conventional models.

Note : The "*UO" means that the output signals of the external I/O device that is specified in UO assignment menu as rack, slot and start when UI or UO is assigned to G or F in PMC internal I/O assignment menu. When "*UO" is assigned to Y, the sequence program can set the status of the output signal of the external I/O device by writing the corresponded Y address from the sequence program.

Example : The following is the correspondence when Y0.0-0.7 is assigned to DO[1-8].

DO[1]	:	Y0.0
DO[2]	:	Y0.1
DO[3]	:	Y0.2
DO[4]	:	Y0.3
DO[5]	:	Y0.4
DO[6]	:	Y0.5

DO[7]	:	Y0.6
DO[8]	:	Y0.7

The specification of the assignment of Y to DO, RO, WO, WSO, SO, *UO is the following.

- The minimum unit of the assignment is 8 point for DO and so on, 1 byte for Y.
- The available address range of Y is from 0 to 127, from 200 to 327, from 400 to 527, from 600 to 727, from 1000 to 1127 in the 1st PMC. The address of the PMC that is not 1st PMC is not available. The top address of Y can be any address in the range.
- Index of output signal of Robot can be set any number in the range that is displayed in the I/O menu.
- When the power failure recovery is enable, the status of the output signal is recovered at power up.
- When the output signal of Robot is set as simulation, the value is Y is reflected to simulated I/O, and the value is not output to the external device.
- If the input signal of Robot that is not assigned to the external I/O device is assigned to Y of PMC, error does not occur, but the value is not output to external I/O device.
- Even though the I/O of Robot is set to complementary, the output to the corresponded PMC signal does not work as complementary.

NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, the PMC external I/O assignment is not available. Please delete all setting in PMC external I/O assignment menu.

NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, if X or Y address is specified in the parameter of functional instruction in the sequence program, the alarm "PRIO-136 BYTE access to x" occurs, and the sequence program is not executed. In this case, please copy the X or Y to internal relay by basic instruction, and specify the internal relay address in the parameter of the functional instruction.

NOTE

The override function is not available for the X or Y area that is assigned in the PMC internal I/O assignment menu. In these area, the simulated I/O status is reflected when the corresponded I/O is simulated. Please use I/O simulation instead of the override function.

NOTE

When there is the assignment of X or Y in PMC internal I/O assignment menu, Memory Common Mode is not available. Please do not set PMC type to 'COM' in PMC setting menu.

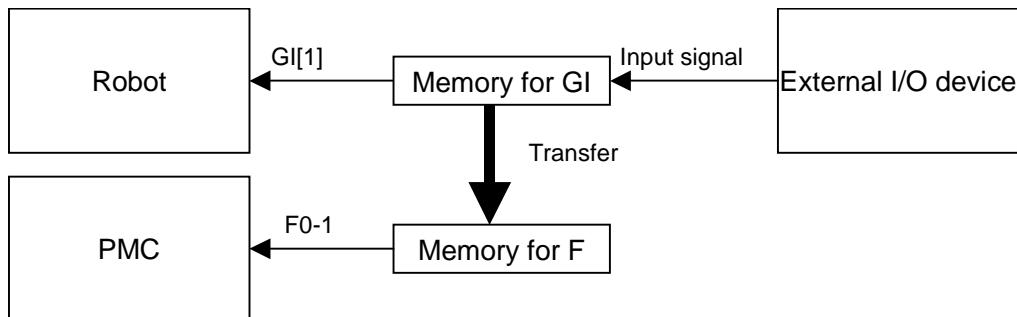
NOTE

This assignment is intended to maintain compatibility with conventional models. For the purpose of input from the external device, please assign the external I/O device to PMC address by PMC external I/O assignment menu in general.

NOTE

When both PMC sequence program and robot program output to the same signal. The value of robot program is output in a moment, but it is overwritten by the value of PMC immediately. Robot program should not output to the signals that is output from PMC sequence program.

3.3.13 Robot : GI,AI → PMC : F



Data is transferred periodically from GI or AI of Robot that is assigned to the external I/O device to the memory for F of PMC.

This assignment is different from the assignment between GI and G, GO and F that are described in the above. The GI or AI needs to be assigned to the external I/O device in I/O assignment (Robot).

PMC can read the value of GI or AI (from 0 to 65535) that is assigned to the external I/O device as the 2 byte value (from -32768 to 32767) of F.

When the value is in the range from 0 to 32767, the value of GI or AI and the value of F are the same. When the value of GI or AI is in the range from 32768 to 65535, the 2 byte value of F read from PMC is the value that 65536 is subtracted from the value of GI or AI.

Example : The following is the correspondence when F0-3 is assigned to GI[1-2].

GI[1]	:	F0-1
GI[2]	:	F2-3

NOTE

The data transfer is performed periodically. Therefore, the data transfer is delayed for the processing period. The processing period becomes long when the number of assigned signals is increased. The current processing period is displayed on the PMC internal I/O assignment menu.

NOTE

This assignment is intended to maintain compatibility with conventional models. For the purpose of input from the external device, please assign the external I/O device to PMC address by PMC external I/O assignment menu in general. The PMC external I/O assignment is better for the response time.

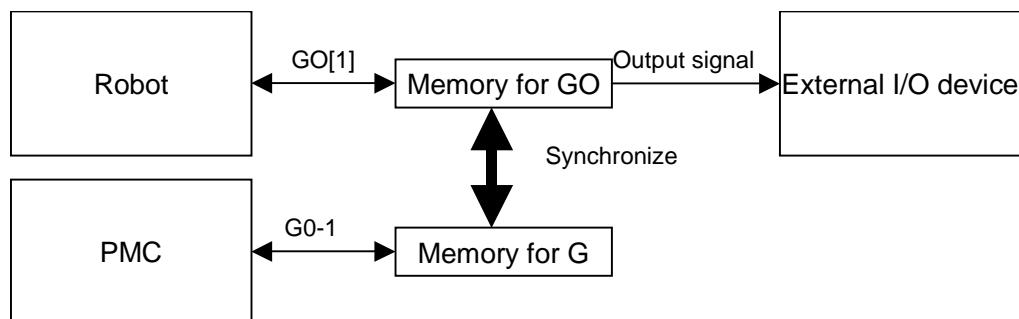
The specification of the assignment of F to GI or AI is the following.

- The minimum unit of the assignment is 1 point for GI or AI, 2 byte for F.
- The top address of F must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- GI or AI index can be set any number in the range that is displayed in the group I/O menu or analog I/O menu.

- When the GI or AI is set as simulation, the simulation value of GI or AI is transferred to the F signal address.

NOTE

- 1 If the GI or AI signal that is not assigned to the external I/O device is assigned to F of PMC, the value of the F is always 0.
- 2 When the GI or AI is offline and it is not set as simulation, the 2 byte value of F that is assigned to the GI or AI is always 0.

3.3.14 Robot : GO,AO ← PMC : G

Data is synchronized periodically between the memory for G of PMC and GO or AO of Robot that is assigned to the external I/O device.

This assignment is different from the assignment between GI and G, GO and F that are described in the above. The GO or AO needs to be assigned to the external I/O device in I/O assignment (Robot).

When Robot changes the value of GO or AO, the data is transferred from the GO or AO that is assigned to the external I/O device to the memory for G of PMC. In this case, the processing is the same as the case "Robot : GI,AI → PMC : F".

When PMC changes the value of G, the data is transferred from the memory for G of PMC to the GO or AO that is assigned to the external I/O device. The 2 byte value (from -32768 to 32767) output to G by PMC is output to external I/O device as GO or AO value (from 0 to 65535).

When the value is in the range from 0 to 32767, the value of GO or AO and the value of G are the same. When the 2 byte value of G is in the range from -32768 to -1, the value of GO or AO is the value that 65536 is added from the 2 byte value of G.

The synchronization processing is performed periodically. The identification whether Robot changes GO, AO or PMC changes G is made according to which is changed from the previous synchronization processing. If both are changed, the data is transferred from the memory for G of PMC to the GO or AO assigned to the external I/O device.

Example : The following is the correspondence when G0-3 is assigned to GO[1-2].

GO[1]	:	G0-1
GO[2]	:	G2-3

NOTE

The data synchronization is performed periodically. Therefore, the data synchronization is delayed for the processing period. The processing period becomes long when the number of assigned signals is increased. The current processing period is displayed on the PMC internal I/O assignment menu.

NOTE

This assignment is intended to maintain compatibility with conventional models. For the purpose of output to the external device, please assign the external I/O device to PMC address by PMC external I/O assignment menu in general. The PMC external I/O assignment is better for the response time.

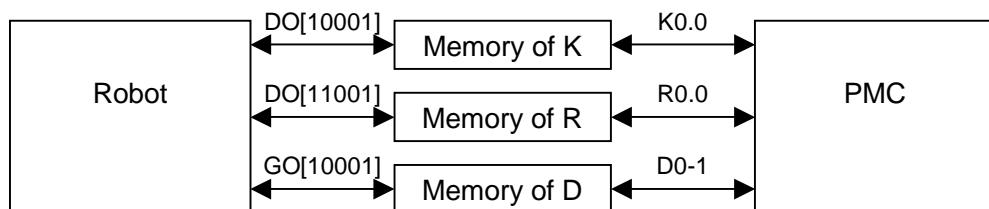
The specification of the assignment of G to GO or AO is the following.

- The minimum unit of the assignment is 1 point for GO or AO, 2 byte for G.
- The top address of G must be multiple of 4. The available address range is from 0 to 767 (from 1000 to 1767 is also available for 1st path).
- GO or AO index can be set any number in the range that is displayed in the group I/O menu or analog I/O menu.
- When the power failure recovery is enable, the status of the G signal that is assigned to GO or AO is recovered at power up.
- When the GO or AO is set as simulation, the value of G is transferred to the simulation value of GO or AO, and the value does not output to the external I/O device.

NOTE

- 1 If the GO or AO signal that is not assigned to the external I/O device is assigned to G of PMC, The synchronization processing is not performed.
- 2 When the GO is offline and it is not set as simulation, the synchronization processing is not performed. In this case, the value of G and the value of GO or AO is sometimes different.

- ### 3.3.15 Robot : DO[10001-10136] → PMC1: K0-17
- ### Robot : DO[10137-10160] → PMC1: K900-902
- ### Robot : DO[11001-23000] → PMC1: R0-1499
- ### Robot : GO[10001-11500] → PMC1: D0-2999



Memory for K, R, D of PMC is assigned to DO, GO of Robot. The DO[10001-] and GO[10001-] are not displayed in I/O menu, but they can be read or written by TP program.

This assignment is for compatibility with conventional models. The assignment for K, R, D is available only for the following assignment patterns.

DO[10001-10136]	17	1:K00000
DO[10137-10160]	3	1:K00900
DO[11001-23000]	1500	1:R00000
GO[10001-11500]	3000	1:D00000

The relation between K, R, D address and DO, GO index is the following.

$$Ka.b \leftrightarrow DO[c]$$

$$K0-17 : a \times 8 + b + 10001 = c$$

$$K900-902 : (a - 900) \times 8 + b + 10137 = c$$

$$\begin{aligned} Ra.b &\leftrightarrow DO[c] \\ a \times 8 + b + 11001 &= c \end{aligned}$$

Da (Upper BYTE), Db (Lower BYTE) \leftrightarrow GO[c]

$$\begin{aligned} a &= ((c - 10001) \times 2) + 1 \\ b &= ((c - 10001) \times 2) \end{aligned}$$

In conventional model, DO[10137-10160] was assigned to K17-19, but it is assigned to K900-902 in this assignment. This assignment is to access the corresponded PMC setting parameter by DO[10137-10160], because the PMC setting parameter was K17-19 in conventional model, and it is K900-999 in this model.

NOTE

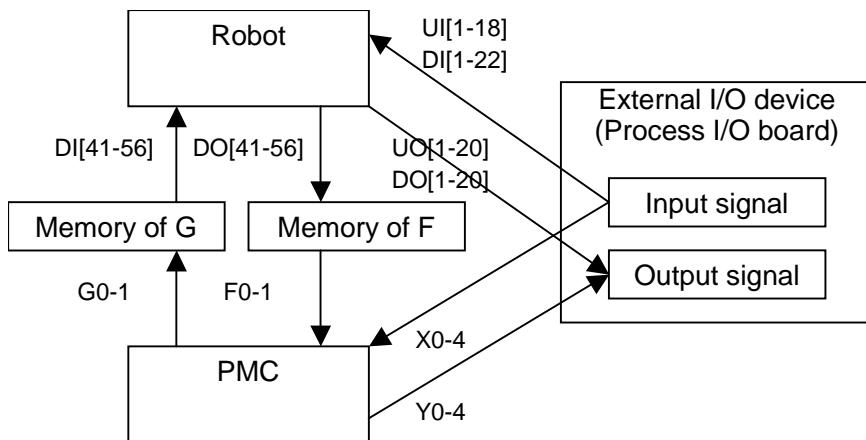
When both PMC sequence program and robot program output to the same signal. The value of robot program is output in a moment, but it is overwritten by the value of PMC immediately. Robot program should not output to the signals that is output from PMC sequence program.

NOTE

This assignment is intended to maintain compatibility with conventional models. For the purpose of the interface between robot and PMC, please use F or G address in general.

3.4 EXAMPLE OF PMC I/O ASSIGNMENT

3.4.1 Example 1 : UOP is Controlled by External I/O Device



This section explains the setting of PMC I/O assignment and the correspondence of the PMC address in the case of the following configuration.

- The process I/O board that has 40 input ports and 40 output ports is connected. (It is identified as rack 0, slot 1.)
- The default I/O assignment (Robot) is used.
- UOP is assigned to the signals of the process I/O board. (UOP is not controlled by PMC sequence program.)
- Both Robot and PMC can access all signals of the process I/O board.
- G0-1 is assigned to DI[41-56] for the interface from PMC to Robot.
- F0-1 is assigned to DO[41-56] for the interface from Robot to PMC

To assign the signals of the process I/O board to X0-4 and Y0-4, setup the PMC external I/O assignment as follows.

Type	Rack	Slot	Size	Addr
1 DI	0	1	5	1:X00000
2 DO	0	1	5	1:Y00000

To assign G0-1 to DI[41-56], and F0-1 to DO[41-56], setup the PMC internal I/O assignment as follows.

Robot data	Size	Address
1 DO[41- 56]	16	1:F00000
2 DI[41- 56]	16	1:G00000

When the controller power is off and on with the above setting, The I/O assignment (Robot) will be the following.

(Because DI[41-56] and DO[41-56] are configured by PMC internal I/O assignment menu, "PMC" is displayed on the right end, and the setting of the line cannot be changed in the I/O assignment menu. The value of rack, slot and start for the line are assigned automatically.)

#	RANGE	RACK	SLOT	START	STAT.
1	UI[1- 8]	0	1	1	ACTIV
2	UI[9- 16]	0	1	9	ACTIV
3	UI[17- 18]	0	1	17	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	UO[1- 8]	0	1	1	ACTIV
2	UO[9- 16]	0	1	9	ACTIV
3	UO[17- 20]	0	1	17	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	DI[1- 22]	0	1	19	ACTIV
2	DI[23- 40]	0	0	0	UNASG
3	DI[41- 72]	33	8	1	PMC

#	RANGE	RACK	SLOT	START	STAT.
1	DO[1- 20]	0	1	21	ACTIV
2	DO[21- 40]	0	0	0	UNASG
3	DO[41- 72]	33	9	1	PMC

In this configuration, the correspondence of the signals of the external I/O device, I/O of Robot and PMC address is the following.

(The value of rack, slot and start for DI[41-56] and DO[41-56] are not specified, because these are assigned automatically.)

3.PMC I/O ASSIGNMENT

B-83254EN/02

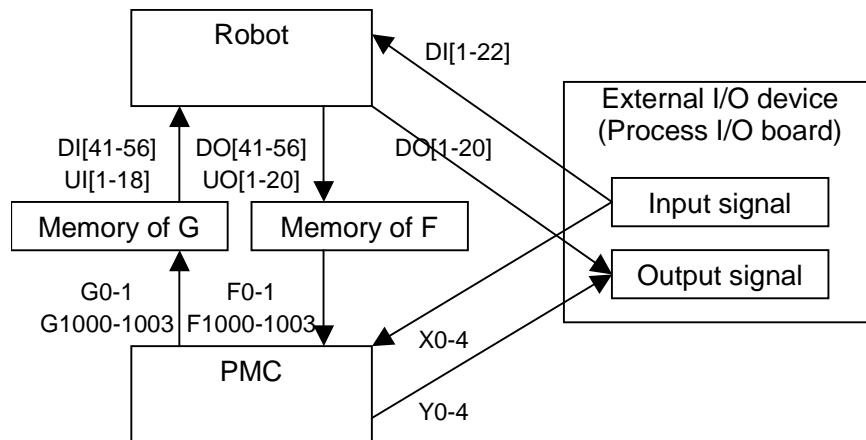
Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Input signal process I/O board	0	1	In 1	UI[1: IMSTP]	PMC1: X0.0
	0	1	In 2	UI[2: HOLD]	PMC1: X0.1
	0	1	In 3	UI[3: SFSPD]	PMC1: X0.2
	0	1	In 4	UI[4: CSTOP1]	PMC1: X0.3
	0	1	In 5	UI[5: RESET]	PMC1: X0.4
	0	1	In 6	UI[6: START]	PMC1: X0.5
	0	1	In 7	UI[7: HOME]	PMC1: X0.6
	0	1	In 8	UI[8: ENBL]	PMC1: X0.7
	0	1	In 9	UI[9: RSR1/PNS1]	PMC1: X1.0
	0	1	In 10	UI[10: RSR2/PNS2]	PMC1: X1.1
	0	1	In 11	UI[11: RSR3/PNS3]	PMC1: X1.2
	0	1	In 12	UI[12: RSR4/PNS4]	PMC1: X1.3
	0	1	In 13	UI[13: RSR5/PNS5]	PMC1: X1.4
	0	1	In 14	UI[14: RSR6/PNS6]	PMC1: X1.5
	0	1	In 15	UI[15: RSR7/PNS7]	PMC1: X1.6
	0	1	In 16	UI[16: RSR8/PNS8]	PMC1: X1.7
	0	1	In 17	UI[17: PNSTROBE]	PMC1: X2.0
	0	1	In 18	UI[18: PROD_START]	PMC1: X2.1
	0	1	In 19	DI[1]	PMC1: X2.2
	0	1	In 20	DI[2]	PMC1: X2.3
	0	1	In 21	DI[3]	PMC1: X2.4
	0	1	In 22	DI[4]	PMC1: X2.5
	0	1	In 23	DI[5]	PMC1: X2.6
	0	1	In 24	DI[6]	PMC1: X2.7
	0	1	In 25	DI[7]	PMC1: X3.0
	0	1	In 26	DI[8]	PMC1: X3.1
	0	1	In 27	DI[9]	PMC1: X3.2
	0	1	In 28	DI[10]	PMC1: X3.3
	0	1	In 29	DI[11]	PMC1: X3.4
	0	1	In 30	DI[12]	PMC1: X3.5
	0	1	In 31	DI[13]	PMC1: X3.6
	0	1	In 32	DI[14]	PMC1: X3.7
	0	1	In 33	DI[15]	PMC1: X4.0
	0	1	In 34	DI[16]	PMC1: X4.1
	0	1	In 35	DI[17]	PMC1: X4.2
	0	1	In 36	DI[18]	PMC1: X4.3
	0	1	In 37	DI[19]	PMC1: X4.4
	0	1	In 38	DI[20]	PMC1: X4.5
	0	1	In 39	DI[21]	PMC1: X4.6
	0	1	In 40	DI[22]	PMC1: X4.7
Output signal process I/O board	0	1	Out 1	UO[1: CMDENBL]	PMC1: Y0.0
	0	1	Out 2	UO[2: SYSRDY]	PMC1: Y0.1
	0	1	Out 3	UO[3: PROGRUN]	PMC1: Y0.2
	0	1	Out 4	UO[4: PAUSED]	PMC1: Y0.3
	0	1	Out 5	UO[5: HELD]	PMC1: Y0.4
	0	1	Out 6	UO[6: FAULT]	PMC1: Y0.5
	0	1	Out 7	UO[7: ATPERCH]	PMC1: Y0.6
	0	1	Out 8	UO[8: TPENBL]	PMC1: Y0.7
	0	1	Out 9	UO[9: BATALM]	PMC1: Y1.0
	0	1	Out 10	UO[10: BUSY]	PMC1: Y1.1
	0	1	Out 11	UO[11: ACK1/SNO1]	PMC1: Y1.2
	0	1	Out 12	UO[12: ACK2/SNO2]	PMC1: Y1.3
	0	1	Out 13	UO[13: ACK3/SNO3]	PMC1: Y1.4

3.PMC I/O ASSIGNMENT

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Output signal process I/O board	0	1	Out 14	UO[14: ACK4/SNO4]	PMC1: Y1.5
	0	1	Out 15	UO[15: ACK5/SNO5]	PMC1: Y1.6
	0	1	Out 16	UO[16: ACK6/SNO6]	PMC1: Y1.7
	0	1	Out 17	UO[17: ACK7/SNO7]	PMC1: Y2.0
	0	1	Out 18	UO[18: ACK8/SNO8]	PMC1: Y2.1
	0	1	Out 19	UO[19: SNACK]	PMC1: Y2.2
	0	1	Out 20	UO[20: Reserve]	PMC1: Y2.3
	0	1	Out 21	DO[1]	PMC1: Y2.4
	0	1	Out 22	DO[2]	PMC1: Y2.5
	0	1	Out 23	DO[3]	PMC1: Y2.6
	0	1	Out 24	DO[4]	PMC1: Y2.7
	0	1	Out 25	DO[5]	PMC1: Y3.0
	0	1	Out 26	DO[6]	PMC1: Y3.1
	0	1	Out 27	DO[7]	PMC1: Y3.2
	0	1	Out 28	DO[8]	PMC1: Y3.3
	0	1	Out 29	DO[9]	PMC1: Y3.4
	0	1	Out 30	DO[10]	PMC1: Y3.5
	0	1	Out 31	DO[11]	PMC1: Y3.6
	0	1	Out 32	DO[12]	PMC1: Y3.7
	0	1	Out 33	DO[13]	PMC1: Y4.0
	0	1	Out 34	DO[14]	PMC1: Y4.1
	0	1	Out 35	DO[15]	PMC1: Y4.2
	0	1	Out 36	DO[16]	PMC1: Y4.3
	0	1	Out 37	DO[17]	PMC1: Y4.4
	0	1	Out 38	DO[18]	PMC1: Y4.5
	0	1	Out 39	DO[19]	PMC1: Y4.6
	0	1	Out 40	DO[20]	PMC1: Y4.7
Memory of G (G0-1)				DI[41]	PMC1: G0.0
				DI[42]	PMC1: G0.1
				DI[43]	PMC1: G0.2
				DI[44]	PMC1: G0.3
				DI[45]	PMC1: G0.4
				DI[46]	PMC1: G0.5
				DI[47]	PMC1: G0.6
				DI[48]	PMC1: G0.7
				DI[49]	PMC1: G1.0
				DI[50]	PMC1: G1.1
				DI[51]	PMC1: G1.2
				DI[52]	PMC1: G1.3
				DI[53]	PMC1: G1.4
				DI[54]	PMC1: G1.5
				DI[55]	PMC1: G1.6
				DI[56]	PMC1: G1.7
Memory of F (F0-1)				DO[41]	PMC1: F0.0
				DO[42]	PMC1: F0.1
				DO[43]	PMC1: F0.2
				DO[44]	PMC1: F0.3
				DO[45]	PMC1: F0.4
				DO[46]	PMC1: F0.5
				DO[47]	PMC1: F0.6
				DO[48]	PMC1: F0.7
				DO[49]	PMC1: F1.0
				DO[50]	PMC1: F1.1

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Memory of F (F0-1)				DO[51]	PMC1: F1.2
				DO[52]	PMC1: F1.3
				DO[53]	PMC1: F1.4
				DO[54]	PMC1: F1.5
				DO[55]	PMC1: F1.6
				DO[56]	PMC1: F1.7

3.4.2 Example 2 : UOP is controlled by PMC



This section explains the setting of PMC I/O assignment and the correspondence of the PMC address in the case of the following configuration.

- The process I/O board that has 40 input ports and 40 output ports is connected.
(It is identified as rack 0, slot 1.)
- The default I/O assignment (Robot) is used.
- UOP is assigned to F and G of PMC, UOP is controlled by PMC sequence program.
- Both Robot and PMC can access all signals of the process I/O board.
- G0-1 is assigned to DI[41-56] for the interface from PMC to Robot.
- F0-1 is assigned to DO[41-56] for the interface from Robot to PMC

To assign the signals of the process I/O board to X0-4 and Y0-4, setup the PMC external I/O assignment as follows.

Type	Rack	Slot	Size	Addr
1 DI	0	1	5	1:X00000
2 DO	0	1	5	1:Y00000

To assign G0-1 to DI[41-56], F0-1 to DO[41-56], G1000-1003 to UI[1-32], and F1000-1003 to UO[1-32], setup the PMC internal I/O assignment as follows. (The lines 3-4 are added to the setting of example 1.)

Robot data	Size	Address
1 DO[41- 56]	16	1:F00000
2 DI[41- 56]	16	1:G00000
3 UO[1- 32]	32	1:F10000
4 UI[1- 32]	32	1:G00000

When the controller power is off and on with the above setting, The I/O assignment (Robot) will be the following.

(Because DI[41-56] and DO[41-56] are configured by PMC internal I/O assignment menu, "PMC" is displayed on the right end, and the setting of the line cannot be changed in the I/O assignment menu. The value of rack, slot and start for the line are assigned automatically.)

#	RANGE	RACK	SLOT	START	STAT.
1	*UI[1- 8]	0	1	1	ACTIV
2	*UI[9- 16]	0	1	9	ACTIV
3	*UI[17- 18]	0	1	17	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	*UO[1- 8]	0	1	1	ACTIV
2	*UO[9- 16]	0	1	9	ACTIV
3	*UO[17- 20]	0	1	17	ACTIV

#	RANGE	RACK	SLOT	START	STAT.
1	DI[1- 22]	0	1	19	ACTIV
2	DI[23- 40]	0	0	0	UNASG
3	DI[41- 72]	33	8	1	PMC

#	RANGE	RACK	SLOT	START	STAT.
1	DO[1- 20]	0	1	21	ACTIV
2	DO[21- 40]	0	0	0	UNASG
3	DO[41- 72]	33	9	1	PMC

The "*" is displayed on the left of UI and UO in the above I/O assignment (Robot) comparison with the example 1.

The value of rack, slot and start for UI and UO are ignored, and the UI/UO are assigned to the memory of G/F according to the setting of the PMC internal I/O assignment menu.

In this configuration, the correspondence of the signals of the external I/O device, I/O of Robot and PMC address is the following.

(The value of rack, slot and start for DI[41-56] and DO[41-56] are not specified, because these are assigned automatically.)

Because the setting of rack, slot and start for UI and UO in I/O assignment (Robot) are ignored, there is not I/O of Robot that is corresponded to the signals, In 1-18 and Out 1-20, of the process I/O board.

The UI and UO are assigned to the memory of G(G1000-1003) and the memory of F(F1000-1003).

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Input signal of process I/O board	0	1	In 1		PMC1: X0.0
	0	1	In 2		PMC1: X0.1
	0	1	In 3		PMC1: X0.2
	0	1	In 4		PMC1: X0.3
	0	1	In 5		PMC1: X0.4
	0	1	In 6		PMC1: X0.5
	0	1	In 7		PMC1: X0.6
	0	1	In 8		PMC1: X0.7
	0	1	In 9		PMC1: X1.0
	0	1	In 10		PMC1: X1.1
	0	1	In 11		PMC1: X1.2
	0	1	In 12		PMC1: X1.3
	0	1	In 13		PMC1: X1.4
	0	1	In 14		PMC1: X1.5
	0	1	In 15		PMC1: X1.6
	0	1	In 16		PMC1: X1.7
	0	1	In 17		PMC1: X2.0

3.PMC I/O ASSIGNMENT

B-83254EN/02

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Input signal of process I/O board	0	1	In 18		PMC1: X2.1
	0	1	In 19	DI[1]	PMC1: X2.2
	0	1	In 20	DI[2]	PMC1: X2.3
	0	1	In 21	DI[3]	PMC1: X2.4
	0	1	In 22	DI[4]	PMC1: X2.5
	0	1	In 23	DI[5]	PMC1: X2.6
	0	1	In 24	DI[6]	PMC1: X2.7
	0	1	In 25	DI[7]	PMC1: X3.0
	0	1	In 26	DI[8]	PMC1: X3.1
	0	1	In 27	DI[9]	PMC1: X3.2
	0	1	In 28	DI[10]	PMC1: X3.3
	0	1	In 29	DI[11]	PMC1: X3.4
	0	1	In 30	DI[12]	PMC1: X3.5
	0	1	In 31	DI[13]	PMC1: X3.6
	0	1	In 32	DI[14]	PMC1: X3.7
	0	1	In 33	DI[15]	PMC1: X4.0
	0	1	In 34	DI[16]	PMC1: X4.1
	0	1	In 35	DI[17]	PMC1: X4.2
	0	1	In 36	DI[18]	PMC1: X4.3
	0	1	In 37	DI[19]	PMC1: X4.4
	0	1	In 38	DI[20]	PMC1: X4.5
	0	1	In 39	DI[21]	PMC1: X4.6
	0	1	In 40	DI[22]	PMC1: X4.7
Output signal process I/O board	0	1	Out 1		PMC1: Y0.0
	0	1	Out 2		PMC1: Y0.1
	0	1	Out 3		PMC1: Y0.2
	0	1	Out 4		PMC1: Y0.3
	0	1	Out 5		PMC1: Y0.4
	0	1	Out 6		PMC1: Y0.5
	0	1	Out 7		PMC1: Y0.6
	0	1	Out 8		PMC1: Y0.7
	0	1	Out 9		PMC1: Y1.0
	0	1	Out 10		PMC1: Y1.1
	0	1	Out 11		PMC1: Y1.2
	0	1	Out 12		PMC1: Y1.3
	0	1	Out 13		PMC1: Y1.4
	0	1	Out 14		PMC1: Y1.5
	0	1	Out 15		PMC1: Y1.6
	0	1	Out 16		PMC1: Y1.7
	0	1	Out 17		PMC1: Y2.0
	0	1	Out 18		PMC1: Y2.1
	0	1	Out 19		PMC1: Y2.2
	0	1	Out 20		PMC1: Y2.3
	0	1	Out 21	DO[1]	PMC1: Y2.4
	0	1	Out 22	DO[2]	PMC1: Y2.5
	0	1	Out 23	DO[3]	PMC1: Y2.6
	0	1	Out 24	DO[4]	PMC1: Y2.7
	0	1	Out 25	DO[5]	PMC1: Y3.0
	0	1	Out 26	DO[6]	PMC1: Y3.1
	0	1	Out 27	DO[7]	PMC1: Y3.2
	0	1	Out 28	DO[8]	PMC1: Y3.3
	0	1	Out 29	DO[9]	PMC1: Y3.4
	0	1	Out 30	DO[10]	PMC1: Y3.5

3.PMC I/O ASSIGNMENT

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Output signal process I/O board	0	1	Out 31	DO[11]	PMC1: Y3.6
	0	1	Out 32	DO[12]	PMC1: Y3.7
	0	1	Out 33	DO[13]	PMC1: Y4.0
	0	1	Out 34	DO[14]	PMC1: Y4.1
	0	1	Out 35	DO[15]	PMC1: Y4.2
	0	1	Out 36	DO[16]	PMC1: Y4.3
	0	1	Out 37	DO[17]	PMC1: Y4.4
	0	1	Out 38	DO[18]	PMC1: Y4.5
	0	1	Out 39	DO[19]	PMC1: Y4.6
	0	1	Out 40	DO[20]	PMC1: Y4.7
Memory of G (G0-1)				DI[41]	PMC1: G0.0
				DI[42]	PMC1: G0.1
				DI[43]	PMC1: G0.2
				DI[44]	PMC1: G0.3
				DI[45]	PMC1: G0.4
				DI[46]	PMC1: G0.5
				DI[47]	PMC1: G0.6
				DI[48]	PMC1: G0.7
				DI[49]	PMC1: G1.0
				DI[50]	PMC1: G1.1
				DI[51]	PMC1: G1.2
				DI[52]	PMC1: G1.3
				DI[53]	PMC1: G1.4
				DI[54]	PMC1: G1.5
				DI[55]	PMC1: G1.6
				DI[56]	PMC1: G1.7
Memory of F (F0-1)				DO[41]	PMC1: F0.0
				DO[42]	PMC1: F0.1
				DO[43]	PMC1: F0.2
				DO[44]	PMC1: F0.3
				DO[45]	PMC1: F0.4
				DO[46]	PMC1: F0.5
				DO[47]	PMC1: F0.6
				DO[48]	PMC1: F0.7
				DO[49]	PMC1: F1.0
				DO[50]	PMC1: F1.1
				DO[51]	PMC1: F1.2
				DO[52]	PMC1: F1.3
				DO[53]	PMC1: F1.4
				DO[54]	PMC1: F1.5
				DO[55]	PMC1: F1.6
				DO[56]	PMC1: F1.7
Memory of G (G1000-1003)				UI[1: IMSTP]	PMC1: G1000.0
				UI[2: HOLD]	PMC1: G1000.1
				UI[3: SFSPD]	PMC1: G1000.2
				UI[4: CSTOPI]	PMC1: G1000.3
				UI[5: RESET]	PMC1: G1000.4
				UI[6: START]	PMC1: G1000.5
				UI[7: HOME]	PMC1: G1000.6
				UI[8: ENBL]	PMC1: G1000.7
				UI[9: RSR1/PNS1]	PMC1: G1001.0
				UI[10: RSR2/PNS2]	PMC1: G1001.1
				UI[11: RSR3/PNS3]	PMC1: G1001.2

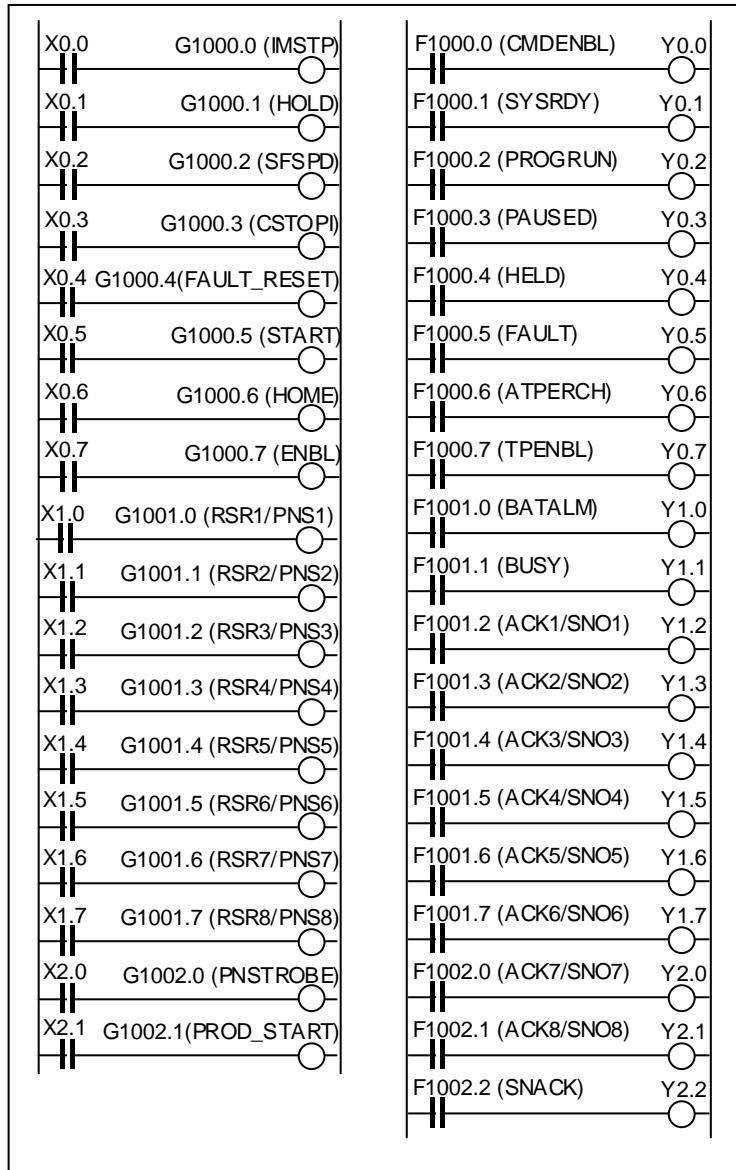
3.PMC I/O ASSIGNMENT

B-83254EN/02

Assigned to	Rack	Slot	Start	I/O of Robot	PMC address
Memory of G (G1000-1003)				UI[12: RSR4/PNS4]	PMC1: G1001.3
				UI[13: RSR5/PNS5]	PMC1: G1001.4
				UI[14: RSR6/PNS6]	PMC1: G1001.5
				UI[15: RSR7/PNS7]	PMC1: G1001.6
				UI[16: RSR8/PNS8]	PMC1: G1001.7
				UI[17: PNSTROBE]	PMC1: G1002.0
				UI[18: PROD_START]	PMC1: G1002.1
Memory of F (F1000-1003)				UO[1: CMDENBL]	PMC1: F1000.0
				UO[2: SYSRDY]	PMC1: F1000.1
				UO[3: PROGRUN]	PMC1: F1000.2
				UO[4: PAUSED]	PMC1: F1000.3
				UO[5: HELD]	PMC1: F1000.4
				UO[6: FAULT]	PMC1: F1000.5
				UO[7: ATPERCH]	PMC1: F1000.6
				UO[8: TPENBL]	PMC1: F1000.7
				UO[9: BATALM]	PMC1: F1001.0
				UO[10: BUSY]	PMC1: F1001.1
				UO[11: ACK1/SNO1]	PMC1: F1001.2
				UO[12: ACK2/SNO2]	PMC1: F1001.3
				UO[13: ACK3/SNO3]	PMC1: F1001.4
				UO[14: ACK4/SNO4]	PMC1: F1001.5
				UO[15: ACK5/SNO5]	PMC1: F1001.6
				UO[16: ACK6/SNO6]	PMC1: F1001.7
				UO[17: ACK7/SNO7]	PMC1: F1002.0
				UO[18: ACK8/SNO8]	PMC1: F1002.1
				UO[19: SNACK]	PMC1: F1002.2
				UO[20: Reserve]	PMC1: F1002.3

When the following sequence program is executed in this configuration, the UOP is controlled by the signals of the process I/O board as the same as the example 1. The sequence program copies the input signals (In 1-18) that were assigned to UI in the example 1 to the G1000.0-1002.1 that are assigned to UI in this configuration. And, it copies the F1000.0-1002.2 that are assigned to UO in this configuration to the output signals (Out 1-19) that were assigned to UO in the example 1.

By changing the sequence program, the UOP signals can be customized easily, for example, to change the condition of program start from the falling edge of the input signal (X0.5) to the raising edge.



3.5 COMPATIBILITY WITH CONVENTIONAL MODELS

Ladder program (LADDERx.PMC) compatibility

The sequence program of R-30iB (R-30iB ROBOT PMC) is highly compatible with the conventional models R-J3, R-J3iB, R-30iA, R-30iA Mate (RBT-SB5) on the source level.

You can use the sequence program of the conventional model on R-30iB by changing the PMC model using FANUC LADDER-III for Robot.

The LADDER.PMC file that is saved by the conventional model cannot be loaded to R-30iB as it is. The file format of the conventional model is "Handy-file Format File", but the file format of R-30iB is changed to "Memory card Format File". And the file name is changed to LADDER1.PMC because of multi path PMC function. Please import the LADDER.PMC as "Handy-file Format File" to FANUC LADDER-III for Robot, and convert the PMC model, and export to LADDER1.PMC as "Memory card Format File", then the LADDER1.PMC can be loaded to R-30iB. Please refer "3.5.1 The Convert Method of Source Program Using FANUC LADDER-III for Robot" for detail of PMC model conversion.

The conventional model does not have PMC external I/O assignment and PMC internal I/O assignment, and the meaning of each PMC address is fixed. To use the sequence program of the conventional model, it is necessary to setup PMC external I/O assignment and PMC internal I/O assignment to make the PMC address meaning same as the conventional model. The compatibility setup function achieves such setting in a lump. Please refer "3.5.2 Compatibility Setup Function" for the detail.

PMC parameter (PARAMx.PMC) compatibility

PMC parameters (PARAM.PMC) outputted from the conventional model R-J3, R-J3iB, R-30iA, R-30iA Mate (RBT-SB5) can be loaded into R-30iB with rename to PARAM1.PMC.

3.5.1 The Convert Method of Source Program Using FANUC LADDER-III for Robot

FANUC LADDER-III for Robot is used to convert a sequence program of other PMC models to R-30iB. Changing PMC model is possible easily with using the "PMC Type changed and save" function of FANUC LADDER-III for Robot.

(1) Converting with "PMC type changed and save" function.

The conversion procedure to the 1st path of R-30iB is as follows.

- i) Activate FANUC LADDER-III for Robot, and open the original ladder program for RBT-SB5.
- ii) Select [File] - [PMC Type changed and save].
- iii) Input the destination ladder program name, and select PMC Type, PMC Path and PMC Memory as follows, and press the [Ok] button.
 - PMC type: R-30iB ROBOT PMC
 - PMC Path: 1st Path
 - PMC Memory: B

(2) Converting with mnemonic conversion

The sequence of the conversion to the 1st path of R-30iB is as follows.

- i) Convert a source program into the mnemonic file by FANUC LADDER-III for Robot. ([Tool] -> [Source Program Convert])
- ii) Change the system parameters in the mnemonic file for RBT-SB5 by text editor. ("4 RBT-SB5" -> "4 R-30iB ROBOT PMC")

If the mnemonic file has insufficient parameters for the 1st path of R-30iB from RBT-SB5, the initial values are set with conversion for the source program.

The mnemonic file format of the system parameter for the 1st path of R-30iB is as follows.

%@0		
2 BINARY	2: Counter type	(BINAY or BCD)
3 NO	3: Operator panel	(YES or NO)
4 R-30iB ROBOT PMC	4: PMC type	(30i-B PMC)
31 1	31: Number of display language (comment)	(1-16)
32 -1	32: CNC display language number 1	(-1, 0-127)
33 0	33: Comment set number 1	(0-16)
%		

- iii) Create a new LAD file for 1st path of R-30iB by FANUC LADDER-III for Robot.
- iv) Convert the mnemonic file to the source program.([Tool] -> [Mnemonic Convert])

3.5.2 Compatibility Setup Function

The compatibility setup function performs the necessary settings in a lump to make the meaning of PMC address of 1st path is the same as the conventional model. By using this function, the sequence program of the conventional model can be used without changing PMC address.

After the compatibility setup function is executed, the meaning of PMC address is the following.

Letter	Type of signal	Range	Internal I/O assignment
X	Input to PMC	X0-127	Assign to DI[1-1024]
		X1000-1004	Assign to *UI[1-40] (External I/O device specified in UI assignment menu)
		X1005-1009	Assign to WI[1-40]
		X1010-1014	Assign to WSI[1-40]
		X1015-1019	
		X1020-1024	Assign to RI[1-40]
		X1025-1026	Assign to SI[0-15]
		X1027-1029	
		X1030-1034	Assign to *UI[41-80] (External I/O device specified in UI assignment menu)
		X1035-1039	
Y	Output from PMC	Y0-127	Assign to DO[1-1024]
		Y1000-1004	Assign to *UO[1-40] (External I/O device specified in UO assignment menu)
		Y1005-1009	Assign to WO[1-40]
		Y1010-1014	Assign to WSO[1-40]
		Y1015-1019	
		Y1020-1024	Assign to RO[1-40]
		Y1025-1026	Assign to SO[0-15]
		Y1027-1029	
		Y1030-1034	Assign to *UO[41-80] (External I/O device specified in UO assignment menu)
		Y1035-1039	
F	Input to PMC	F0-127	Transfer from GI[1-64]
		F128-255	Transfer from AI[1-64]
		F1000-1007	Assign to UO[1-64]
		F1008-1011	Assign to INFO[1-32]
		F1012-1255	
G	Output from PMC	G0-127	Synchronize with GO[1-64]
		G128-255	Synchronize with AO[1-64]
		G1000-1007	Assign to UI[1-64]
		G1008-1011	Assign to UALM[1-32]
		G1012-1255	
K	Keep relay	K0-16 K900-902	Assign to DO[10001-10136] Assign to DO[10137-10160]
R	General internal relay	R0-1499	Assign to DO[11001-23000]
		R9000-9117	
D	Data table	D0-2999	Assign to GO[10001-11500]

The setting of PMC internal I/O assignment to make this PMC address mapping is the following. When the compatibility setup function is executed, all setting in PMC external I/O assignment and PMC internal I/O assignment are deleted and the following setting is made.

PMC1 <RUNNING>			
	Internal I/O assignment	Size	Address
1	DI[1- 1024]	128	1:X00000
2	*UI[1- 40]	5	1:X01000
3	WI[1- 40]	5	1:X01005
4	WSI[1- 40]	5	1:X01010
5	RI[1- 40]	5	1:X01020
6	SI[0- 15]	2	1:X01025
7	*UI[41- 80]	5	1:X01030
8	DO[1- 1024]	128	1:Y00000
9	*UO[1- 40]	5	1:Y01000
10	WO[1- 40]	5	1:Y01005
11	WSO[1- 40]	5	1:Y01010
12	RO[1- 40]	5	1:Y01020
13	SO[0- 15]	2	1:Y01025
14	*UO[41- 80]	5	1:Y01030
15	GI[1- 64]	128	1:F00000
16	AI[1- 64]	128	1:F00128
17	UO[1- 64]	8	1:F01000
18	INFO[1- 32]	4	1:F01008
19	GO[1- 64]	128	1:G00000
20	AO[1- 64]	128	1:G00128
21	UI[1- 64]	8	1:G01000
22	UALM[1- 32]	4	1:G01008
23	DO[10001-10136]	17	1:K00000
24	DO[10137-10160]	3	1:K00900
25	DO[11001-23000]	1500	1:R00000
26	GO[10001-11500]	3000	1:D00000

[TYPE] [DATA] [FUNC] CLR_ALL

Execute the compatibility setup function

When the compatibility setup function is executed, the following process is performed.

- PMC internal I/O assignment and PMC external I/O assignment are deleted.
- The compatibility setting is made in PMC internal I/O assignment.

After the compatibility setup is executed, there is no way to recover the original setting. Please save the backup before executing the compatibility setup function.

To disable the compatibility setup, please clear all setting of PMC internal I/O assignment. By pressing F5(CLR_ALL), all setting of PMC internal I/O assignment can be cleared.

Operation to execute the compatibility setup function

- (1) MENU → "I/O" → F1[TYPE] → "PMC", then PMC menu is displayed.
- (2) Press F3[FUNC], and select "Compatibility setup" in the displayed menu.
- (3) The following message is displayed. Press F4(YES).

Execute compatibility setup function ?	
YES	NO
- (4) The following message is displayed. Press F4(OK).

Internal/external I/O asg are cleared.	
OK	CANCEL

PMC address assignment by rack 33

For the compatibility with conventional models, PMC address can be assigned to I/O of robot in I/O assignment menu as rack 33 slot 1-7. This is only for the compatibility purpose, and only the PMC address range that was available in the conventional models is supported.

The following is the PMC address that can be assigned in I/O assignment menu.

PMC address	Rack	Slot	Relation between address a.b and start point c
K0-16 of 1st path	33	1	$a \times 8 + b + 1 = c$
K900-902 of 1st path (note 1)	33	1	$(a - 900) \times 8 + b + 137 = c$
R0-1499 of 1st path	33	2	$a \times 8 + b + 1 = c$
D0-2999 of 1st path	33	3	$a \times 8 + b + 1 = c$
T0-79 of 1st path	33	4	$a \times 8 + b + 1 = c$
C0-79 of 1st path	33	5	$a \times 8 + b + 1 = c$
F1000-1006 of 1st path	33	6	$(a - 1000) \times 8 + b + 1 = c$
G1000-1006 of 1st path	33	7	$(a - 1000) \times 8 + b + 1 = c$

Note 1: The PMC setting parameter was K17-19 in the conventional models, but it is K900-999 in this model. For the compatibility with the conventional models, the rack 33, slot 1, start point 137-160 is corresponded to K900-902 instead of K17-19.

Instructions for compatibility setting

NOTE

Because there are assignments of X or Y in PMC internal I/O assignment for the compatibility setting, if X or Y address is specified in the parameter of functional instruction in the sequence program, the alarm "PRIO-136 BYTE access to x" occurs, and the sequence program is not executed. In this case, please copy the X or Y to internal relay by basic instruction, and specify the internal relay address in the parameter of the functional instruction.

NOTE

The override function is not available for the X or Y area that is assigned in the PMC internal I/O assignment menu. In these area, the simulated I/O status is reflected when the corresponded I/O is simulated. Please use I/O simulation instead of the override function.

NOTE

The synchronization processing of the 2nd level sequence part does not work for the X area that is assigned in the PMC internal I/O assignment menu. The signals in this area can be changed during 1 scan of the 2nd level sequence program. When the signal is read from two or more part of the 2nd level sequence part, please copy the signal to an internal relay at the beginning of the 2nd level sequence part so that the subsequent operation of the 2nd level sequence program part references the internal relay.

NOTE

Because there are assignments of X or Y in PMC internal I/O assignment for the compatibility setting, the PMC external I/O assignment is not available. Please delete all setting in PMC external I/O assignment menu.

Instructions for compatibility

The sequence program of the conventional model becomes available to be used in the current model by the execution of the compatible setup function, but there are some differences from the conventional model as follows. Please take care for the differences.

NOTE

- 1 GI/O and AI/O are synchronized with PMC address periodically. Therefore, the data synchronization can delay for the scanning time. If the delay causes problem, please assign the external I/O devices to X or Y by the PMC external I/O assignment menu, and please change the sequence program to access the X or Y address. The current processing period is displayed on the PMC internal I/O assignment menu. The processing period depends on whether the synchronized I/O is assigned to the external I/O device or not. When the all synchronized I/O is assigned to the external I/O device, the processing period is 32 msec.
- 2 In the conventional model, F1008.1(SYSRST) is set to 1 for only 1 scan after reset input. However, in the current model, INFO[2] (SYSRST) is set to 1 for 100 msec after reset input. If it causes problem that INFO[2] is set to 1 for 100 msec, please use DIFU (rising edge detection) instruction in the sequence program to read INFO[2].
- 3 In the conventional model, the PMC setting parameter was K17-19, but it is K900-999 in this model. If the sequence program accesses the PMC setting parameters (K17-19), the sequence program needs to be changed to access K900-902.
- 4 In the conventional model, the PMC function will be the initial configuration when the setting parameter K17.3 "Use standard ladder" is ENABLE (1). This setting parameter does not exist in this model, but the following setting makes the same condition.
 - Clear all UI and UO assignment in PMC internal I/O assignment menu.
 - Set the setting parameter K900.2 "Stop PMC at startup" to ENABLE (1).

4 LADDER LANGUAGE

4.1 BASIC INSTRUCTIONS

Designing a sequence program entails drawing a ladder diagram. Draw a ladder diagram by using relay contact symbols as well as symbols representing the functional instructions described later. The logic laid out in the ladder diagram is input to the programmer as a sequence program.

You can input a sequence program to the programmer in two ways - the relay symbol input method whereby relay contact symbols and functional instruction symbols drawn in the ladder diagram are used as they are (-| |-, -| /|-, -O-, etc.) and the mnemonic format input method that uses the mnemonic language (PMC instructions such as RD, AND, and OR).

The relay symbol input method allows the ladder diagram format to be used as it is, thus letting you input a sequence program in an intuitive, easy-to-understand manner. You will virtually have no trouble creating a program even if you have little or no knowledge of the PMC instructions (basic instructions such as RD, AND, and OR).

In fact, however, the content of a sequence program that is input using the relay symbol input method is internally converted to instructions that are equivalent to the corresponding PMC instructions. Also, you need to fully understand the functionalities of the functional instructions that are described later. It is therefore necessary for you to carefully read the descriptions of the basic and functional instructions that are given later in this manual.

When reading the descriptions of the PMC instructions, keep the following in mind.

(1) Signal addresses

An address is assigned to every relay coil and contact - that is, every signal - drawn in a ladder diagram (see Fig. 4.1 (a)). An address consists of an address number and a bit number. A zero at the beginning of an address may be omitted. For detailed information about addresses, see Section 2.2.

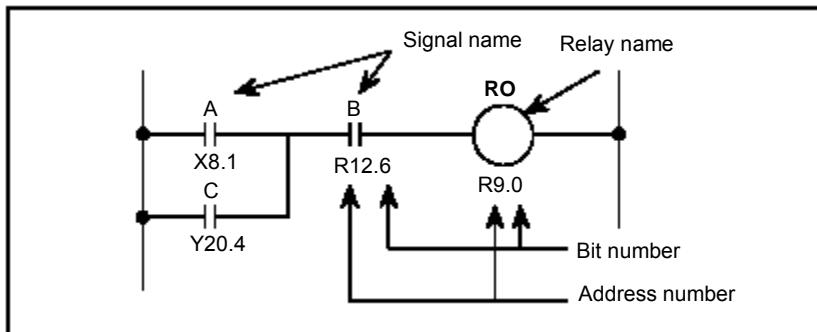


Fig. 4.1 (a) Signal addresses

(2) Types of instruction

There are two types of PMC instruction - basic instructions and functional instructions.

(a) Basic instructions

The basic instructions are most frequently used in designing a sequence program. There are 24 instructions, including AND and OR, each of which performs a one-bit operation.

(b) Functional instructions

The functional instructions are intended to make it easy to program those machine operations that are difficult to code with the basic instructions alone. For the types of functional instructions, see Subsection 2.1.7 or 2.1.8.

(3) Storage of logical operation results

There is a register that stores the interim results of logical operations during the execution of a sequence program.

This register consists of a total of nine bits, which is divided into a one-bit segment and an eight-bit segment as shown in Fig. 4.1 (b).

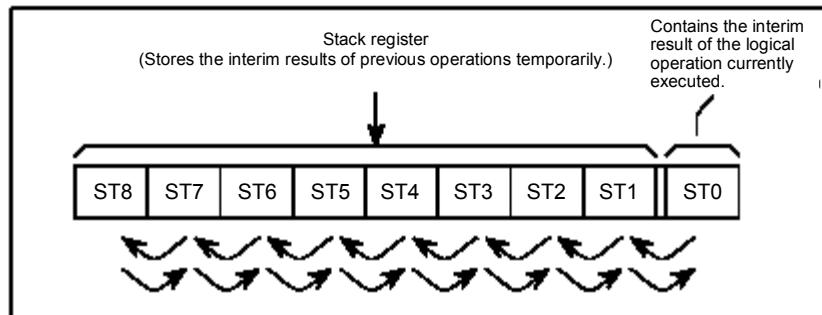


Fig. 4.1 (b) Structure of the register storing the results of logical operations

When an instruction (such as RD.STK) that temporarily stores the interim result of a logical operation is executed, the current content of the register is shifted to the left and the interim logical operation result is stacked in the register, as shown in the above figure. Conversely, when an instruction (such as AND.STK) that retrieves a stacked signal is executed, the register content is shifted to the right and the signal is retrieved. The last stacked signal is retrieved first. For information about the actual uses and operations of these instructions, see the relevant descriptions in this manual.

4.1.1 Details of the Basic Instructions

Table 4.1.1 lists the types of the basic instructions and explains the processing they perform. The difference between the two types of formats shown under Instruction is described below.

Mnemonic format:

The instructions are displayed in this format when you edit or print a ladder program that has been converted to the mnemonic format with FANUC LADDER-III for Robot, by using a commercially available text editor.

Mnemonic format (abbreviated):

These are the abbreviated forms of instructions that you can use when editing a ladder program that has been converted to the mnemonic format with FANUC LADDER-III for Robot, by using a commercially available text editor. If you input a file in this abbreviated format and convert it again to the ladder diagram format with FANUC LADDER-III for Robot, the code in the file can still be recognized as being written in the valid mnemonic format.

Detailed explanations of the individual basic instructions follow.

Table 4.1.1

No.	Instruction		Processing
	Mnemonic format	Mnemonic format (abbreviated)	
1	RD	R	Reads the status of the specified signal and sets it in the ST0 bit.
2	RD.NOT	RN	Reads and reverses the logical status of the specified signal and sets it in the ST0 bit.
3	WRT	W	Outputs the logical operation result (the status of the ST0 bit) to the specified address.
4	WRT.NOT	WN	Reverses and outputs the logical operation result (the status of the ST0 bit) to the specified address.
5	AND	A	Produces a logical product.

No.	Instruction		Processing
	Mnemonic format	Mnemonic format (abbreviated)	
6	AND.NOT	AN	Reverses the logical status of the specified signal and produces a logical product.
7	OR	O	Produces a logical sum.
8	OR.NOT	ON	Reverses the logical status of the specified signal and produces a logical sum.
9	RD.STK	RS	Shifts the register content one bit to the left and sets the status of the signal at the specified address in the ST0 bit.
10	RD.NOT.STK	RNS	Shifts the register content one bit to the left, reads and reverses the logical status of the signal at the specified address, and sets it in the ST0 bit.
11	AND.STK	AS	Sets the logical product of the ST0 and ST1 bits in the ST1 bit and shifts the register content one bit to the right.
12	OR.STK	OS	Sets the logical sum of the ST0 and ST1 bits in the ST1 bit and shifts the register content one bit to the right.
13	SET	SET	Finds the logical sum of the ST0 bit and the status of the signal at the specified address and outputs it to the specified address.
14	RST	RST	Finds the logical product of the reversed status of the ST0 bit and the status of the signal at the specified address and outputs it to the specified address.
15	RDPT	RPT	Positive transition contact instruction. When rising transition (0→1) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.
16	ANDPT	APT	Positive transition contact instruction. When rising transition (0→1) of the specified signal is detected, ST0 bit is not changed. Otherwise "0" is set to the ST0 bit.
17	ORPT	OPT	Positive transition contact instruction. When rising transition (0→1) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise ST0 bit is not changed.
18	RDPT.STK	RPTS	Positive transition contact instruction. Shifts the stack register content one bit to the left and when rising transition (0→1) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit .
19	RDNT	RNT	Negative transition contact instruction. When falling transition (1→0) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.
20	ANDNT	ANT	Negative transition contact instruction. When falling transition (1→0) of the specified signal is detected ST0 bit is not changed. Otherwise "0" is set to the ST0 bit.
21	ORNT	ONT	Negative transition contact instruction. When falling transition (1→0) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise ST0 bit is not changed.
22	RDNT.STK	RNTS	Negative transition contact instruction. Shifts the stack register content one bit to the left and when falling transition (1→0) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.
23	PUSH	PS	Instruction to make a branch of circuit. Shifts the stack register one bit to the left. The contents of ST0 bit is not changed.
24	POP	PP	Instruction to make a branch of circuit. Shifts the stack register content one bit to the right. (ST1→ST0)

4.1.2 RD Instruction

Format

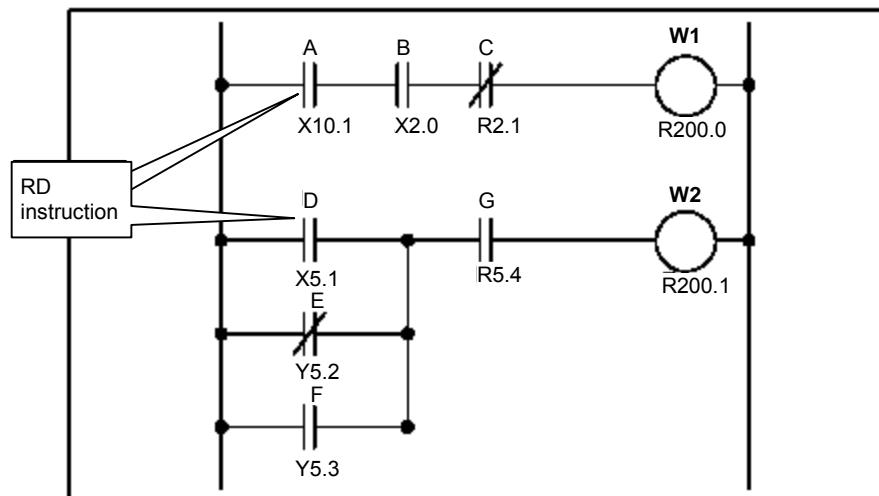


Fig. 4.1.2

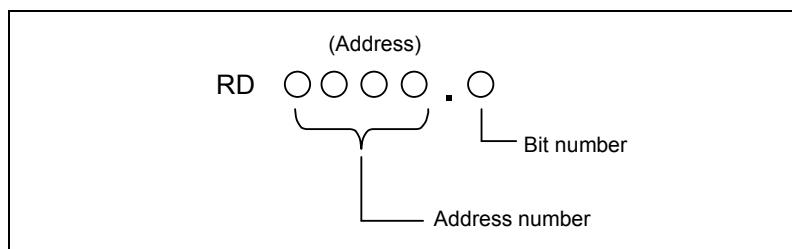


Table 4.1.2

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .1		A
2	AND	X2 .0		B
3	AND.NOT	R2 .1		C
4	WRT	R200 .0		W1 output
5	RD	X5 .1		D
6	OR.NOT	Y5 .2		E
7	OR	Y5 .3		F
8	AND	R5 .4		G
9	WRT	R200 .1		W2 output

ST2	ST1	ST0
		A
		A·B
		A·B·C
		A·B·C̄
		D
		D + Ē
		D + Ē + F
		(D + Ē + F)·G
		(D + Ē + F)·G

Operation

- (1) Use this instruction to start coding from contact A (-| | -). For examples of how the RD instruction is used, see the ladder diagram shown in Fig. 4.1.2 and the input example in the mnemonic format given in Table 4.1.2.
- (2) The instruction reads the status (0 or 1) of the signal at the specified address and sets it in the ST0 bit.
- (3) The signal (contact) to be read by the RD instruction may be any signal (contact) that is input as a logical condition of a coil (output).

4.1.3 RD.NOT Instruction

Format

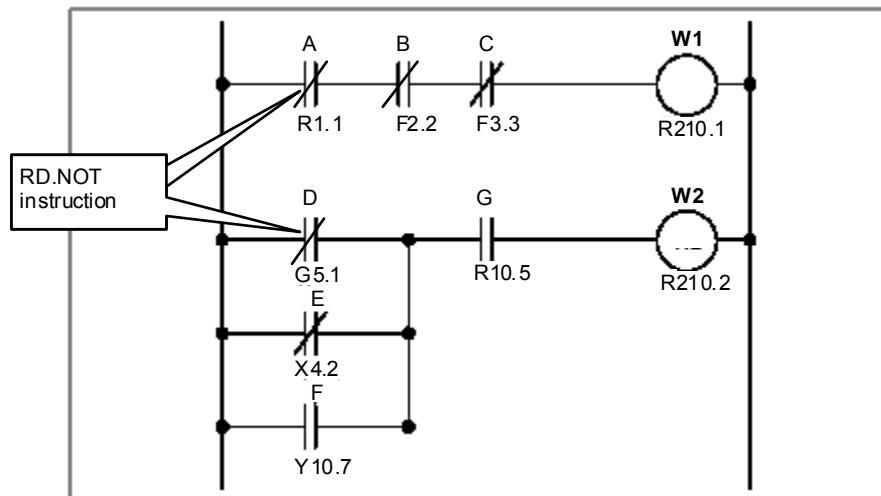


Fig. 4.1.3

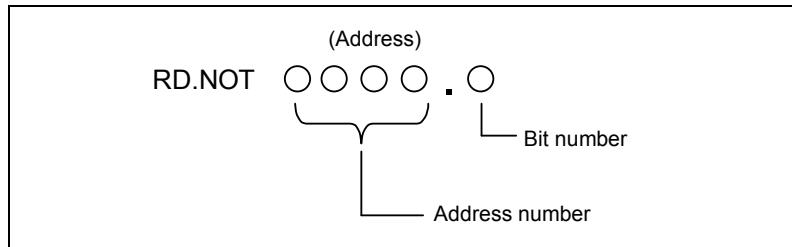


Table 4.1.3

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD.NOT	R1.1		A
2	AND.NOT	F2.2		B
3	AND.NOT	F3.3		C
4	WRT	R210.1		W1 output
5	RD.NOT	G5.1		D
6	OR.NOT	X4.2		E
7	OR	Y10.7		F
8	AND	R10.5		G
9	WRT	R210.2		W2 output

ST2	ST1	ST0
		\bar{A}
		$\bar{A} \cdot \bar{B}$
		$\bar{A} \cdot \bar{B} \cdot \bar{C}$
		$\bar{A} \cdot \bar{B} \cdot \bar{C}$
		\bar{D}
		$\bar{D} + \bar{E}$
		$\bar{D} + \bar{E} + \bar{F}$
		$(\bar{D} + \bar{E} + \bar{F}) \cdot G$
		$(\bar{D} + \bar{E} + \bar{F}) \cdot G$

Operation

- (1) Use this instruction to start coding from contact B ($-|/|-$). For examples of how the RD.NOT instruction is used, see the ladder diagram shown in Fig. 4.1.3 and the input example in the mnemonic format given in Table 4.1.3.
- (2) The instruction reads and reverses the logical status of the signal at the specified address and sets it in the ST0 bit.
- (3) The signal (contact) to be read by the RD.NOT instruction may be any contact B that is input as a logical condition of a coil.

4.1.4 WRT Instruction

Format

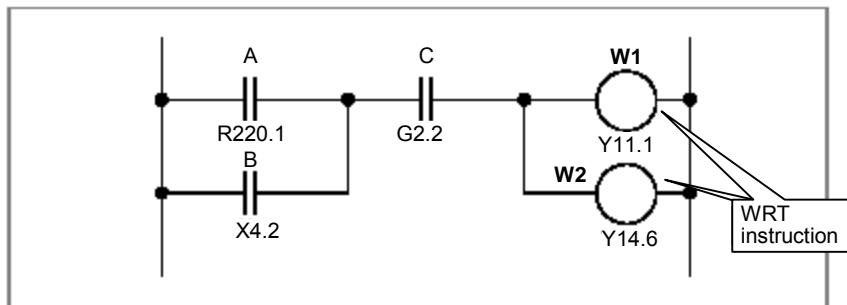


Fig. 4.1.4

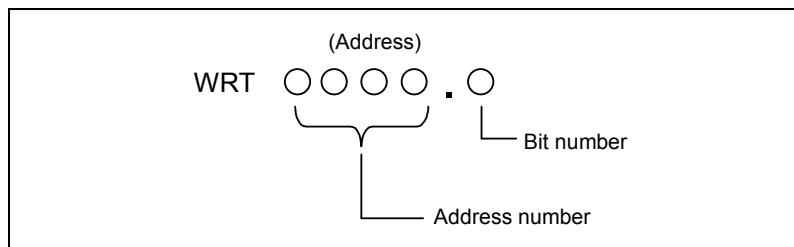


Table 4.1.4

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	R220 .1		A
2	OR	X4 .2		B
3	AND	G2 .2		C
4	WRT	Y11 .1		W1 output
5	WRT	Y14 .6		W2 output

ST2	ST1	ST0
		A
		A + B
		(A + B)·C
		(A + B)·C
		(A + B)·C

Operation

- (1) The WRT instruction outputs the result of the logical operation, namely the status of the ST0 bit (0 or 1), to the specified address.
- (2) The instruction can also output a logical operation result to two or more addresses simultaneously. In that case, use the WRT instruction as shown in Fig. 4.1.4 and Table 4.1.4.

CAUTION

In each WRT,WRT.NOT instruction, specify different address. Double coil, which means a coil with an address is used more than once in a ladder program, may have troubles of the execution timing in the sequence program. Don't use "double coil".

4.1.5 WRT.NOT Instruction

Format

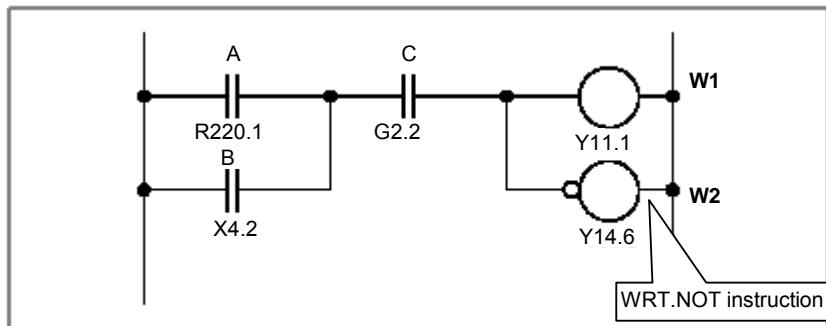


Fig. 4.1.5

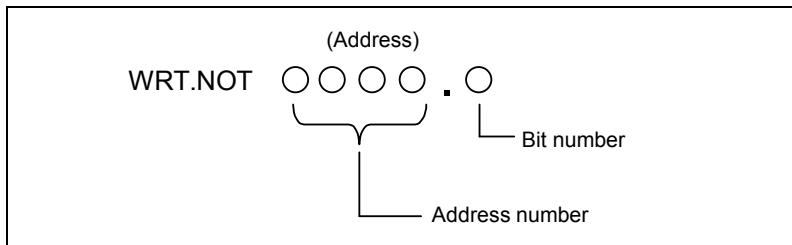


Table 4.1.5

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	R220 .1		A
2	OR	X4 .2		B
3	AND	G2 .2		C
4	WRT	Y11 .1		W1 output
5	WRT.NOT	Y14 .6		W2 output

ST2	ST1	ST0
		A
		A + B
		(A + B)·C
		(A + B)·C
		(A + B)·C

Operation

The WRT.NOT instruction reverses and outputs the result of the logical operation, namely the status of the ST0 bit, to the specified address. Fig. 4.1.5 and Table 4.1.5 show examples of how the WRT.NOT instruction is used.

⚠ CAUTION

In each WRT,WRT.NOT instruction, specify different address. Double coil, which means a coil with an address is used more than once in a ladder program, may have troubles of the execution timing in the sequence program. Don't use "double coil".

4.1.6 AND Instruction

Format

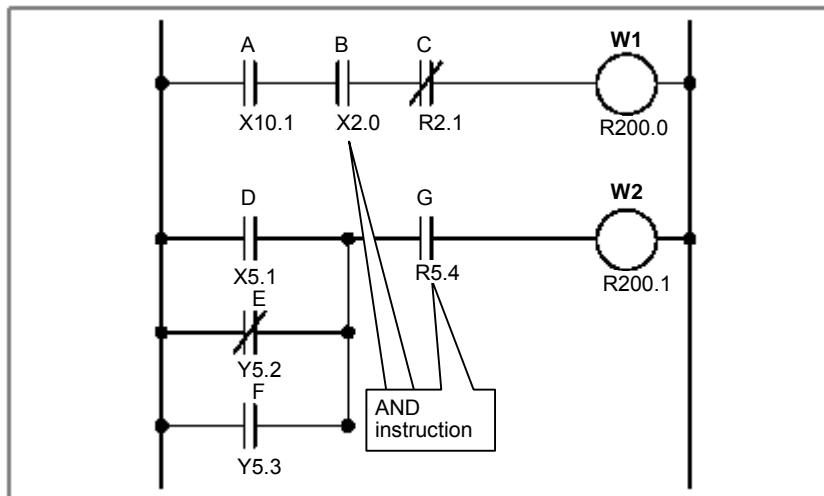


Fig. 4.1.6

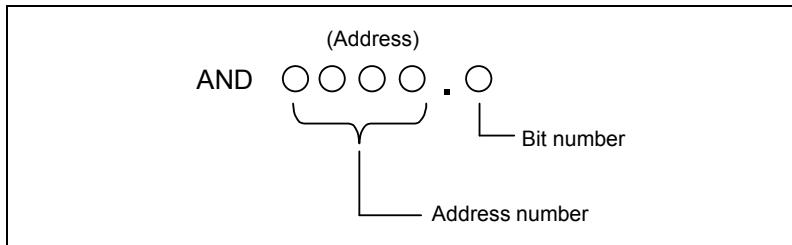


Table 4.1.6

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .1		A
2	AND	X2 .0		B
3	AND.NOT	R2 .1		C
4	WRT	R200 .0		W1 output
5	RD	X5 .1		D
6	OR.NOT	Y5 .2		E
7	OR	Y5 .3		F
8	AND	R5 .4		G
9	WRT	R200 .1		W2 output

ST2	ST1	ST0
		A
		A·B
		A·B·C
		A·B·C̄
		D
		D + Ē
		D + Ē + F
		(D + Ē + F)·G
		(D + Ē + F)·G

Operation

- (1) This instruction produces a logical product.
- (2) For examples of how the AND instruction is used, see Fig. 4.1.6 and Table 4.1.6.

4.1.7 AND,NOT Instruction

Format

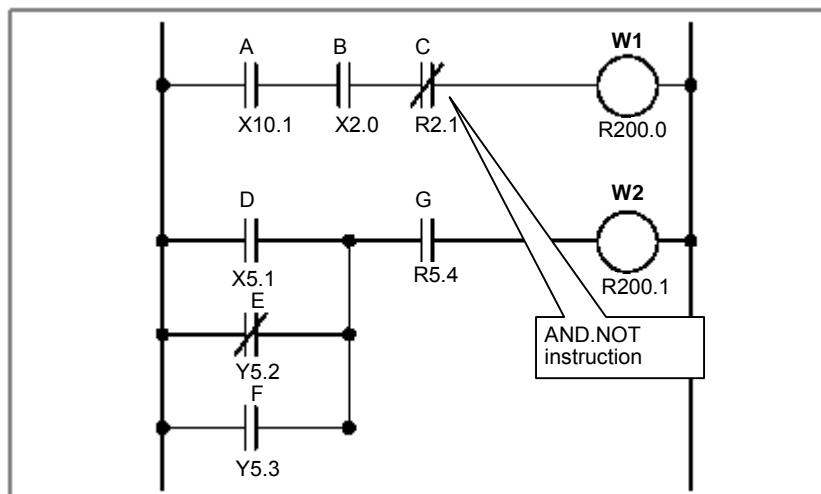


Fig. 4.1.7

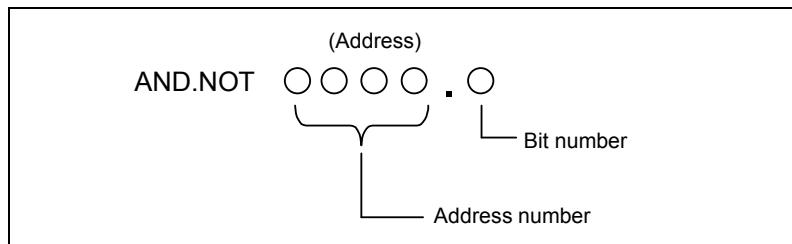


Table 4.1.7

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .1		A
2	AND	X2 .0		B
3	AND.NOT	R2 .1		C
4	WRT	R200 .0		W1 output
5	RD	X5 .1		D
6	OR.NOT	Y5 .2		E
7	OR	Y5 .3		F
8	AND	R5 .4		G
9	WRT	R200 .1		W2 output

ST2	ST1	ST0
		A
		A·B
		A·B·C
		A·B·C
		D
		D + E
		D + E + F
		(D + E + F)·G
		(D + E + F)·G

Operation

- (1) This instruction reverses the status of the signal at the specified address and produces a logical product.
 - (2) For examples of how the AND.NOT instruction is used, see Fig. 4.1.7 and Table 4.1.7.

4.1.8 OR Instruction

Format

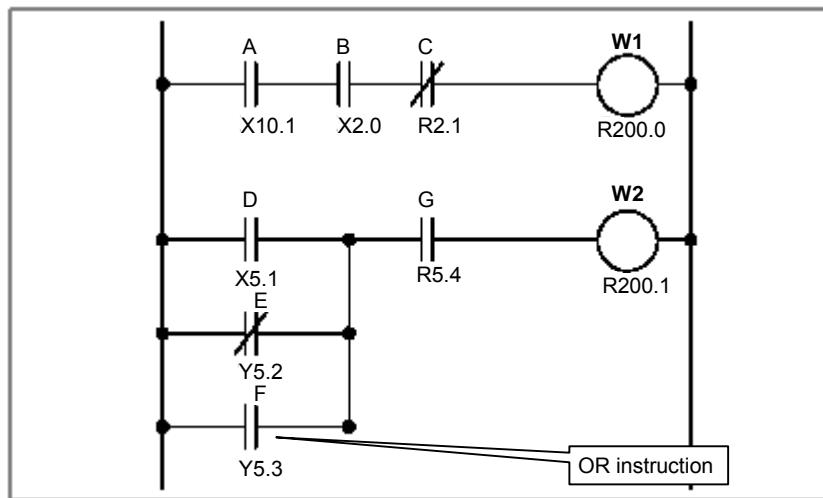


Fig. 4.1.8

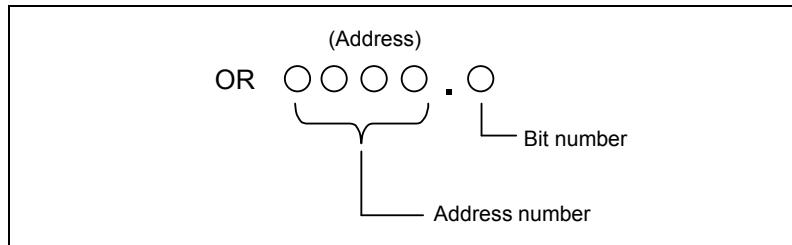


Table 4.1.8

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .1		A
2	AND	X2 .0		B
3	AND.NOT	R2 .1		C
4	WRT	R200 .0		W1 output
5	RD	X5 .1		D
6	OR.NOT	Y5 .2		E
7	OR	Y5 .3		F
8	AND	R5 .4		G
9	WRT	R200 .1		W2 output

ST2	ST1	ST0
		A
		A·B
		A·B·C
		A·B·C̄
		D
		D + E
		D + Ē + F
		(D + Ē + F)·G
		(D + Ē + F)·G

Operation

- (1) This instruction produces a logical sum.
- (2) For examples of how the OR instruction is used, see Fig. 4.1.8 and Table 4.1.8.

4.1.9 OR.NOT Instruction

Format

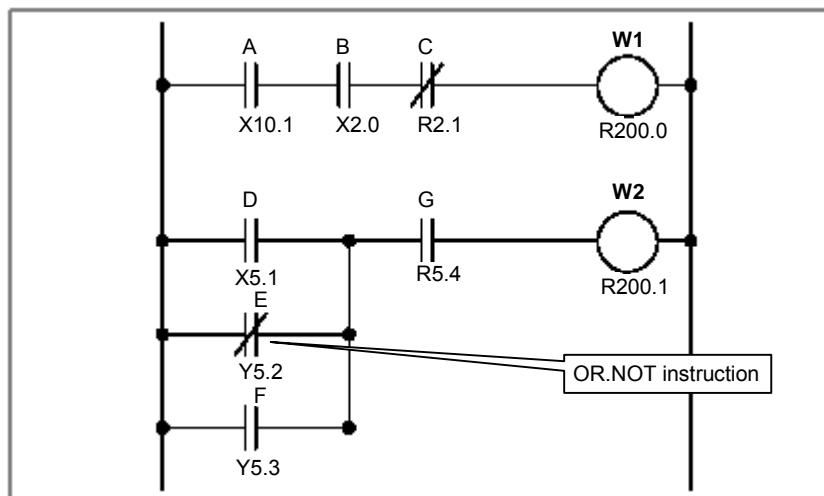


Fig. 4.1.9

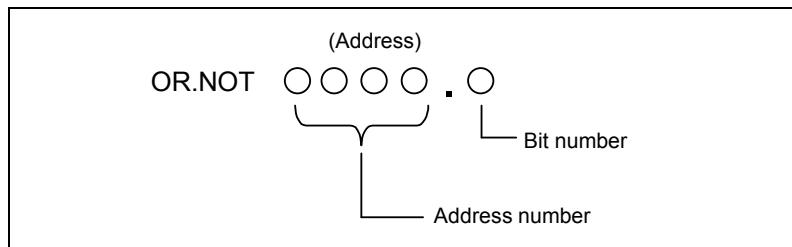


Table 4.1.9

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .1		A
2	AND	X2 .0		B
3	AND.NOT	R2 .1		C
4	WRT	R200 .0		W1 output
5	RD	X5 .1		D
6	OR.NOT	Y5 .2		E
7	OR	Y5 .3		F
8	AND	R5 .4		G
9	WRT	R200 .1		W2 output

ST2	ST1	ST0
		A
		A·B
		A·B·C
		A·B·C
		D
		D + E
		D + E + F
		(D + E + F)·G
		(D + E + F)·G

Operation

- (1) This instruction reverses the status of the signal at the specified address and produces a logical sum.
 - (2) For examples of how the OR.NOT instruction is used, see Fig. 4.1.9 and Table 4.1.9.

4.1.10 RD.STK Instruction

Format

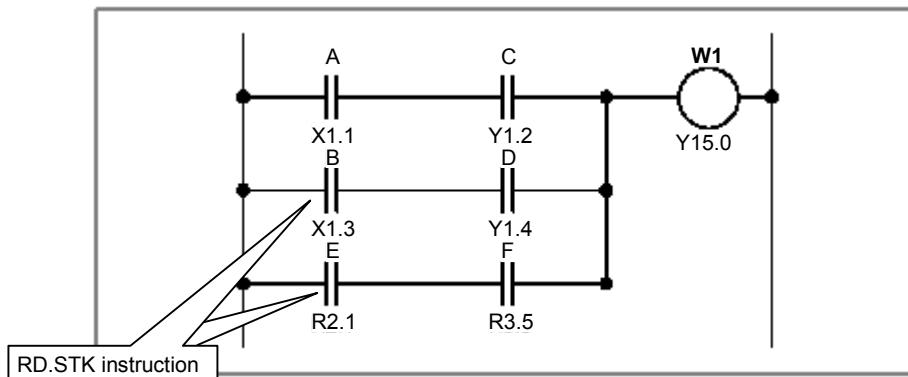


Fig. 4.1.10

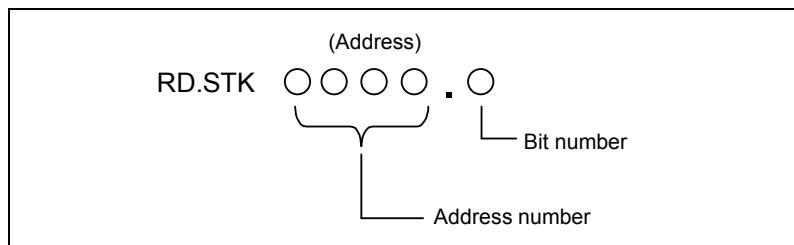


Table 4.1.10

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X1 .1		A
2	AND	Y1 .2		C
3	RD.STK	X1 .3		B
4	AND	Y1 .4		D
5	OR.STK			
6	RD.STK	R2 .1		E
7	AND	R3 .5		F
8	OR.STK			
9	WRT	Y15 .0		W1 output

ST2	ST1	ST0
		A
		A·C
	A·C	B
	A·C	B·D
		A·C + B·D
	A·C + B·D	E
	A·C + B·D	E·F
		A·C + B·D + E·F
		A·C + B·D + E·F

Operation

- (1) The RD.STK instruction stacks the interim result of a logical operation. Use this instruction when the signal you specify is contact A (- | -). After shifting the register content one bit to the left, the instruction sets the status of the signal at the specified address in the ST0 bit.
- (2) For examples of how the RD.STK instruction is used, see Fig. 4.1.10 and Table 4.1.10.

4.1.11 RD.NOT.STK Instruction

Format

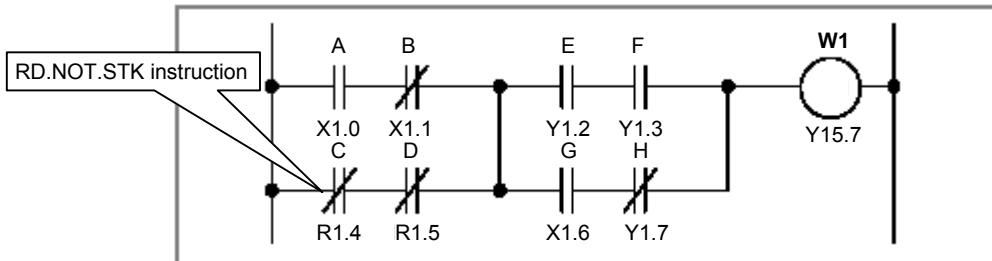


Fig. 4.1.11

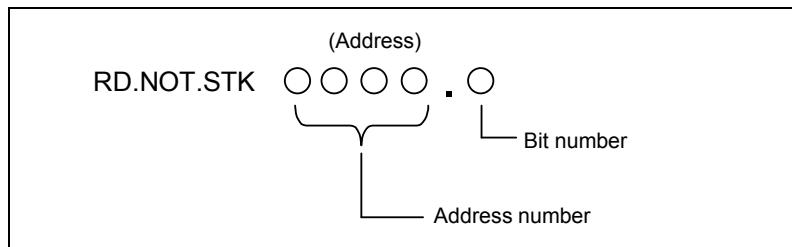


Table 4.1.11

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X1 .0	A	
2	AND.NOT	X1 .1	B	
3	RD.NOT.STK	R1 .4	C	
4	AND.NOT	R1 .5	D	
5	OR.STK			
6	RD.STK	Y1 .2	E	
7	AND	Y1 .3	F	
8	RD.STK	Y1 .6	G	
9	AND.NOT	Y1 .7	H	
10	OR.STK			
11	AND.STK			
12	WRT	Y15 .7	W1 output	

ST2	ST1	ST0
		A
		$A \bar{B}$
	$A \bar{B}$	\bar{C}
	$A \bar{B}$	$\bar{C} \bar{D}$
		$A \bar{B} + \bar{C} \bar{D}$
	$A \bar{B} + \bar{C} \bar{D}$	E
	$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$
$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$	G
$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$	$G \cdot \bar{H}$
	$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F + G \cdot \bar{H}$
		$(A \bar{B} + \bar{C} \bar{D}) \cdot (E \cdot F + G \cdot \bar{H})$
		$(A \bar{B} + \bar{C} \bar{D}) \cdot (E \cdot F + G \cdot \bar{H})$

Operation

- (1) The RD.NOT.STK instruction stacks the interim result of a logical operation. Use this instruction when the signal you specify is contact B ($-|/|-$). After shifting the register content one bit to the left, the instruction reverses the status of the signal at the specified address and sets it in the ST0 bit.
- (2) For examples of how the RD.NOT.STK instruction is used, see Fig. 4.1.11 and Table 4.1.11.

4.1.12 AND.STK Instruction

Format

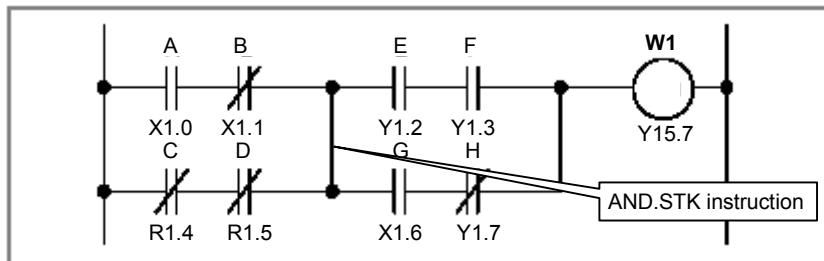


Fig. 4.1.12 (a)

AND.STK

Table 4.1.12

Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X1 .0		A
2	AND.NOT	X1 .1		B
3	RD.NOT.STK	R1 .4		C
4	AND.NOT	R1 .5		D
5	OR.STK			
6	RD.STK	Y1 .2		E
7	AND	Y1 .3		F
8	RD.STK	Y1 .6		G
9	AND.NOT	Y1 .7		H
10	OR.STK			
11	AND.STK			
12	WRT	Y15 .7		W1 output

Status of operation result		
ST2	ST1	ST0
		A
		$A \bar{B}$
	$A \bar{B}$	\bar{C}
	$A \bar{B}$	$\bar{C} \bar{D}$
		$A \bar{B} + \bar{C} \bar{D}$
	$A \bar{B} + \bar{C} \bar{D}$	E
	$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$
$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$	G
$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F$	$G \bar{H}$
	$A \bar{B} + \bar{C} \bar{D}$	$E \cdot F + G \bar{H}$
		$(A \bar{B} + \bar{C} \bar{D}) \cdot (E \cdot F + G \bar{H})$
		$(A \bar{B} + \bar{C} \bar{D}) \cdot (E \cdot F + G \bar{H})$

Operation

- (1) The AND.STK instruction finds the logical product of the operation result stored in the ST0 bit and that stored in the ST1 bit and sets it in the ST1 bit. The instruction then shifts the register content one bit to the right and puts the resulting logical product into the ST0 bit. Fig. 4.1.12 (b) shows a detailed image of what is shown in Fig. 4.1.12 (a).

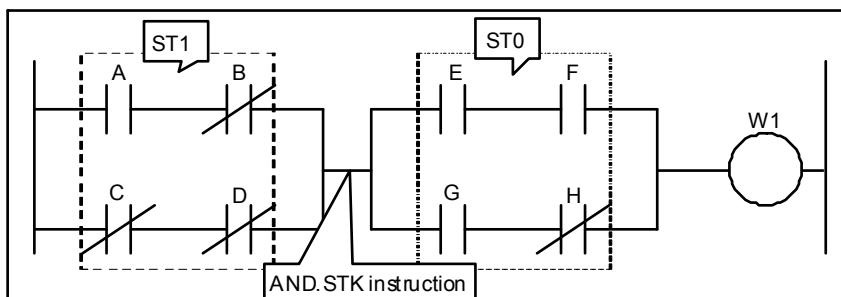


Fig. 4.1.12 (b)

- (2) For examples of how the AND.STK instruction is used, see Fig. 4.1.12 (a) and Table 4.1.12.

4.1.13 OR.STK Instruction

Format

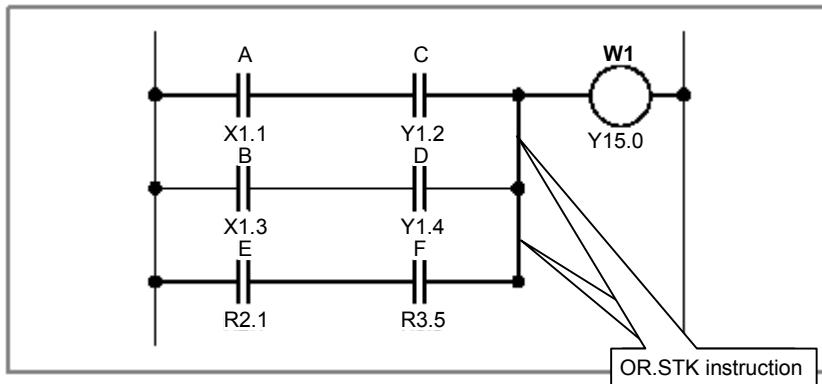


Fig. 4.1.13 (a)



Table 4.1.13

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X1 .1		A
2	AND	Y1 .2		C
3	RD.STK	X1 .3		B
4	AND	Y1 .4		D
5	OR.STK			
6	RD.STK	R2 .1		E
7	AND	R3 .5		F
8	OR.STK			
9	WRT	Y15 .0		W1 output

ST2	ST1	ST0
		A
		A·C
	A·C	B
	A·C	B·D
		A·C + B·D
	A·C + B·D	E
	A·C + B·D	E·F
		A·C + B·D + E·F
		A·C + B·D + E·F

Operation

- (1) The OR.STK instruction finds the logical sum of the operation result stored in the ST0 bit and that stored in the ST1 bit and sets it in the ST1 bit. The instruction then shifts the register content one bit to the right and puts the resulting logical sum into the ST0 bit. Fig. 4.1.13 (b) shows a detailed image of what is shown in Fig. 4.1.13 (a).

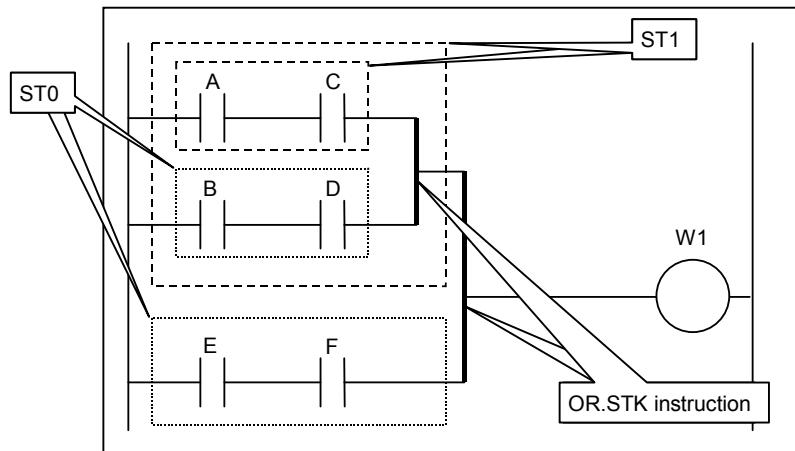


Fig. 4.1.13 (b)

- (2) For examples of how the OR.STK instruction is used, see Fig. 4.1.13 (a) and Table 4.1.13.

⚠ CAUTION

In the example shown in Table 4.1.13, the OR.STK instruction is specified at step number 5. You will obtain the same result if you place the OR.STK instruction between step numbers 7 and 8. However, coding similar instructions, such as OR.STK and AND.STK, successively makes you prone to errors. It is therefore recommended to code your program as shown in Table 4.1.13.

4.1.14 SET Instruction

Format

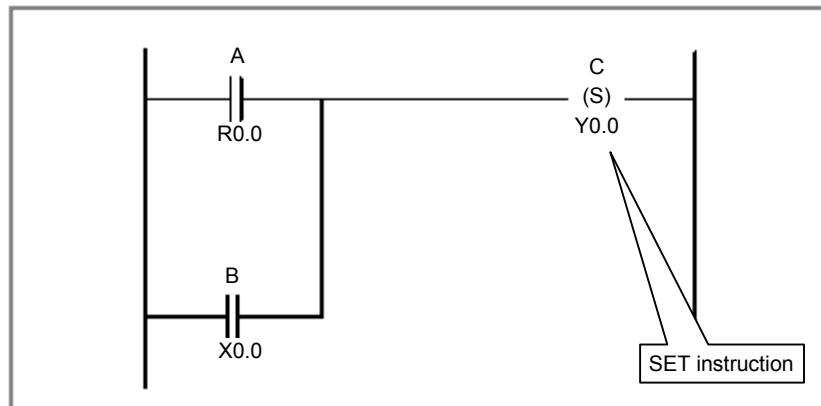


Fig. 4.1.14

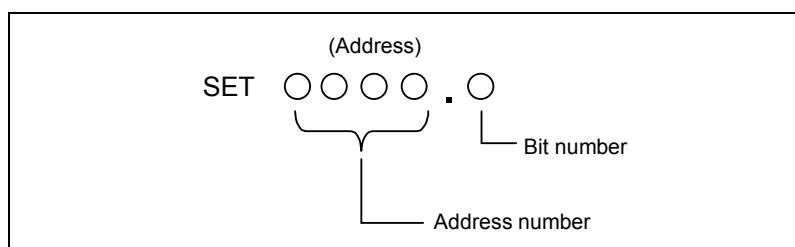


Table 4.1.14

Mnemonic format					Status of operation result		
Step number	Instruction	Address No.	Bit No.	Remarks	ST2	ST1	ST0
1	RD	R0 .0		A			A
2	OR	X0 .0		B			A + B
3	SET	Y0 .0		Y0.0 output			A + B

Operation

- (1) This instruction keeps the status of the specified address to ON. It finds the logical sum of the operation result (ST0) and the specified address and outputs it to the specified address.
- (2) For examples of how the SET instruction is used, see Fig. 4.1.14 and Table 4.1.14.
- (3) Caution
 - Relationship with COM and COME
When placed between the COM and COME instructions, the SET instruction behaves as follows:
When the COM condition is set to ON (ACT = 1), the SET instruction runs normally.
When the COM condition is set to OFF (ACT = 0), the SET instruction does not run.

4.1.15 RST Instruction

Format

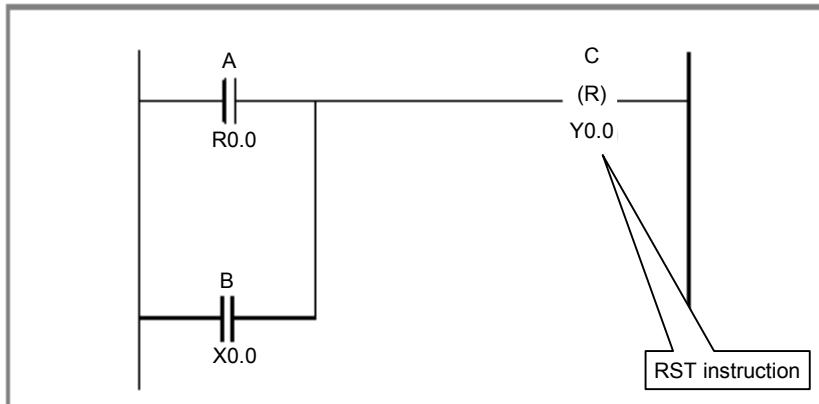


Fig. 4.1.15

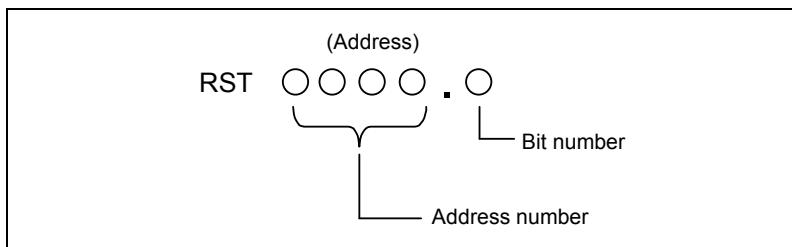


Table 4.1.15

Mnemonic format					Status of operation result		
Step number	Instruction	Address No.	Bit No.	Remarks	ST2	ST1	ST0
1	RD	R0 .0		A			A
2	OR	X0 .0		B			A + B
3	RST	Y0 .0		Y0.0 output			A + B

Operation

- (1) This instruction keeps the status of the specified address to OFF. It finds the logical product of the operation result (ST0) and the specified reversed address and outputs it to the specified address.
- (2) For examples of how the RST instruction is used, see Fig. 4.1.15 and Table 4.1.15.
- (3) Caution
 - Relationship with COM and COME
When placed between the COM and COME instructions, the RST instruction behaves as follows:
When the COM condition is set to ON (ACT = 1), the RST instruction runs normally.
When the COM condition is set to OFF (ACT = 0), the RST instruction does not run.

4.1.16 RDPT Instruction

Positive transition contact instruction. When rising transition ($0 \rightarrow 1$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.16(a) shows the ladder format and Table 4.1.16(a) shows the mnemonic format.

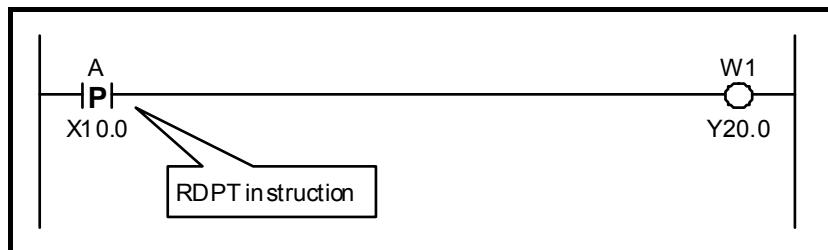


Fig. 4.1.16 (a) Format of RDPT instruction

Table 4.1.16 Mnemonic of RDPT instruction

Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RDPT	X10.0		A
2	WRT	Y20.0		W1 output

Status of operation result

ST2	ST1	ST0
		A(PT)
		A(PT)

Operation

Timing chart in the above example is as follows.

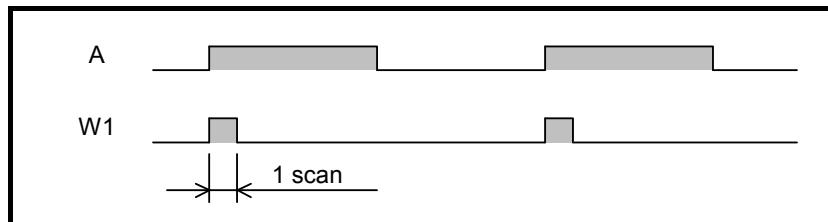


Fig. 4.1.16 (b) Timing chart of RDPT instruction

! CAUTION

- 1 The bit, already turned on when a program is started after program reading by the Input/Output function or Power ON, turns on the output with the scan at the beginning just after starting.
 - 2 An output may not be turned on when a bit changes with OFF→ON→OFF during 1 scan. Moreover, when validating the result of ladder edit, scan time temporarily becomes larger.
 - 3 In ladder edit, when the bit of the edited contact turns on, an output is turned on with the scan beginning after the edit.
 - 4 When this instruction is skipped by Jump instruction or subroutine call instruction, this instruction is not executed and the output of instruction does not change.
 - 5 This instruction uses a work memory internally in order to detect bit transition. The FANUC LADDER-III for Robot searches the work memory automatically in the domain which can be used, and is assigned. Therefore, the program edited in a different procedure becomes mismatching at the comparing, even when the appearance of ladder diagram is the same.

4.1.17 ANDPT Instruction

Positive transition contact instruction. When rising transition ($0 \rightarrow 1$) of the specified signal is detected, ST0 bit is not changed. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.17(a) shows the ladder format and Table 4.1.17(a) shows the mnemonic format.

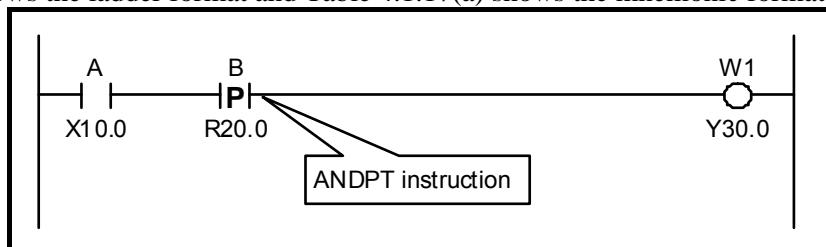


Fig. 4.1.17 (a) Format of ANDPT Instruction

Table 4.1.17 Mnemonic of ANDPT Instruction

Table 1.1.1.1. Mnemonic of ANDT1 instruction

Mnemonic format				
Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	ANDPT	R20 .0		B
3	WRT	Y30 .0		W1 output

ST2	ST1	ST0
		A
		$A \bullet B \text{ (PT)}$
		$A \bullet B \text{ (PT)}$

Operation

Timing chart in the above example is as follows.

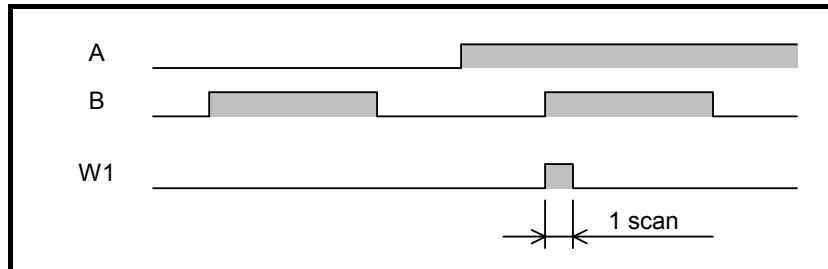


Fig. 4.1.17 (b) Timing chart of ANDPT Instruction

NOTE

Please refer to "4.1.16 RDPT Instruction" notes about this instruction.

4.1.18 ORPT Instruction

Positive transition contact instruction. When rising transition ($0 \rightarrow 1$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise ST0 bit is not changed.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.18(a) shows the ladder format and Table 4.1.18(a) shows the mnemonic format.

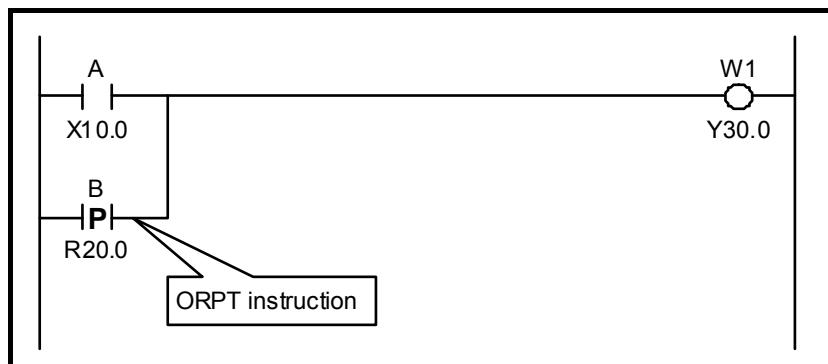


Fig. 4.1.18 (a) Format of ORPT Instruction

Table 4.1.18 Mnemonic of ORPT Instruction

Table 1.1.16 Mnemonic of CPU 1 instruction

Mnemonic format				
Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	ORPT	R20 .0		B
3	WRT	Y30 .0		W1 output

Status of operation result		
ST2	ST1	ST0
		A
		A + B (PT)
		A + B (PT)

Operation

Timing chart in the above example is as follows.

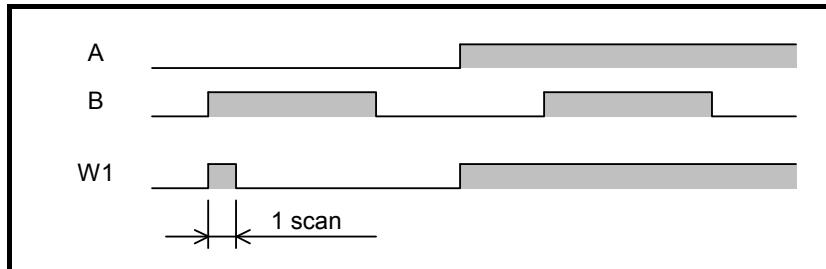


Fig. 4.1.18 (b) Timing chart of ORPT Instruction

NOTE

Please refer to "4.1.16 RDPT Instruction" notes about this instruction.

4.1.19 RDPT.STK Instruction

Positive transition contact instruction. Shifts the stack register content one bit to the left and when rising transition ($0 \rightarrow 1$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.19(a) shows the ladder format and Table 4.1.19(a) shows the mnemonic format.

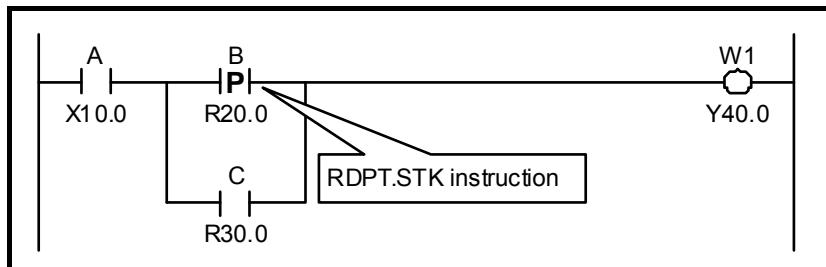


Fig. 4.1.19 (a) Format of RDPT.STK Instruction

Table 4.1.19 Mnemonic of RDPT.STK Instruction

Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	RDPT.STK	R20 .0		B
3	OR	R30 .0		C
4	AND.STK			
5	WRT	Y40 .0		W1 output

Status of operation result

ST2	ST1	ST0
		A
	A	B(PT)
	A	B(PT) + C
		A • (B(PT) + C)
		A • (B(PT) + C)

Operation

Timing chart in the above example is as follows.

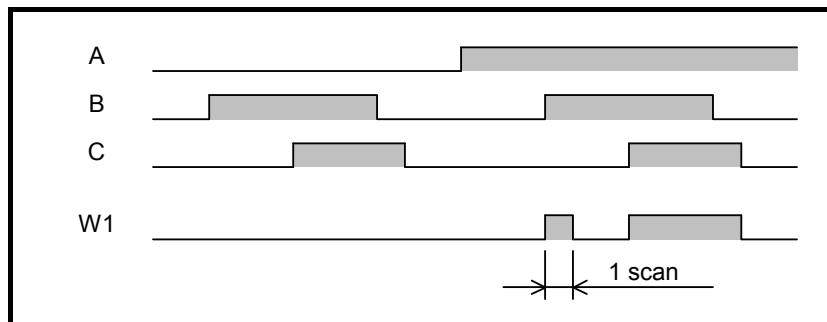


Fig. 4.1.19 (b) Timing chart of RDPT.STK Instruction

NOTE

Please refer to "4.1.16 RDPT Instruction" notes about this instruction.

4.1.20 RDNT Instruction

Negative transition contact instruction. When falling transition ($1 \rightarrow 0$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.20(a) shows the ladder format and Table 4.1.20(a) shows the mnemonic format.

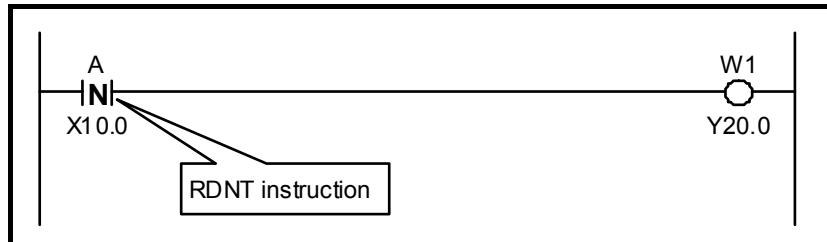


Fig. 4.1.20 (a) Format of RDNT Instruction

Table 4.1.20 Mnemonic of RDNT Instruction

Mnemonic format **Status of operation results**

Step number	Instruction	Address No.	Bit No.	Remarks
1	RDNT	X10 .0		A
2	WRT	Y20 .0		W1 output

Status of operation result		
ST2	ST1	ST0
		A(NT)
		A(NT)

Operation

Timing chart in the above example is as follows.

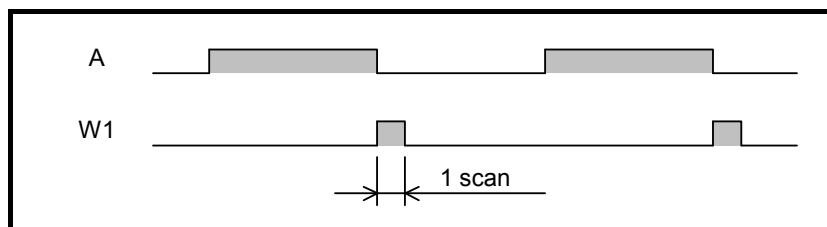


Fig. 4.1.20 (b) Timing chart of RDNT Instruction

CAUTION

- 1 The bit, already turned off when a program is started after program reading by the Input/Output function or Power ON, turns on the output with the scan at the beginning just after starting.
- 2 An output may not be turned on when a bit changes with ON→OFF→ON during 1 scan. Moreover, when validating the result of ladder edit, scan time temporarily becomes larger.
- 3 In ladder edit, when the bit contained in the edited ladder net has already turned off, only the edited contact does not turn on with the scan beginning after the edit.
- 4 When this instruction is skipped by Jump instruction or subroutine call instruction, this instruction is not executed and the output of instruction does not change.
- 5 This instruction uses a work memory internally in order to detect bit transition. The FANUC LADDER-III for Robot searches the work memory automatically in the domain which can be used, and is assigned. Therefore, the program edited in a different procedure becomes mismatching at the comparing, even when the appearance of ladder diagram is the same.

4.1.21 ANDNT Instruction

Negative transition contact instruction. When falling transition ($1 \rightarrow 0$) of the specified signal is detected ST0 bit is not changed. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more points in ladder circuit.

Format

Fig. 4.1.21(a) shows the ladder format and Table 4.1.21(a) shows the mnemonic format.

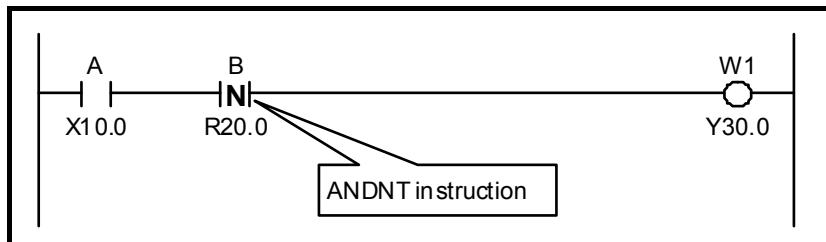


Fig. 4.1.21 (a) Format of ANDNT Instruction

Table 4.1.21 Mnemonic of ANDNT Instruction

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	ANDNT	R20 .0		B
3	WRT	Y30 .0		W1 output

ST2	ST1	ST0
		A
		A • B (NT)
		A • B (NT)

Operation

Timing chart in the above example is as follows.

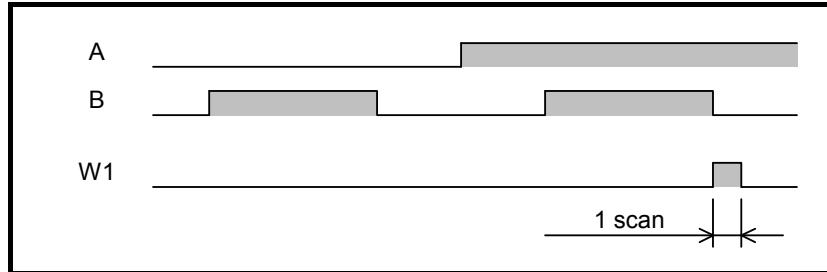


Fig. 4.1.21 (b) Timing chart of ANDNT Instruction

NOTE

Please refer to "4.1.20 RDNT Instruction" notes about this instruction.

4.1.22 ORNT Instruction

Negative transition contact instruction. When falling transition ($1 \rightarrow 0$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise ST0 bit is not changed.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.22(a) shows the ladder format and Table 4.1.22(a) shows the mnemonic format.

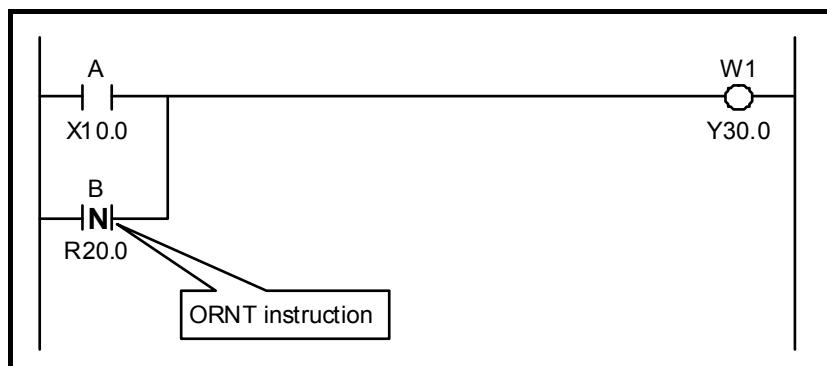


Fig. 4.1.22 (a) Format of ORNT Instruction

Table 4.1.22 Mnemonic of ORNT Instruction

Mnemonic format

Status of operation result

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	ORNT	R20 .0		B
3	WRT	Y30 .0		W1 output

ST2	ST1	ST0
		A
		A + B (NT)
		A + B (NT)

Operation

Timing chart in the above example is as follows.

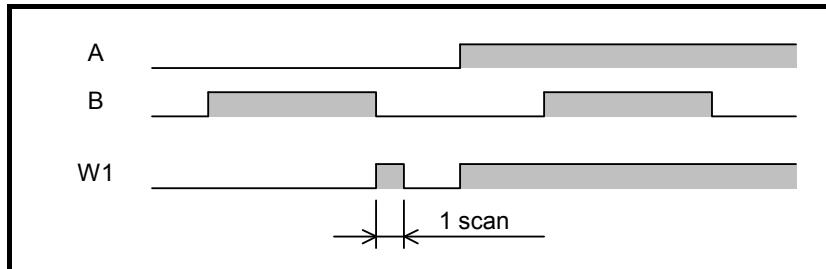


Fig. 4.1.22 (b) Timing chart of ORNT Instruction

NOTE

Please refer to "4.1.20 RDNT Instruction" notes about this instruction.

4.1.23 RDNT.STK Instruction

Negative transition contact instruction. Shifts the stack register content one bit to the left and when falling transition ($1 \rightarrow 0$) of the specified signal is detected, "1" is set to the ST0 bit. Otherwise "0" is set to the ST0 bit.

This instruction can specify the same address in two or more point in ladder circuit.

Format

Fig. 4.1.23(a) shows the ladder format and Table 4.1.23(a) shows the mnemonic format.

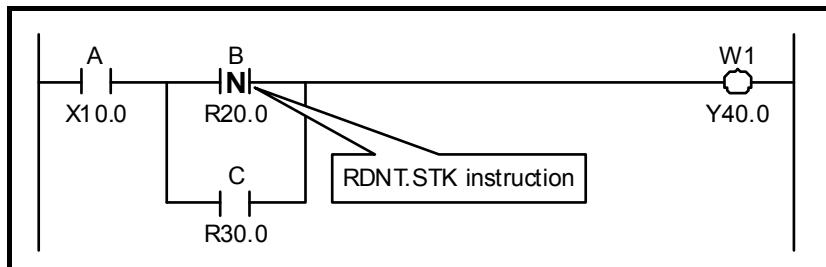


Fig. 4.1.23 (a) Format of RDNT.STK Instruction

Table 4.1.23 Mnemonic of RDNT.STK Instruction

Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10.0		A
2	RDNT.STK	R20.0		B
3	OR	R30.0		C
4	AND.STK			
5	WRT	Y40.0		W1 output

Status of operation result		
ST2	ST1	ST0
		A
	A	B(NT)
	A	B(NT) + C
		A • (B(NT) + C)
		A • (B(NT) + C)

Operation

Timing chart in the above example is as follows.

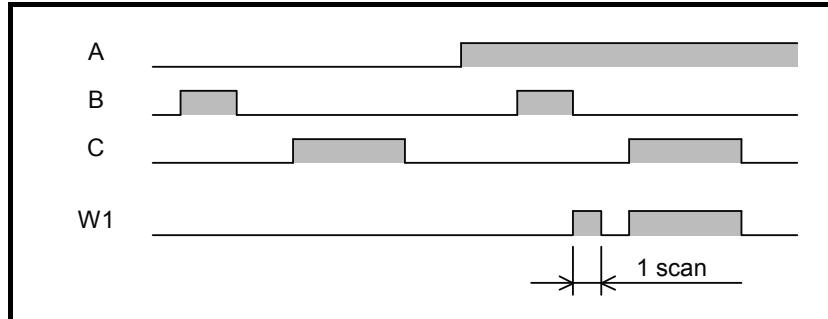


Fig. 4.1.23 (b) Timing chart of RDNT.STK Instruction

NOTE

Please refer to "4.1.20 RDNT Instruction" notes about this instruction.

4.1.24 PUSH Instruction / POP Instruction

Instruction to make a branch of circuit.

A PUSH instruction shifts the stack register one bit to the left. The current operation result (ST0) is not changed.

A POP instruction shifts the stack register one bit to the right

Format

Fig. 4.1.24 shows the ladder format and Table 4.1.24 shows the mnemonic format.

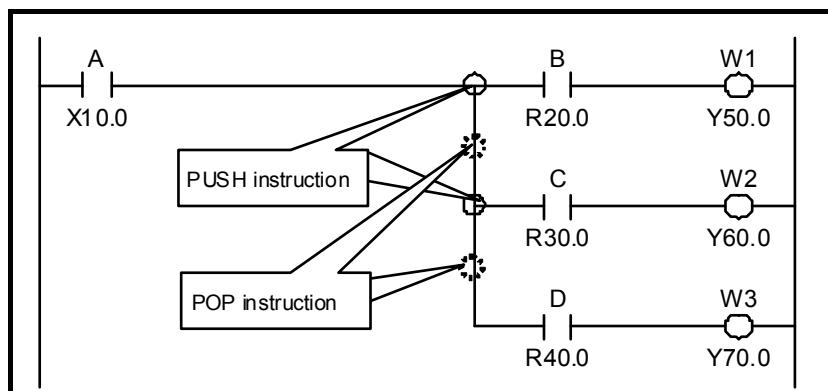


Fig. 4.1.24 Format of PUSH and POP Instructions

Table 4.1.24 Mnemonic of PUSH and POP Instructions

Mnemonic format

Mnemonic Format				
Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	X10 .0		A
2	PUSH			
3	AND	R20 .0		B
4	WRT	Y50 .0		W1 output
5	POP			
6	PUSH			
7	AND	R30 .0		C
8	WRT	Y60 .0		W2 output
9	POP			
10	AND	R40 .0		D
11	WRT	Y70 .0		W3 output

Status of operation result		
ST2	ST1	ST0
		A
	A	A
	A	A • B
	A	A • B
		A
	A	A
	A	A • C
	A	A • C
		A
		A • D
		A • D

Operation

- (1) In the above example, the value of A stored in ST0 is shift to ST1 by PUSH instruction before performing the logical product of A and B. The value of ST0 is not changed.
- (2) After outputting the operation result of the logical product of A and B to W1, the value of A stored in ST1 is shifted to ST0 by POP instruction.
- (3) Before performing the logical product of A and C, the value of A stored in ST0 is shifted to ST1 by PUSH instruction. The value of ST0 is not changed.
- (4) After outputting the operation result of the logical product of A and C to W2, the value of A stored in ST1 is shifted to ST0 by POP instruction.
- (5) The logical product of A and D is performed and an operation result is outputted to W3.

4.2 FUNCTIONAL INSTRUCTIONS

When creating a sequence program, you may find it difficult to code certain types of functions with the basic instructions alone that perform a one-bit logical operation each. One example is a shortcut control function for a rotating part that involves numeric and other complex operations. To facilitate the programming of these functions that are difficult to code with the basic instructions alone, a set of functional instructions are available.

This section describes how to use each functional instruction. For a list of the functional instructions and information about their specifications, see Subsection 2.1.7 or 2.1.8.

4.2.1 Format of the Functional Instructions

Before detailed descriptions of the individual functional instructions are given, this subsection explains the format of the functional instructions and their general specifications. Be sure to read this subsection because it contains important information such as the rules regarding the use of the functional instructions.

(1) Format of the functional instructions

Since the functional instructions cannot be represented using relay symbols, they need to be represented in the format shown in Fig. 4.2.1 (a). The structure of a functional instruction consists of control conditions, an instruction, parameters, an output coil (W1), a functional instruction operation result register (R9000 to R9005 or Z0 to Z5).

4. LADDER LANGUAGE

B-83254EN/02

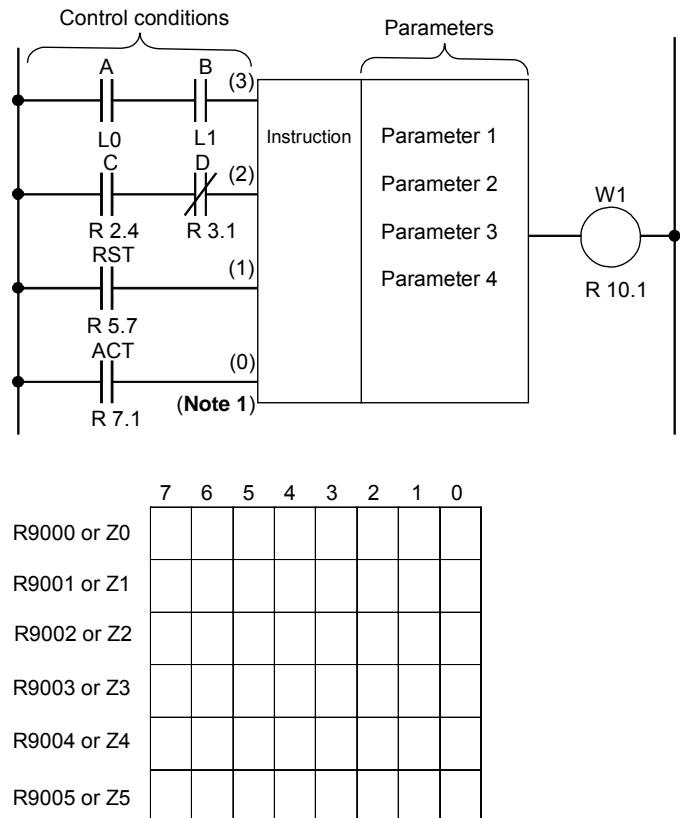


Fig. 4.2.1 (a) Structure of a functional instruction

Table 4.2.1 Coding format of the functional instructions

Step number	Instruction	Address No.	Bit No.	Remarks	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· \bar{D}
5	RD.STK	R5	. 7	RST		A·B	C· \bar{D}	RST
6	RD.STK	R7	. 1	ACT	A·B	C· \bar{D}	RST	ACT
7	SUB	OO		Instruction	A·B	C· \bar{D}	RST	ACT
8	(PRM) (Note 2)	OOOO		Parameter 1	A·B	C· \bar{D}	RST	ACT
9	(PRM)	OOOO		Parameter 2	A·B	C· \bar{D}	RST	ACT
10	(PRM)	OOOO		Parameter 3	A·B	C· \bar{D}	RST	ACT
11	(PRM)	OOOO		Parameter 4	A·B	C· \bar{D}	RST	ACT
12	WRT	R10	. 1	W1 output	A·B	C· \bar{D}	RST	W1

NOTE

- NOTE**

 - 1 The number within each pair of parentheses shown for the control conditions represents the position in the register where the result is to be stored.
 - 2 The term (PRM) in the Instruction fields for step numbers 8 to 11 means a parameter. You do not need to input the term (PRM); just enter an address or numeric data.

(2) Control conditions

The number of control conditions and the meanings of those conditions differ for each functional instruction.

The control conditions are stored in the register, as shown in Table 4.2.1 (a). Once set, therefore, the sequence of the control conditions is fixed. You cannot change the sequence or omit any of the control conditions.

⚠ CAUTION

All functional instructions give precedence to the RST processing when they include RST in their control conditions. Therefore, when RST = 1, the functional instruction carries out the RST processing even if ACT = 0.

(3) Instruction

For the types of functional instructions, see Subsection 2.1.7 or 2.1.8.

To input the instruction with relay symbols, use the soft keys of the programmer.

(4) Parameters

Unlike the basic instructions, the functional instructions deal with numeric values. Therefore, reference data values and addresses storing data may be entered in their parameters. The number of parameters and the meanings of those parameters differ for each functional instruction.

(5) W1

W1 is the destination to which the functional instruction outputs its operation result when that result can be represented by a one-bit value, 0 or 1. The designer can freely decide the address of W1. The meaning of W1 differs for each functional instruction. Some functional instructions do not have the W1 output.

(6) Data to be processed

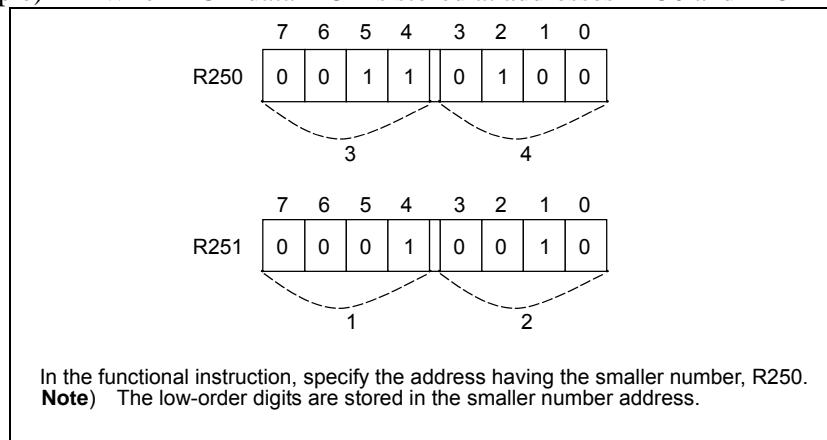
The data processed by the functional instructions is in two formats - binary coded decimal (BCD) format and binary format. Dealing with all numeric data in the binary format is recommended.

(7) Examples of numeric data

(a) BCD format data

Basically, the data processed in the BCD format is handled in units of one byte (0 to 99), two bytes (0 to 9999), or four bytes (0 to 99,999,999; for the DCNVB instruction only). A four-digit BCD data block is stored in two bytes of consecutive addresses, as in the following example.

(Example) When BCD data 1234 is stored at addresses R250 and R251



(b) Binary format data

Basically, the data processed in the binary format is handled in units of one byte (-128 to +127), two bytes (-32,768 to +32,767), or four bytes (-2,147,483,648 to +2,147,483,647). The data is stored at addresses R200, R201, R202, and R203, as shown below. Note that negative numbers are set as two's complements.

<p>One-byte data (-128 to +127)</p> <table border="1" style="margin-bottom: 10px;"> <tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>R200</td><td>\pm</td><td>2^6</td><td>2^5</td><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td><td>2^0</td></tr> </table> <p>Sign { 0: Positive 1: Negative</p>		7	6	5	4	3	2	1	0	R200	\pm	2^6	2^5	2^4	2^3	2^2	2^1	2^0	<p>(Example) One-byte data</p> <table border="1" style="margin-bottom: 10px;"> <tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> </table> <p>(+1)</p> <p>↓</p> <table border="1" style="margin-bottom: 10px;"> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table> <p>(- 1)</p> <p>↓</p> <table border="1" style="margin-bottom: 10px;"> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table> <p>(+127)</p> <p>↓</p> <table border="1" style="margin-bottom: 10px;"> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> </table> <p>(- 127)</p>	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
	7	6	5	4	3	2	1	0																																																			
R200	\pm	2^6	2^5	2^4	2^3	2^2	2^1	2^0																																																			
7	6	5	4	3	2	1	0																																																				
0	0	0	0	0	0	0	1																																																				
1	1	1	1	1	1	1	1																																																				
0	1	1	1	1	1	1	1																																																				
1	0	0	0	0	0	0	1																																																				
<p>Two-byte data (-32,768 to +32,767)</p> <table border="1" style="margin-bottom: 10px;"> <tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>R200</td><td>2^7</td><td>2^6</td><td>2^5</td><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td><td>2^0</td></tr> </table> <table border="1" style="margin-bottom: 10px;"> <tr><td>R201</td><td>\pm</td><td>2^{14}</td><td>2^{13}</td><td>2^{12}</td><td>2^{11}</td><td>2^{10}</td><td>2^9</td><td>2^8</td></tr> </table>		7	6	5	4	3	2	1	0	R200	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	R201	\pm	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8																																
	7	6	5	4	3	2	1	0																																																			
R200	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0																																																			
R201	\pm	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8																																																			
<p>Four-byte data (-2,147,483,648 to +2,147,483,647)</p> <table border="1" style="margin-bottom: 10px;"> <tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>R200</td><td>2^7</td><td>2^6</td><td>2^5</td><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td><td>2^0</td></tr> </table> <table border="1" style="margin-bottom: 10px;"> <tr><td>R201</td><td>2^{15}</td><td>2^{14}</td><td>2^{13}</td><td>2^{12}</td><td>2^{11}</td><td>2^{10}</td><td>2^9</td><td>2^8</td></tr> </table> <table border="1" style="margin-bottom: 10px;"> <tr><td>R202</td><td>2^{23}</td><td>2^{22}</td><td>2^{21}</td><td>2^{20}</td><td>2^{19}</td><td>2^{18}</td><td>2^{17}</td><td>2^{16}</td></tr> </table> <table border="1" style="margin-bottom: 10px;"> <tr><td>R203</td><td>\pm</td><td>2^{30}</td><td>2^{29}</td><td>2^{28}</td><td>2^{27}</td><td>2^{26}</td><td>2^{25}</td><td>2^{24}</td></tr> </table>		7	6	5	4	3	2	1	0	R200	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	R201	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	R202	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	R203	\pm	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}														
	7	6	5	4	3	2	1	0																																																			
R200	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0																																																			
R201	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8																																																			
R202	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}																																																			
R203	\pm	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}																																																			

In the functional instruction, specify the address having the smallest number, R200.

(8) Addresses of numeric data processed by functional instructions

When the numeric data to be processed by a functional instruction consists of two or four bytes, it is recommended to specify an even number or a multiple of four as the address of the numeric data in the relevant parameter of that functional instruction. Specifying an even-numbered or multiple-of-four address causes the functional instruction to execute slightly faster.

In the case of a functional instruction that mainly deals with binary data, such a parameter is marked with an asterisk (*) in the parameter field of the diagram illustrating the format of the functional instruction, as shown below.

An even-numbered or multiple-of-four address means that the letter R is followed by an even number or a multiple of four in the case of an internal relay, or that the letter D is followed by an even number or a multiple of four in the case of a data table.

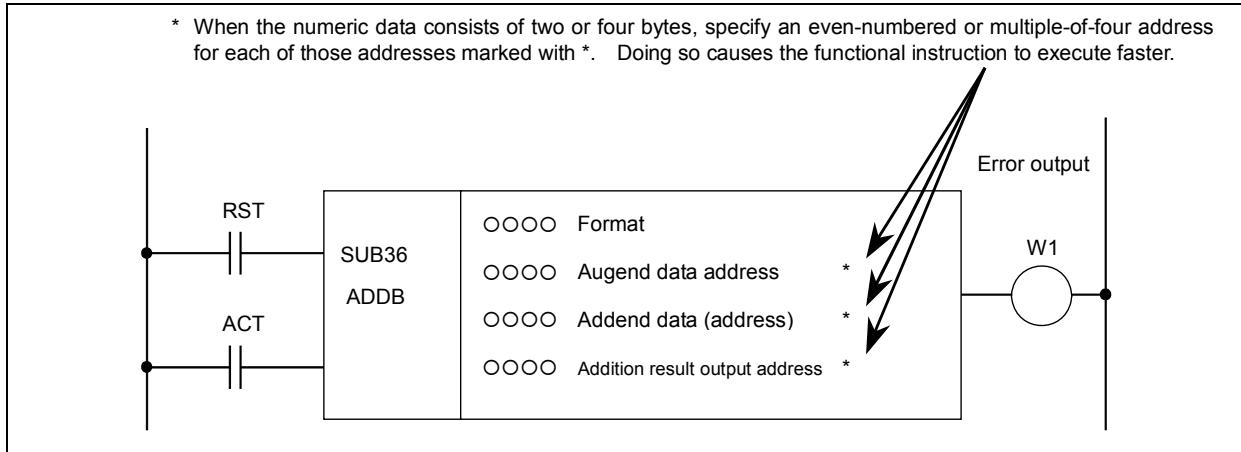


Fig. 4.2.1 (b)

- (9) Functional instruction operation result register
(R9000 to R9005, Z0 to Z5) (See Fig. 4.2.1 (c).)

The results of executing functional instructions are set in this register. The register is shared by all the functional instructions. Therefore, if you do not reference the register immediately after executing the target functional instruction, the operation data of that instruction is erased as a subsequent functional instruction is executed.

Also note that the operation data of this register cannot be exchanged between sequence programs of different levels. For example, when the subtraction instruction (SUBB) is executed in a first level program and the result of its execution is set in the register, a second level program cannot reference the set operation data by reading the register in the R9000 or Z0 range.

The operation data set in this register can be shared by sequence programs of the same level and is maintained until immediately before a functional instruction is executed that sets subsequent operation data in the register. The operation data to be set in this register differs for each functional instruction. The sequence program can read this data but not write to this register.

	7	6	5	4	3	2	1	0
R9000, Z0								
R9001, Z1								
R9002, Z2								
R9003, Z3								
R9004, Z4								
R9005, Z5								

Fig. 4.2.1 (c)

This register consists of six bytes, from R9000 to R9005 or Z0 to Z5. A single block of data can be read from the register in bits or bytes at a time.

To read the data of the first bit of R9000, for example, specify RD R9000.1.

4.3 TIMER

The following types of timer instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	TMR	3	On-delay timer
2	TMRB	24	Fixed on-delay timer
3	TMRBF	77	Fixed off-delay timer
4	TMRC	54	On-delay timer
5	TMRST	221	Stop watch timer (1 ms accuracy)
6	TMRSS	222	Stop watch timer (1 sec accuracy)

4.3.1 TMR (On-delay Timer: SUB 3)

This is an on-delay timer.

Since you set the time in nonvolatile memory (T address) using the timer screen, you can change the set time without changing the ladder diagram.

The timer number you specify in the parameter is the number displayed on the timer screen. The data type in this instruction is binary type.

Format

Fig. 4.3.1 (a) shows the ladder format and Table 4.3.1 shows the mnemonic format.

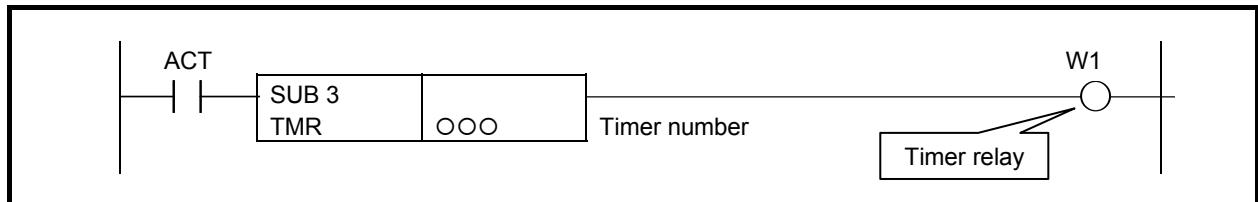


Fig. 4.3.1 (a) Format of TMR instruction

Table 4.3.1 Mnemonic of TMR instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	TMR	000		Timer number				↓
3	WRT	0000 .0		Timer relay output				W1

In the above mnemonic format, instruction name "TMR" at step number 2 can be abbreviated as "T".

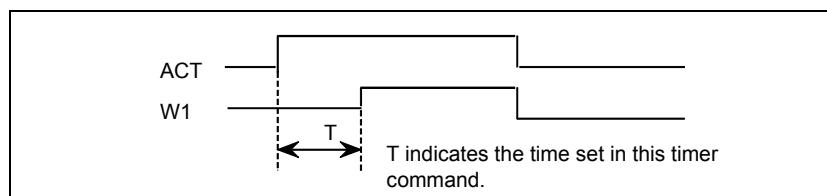


Fig. 4.3.1 (b) Operation of the timer

Control condition

ACT = 0: Turns off W1.

ACT = 1: Starts the timer.

Parameter

Set the timer number.

⚠ WARNING

- 1 If the timer number is duplicated, or falls outside the valid range, the operation will be unpredictable.
- 2 When using the Common PMC Memory mode, don't use a duplicate timer number in multiple PMC paths.

Setting timers

The initial value of the timer setting time can be set in steps of 48 msec for timer numbers 1 to 8 and in steps of 8 msec for timer numbers 9 and later. (For information about the number of timers of each PMC, see the table below.) The setting time value is rounded down to a multiple of the unit time.

For example, if 38 msec is set, the remainder 6 ($38 = 8 \times 4 + 6$) is discarded, and only 32 msec is actually set.

Initial number of the timer setting time	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
48-msec timer number	1 to 8	1 to 8	1 to 8	1 to 8
8-msec timer number	9 to 40	9 to 250	9 to 500	9 to 500

Timer accuracy

The timer screen allows you to set the accuracy of each timer individually. The setting time range and error are as shown below.

Timer type and number	Setting time	Error
48 msec (1 to 8) (initial value)	48 msec to 1572.8 sec	0 to ± 1 st level sweep interval (4/8 msec)
8 msec (9 or larger) (initial value)	8 msec to 262.1 sec	0 to ± 1 st level sweep interval (4/8 msec)
1 msec (1 or larger)	1 msec to 32.7 sec	0 to ± 1 st level sweep interval (4/8 msec)
10 msec (1 or larger)	10 msec to 327.7 sec	0 to ± 1 st level sweep interval (4/8 msec)
100 msec (1 or larger)	100 msec to 54.6 min	0 to ± 1 st level sweep interval (4/8 msec)
1 sec (1 or larger)	1 sec to 546 min	0 to ± 1 st level sweep interval (4/8 msec)
1 min (1 or larger)	1 min to 546 h	0 to ± 1 sec

Error is caused only by operation time of the timer instruction. For example, when a timer instruction is used in the 2nd level sequence part, the variation does not include the delay time (Max. 2nd level sequence one cycle time) until the sequence actuates after the set time is reached.

Timer relay (W1)

When the time preset is reached with ACT = 1, the timer relay turns on. The designer can freely decide the address of W1.

4.3.2 TMRB (Fixed On-delay Timer: SUB 24)

This timer is used as a fixed on-delay timer.

Time present in this fixed timer is written to the memory together with the sequence program, so the time once set cannot be changed unless the whole sequence program is exchanged. The data type in this instruction is binary type.

Format

Fig. 4.3.2 (a) shows the ladder format and Table 4.3.2 shows the mnemonic format.

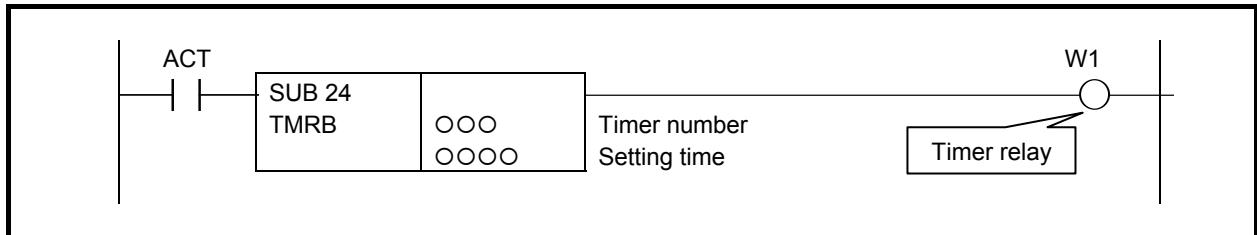


Fig. 4.3.2 (a) Format of TMRB instruction

Table 4.3.2 Mnemonic of TMRB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		24	TMRB instruction				
3	(PRM)		000	Timer number				
4	(PRM)		0000	Setting time				
5	WRT	0000 .0		Timer relay output				W1

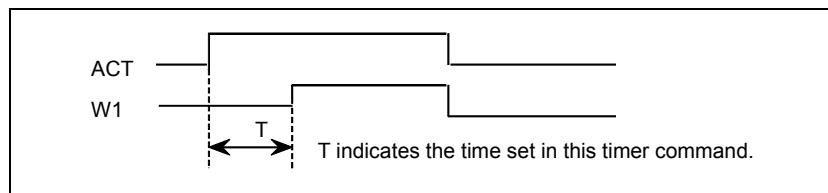


Fig. 4.3.2 (b) Timer operation

Control condition

ACT = 0: Turns off W1.

ACT = 1: Starts the timer.

Parameters

Specify the timer number of a fixed timer. The timer numbers and the setting time range are as shown below.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Timer number	1 to 100	1 to 500	1 to 1000	1 to 1500
Setting time	1 to 32,760,000 (msec)			

⚠ WARNING

If the same timer number is used more than once or if a timer number out of the valid range is used, operation is unpredictable.

The maximum setting time is approximately 546 minutes.

Precision of the timer

Variation in the setting time is between 0 and ± 1 st level execution cycle (4/8 msec). The varying time in this timer is caused only if an error occurred when the timer instruction performs the operation process.

Error caused by sequence program processing time (time of 1 cycle of the second level), etc. are not included.

Timer relay (W1)

The output W1 is turned on after certain time preset in the parameter of this instruction passes after ACT = 1. The designer can freely decide the address of W1.

4.3.3 TMRBF (Fixed Off-delay Timer: SUB 77)

This is the off-delay timer function whose timer preset value is fixed.

The timer preset value is written into the sequence program memory. Therefore, you have to modify sequence program if you want to change the timer value. The data type in this instruction is binary type.

Format

Fig 4.3.3(a) shows the ladder format and Table 4.3.3 shows the mnemonic format.

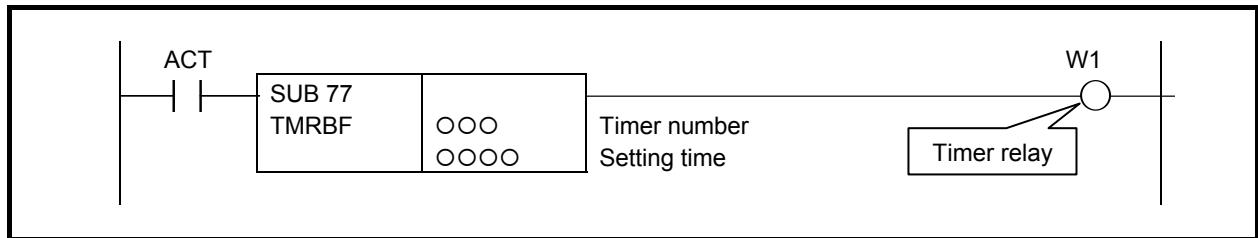


Fig. 4.3.3 (a) Format of TMRBF instruction

Table 4.3.3 Mnemonic of TMRBF instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		77	TMRBF instruction				
3	(PRM)		000	Timer number				
4	(PRM)		0000	Setting time				
5	WRT	0000 .0		Timer relay output				W1

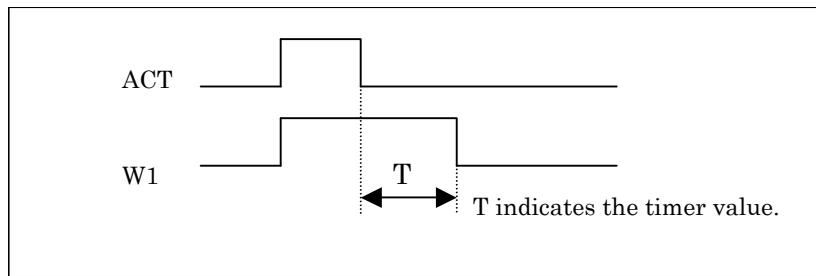


Fig. 4.3.3 (b) Timer operation

Control condition

ACT=0: Starts the timer.

ACT=1: Reset the timer and turn on W1.

Parameters

Specify the timer number of the fixed timer to the 1st parameter. You have to specify the unique timer number for all the TMRB (SUB 24) and TMRBF (SUB 77) instructions.

Specify the timer value of the fixed timer to the 2nd parameter. The unit is millisecond.

The available timer number and timer value is shown below.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Timer number	1 to 100	1 to 500	1 to 1000	1 to 1500
Setting time	1 to 32,760,000 (msec)	1 to 32,760,000 (msec)	1 to 32,760,000 (msec)	1 to 32,760,000 (msec)

 **WARNING**

If the timer number of TMRB or TMRBF is conflicted or if the timer number is out of range, the operation is not guaranteed.

The maximum timer value is approximately 546 minutes.

Precision of the timer

The fluctuations of the time-up time caused by the timer operation are ± 1 st level execution cycle (4/8 msec) plus the delay caused by the execution cycle of the sequence program.

Timer relay (W1)

When the input ACT is turned on, the output W1 will be turned on immediately and the timer instruction will be reset. After that, when the input ACT is turned off, the timer instruction will be started and the output W1 will be turned off after the specified time. If the input ACT is turned on again before the time-up, the timer will be reset.

You can use any valid coil address for the W1.

4.3.4 TMRC (On-delay Timer: SUB 54)

This is the on-delay timer.

A timer setting time is set at an arbitrary address. There is no limit to the number of timers as long as memory areas can be allocated for the timer instruction to use. The data type in this instruction is binary type.

Format

Fig. 4.3.4 (a) shows the ladder format and Table 4.3.4 shows the mnemonic format.

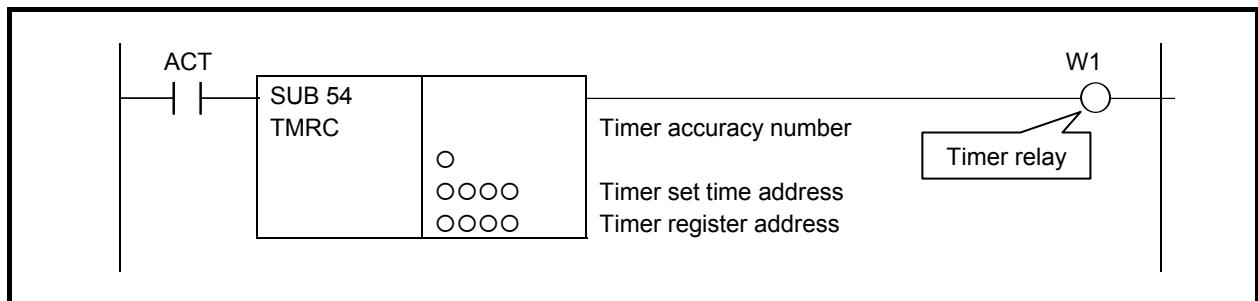
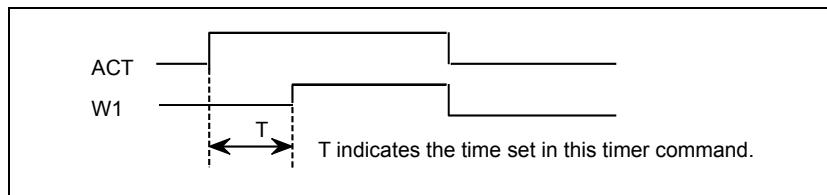


Fig. 4.3.4 (a) Format of TMRC instruction

Table 4.3.4 Mnemonic of TMRC instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB	54		TMRC instruction				
3	(PRM)	000		Timer accuracy number				
4	(PRM)	0000		Timer set time address				
5	(PRM)	0000		Timer register address				
6	WRT	0000 .0		Timer relay output				W1

**Fig. 4.3.4 (b) Timer operation**

Control condition

ACT = 0: Turns off W1.

ACT = 1: Starts the timer.

Parameters

(a) Timer accuracy

The timer accuracy values, setting time range, and error are as shown below.

Timer accuracy	Setting number	The range of setting time (Note)	Margin of error
8 msec	0	8 msec to about 262.1 sec	0 to ± 1 st level sweep interval (4/8 msec)
48 msec	1	48 msec to about 26.2 min	0 to ± 1 st level sweep interval (4/8 msec)
1 sec	2	1 sec to about 546 min	0 to ± 1 st level sweep interval (4/8 msec)
10 sec	3	10 sec to about 91 h	0 to ± 1 st level sweep interval (4/8 msec)
1 min	4	1 min to about 546 h	0 to ± 1 sec
1 msec	5	1 msec to about 32.7 sec	0 to ± 1 st level sweep interval (4/8 msec)
10 msec	6	10 msec to about 327.7 sec	0 to ± 1 st level sweep interval (4/8 msec)
100 msec	7	100 msec to about 54.6 min	0 to ± 1 st level sweep interval (4/8 msec)

Error exclusively refers to that taking place while the timer instruction carries out its operation. It does not include, for example, error that occurs when the timer instruction is used in the 2nd level sequence program, such as the delay from the expiry of the timer until the sequence program initiates processing (time equivalent to one cycle of the 2nd level at worst).

NOTE

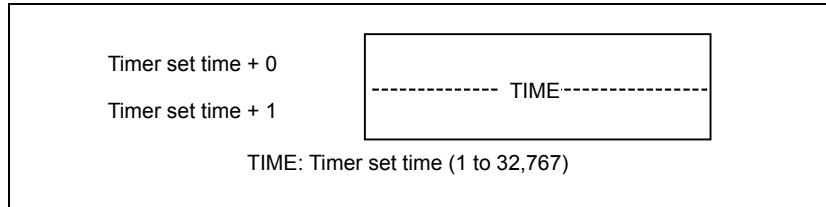
The value range of the setting time is between 0 and 32,767 for all timer accuracies. For example, when the timer accuracy is 8 msec, the value 0 means 8 msec and the value 32,767 means 262,136 msec.

(b) Timer set time address

Sets the first address of the timer set time field.

The continuous 2-byte memory space is required for the timer set time field.

The data table (field D) is normally used as this field.



The timer setting time is converted to the binary format based on the timer accuracy (in units of 8 msec, 48 msec, etc.).

The timer setting time is shown as follows:

8 msec	8 to 262,136 msec
48 msec	48 to 1,572,816 msec
1 sec	1 to 32,767 sec
10 sec	10 to 327,670 sec
1 min	1 to 32,767 min
1 msec	1 to 32,767 msec
10 msec	10 to 327,670 msec
100 msec	100 to 3,276,700 msec

(c) Timer register address

Set the start address of a timer register area.

A timer register area must be allocated to a continuous four-byte memory area starting from the set address. The user area (R area) is used as a timer register area. This area should be used by the PMC system, and therefore should not be used by the sequence program.



Timer relay (W1)

The output W1 is turned on when the time specified in the parameter of this instruction elapses after ACT is set to 1. The designer can freely decide the address of W1.

4.3.5 TMRST (Stop Watch Timer (1ms Accuracy) : SUB 221) TMRSS (Stop Watch Timer (1sec Accuracy) : SUB 222)

This is stop watch timer.

The stop watch timer instruction accumulates periods of time during which ACT=1 is set, and preserves the cumulative value as an integration time. The integration time is not cleared when ACT=0. Instead, when ACT=1 is set again, a continued measurement is made.

In "Setting time", a constant or a PMC memory address for storing data can be specified. An integration time is output to a specified PMC address, so that the integration time can be output to the outside or used for another operation.

When the integration time has reached "Setting time", timer relay W1=1 is set. If ACT=1 even when the integration time has exceeded "Setting time", a measurement is continued until a maximum time is reached. During this period as well, timer relay W1=1 is set.

To reset the integration time and timer relay to 0, set RST (Reset)=1.

As indicated below, two types of the stop watch timer instructions are available according to the timer accuracy.

Table 4.3.5 (a) Kinds of stop watch timer

	Instruction name	SUB No.	Timer accuracy
1	TMRST	221	1 millisecond (ms)
2	TMRSS	222	1 sec

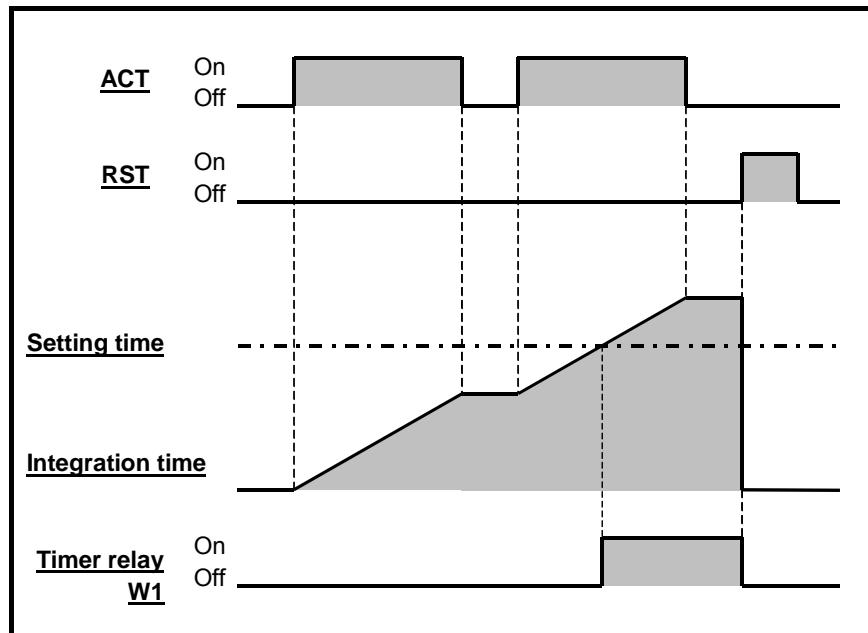


Fig. 4.3.5 (a) Time chart of TMRST and TMRSS Instruction

Format

Fig. 4.3.5 (b) shows the ladder format and Table 4.3.5 (b) shows the mnemonic format.

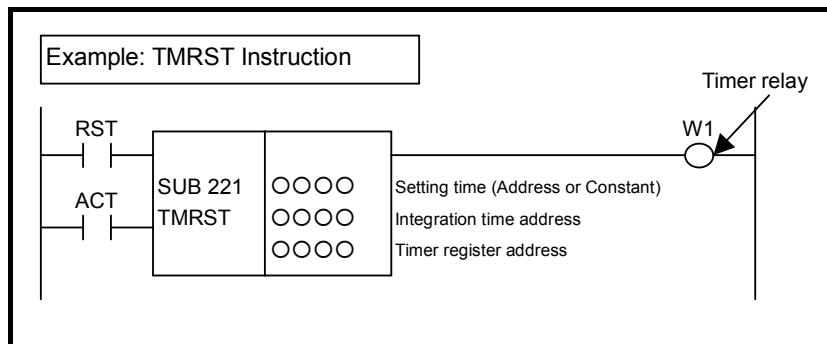


Fig. 4.3.5 (b) Format of TMRST and TMRSS instruction

Table 4.3.5 (b) Mnemonic of TMRST and TMRSS instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	000 .0 0		RST
2	RD.STK	000 .0 0		ACT
3	SUB	221		SUB No. (TMRST Instruction)
4	(PRM)	0000		Setting time (Address or Constant)
5	(PRM)	0000		Integration time address
6	(PRM)	0000		Timer register address
7	WRT	000 .0 0		Timer relay output

Control condition

- (a) Reset (RST)

RST=0 : Reset operation is canceled.

RST=1 : Reset operation is executed.

The integration time is reset to 0.

Even when input signal ACT=1 is set, reset operation has priority, and the stop watch timer is stopped. W1=0 is also set.

- (b) Input signal (ACT)

ACT=0 : Integration is stopped.

ACT=1 : Integration is started.

NOTE

Set RST to 1 only when reset operation is needed. Usually, set RST to 0.

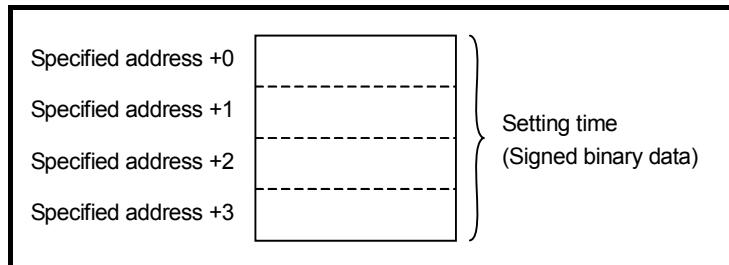
Parameters

- (a) Setting time

Specify a time-out period for the timer. A value from 0 to 2147483647 may be specified. If a value out of this range is specified, integration operation is performed but timer relay W1=0 is set at all times.

In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified, specify "Setting time" as signed binary data by using the contiguous four bytes of memory starting from the specified address.



Instruction name	Setting time
TMRST	0 to 2147483647 millisecond
TMRSS	0 to 2147483647 second

NOTE

When a Setting time is rewritten during execution of instruction, the result is reflected immediately.

(b) Integration time address

Specify a PMC memory address for storing the integration time of the timer. One integration time count corresponds to the timer accuracy.

An integration time address must be allocated to a continuous four-byte memory area starting from the set address.

To preserve the integration time when the power to the Robot is turned on/off, the D area is usually used. However, the delay corresponding to the cycle of backup of D area may cause an error at the Integration time after turning on a power supply again when a power supply is turned off in integration.

The figure below shows the relationships of the actual accumulation of integration time, ladder execution cycle, ACT On/Off operation, and timer relay output.

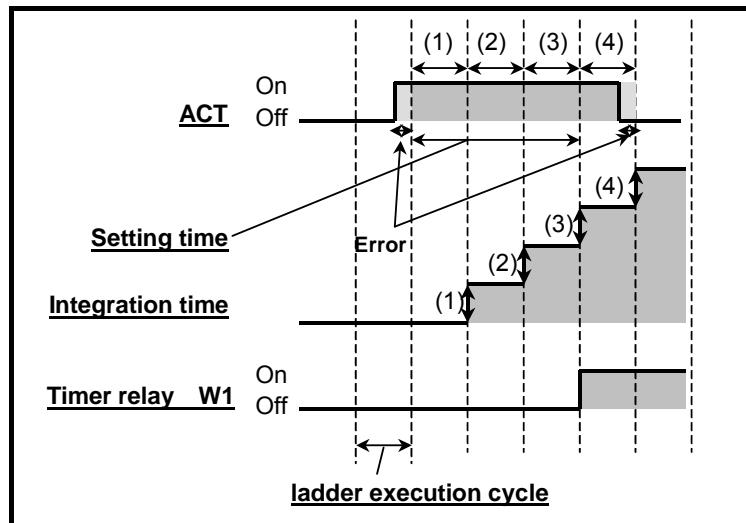


Fig. 4.3.5 (c) Increment of Integration time

Integration time accumulation starts in the execution cycle immediately after ACT=1 (On) is set, and continues until an execution cycle where ACT=0 (Off) is set. Timer relay W1=1 (On) is set when the integration time has reached "Setting time".

A maximum error per measurement section (pair of ACT On/Off) is " \pm ladder execution cycle time".

NOTE

Please do not perform rewriting of integration time during execution of instruction.

(c) Timer register address (work memory)

Specify the address of a 2-byte PMC memory area to be used for integration time calculation. The sequence program should not use this area. Usually, the R area is used.

Timer relay (W1)

W1 is turned on when the integration time has reached the set time.

NOTE

W1 must not be omitted.

4.4 COUNTER

The following types of counter instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	CTR	5	Counter processing
2	CTRB	56	Fixed counter processing
3	CTRC	55	Counter processing
4	CTRD	223	Counter processing (4-byte length)

4.4.1 CTR (Counter: SUB 5)

CTR is used as a counter. Counters are used for various purposes.

Numerical data such as preset values and count values can be used with either BCD format or binary format by a system parameter of PMC.

⚠ WARNING

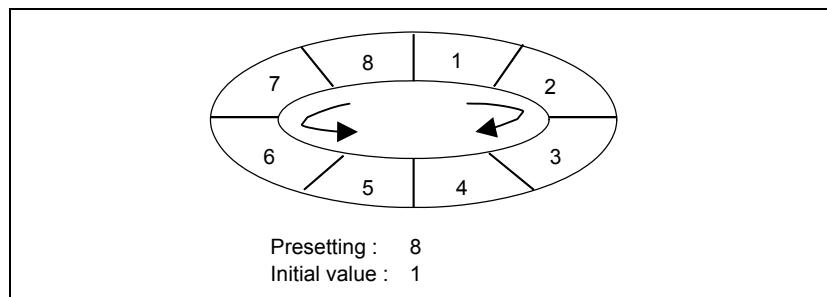
When an incorrect BCD data was set to a BCD type counter, the movement of CTR cannot be sure.

If changing the counter type, be sure to reconfigure the preset value and count value.

This counter has the following functions to meet various applications.

- (a) Preset counter
Outputs a signal when the preset count is reached. The number can be preset from the counter screen, or set in the sequence program.
- (b) Ring counter
Upon reaching the preset count, returns to the initial value by issuing another count signal.
- (c) Up/down counter
The count can be either up or down.
- (d) Selection of initial value
Selects the initial value as either 0 or 1.

A combination of the preceding functions results in the ring counter below.



Such a counter permits the position of a rotor to be memorized.

Format

Fig. 4.4.1 (a) shows the ladder format and Table 4.4.1 shows the mnemonic format.

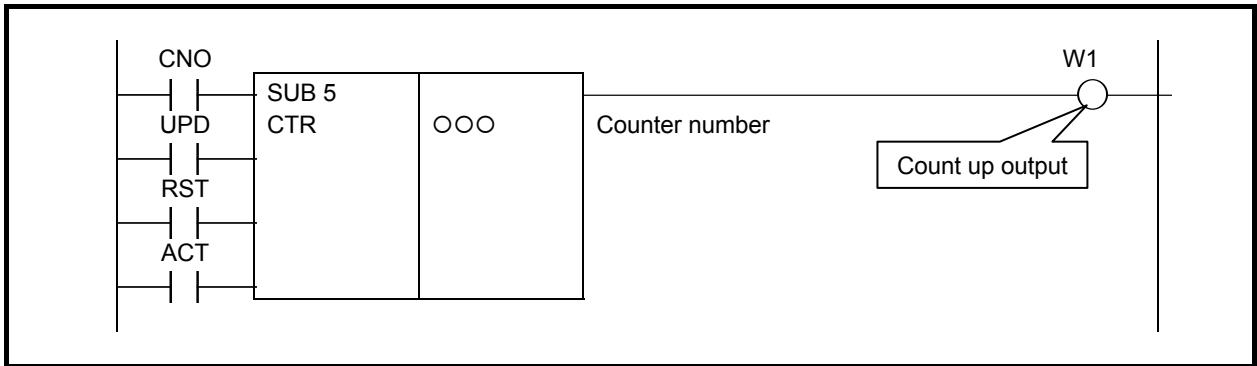


Fig. 4.4.1 (a) Format of CTR instruction

Table 4.4.1 Mnemonic of CTR instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		CNO				CNO
2	RD.STK	0000 .O		UPD				UPD
3	RD.STK	0000 .O		RST				RST
4	RD.STK	0000 .O		ACT				ACT
5	SUB	5		CTR instruction				
6	(PRM)	000		Counter number				
7	WRT	0000 .O		Counter output				W1

Control conditions

- (a) Specify the initial value. (CNO)

CNO = 0: Begins the value of the counter with 0.

0, 1, 2, 3,, n.

CNO = 1: Begins the value of the counter with 1 (0 is not used).

1, 2, 3,, n.

- (b) Specify up or down counter. (UPDOWNM)

UPD = 0: Up counter. The counter begins with 0 when CNO = 0; 1 when CNO = 1.

UPD = 1: Down counter. The counter begins with the preset value.

- (c) Reset (RST)

RST = 0: Releases reset.

RST = 1: Enables reset.

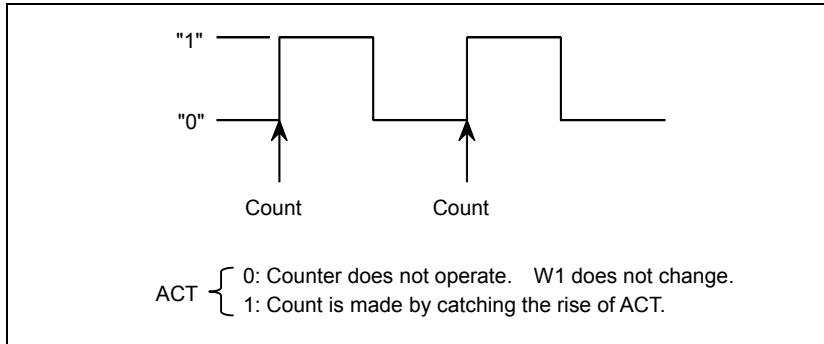
W1 becomes 0.

The integrated value is reset to the initial value.

⚠ CAUTION

Set RST to 1, only when reset is required.

- (d) Count signal (ACT)



Parameter

(a) Counter number

The numbers that can be used are shown below.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Counter number	1 to 20	1 to 100	1 to 200	1 to 300

The preset value and cumulative value that can be set are as follows:

Binary counter: 0 to 32,767

BCD counter: 0 to 9,999

⚠ WARNING

- If the counter number is duplicated, or falls outside the valid range, the operation will be unpredictable.
- When using the Common PMC Memory mode, don't use a duplicate counter number in multiple PMC paths.

Countup output (W1)

In case of up counter mode(UPD=0), when the count is up to a preset value, W1 = 1.

In case of down counter mode(UPD=1) and initial value 0(CNO=0), when the counter reaches 0, W1 is set to 1.

In case of down counter mode (UPD=1) and initial value 1(CNO=1), when the counter reaches 1, W1 is set to 1.

The address of W1 can be determined arbitrarily.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Examples of using the counter

[Example 1]

As a preset counter (See Fig. 4.4.1 (b).)

The number of work pieces to be machined is counted. When the number reaches the preset count, a signal is output.

- L1 is a circuit to make logic 1.
- Since the count ranges from 0 to 9,999, contact B of L1 is used for making CNO = 0.
- Since it is to be an up counter, contact B of L1 is used to make UPD = 0.
- The reset signal of the counter uses input signal CRST.M from the machine tool.

- The count signal is M30X, which was decoded from the NC output M code. M30X contains contact B of CUP to prevent counting past the preset value, as long as reset is not enabled after countup.

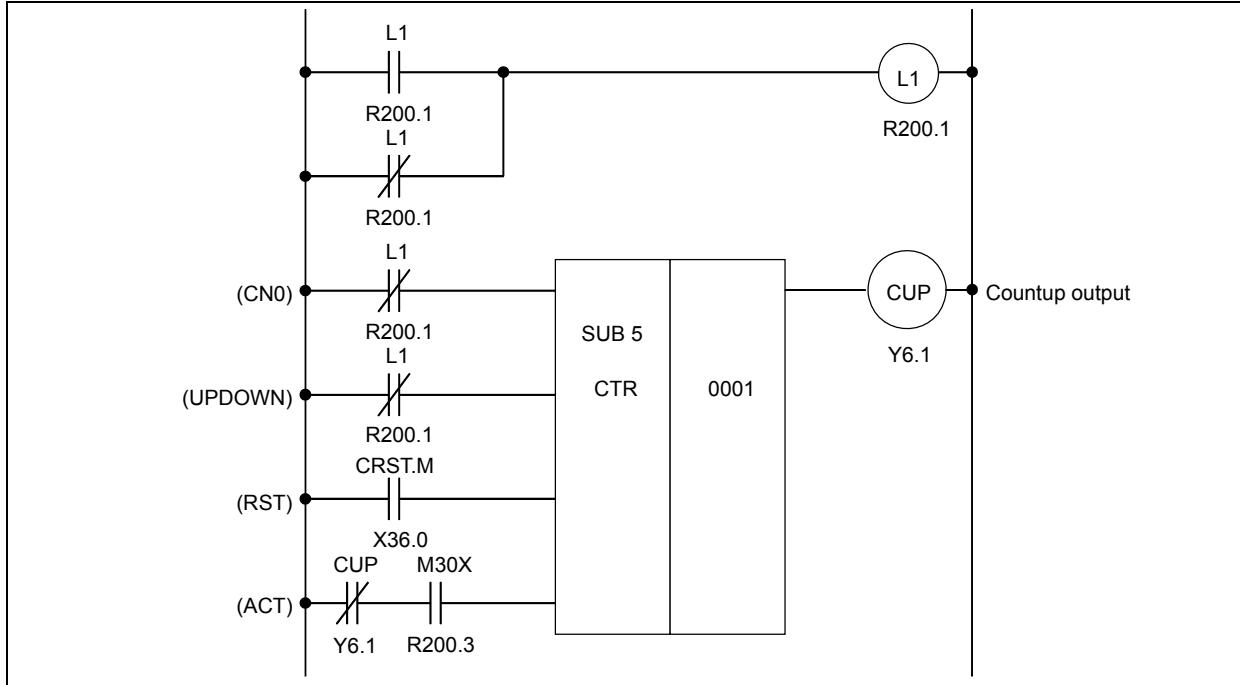


Fig. 4.4.1 (b) Ladder diagram for the counter, example 1

[Example 2]

Use of the counter to store the position of a rotor. (See Fig. 4.4.1 (c).)

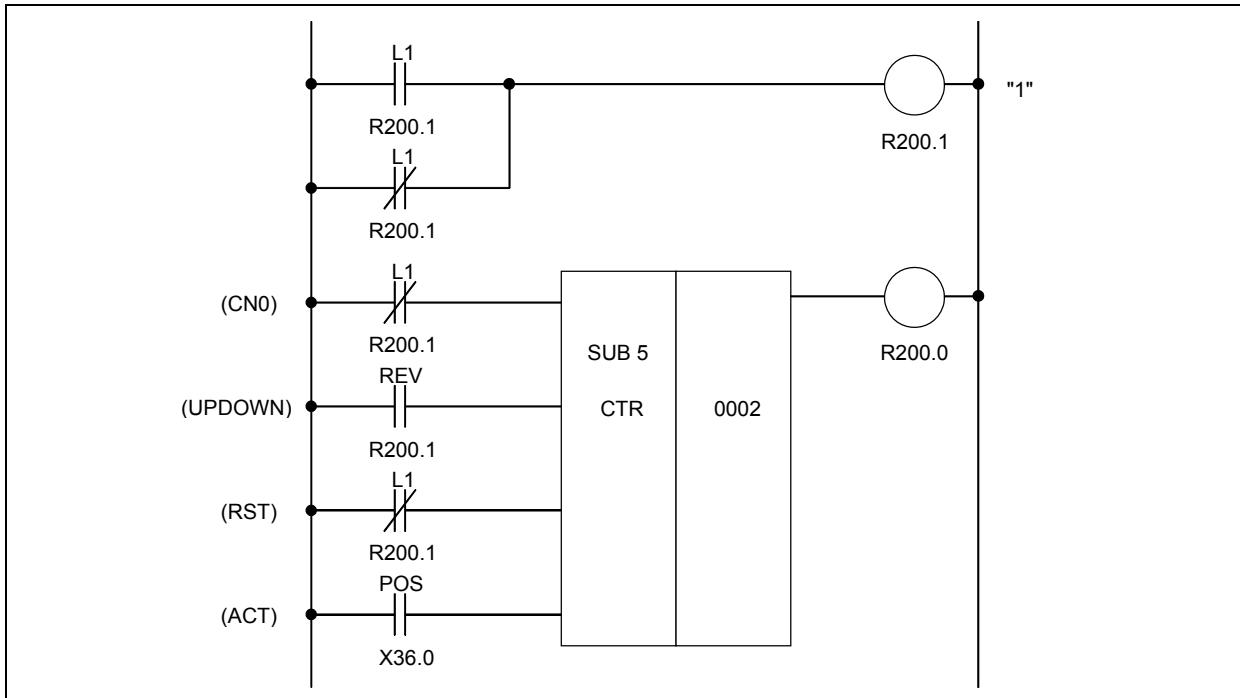


Fig. 4.4.1 (c) Ladder diagram for the counter, example 2

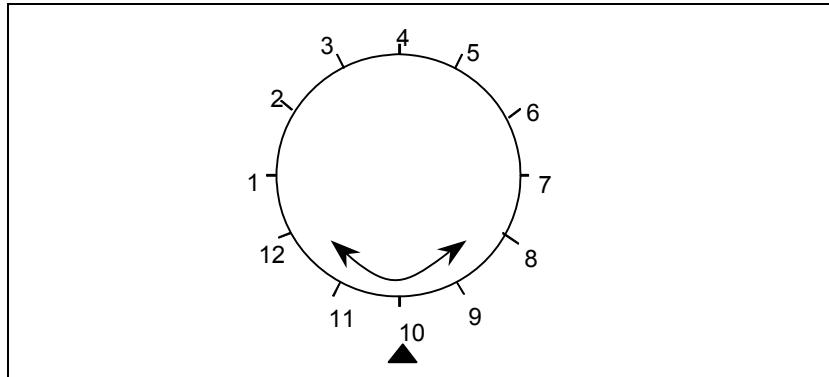


Fig. 4.4.1 (d) Indexing for a rotor

Fig. 4.4.1 (c) shows a ladder diagram for a counter to store the position of a rotor of Fig. 4.4.1 (d).

- (1) Control conditions
 - (a) Count start number
When a 12-angle rotor shown in Fig. 4.4.1 (d) is used, the count starting number is 1. Contact A of L1 is used for making CNO = 1.
 - (b) Specify up and down
The signal REV changes according to the then direction of rotation. It becomes 0 for forward rotation and 1 for reverse rotation. Thus, the counter is an up counter for forward rotation and a down counter for reverse rotation.
 - (c) Reset
In this example, since W1 is not used, RST = 0, and contact B of L1 is used.
 - (d) Count signal
The count signal POS turns on and off 12 times each time the rotor rotates once.
- (2) Counter number and W1
In this example, the second counter is used. The result of W1 is not used, but its address must be determined.
- (3) Operation
 - (a) Setting the preset value
Since the rotor to be controlled is 12-angle as shown in Fig. 4.4.1 (d), 12 must be preset in the counter. It is set from the counter screen.
 - (b) Setting the current value
When the power is turned on, the position of the rotor must be equated with the count on the counter. The count is set via the counter screen. Once a current value is set, then correct current positions will be loaded to the counter every time.
 - (c) The POS signal turns on and off each time the rotor rotates.
The number of times of the POS signal turns on and off is counted by the counter 2, as below.
1, 2, 3, . . . 11, 12, 1, 2, . . .
for forward rotation
1, 12, 11, . . . 3, 2, 1, 12 . . .
for reverse rotation

4.4.2 CTRB (Fixed Counter: SUB 56)

CTRБ is used as a counter. Numerical data such as preset values and count values can be used with binary format. This counter has the following functions to meet various applications.

- (a) Preset counter
Preset the count value. If the count reaches this preset value, the W1 output turns on.
- (b) Ring counter
This is the ring counter which is reset to the initial value when the count signal is input after the count reaches the preset value.
- (c) Up/down counter

This is the reversible counter to be used as both up counter and down counter.

- (d) Selection of initial value
Either 0 or 1 can be selected as the initial value.

Format

Fig. 4.4.2 shows the ladder format and Table 4.4.2 shows the mnemonic format.

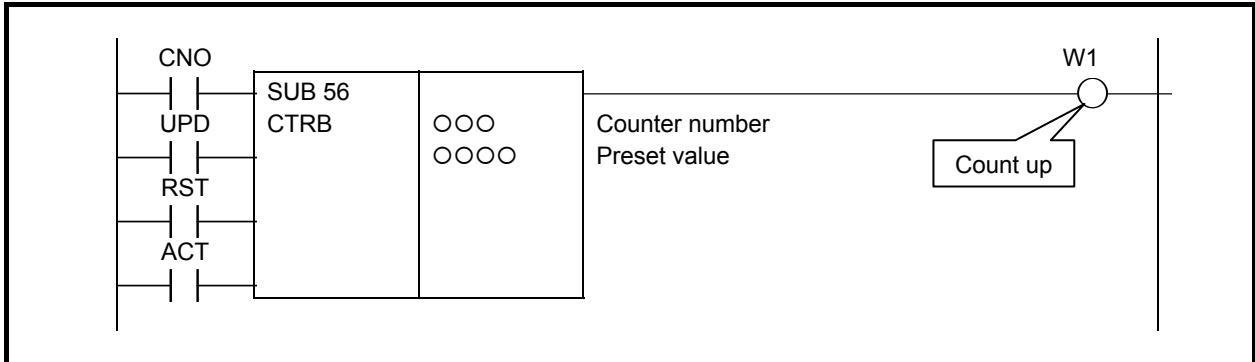


Fig. 4.4.2 Format of CTRB instruction

Table 4.4.2 Mnemonic of CTRB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		CNO				CNO
2	RD.STK	0000 .O		UPD				UPD
3	RD.STK	0000 .O		RST		CNO	UPD	RST
4	RD.STK	0000 .O		ACT	CNO	UPD	RST	ACT
5	SUB	56		CTRB instruction				
6	(PRM)	000		Counter number				
7	(PRM)	0000		Preset value				
8	WRT	0000 .O		Count up output				W1

Control conditions

- (a) Specifying the initial value (CNO)
 - CNO = 0: The counter value starts with "0". 0,1,2,3,.....,n
 - CNO = 1: The counter value starts with "1". 1,2,3,.....,n
- (b) Specifying up or down (UPD)
 - UPD = 0: Up counter
The initial value is "0" when CNO = 0 or "1" when CNO = 1.
 - UPD = 1: Down counter
The initial value is the preset value.
- (c) Reset (RST)
 - RST = 0: Cancels reset.
 - RST = 1: Resets. W1 is reset to 0. The accumulated value is reset to the initial value.
- (d) Count signal (ACT)
 - ACT = 0: The counter does not operate. W1 does not change.
 - ACT = 1: The counter operates at the rise of this signal.

Parameters

- (a) Counter number
The numbers that can be used are shown below.

	1st to 5th path PMC			
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D
Counter number	1 to 20	1 to 100	1 to 200	1 to 300

(b) Preset value

Following value can be set as preset value.

Binary counter: 0 to 32,767

* CTRB is always binary counter. System parameter is ineffective.

⚠ WARNING

- 1 If the counter number is duplicated, or falls outside the valid range, the operation will be unpredictable.
- 2 When using the Common PMC Memory mode, don't use a duplicate counter number in multiple PMC paths.

Countup output (W1)

In case of the up counter mode (UPD=0), when the counter value reaches the preset value, W1 is set to 1. In case of the down counter mode (UPD=1) and initial value 0(CNO=0), when the counter value reaches 0, W1 is set to 1.

In case of the down counter mode (UPD=1) and initial value 1(CNO=1), when the counter value reaches 1, W1 is set to 1.

The W1 address can be specified arbitrarily.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Accumulate value

The address C5000s are used for accumulate value of the CTRB. Each CTRB consumes 2 bytes.

CTRB of counter number 1 uses C5000-5001 and CTRB of number 2 uses C5002-5003 for their accumulate values.

4.4.3 CTRC (Counter: SUB 55)

The numeral data of this counter are all binary. This counter has the following functions and can be used according to the application:

(a) Preset counter

Preset the count value. If the count reaches this preset value, the W1 output turns on.

(b) Ring counter

This is the ring counter which is reset to the initial value when the count signal is input after the count reaches the preset value.

(c) Up/down counter

This is the reversible counter to be used as both the up counter and down counter.

(d) Selection of the initial value

Either 0 or 1 can be selected as the initial value.

Format

Fig. 4.4.3 shows the ladder format and Table 4.4.3 shows the mnemonic format.

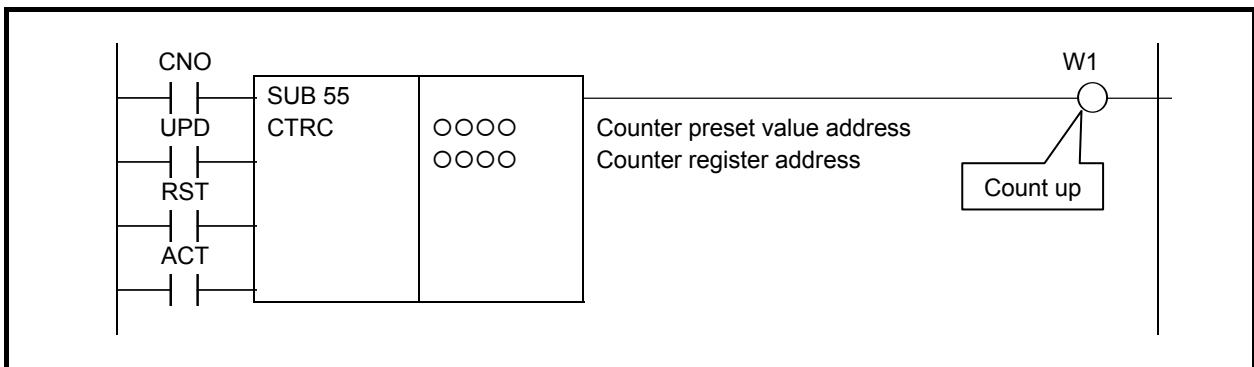


Fig. 4.4.3 Format of CTRC instruction

Table 4.4.3 Mnemonic of CTRC instruction

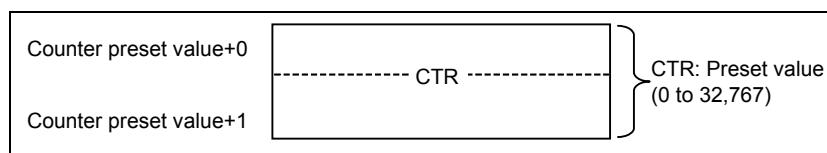
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		CNO				CNO
2	RD.STK	0000 .O		UPD				UPD
3	RD.STK	0000 .O		RST				RST
4	RD.STK	0000 .O		ACT				ACT
5	SUB	55		CTRC instruction				
6	(PRM)	0000		Counter preset value address				
7	(PRM)	0000		Counter register address				
8	WRT	0000 .O		Count up output				

Control conditions

- (a) Specifying the initial value (CNO)
 - CNO = 0: The count value starts with "0". 0, 1, 2, 3, ... n
 - CNO = 1: The count value starts with "1". 1, 2, 3, ... n
- (b) Specifying up or down count (UPD)
 - UPD = 0: Up counter. The initial value is "0" when CNO = 0 or "1" when CNO = 1.
 - UPD = 1: Down counter. The initial value is the preset value.
- (c) Reset (RST)
 - RST = 0: Reset cancelled.
 - RST = 1: Reset. W1 is reset to "0". The accumulated value is reset to the initial value.
- (d) Count signal (ACT)
 - ACT = 0: The counter does not operate. W1 does not change.
 - ACT = 1: The counter operates at the rise of this signal.

Parameters

- (a) Counter preset value address
 - The first address of the counter preset value field is set.
 - The continuous 2-byte memory space from the first address is required for this field. Field D is normally used.

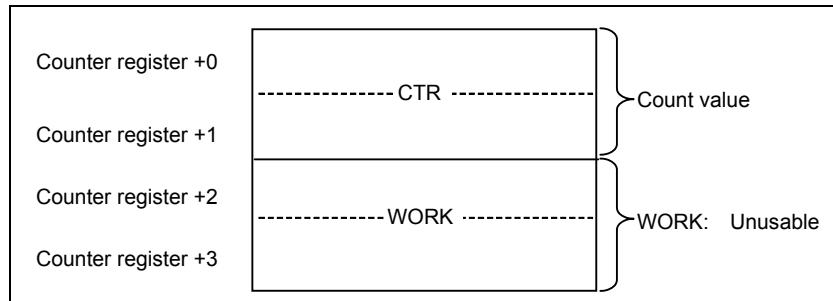


The counter preset value is binary. Therefore, it ranges from 0 to 32,767.

(b) Counter register address

The first address of the counter register field is set.

The continuous 4-byte memory space from the first address is required for this field. Field D is normally used.



CAUTION

When R address is specified as the counter register address, the counter starts with count value "0" at power on.

Countup output (W1)

In case of the up counter mode (UPD=0), when the counter value reaches the preset value, W1 is set to 1. In case of the down counter mode (UPD=1) and initial value 0(CNO=0), when the counter value reaches 0, W1 is set to 1.

In case of the down counter mode (UPD=1) and initial value 1(CNO=1), when the counter value reaches 1, W1 is set to 1.

The W1 address can be specified arbitrarily.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.4.4 CTRD (Counter (4 Bytes Length) : SUB 223)

This instruction is a counter of 4 bytes length binary data. This counter has the following functions and can be used according to the application:

(a) Preset counter

Preset the count. If the count reaches this preset value, the W1 output turns on.

(b) Ring counter

This is the ring counter which is reset to the initial value when the count signal is input after the count reaches the preset value.

(c) Up/down counter

This is the reversible counter to be used as both the up counter and down counter.

(d) Selection of the initial value

Either 0 or 1 can be selected as the initial value.

Format

Fig. 4.4.4 shows the ladder format and Table 4.4.4 shows the mnemonic format.

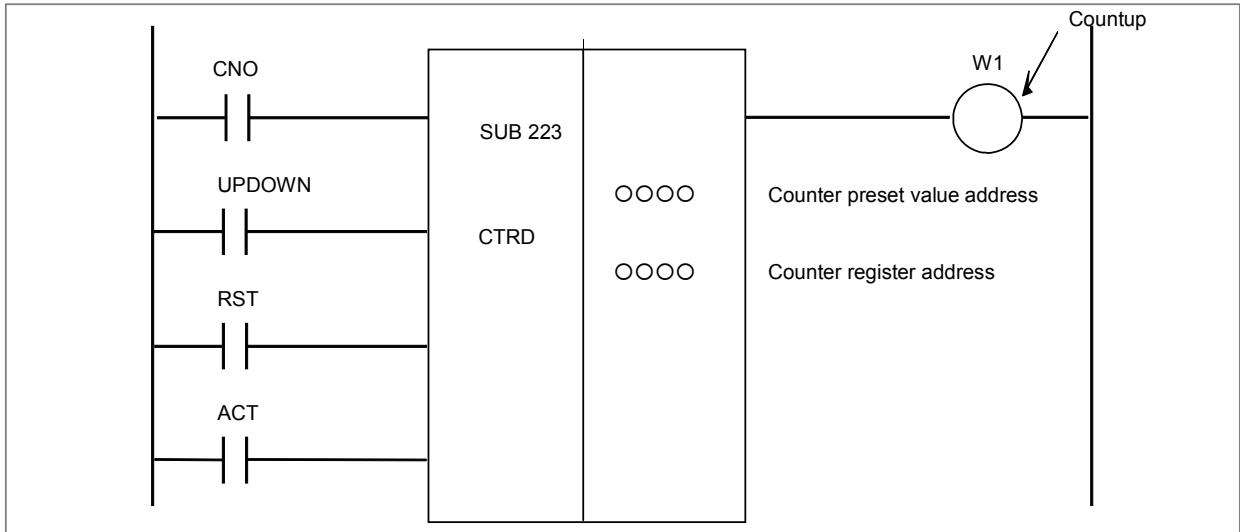


Fig. 4.4.4 Format of CTRD instruction

Table 4.4.4 Mnemonic of CTRD instruction

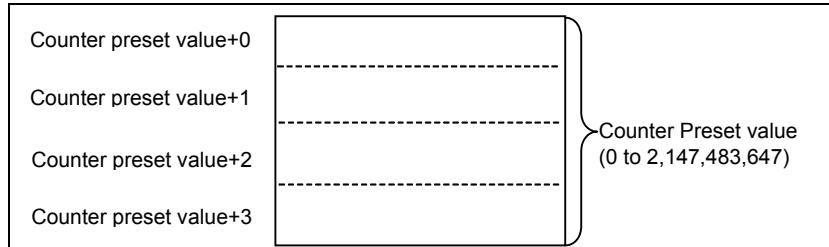
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		CNO				CNO
2	RD. STK	0000 .0		UPDOWN			CNO	UPDOWN
3	RD. STK	0000 .0		RST		CNO	UPDOWN	RST
4	RD. STK	0000 .0		ACT	CNO	UPDOWN	RST	ACT
5	SUB	223		CTRDI instruction				
6	(PRM)	0000		Counter preset value address				
7	(PRM)	0000		Counter register address				
8	WRT	0000 .0		Count up output				W1

Control conditions

- (a) Specifying the initial value (CNO)
CNO = 0: The count value starts with "0". 0, 1, 2, 3, . . . n
CNO = 1: The count value starts with "1". 1, 2, 3, . . . n
 - (b) Specifying up or down count (UPDOWN)
UPDOWN = 0: Up counter. The initial value is "0" when CNO = 0 or "1" when CNO = 1.
UPDOWN = 1: Down counter. The initial value is the preset value.
 - (c) Reset (RST)
RST = 0: Reset cancelled.
RST = 1: Reset. W1 is reset to "0". The accumulated value is reset to the initial value.
 - (d) Count signal (ACT)
ACT = 0: The counter does not operate. W1 does not change.
ACT = 1: The counter operates at the rise of this signal.

Parameters

- (a) Counter preset value address
The first address of the counter preset value field is set.
The continuous 4-byte memory space from the first address is required for this field. Address D is normally used.

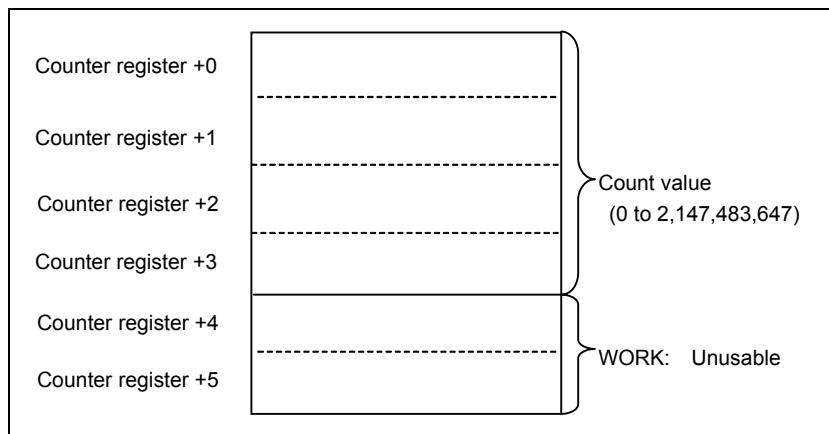


The counter preset value is binary. Therefore, it ranges from 0 to 2,147,483,647.

(b) Counter register address

The first address of the counter register field is set.

The continuous 6-byte memory space from the first address is required for this field. Address D is normally used.



CAUTION

When R address is specified as the counter register address, the counter starts with count value "0" at power on.

Countup output (W1)

In case of the up counter mode (UPDOWN=0), when the counter value reaches the preset value, W1 is set to 1.

In case of the down counter mode (UPDOWN=1) and initial value 0(CNO=0), when the counter value reaches 0, W1 is set to 1.

In case of the down counter mode (UPDOWN=1) and initial value 1(CNO=1), when the counter value reaches 1, W1 is set to 1.

The W1 address can be specified arbitrarily.

NOTE

W1 is not ommissible.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.5 DATA TRANSFER

The following types of data transfer instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	MOVB	43	Transfer of 1 byte
2	MOVW	44	Transfer of 2 bytes
3	MOVD	47	Transfer of 4 bytes
4	MOVN	45	Transfer of an arbitrary number of bytes
5	MOVE	8	Logical product transfer
6	MOVOR	28	Data transfer after logical sum
7	XMOVB	35	Binary index modifier data transfer
8	XMOV	18	Indexed data transfer
9	MOVBT	224	Bit transfer
10	SETNB	225	Data setting (1 byte length)
11	SETNW	226	Data setting (2 bytes length)
12	SETND	227	Data setting (4 bytes length)
13	XCHGB	228	Data exchange (1 byte length)
14	XCHGW	229	Data exchange (2 bytes length)
15	XCHGD	230	Data exchange (4 bytes length)
16	SWAPW	231	Data swap (2 bytes length)
17	SWAPD	232	Data swap (4 bytes length)
18	DSCHB	34	Binary data search
19	DSCH	17	Data search

4.5.1 MOVB (Transfer of 1 Byte: SUB 43)

The MOVB instruction transfers 1-byte data from a specified source address to a specified destination address.

Format

Fig. 4.5.1 shows the ladder format and Table 4.5.1 shows the mnemonic format.

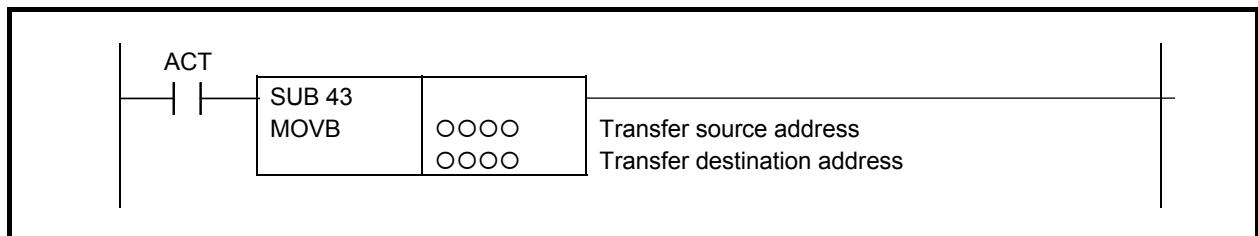


Fig. 4.5.1 Format of MOVB instruction

Table 4.5.1 Mnemonic of MOVB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		43	MOVB instruction				
3	(PRM)	0000		Transfer source address				
4	(PRM)	0000		Transfer destination address				

Control condition

- (a) Execution specification
ACT = 0: No data is transferred.
ACT = 1: One-byte data is transferred.

- (a) Transfer source address
Specify the source address for the transfer.
 - (b) Transfer destination address
Specify the destination address for the transfer

Format

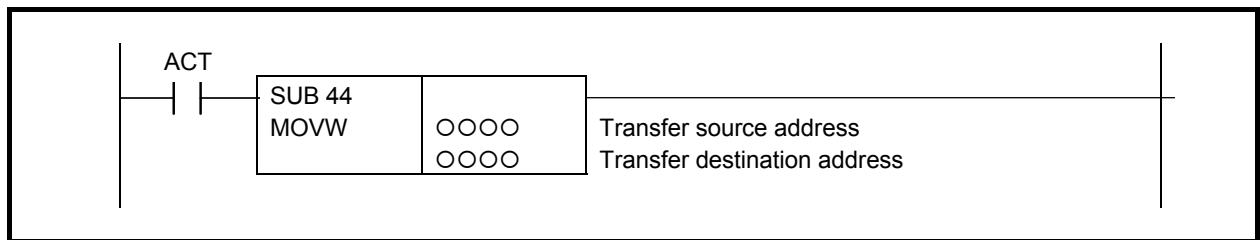


Fig. 4.5.2 Format of MOVW instruction

Table 4.5.2 Mnemonic of MOVW instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		44	MOVW instruction				
3	(PRM)	0000		Transfer source address				
4	(PRM)	0000		Transfer destination address				▼

Control condition

- (a) Execution specification
 $\text{ACT} = 0$: No data is transferred.
 $\text{ACT} = 1$: Two-byte data is transferred.

Parameters

- (a) Transfer source address
Specify the source address for the transfer.
 - (b) Transfer destination address
Specify the destination address for the transfer.

NOTE

Take care not to specify overlapped areas for source and destination. If the source and destination areas are overlapped with each other, the result is not guaranteed.

4.5.3 MOVD (Transfer of 4 Bytes: SUB 47)

The MOVD instruction transfers 4-byte data from a specified source address to a specified destination address.

Format

Fig. 4.5.3 shows the ladder format and Table 4.5.3 shows the mnemonic format.

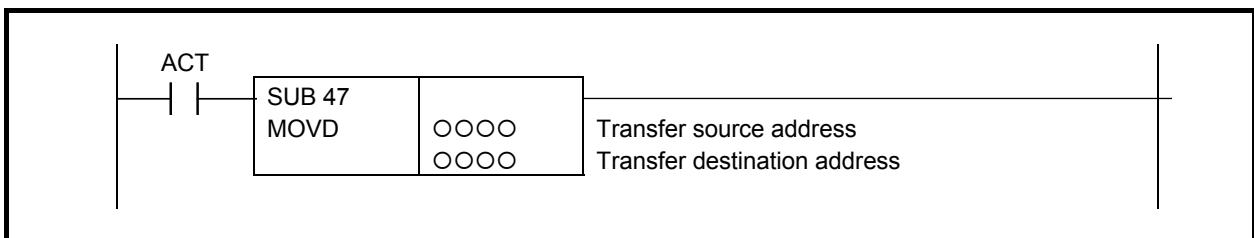


Fig. 4.5.3 Format of MOVD instruction

Table 4.5.3 Mnemonic of MOVD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		47	MOVD instruction				
3	(PRM)	0000		Transfer source address				
4	(PRM)	0000		Transfer destination address				▼

Control condition

- (a) Input signal
 - ACT = 0: No data is transferred.
 - ACT = 1: Four-byte data is transferred.

Parameters

- (a) Transfer source address
 - Specify the source address for the transfer.
- (b) Transfer destination address
 - Specify the destination address for the transfer.

NOTE

Take care not to specify overlapped areas for source and destination. If the source and destination areas are overlapped with each other, the result is not guaranteed.

4.5.4 MOVN (Transfer of an Arbitrary Number of Bytes: SUB 45)

The MOVN instruction transfers data consisting of an arbitrary number of bytes from a specified source address to a specified destination address.

Format

Fig. 4.5.4 shows the ladder format and Table 4.5.4 shows the mnemonic format.

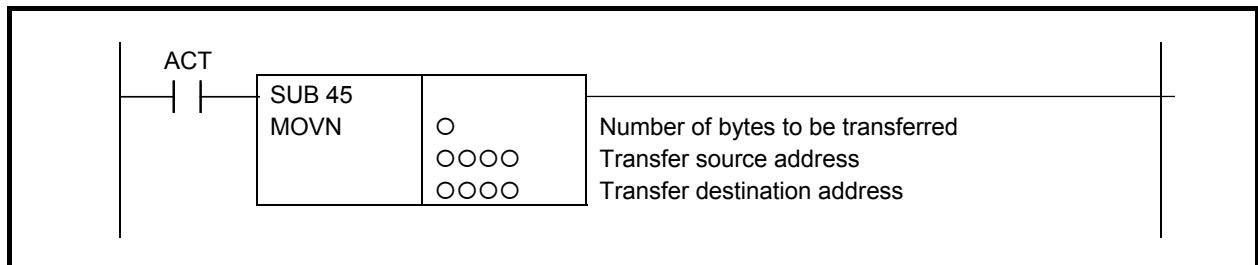


Fig. 4.5.4 Format of MOVN instruction

Table 4.5.4 Mnemonic of MOVN instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		45	MOVN instruction				
3			O	Number of bytes to be transferred				
4	(PRM)		0000	Transfer source address				
5	(PRM)		0000	Transfer destination address				

Control condition

- (a) Execution specification
ACT = 0: No data is transferred.
ACT = 1: A specified number of bytes are transferred.

Parameters

- (a) Number of bytes to be transferred
Specify the number of bytes to be transferred. An odd number can also be specified. A number from 1 to 9,999 can be specified.

⚠ CAUTION

Make sure that the source data area and destination data area are within the PMC address range.

- (b) Transfer source address
Specify the source address for the transfer.
- (c) Transfer destination address
Specify the destination address for the transfer.

NOTE

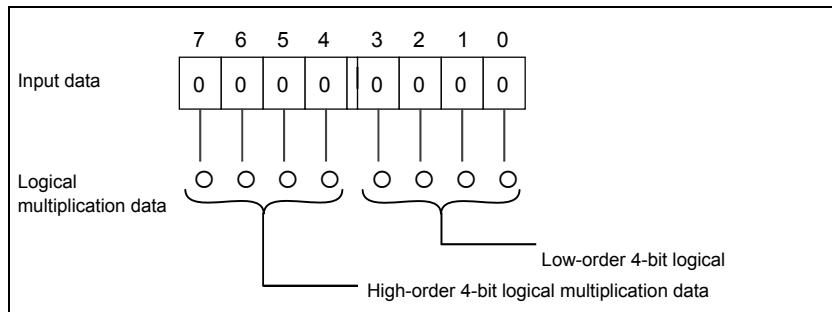
Take care not to specify overlapped areas for source and destination. If the source and destination areas are overlapped with each other, the result is not guaranteed.

4.5.5 MOVE (Logical Product Transfer: SUB 8)

ANDs logical multiplication data and input data, and outputs the results to a specified address. Can also be used to remove unnecessary bits from an eight-bit signal in a specific address, etc.

(Logical multiplication data) (Input data) to a specified address

The input data is one byte (eight bits).



Format

Fig. 4.5.5 (a) shows the ladder format and Table 4.5.5 shows the mnemonic format.

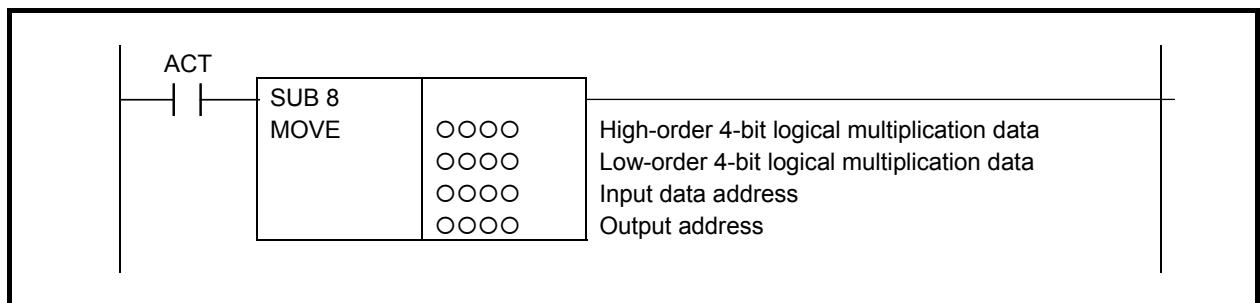


Fig. 4.5.5 (a) Format of MOVE instruction

Table 4.5.5 Mnemonic of MOVE instruction

Mnemonic format				Memory status of control condition
Step number	Instruction	Address No.	Bit No.	
1	RD	0000 .0		ACT
2	SUB		8	MOVE instruction
3	(PRM)	0000		High-order 4-bit logical multiplication data
4	(PRM)	0000		Low-order 4 bit logical multiplication data
5	(PRM)	0000		Input data address
6	(PRM)	0000		Output address

Execution command

ACT = 0: MOVE instruction not executed.

ACT = 1: MOVE instruction is executed.

Example of using the MOVE instruction

If a code signal and another signal co-exist at address X35 for an input signal from the machine tool, to compare the code signal and a code signal at another address, the rest of signals in address X35 become an obstacle. Thus, the MOVE instruction can be used to output only the code signal at address X35 address R210.

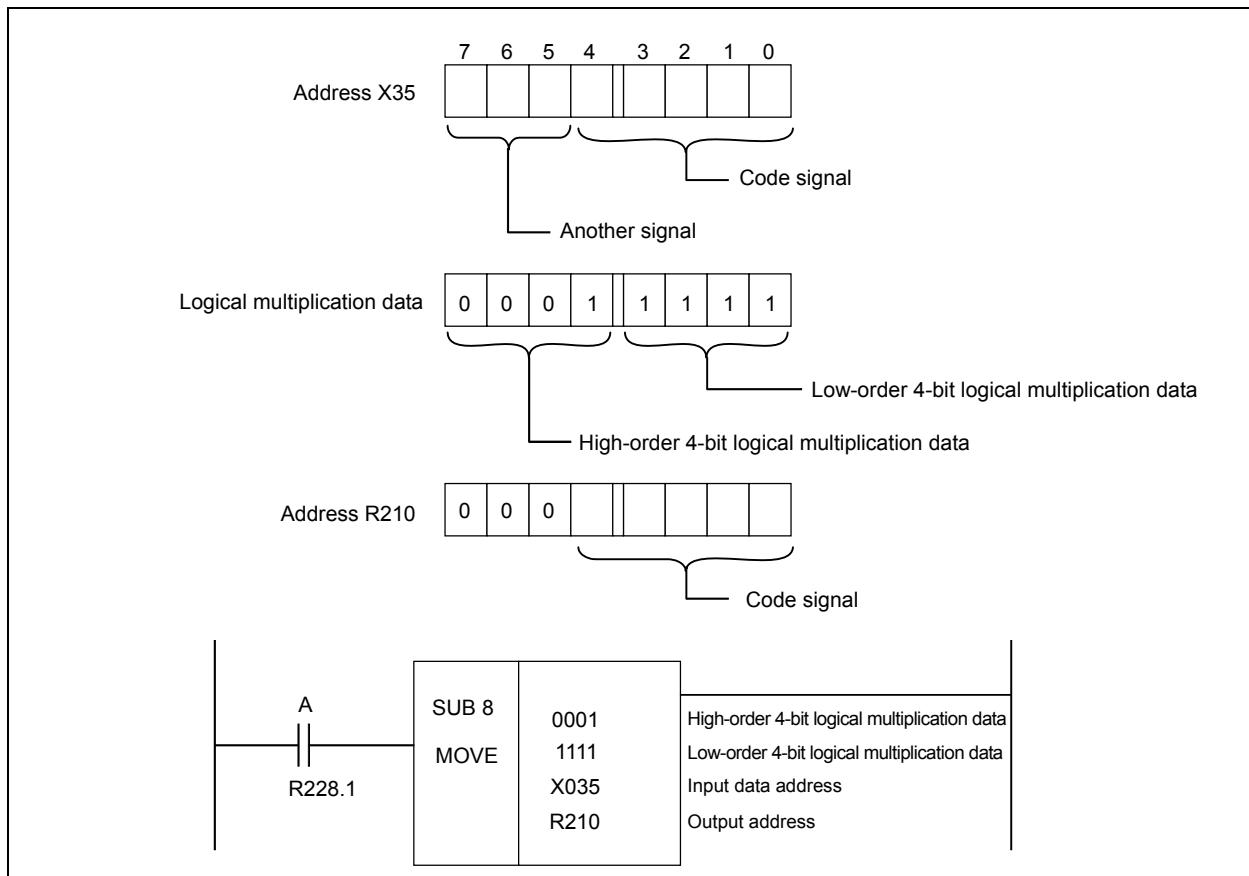
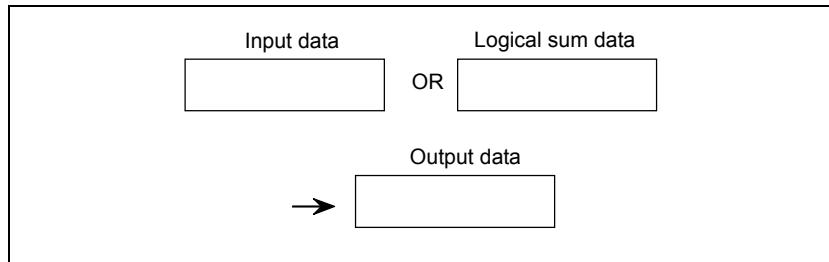


Fig. 4.5.5 (b) MOVE instruction ladder diagram

4.5.6 MOVOR (Data Transfer After Logical Sum: SUB 28)

This instruction ORs the input data and the logical sum data and transfers the result to the destination.



Format

Fig. 4.5.6 shows the ladder format and Table 4.5.6 shows the mnemonic format.

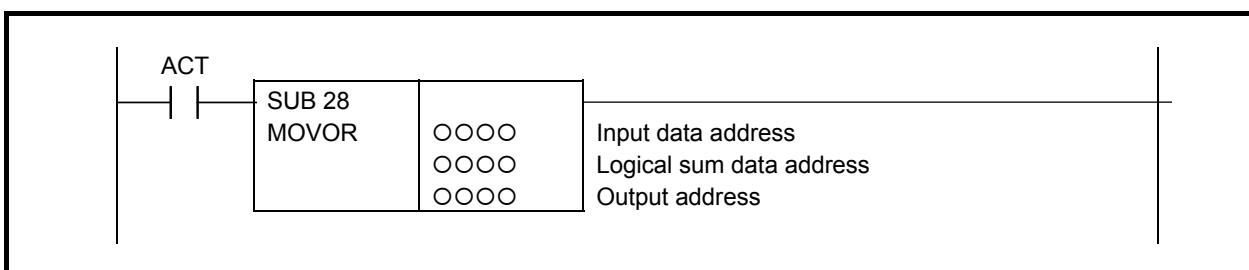


Fig. 4.5.6 Format of MOVOR instruction

Table 4.5.6 Mnemonic of MOVOR instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		28	MOVOR instruction				
3	(PRM)	0000		Input data address				
4	(PRM)	0000		Logical sum data address				
5	(PRM)	0000		Output address				▼

Control condition

- (a) Command (ACT)
 ACT = 0: Do not execute MOVOR.
 ACT = 1: Execute MOVOR.

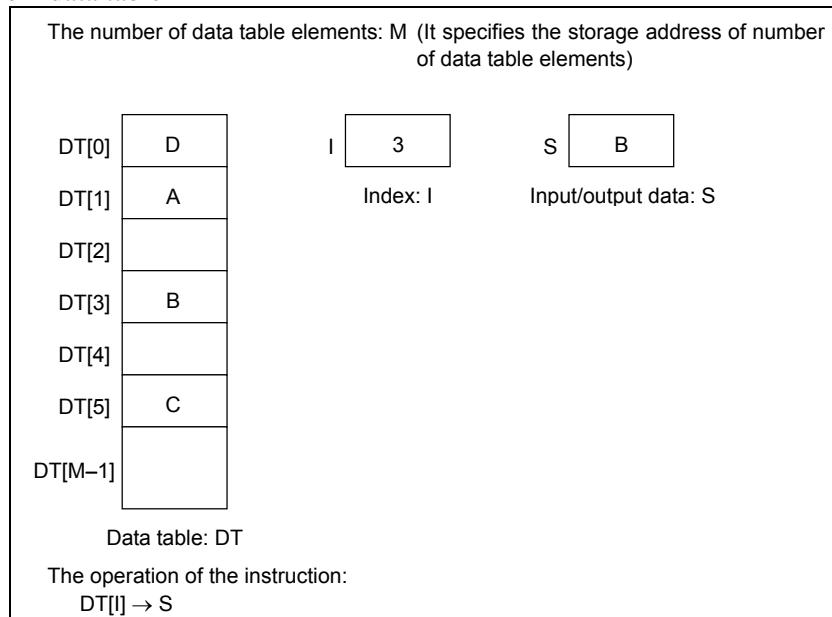
Parameters

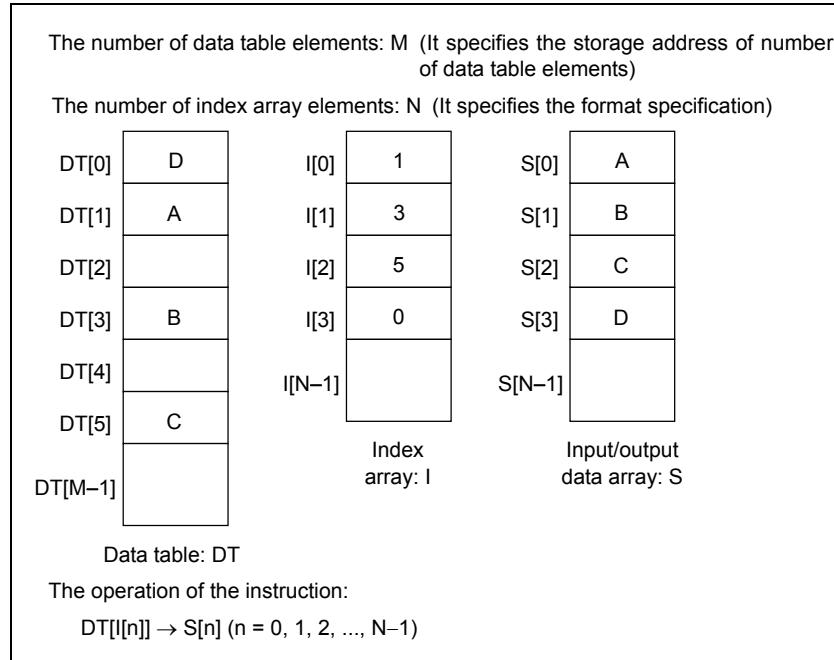
- (a) Input data address
 Specifies the address for the input data.
 (b) Logical sum data address
 Specifies the address of the logical sum data with which to OR the transferred data.
 (c) Output address
 This is the address to contain the logical sum obtained. It is also possible to obtain the logical sum (OR) of the input and the logical sum data and output the result in the logical sum data address. For this, you must set the logical sum data address for the output address.

4.5.7 XMOVB (Binary Index Modifier Data Transfer: SUB 35)

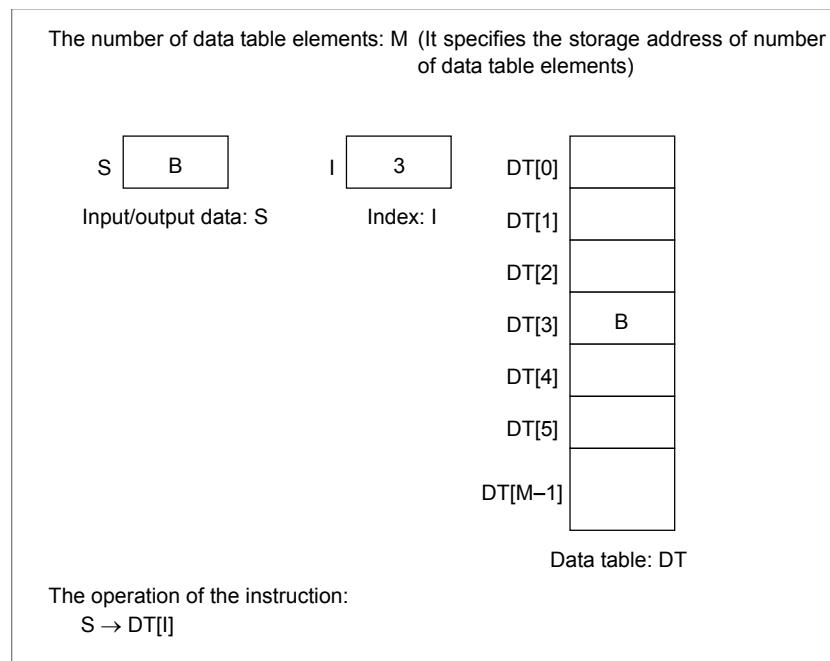
Reads or rewrites the contents of the data table. The value type in this instruction is binary. There are two specifications - basic specification and extended specification - for setting the format specification parameter in the XMOVB instruction. The extended specification allows two or more sets of data to be read or written with a single instruction. For the details of the setting of a format specification parameter, see the description of parameters.

- (a) Read data from data table

**Fig. 4.5.7 (a) Read data from data table (basic specification)**

**Fig. 4.5.7 (b) Read data from data table (extended specification)**

(b) Write data to data table

**Fig. 4.5.7 (c) Write data to data table (basic specification)**

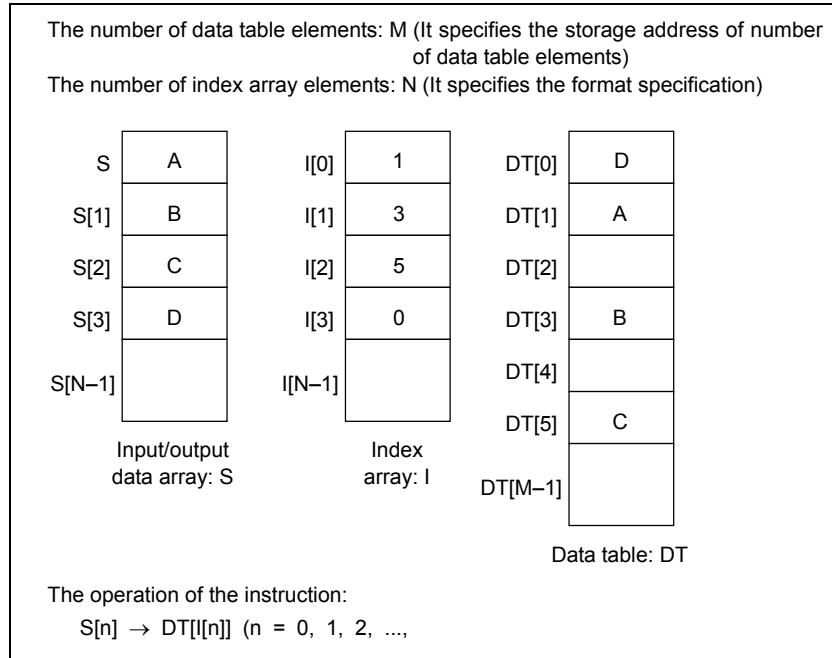


Fig. 4.5.7 (d) Write data to data table (extended specification)

Format

Figs. 4.5.7 (e) and (f) show the ladder format and Tables 4.5.7 (a) and (b) show the mnemonic format.

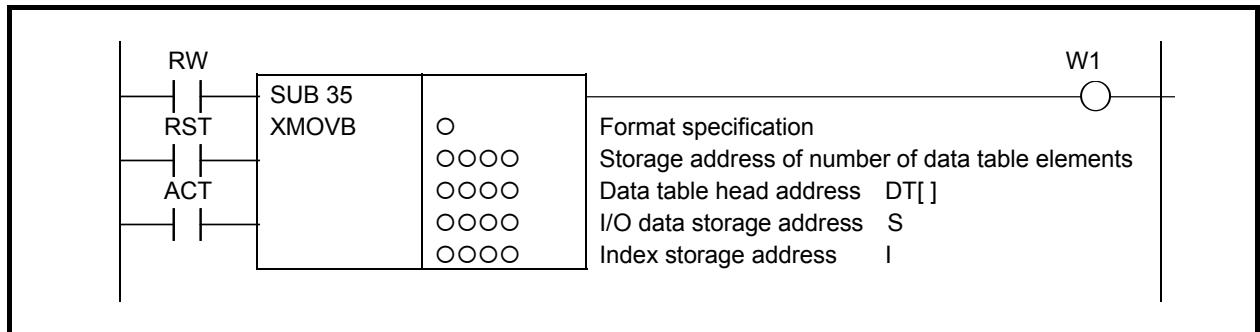


Fig. 4.5.7 (e) Format of XMOVB instruction (basic specification)

Table 4.5.7 (a) Mnemonic of MOVOR instruction (basic specification)

Step number	Instruction	Mnemonic format		Memory status of control condition
		Address No.	Bit No.	
1	RD	0000 .O	RW	
2	RD.STK	0000 .O	RST	
3	RD.STK	0000 .O	ACT	
4	SUB	35	XMOVB instruction	
5	(PRM)	O	Format specification	
6	(PRM)	OOOO	Storage address of number of data table elements	
7	(PRM)	OOOO	Data table head address	
8	(PRM)	OOOO	I/O data storage address	
9	(PRM)	OOOO	Index storage address	
10	WRT	0000 .O	Error output	W1

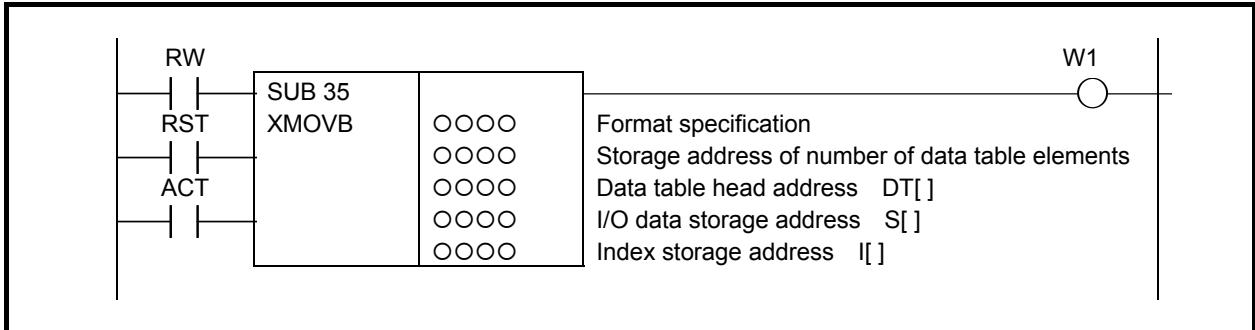


Fig. 4.5.7 (f) Format of XMOVB instruction (extended specification)

Table 4.5.7 (b) Mnemonic of MOVOR instruction (extended specification)

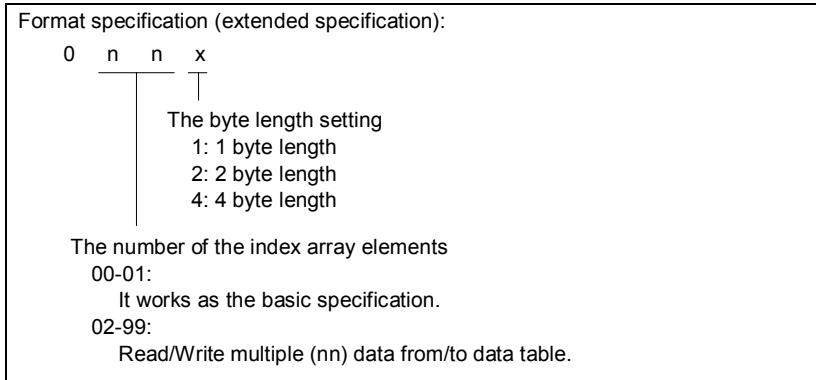
Mnemonic format				Memory status of control condition				
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O	RW					RW
2	RD.STK	0000 .O	RST				RW	RST
3	RD.STK	0000 .O	ACT		RW	RST	ACT	
4	SUB	35		XMOVB instruction				
5	(PRM)	0000		Format specification				
6	(PRM)	0000		Storage address of number of data table elements				
7	(PRM)	0000		Data table head address				
8	(PRM)	0000		I/O data storage address				
9	(PRM)	0000		Index storage address				
10	WRT	0000 .O		Error output				W1

Control conditions

- (a) Read, write designation (RW)
 - RW = 0: Read data from data table.
 - RW = 1: Write data to data table.
- (b) Reset (RST)
 - RST = 0: Reset release.
 - RST = 1: Reset. W1 = 0.
- (c) Activation command (ACT)
 - ACT = 0: Do not execute XMOVB instruction. There is no change in W1.
 - ACT = 1: Execute XMOVB instruction.

Parameters

- (a) Format specification
 - Specifies data length. Specify byte length in the first digit of the parameter.
 - 0001:1-byte length data
 - 0002:2-byte length data
 - 0004:4-byte length data
 - When setting format specification in the following extended format, XMOVB can read/write multiple data in data table in 1 instruction.
 - Specifies data length (1, 2, or 4) to the 1st digit as above-mentioned. Specifies the number of the index array elements to the 2nd and 3rd digit. Specifies 0 to the 4th digit.
 - 0nn1:In case of reading/writing multiple (nn) data in data table by 1 byte length
 - 0nn2:In case of reading/writing multiple (nn) data in data table by 2 bytes length
 - 0nn4:In case of reading/writing multiple (nn) data in data table by 4 bytes length
 - The nn is the numerical value from 02 to 99. When setting 00 or 01, it works as the basic specification in which one data transfer is performed by one instruction.



(b) Storage address of number of data table elements

Set to the memory at the byte length which set the number of the data table elements in "(a) Format specification" and set the address to this parameter. The value which you can set depends on the "(a) Format specification" setting.

- 1 byte length: 1 to 255
- 2 bytes length: 1 to 16384
- 4 bytes length: 1 to 16384

(c) Data table head address

Sets head address in the data table.

The memory of (byte length) × (number of data table elements) which was set in "(a) Format specification" and "(b) Storage address of number of data table elements" is necessary.

(d) Input/Output data storage address

In case of the reading, set the address of the memory which stores a reading result. In case of the writing, set the address of the memory which stores a writing result. The memory with the byte length which set in "(a) Format specification" is necessary.

When setting format specification in the extended format, set the head address of the array. (In case of the reading, set the head address of the array in which a reading result is stored. In case of the writing, set the head address of the array in which a writing result is stored.) The memory of (byte length) × (number of index array elements) which was set in "(a) Format specification" is necessary.

(e) Index storage address

Set the address of the memory in which an index value is stored. The memory with the byte length set in "(a) Format specification" is necessary. The effective range of number of data in index is as follows according to the byte length set in "(a) Format specification".

Actually, set the value which is smaller than the value to set in "(b) Storage address of number of data table elements" to the index.

When setting an index value above the value to set in "(b) Storage address of number of data table elements", it causes an error output W1 = 1 in instruction execution.

- 1 byte length: 0 to 254
- 2 bytes length: 0 to 32,766
- 4 bytes length: 0 to 2,147,483,646

When setting format specification in the extended format, set an address at the head of the array in which an index value is stored. The memory of (byte length) × (number of data in index array) which was set in "(a) Format specification" is necessary.

⚠ WARNING

- 1 You can not specify the table that includes different kind of address type or discontinuous address area. In this case, operation is not guaranteed.

⚠ WARNING

- 2 You have to set the "Storage address of number of data table elements" and the "Data table head address" not to exceed the limit of its continuous address area. If the table exceeds the limit of the continuous address area, operation is not guaranteed. For example, when a range of address R is 0 to 7999 and the "Format specification" is set to 1 and the "Data table head address" is set to "R7990", you can set 10 or less to the "Storage address of number of data table elements".

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

W1 = 0: No error

W1 = 1: Error found.

In the case where the index value set in "(e) Index storage address" exceeds the value set in "(b) Storage address of number of data table elements", it becomes W1 = 1. The reading or writing of the data table isn't executed.

When "(a) Format specification" is used for operation in the extended format, if the values of one or more elements in the index array specified in (e) are greater than the value set in "(b) Storage address of number of data table elements", it becomes W1 = 1. The reading or writing of a data table is executed for the normal index values but not executed as for the wrong index values.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Example for extended specification

- (a) Read data from data table (extended specification)

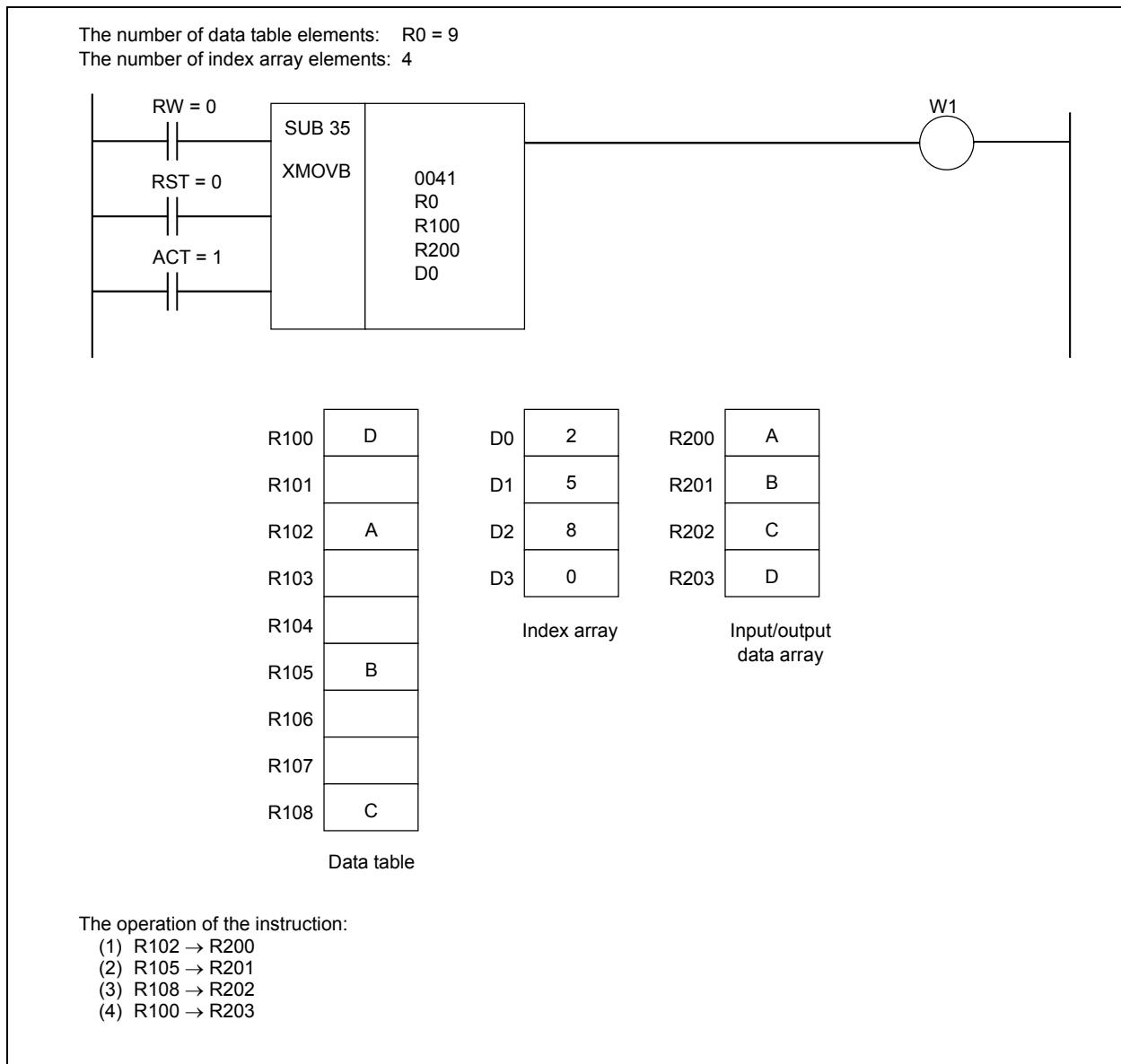
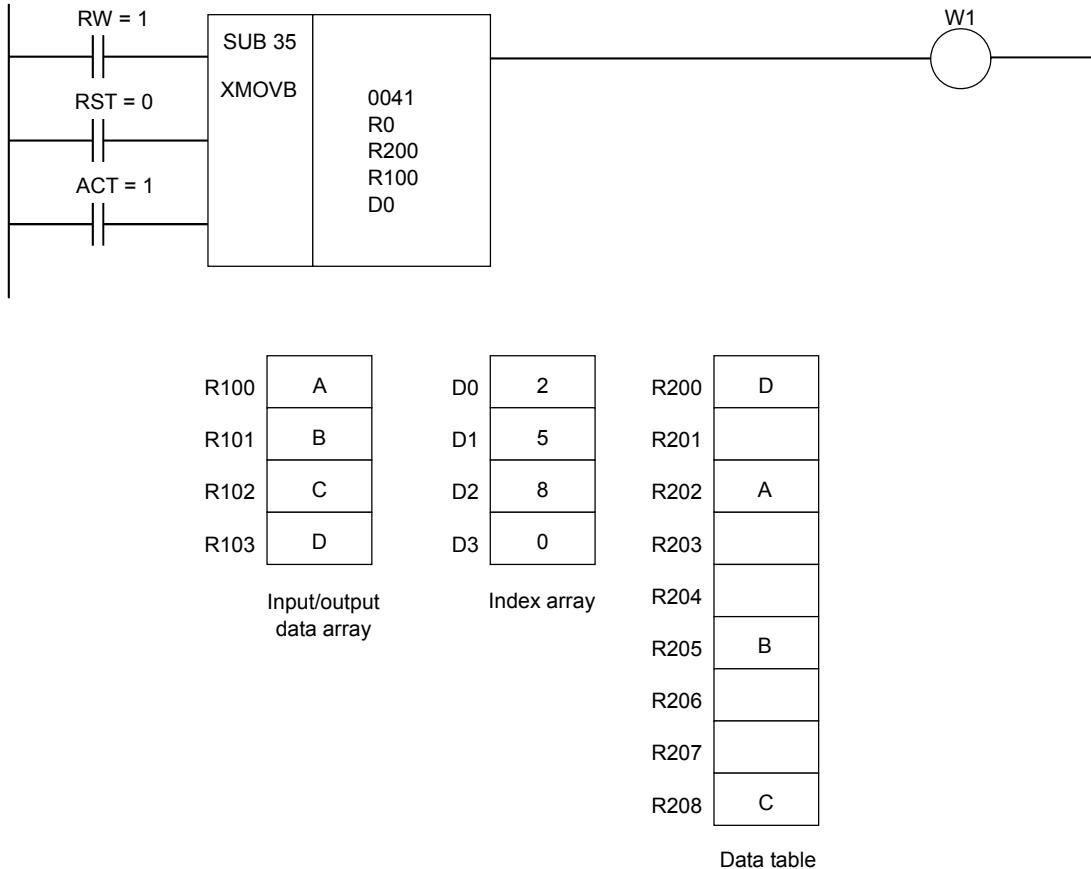


Fig. 4.5.7 (g) Example for XMOVB instruction (extended specification)

(b) Write data to data table (extended specification)

The number of data table elements: R0 = 9
 The number of index array elements: 4



The operation of the instruction:

- (1) R100 → R202
- (2) R101 → R205
- (3) R102 → R208
- (4) R103 → R200

Fig. 4.5.7 (h) Example for XMOVB instruction (extended specification)

4.5.8 XMOV (Indexed Data Transfer: SUB 18)

Reads or rewrites the contents of the data table. The value type in this instruction is binary.

⚠ CAUTION

The data table heading address specified here is table internal number 0. The table internal number specified here, however, is different from that mentioned in Subsection 2.2.11.

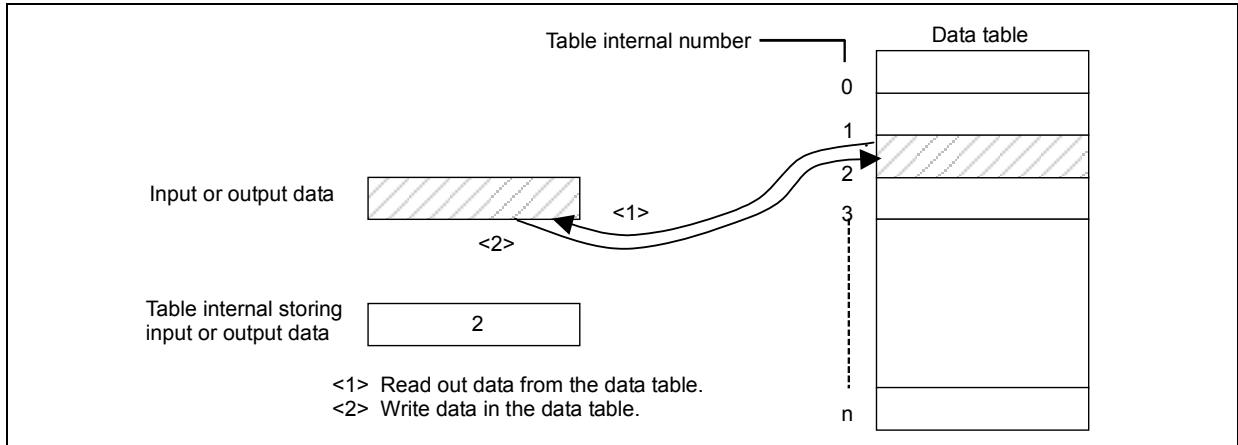


Fig. 4.5.8 (a) Reading and writing of data

Format

Fig. 4.5.8 (b) shows the ladder format and Table 4.5.8 shows the mnemonic format.

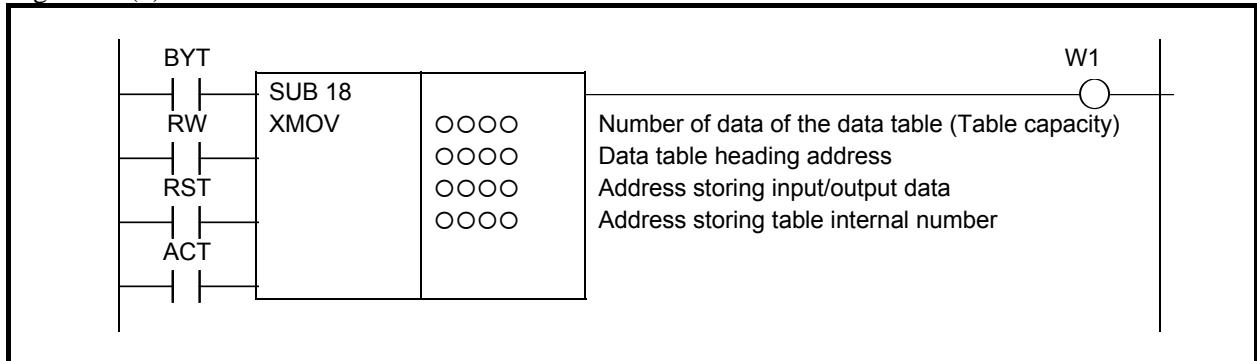


Fig. 4.5.8 (b) Format of XMOV instruction

Table 4.5.8 Mnemonic of XMOV instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		RW			BYT	RW
3	RD.STK	0000 .0		RST		BYT	RW	RST
4	RD.STK	0000 .0		ACT	BYT	RW	RST	ACT
5	SUB	18		XMOV instruction				
6	(PRM)	0000		Number of data of the data table				
7	(PRM)	0000		Data table heading address				
8	(PRM)	0000		Address storing input/output data				
9	(PRM)	0000		Address storing table internal number				
10	WRT	0000 .0		Error output				▼

Control conditions

- (a) Specify the number of digits of data. (BYT)
BYT = 0: Data stored in the data table, BCD in two digits long.
BYT = 1: Data stored in the data table, BCD in four digits long.
 - (b) Specify read or write (RW)
RW = 0: Data is read from the data table.
RW = 1: Data is written in the data table.
 - (c) Reset (RST)
RST = 0: Release reset.
RST = 1: Enables reset, that is, sets W1 to 0.

- (d) Execution command (ACT)
 ACT = 0: The XMOV instruction is not executed. W1 does not change.
 ACT = 1: The XMOV instruction is executed.

Parameters

- (a) Number of data of the data table
 Specifies the size of the data table. If the beginning of the data table is 0 and the end is n, n + 1 is set as the number of data of the data table. The value, which you can set, depends on the control condition "BYT".
 BYT=0: 1 to 99
 BYT=1: 1 to 9999
- (b) Data table heading address
 Addresses that can be used in a data table are fixed. When preparing a data table, the addresses to be used must be determined beforehand, and the head address placed in that data table.
- (c) Address storing input/output data
 The input/output data storage address is the address storing the specified data, and is external to the data table. The contents of the data table is read or rewritten.
- (d) Address storing table internal number
 The table internal number storage address is the address storing the table internal number of the data to be read or rewritten.
 This address requires memory specified by the number-of-digits designation (BYT).

WARNING

- 1 You can not specify the table that includes different kind of address type or discontinuous address area. In this case, operation is not guaranteed.
- 2 You have to set the "Number of data of the data table" and the "Data table heading address" not to exceed the limit of its continuous address area. If the table exceeds the limit of the continuous address area, operation is not guaranteed. For example, when a range of address R is 0 to 7999 and the control condition "BYT" is set to 0 and the "Data table heading address" is set to "R7990", you can set 10 or less to the "Number of data of the data table".

Error output

W1 = 0: There is no error.

W1 = 1: There is an error.

An error occurs if a table internal number exceeding the previously programmed number of the data table is specified.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.5.9 MOVBT (Bit Transfer: SUB 224)

The Bit transfer instruction transfers multiple successive bits at a specified position to a destination address.

Transfer source data is specified in "Transfer source address" and "Transfer source bit position". Transfer destination data is specified in "Transfer destination address" and "Transfer destination bit position".

From "Transfer source bit position", data consisting of successive bits as many as "Number of bits to be transferred" is transferred to "Transfer destination address".

When 3 bits are transferred from R100.1 to R500.4:

Number of bits to be transferred = 3
 Transfer source address = R100
 Transfer source bit position = 1
 Transfer destination address = R500
 Transfer destination bit position = 4

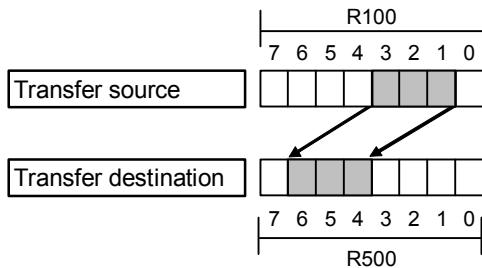


Fig. 4.5.9 (a) Example of MOVBT instruction (1)

When 3 bits are transferred from R100.6 to R500.3:

Number of bits to be transferred = 3
 Transfer source address = R100
 Transfer source bit position = 6
 Transfer destination address = R500
 Transfer destination bit position = 3

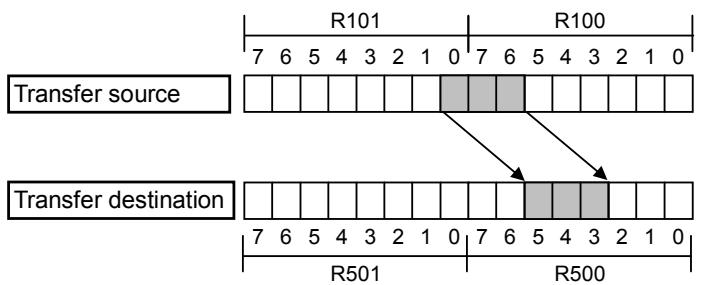


Fig. 4.5.9 (b) Example of MOVBT instruction (2)

When 3 bits are transferred from R100.4 to R500.7:

Number of bits to be transferred = 3
 Transfer source address = R100
 Transfer source bit position = 4
 Transfer destination address = R500
 Transfer destination bit position = 7

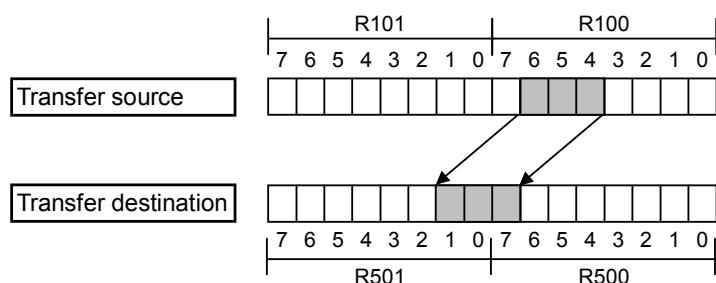


Fig. 4.5.9 (c) Example of MOVBT instruction (3)

Format

Fig. 4.5.9(d) shows the ladder format and Table 4.5.9 shows the mnemonic format.

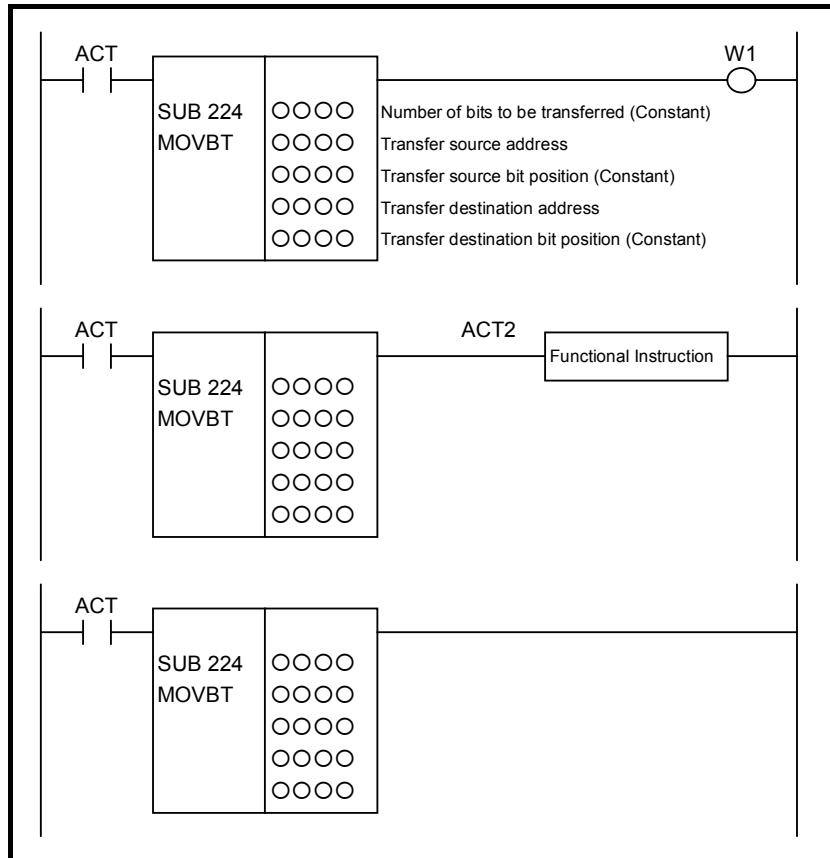


Fig. 4.5.9 (d) Format of MOVBT instruction

Table 4.5.9 Mnemonic of MOVBT instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		224	MOVBT instruction				
3	(PRM)	0000		Number of bits to be transferred				
4	(PRM)	0000		Transfer source address				
5	(PRM)	0000		Transfer source bit position				
6	(PRM)	0000		Transfer destination address				
7	(PRM)	0000		Transfer destination bit position				
8	WRT	0000 .0		Normal end output				W1

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of bits to be transferred
Specify the number of bits to be transferred. A number from 1 to 256 may be specified.
- (b) Transfer source address

Specify the source address for the transfer.

NOTE

Bits are transferred even when "Transfer source address" and "Transfer destination address" overlap each other.

- (c) Transfer source bit position

Specify the transfer start bit position of transfer source data. A number from 0 to 7 may be specified.

- (d) Transfer destination address

Specify the destination address for the transfer.

NOTE

Bits are transferred even when "Transfer source address" and "Transfer destination address" overlap each other.

- (e) Transfer destination bit position

Specify the top bit position of transfer destination data. A number from 0 to 7 may be specified.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.5.10 SETNB (Data Setting (1 Byte Length) : SUB 225) SETNW (Data Setting (2 Bytes Length) : SUB 226) SETND (Data Setting (4 Bytes Length) : SUB 227)

The data setting instruction sets the same value in multiple data items at contiguous addresses.

In "Setting data", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of data setting instructions are available according to the type of data to be set. In each instruction, "Setting data" and the data at "Setting destination address" are of the same data type.

Table4.5.10 (a) Kinds of data setting instruction

	Instruction name	SUB No.	Data type
1	SETNB	225	1 byte length data
2	SETNW	226	2 bytes length data
3	SETND	227	4 bytes length data

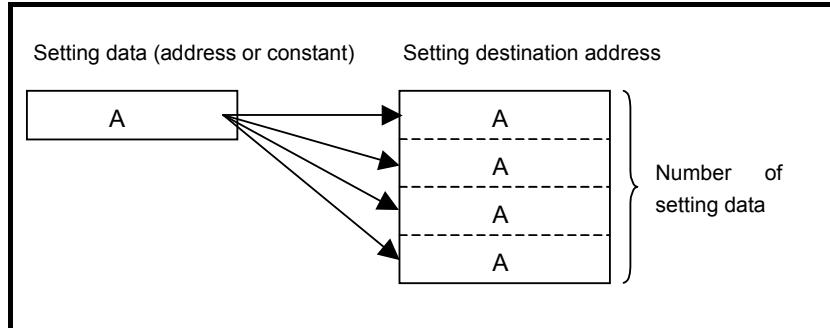


Fig. 4.5.10 (a) Example of data setting instruction

Format

Fig. 4.5.10(b) shows the ladder format and Table 4.5.10(b) shows the mnemonic format.

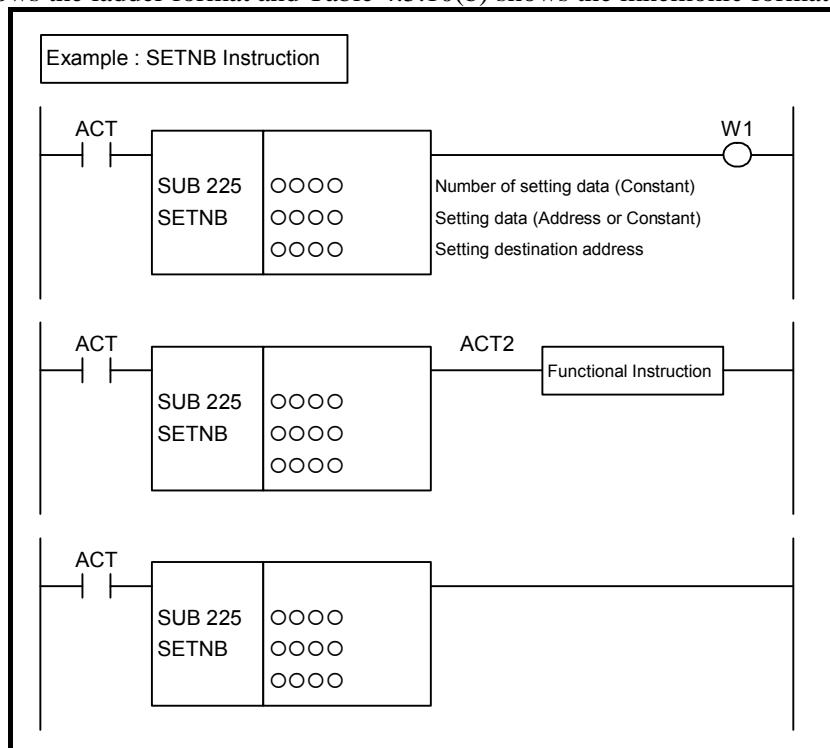


Fig. 4.5.10 (b) Format of SETNB, SETNW, SETND instruction

Table 4.5.10 (b) Mnemonic of SETNB, SETNW, SETND instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	OOOO .O		ACT
2	SUB	225		SUB No. (SETNB instruction)
3	(PRM)	OOOO		Number of setting data (Constant)
4	(PRM)	OOOO		Setting data (Address or Constant)
5	(PRM)	OOOO		Setting destination address
6	WRT	OOOO .O		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control condition

(a) Input signal

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

- (a) Number of setting data

Specify the number of setting data items. A number from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Number of setting data", so that the area from "Setting destination address" may be arranged within valid address range.

- (b) Setting data

Specify data to be set. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available values
SETNB	-128 to 127
SETNW	-32768 to 32767
SETND	-2147483648 to 2147483647

- (c) Setting destination address

Specify a setting destination address.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.5.11 XCHGB (Data Exchange (1 Byte Length) : SUB 228) XCHGW (Data Exchange (2 Bytes Length) : SUB 229) XCHGD (Data Exchange (4 Bytes Length) : SUB 230)

The data exchange instruction exchanges data between two specified addresses.

As indicated below, three types of data exchange instructions are available according to the type of data to be exchanged. In each instruction, the data items at exchange addresses are of the same data type.

Table 4.5.11 (a) Kinds of data exchange instruction

	Instruction name	SUB No.	Data type
1	XCHGB	228	1 byte length data
2	XCHGW	229	2 bytes length data
3	XCHGD	230	4 bytes length data

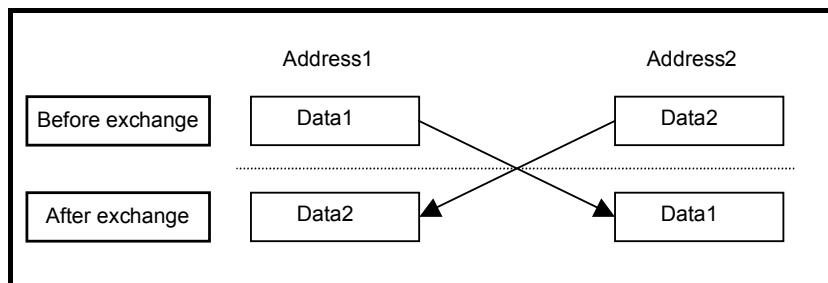


Fig. 4.5.11 (a) Example of data exchange instruction

Format

Fig. 4.5.11(b) shows the ladder format and Table 4.5.11(b) shows the mnemonic format.

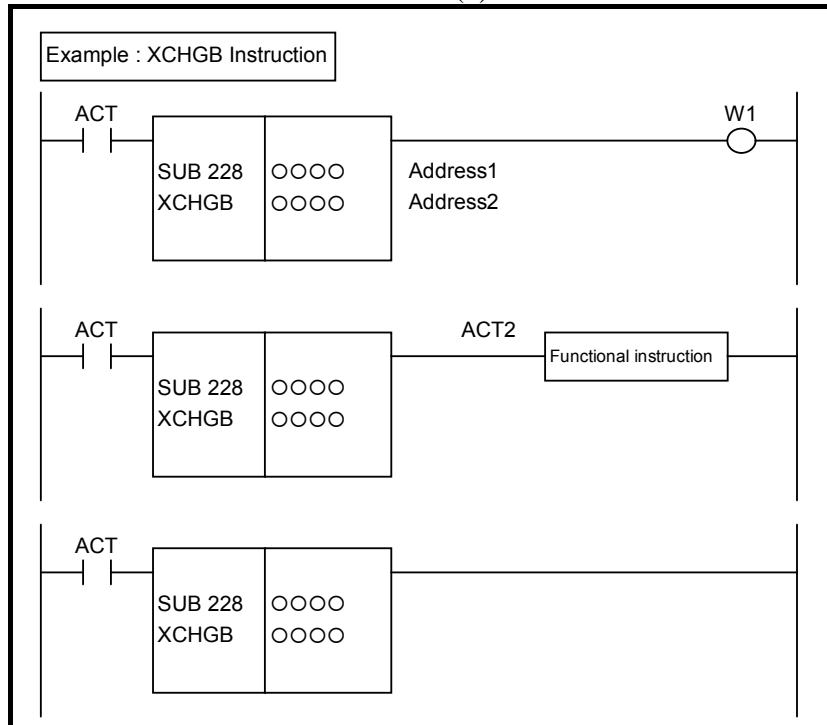


Fig. 4.5.11 (b) Format of XCHGB, XCHGW, XCHGD instruction

Table 4.5.11(b) Mnemonic of XCHGB, XCHGW, XCHGD instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .O		ACT
2	SUB		228	SUB No. (XCHGB instruction)
3	(PRM)	0000		Address 1
4	(PRM)	0000		Address 2
5	WRT	0000 .O		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Address 1
Specify the 1st address which exchanges data.
- (b) Address 2
Specify the 2nd address which exchanges data.

NOTE

If Address 1 and Address 2 areas are overlapped with each other, the result is not guaranteed.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.5.12 SWAPW (Data Swap (2 Bytes Length) : SUB 231) SWAPD (Data Swap (4 Bytes Length) : SUB 232)

The data swap instruction swaps the high-order data and low-order data of multiple data items at contiguous addresses with each other.

The number of data items to be swapped is specified using a constant. Swap source data and a result output destination are specified using addresses.

As indicated below, two types of data swap instructions are available according to the type of data to be swapped. The SWAPW instruction swaps the higher one byte and lower one byte of each data item with each other. The SWAPD instruction swaps the higher two bytes and lower two bytes of each data item with each other.

In each instruction, source data and output data are of the same data type.

Table 4.5.12 (a) Kinds of data swap instruction

	Instruction name	SUB No.	Data type
1	SWAPW	231	2 bytes length data
2	SWAPD	232	4 bytes length data

Example which swaps data 2 bytes long:

Number of data = 10
Source data top address = R100
Result output top address = D500

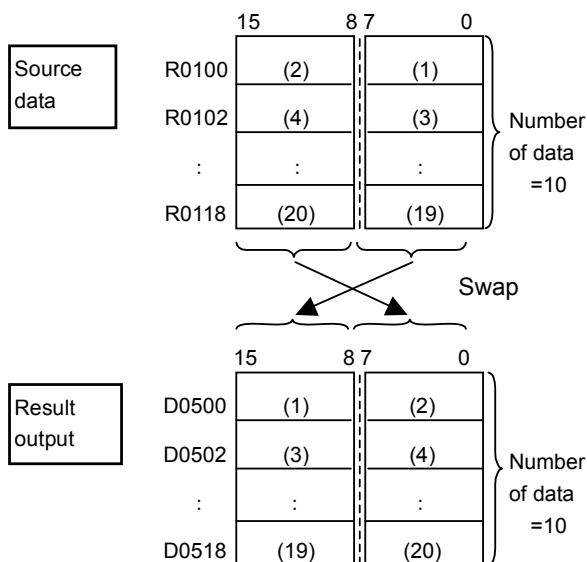


Fig. 4.5.12 (a) Example of SWAPW instruction

Example which swaps data 4 bytes long:

Number of data = 10

Source data top address = R100

Result output top address = D500

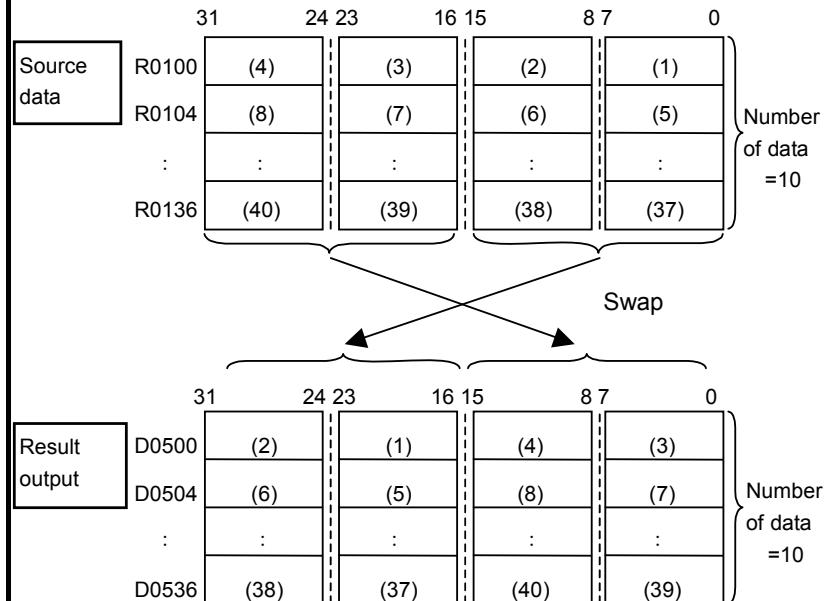


Fig. 4.5.12 (b) Example of SWAPD instruction

Format

Fig. 4.5.12(c) shows the ladder format and Table 4.5.12(b) shows the mnemonic format.

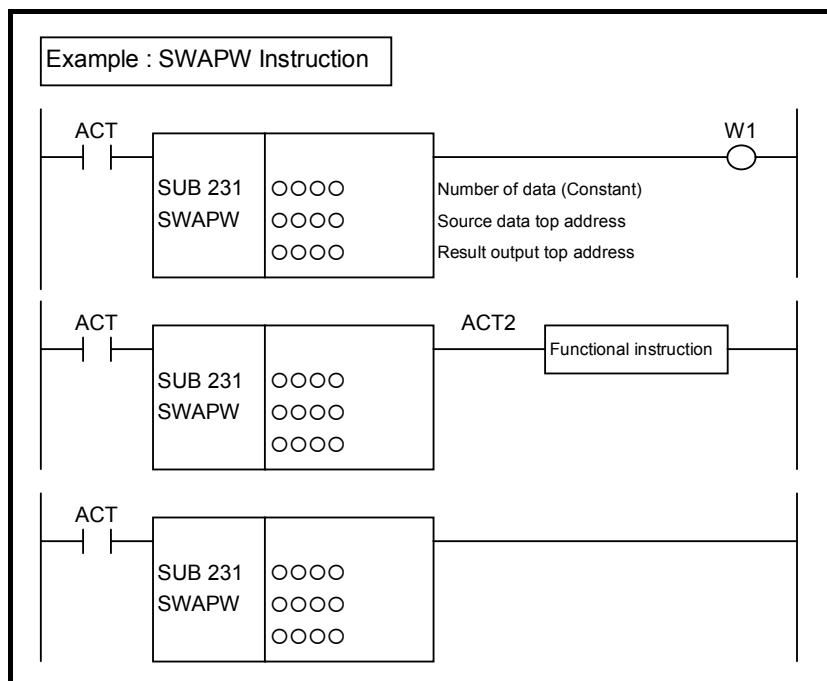


Fig. 4.5.12 (c) Format of SWAPW, SWAPD instruction

Table 4.5.12 (b) Mnemonic of SWAPW, SWAPD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		231	SUB No. (SWAPW instruction)				
3	(PRM)	0000		Number of data (Constant)				
4	(PRM)	0000		Source data top address				
5	(PRM)	0000		Result output top address				
6	WRT	0000 .0		Normal end output				W1

Control condition

- (a) Input signal
 ACT = 0: Instruction not executed.
 ACT = 1: Executed.

Parameters

- (a) Number of data
 Specify the number of data items to be swapped. A number from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Number of data", so that both of the area from "Source data top address" and the area from "Result output top address" may be arranged within valid address range.

- (b) Source data top address
 Specifies the top address in which the swap data is stored.
- (c) Result output top address
 Specifies the top of address which stores the result of an operation.

NOTE

If "Source data top address" and "Result output top address" match each other completely, the instruction is executed normally. If the source data area partially overlaps the result output area, normal operation of the instruction is not guaranteed.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.5.13 DSCHB (Binary Data Search: SUB 34)

This function instruction instructs data search in the data table. DSCHB searches the data table for a specified data, outputs an address storing it counting from the beginning of the data table. If the data cannot be found, an output is made accordingly.

4. LADDER LANGUAGE

B-83254EN/02

The numerical data handled in this instruction are all in binary format and number of data (table capacity) in the data table can be specified by specifying the address, thus allowing change in table capacity even after writing the sequence program in the flash ROM.

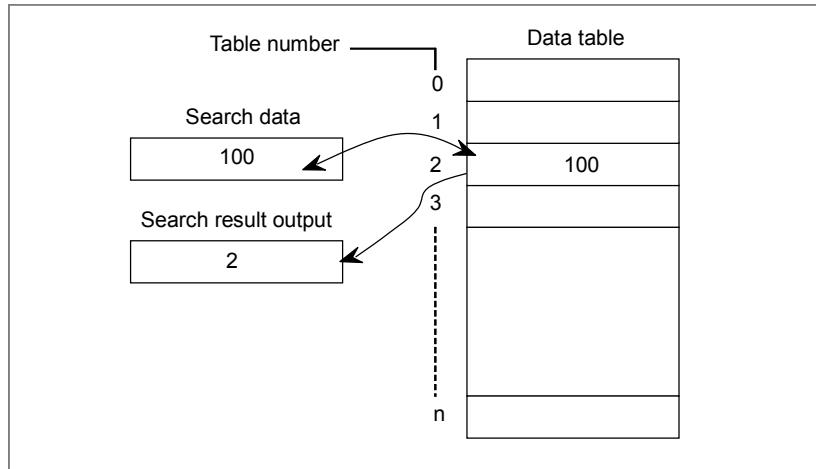


Fig. 4.5.13 (a)



CAUTION

You can specify any R, E and D address for the data table in this functional instruction.

Format

Fig. 4.5.13 (b) shows the ladder format and Table 4.5.13 shows the mnemonic format.

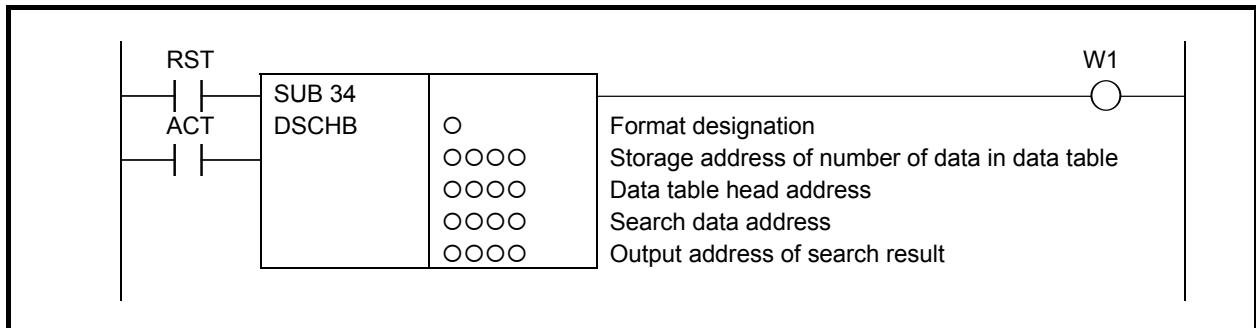


Fig. 4.5.13 (b) Format of DSCHB instruction

Table 4.5.13 Mnemonic of DSCHB instruction

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		RST				RST
2	RD.STK	0000 .0		ACT			RST	ACT
3	SUB		34	DSCHB instruction				
4	(PRM)		0	Format designation				
5	(PRM)	0000		Storage address of number of data in data table				
6	(PRM)	0000		Data table head address				
7	(PRM)	0000		Search data address				
8	(PRM)	0000		Output address of search result				
9	WRT	0000 .0		Search result				

Control conditions

- (a) Reset (RST)
 - RST = 0: Release reset
 - RST = 1: Reset. W1 = "0".

- (b) Activation command (ACT)
 - ACT = 0: Do not execute DSCHB instruction. W1 does not change.
 - ACT = 1: Execute DSCHB instruction. If the search data is found, table number where the data is stored will be output. If the search data is not found, W1 becomes 1.

Parameters

- (a) Format specification
 - Specifies data length. Specify byte length in the first digit of the parameter.
 - 1: 1 byte length
 - 2: 2 bytes length
 - 4: 4 bytes length

- (b) Storage address of number of data in data table
 - Specifies address in which number of data in the data table is set.
 - This address requires memory of number of byte according to the format designation.
 - Number of data in the table is n + 1 (head number in the table is 0 and the last number is n). The value which you can set depends on the "(a) Format designation".
 - 1 byte length: 1 to 255
 - 2 bytes length: 1 to 16384
 - 4 bytes length: 1 to 16384

- (c) Data table head address
 - Sets head address of data table.

- (d) Search data address
 - Address in which search data is set.

- (e) Output address of search result
 - After searching, if search data is found, the table number where the data is stored will be output.
 - The searched table number is output in this search result output address. This address requires memory of number of byte according to the format designation.

⚠ WARNING

- 1 You can not specify the table that includes different kind of address type or discontinuous address area. In this case, operation is not guaranteed.
- 2 You have to set the "Storage address of number of data table elements" and the "Data table head address" not to exceed the limit of its continuous address area. If the table exceeds the limit of the continuous address area, operation is not guaranteed. For example, when a range of address R is 0 to 7999 and the "Format specification" is set to 1 and the "Data table head address" is set to "R7990", you can set 10 or less to the "Storage address of number of data table elements".

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Search result (W1)

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

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.5.14 DSCH (Data Search: SUB 17)

This function instruction instructs data search in the data table. DSCH searches the data table for a specified data, outputs an address storing it counting from the beginning of the data table. If the data cannot be found, an output is made accordingly. The value type in this instruction is BCD.

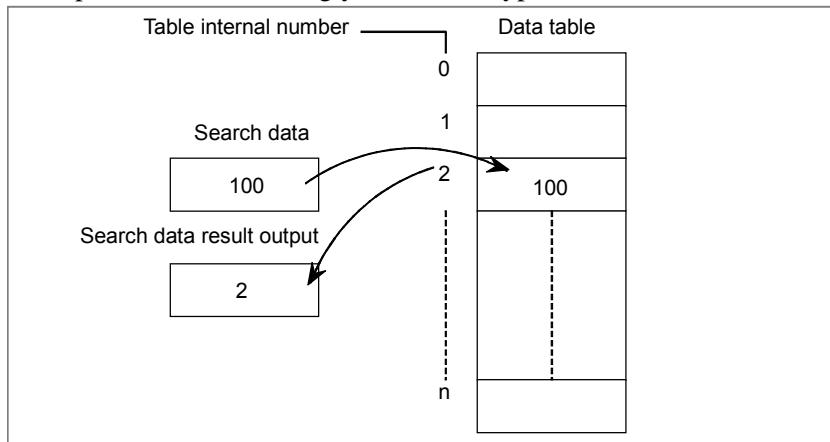


Fig. 4.5.14 (a)

⚠ CAUTION

You can specify any R, E and D address for the data table in this functional instruction.

Format

Fig. 4.5.14 (b) shows the ladder format and Table 4.5.14 shows the mnemonic format.

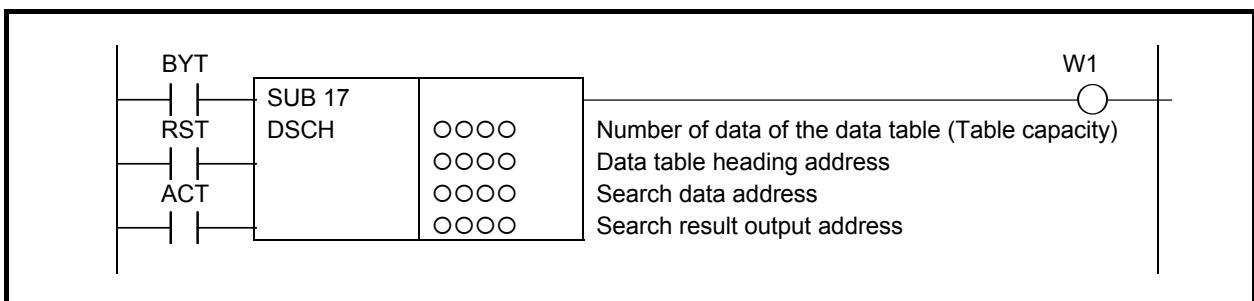


Fig. 4.5.14 (b) Format of DSCH instruction

Table 4.5.14 Mnemonic of DSCH instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		RST			BYT	RST
3	RD.STK	0000 .0		ACT	BYT	RST	ACT	
4	SUB		17	DSCH instruction				
5	(PRM)	0000		Number of data of the data table				
6	(PRM)	0000		Data table heading address				
7	(PRM)	0000		Search data address				
8	(PRM)	0000		Search result output address				
9	WRT	0000 .0		Search result	▼	▼		W1

Control conditions

- (a) Specify data size. (BYT)
 - BYT = 0: Data stored in the data table, BCD two digits long.
 - BYT = 1: Data stored in the data table, BCD four digits long.
- (b) Reset (RST)
 - RST = 0: Release reset
 - RST = 1: Enables a reset, that is, sets W1 to 0.
- (c) Execution command (ACT)
 - ACT = 0: The DSCH instruction is not executed. W1 does not change.
 - ACT = 1: The DSCH is executed, and the table internal number storing the desired data is output. If the data cannot be found, W1 = 1.

Parameters

- (a) Number of data of the data table
 - Specifies the size of the data table. If the beginning of the data table is 0 and the end is n, n + 1 is set as the number of data of the data table. The value which you can set depends on the control condition "BYT".
 - BYT=0: 1 to 99
 - BYT=1: 1 to 9999
- (b) Data table heading address
 - Addresses that can be used in a data table are fixed. When preparing a data table, the addresses to be used must be determined beforehand, specify the head address of a data table here.
- (c) Search data address
 - Indicates the address of the data to be searched.
- (d) Search result output address
 - If the data being searched for is found, the internal number of the table storing the data is output to this field. This address field is called a search result output address field.
 - The search result output address field requires memory whose size is the number of bytes conforming to the size of the data specified by BYT.

⚠ WARNING

- 1 You can not specify the table that includes different kind of address type or discontinuous address area. In this case, operation is not guaranteed.
- 2 You have to set the "Number of data of the data table" and the "Data table heading address" not to exceed the limit of its continuous address area. If the table exceeds the limit of the continuous address area, operation is not guaranteed. For example, when a range of address R is 0 to 7999 and the control condition "BYT" is set to 0 and the "Data table heading address" is set to "R7990", you can set 10 or less to the "Number of data of the data table".

Search result (W1)

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



CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.6 TABLE DATA

The following types of table data instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	TBLRB	233	Reading data from table (1 byte length)
2	TBLRW	234	Reading data from table (2 bytes length)
3	TBLRD	235	Reading data from table (4 bytes length)
4	TBLRN	236	Reading data from table (Arbitrary byte length)
5	TBLWB	237	Writing data to table (1 byte length)
6	TBLWW	238	Writing data to table (2 bytes length)
7	TBLWD	239	Writing data to table (4 bytes length)
8	TBLWN	240	Writing data to table (Arbitrary byte length)
9	DSEQB	241	Searching data from table (=)(1 byte length)
10	DSEQW	242	Searching data from table (=)(2 bytes length)
11	DSEQD	243	Searching data from table (=)(4 bytes length)
12	DSNEB	244	Searching data from table (≠)(1 byte length)
13	DSNEW	245	Searching data from table (≠)(2 bytes length)
14	DSNED	246	Searching data from table (≠)(4 bytes length)
15	DSGTB	247	Searching data from table (>)(1 byte length)
16	DSGTW	248	Searching data from table (>)(2 bytes length)
17	DSGTD	249	Searching data from table (>)(4 bytes length)
18	DSLTB	250	Searching data from table (<)(1 byte length)
19	DSLTw	251	Searching data from table (<)(2 bytes length)
20	DSLTD	252	Searching data from table (<)(4 bytes length)
21	DSGEB	253	Searching data from table (≥)(1 byte length)
22	DSGEW	254	Searching data from table (≥)(2 bytes length)
23	DSGED	255	Searching data from table (≥)(4 bytes length)
24	DSLEB	256	Searching data from table (≤)(1 byte length)
25	DSLEW	257	Searching data from table (≤)(2 bytes length)
26	DSLED	258	Searching data from table (≤)(4 bytes length)
27	DMAXB	259	Maximum data (1 byte length)
28	DMAXW	260	Maximum data (2 bytes length)
29	DMAXD	261	Maximum data (4 bytes length)
30	DMINB	262	Minimum data (1 byte length)
31	DMINW	263	Minimum data (2 bytes length)
32	DMIND	264	Minimum data (4 bytes length)

4.6.1 TBLRB (Reading Data from Table (1 Byte Length) : SUB 233) TBLRW (Reading Data from Table (2 Bytes Length) : SUB 234) TBLRD (Reading Data from Table (4 Bytes Length) : SUB 235)

The Reading data from table instruction transfers data from a specified position in a table to another address.

The top of a table is specified in "Table top address". In "Reading position", a data position is specified relative to the top data position assumed to be 0.

In "Reading position", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Reading data from table instructions are available according to the type of data to be read from a table. In each instruction, the data in the table and data at "Transfer destination address" are of the same data type. However, the data type of "Reading position" is two-byte signed binary data at all times.

Table 4.6.1 (a) Kinds of Reading data from table instruction

	Instruction name	SUB No.	Data type
1	TBLRB	233	1 byte length data
2	TBLRW	234	2 bytes length data
3	TBLRD	235	4 bytes length data

When data 2 bytes long is read:

Number of data = 12
Table top address = D100
Reading position = D200
Transfer destination address = D300

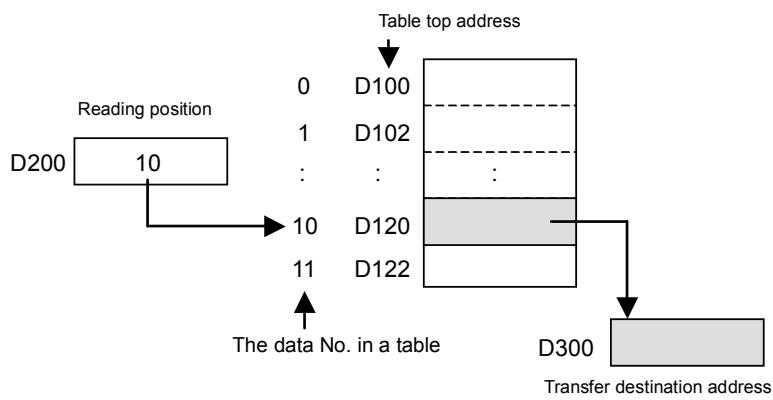


Fig. 4.6.1 (a) Example of TBLRW instruction

Format

Fig. 4.6.1(b) shows the ladder format and Table 4.6.1(b) shows the mnemonic format.

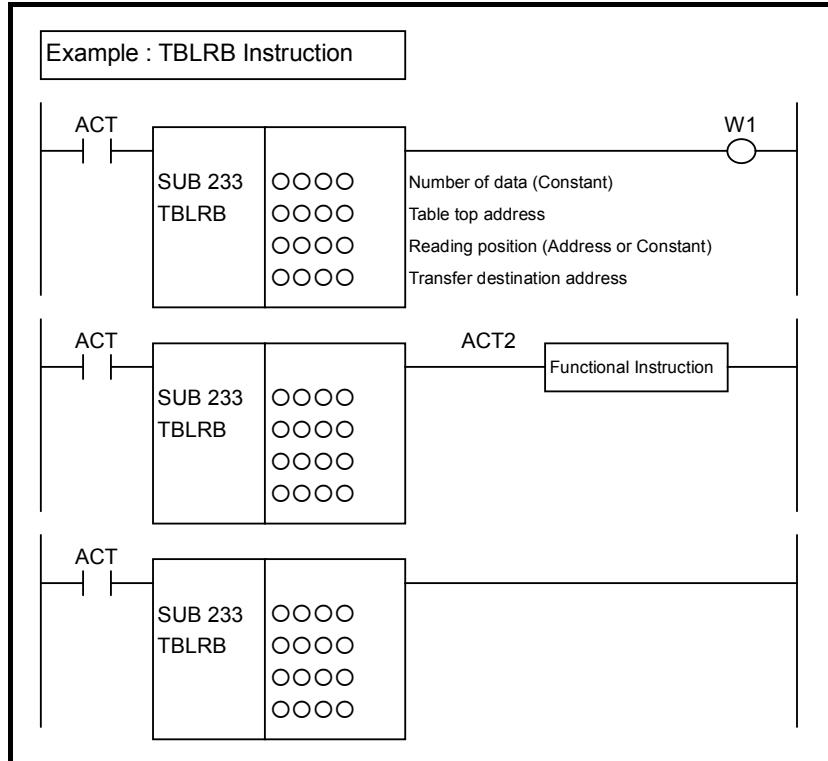


Fig. 4.6.1 (b) Format of TBLRB, TBLRW, TBLRD instruction

Table 4.6.1 (b) Mnemonic of TBLRB, TBLRW, TBLRD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	233		SUB No. (TBLRB instruction)
3	(PRM)	0000		Number of data (Constant)
4	(PRM)	0000		Table top address
5	(PRM)	0000		Reading position (Address or Constant)
6	(PRM)	0000		Transfer destination address
7	WRT	0000 .0		Normal end output

Control condition

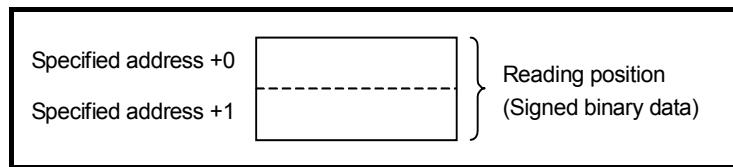
- (a) Input signal
 ACT = 0: Instruction not executed.
 ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
 - (b) Table top address
Specify the top address of a table.
 - (c) Reading position
Specify a data position relative to the top data position assumed to be 0. A value from 0 to the number of data items less 1 may be specified. If a value not within this valid range is specified, no transfer operation is performed, and W1=0 is set.

In this parameter, a constant or a PMC memory address can be specified.

If an address is specified, specify "Reading position" as signed binary data by using the contiguous two bytes of memory starting from the specified address.



- (d) Transfer destination address

Specify the destination address for the read data.

Output (W1)

W1=1: When a transfer operation is terminated normally

W1=0: When no transfer operation is executed (ACT=0)

When a value not within the valid range is specified in "Reading position"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.6.2 TBLRN (Reading Data from Table (Arbitrary Bytes Length) : SUB 236)

The Reading data from table instruction transfers data of a specified size from a specified position in a table to another address.

The top of a table is specified in "Table top address". In "Reading position", a data position is specified relative to the top data position assumed to be 0. In "Reading position", a constant or a PMC memory address for storing data can be specified.

The byte length of data to be read from the table is specified in "Data size". The data in the table and data at "Transfer destination address" are of the same data length. However, the data type of "Reading position" is two-byte signed binary data at all times.

When data 6 bytes long is read:

Number of data	= 12
Data size	= 6
Table top address	= D100
Reading position	= D200
Transfer destination address	= D300

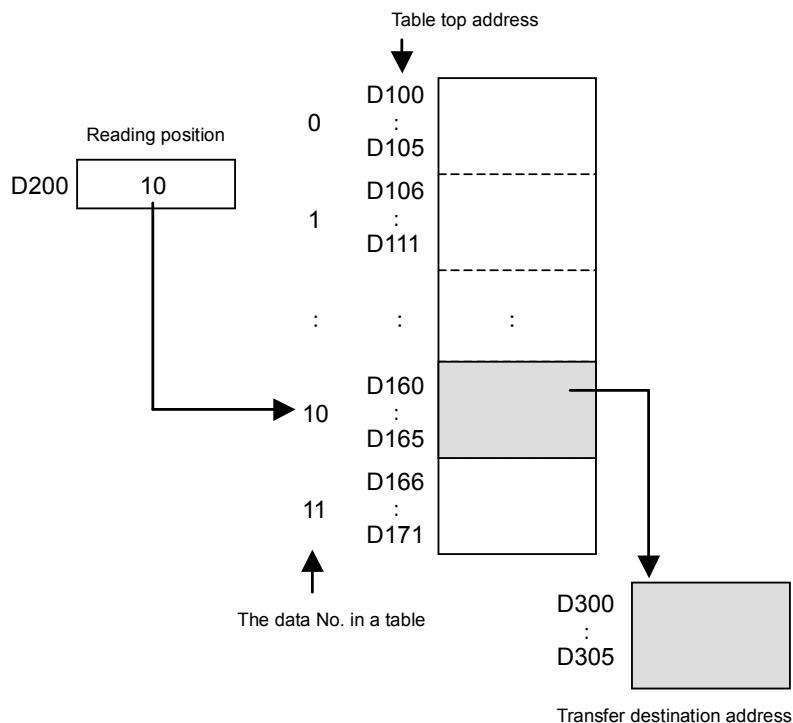


Fig. 4.6.2 (a) Example of TBLRN instruction

Format

Fig. 4.6.2(b) shows the ladder format and Table 4.6.2(a) shows the mnemonic format.

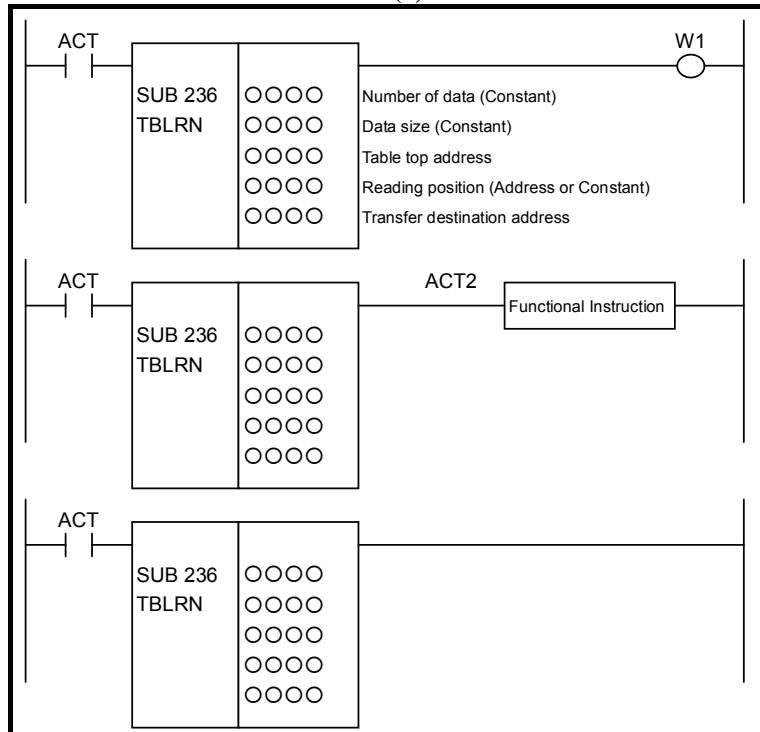


Fig. 4.6.2 (b) Format of TBLRN instruction

**Table 4.6.2 (a) Mnemonic of TBLRN instruction
Mnemonic format**

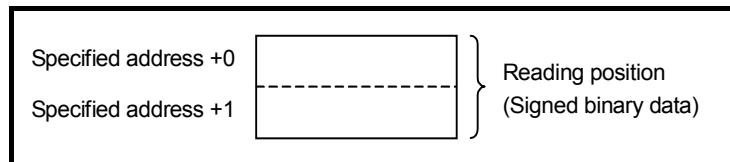
Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition			
					ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		236	SUB No. (TBLRN instruction)				
3	(PRM)	0000		Number of data (Constant)				
4	(PRM)	0000		Data size (Constant)				
5	(PRM)	0000		Table top address				
6	(PRM)	0000		Reading position (Address or Constant)				
7	(PRM)	0000		Transfer destination address				
8	WRT	0000 .0		Normal end output				W1

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
- (b) Data size
Specify the byte length of data to be read. A value from 1 to 256 may be specified.
- (c) Table top address
Specify the top address of a table.
- (d) Reading position
Specify a data position relative to the top data position assumed to be 0. A value from 0 to the number of data items less 1 may be specified. If a value not within this valid range is specified, no transfer operation is performed, and W1=0 is set.
In this parameter, a constant or a PMC memory address can be specified.
If an address is specified, specify "Reading position" as signed binary data by using the contiguous two bytes of memory starting from the specified address.



- (e) Transfer destination address
Specify the destination address for the read data.

NOTE

The operation of the instruction is not guaranteed if "Transfer destination address" overlaps the table. Specify "Transfer destination address" that does not overlap the table.

Output (W1)

- W1=1: When a transfer operation is terminated normally
- W1=0: When no transfer operation is executed (ACT=0)

When a value not within the valid range is specified in "Reading position"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.6.3 TBLWB (Writing Data to Table (1 Byte Length) : SUB 237) TBLWW (Writing Data to Table (2 Bytes Length) : SUB 238) TBLWD (Writing Data to Table (4 Bytes Length) : SUB 239)

The Writing data to table instruction writes data to a specified position in a table.

The top of a table is specified in "Table top address". In "Writing position", a data position is specified relative to the top data position assumed to be 0. In "Writing position", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Writing data to table instructions are available according to the type of data to be written to a table. In each instruction, the data in the table and transfer data are of the same data type. However, the data type of "Writing position" is two-byte signed binary data at all times.

Table 4.6.3 (a) Kinds of Writing data to table instruction

	Instruction name	SUB No.	Data type
1	TBLWB	237	1 byte length data
2	TBLWW	238	2 bytes length data
3	TBLWD	239	4 bytes length data

When data 2 bytes long is written:

Number of data = 12
Table top address = D100
Writing position = D200
Transfer data = D300

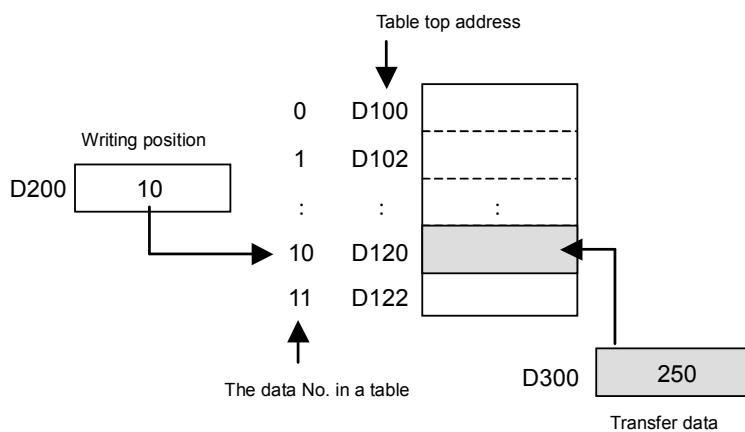


Fig. 4.6.3 (a) Example of TBLWW instruction

Format

Fig. 4.6.3(b) shows the ladder format and Table 4.6.3(b) shows the mnemonic format.

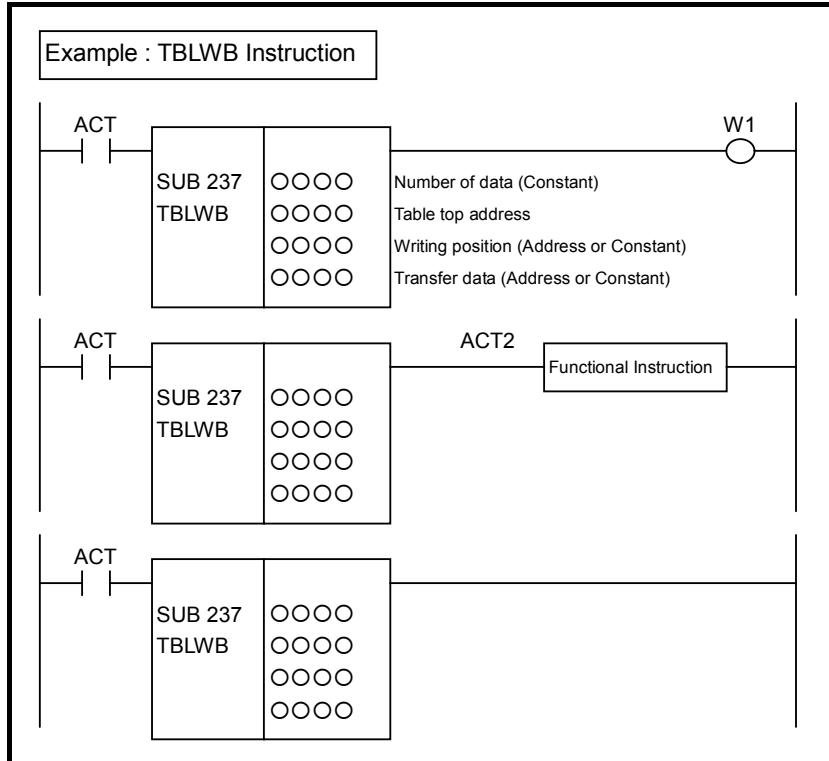


Fig. 4.6.3 (b) Format of TBLWB, TBLWW, TBLWD instruction

Table 4.6.3 (b) Mnemonic of TBLWB, TBLWW, TBLWD instruction

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0	ACT	
2	SUB	237		SUB No. (TBLWB instruction)
3	(PRM)	0000		Number of data (Constant)
4	(PRM)	0000		Table top address
5	(PRM)	0000		Writing position (Address or Constant)
6	(PRM)	0000		Transfer data (Address or Constant)
7	WRT	0000 .0		Normal end output

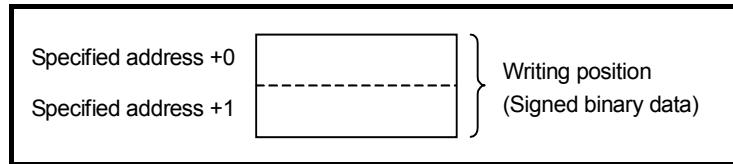
Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
 - (b) Table top address
Specify the top address of a table.
 - (c) Writing position
Specify a data position relative to the top data position assumed to be 0. A value from 0 to the number of data items less 1 may be specified. If a value not within this valid range is specified, no transfer operation is performed, and W1=0 is set.

In this parameter, a constant or a PMC memory address can be specified. If an address is specified, specify "Writing position" as signed binary data by using the contiguous two bytes of memory starting from the specified address.



(d) Transfer data

Specify data to be written. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available value
TBLWB	-128 to 127
TBLWW	-32768 to 32767
TBLWD	-2147483648 to 2147483647

Output (W1)

W1=1: When a transfer operation is terminated normally

W1=0: When no transfer operation is executed (ACT=0)

When a value not within the valid range is specified in "Writing position"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.6.4 TBLWN (Writing Data to Table (Arbitrary Bytes Length) : SUB 240)

The Writing data to table instruction writes data of a specified size to a specified position in a table.

The top of a table is specified in "Table top address". In "Writing position", a data position is specified relative to the top data position assumed to be 0. In "Writing position", a constant or a PMC memory address for storing data can be specified.

The byte length of data to be written to the table is specified in "Data size". The data in the table and data at "Transfer data top address" are of the same data length. However, the data type of "Writing position" is two-byte signed binary data at all times.

When data 6 bytes long is written:

Number of data = 12
 Data size = 6
 Table top address = D100
 Writing position = D200
 Transfer data top address = D300

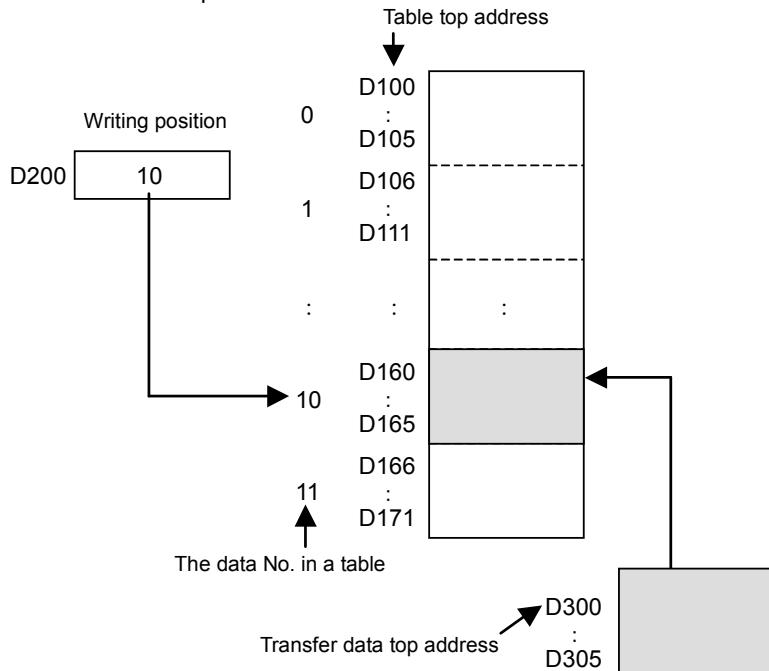


Fig. 4.6.4 (a) Example of TBLWN instruction

Format

Fig. 4.6.4(b) shows the ladder format and Table 4.6.4 shows the mnemonic format.

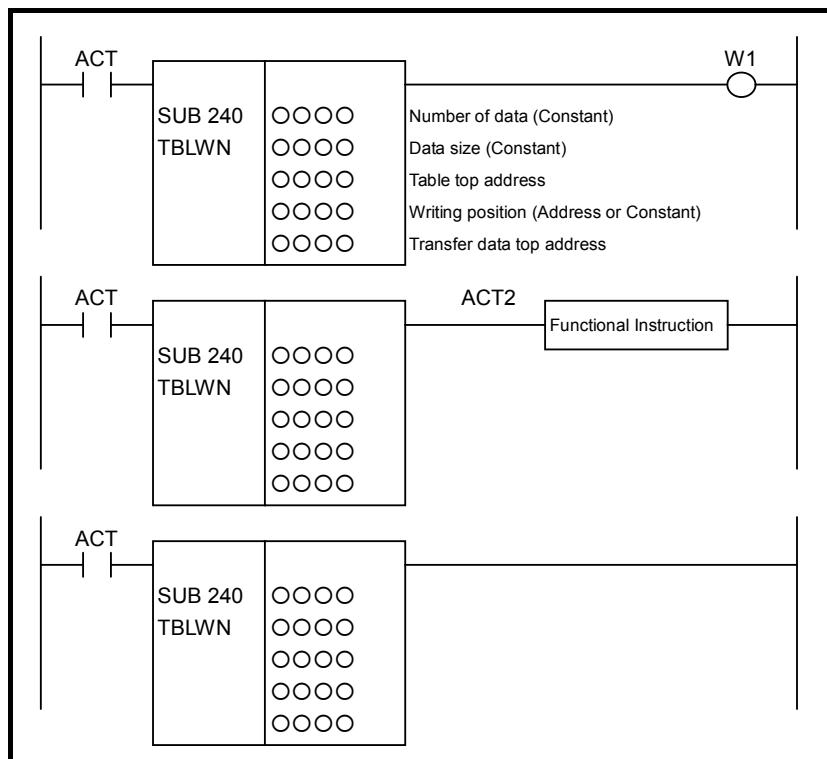


Fig. 4.6.4 (b) Format of TBLWN instruction

**Table 4.6.4 Mnemonic of TBLWN instruction
Mnemonic format**

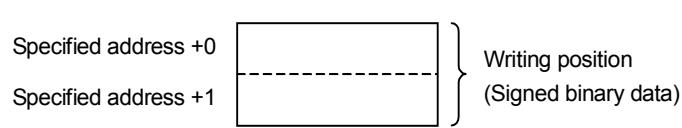
Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
ST3	ST2	ST1	ST0		
1	RD	0000 .0		ACT	
2	SUB		240	SUB No. (TBLWN instruction)	
3	(PRM)	0000		Number of data (Constant)	
4	(PRM)	0000		Data size (Constant)	
5	(PRM)	0000		Table top address	
6	(PRM)	0000		Writing position (Address or Constant)	
7	(PRM)	0000		Transfer data top address	
8	WRT	0000 .0		Normal end output	W1

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
- (b) Data size
Specify the byte length of data to be written. A value from 1 to 256 may be specified.
- (c) Table top address
Specify the top address of a table.
- (d) Writing position
Specify a data position relative to the top data position assumed to be 0. A value from 0 to the number of data items less 1 may be specified. If a value not within this valid range is specified, no transfer operation is performed, and W1=0 is set.
In this parameter, a constant or a PMC memory address can be specified.
If an address is specified, specify "Writing position" as signed binary data by using the contiguous two bytes of memory starting from the specified address.



- (e) Transfer data top address
Specify the start address of data to be written.

NOTE

The operation of the instruction is not guaranteed if "Transfer data top address" overlaps the table. Specify "Transfer data top address" that does not overlap the table.

Output (W1)

W1=1: When a transfer operation is terminated normally

W1=0: When no transfer operation is executed (ACT=0)

When a value not within the valid range is specified in "Writing position"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

- 4.6.5 DSEQB (Searching Data from Table(=)(1 Byte Length):SUB 241)
DSEQW (Searching Data from Table(=)(2 Bytes Length):SUB 242)
DSEQD (Searching Data from Table(=)(4 Bytes Length):SUB 243)
DSNEB (Searching Data from Table(≠)(1 Byte Length):SUB 244)
DSNEW (Searching Data from Table(≠)(2 Bytes Length):SUB 245)
DSNED (Searching Data from Table(≠)(4 Bytes Length):SUB 246)
DSGTB (Searching Data from Table(>)(1 Byte Length):SUB 247)
DSGTw (Searching Data from Table(>)(2 Bytes Length):SUB 248)
DSGTD (Searching Data from Table(>)(4 Bytes Length):SUB 249)
DSLTB (Searching Data from Table(<)(1 Byte Length):SUB 250)
DSLTw (Searching Data from Table(<)(2 Bytes Length):SUB 251)
DSLTD (Searching Data from Table(<)(4 Bytes Length):SUB 252)
DSGEB (Searching Data from Table(≥)(1 Byte Length):SUB 253)
DSGEW (Searching Data from Table(≥)(2 Bytes Length):SUB 254)
DSGED (Searching Data from Table(≥)(4 Bytes Length) :SUB 255)
DSLEB (Searching Data from Table(≤)(1 Byte Length) :SUB 256)
DSLEW (Searching Data from Table(≤)(2 Bytes Length) :SUB 257)
DSLED (Searching Data from Table(≤)(4 Bytes Length) :SUB 258)**

The instruction searches a table for data that satisfies a specified condition and acquires the position of found data.

As indicated below, eighteen types of Searching data from table instructions are available according to the search condition and data type. In each instruction, the data in the table and "Search data" are of the same data type. However, the data type of "Search starting position" and "Find position output address" is two-byte signed binary data at all times.

Table 4.6.5 (a) Kinds of Searching data from table instruction

	Instruction name	SUB No.	Search condition	Data type
1	DSEQB	241	=	1 byte length signed binary data
2	DSEQW	242		2 bytes length signed binary data
3	DSEQD	243		4 bytes length signed binary data
4	DSNEB	244	≠	1 byte length signed binary data
5	DSNEW	245		2 bytes length signed binary data
6	DSNED	246		4 bytes length signed binary data
7	DSGTB	247	>	1 byte length signed binary data
8	DSGTW	248		2 bytes length signed binary data
9	DSGTD	249		4 bytes length signed binary data
10	DSLTB	250	<	1 byte length signed binary data
11	DSLTw	251		2 bytes length signed binary data
12	DSLTD	252		4 bytes length signed binary data
13	DSGEB	253	≥	1 byte length signed binary data
14	DSGEW	254		2 bytes length signed binary data
15	DSGED	255		4 bytes length signed binary data
16	DSLEB	256	≤	1 byte length signed binary data
17	DSLEW	257		2 bytes length signed binary data
18	DSLED	258		4 bytes length signed binary data

Table 4.6.5 (b) Concurrence conditions of search data

Instruction	Search condition	Concurrence conditions
DSEQx	=	Table data = search data
DSNEx	≠	Table data ≠ search data
DSGTx	>	Table data > search data
DSLTx	<	Table data < search data
DSGEx	≥	Table data ≥ search data
DSLEx	≤	Table data ≤ search data

The top of a table is specified in "Table top address". In "Search starting position", a data position is specified relative to the top data position assumed to be 0. A value output to "Find position output address" is also indicated as a data position relative to the top data position assumed to be 0. In "Search starting position", a constant or a PMC memory address for storing data can be specified.

If a value not within the valid range is specified in "Search starting position", -1 is output to "Find position output address", and W1=0 is set.

Moreover, if data that satisfies a specified condition is not found in the area from "Search starting position" to the end of the table as a result of search operation, -1 is output to "Find position output address", and W1=0 is set.

When data larger than "Search data" is to be found:

Number of data = 20
 Table top address = D100
 Search starting position = D200
 Search data = D300
 Find position output address = D400

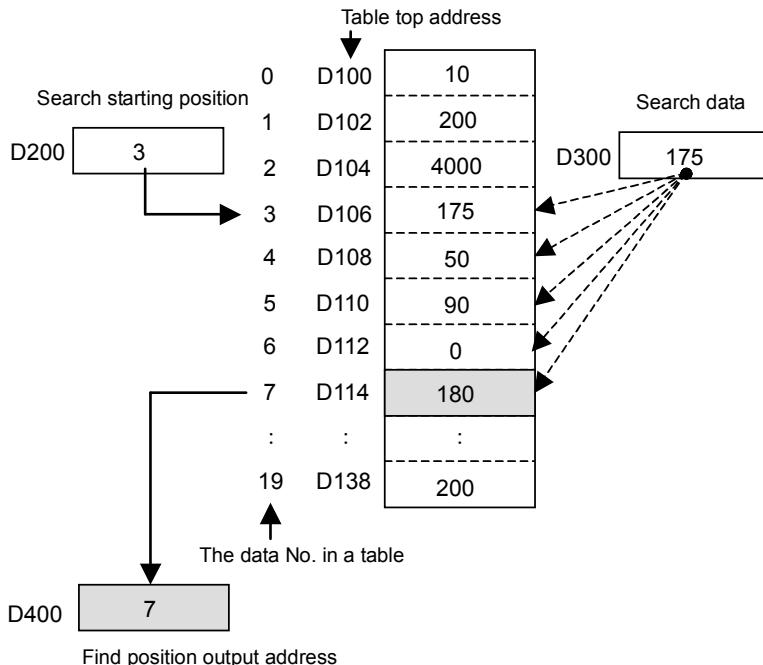


Fig. 4.6.5 (a) Example of DSGTW instruction

Format

Fig. 4.6.5(b) shows the ladder format and Table 4.6.5(c) shows the mnemonic format.

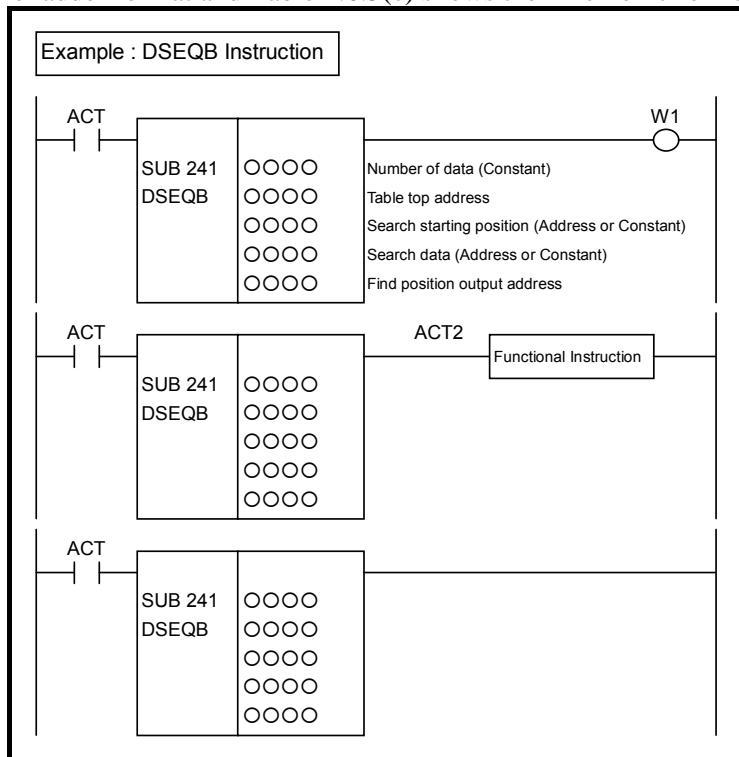


Fig. 4.6.5 (b) Format of DSEQx, DSNEEx, DSGTx, DSLTx, DSGEx, DSLEX instruction

Table 4.6.5 (c) Mnemonic of DSEQx, DSNEEx, DSGTx, DSLTx, DSGEx, DSLEEx instruction
Mnemonic format **Memory status of control condition**

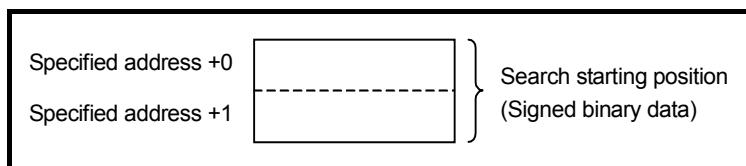
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		241	SUB No. (DSEQB instruction)				
3	(PRM)	0000		Number of data (Constant)				
4	(PRM)	0000		Table top address				
5	(PRM)	0000		Search starting position (Address or Constant)				
6	(PRM)	0000		Search data (Address or Constant)				
7	(PRM)	0000		Find position output address				
8	WRT	0000 .0		Result output				W1

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
- (b) Table top address
Specify the top address of a table.
- (c) Search starting position
Specify a search start data position relative to the top data position assumed to be 0. A value from 0 to the number of data items less 1 may be specified. If a value not within this valid range is specified, no search operation is performed, -1 is output to "Find position output address", and W1=0 is set.
In this parameter, a constant or a PMC memory address can be specified.
If an address is specified, specify "Search starting position" as signed binary data by using the contiguous two bytes of memory starting from the specified address.



- (d) Search data
Specify a value to be compared with in search operation. A comparison is made with this data according to the search condition of each instruction, and the position of data that satisfies the search condition is acquired.
In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

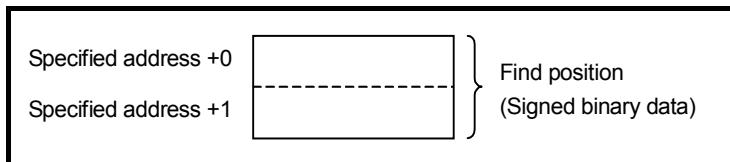
Instruction name	Available value
DSEQB DSNEB DSGTB DSLTB DSGTB DSLEB	-128 to 127
DSEQW DSNEW DSGTW DSLTW DSGTW DSLEW	-32768 to 32767
DSEQD DSNED DSGTD DSLTD DSGTD DSLED	-2147483648 to 2147483647

(e) Find position output address

Specify the address for outputting data that satisfies the specified condition as a result of search.

A find position is output as two-byte signed binary data.

If no data satisfies the specified condition, -1 is output, and W1=0 is set.

**Output (W1)**

W1=1: When data that satisfies a specified condition is found

W1=0: When no search operation is executed (ACT=0)

When data that satisfies a specified condition is not found in the area from "Search starting position" to the end of the table

When a value not within the valid range is set in "Search starting position"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.6.6 DMAXB (Maximum Data (1 Byte Length): SUB 259) DMAXW (Maximum Data (2 Bytes Length) : SUB 260) DMAXD (Maximum Data (4 Bytes Length) : SUB 261)

The Maximum data instruction searches a table for maximum data and acquires the value and position of found maximum data.

The top of a table is specified in "Table top address". A value output to "Find position output address" is indicated as a data position relative to the top data position assumed to be 0.

A search is made starting at the top of a table. In "Number of search data", the number of data items to be searched in the area from the top of a table to a desired search position is specified.

As indicated below, three types of Maximum data instructions are available according to the data type of a table to be searched. In each instruction, the data in the table and data at "Maximum data output address" are of the same data type. However, the data type of "Number of search data" and "Find position output address" is two-byte signed binary data at all times.

Table4.6.6 (a) Kinds of Maximum data instruction

	Instruction name	SUB No.	Data type
1	DMAXB	259	1 byte length signed data
2	DMAXW	260	2 bytes length signed data
3	DMAXD	261	4 bytes length signed data

When a table is searched for maximum data and its position:

Number of data = 20
 Table top address = D100
 Number of search data = D400
 Maximum data output address = D200
 Find position output address = D300

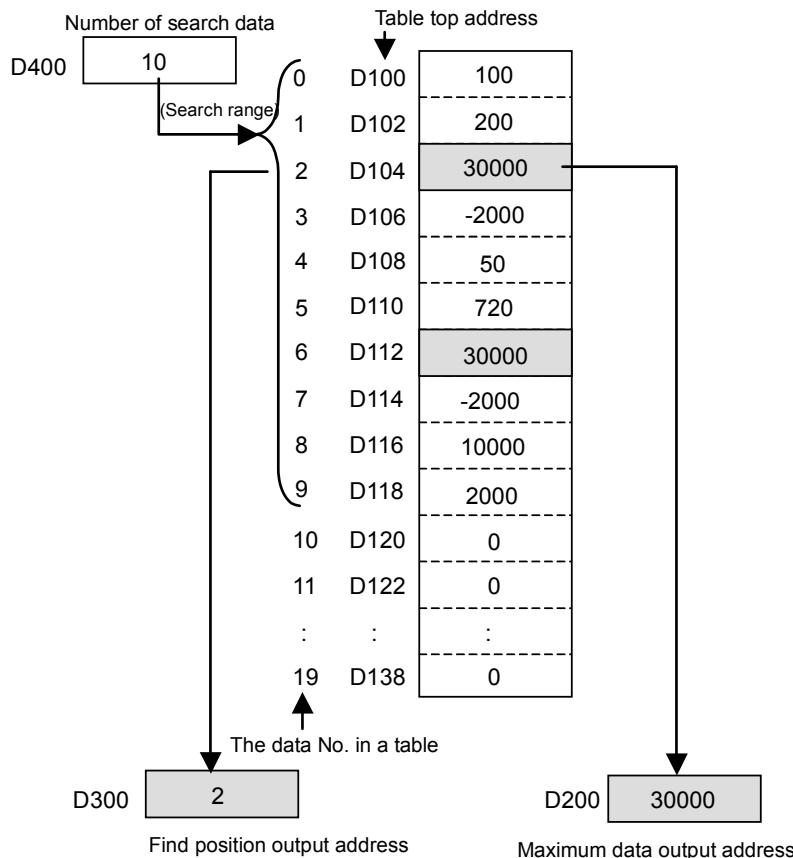


Fig. 4.6.6 (a) Example of DMAXW instruction

Format

Fig. 4.6.6(b) shows the ladder format and Table 4.6.6(b) shows the mnemonic format.

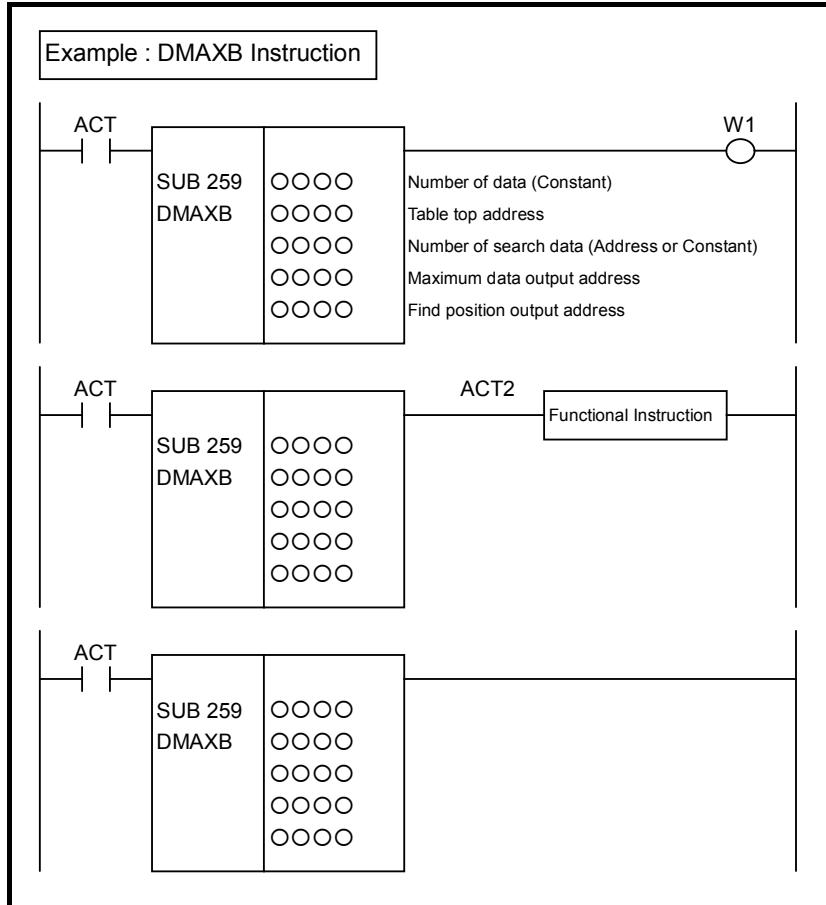


Fig. 4.6.6 (b) Format of DMAXB, DMAXW, DMAXD instruction

Table 4.6.6 (b) Mnemonic of DMAXB, DMAXW, DMAXD instruction

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	259		SUB No. (DMAXB instruction)
3	(PRM)	0000		Number of data (Constant)
4	(PRM)	0000		Table top address
5	(PRM)	0000		Number of search data (Address or Constant)
6	(PRM)	0000		Maximum data output address
7	(PRM)	0000		Find position output address
8	WRT	0000 .0		Normal end output

Control condition

- (a) Input signal
 ACT = 0: Instruction not executed.
 ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
This parameter indicates the total number of data items of a table. A data range to be searched is specified using the "Number of search data" parameter.

(b) Table top address

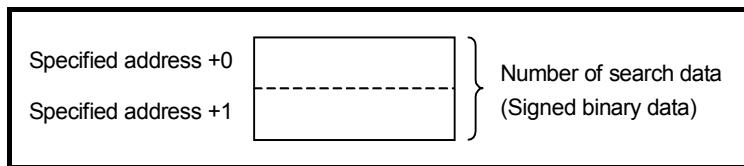
Specify the top address of a table.

(c) Number of search data

Specify the number of data items to be searched for maximum data in a table. A value from 1 to the value specified in the "Number of data" parameter may be specified. If a value not within this valid range is specified, 0 is output to "Maximum data output address", -1 is output to "Find position output address", and W1=0 is set.

In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified, specify a search range as signed binary data by using the contiguous two bytes of memory starting from the specified address.



(d) Maximum data output address

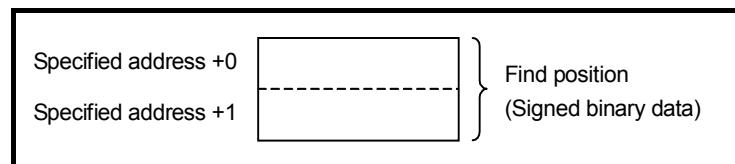
Specify the address to which maximum data is to be output as a result of search operation.

(e) Find position output address

Specify the address to which the position of maximum data is to be output as a result of search operation.

A find position is output as two-byte signed binary data.

As data position information, the top of the table is indicated as 0, and the end of the table is indicated as the number of data items less 1. If multiple maximum data items are found, the position nearest to the top of the table is output.

**Output (W1)**

W1=1: When a search operation is terminated normally

W1=0: When no search operation is executed (ACT=0)

When, a value not within the valid range is set in "Number of search data"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.6.7 DMINB (Minimum Data (1 Byte Length): SUB 262)**DMINW (Minimum Data (2 Bytes Length): SUB 263)****DMIND (Minimum Data (4 Bytes Length): SUB 264)**

The Minimum data instruction searches a table for minimum data and acquires the value and position of found minimum data.

The top of a table is specified in "Table top address". A value output to "Find position output address" is indicated as a data position relative to the top data position assumed to be 0.

A search is made starting at the top of a table. In "Number of search data", the number of data items to be searched in the area from the top of a table to a desired search position is specified.

As indicated below, three types of Minimum data instructions are available according to the data type of a table to be searched. In each instruction, the data in the table and data at "Minimum data output address" are of the same data type. However, the data type of "Number of search data" and "Find position output address" is two-byte signed binary data at all times.

Table 4.6.7 (a) Kinds of Minimum data instruction

	Instruction name	SUB No.	Data type
1	DMINB	262	1 byte length signed data
2	DMINW	263	2 bytes length signed data
3	DMIND	264	4 bytes length signed data

When a table is searched for minimum data and its position:

Number of data = 20
 Table top address = D100
 Number of search data = D400
 Minimum data output address = D200
 Find position output address = D300

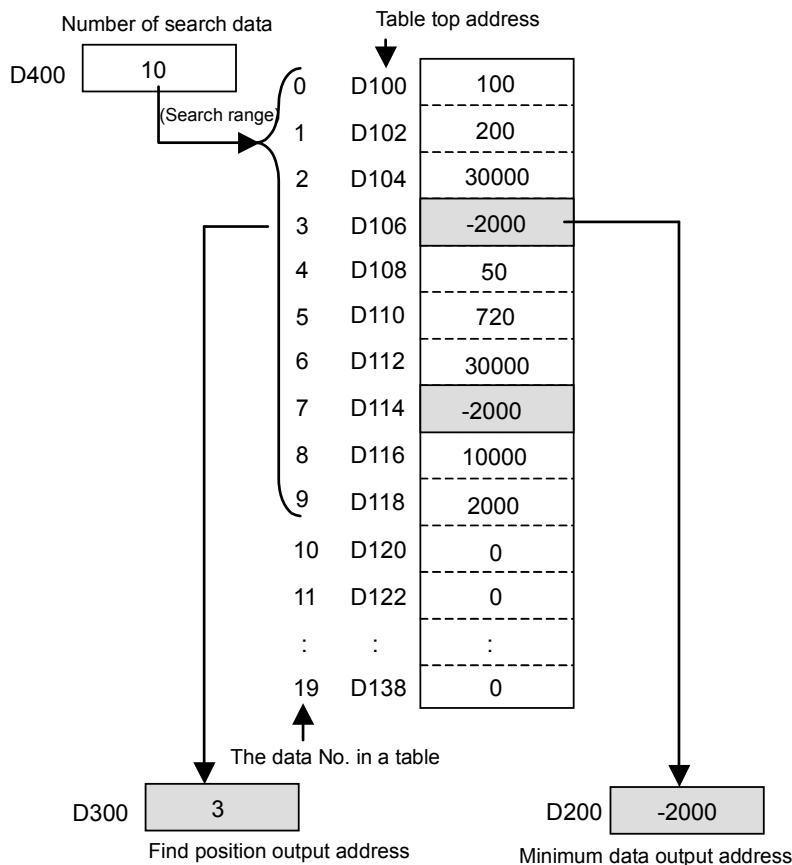


Fig. 4.6.7 (a) Example of DMINW instruction

Format

Fig. 4.6.7(b) shows the ladder format and Table 4.6.7(b) shows the mnemonic format.

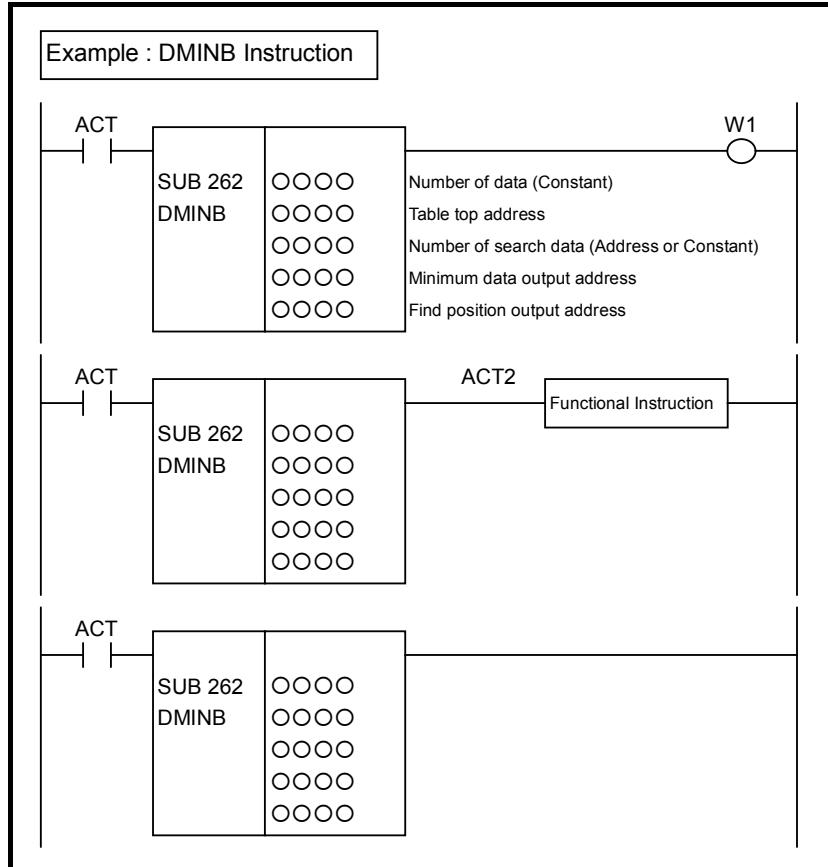


Fig. 4.6.7 (b) Format of DMINB, DMINW, DMIND instruction

Table 4.6.7 (b) Mnemonic of DMINB, DMINW, DMIND instruction

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	262		SUB No. (DMINB instruction)
3	(PRM)	0000		Number of data (Constant)
4	(PRM)	0000		Table top address
5	(PRM)	0000		Number of search data (Address or Constant)
6	(PRM)	0000		Minimum data output address
7	(PRM)	0000		Find position output address
8	WRT	0000 .0		Normal end output

Control condition

- (a) Input signal
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Number of data
Specify the number of data items in a table. Ensure that the entire table is within the valid address range.
This parameter indicates the total number of data items of a table. A data range to be searched is specified using the "Number of search data" parameter.

(b) Table top address

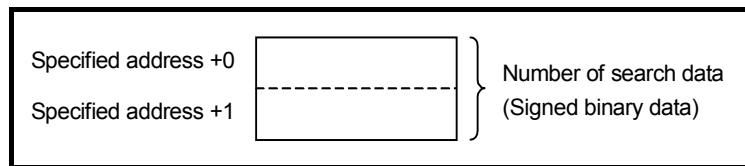
Specify the top address of a table.

(c) Number of search data

Specify the number of data items to be searched for minimum data in a table. A value from 1 to the value specified in the "Number of data" parameter may be specified. If an invalid value is specified, 0 is output to "Minimum data output address", -1 is output to "Find position output address", and W1=0 is set.

In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified, specify a search range as signed binary data by using the contiguous two bytes of memory starting from the specified address.



(d) Minimum data output address

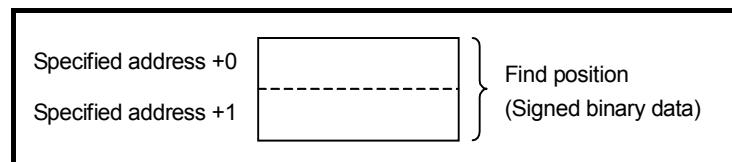
Specify the address to which minimum data is to be output as a result of search operation.

(e) Find position output address

Specify the address to which the position of minimum data is to be output as a result of search operation.

A find position is output as two-byte signed binary data.

As data position information, the top of the table is indicated as 0, and the end of the table is indicated as the number of data items less 1. If multiple minimum data items are found, the position nearest to the top of the table is output.

**Output (W1)**

W1=1: When a search operation is terminated normally

W1=0: When no search operation is executed (ACT=0)

When a value not within the valid range is set in "Number of search data"

NOTE

W1 may be omitted. Moreover, another functional instruction can be connected instead of a coil.

4.7 COMPARISON

The following types of comparison instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	EQB	200	1 byte Binary comparison (equal) (*1)
2	EQW	201	2 byte Binary comparison (equal) (*1)
3	EQD	202	4 byte Binary comparison (equal) (*1)

	Instruction name	Sub number	Processing
4	NEB	203	1 byte Binary comparison (not equal) (*1)
5	NEW	204	2 byte Binary comparison (not equal) (*1)
6	NED	205	4 byte Binary comparison (not equal) (*1)
7	GTB	206	1 byte Binary comparison (greater than) (*1)
8	GTW	207	2 byte Binary comparison (greater than) (*1)
9	GTD	208	4 byte Binary comparison (greater than) (*1)
10	LTB	209	1 byte Binary comparison (less than) (*1)
11	LTW	210	2 byte Binary comparison (less than) (*1)
12	LTD	211	4 byte Binary comparison (less than) (*1)
13	GEB	212	1 byte Binary comparison (greater or equal) (*1)
14	GEW	213	2 byte Binary comparison (greater or equal) (*1)
15	GED	214	4 byte Binary comparison (greater or equal) (*1)
16	LEB	215	1 byte Binary comparison (less or equal) (*1)
17	LEW	216	2 byte Binary comparison (less or equal) (*1)
18	LED	217	4 byte Binary comparison (less or equal) (*1)
19	RNGB	218	1 byte Binary comparison (range) (*1)
20	RNGW	219	2 byte Binary comparison (range) (*1)
21	RNGD	220	4 byte Binary comparison (range) (*1)
22	COMPB	32	Comparison between binary data
23	COMP	15	Comparison
24	COIN	16	Coincidence check

NOTE

- 1 You can set either constant or address to each parameter for the "(*1)" marked instruction. When you input a number to its parameter on LADDER editing screen, the input is recognized as a constant parameter. When you input a symbol that is composed of digits and that may be considered as a number, the input is recognized as a number and treated as a constant parameter too. If you want to set such address that has a confusing symbol, you have to input the address, not the symbol, to the parameter.

4.7.1 Signed Binary Comparison (=) EQB (1 Byte Length: SUB 200) EQW (2 Bytes Length: SUB 201) EQD (4 Bytes Length: SUB 202)

Using this instruction, you can know whether the "Data 1" equals to the "Data 2" or not.

The EQB instruction handles 1 byte length signed binary data.

The EQW instruction handles 2 bytes length signed binary data.

The EQD instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.1 shows the ladder format and Table 4.7.1 shows the mnemonic format.

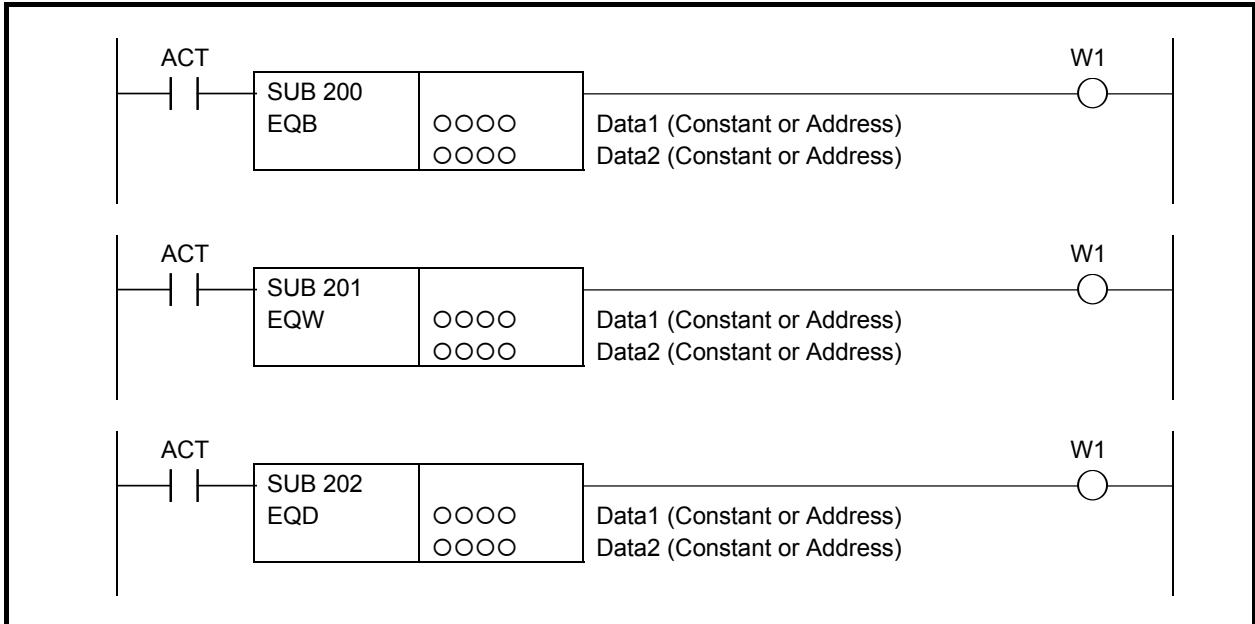


Fig. 4.7.1 Format of EQB, EQW and EQD instructions

Table 4.7.1 Mnemonic of EQB, EQW and EQD instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		200 201 202	EQB instruction EQW instruction EQD instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				▼
5	WRT	0000 .0		Result				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

(a) Data 1

(b) Data 2

You can specify the constant or any address. The valid data range is shown below.

EQB: -128 to 127

EQW: -32768 to 32767

EQD: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1" = "Data 2"

W1=0: - When ACT=0

- When ACT=1 and "Data 1" ≠ "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.2 Signed Binary Comparison (\neq) NEB (1 Byte Length: SUB 203) NEW (2 Bytes Length: SUB 204) NED (4 Bytes Length: SUB 205)

Using this instruction, you can know whether the "Data 1" does not equal to the "Data 2" or not.

The NEB instruction handles 1 byte length signed binary data.

The NEW instruction handles 2 bytes length signed binary data.

The NED instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.2 shows the ladder format and Table 4.7.2 shows the mnemonic format.

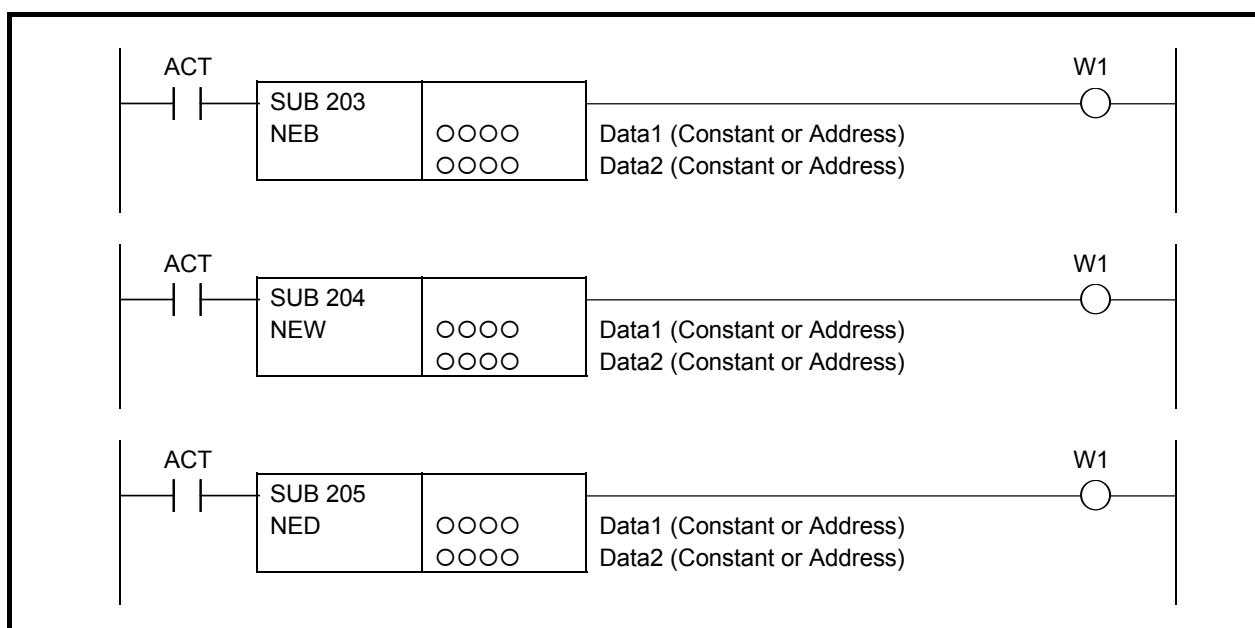


Fig. 4.7.2 Format of NEB, NEW and NED instructions

Table 4.7.2 Mnemonic of NEB, NEW and NED instructions

Step number	Instruction	Mnemonic format		Memory status of control condition
		Address No.	Bit No.	
1	RD	0000 .O	ACT	
2	SUB		203 204 205	NEB instruction NEW instruction NED instruction
3	(PRM)	0000		Data1 (Constant or Address)
4	(PRM)	0000		Data2 (Constant or Address)
5	WRT	0000 .O		Result
				ST3 ST2 ST1 ST0
				ACT
				▼
				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

- (a) Data 1
- (b) Data 2

You can specify the constant or any address. The valid data range is shown below.

NEB: -128 to 127

NEW: -32768 to 32767

NED: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1" ≠ "Data 2"

W1=0: - When ACT=0

- When ACT=1 and "Data 1" = "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.3 Signed Binary Comparison (>) GTB (1 Byte Length: SUB 206) GTW (2 Bytes Length: SUB 207) GTD (4 Bytes Length: SUB 208)

Using this instruction, you can know whether the "Data 1" is greater than the "Data 2" or not.

The GTB instruction handles 1 byte length signed binary data.

The GTW instruction handles 2 bytes length signed binary data.

The GTD instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.3 shows the ladder format and Table 4.7.3 shows the mnemonic format.

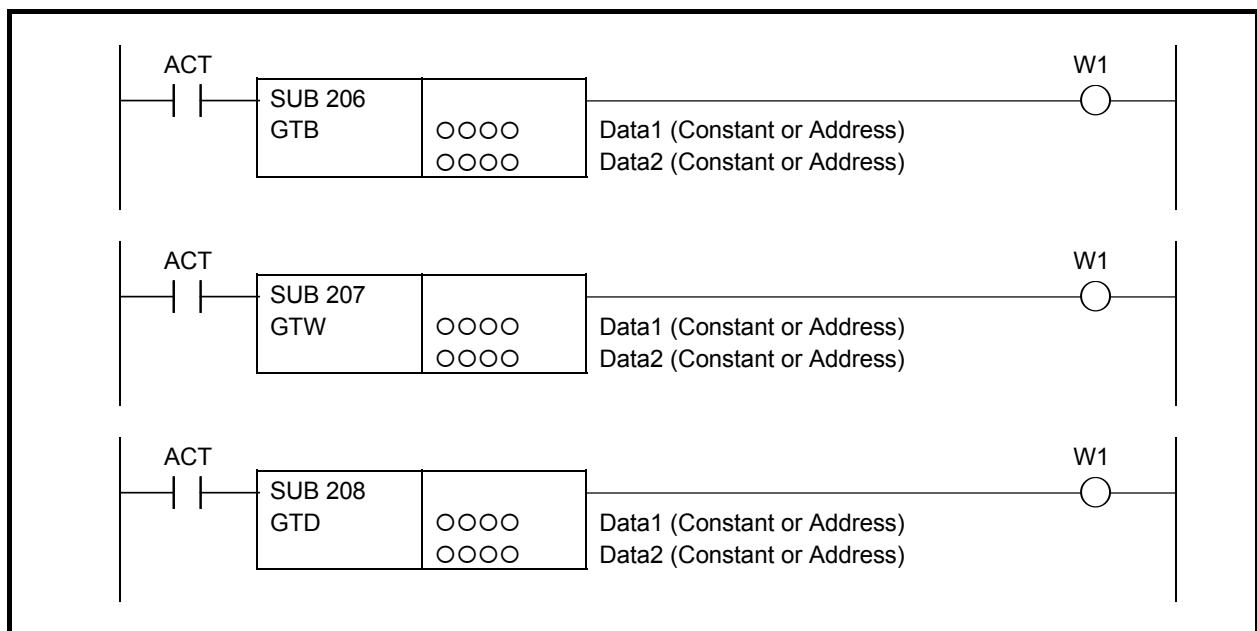


Fig. 4.7.3 Format of GTB, GTW and GTD instructions

Table 4.7.3 Mnemonic of GTB, GTW and GTD instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		206 207 208	GTB instruction GTW instruction GTD instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				▼
5	WRT	0000 .O		Result				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

(a) Data 1

(b) Data 2

You can specify the constant or any address. The valid data range is shown below.

GTB: -128 to 127

GTW: -32768 to 32767

GTD: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1" > "Data 2"

W1=0: - When ACT=0 "

- When ACT=1 and "Data 1" \leq "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.4 Signed Binary Comparison (< LTB (1 Byte Length: SUB 209) LTW (2 Bytes Length: SUB 210) LTD (4 Bytes Length: SUB 211)

Using this instruction, you can know whether the "Data 1" is smaller than the "Data 2" or not.

The LTB instruction handles 1 byte length signed binary data.

The LTW instruction handles 2 bytes length signed binary data.

The LTD instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.4 shows the ladder format and Table 4.7.4 shows the mnemonic format.

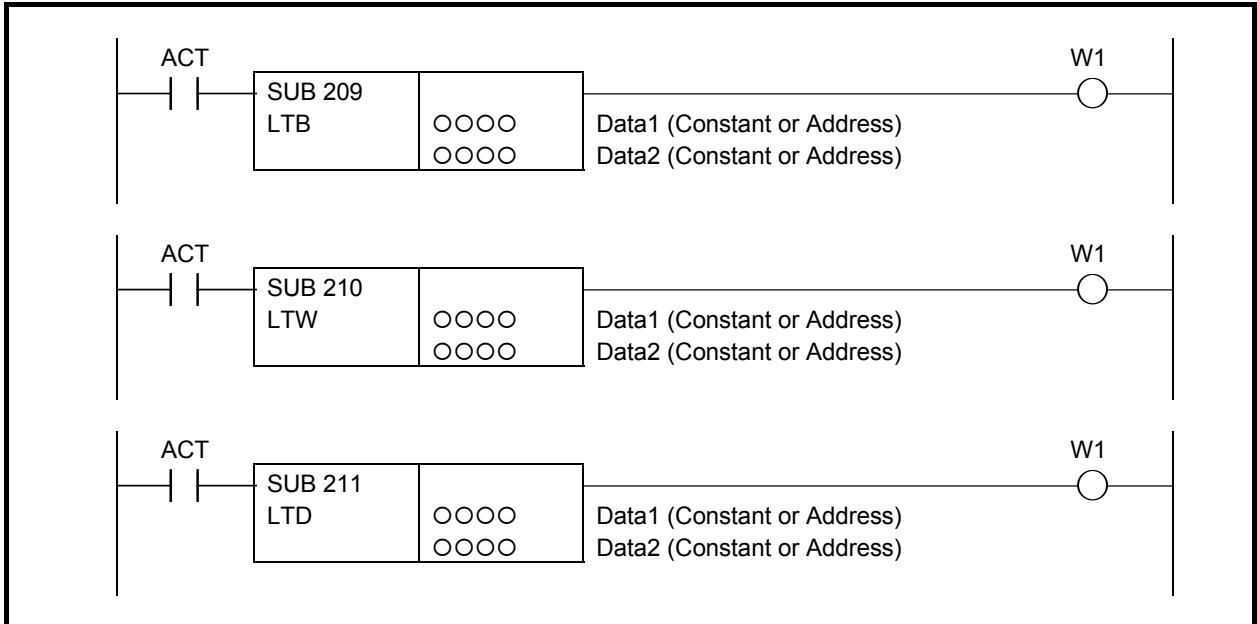


Fig. 4.7.4 Format of LTB, LTW and LTD instructions

Table 4.7.4 Mnemonic of LTB, LTW and LTD instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		209 210 211	LTB instruction LTW instruction LTD instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				▼
5	WRT	0000 .0		Result				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

(a) Data 1

(b) Data 2

You can specify the constant or any address. The valid data range is shown below.

LTB: -128 to 127

LTW: -32768 to 32767

LTD: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1" < "Data 2"

W1=0: - When ACT=0

- When ACT=1 and "Data 1" \geq "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.5 Signed Binary Comparison (\geq) GEB (1 Byte Length: SUB 212) GEW (2 Bytes Length: SUB 213) GED (4 Bytes Length: SUB 214)

Using this instruction, you can know whether the "Data 1" is equal or greater than the "Data 2" or not.

The GEB instruction handles 1 byte length signed binary data.

The GEW instruction handles 2 bytes length signed binary data.

The GED instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.5 shows the ladder format and Table 4.7.5 shows the mnemonic format.

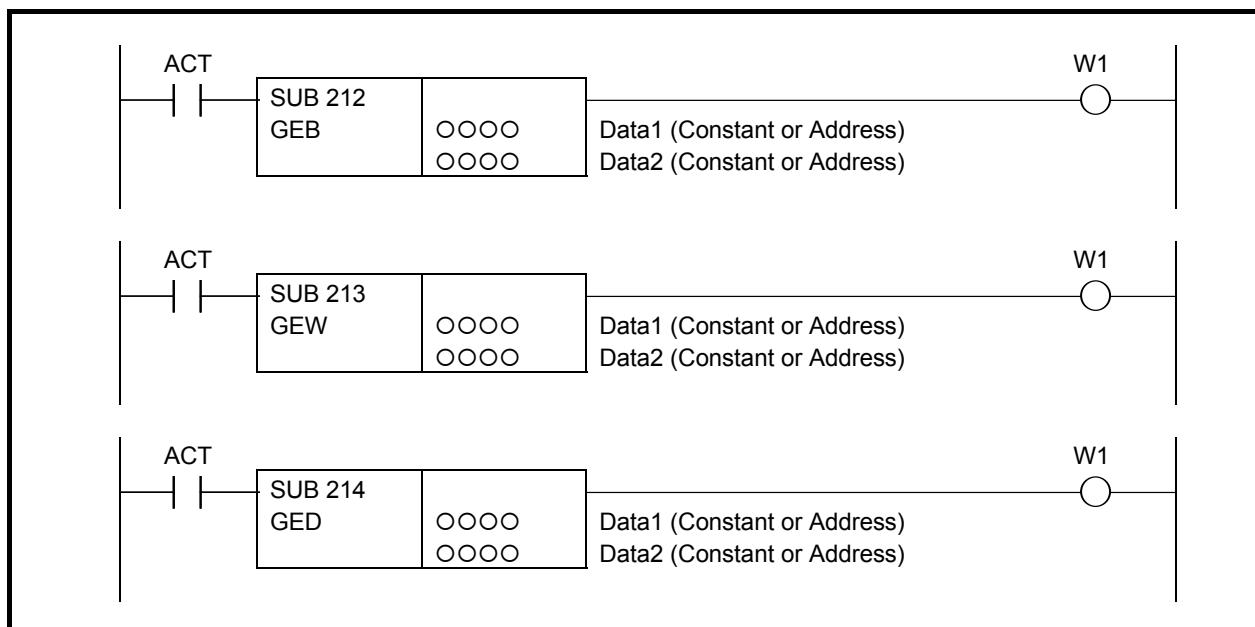


Fig. 4.7.5 Format of GEB, GEW and GED instructions

Table 4.7.5 Mnemonic of GEB, GEW and GED instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		212 213 214	GEB instruction GEW instruction GED instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				
5	WRT	0000 .0		Result				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

- (a) Data 1
- (b) Data 2

You can specify the constant or any address. The valid data range is shown below.

GEB: -128 to 127

GEW: -32768 to 32767

GED: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1" \geq "Data 2"

W1=0: - When ACT=0

- When ACT=1 and "Data 1" < "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.6 Signed Binary Comparison (\leq)

LEB (1 Byte Length: SUB 215)

LEW (2 Bytes Length: SUB 216)

LED (4 Bytes Length: SUB 217)

Using this instruction, you can know whether the "Data 1" is equal or smaller than the "Data 2" or not.

The LEB instruction handles 1 byte length signed binary data.

The LEW instruction handles 2 bytes length signed binary data.

The LED instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.6 shows the ladder format and Table 4.7.6 shows the mnemonic format.

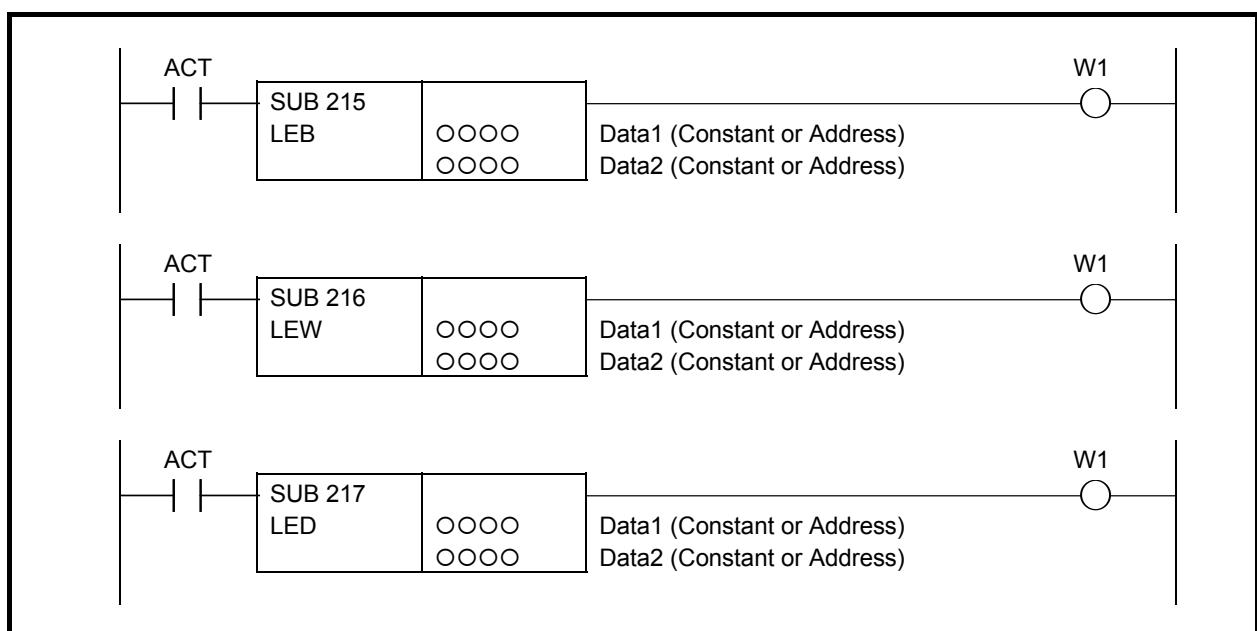


Fig. 4.7.6 Format of LEB, LEW and LED instructions

Table 4.7.6 Mnemonic of LEB, LEW and LED instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		215 216 217	LEB instruction LEW instruction LED instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				▼
5	WRT	0000 .O		Result				W1

Control condition

- (a) Command (ACT)
 ACT=0: Do not execute the instruction. The W1 becomes 0.
 ACT=1: Execute the instruction. The result is output to W1.

Parameters

- (a) Data 1
 (b) Data 2
 You can specify the constant or any address. The valid data range is shown below.
 LEB: -128 to 127
 LEW: -32768 to 32767
 LED: -2147483648 to 2147483647

Output (W1)

- The result is output to W1.
 W1=1: - When ACT=1 and "Data 1" \leq "Data 2"
 W1=0: - When ACT=0
 - When ACT=1 and "Data 1" > "Data2"

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.7 Signed Binary Comparison (Range)

RNGB (1 Byte Length: SUB 218)

RNGW (2 Bytes Length: SUB 219)

RNGD (4 Bytes Length: SUB 220)

This instruction is the range comparison function. When following data conditions, the output W1 becomes 1.

- "Data 1" \leq "Input data" \leq "Data 2" or
- "Data 2" \leq "Input data" \leq "Data 1"

The RNGB instruction handles 1 byte length signed binary data.

The RNGW instruction handles 2 bytes length signed binary data.

The RNGD instruction handles 4 bytes length signed binary data.

Format

Fig. 4.7.7 shows the ladder format and Table 4.7.7 shows the mnemonic format.

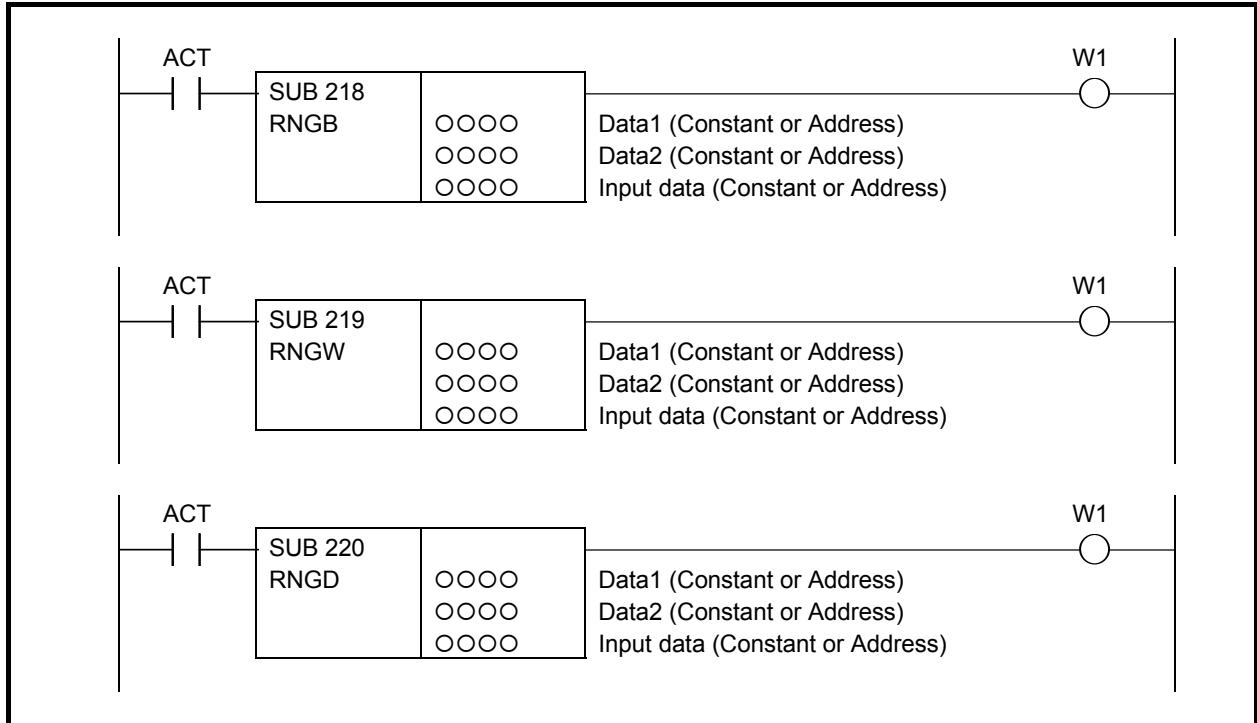


Fig. 4.7.7 Format of RNGB, RNGW and RNGD instructions

Table 4.7.7 Mnemonic of RNGB, RNGW and RNGD instructions

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		215 216 217	LEB instruction LEW instruction LED instruction				
3	(PRM)	0000		Data1 (Constant or Address)				
4	(PRM)	0000		Data2 (Constant or Address)				
5	(PRM)	0000		Input data (Constant or Address)				
6	WRT	0000 .0		Result				W1

Control condition

(a) Command (ACT)

ACT=0: Do not execute the instruction. The W1 becomes 0.

ACT=1: Execute the instruction. The result is output to W1.

Parameters

(a) Data 1

(b) Data 2

(c) Input data

You can specify the constant or any address. The valid data range is shown below.

RNGB: -128 to 127

RNGW: -32768 to 32767

RNGD: -2147483648 to 2147483647

Output (W1)

The result is output to W1.

W1=1: - When ACT=1 and "Data 1 \leq Input data \leq Data 2"

- W1=0: - When ACT=1 and "Data 2 \leq Input data \leq Data 1"
 - When ACT=0
 - When ACT=1 and except for above condition.

Operation Output Register (R9000, Z0)

This instruction does not update the operation output register. So, the operation output register will not change after this instruction.

4.7.8 COMPB (Comparison Between Binary Data: SUB 32)

This instruction compares 1, 2, and 4 byte binary data with one another. Results of comparison are set in the operation output register (R9000, Z0). Sufficient number of bytes are necessary in the memory to hold the input data and comparison data.

Format

Fig. 4.7.8 shows the ladder format and Table 4.7.8 shows the mnemonic format.

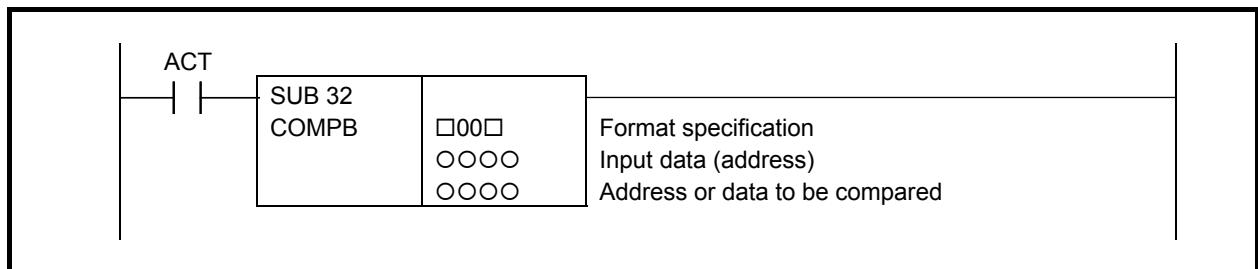


Fig. 4.7.8 Format of COMPB instruction

Table 4.7.8 Mnemonic of COMPB instruction

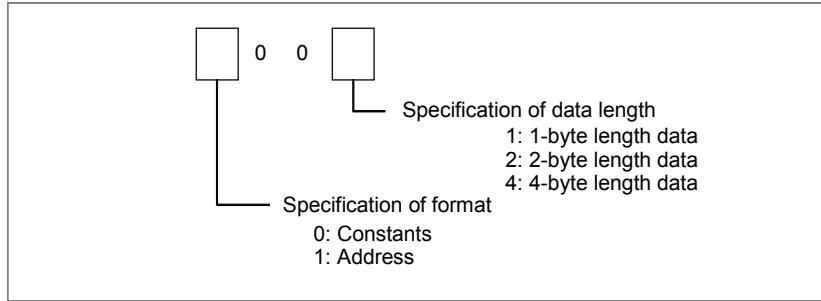
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		32	COMPB instruction				
3	(PRM)		□00□	Format specification				
4	(PRM)		0000	Input data (address)				
5	(PRM)		0000	Address of data to be compared				W1

Control condition

- (a) Command (ACT)
 ACT = 0: Do not execute COMPB.
 ACT = 1: Execute COMPB.

Parameters

- (a) Format specification
 Specify data length (1,2, or 4 bytes) and format for the input data ('constants data' or 'address data').



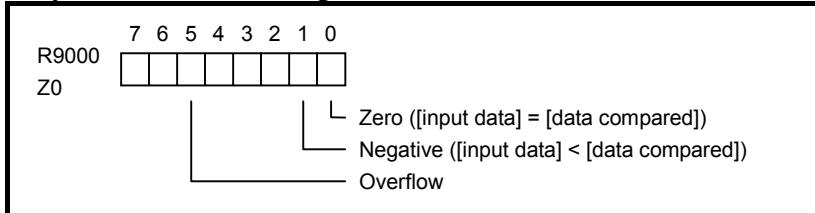
- (b) Input data (address)
Format for the input data is determined by the specification in (a).
- (c) Address of data to be compared
Indicates the address in which the comparison data is stored.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Operation output register (R9000, Z0)

The data involved in the operation are set in this register. This register is set with data on operation. If register bit 1 is on, they indicate the following:



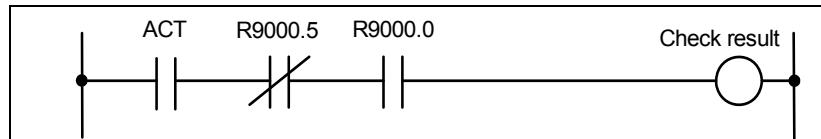
The following table shows the relationship among the [input data], [data compared], and operation output register.

	R9000.5 Z0.5	R9000.1 Z0.1	R9000.0 Z0.0
[Input data] = [data compared]	0	0	1
[Input data] > [data compared]	0	0	0
[Input data] < [data compared]	0	1	0
Overflow	1	0	0

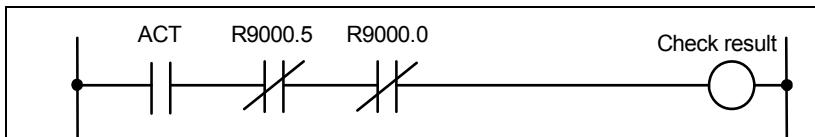
Programming examples for the operation output register

Programming examples of comparison between two positive values are shown below.

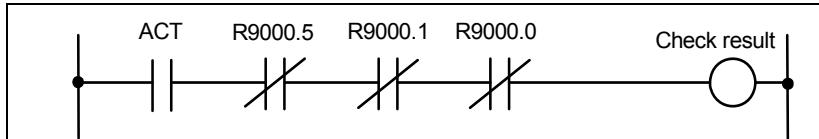
- (1) When checking that [input data] = [data compared]



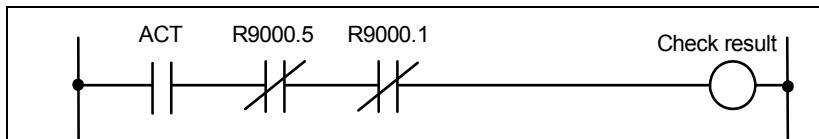
- (2) When checking that [input data] ≠ [data compared]



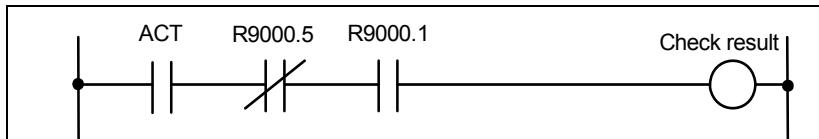
- (3) When checking that [input data] > [data compared]



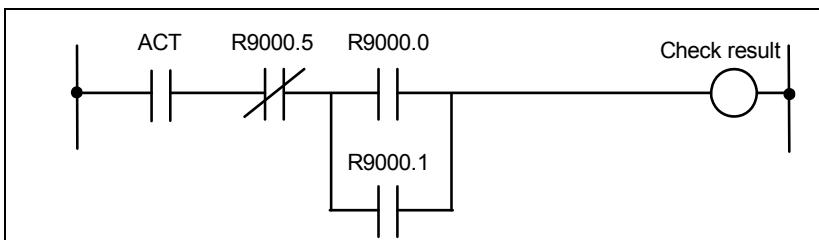
- (4) When checking that $[input\ data] \geq [data\ compared]$



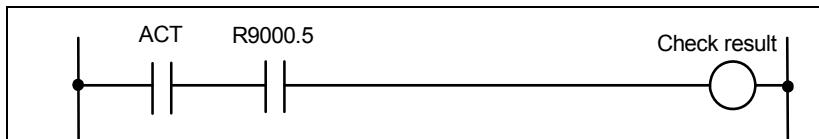
- (5) When checking that $[input\ data] < [data\ compared]$



- (6) When checking that $[input\ data] \leq [data\ compared]$



- (7) When checking for an overflow of the comparison operation



4.7.9 COMP (Comparison: SUB 15)

Compares input and comparison values. The value type in this instruction is BCD.

Format

Fig. 4.7.9 shows the ladder format and Table 4.7.9 shows the mnemonic format.

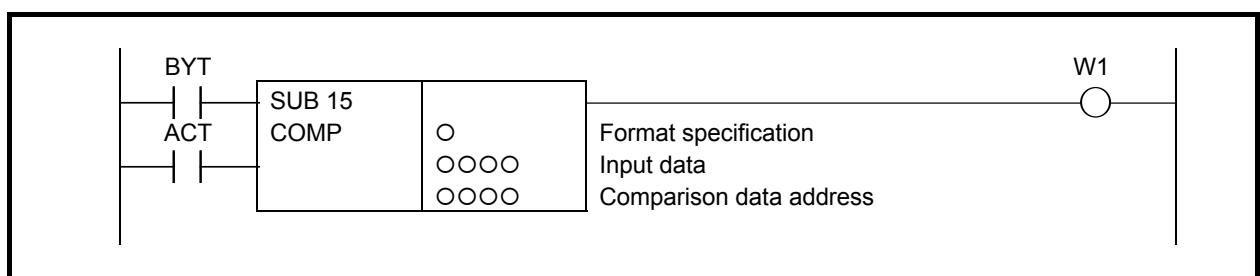


Fig. 4.7.9 Format of COMP instruction

Table 4.7.9 Mnemonic of COMP instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		ACT			BYT	ACT
3	SUB		15	COMP instruction				
4	(PRM)		0	Format specification				
5	(PRM)	0000		Input data				
6	(PRM)	0000		Comparison data address				
7	WRT	0000 .0		Comparison result output				W1

Control conditions

- (a) Specify the data size. (BYT)
 - BYT = 0: Process data (input value and comparison value) is BCD two digits long.
 - BYT = 1: Process data (input value and comparison value) is BCD four digits long.
- (b) Execution command (ACT)
 - ACT = 0: The COMP instruction is not executed. W1 does not alter.
 - ACT = 1: The COMP instruction is executed and the result is output to W1.

Parameters

- (a) Format specification
 - 0: Specifies input data with a constant.
 - 1: Specifies input data with an address
 - Not specify input data directly, but specify an address storing input data.
- (b) Input data
 - The input data can be specified as either a constant or the address storing it. The selection is made by a parameter of format specification.
- (c) Comparison data address
 - Specifies the address storing the comparison data.

 **CAUTION**

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Comparison result output(W1)

- W1 = 0: Input data > Comparison data
- W1 = 1: Input data \leq Comparison data

4.7.10 COIN (Coincidence Check: SUB 16)

Checks whether the input value and comparison value coincide.
The value type in this instruction is BCD.

Format

Fig. 4.7.10 shows the ladder format and Table 4.7.10 shows the mnemonic format.

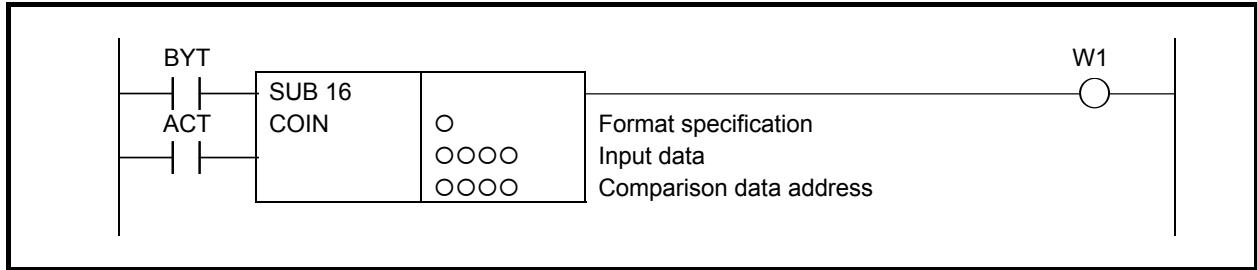


Fig. 4.7.10 Format of COIN instruction

Table 4.7.10 Mnemonic of COIN instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		BYT				BYT
2	RD.STK	0000 .O		ACT			BYT	ACT
3	SUB	16		COIN instruction				
4	(PRM)	O		Format specification				
5	(PRM)	0000		Input data				
6	(PRM)	0000		Comparison data address				
7	WRT	0000 .O		Comparison result output			▼	W1

Control conditions

- (a) Specify the data size.
BYT = 0: Process data (input value, and comparison values).
Each BCD is two digits long.
BYT = 1: Each BCD four digits long.
- (b) Execution command
ACT = 0: The COIN instruction is not executed. W1 does not change.
ACT = 1: The COIN instruction is executed and the results is output to W1.

Parameters

- (a) Format specification
0: Specifies input data as a constant.
1: Specifies input data as an address.
- (b) Input data
The input data can be specified as either a constant or an address storing it. The selection is made by a parameter of format designation.
- (c) Comparison data address
Specifies the address storing the comparison data.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Comparison result output (W1)

- W1 = 0: Input data ≠ Comparison data
- W1 = 1: Input data = Comparison data

4.8 BIT OPERATION

The following types of bit operation instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	DIFU	57	Rising edge detection
2	DIFD	58	Falling edge detection
3	EOR	59	Exclusive OR
4	AND	60	Logical AND
5	OR	61	Logical OR
6	NOT	62	Logical NOT
7	PARI	11	Parity check
8	SFT	33	Shift register
9	EORB	265	Exclusive OR (1 byte length)
10	EORW	266	Exclusive OR (2 bytes length)
11	EORD	267	Exclusive OR (4 bytes length)
12	ANDB	268	Logical AND (1 byte length)
13	ANDW	269	Logical AND (2 bytes length)
14	ANDD	270	Logical AND (4 bytes length)
15	ORB	271	Logical OR (1 byte length)
16	ORW	272	Logical OR (2 bytes length)
17	ORD	273	Logical OR (4 bytes length)
18	NOTB	274	Logical NOT (1 byte length)
19	NOTW	275	Logical NOT (2 bytes length)
20	NOTD	276	Logical NOT (4 bytes length)
21	SHLB	277	Bit shift left (1 byte length)
22	SHLW	278	Bit shift left (2 bytes length)
23	SHLD	279	Bit shift left (4 bytes length)
24	SHLN	280	Bit shift left (Arbitrary bytes length)
25	SHRB	281	Bit shift right (1 byte length)
26	SHRW	282	Bit shift right (2 bytes length)
27	SHRD	283	Bit shift right (4 bytes length)
28	SHRN	284	Bit shift right (Arbitrary bytes length)
29	ROLB	285	Bit rotation left (1 byte length)
30	ROLW	286	Bit rotation left (2 bytes length)
31	ROLD	287	Bit rotation left (4 bytes length)
32	ROLN	288	Bit rotation left (Arbitrary bytes length)
33	RORB	289	Bit rotation right (1 byte length)
34	RORW	290	Bit rotation right (2 bytes length)
35	RORD	291	Bit rotation right (4 bytes length)
36	RORN	292	Bit rotation right (Arbitrary bytes length)
37	BSETB	293	Bit set (1 byte length)
38	BSETW	294	Bit set (2 bytes length)
39	BSETD	295	Bit set (4 bytes length)
40	BSETN	296	Bit set (Arbitrary bytes length)
41	BRSTB	297	Bit reset (1 byte length)
42	BRSTW	298	Bit reset (2 bytes length)
43	BRSTD	299	Bit reset (4 bytes length)
44	BRSTN	300	Bit reset (Arbitrary bytes length)
45	BTSTB	301	Bit test (1 byte length)
46	BTSTW	302	Bit test (2 bytes length)
47	BTSTD	303	Bit test (4 bytes length)
48	BTSTN	304	Bit test (Arbitrary bytes length)

	Instruction name	Sub number	Processing
49	BPOSB	305	Bit search (1 byte length)
50	BPOSW	306	Bit search (2 bytes length)
51	BPOSD	307	Bit search (4 bytes length)
52	BPOSN	308	Bit search (Arbitrary bytes length)
53	BCNTB	309	Bit count (1 byte length)
54	BCNTW	310	Bit count (2 bytes length)
55	BCNTD	311	Bit count (4 bytes length)
56	BCNTN	312	Bit count (Arbitrary bytes length)

4.8.1 DIFU (Rising Edge Detection: SUB 57)

The DIFU instruction sets the output signal to 1 for one scanning cycle on a rising edge of the input signal.

Format

Fig. 4.8.1 shows the ladder format and Table 4.8.1 shows the mnemonic format.

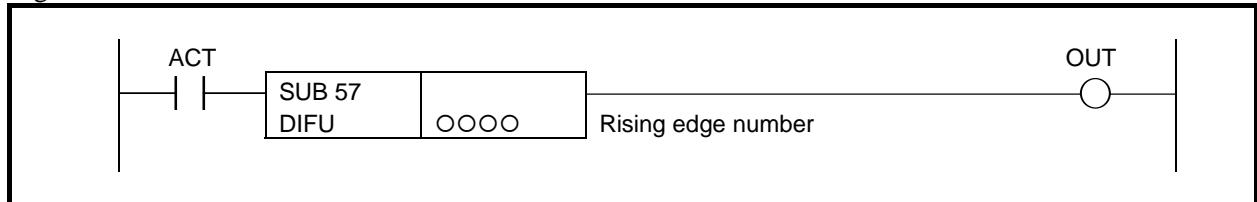


Fig. 4.8.1 Format of DIFU instruction

Table 4.8.1 Mnemonic of DIFU instruction

Mnemonic format					Memory status of control condition
Step number	Instruction	Address No.	Bit No.	Remarks	
1	RD	0000 .O		ACT	
2	SUB		57	DIFU instruction	
3	(PRM)	0000		Rising edge number	
4	WRT	0000 .O		OUT	

Control conditions

(a) Input signal (ACT)

On a rising edge ($0 \rightarrow 1$) of the input signal, the output signal is set to 1.

Detection result

(a) Output signal (OUT)

The output signal level remains at 1 for one scanning cycle of the ladder level where this functional instruction is operating.

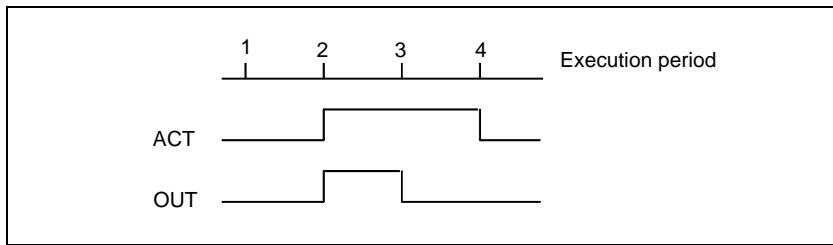
Parameters

	1st to 5th path PMC				Dual check safety PMC
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D	
Rising edge number	1 to 256	1 to 1000	1 to 2000	1 to 3000	1 to 256

⚠ WARNING

If the same number is used for another DIFU instruction or a DIFD instruction (described later) in one Ladder diagram, operation is not guaranteed.

Operation



4.8.2 DIFD (Falling Edge Detection: SUB 58)

The DIFD instruction set the output signal to 1 for one scanning period on a falling edge of the input signal.

Format

Fig. 4.8.2 shows the ladder format and Table 4.8.2 shows the mnemonic format.

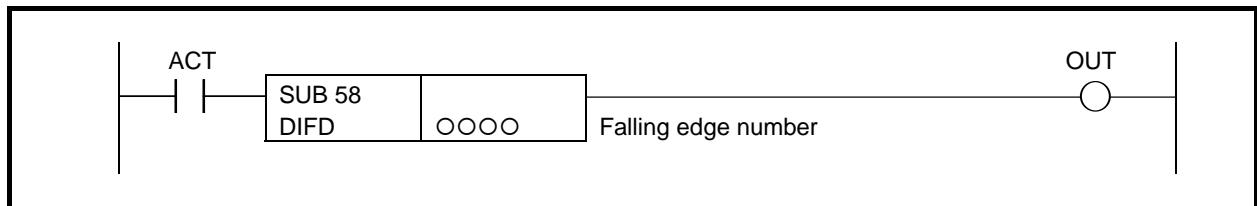


Fig. 4.8.2 Format of DIFD instruction

Table 4.8.2 Mnemonic of DIFD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		58	DIFD instruction				
3	(PRM)	0000		Falling edge number				▼
4	WRT	0000 .O		OUT				OUT

Control conditions

(a) Input signal (ACT)

On a falling edge ($1 \rightarrow 0$) of the input signal, the output signal is set to 1.

Detection result

(a) Output signal (OUT)

The output signal level remains at 1 for one scanning period of the ladder level where this functional instruction is operating.

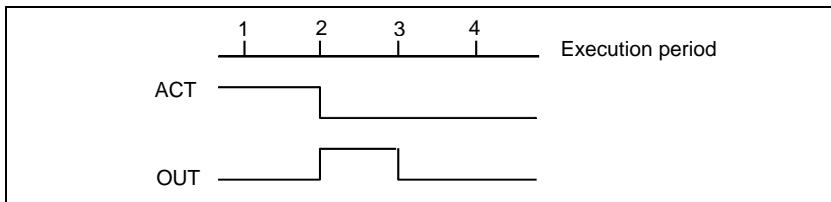
Parameters

	1st to 5th path PMC				Dual check safety PMC
	PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D	
Falling edge number	1 to 256	1 to 1000	1 to 2000	1 to 3000	1 to 256

WARNING

If the same number is used for another DIFD instruction or a DIFU instruction (described above) in one ladder diagram, operation is not guaranteed.

Operation



4.8.3 EOR (Exclusive OR: SUB 59)

The EOR instruction exclusive-ORs the contents of address A with a constant (or the contents of address B), and stores the result at address C. The value type in this instruction is binary.

Format

Fig. 4.8.3 shows the ladder format and Table 4.8.3 shows the mnemonic format.

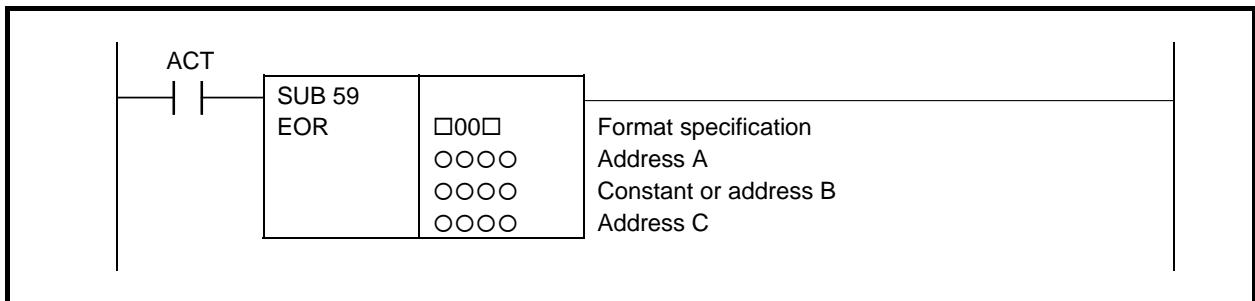


Fig. 4.8.3 Format of EOR instruction

Table 4.8.3 Mnemonic of EOR instruction

Mnemonic format					Memory status of control condition
Step number	Instruction	Address No.	Bit No.	Remarks	ST3 ST2 ST1 ST0
1	RD	OOOO .O		ACT	
2	SUB	59		EOR instruction	
3	(PRM)	□00□		Format specification	
4	(PRM)	OOOO		Address A	
5	(PRM)	OOOO		Constant or address B	
6	(PRM)	OOOO		Address C	▼

Control conditions

- (a) Input signal (ACT)
 - ACT=0: The EOR instruction is not executed.
 - ACT=1: The EOR instruction is executed.

Parameters

- (a) Format specification

Specify a data length (1, 2, or 4 bytes), and an input data format (constant or address specification).



(b) Address A

Input data to be exclusive-ORed. The data that is held starting at this address and has the data length specified in format specification is treated as input data.

(c) Constant or address B

Input data to be exclusive-ORed with. When address specification is selected in format specification, the data that is held starting at this address and has the data length specified in format specification is treated as input data.

(d) Address C

Address used to store the result of an exclusive OR operation. The result of an exclusive OR operation is stored starting at this address, and has the data length specified in format specification.

CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Operation

When address A and address B hold the following data:

Address A	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr></table>	1	1	1	0	0	0	1	1
1	1	1	0	0	0	1	1		
Address B	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1		

The result of the exclusive OR operation is as follows:

Address C	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	0	1	1	0	1	1	0
1	0	1	1	0	1	1	0		

4.8.4 AND (Logical AND: SUB 60)

The AND instruction ANDs the contents of address A with a constant (or the contents of address B), and stores the result at address C. The value type in this instruction is binary.

Format

Fig. 4.8.4 shows the ladder format and Table 4.8.4 shows the mnemonic format.

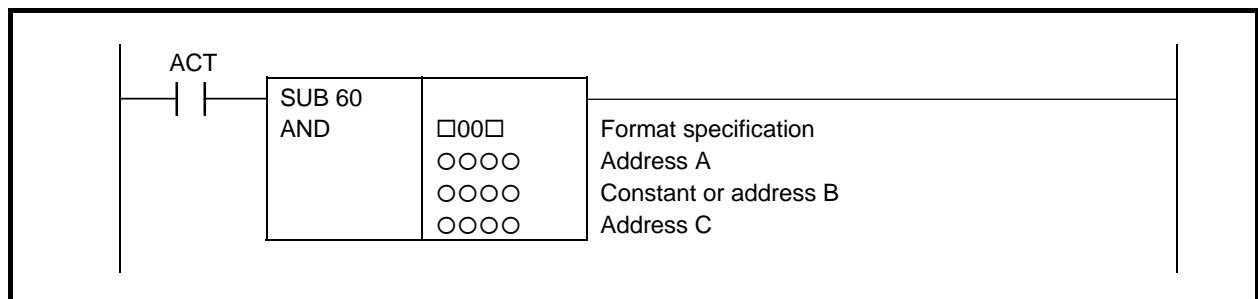


Fig. 4.8.4 Format of AND instruction

Table 4.8.4 Mnemonic of AND instruction

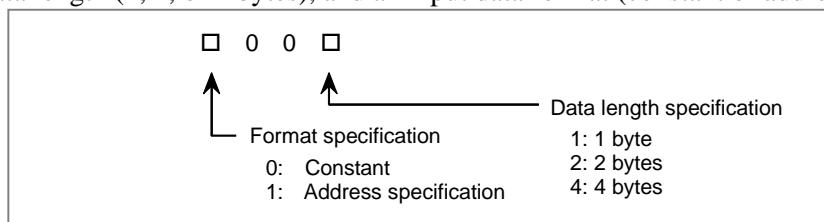
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB	60		AND instruction				
3	(PRM)	□00□		Format specification				
4	(PRM)	0000		Address A				
5	(PRM)	0000		Constant or address B				
6	(PRM)	0000		Address C				

Control conditions

- (a) Input signal (ACT)
 ACT=0: The AND instruction is not executed.
 ACT=1 : The AND instruction is executed.

Parameters

- (a) Format specification
 Specify a data length (1, 2, or 4 bytes), and an input data format (constant or address specification).



- (b) Address A
 Input data to be ANDed. The data that is held starting at this address and has the data length specified in format specification is treated as input data.
- (c) Constant or address B
 Input data to be ANDed with. When address specification is selected in format specification, the data that is held starting at this address and has the data length specified in format specification is treated as input data.
- (d) Address C
 Address used to store the result of an AND operation. The result of an AND operation is stored starting at this address, and has the data length specified in format specification.

⚠ CAUTION

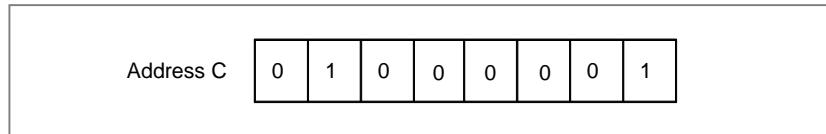
Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Operation

When address A and address B hold the following data:

Address A	1	1	1	0	0	0	1	1
Address B	0	1	0	1	0	1	0	1

The result of the AND operation is as follows:



4.8.5 OR (Logical OR: SUB 61)

The OR instruction ORs the contents of address A with a constant (or the contents of address B), and stores the result at address C. The value type in this instruction is binary.

Format

Fig. 4.8.5 shows the ladder format and Table 4.8.5 shows the mnemonic format.

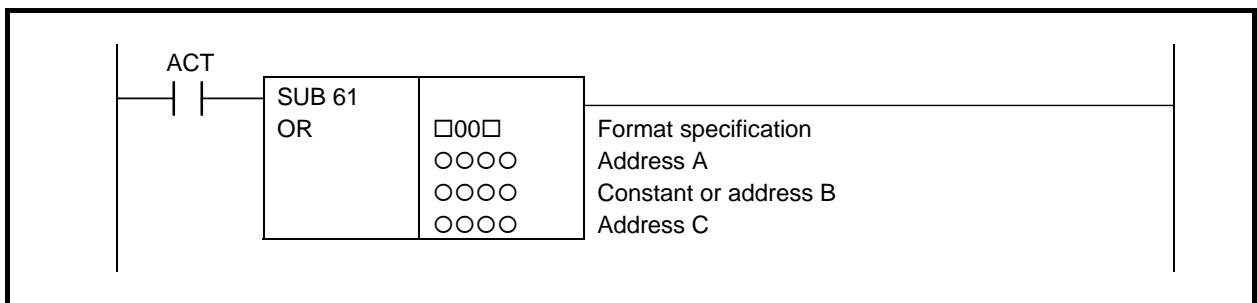


Fig. 4.8.5 Format of OR instruction

Table 4.8.5 Mnemonic of OR instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	OOOO .O		ACT				ACT
2	SUB		61	OR instruction				
3	(PRM)		□00□	Format specification				
4	(PRM)		OOOO	Address A				
5	(PRM)		OOOO	Constant or address B				
6	(PRM)		OOOO	Address C				▼

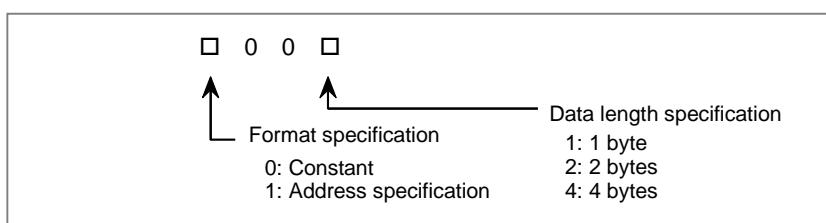
Control conditions

- (a) Input signal (ACT)
 - ACT=0: The OR instruction is not executed.
 - ACT=1: The OR instruction is executed.

Parameters

- (a) Format specification

Specify a data length (1, 2, or 4 bytes), and an input data format (constant or address specification).



(b) Address A

Input data to be ORed. The data that is held starting at this address and has the data length specified in format specification is treated as input data.

(c) Constant or address B

Input data to be ORed with. When address specification is selected in format specification, the data that is held starting at this address and has the data length specified in format specification is treated as input data.

(d) Address C

Address used to store the result of an OR operation. The result of an OR operation is stored starting at this address, and has the data length specified in format specification.

CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Operation

When address A and address B hold the following data:

Address A	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr></table>	1	1	1	0	0	0	1	1
1	1	1	0	0	0	1	1		
Address B	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1		

The result of the OR operation is as follows:

Address C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1		

4.8.6 NOT (Logical NOT: SUB 62)

The NOT instruction inverts each bit of the contents of address A, and stores the result at address B.

Format

Fig. 4.8.6 shows the ladder format and Table 4.8.6 shows the mnemonic format.

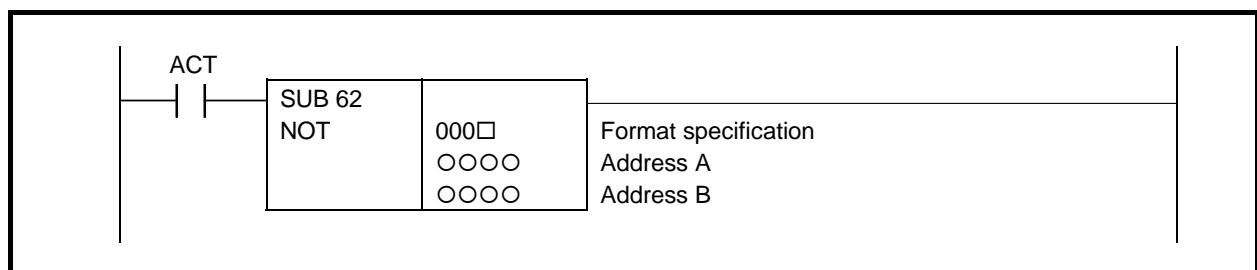


Fig. 4.8.6 Format of NOT instruction

Table 4.8.6 Mnemonic of NOT instruction

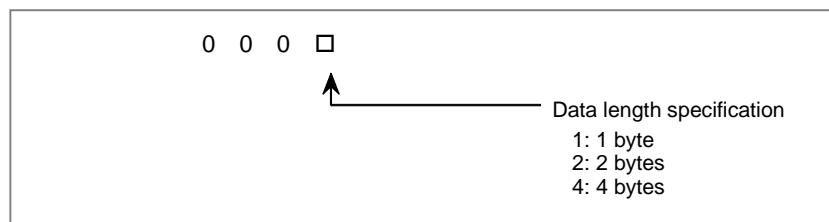
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		62	NOT instruction				
3	(PRM)	0000□		Format specification				
4	(PRM)	0000		Address A				
5	(PRM)	0000		Address C				

Control conditions

- (a) Input signal (ACT)
 ACT=0: The NOT instruction is not executed.
 ACT=1: The NOT instruction is executed.

Parameters

- (a) Format specification
 Specify a data length (1, 2, or 4 bytes).



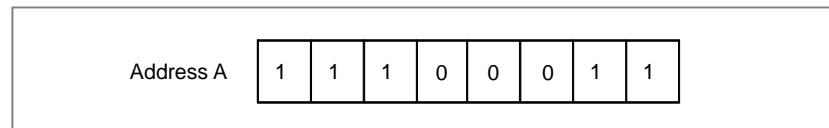
- (b) Address A
 Input data to be inverted bit by bit. The data that is held starting at this address and has the data length specified in format specification is treated as input data.
- (c) Address B
 Address used to output the result of a NOT operation. The result of a NOT operation is stored starting at this address, and has the data length specified in format specification.

⚠ CAUTION

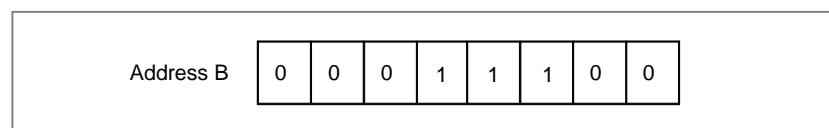
Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Operation

When address A holds the following data:



The result of the NOT operation is as follows:



4.8.7 PARI (Parity Check: SUB 11)

Checks the parity of code signals, and outputs an error if an abnormality is detected. Specifies either an even- or odd-parity check. Only one-byte (eight bits) of data can be checked.

Format

Fig. 4.8.7 (a) shows the ladder format and Table 4.8.7 shows the mnemonic format.

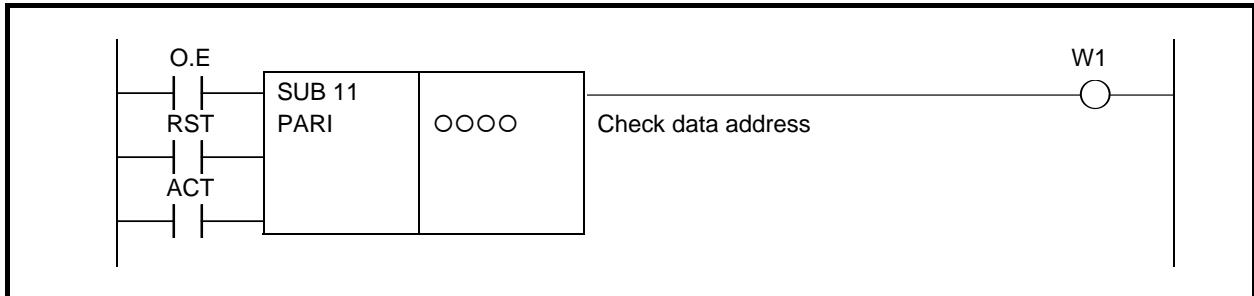


Fig. 4.8.7 (a) Format of PARI instruction

Table 4.8.7 Mnemonic of PARI instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		O.E				O.E
2	RD.STK	0000 .O		RST			O.E	RST
3	RD.STK	0000 .O		ACT		O.E	RST	ACT
4	SUB		11	PARI instruction				
5	(PRM)	0000		Check data address				
6	WRT	0000 .O		Error output				

Control conditions

- (a) Specify even or odd. (O.E)
 - O.E=0: Even-parity check
 - O.E=1: Odd-parity check
- (b) Reset (RST)
 - RST=0:
Disables reset.
 - RST=1:
Sets error output W1 to 0. That is, when a parity error occurs, setting RST to 1 results in resetting.
- (c) Execution command (ACT)
 - ACT=0: Parity checks are not performed. W1 does not alter.
 - ACT=1: Executes the PARI instruction, performing a parity check.

Error output (W1)

If the results of executing the PARI instruction is abnormal, W1=1 and an error is posted. The W1 address can be determined arbitrarily.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Example of using the PARI instruction

Fig. 4.8.7 (b) shows odd-parity checking of a code signal entered at address X036.

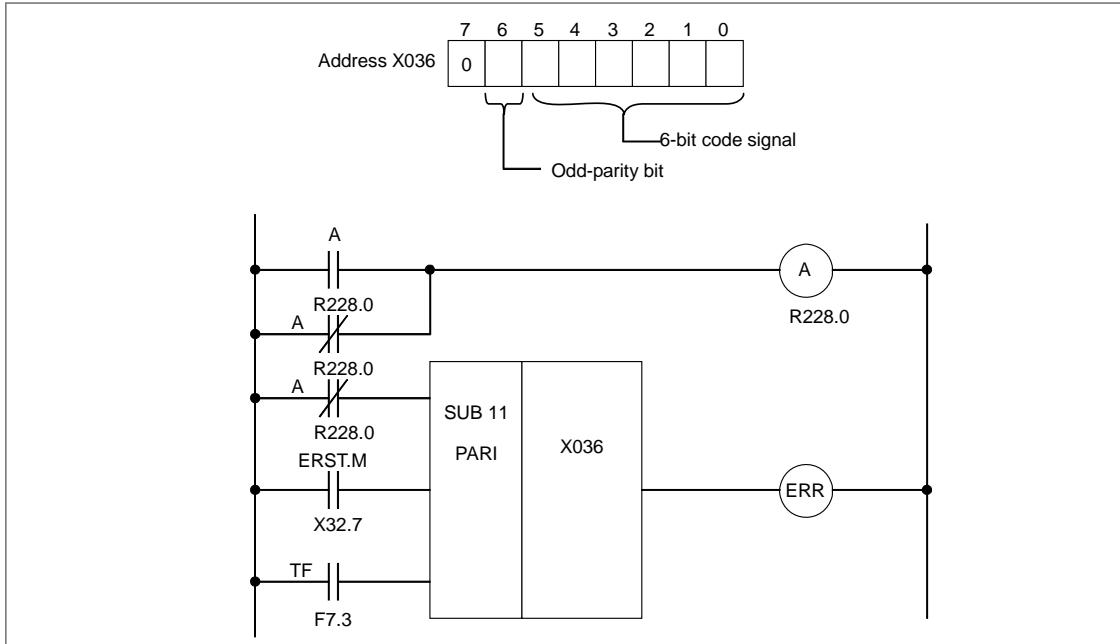


Fig. 4.8.7 (b) Ladder diagram for the PARI instruction



NOTE
For bits 0 to 7, bits other than those for the parity check must be 0.

4.8.8 SFT (Shift Register: SUB 33)

This instruction shifts 2-byte (16-bit) data by a bit to the left or right. Note that W1=1 when data "1" is shifted from the left extremity (bit 15) in left shift or from the right extremity (bit 0) in right shift.

Format

Fig. 4.8.8 shows the ladder format and Table 4.8.8 shows the mnemonic format.

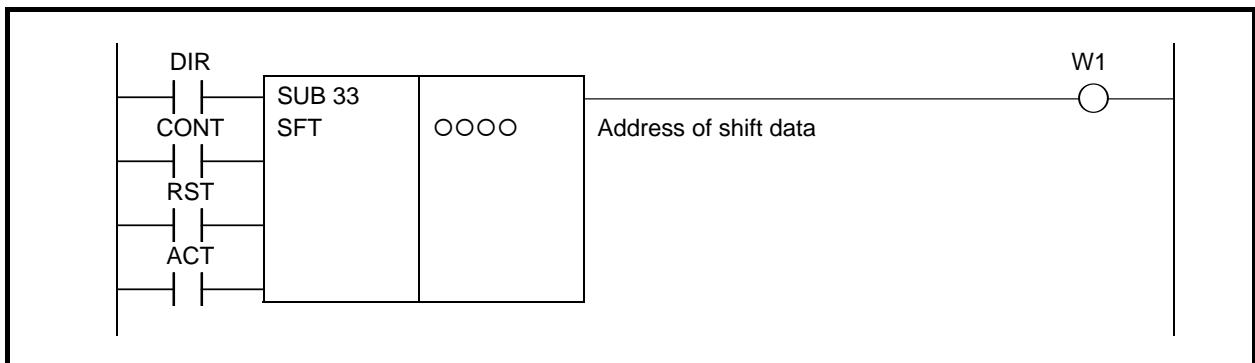


Fig. 4.8.8 Format of SFT instruction

Table 4.8.8 Mnemonic of SFT instruction

Mnemonic format				Memory status of control condition				
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		DIR				DIR
2	RD.STK	0000 .O		CONT			DIR	CONT
3	RD.STK	0000 .O		RST	DIR	CONT	RST	
4	RD.STK	0000 .O		ACT	DIR	CONT	RST	ACT
5	SUB	33		SFT instruction				
6	(PRM)	0000		Address of shift data				
7	WRT	0000 .O		Shifted-out output				

Control conditions

- (a) Shift direction specification (DIR)]

DIR=0: Left shift

DIR=1: Right shift

- (b) Condition specification (CONT)

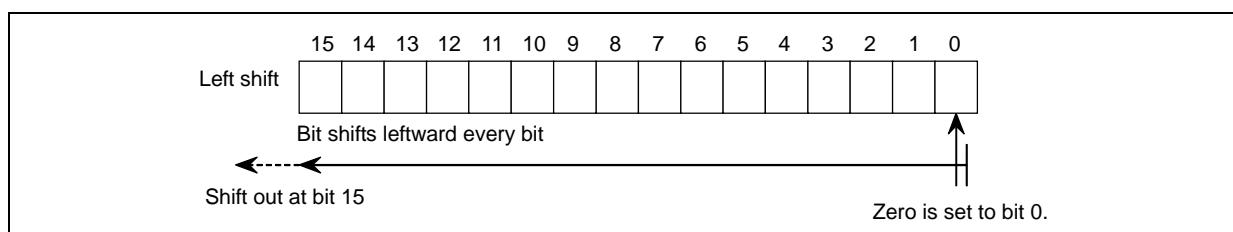
CONT=0:

On "1" bit shifts by one bit in the specified direction.

The condition of an adjacent bit (either right or left adjacent bit according to the specification of shift direction DIR) is set to the original bit position of the on "1" bit.

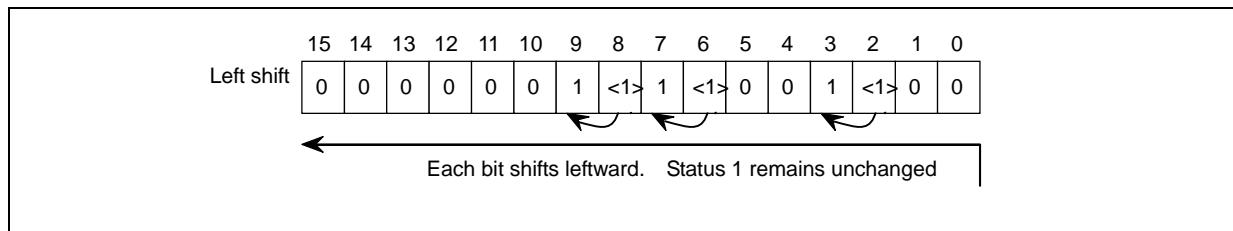
Also, "0" is set to bit 0 after shifting in the left direction or set to bit 15 after shifting in the right direction.

In case of leftward shift;



CONT=1:

Shift is the same as above, but 1s are set to shifted bits.



- (c) Reset (RST)

The shifted out data (W1=1) is reset (W1=0).

RST=0: W1 is not reset.

RST=1: W1 is reset (W1=0).

- (d) Actuation signal (ACT)

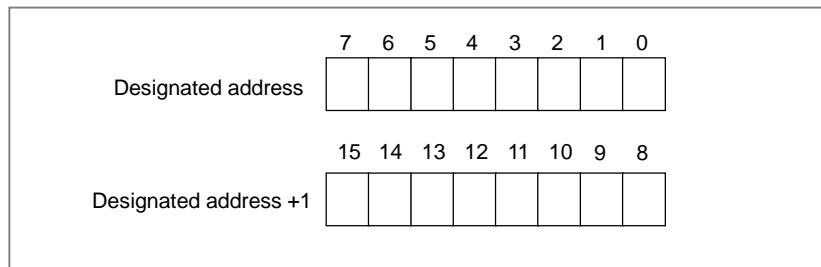
Shift processing is done when ACT=1. For shifting one bit only, execute an instruction when ACT=1, and then, set ACT to 0 (ACT=0).

Parameters

(a) Shift data addresses

Sets shift data addresses. These designated addresses require a continuous 2-byte memory for shift data.

Bit numbers are represented by bit 0 to 15 as shown below. When addresses are designated for programming, an address number is attached every 8 bits, and the designable bit numbers are 0 to 7.



Shifted out

W1=0: "1" was not shifted out because of the shift operation.

W1=1: "1" was shifted out because of the shift operation.



CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.8.9 EORB (Exclusive OR (1 Byte Length) : SUB 265) EORW (Exclusive OR (2 Bytes Length) : SUB 266) EORD (Exclusive OR (4 Bytes Length) : SUB 267)

The Exclusive OR instruction exclusive-ORs "Data A" with "Data B", and outputs the result to "Address C".

In "Data A" and "Data B", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Exclusive OR instructions are available according to the type of data to be operated. In each instruction, "Data A", "Data B", and the data at "Address C" are of the same data type.

Table 4.8.9 (a) Kinds of Exclusive OR instruction

	Instruction name	SUB No.	Data type
1	EORB	265	1 byte length
2	EORW	266	2 bytes length
3	EORD	267	4 bytes length

Format

Fig. 4.8.9(a) shows the ladder format and Table 4.8.9(b) shows the mnemonic format.

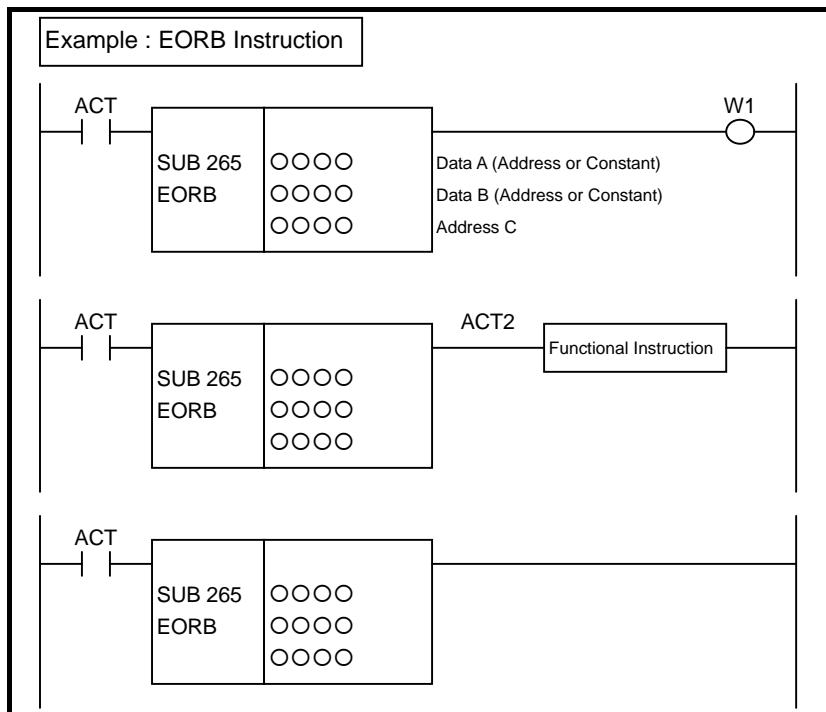


Fig. 4.8.9 (a) Format of EORB, EORW, EORD instruction

Table 4.8.9(b) Mnemonic of EORB, EORW, EORD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		265	SUB No. (EORB instruction)				
3	(PRM)	0000		Data A (Address or Constant)				
4	(PRM)	0000		Data B (Address or Constant)				
5	(PRM)	0000		Address C				
6	WRT	0000 .0		Normal end output				W1

Control conditions

(a) Input signal (ACT)

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

(a) Data A

Specify input data to be exclusive-ORED. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available value
EORB	-128 to 127
EORW	-32768 to 32767
EORD	-2147483648 to 2147483647

(b) Data B

Specify input data to be exclusive-ORed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the same range as for "Data A" may be specified.

(c) Address C

Specify the address to which the result of exclusive-OR operation is to be output.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

Operation

When "Data A" and "Data B" hold the following values, the value indicated below is output to "Address C":

Binary data																
Data A	<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>							1	1	1	0	0	0	1	1	(-29)
1	1	1	0	0	0	1	1									
Data B	<table border="1"> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table>							0	1	0	1	0	1	0	1	(85)
0	1	0	1	0	1	0	1									
Address C	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>							1	0	1	1	0	1	1	0	(-74)
1	0	1	1	0	1	1	0									

Fig. 4.8.9 (b) Example of operation of the EORB, EORW, and EORD instructions

4.8.10 ANDB (Logical AND (1 Byte Length) : SUB 268) ANDW (Logical AND (2 Bytes Length) : SUB 269) ANDD (Logical AND (4 Bytes Length) : SUB 270)

The Logical AND instruction logical-ANDs "Data A" with "Data B", and outputs the result to "Address C".

In "Data A" and "Data B", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Logical AND instructions are available according to the type of data to be operated. In each instruction, "Data A", "Data B", and the data at "Address C" are of the same data type.

Table 4.8.10 (a) Kinds of Logical AND instruction

	Instruction name	SUB No.	Data type
1	ANDB	268	1 byte length
2	ANDW	269	2 bytes length
3	ANDD	270	4 bytes length

Format

Fig. 4.8.10(a) shows the ladder format and Table 4.8.10(b) shows the mnemonic format.

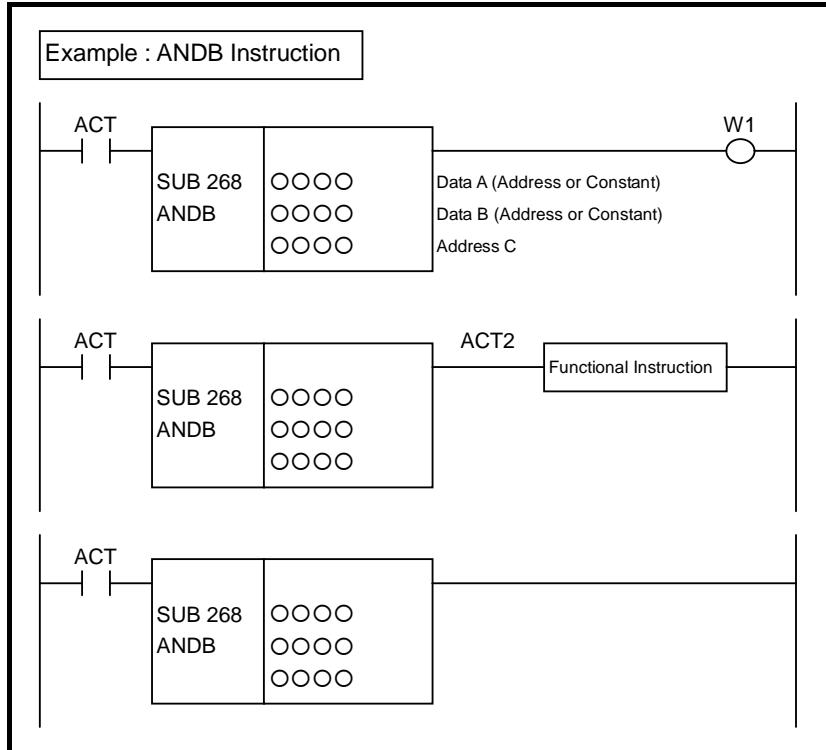


Fig. 4.8.10 (a) Format of ANDB, ANDW, ANDD instruction

Table 4.8.10(b) Mnemonic of ANDB, ANDW, ANDD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .O		ACT
2	SUB		268	SUB No. (ANDB instruction)
3	(PRM)	0000		Data A (Address or Constant)
4	(PRM)	0000		Data B (Address or Constant)
5	(PRM)	0000		Address C
6	WRT	0000 .O		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data A

Specify input data to be logical-ANDed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available value
ANDB	-128 to 127
ANDW	-32768 to 32767
ANDD	-2147483648 to 2147483647

- (b) Data B

Specify input data to be logical-ANDed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the same range as for "Data A" may be specified.

(c) Address C

Specify the address to which the result of logical-AND operation is to be output.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

Operation

When "Data A" and "Data B" hold the following values, the value indicated below is output to "Address C":

Binary data																
Data A	<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>								1	1	1	0	0	0	1	1
1	1	1	0	0	0	1	1									
Data B	<table border="1"> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table>								0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1									
Address C	<table border="1"> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> </table>								0	1	0	0	0	0	0	1
0	1	0	0	0	0	0	1									

Fig. 4.8.10 (b) Example of operation of the ANDB, ANDW, and ANDD instructions

4.8.11 ORB (Logical OR (1 Byte Length) : SUB 271) ORW (Logical OR (2 Bytes Length) : SUB 272) ORD (Logical OR (4 Bytes Length) : SUB 273)

The Logical OR instruction logical-ORs "Data A" with "Data B", and outputs the result to "Address C". In "Data A" and "Data B", a constant or a PMC memory address for storing data can be specified. As indicated below, three types of Logical OR instructions are available according to the type of data to be operated. In each instruction, "Data A", "Data B", and the data at "Address C" are of the same data type.

Table 4.8.11 (a) Kinds of Logical OR instruction

	Instruction name	SUB No.	Data type
1	ORB	271	1 byte length
2	ORW	272	2 bytes length
3	ORD	273	4 bytes length

Format

Fig. 4.8.11(a) shows the ladder format and Table 4.8.11(b) shows the mnemonic format.

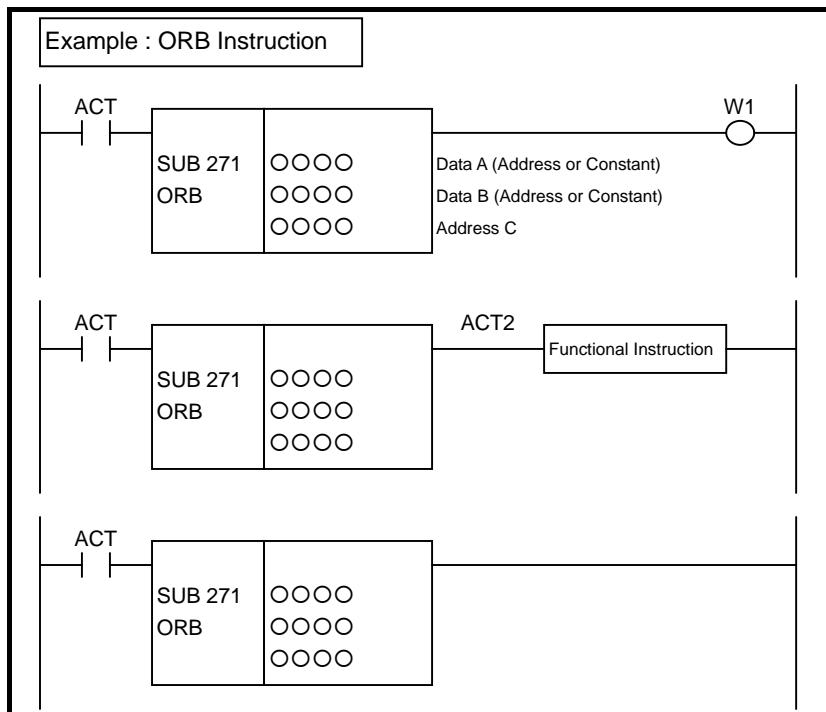


Fig. 4.8.11 (a) Format of ORB, ORW, ORD instruction

Table 4.8.11(b) Mnemonic of ORB, ORW, ORD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		271	SUB No. (ORB instruction)				
3	(PRM)	0000		Data A (Address or Constant)				
4	(PRM)	0000		Data B (Address or Constant)				
5	(PRM)	0000		Address C				
6	WRT	0000 .0		Normal end output				W1

Control conditions

- (a) Input signal (ACT)
 ACT = 0: Instruction not executed.
 ACT = 1: Executed.

Parameters

- (a) Data A
 Specify input data to be logical-ORed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available value
ORB	-128 to 127
ORW	-32768 to 32767
ORD	-2147483648 to 2147483647

- (b) Data B

Specify input data to be logical-ORed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the same range as for "Data A" may be specified.

(c) Address C

Specify the address to which the result of logical-OR operation is to be output.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

Operation

When "Data A" and "Data B" hold the following values, the value indicated below is output to "Address C":

Binary data									
Data A	1	1	1	0	0	0	1	1	(-29)
Data B	0	1	0	1	0	1	0	1	(85)
<hr/>									
Address C	1	1	1	1	0	1	1	1	(-9)

Fig. 4.8.11 (b) Example of operation of the ORB, ORW, and ORD instructions

4.8.12 NOTB (Logical NOT (1 Byte Length) : SUB 274) NOTW (Logical NOT (2 Bytes Length) : SUB 275) NOTD (Logical NOT (4 Bytes Length) : SUB 276)

The Logical NOT instruction performs a logical-NOT operation on "Data A" and outputs the result to "Address B".

In "Data A", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Logical NOT instructions are available according to the type of data to be operated. In each instruction, "Data A" and the data at "Address B" are of the same data type.

Table 4.8.12 (a) Kinds of Logical NOT instruction

	Instruction name	SUB No.	Data type
1	NOTB	274	1 byte length
2	NOTW	275	2 bytes length
3	NOTD	276	4 bytes length

Format

Fig. 4.8.12(a) shows the ladder format and Table 4.8.12(b) shows the mnemonic format.

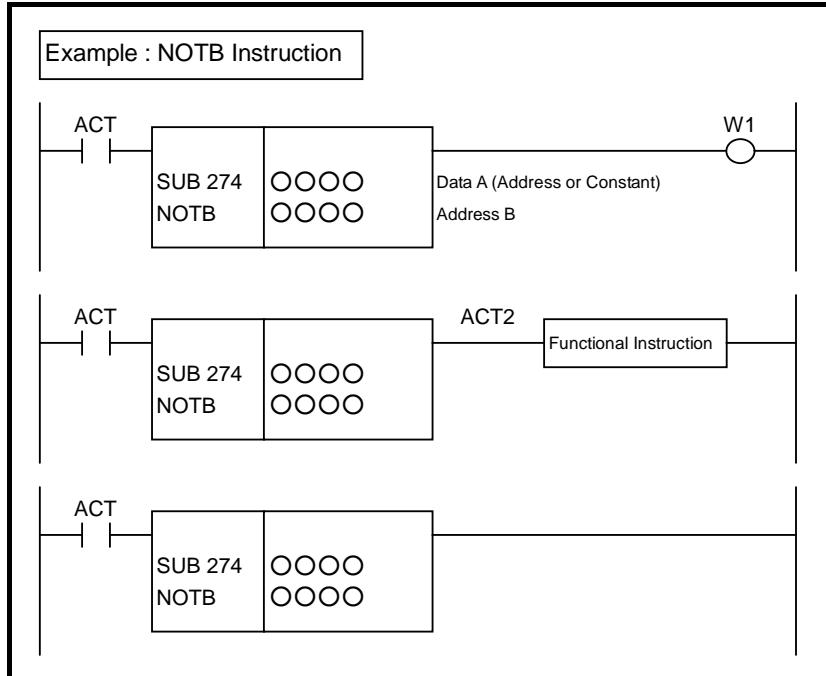


Fig. 4.8.12 (a) Format of NOTB, NOTW, NOTD instruction

Table 4.8.12(b) Mnemonic of NOTB, NOTW, NOTD instruction

Mnemonic format**Memory status of control condition**

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .O		ACT
2	SUB		274	SUB No. (ORB instruction)
3	(PRM)	0000		Data A (Address or Constant)
4	(PRM)	0000		Address B
5	WRT	0000 .O		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

(a) Input signal (ACT)

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

(a) Data A

Specify input data on which a logical-NOT operation is to be performed. In this parameter, a constant or a PMC memory address for storing data can be specified. Specify data by using signed binary data. A value within the following range may be specified:

Instruction name	Available value
NOTB	-128 to 127
NOTW	-32768 to 32767
NOTD	-2147483648 to 2147483647

(b) Address B

Specify the address to which the result of logical-NOT operation is to be output.

Output (W1)

When the instruction is executed, W1=1 is set. That is, W1 always assumes the same state as ACT.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

Operation

When "Data A" holds the following value, the value indicated below is output to "Address B":

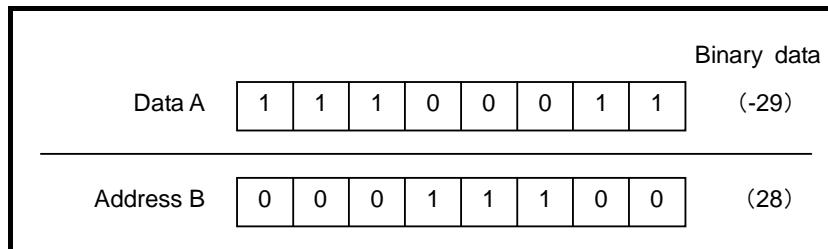


Fig. 4.8.12 (b) Example of operation of the NOTB, NOTW, and NOTD instructions

4.8.13 SHLB (Bit Shift Left (1 Byte Length) : SUB 277) SHLW (Bit Shift Left (2 Bytes Length) : SUB 278) SHLD (Bit Shift Left (4 Bytes Length) : SUB 279)

The Bit shift left instruction shifts bit data to the left by a specified number of bits. In the empty bit position(s) after shift operation, 0 is shifted in. The result of shift operation is output to a specified address.

As indicated below, three types of Bit shift left instructions are available according to the type of data to be operated. Shift source bit data and the data at a shift result output address are of the same data type.

Table4.8.13 (a) Kinds of Bit shift left instruction

	Instruction name	SUB No.	Data type
1	SHLB	277	1 byte length data
2	SHLW	278	2 bytes length data
3	SHLD	279	4 bytes length data

The value of the last bit shifted out by a shift operation is output to W1. The value(s) of the preceding left-side bit(s) are lost.

When data is shifted by 5 bits:

Shifting source data = R100

Number of shift bits = 5

Shift result output address = R102

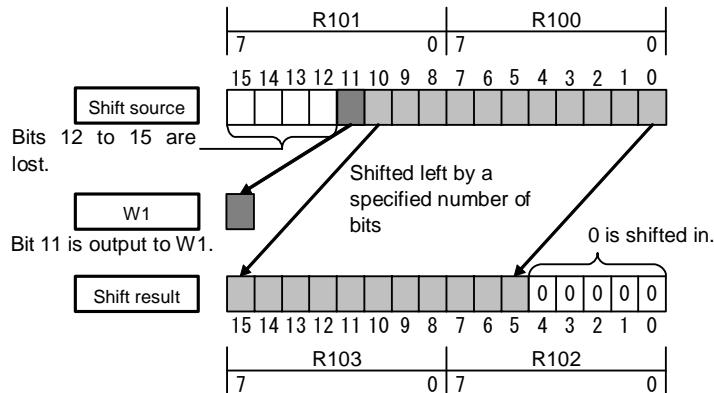


Fig. 4.8.13 (a) Example of SHLW instruction

If 0 or a negative value is specified in "Number of shift bits", the data specified in "Shift source data" is output to "Shift result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.13(b) shows the ladder format and Table 4.8.13(b) shows the mnemonic format.

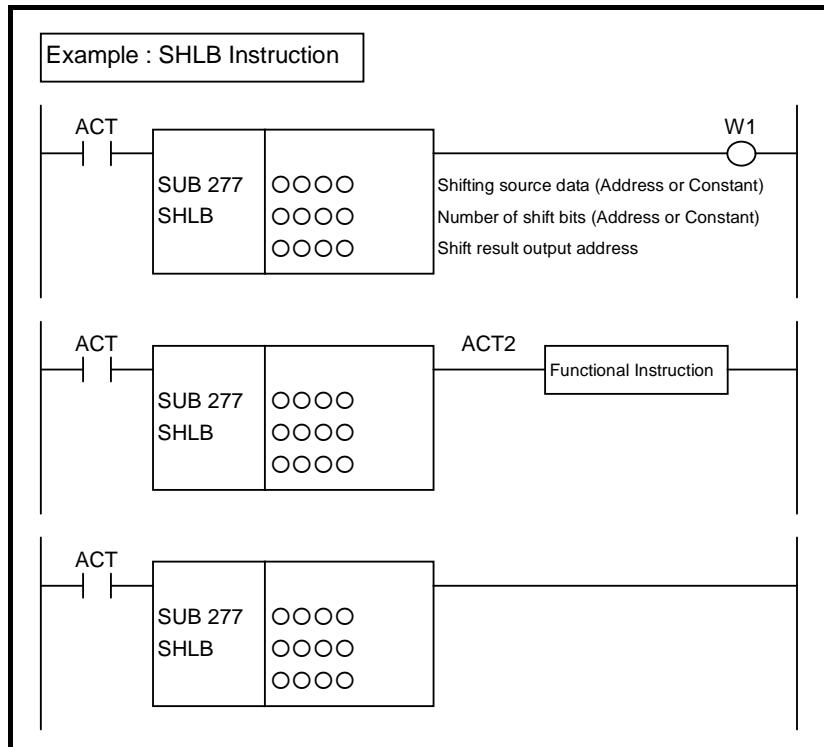


Fig. 4.8.13 (b) Format of SHLB, SHLW, SHLD instruction

**Table 4.8.13(b) Mnemonic of SHLB, SHLW, SHLD instruction
Mnemonic format**

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
					ST3 ST2 ST1 ST0
1	RD	0000 .O		ACT	
2	SUB	277		SUB No. (SHLB instruction)	
3	(PRM)	0000		Shifting source data (Address or Constant)	
4	(PRM)	0000		Number of shift bits (Address or Constant)	
5	(PRM)	0000		Shift result output address	
6	WRT	0000 .O		Shift out status output	ACT W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Shifting source data
 - Specify bit shift source data. In this parameter, a constant or a PMC memory address for storing data can be specified.
 - Specify data by signed binary data. A value within the following range may be specified:

Instruction name	Available value
SHLB	-128 to 127
SHLW	-32768 to 32767
SHLD	-2147483648 to 2147483647

- (b) Number of shift bits
 - By using signed binary data, specify the number of bits to be shifted. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the data specified in "Shifting source data" is shifted by a specified number of bits, and the result of shift operation is output to "Shift result output address". If 0 is specified, the data specified in "Shifting source data" is output to "Shift result output address" without modification, and W1=0 is set.
 - If a negative value is specified in this parameter, the data specified in "Shifting source data" is output to "Shift result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No shift operation in the opposite direction is performed.
 - In this parameter, a constant or a PMC memory address for storing data can be specified.
 - If an address is specified in this parameter, specify "Number of shift bits" by using memory of the same size as for data type handled by each instruction. For example, with the SHLW instruction, specify "Number of shift bits" by using memory 2 bytes long.
- (c) Shift result output address
 - Specify the address to which the result of bit shift operation is to be output. The result of shift operation is output to memory of the same size as for "Shifting source data".

Output (W1)

- W1=1: When the value of the last bit shifted out is 1
- W1=0: When no shift operation is executed (ACT=0)
 - When the value of the last bit shifted out is 0
 - When 0 or a negative value is specified in "Number of shift bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.14 SHLN (Bit Shift Left (Arbitrary Bytes Length) : SUB 280)

The Bit shift left instruction shifts bit data to the left by a specified number of bits. In the empty bit position(s) after shift operation, 0 is shifted in. The result of shift operation is output to a specified address.

The Bit shift left instruction performs a bit shift operation on a bit string of a specified data size. Shifting source data and the result of shift operation are of the same data size.

The value of the last bit shifted out by shift operation is output to W1. The value(s) of the preceding left-side bit(s) are lost.

When data is shifted by 5 bits:

Data size	= 3
Shifting source data top address	= R100
Number of shift bits	= 5
Shift result output address	= R103

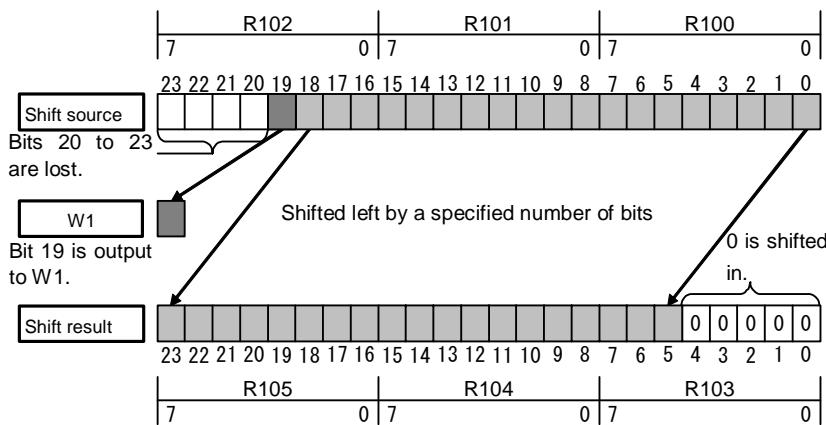


Fig. 4.8.14 (a) Example of SHLN instruction

If 0 or a negative value is specified in "Number of shift bits", the shift source data is output to "Shift result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.14(b) shows the ladder format and Table 4.8.14 shows the mnemonic format.

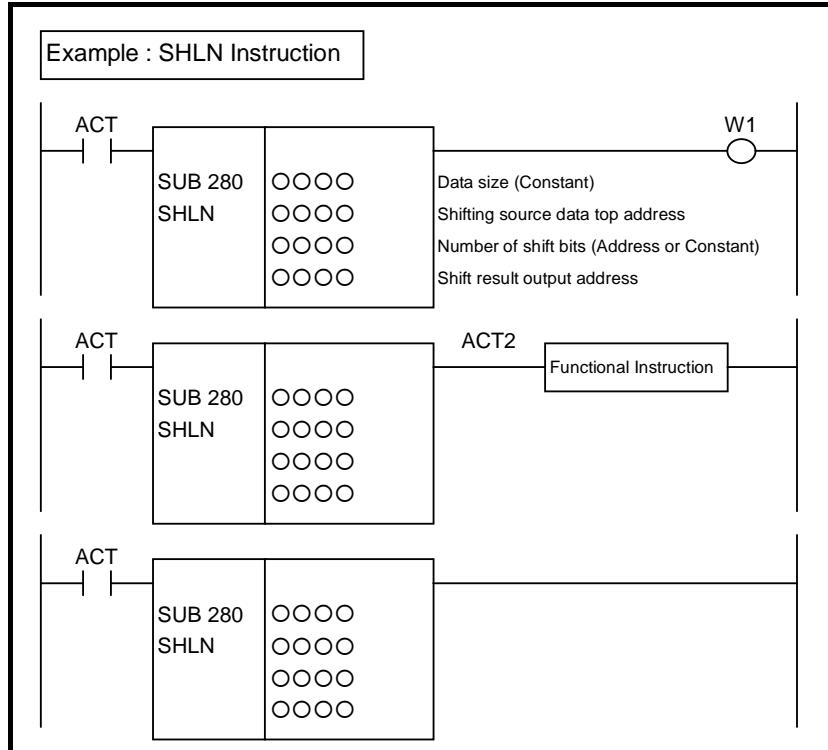


Fig. 4.8.14 (b) Format of SHLN instruction

Table 4.8.14 Mnemonic of SHLN instruction

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	280		SUB No. (SHLN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Shifting source data top address
5	(PRM)	0000		Number of shift bits (Address or Constant)
6	(PRM)	0000		Shift result output address
7	WRT	0000 .0		Shift out status output

Memory status of control condition

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Data size
Specify the number of bytes of data on which a bit shift operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that both of the area from "Shifting source data top address" and the area from "Shift result output address" may be arranged within valid address range.

- (b) Shifting source data top address

Specify the start address of bit shift source data.
Specify a data size in "Data size" mentioned in (a) above.

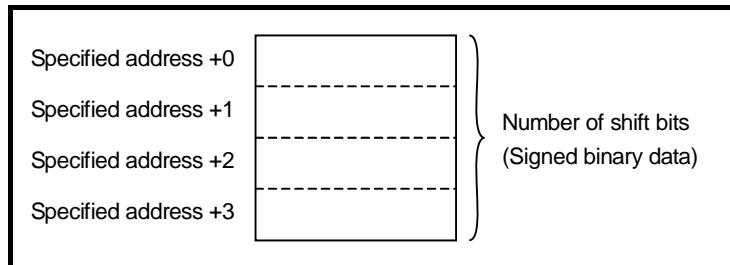
(c) Number of shift bits

By using 4-byte signed binary data, specify the number of bits to be shifted. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the shifting source data is shifted by a specified number of bits, and the result of shift operation is output to "Shift result output address". A value from 1 to ("Data size" × 8) may be specified. For example, if 6 is specified in "Data size", a value from 1 to 48 may be specified in this parameter. If a value greater than the valid range is specified, 0 is output to "Shift result output address", and W1=0 is set.

If 0 is specified, the shifting source data is output to "Shift result output address" without modification, and W1=0 is set.

If a negative value is specified in this parameter, the shifting source data is output to "Shift result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No shift operation in the opposite direction is performed.

In this parameter, a constant or a PMC memory address for storing data can be specified.



(d) Shift result output address

Specify the start address of an area to which the result of bit shift operation is to be output. The result of shift operation is output to memory of the same size as for shifting source data.

Output (W1)

W1=1: When the value of the last bit shifted out is 1

W1=0: When no shift operation is executed (ACT=0)

When the value of the last bit shifted out is 0

When 0 or a negative value is specified in "Number of shift bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.15 SHRB (Bit Shift Right (1 Byte Length) : SUB 281) SHRW (Bit Shift Right (2 Bytes Length) : SUB 282) SHRD (Bit Shift Right (4 Bytes Length) : SUB 283)

The Bit shift right instruction shifts bit data to the right by a specified number of bits. In the empty bit position(s) after shift operation, 0 is shifted in. The result of shift operation is output to a specified address.

As indicated below, three types of Bit shift right instructions are available according to the type of data to be operated. Shift source bit data and the data at a shift result output address are of the same data type.

Table 4.8.15 (a) Kinds of Bit shift right instruction

	Instruction name	SUB No.	Data type
1	SHRB	281	1 byte length data
2	SHRW	282	2 bytes length data
3	SHRD	283	4 bytes length data

The value of the last bit shifted out by shift operation is output to W1. The value(s) of the following right-side bit(s) are lost.

When data is shifted by 5 bits:

Shifting source data = R100

Number of shift bits = 5

Shift result output address = R102

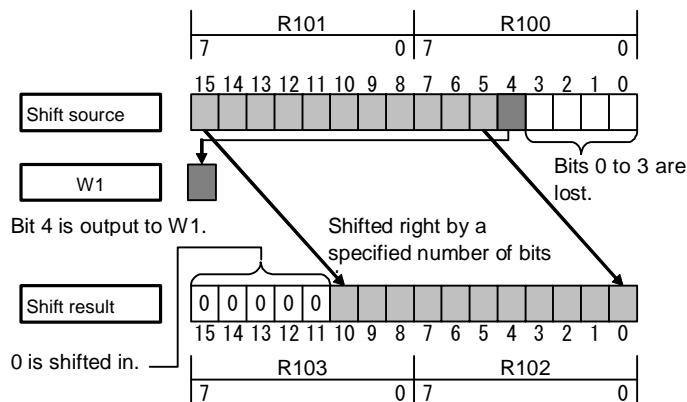


Fig. 4.8.15 (a) Example of SHRW instruction

If 0 or a negative value is specified in "Number of shift bits", the shift source data is output to "Shift result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.15(b) shows the ladder format and Table 4.8.15(b) shows the mnemonic format.

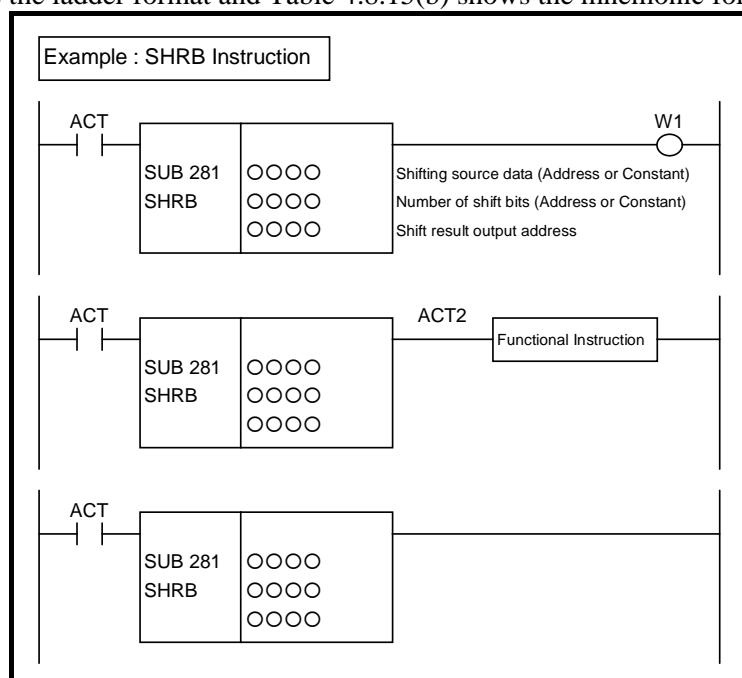


Fig. 4.8.15 (b) Format of SHRB, SHRW, SHRD instruction

**Table 4.8.15(b) Mnemonic of SHRB, SHRW, SHRD instruction
Mnemonic format**

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
					ST3 ST2 ST1 ST0
1	RD	0000 .0		ACT	
2	SUB		281	SUB No. (SHRB instruction)	
3	(PRM)	0000		Shifting source data (Address or Constant)	
4	(PRM)	0000		Number of shift bits (Address or Constant)	
5	(PRM)	0000		Shift result output address	
6	WRT	0000 .0		Shift out status output	ACT W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Shifting source data

Specify bit shift source data. In this parameter, a constant or a PMC memory address for storing data can be specified.
Specify data by signed binary data. A value within the following range may be specified:

Instruction name	Available value
SHRB	-128 to 127
SHRW	-32768 to 32767
SHRD	-2147483648 to 2147483647

- (b) Number of shift bits

By using signed binary data, specify the number of bits to be shifted. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the data specified in "Shifting source data" is shifted by a specified number of bits, and the result of shift operation is output to "Shift result output address". If 0 is specified, the data specified in "Shifting source data" is output to "Shift result output address" without modification, and W1=0 is set.
If a negative value is specified in this parameter, the data specified in "Shifting source data" is output to "Shift result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No shift operation in the opposite direction is performed.
In this parameter, a constant or a PMC memory address for storing data can be specified.
If an address is specified in this parameter, specify "Number of shift bits" by using signed binary data of the same size as for data type handled by each instruction. For example, with the SHRW instruction, specify "Number of shift bits" by using 2-byte signed binary data.

- (c) Shift result output address

Specify the address to which the result of bit shift operation is to be output. The result of shift operation is output to memory of the same size as for "Shifting source data".

Output (W1)

- W1=1: When the value of the last bit shifted out is 1
- W1=0: When no shift operation is executed (ACT=0)
 - When the value of the last bit shifted out is 0
 - When 0 or a negative value is specified in "Number of shift bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.16 SHRN (Bit Shift Right (Arbitrary Bytes Length) : SUB 284)

The Bit shift right instruction shifts bit data to the right by a specified number of bits. In the empty bit position(s) after shift operation, 0 is shifted in. The result of shift operation is output to a specified address.

The Bit shift right instruction performs a bit shift operation on a bit string of a specified data size. Shifting source data and the result of shift operation are of the same data size.

The value of the last bit shifted out by shift operation is output to W1. The value(s) of the following right-side bit(s) are lost.

When data is shifted by 5 bits:

Data size	= 3
Shifting source data top address	= R100
Number of shift bits	= 5
Shift result output address	= R103

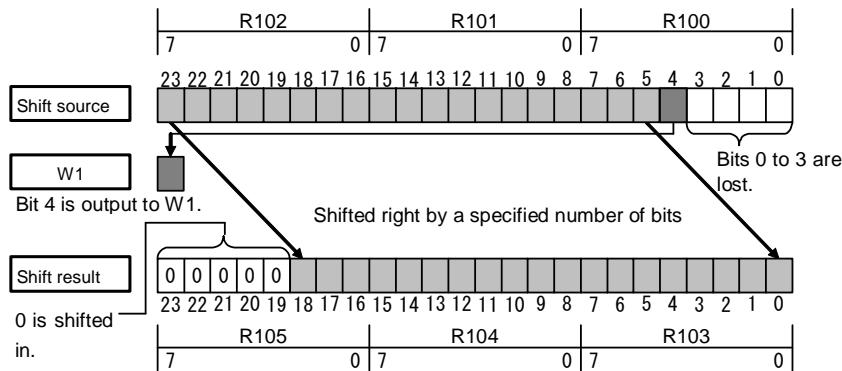


Fig. 4.8.16 (a) Example of SHRN instruction

If 0 or a negative value is specified in "Number of shift bits", the shift source data is output to "Shift result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.16(b) shows the ladder format and Table 4.8.16 shows the mnemonic format.

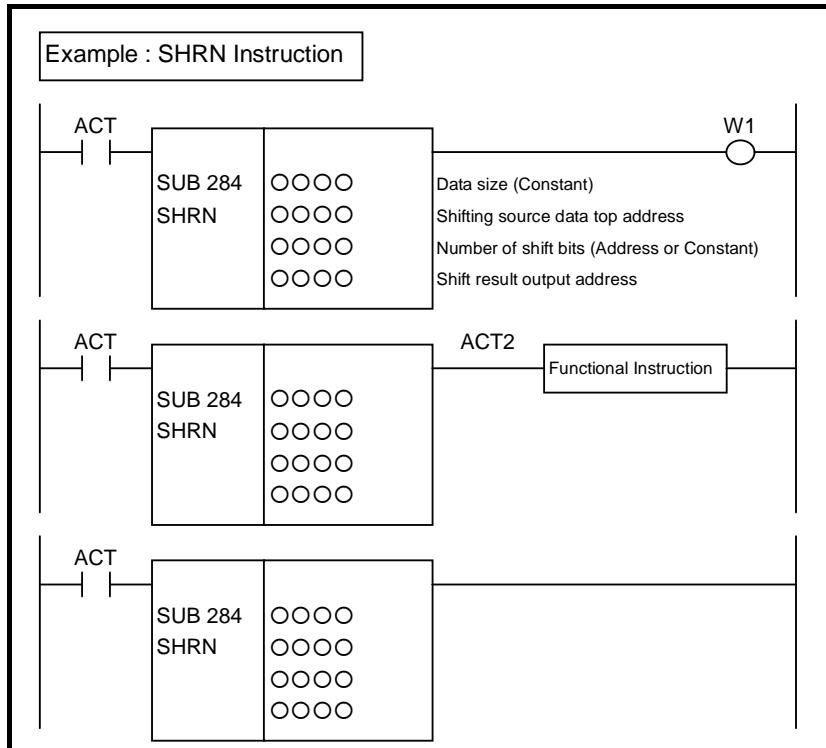


Fig. 4.8.16 (b) Format of SHRN instruction

Table 4.8.16 Mnemonic of SHRN instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	284		SUB No. (SHRN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Shifting source data top address
5	(PRM)	0000		Number of shift bits (Address or Constant)
6	(PRM)	0000		Shift result output address
7	WRT	0000 .0		Shift out status output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Data size
Specify the number of bytes of data on which a bit shift operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that both of the area from "Shifting source data top address" and the area from "Shift result output address" may be arranged within valid address range.

(b) Shifting source data top address

Specify the start address of bit shift source data.

Specify a data size in "Data size" mentioned in (a) above.

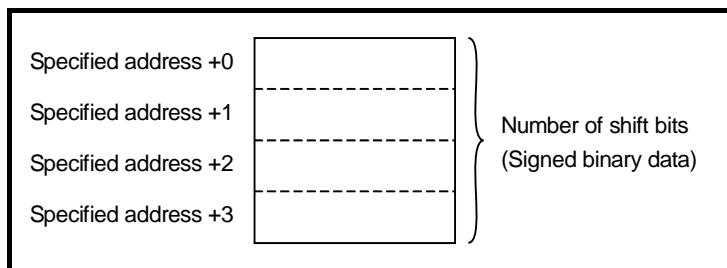
(c) Number of shift bits

By using 4-byte signed binary data, specify the number of bits to be shifted. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the shifting source data is shifted by a specified number of bits, and the result of shift operation is output to "Shift result output address". A value from 1 to ("Data size" × 8) may be specified. For example, if 6 is specified in "Data size", a value from 1 to 48 may be specified in this parameter. If a value greater than the valid range is specified, 0 is output to "Shift result output address", and W1=0 is set.

If 0 is specified, the shifting source data is output to "Shift result output address" without modification, and W1=0 is set.

If a negative value is specified in this parameter, the shifting source data is output to "Shift result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No shift operation in the opposite direction is performed.

In this parameter, a constant or a PMC memory address for storing data can be specified.



(d) Shift result output address

Specify the start address of an area to which the result of bit shift operation is to be output. The result of shift operation is output to memory of the same size as for shifting source data.

Output (W1)

W1=1: When the value of the last bit shifted out is 1

W1=0: When no shift operation is executed (ACT=0)

When the value of the last bit shifted out is 0

When 0 or a negative value is specified in "Number of shift bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.17 ROLB (Bit Rotation Left (1 Byte Length) : SUB 285)**ROLW (Bit Rotation Left (2 Bytes Length) : SUB 286)****ROLD (Bit Rotation Left (4 Bytes Length) : SUB 287)**

The Bit rotation left instruction rotates bit data to the left by a specified number of bits. The result of rotation operation is output to a specified address.

As indicated below, three types of Bit rotation left instructions are available according to the type of data to be operated. Rotation source bit data and the data at a rotation result output address are of the same data type.

Table 4.8.17 (a) Kinds of Bit rotation left instruction

	Instruction name	SUB No.	Data type
1	ROLB	285	1 byte length data
2	ROLW	286	2 bytes length data
3	ROLD	287	4 bytes length data

When data is rotated by 5 bits:

Rotation source data = R100
 Number of rotation bits = 5
 Rotation result output address = R102

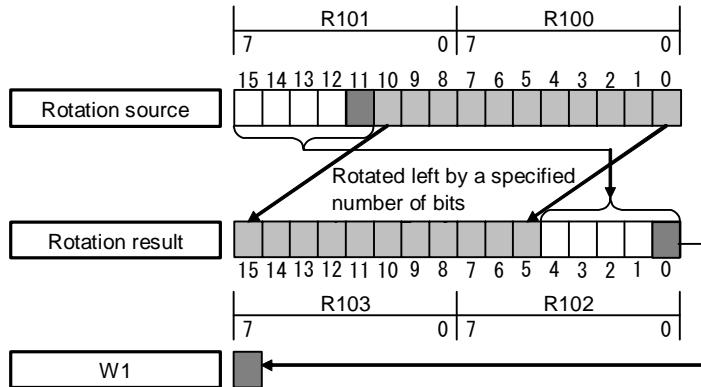


Fig. 4.8.17 (a) Example of ROLW instruction

The value of bit 0 after rotation is output to W1.

If 0 is specified in "Number of rotation bits", the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.17(b) shows the ladder format and Table 4.8.17(b) shows the mnemonic format.

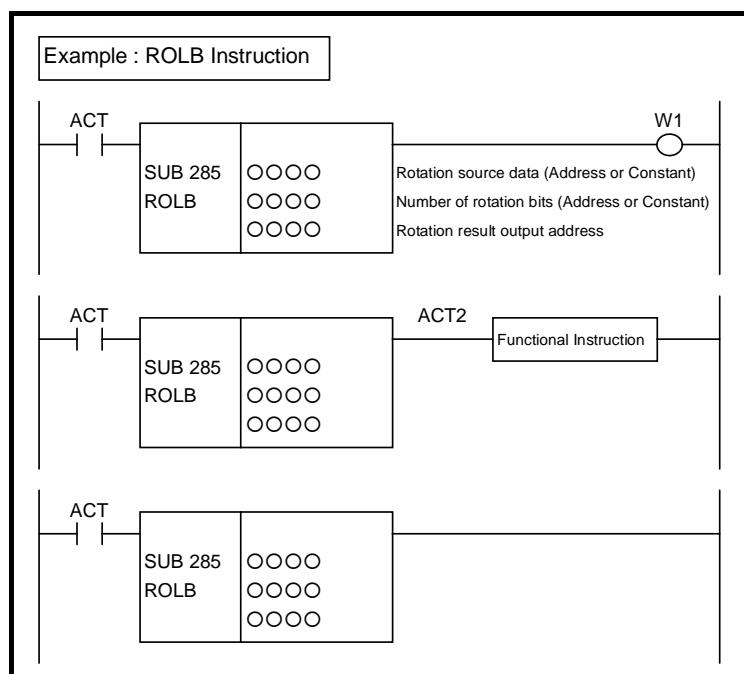


Fig. 4.8.17 (b) Format of ROLB, ROLW, ROLD instruction

**Table 4.8.17(b) Mnemonic of ROLB, ROLW, ROLD instruction
Mnemonic format**

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition			
					ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		285	SUB No. (ROLB instruction)				
3	(PRM)	0000		Rotation source data (Address or Constant)				
4	(PRM)	0000		Number of rotation bits (Address or Constant)				
5	(PRM)	0000		Rotation result output address				
6	WRT	0000 .0		Last rotation bit output				W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Rotation source data

Specify bit rotation source data. In this parameter, a constant or a PMC memory address for storing data can be specified.
Specify data by signed binary data. A value within the following range may be specified:

Instruction name	Available value
ROLB	-128 to 127
ROLW	-32768 to 32767
ROLD	-2147483648 to 2147483647

- (b) Number of rotation bits

By using signed binary data, specify the number of bits to be rotated. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the data specified in "Rotation source data" is rotated by a specified number of bits, and the result of rotation operation is output to "Rotation result output address". If 0 is specified, the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set.
If a negative value is specified in this parameter, the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No rotation operation in the opposite direction is performed.
In this parameter, a constant or a PMC memory address for storing data can be specified.
If an address is specified in this parameter, specify "Number of rotation bits" by using signed binary data of the same size as for data type handled by each instruction. For example, with the ROLW instruction, specify "Number of rotation bits" by using 2-byte signed binary data.

- (c) Rotation result output address

Specify the address to which the result of rotation operation is to be output. The result of rotation operation is output to memory of the same size as for "Rotation source data".

Output (W1)

- W1=1: When the value of bit 0 after rotation is 1
- W1=0: When no rotation operation is executed (ACT=0)
 - When the value of bit 0 after rotation is 0
 - When 0 or a negative value is specified in "Number of rotation bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.18 ROLN (Bit Rotation Left (Arbitrary Bytes Length) : SUB 288)

The Bit rotation left instruction rotates bit data to the left by a specified number of bits. The result of rotation operation is output to a specified address.

The Bit rotation left instruction performs a bit rotation operation on a bit string of a specified data size. Rotation source data and the result of rotation operation are of the same data size.

When data is rotated by 5 bits:

Data size	= 3
Rotation source data top address	= R100
Number of rotation bits	= 5
Rotation result output address	= R103

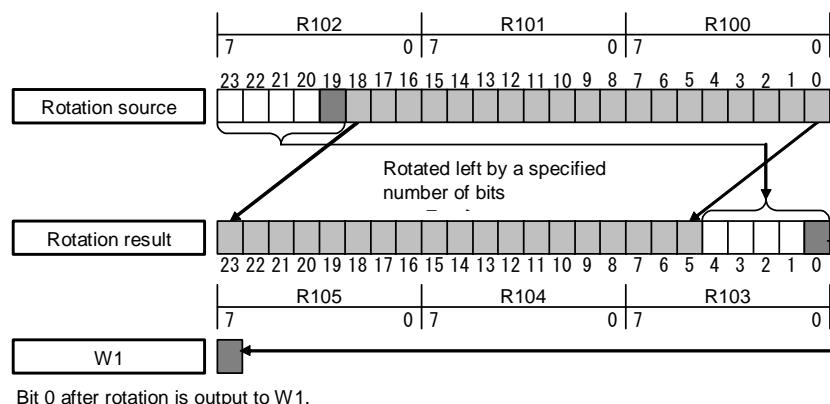


Fig. 4.8.18 (a) Example of ROLN instruction

The value of bit 0 after rotation is output to W1.

If 0 is specified in "Number of rotation bits", the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.18(b) shows the ladder format and Table 4.8.18 shows the mnemonic format.

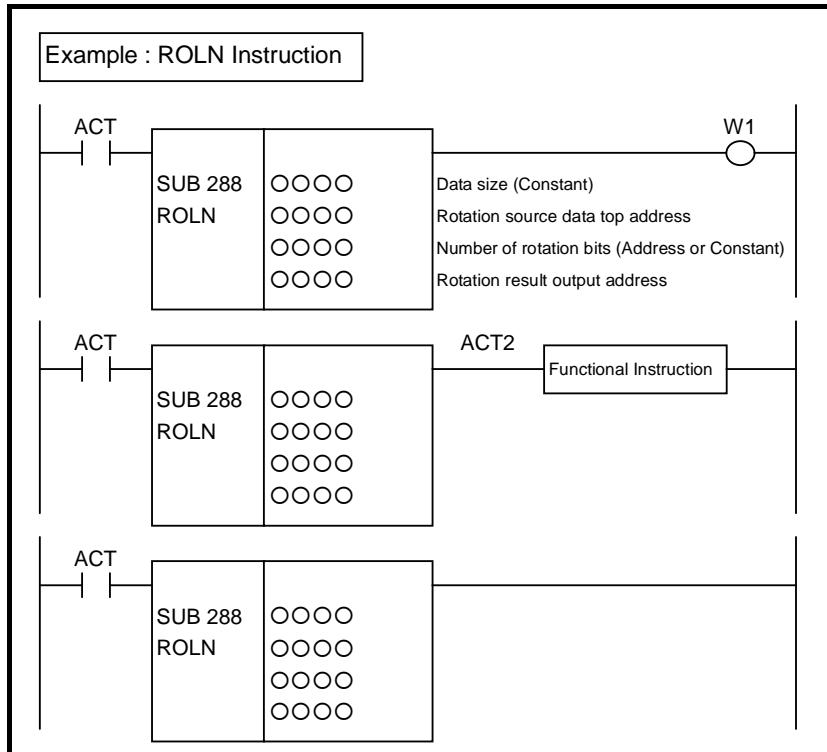


Fig. 4.8.18 (b) Format of ROLN instruction

Table 4.8.18 Mnemonic of ROLN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	288		SUB No. (ROLN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Rotation source data top address
5	(PRM)	0000		Number of rotation bits (Address or Constant)
6	(PRM)	0000		Rotation result output address
7	WRT	0000 .0		Last rotation bit output

Memory status of control condition

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Data size
Specify the number of bytes of data on which a bit rotation operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that both of the area from "Rotation source data top address" and the area from "Rotation result output address" may be arranged within valid address range.

- (b) Rotation source data top address

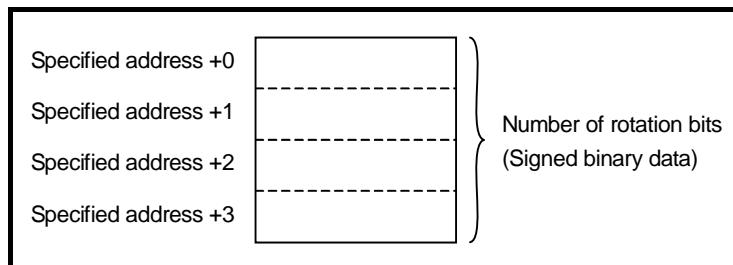
Specify the start address of rotation source data. Specify a data size in "Data size" mentioned in (a) above.

(c) Number of rotation bits

By using 4-byte signed binary data, specify the number of bits to be rotated. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the rotation source data is rotated by a specified number of bits, and the result of rotation operation is output to "Rotation result output address". A value from 1 to ("Data size" × 8) may be specified. For example, if 6 is specified in "Data size", a value from 1 to 48 may be specified in this parameter. If a value greater than the valid range is specified, the number of specified bits is divided by the value obtained by "Data size" × 8 then a rotation operation is performed using the remainder as the specified number of bits. If 0 is specified, the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set.

If a negative value is specified in this parameter, the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No rotation operation in the opposite direction is performed.

In this parameter, a constant or a PMC memory address for storing data can be specified.



(d) Rotation result output address

Specify the start address of an area to which the result of rotation operation is to be output. The result of rotation operation is output to memory of the same size as for rotation source data.

Output (W1)

W1=1: When the value of bit 0 after rotation is 1

W1=0: When no rotation operation is executed (ACT=0)

When the value of bit 0 after rotation is 0

When 0 or a negative value is specified in "Number of rotation bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.19 RORB (Bit Rotation Right (1 Byte Length) : SUB 289) RORW (Bit Rotation Right (2 Bytes Length) : SUB 290) RORD (Bit Rotation Right (4 Bytes Length) : SUB 291)

The Bit rotation right instruction rotates bit data to the right by a specified number of bits. The result of rotation operation is output to a specified address.

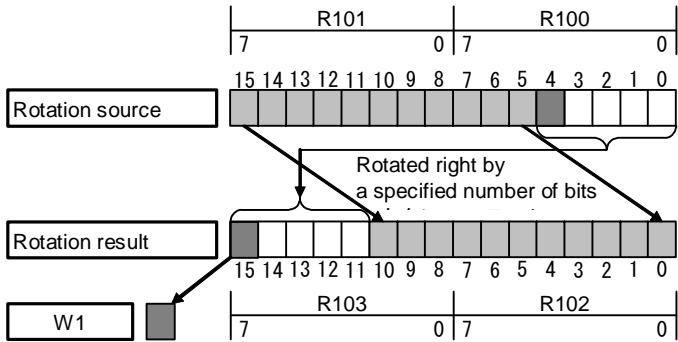
As indicated below, three types of Bit rotation right instructions are available according to the type of data to be operated. Rotation source bit data and the data at a rotation result output address are of the same data type.

Table 4.8.19 (a) Kinds of Bit rotation right instruction

	Instruction name	SUB No.	Data type
1	RORB	289	1 byte length data
2	RORW	290	2 bytes length data
3	RORD	291	4 bytes length data

When data is rotated by 5 bits:

Rotation source data = R100
 Number of rotation bits = 5
 Rotation result output address = R102



The most significant bit after rotation is output to W1.

Fig. 4.8.19 (a) Example of RORW instruction

The value of the most significant bit (bit 15 in the example above) after rotation is output to W1.

If 0 is specified in "Number of rotation bits", the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.19(b) shows the ladder format and Table 4.8.19(b) shows the mnemonic format.

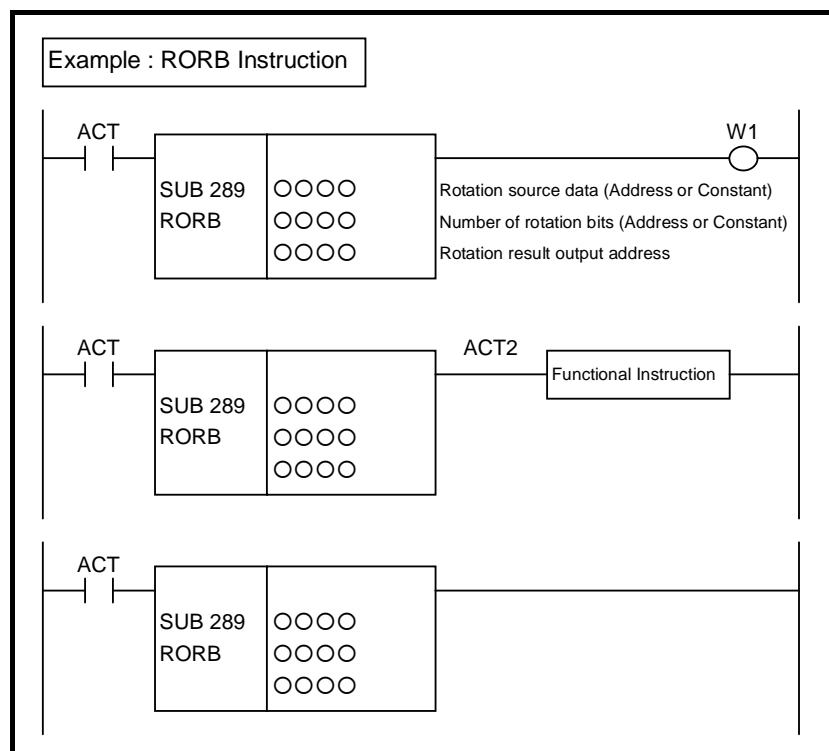


Fig. 4.8.19 (b) Format of RORB, RORW, RORD instruction

**Table 4.8.19(b) Mnemonic of RORB, RORW, RORD instruction
Mnemonic format**

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
					ST3 ST2 ST1 ST0
1	RD	0000 .0		ACT	
2	SUB	289		SUB No. (RORB instruction)	
3	(PRM)	0000		Rotation source data (Address or Constant)	
4	(PRM)	0000		Number of rotation bits (Address or Constant)	
5	(PRM)	0000		Rotation result output address	
6	WRT	0000 .0		Last rotation bit output	ACT W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Rotation source data

Specify bit rotation source data. In this parameter, a constant or a PMC memory address for storing data can be specified.
Specify data by signed binary data. A value within the following range may be specified:

Instruction name	Available value
RORB	-128 to 127
RORW	-32768 to 32767
RORD	-2147483648 to 2147483647

- (b) Number of rotation bits

By using signed binary data, specify the number of bits to be rotated. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the data specified in "Rotation source data" is rotated by a specified number of bits, and the result of rotation operation is output to "Rotation result output address". If 0 is specified, the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set.
If a negative value is specified in this parameter, the data specified in "Rotation source data" is output to "Rotation result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No rotation operation in the opposite direction is performed.
In this parameter, a constant or a PMC memory address for storing data can be specified.
If an address is specified in this parameter, specify "Number of rotation bits" by using signed binary data of the same size as for data type handled by each instruction. For example, with the RORW instruction, specify "Number of rotation bits" by using 2-byte signed binary data.

- (c) Rotation result output address

Specify the address to which the result of rotation operation is to be output. The result of rotation operation is output to memory of the same size as for "Rotation source data".

Output (W1)

- W1=1: When the value of the most significant bit after rotation is 1
- W1=0: When no rotation operation is executed (ACT=0)
 - When the value of the most significant bit after rotation is 0
 - When 0 or a negative value is specified in "Number of rotation bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.20 RORN (Bit Rotation Right (Arbitrary Bytes Length) : SUB 292)

The Bit rotation right instruction rotates bit data to the right by a specified number of bits. The result of rotation operation is output to a specified address.

The Bit rotation right instruction performs a bit rotation operation on a bit string of a specified data size. Rotation source data and the result of rotation operation are of the same data size.

When data is rotated by 5 bits:

Data size	= 3
Rotation source data top address	= R100
Number of rotation bits	= 5
Rotation result output address	= R103

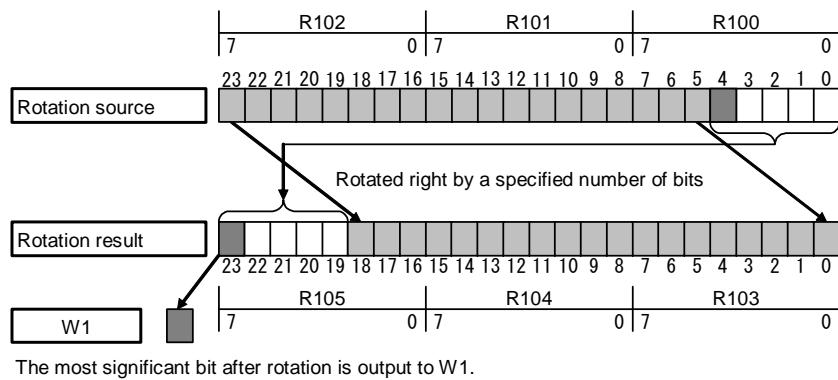


Fig. 4.8.20 (a) Example of RORN instruction

The value of the most significant bit (bit 15 in the example above) at the last address after rotation is output to W1.

If 0 is specified in "Number of rotation bits", the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set.

Format

Fig. 4.8.20(b) shows the ladder format and Table 4.8.20 shows the mnemonic format.

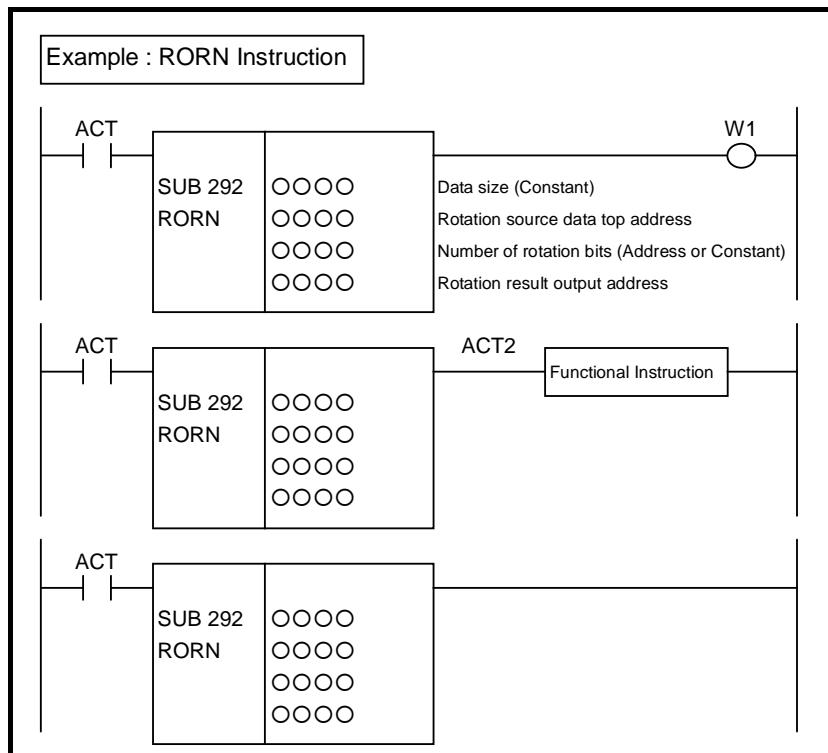


Fig. 4.8.20(b) Format of RORN instruction

Table 4.8.20 Mnemonic of RORN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
1	RD	0000 .0		ACT	ST3 ST2 ST1 ST0
2	SUB	292		SUB No. (RORN instruction)	
3	(PRM)	0000		Data size (Constant)	
4	(PRM)	0000		Rotation source data top address	
5	(PRM)	0000		Number of rotation bits (Address or Constant)	
6	(PRM)	0000		Rotation result output address	
7	WRT	0000 .0		Last rotation bit output	W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data size

Specify the number of bytes of data on which a bit rotation operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that both of the area from "Rotation source data top address" and the area from "Rotation result output address" may be arranged within valid address range.

(b) Rotation source data top address

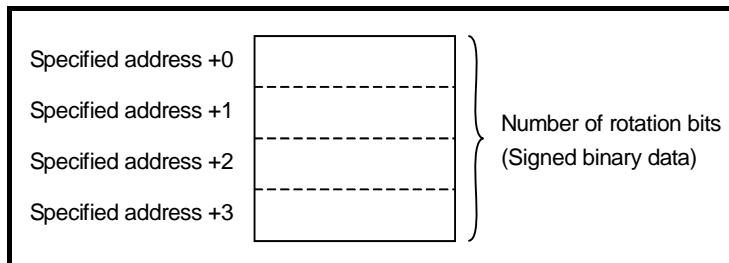
Specify the start address of rotation source data. Specify a data size in "Data size" mentioned in (a) above.

(c) Number of rotation bits

By using 4-byte signed binary data, specify the number of bits to be rotated. In this parameter, specify 0 or a greater value. If a value greater than 0 is specified, the rotation source data is rotated by a specified number of bits, and the result of rotation operation is output to "Rotation result output address". A value from 1 to ("Data size" × 8) may be specified. For example, if 6 is specified in "Data size", a value from 1 to 48 may be specified in this parameter. If a value greater than the valid range is specified, the number of specified bits is divided by the value obtained by "Data size" × 8 then a rotation operation is performed using the remainder as the specified number of bits. If 0 is specified, the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set.

If a negative value is specified in this parameter, the rotation source data is output to "Rotation result output address" without modification, and W1=0 is set as in the case where 0 is specified in this parameter. No rotation operation in the opposite direction is performed.

In this parameter, a constant or a PMC memory address for storing data can be specified.



(d) Rotation result output address

Specify the start address of an area to which the result of rotation operation is to be output. The result of rotation operation is output to memory of the same size as for rotation source data.

Output (W1)

W1=1: When the value of the most significant bit at the last address after rotation is 1

W1=0: When no rotation operation is executed (ACT=0)

When the value of the most significant bit at the last address after rotation is 0

When 0 or a negative value is specified in "Number of rotation bits"

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.21 BSETB (Bit Set (1 Byte Length) : SUB 293)**BSETW (Bit Set (2 Bytes Length) : SUB 294)****BSETD (Bit Set (4 Bytes Length) : SUB 295)**

The Bit set instruction sets the bit at a specified bit position to ON (=1).

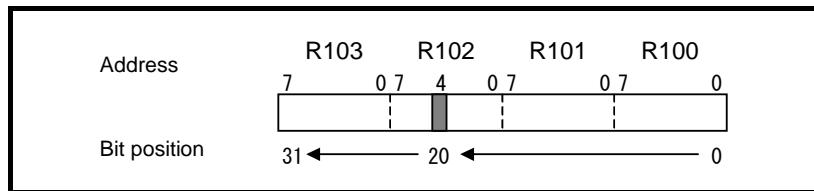
As indicated below, three types of Bit set instructions are available according to the type of data to be operated.

Table 4.8.21 (a) Kinds of Bit set instruction

	Instruction name	SUB No.	Data type	Useful range of bit position
1	BSETB	293	1 byte length data	0 to 7
2	BSETW	294	2 bytes length data	0 to 15
3	BSETD	295	4 bytes length data	0 to 31

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of a specified address.

For example, if R100 is specified in "Data address", and 20 is specified in "Bit position" with the BSETD instruction, R102.4 is set to ON.



Format

Fig. 4.8.21(a) shows the ladder format and Table 4.8.21(b) shows the mnemonic format.

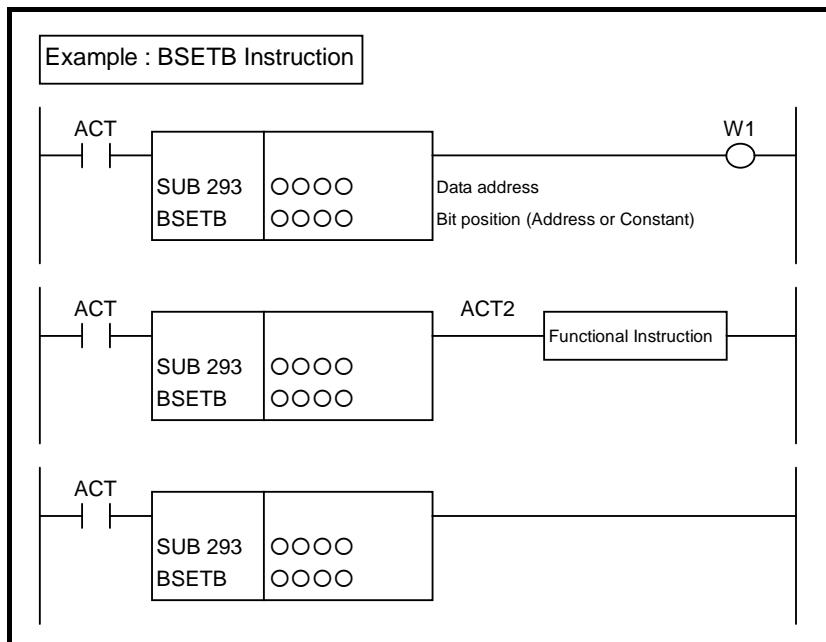


Fig. 4.8.21 (a) Format of BSETB, BSETW, BSETD instruction

Table 4.8.21(b) Mnemonic of BSETB, BSETW, BSETD instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB	293		SUB No. (BSETB instruction)				
3	(PRM)	0000		Data address				
4	(PRM)	0000		Bit position (Address or Constant)				
5	WRT	0000 .0		Normal end output				W1

Control conditions

(a) Input signal (ACT)

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

(a) Data address

Specify the address of data on which a bit operation is to be performed.

(b) Bit position

Specify the position of a bit to be set to ON. Specify 0 or a greater value in "Bit position". For the range of values specifiable in "Bit position", see Table 4.8.21 (a), "Kinds of Bit set instruction". In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified in this parameter, specify "Bit position" by using signed binary data of the same size as for data type handled by each instruction. For example, with the BSETW instruction, specify "Bit position" by using 2-byte signed binary data.

If a value not within the valid range is specified in this parameter, the data is not modified, and W1=0 is set.

Output (W1)

W1=1: When an operation is terminated normally

W1=0: When no operation is executed (ACT=0)

When "Bit position" is not within the valid range

NOTE

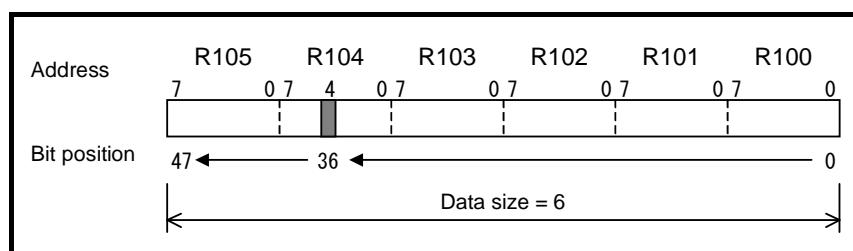
W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.22 BSETN (Bit Set (Arbitrary Bytes Length) : SUB 296)

The Bit set instruction sets the bit at a specified bit position in a bit string of the size specified in "Data size" to ON (=1).

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of the start address.

For example, if R100 is specified in "Data top address", 6 is specified in "Data size", and 36 is specified in "Bit position", R104.4 is set to ON.



Format

Fig. 4.8.22 shows the ladder format and Table 4.8.22 shows the mnemonic format.

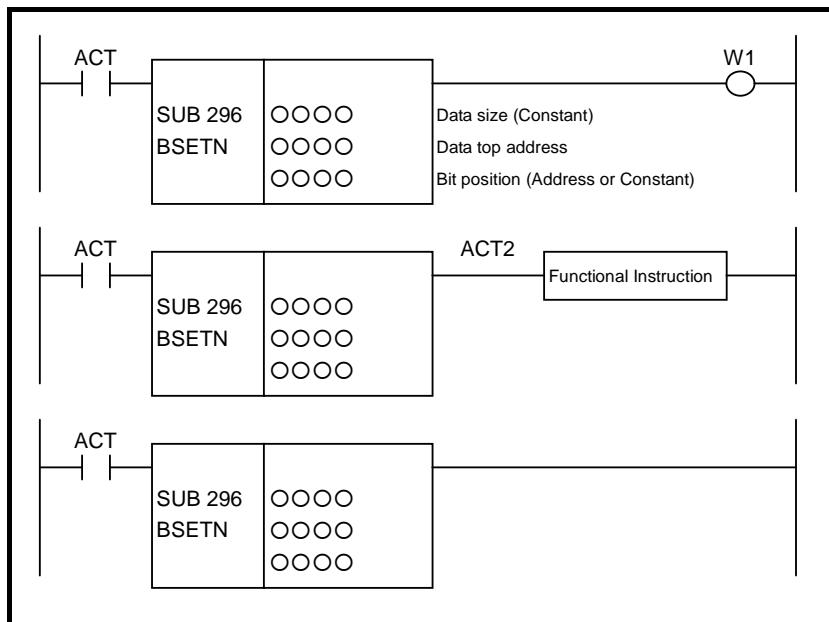


Fig. 4.8.22 Format of BSETN instruction

Table 4.8.22 Mnemonic of BSETN instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .O		ACT
2	SUB	296		SUB No. (BSETN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Data top address
5	(PRM)	0000		Bit position (Address or Constant)
6	WRT	0000 .O		Normal end output

ST3	ST2	ST1	ST0
			ACT
			▼
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Data size
Specify the number of bytes of data on which a bit operation is to be performed. A value from 1 to 256 may be specified.

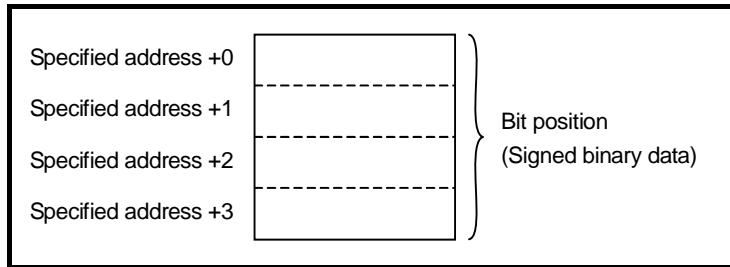
NOTE

Please specify a valid number to the "Data size", so that the area from "Data top address" may be arranged within valid address range.

- (b) Data top address
Specify the start address of data.
- (c) Bit position

By using 4-byte signed binary data, specify the position of a bit to be set to ON. A value from 0 to ("Data size" × 8 - 1) may be specified. For example, if 6 is specified in "Data size", a value from 0 to 47 may be specified.

In this parameter, a constant or a PMC memory address for storing data can be specified.



If a value not within the valid range is specified in this parameter, the data is not modified, and W1=0 is set.

Output (W1)

W1=1: When an operation is terminated normally

W1=0: When no operation is executed (ACT=0)

When "Bit position" is not within the valid range

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.23 BRSTB (Bit Reset (1 Byte Length) : SUB 297) BRSTW (Bit Reset (2 Bytes Length) : SUB 298) BRSTD (Bit Reset (4 Bytes Length) : SUB 299)

The Bit reset instruction sets the bit at a specified bit position to OFF (=0).

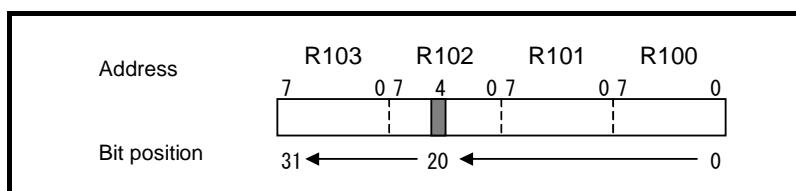
As indicated below, three types of Bit reset instructions are available according to the type of data to be operated.

Table 4.8.23 (a) Kinds of Bit reset instruction

	Instruction name	SUB No.	Data type	Useful range of bit position
1	BRSTB	297	1 byte length data	0 to 7
2	BRSTW	298	2 bytes length data	0 to 15
3	BRSTD	299	4 bytes length data	0 to 31

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of a specified address.

For example, if R100 is specified in "Data address", and 20 is specified in "Bit position" with the BRSTD instruction, R102.4 is set to OFF.



Format

Fig. 4.8.23 shows the ladder format and Table 4.8.23(b) shows the mnemonic format.

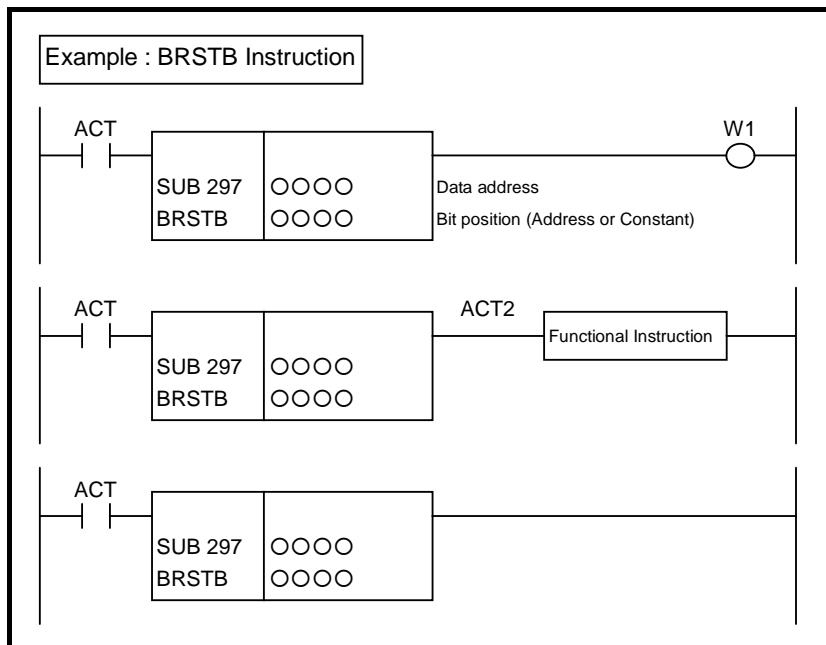


Fig. 4.8.23 Format of BRSTB, BRSTW, BRSTD instruction

Table 4.8.23(b) Mnemonic of BRSTB, BRSTW, BRSTD instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	297		SUB No. (BRSTB instruction)
3	(PRM)	0000		Data address
4	(PRM)	0000		Bit position (Address or Constant)
5	WRT	0000 .0		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

(a) Input signal (ACT)

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

(a) Data address

Specify the address of data on which a bit operation is to be performed.

(b) Bit position

Specify the position of a bit to be set to OFF. Specify 0 or a greater value in "Bit position". For the range of values specifiable in "Bit position", see Table 4.8.23 (a), "Kinds of Bit reset instruction".

In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified in this parameter, specify "Bit position" by using signed binary data of the same size as for data type handled by each instruction. For example, with the BRSTW instruction, specify "Bit position" by using 2-byte signed binary data.

If a value not within the valid range is specified in this parameter, the data is not modified, and W1=0 is set.

Output (W1)

W1=1: When an operation is terminated normally

W1=0: When no operation is executed (ACT=0)

When "Bit position" is not within the valid range

NOTE

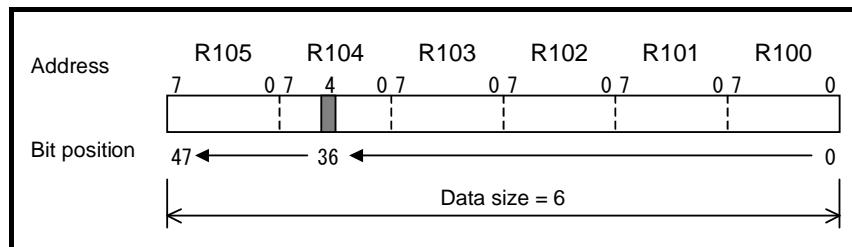
W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.24 BRSTN (Bit Reset (Arbitrary Bytes Length) : SUB 300)

The Bit reset instruction sets the bit at a specified bit position in a bit string of the size specified in "Data size" to OFF (=0).

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of the start address.

For example, if R100 is specified in "Data top address", 6 is specified in "Data size", and 36 is specified in "Bit position", R104.4 is set to OFF.



Format

Fig. 4.8.24 shows the ladder format and Table 4.8.24 shows the mnemonic format.

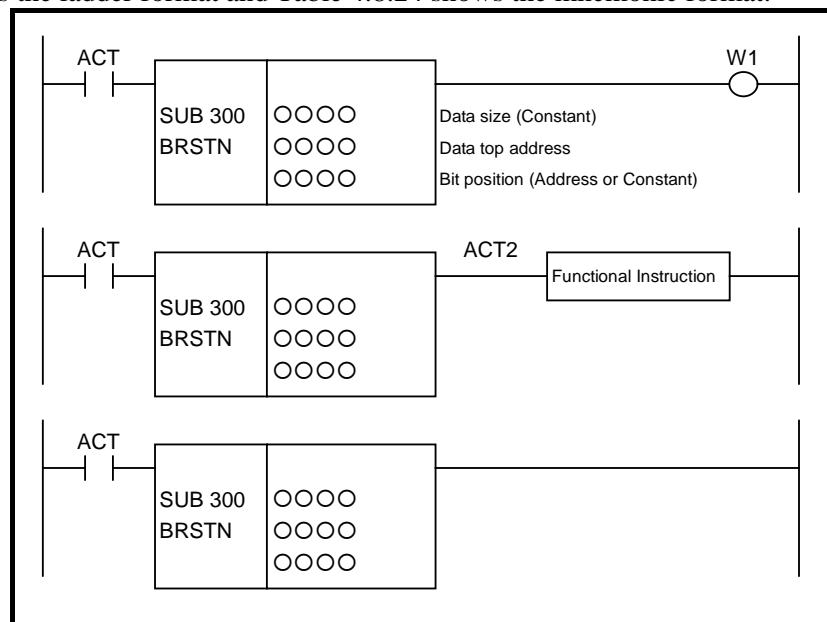


Fig. 4.8.24 Format of BRSTN instruction

Table 4.8.24 Mnemonic of BRSTN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition
ST3	ST2	ST1	ST0		
1	RD	0000 .0		ACT	
2	SUB	300		SUB No. (BRSTN instruction)	
3	(PRM)	0000		Data size (Constant)	
4	(PRM)	0000		Data top address	
5	(PRM)	0000		Bit position (Address or Constant)	
6	WRT	0000 .0		Normal end output	W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

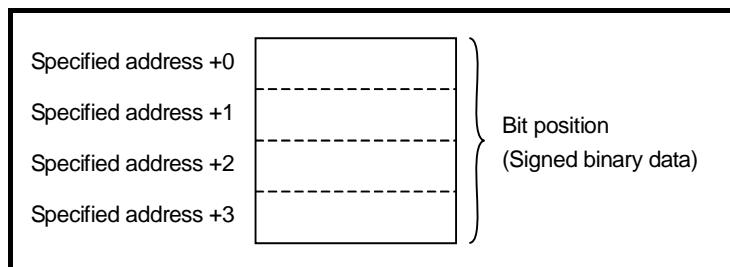
Parameters

- (a) Data size
 - Specify the number of bytes of data on which a bit operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that the area from "Data top address" may be arranged within valid address range.

- (b) Data top address
 - Specify the start address of data.
- (c) Bit position
 - By using 4-byte signed binary data, specify the position of a bit to be set to OFF. A value from 0 to ("Data size" × 8 - 1) may be specified. For example, if 6 is specified in "Data size", a value from 0 to 47 may be specified.
 - In this parameter, a constant or a PMC memory address for storing data can be specified.



If a value not within the valid range is specified in this parameter, the data is not modified, and W1=0 is set.

Output (W1)

- W1=1: When an operation is terminated normally
- W1=0: When no operation is executed (ACT=0)
 - When "Bit position" is not within the valid range

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.25 BTSTB (Bit Test (1 Byte Length) : SUB 301) BTSTW (Bit Test (2 Bytes Length) : SUB 302) BTSTD (Bit Test (4 Bytes Length) : SUB 303)

The Bit test instruction outputs the value of the bit at a specified bit position.

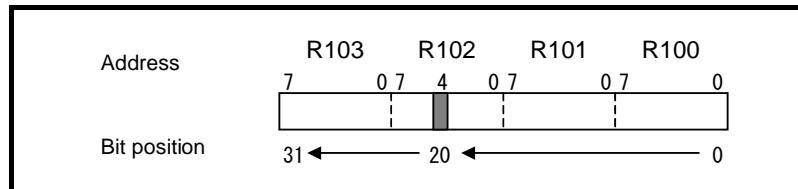
As indicated below, three types of Bit test instructions are available according to the type of data to be operated.

Table 4.8.25 (a) Kinds of Bit test instruction

	Instruction name	SUB No.	Data type	Useful range of bit position
1	BTSTB	301	1 byte length data	0 to 7
2	BTSTW	302	2 bytes length data	0 to 15
3	BTSTD	303	4 bytes length data	0 to 31

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of a specified address.

For example, if R100 is specified in "Data address", and 20 is specified in "Bit position" with the BTSTD instruction, the state of R102.4 is output.



Format

Fig. 4.8.25 shows the ladder format and Table 4.8.25(b) shows the mnemonic format.

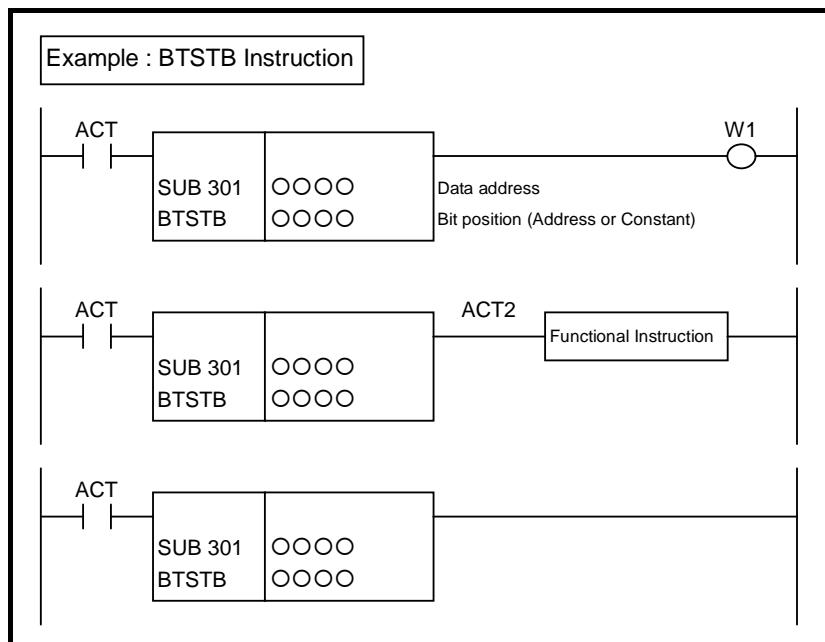


Fig. 4.8.25 Format of BTSTB, BTSTW, BTSTD instruction

Table 4.8.25 (b) Mnemonic of BTSTB, BTSTW, BTSTD instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition			
					ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		301	SUB No. (BTSTB instruction)				
3	(PRM)	0000		Data address				
4	(PRM)	0000		Bit position (Address or Constant)				▼
5	WRT	0000 .0		Bit status output				W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data address

Specify the address of data on which a bit operation is to be performed.
- (b) Bit position

Specify the position of a bit whose state is to be output. Specify 0 or a greater value in "Bit position". For the range of values specifiable in "Bit position", see Table 4.8.25 (a), "Kinds of Bit test instruction".

In this parameter, a constant or a PMC memory address for storing data can be specified.

If an address is specified in this parameter, specify "Bit position" by using signed binary data of the same size as for data type handled by each instruction. For example, with the BTSTW instruction, specify "Bit position" by using 2-byte signed binary data.

If a value not within the valid range is specified in this parameter, W1=0 is set.

Output (W1)

- W1=1: Specified bit is 1.
- W1=0: Specified bit is 0.
 - When no operation is executed (ACT=0)
 - When "Bit position" is not within the valid range

NOTE

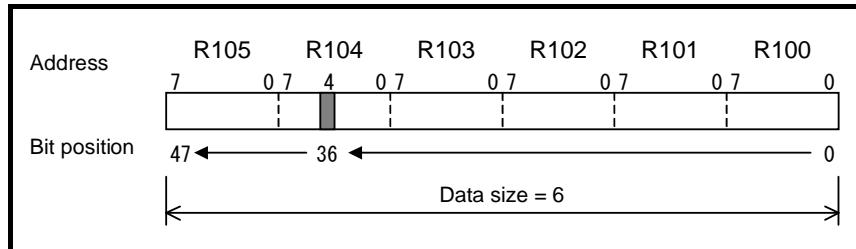
W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.26 BTSTN (Bit Test (Arbitrary Bytes Length) : SUB 304)

The Bit test instruction outputs the value of the bit at a specified bit position in a bit string of the size specified in "Data size".

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of the start address.

For example, if R100 is specified in "Data top address", 6 is specified in "Data size", and 36 is specified in "Bit position", the bit state of R104.4 is output.



Format

Fig. 4.8.26 shows the ladder format and Table 4.8.26 shows the mnemonic format.

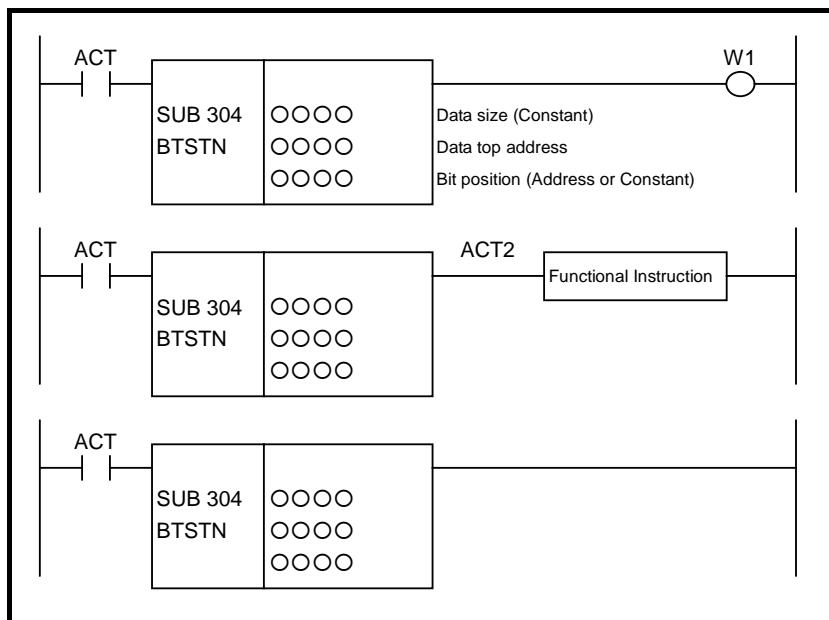


Fig. 4.8.26 Format of BTSTN instruction

Table 4.8.26 Mnemonic of BTSTN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	304		SUB No. (BTSTN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Data top address
5	(PRM)	0000		Bit position (Address or Constant)
6	WRT	0000 .0		Normal end output

Memory status of control condition

ST3	ST2	ST1	ST0
			ACT
			▼
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Data size
Specify the number of bytes of data on which a bit operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that the area from "Data top address" may be arranged within valid address range.

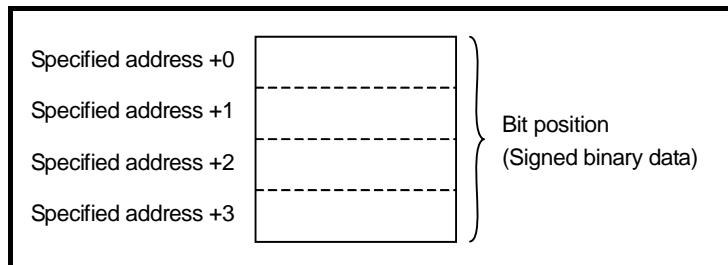
- (b) Data top address

Specify the start address of data.

- (c) Bit position

By using 4-byte signed binary data, specify the position of a bit whose state is to be output. A value from 0 to ("Data size" \times 8 - 1) may be specified. For example, if 6 is specified in "Data size", a value from 0 to 47 may be specified.

In this parameter, a constant or a PMC memory address for storing data can be specified.



If a value not within the valid range is specified in this parameter, W1=0 is set.

Output (W1)

W1=1: Specified bit is 1.

W1=0: Specified bit is 0.

When no operation is executed (ACT=0)

When "Bit position" is not within the valid range

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.27 BPOSB (Bit Search (1 Byte Length) : SUB 305)**BPOSW (Bit Search (2 Bytes Length) : SUB 306)****BPOSD (Bit Search (4 Bytes Length) : SUB 307)**

The Bit search instruction acquires the bit position of a bit set to ON (=1).

As indicated below, three types of Bit search instructions are available according to the type of data to be operated.

Table 4.8.27 (a) Kinds of Bit search instruction

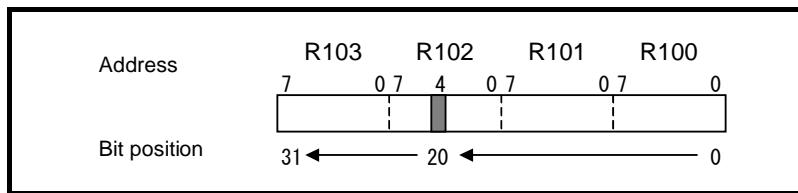
	Instruction name	SUB No.	Data type	Useful range of bit position
1	BPOSB	305	1 byte length data	0 to 7
2	BPOSW	306	2 bytes length data	0 to 15
3	BPOSD	307	4 bytes length data	0 to 31

Bits are searched in the order from bit 0 to bit 7 at the data top address then bits are searched in the order from bit 0 to bit 7 at the next address. In this way, bit search operation is further performed for up to bit 7 of the last address.

The bit position of the bit that is first found to be ON is output.

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of the start address.

For example, if R100 is specified in "Data address" with the BPOSD instruction, and only R102.4 is set to ON, 20 is output to "Bit position output address".



Format

Fig. 4.8.27 shows the ladder format and Table 4.8.27(b) shows the mnemonic format.

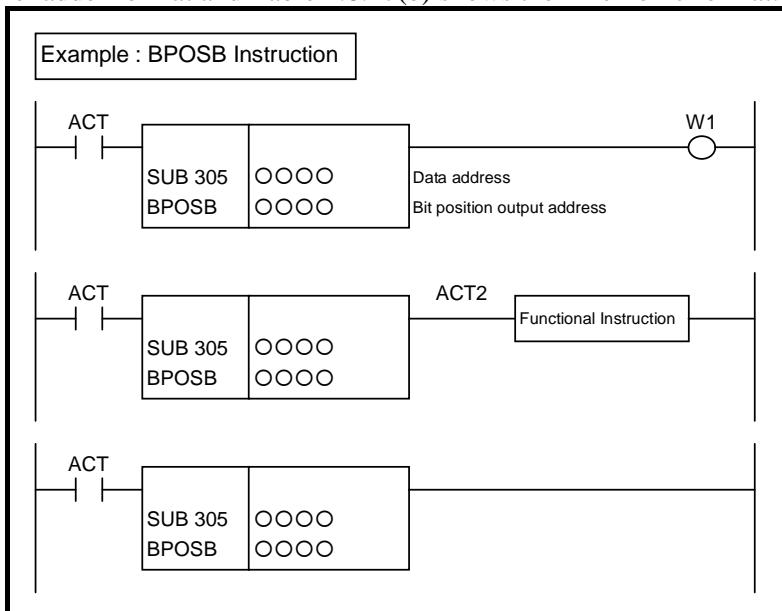


Fig. 4.8.27 Format of BPOSB, BPOSW, BPOSD instruction

Table 4.8.27(b) Mnemonic of BPOSB, BPOSW, BPOSD instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	305		SUB No. (BPOSB instruction)
3	(PRM)	0000		Data address
4	(PRM)	0000		Bit position output address
5	WRT	0000 .0		Normal end output

Memory status of control condition

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data address
 - Specify the address of data on which a bit operation is to be performed.
- (b) Bit position output address
 - Specify the address to which the position of a bit found to be ON as the result of search operation is to be output. Starting at the specified address, a bit position is output by using signed binary data

of the same size as for data type handled by each instruction. For example, with the BPOSW instruction, a bit position is output by using 2-byte signed binary data.

The start bit position number is 0.

If no bit is found to be ON as the result of search operation, -1 is output, and W1=0 is set.

Output (W1)

W1=1 is set when the instruction is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when there is no bit set to ON.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.28 BPOSN (Bit Search (Arbitrary Bytes Length) : SUB 308)

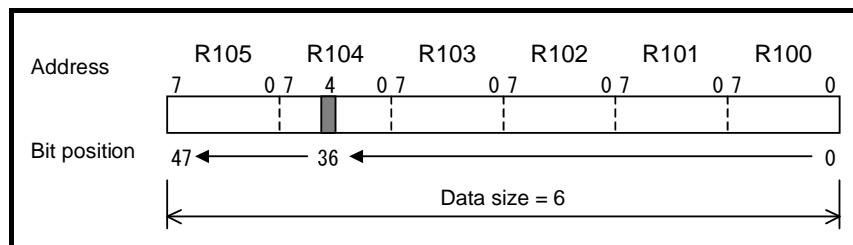
The Bit search instruction acquires the bit position of a bit set to ON (=1) in a bit string of the size specified in "Data size".

Bits are searched in the order from bit 0 to bit 7 at "Data top address" then bits are searched in the order from bit 0 to bit 7 at the next address. In this way, bit search operation is further performed for up to bit 7 of the last address.

The bit position of the bit that is first found to be ON is output.

A bit position is identified by sequentially counting bit positions, starting with 0, from the least significant bit of the start address.

For example, if R100 is specified in "Data top address", 6 is specified in "Data size", and only R104.4 is set to ON, 36 is output to "Bit position output address".



Format

Fig. 4.8.28 shows the ladder format and Table 4.8.28 shows the mnemonic format.

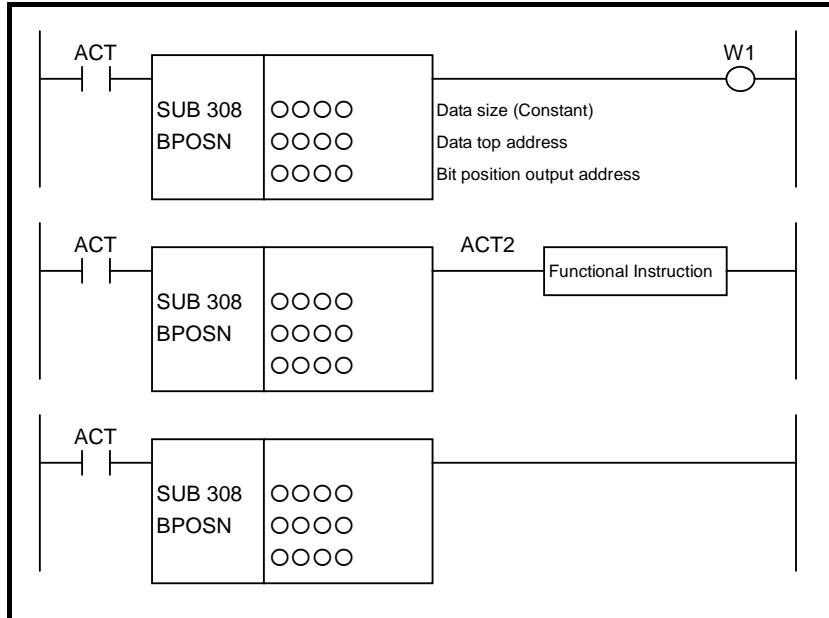


Fig. 4.8.28 Format of BPOSN instruction

Table 4.8.28 Mnemonic of BPOSN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	308		SUB No. (BPOSN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Data top address
5	(PRM)	0000		Bit position output address
6	WRT	0000 .0		Normal end output

Memory status of control condition

ST3	ST2	ST1	ST0
			ACT
			▼
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

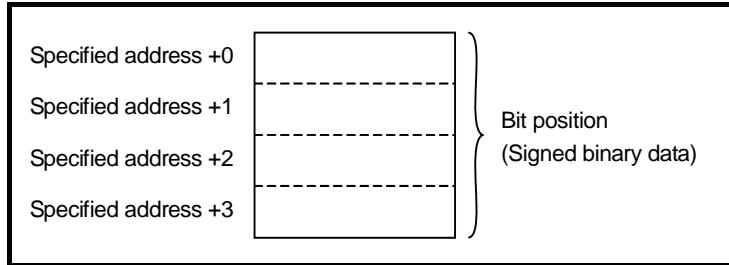
Parameters

- (a) Data size
Specify the number of bytes of data on which a bit operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that the area from "Data top address" may be arranged within valid address range.

- (b) Data top address
Specify the start address of data.
- (c) Bit position output address
Specify the address to which a found bit position is to be output. A bit position is output by using 4-byte signed binary data.
The start bit position number is 0.
If no bit is found to be ON as the result of search operation, -1 is output, and W1=0 is set.



Output (W1)

W1=1: When an operation is terminated normally

W1=0: When no operation is executed (ACT=0)

When there is no bit set to ON

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.29 BCNTB (Bit Count (1 Byte Length) : SUB 309) BCNTW (Bit Count (2 Bytes Length) : SUB 310) BCNTD (Bit Count (4 Bytes Length) : SUB 311)

The Bit count instruction acquires the number of bits set to ON (=1).

As indicated below, three types of Bit count instructions are available according to the type of data to be operated.

Table 4.8.29 (a) Kinds of Bit count instruction

	Instruction name	SUB No.	Data type	Useful range of bit position
1	BCNTB	309	1 byte length data	0 to 7
2	BCNTW	310	2 bytes length data	0 to 15
3	BCNTD	311	4 bytes length data	0 to 31

Format

Fig. 4.8.29 shows the ladder format and Table 4.8.29(b) shows the mnemonic format.

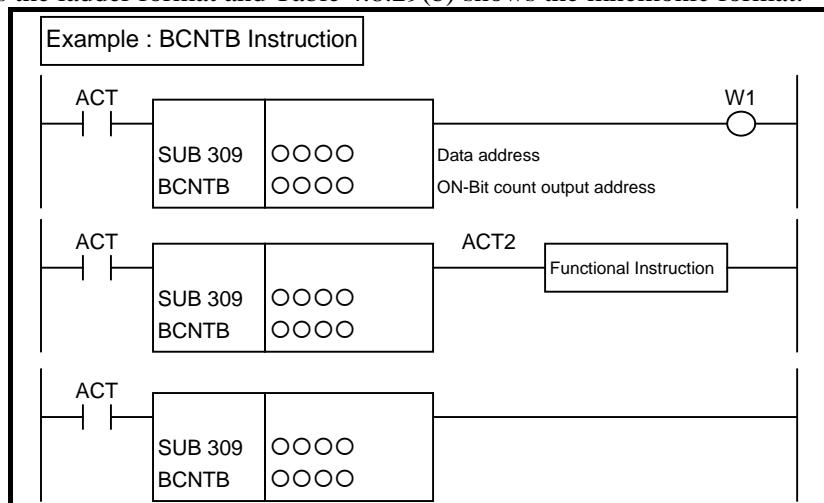


Fig. 4.8.29 Format of BCNTB, BCNTW, BCNTD instruction

**Table 4.8.29(b) Mnemonic BCNTB, BCNTW, BCNTD instruction
Mnemonic format**

Step number	Instruction	Address No.	Bit No.	Remarks	Memory status of control condition			
					ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		309	SUB No. (BCNTB instruction)				
3	(PRM)	0000		Data address				
4	(PRM)	0000		ON-Bit count output address				
5	WRT	0000 .O		Normal end output				W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data address

Specify the address of data on which a bit operation is to be performed.
- (b) ON-Bit count output address

Specify the address to which the number of bits set to ON is to be output. Starting at the specified address, the number of bits set to ON is output to memory of the same size as for data type handled by each instruction. For example, with the BCNTW instruction, the number of bits set to ON is output by using 2-byte signed binary data.

Output (W1)

- W1=1 : When an operation is executed (ACT=1)
- W1=0 : When no operation is executed (ACT=0)

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.8.30 BCNTN (Bit Count (Arbitrary Bytes Length) : SUB 312)

The Bit count instruction acquires the number of bits set to ON (=1) in a bit string of the size specified in "Data size".

Format

Fig. 4.8.30 shows the ladder format and Table 4.8.30 shows the mnemonic format.

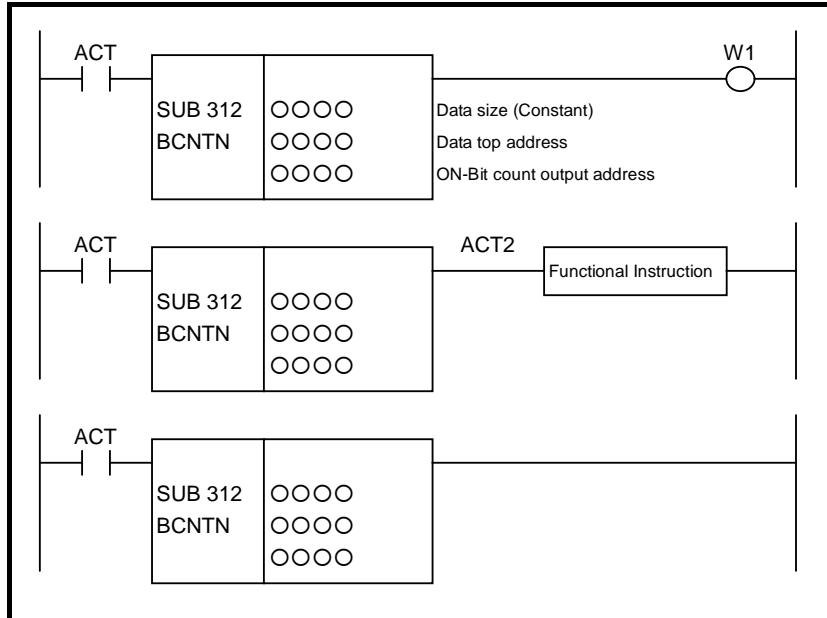


Fig. 4.8.30 Format of BCNTN instruction

Table 4.8.30 Mnemonic of BCNTN instruction
Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	312		SUB No. (BCNTN instruction)
3	(PRM)	0000		Data size (Constant)
4	(PRM)	0000		Data top address
5	(PRM)	0000		ON-Bit count output address
6	WRT	0000 .0		Normal end output

Memory status of control condition

ST3	ST2	ST1	ST0
			ACT
			▼
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

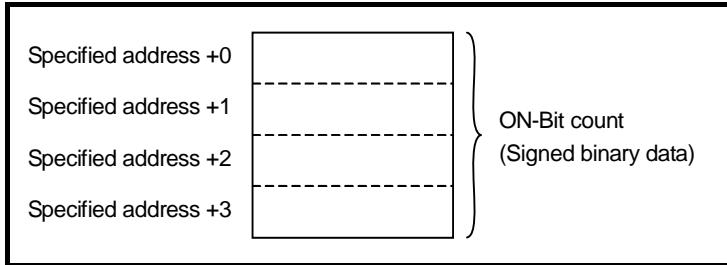
Parameters

- (a) Data size
Specify the number of bytes of data on which a bit operation is to be performed. A value from 1 to 256 may be specified.

NOTE

Please specify a valid number to the "Data size", so that both of the area from "Data top address" and the area from "Shift result output address" may be arranged within valid address range.

- (b) Data top address
Specify the start address of data.
- (c) ON-Bit count output address
Specify the address to which the number of bits set to ON is to be output. The number of bits set to ON is output by using 4-byte signed binary data.



Output (W1)

W1=1 : When an operation is executed (ACT=1)

W1=0 : When no operation is executed (ACT=0)

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.9 CODE CONVERSION

The following types of code conversion instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	COD	7	Code conversion
2	CODB	27	Binary code conversion
3	DCNV	14	Data conversion
4	DCNVB	31	Extended data conversion
5	DEC	4	Decoding
6	DECB	25	Binary decoding
7	TBCDB	313	Binary to BCD conversion (1 byte length)
8	TBCDW	314	Binary to BCD conversion (2 bytes length)
9	TBCDD	315	Binary to BCD conversion (4 bytes length)
10	FBCDB	316	BCD to Binary conversion (1 byte length)
11	FBCDW	317	BCD to Binary conversion (2 bytes length)
12	FBCDD	318	BCD to Binary conversion (4 bytes length)

4.9.1 COD (Code Conversion: SUB 7)

Converts BCD codes into an arbitrary two- or four-digits BCD numbers. For code conversion shown in Fig. 4.9.1 (a) the conversion input data address, conversion table, and convert data output address must be provided.

Set a table address, in which the data to be retrieved from the conversion table is contained, to conversion table input data address in a two-digits BCD number. The conversion table is entered in sequence with the numbers to be retrieved in the two- or four-digits number. The contents of the conversion table of the number entered in the conversion input data address is output to the convert data output address. As shown in Fig. 4.9.1 (a), when 3 is entered in the conversion input data address, the contents 137 located at 3 in the conversion table is output to the convert data output address.

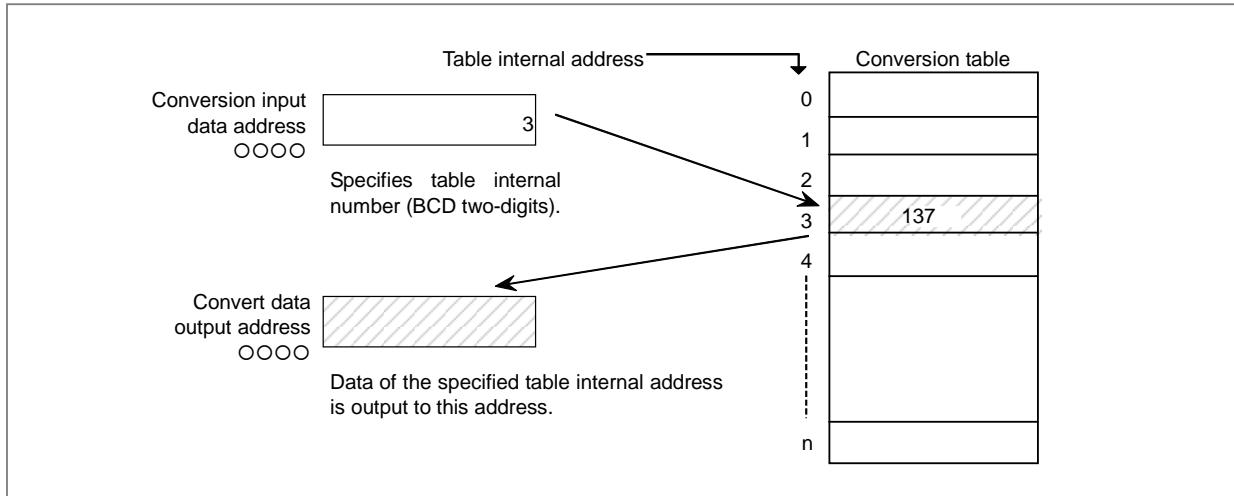


Fig. 4.9.1 (a) Code conversion diagram

Format

Fig. 4.9.1 (b) shows the ladder format and Table 4.9.1 shows the mnemonic format.

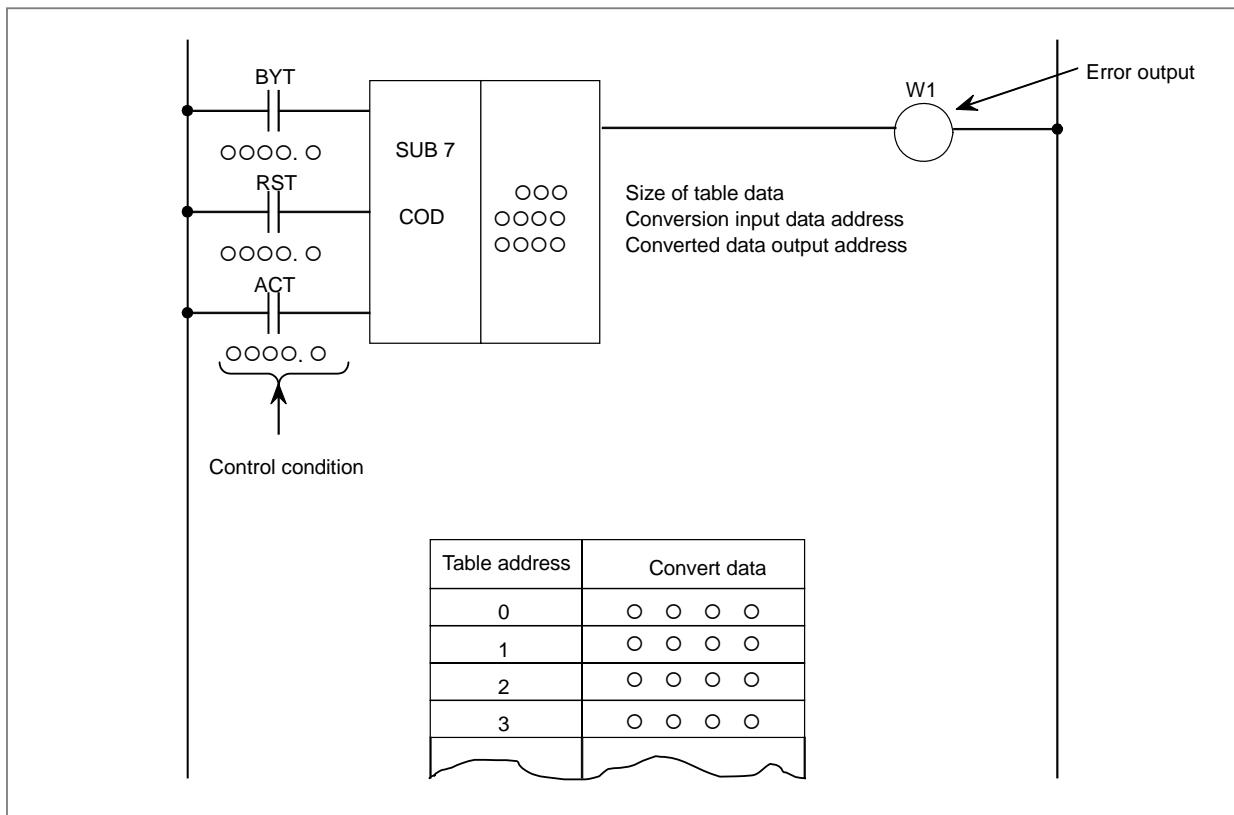


Fig. 4.9.1 (b) Format of COD instruction

Table 4.9.1 Mnemonic of COD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		RST			BYT	RST
3	RD.STK	0000 .0		ACT		BYT	RST	ACT
4	SUB		7	COD instruction				
5	(PRM)	000		Size of table data				
6	(PRM)	0000		Conversion input data address				
7	(PRM)	0000		Convert data output address				
8	(PRM)	0000		Convert data at table address 0				
9	(PRM)	0000		Convert data at table address 1				
:	:	:		:				
7 + n	(PRM)	0000		Data at (n (convert data at table address) - 1)				
7 + n + 1	WRT	0000 .0		Error output				W1

Control conditions

- (a) Specify the data size. (BYT)
 - BYT=0: Specifies that the conversion table data is to be BCD two digits.
 - BYT=1: Specifies that the conversion table data is to be BCD four digits.
- (b) Error output reset (RST)
 - RST=0: Disable reset
 - RST=1: Sets error output W1 to 0 (resets).
- (c) Execution command (ACT)
 - ACT=0: The COD instruction is not executed. W1 does not change.
 - ACT=1: Executed.

Parameters

- (a) Size of table data
 - A conversion table data address from 00 to 99 can be specified.
 - Specify n+1 as the size of table when n is the last table internal number.
- (b) Conversion input data address
 - The conversion table address includes a table address in which converted data is loaded. Data in the conversion table can be retrieved by specifying a conversion table address.
 - One byte (BCD 2-digit) is required for this conversion input data address.
- (c) Convert data output address
 - The convert data output address is the address where the data stored in the table is to be output.
 - The convert data BCD two digits in size, requires only a 1-byte memory at the convert data output address.
 - Convert data BCD four digits in size, requires a 2-byte memory at the convert data output address.

Error output (W1)

If an error occurs in the conversion input address during execution of the COD instruction, W1=1 to indicate an error.

For example, W1=1 results if a number exceeding the table size specified in the sequence program is specified as the conversion input address. When W1=1, it is desirable to effect an appropriate interlock, such as having the error lamp on the machine tool operator's panel light or stopping axis feed.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Conversion data table

The size of the conversion data table is from 00 to 99.

The conversion data can be either BCD two digits or four digits, which is specified depends on the control conditions.

4.9.2 CODB (Binary Code Conversion: SUB 27)

This instruction converts data in binary format to an optional binary format 1-byte, 2-byte, or 4-byte data.

Conversion input data address, conversion table, and conversion data output address are necessary for data conversion; as shown in Fig. 4.9.2 (a).

Compared to the "COD Function Instruction", this CODB function instruction handles numerical data 1-, 2- and 4-byte length binary format data, and the conversion table can be extended to maximum 256.

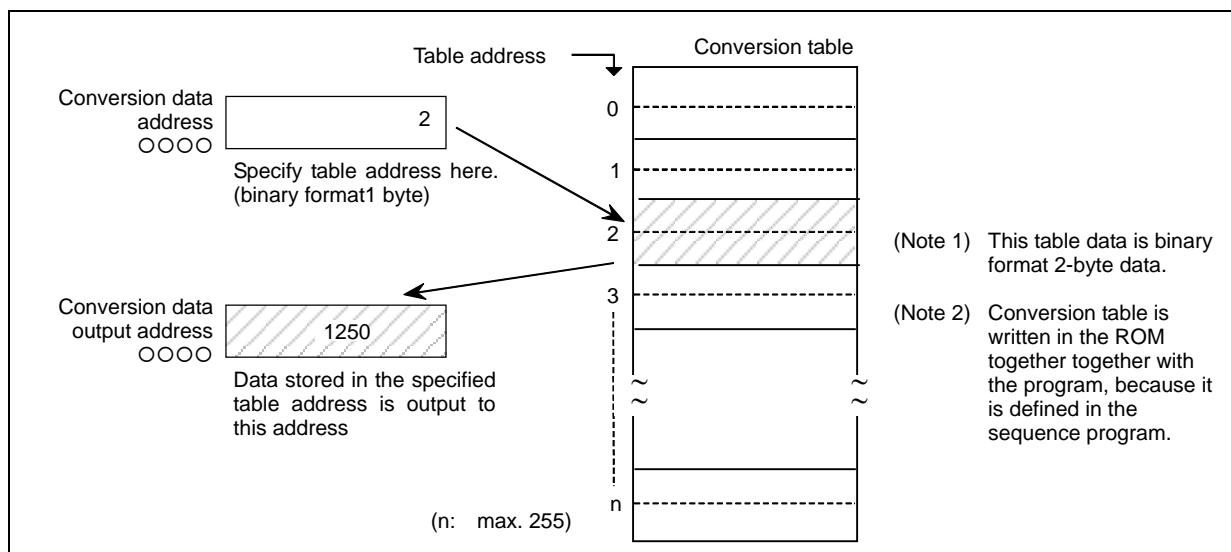


Fig. 4.9.2 (a) Code conversion diagram

Format

Fig. 4.9.2 (b) shows the ladder format and Table 4.9.2 shows the mnemonic format.

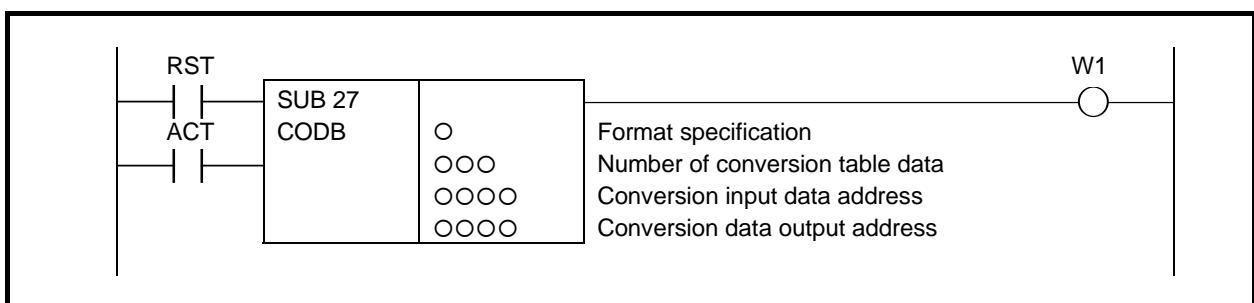


Fig. 4.9.2 (b) Format of CODB instruction

Table 4.9.2 Mnemonic of CODB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		RST				RST
2	RD.STK	0000 .O		ACT			RST	ACT
3	SUB		27	CODB instruction				
4	(PRM)		O	Format specification				
5	(PRM)	000		Size of table data				
6	(PRM)	0000		Conversion input data address				
7	(PRM)	0000		Convert data output address				
8	(PRM)	0000		Convert data at table address 0				
9	(PRM)	0000		Convert data at table address 1				
:	:	:		:				
7 + n	(PRM)	0000		Data at (n (convert data at table address) - 1)				
7 + n + 1	WRT	0000 .O		Timer relay output				W1

Control conditions

- (a) Reset (RST)
RST=0: Do not reset.
RST=1: Reset error output W1 (W1=0).
- (b) Activate command (ACT)
ACT=0: Do not execute CODB instruction
ACT=1: Execute CODB instruction.

Parameters

- (a) Format specification
Designates binary numerical size in the conversion table.
 - 1: Numerical data is binary 1-byte data.
 - 2: Numerical data is binary 2-byte data.
 - 4: Numerical data is binary 4-byte data.
- (b) Number of conversion table data
Designates size of conversion table. 256 (0 to 255) data can be made.
- (c) Conversion input data address
Data in the conversion data table can be taken out by specifying the table number. The address specifying the table number is called conversion input data address, and 1-byte memory is required from the specified address.
- (d) Conversion data output address
Address to output data stored in the specified table number is called conversion data output address. Memory of the byte length specified in the format designation is necessary from the specified address.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Conversion data table

Size of the conversion data table is maximum 256 (from 0 to 255).

Error output (W1)

If the table number in the conversion input data address exceeds the number of the conversion table data when executing the CODB instruction, W1=1.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.9.3 DCNV (Data Conversion: SUB 14)

Converts binary-code into BCD-code and vice versa.

Format

Fig. 4.9.3 shows the ladder format and Table 4.9.3 shows the mnemonic format.

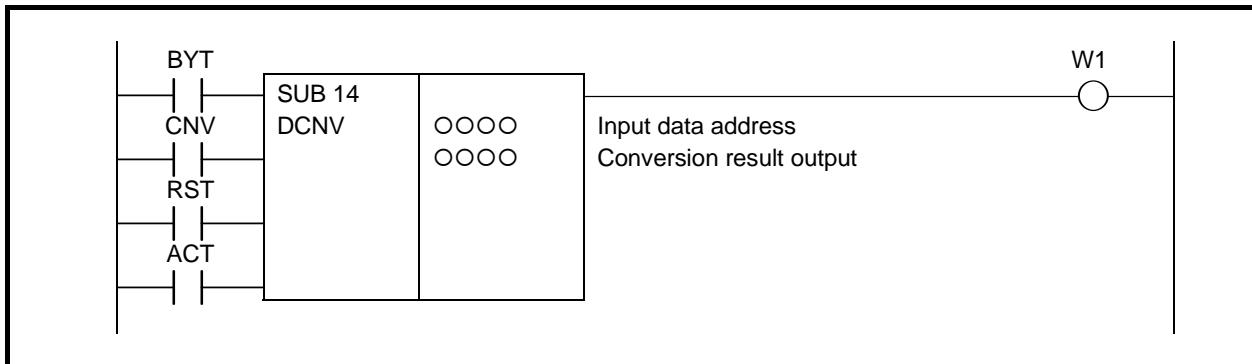


Fig. 4.9.3 Format of DCNV instruction

Table 4.9.3 Mnemonic of DCNV instruction

Mnemonic format

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		BYT
2	RD.STK	0000 .0		CNV
3	RD.STK	0000 .0		RST
4	RD.STK	0000 .0		ACT
5	SUB	14		DCNV instruction
6	(PRM)	0000		Input data address
7	(PRM)	0000		Conversion result output address
8	WRT	0000 .0		W1 error output

Memory status of control condition

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

Control conditions

- (a) Specify data size. (BYT)

BYT=0: Process data in length of one byte (8 bits)

BYT=1: Process data in length of two bytes (16 bits)

- (b) Specify the type of conversion (CNV)

CNV=0: Converts binary-code into BCD-code.

CNV=1: Converts BCD-code into binary-code.

- (c) Reset (RST)

RST=0: Disables reset.

RST=1: Resets error output W1. That is, setting RST to 1 when W1=1, makes W1=0.

(d) Execution command (ACT)

ACT=0: Data is not converted. W1 will not alter.

ACT=1: Data is converted.

Parameters

(a) Input data address

Specify the address of the input data

(b) Output address after conversion

Specify the address output data converted into BCD or binary type

Error output (W1)

W1=0: Normal

W1=1: Conversion error

The input data which should be BCD data, is binary data, or the data size (byte length) specified in advance is exceeded when converting binary data into BCD data.

 **CAUTION**

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.9.4 DCNVB (Extended Data Conversion: SUB 31)

This instruction converts 1, 2, and 4-byte binary code into BCD code or vice versa. To execute this instruction, you must preserve the necessary number of bytes in the memory for the conversion result output data.

Format

Fig. 4.9.4 shows the ladder format and Table 4.9.4 shows the mnemonic format.

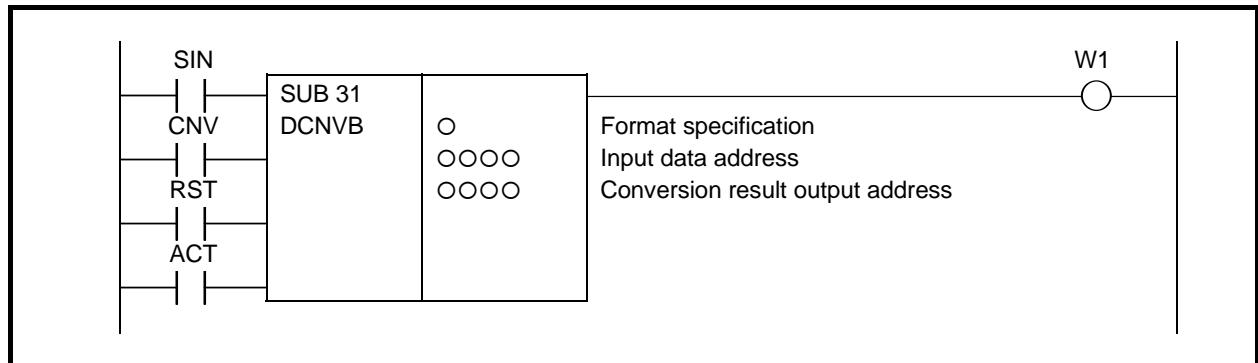


Fig. 4.9.4 Format of DCNVB instruction

Table 4.9.4 Mnemonic of DCNVB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		SIN				SIN
2	RD.STK	0000 .0		CNV			SIN	CNV
3	RD.STK	0000 .0		RST		SIN	CNV	RST
4	RD.STK	0000 .0		ACT	SIN	CNV	RST	ACT
5	SUB		31	DCNVB instruction				
6	(PRM)		0	Size of table data				
7	(PRM)	0000		Conversion input data address				
8	(PRM)	0000		Conversion data output address				
9	WRT	0000 .0		Error output				W1

Control conditions

- (a) Sign of the data to be converted (SIN)

This parameter is significant only when you are converting BCD data into binary coded data. It gives the sign of the BCD data.

Note that though it is insignificant when you are converting binary into BCD data, you cannot omit it.

SIN=0: Data (BCD code) to be input is positive.

SIN=1: Data (BCD code) to be input is negative.

- (b) Type of conversion (CNV)

CNV=0: Convert binary data into BCD data

CNV=1: Convert BCD data into binary data.

- (c) Reset (RST)

RST=0: Release reset

RST=1: Reset error output W1. In other words, set W1=0.

- (d) Execution command (ACT)

ACT=0: Data is not converted. The value of W1 remains unchanged.

ACT=1: Data is converted.

Parameters

- ### (a) Format specification

Specify data length (1,2, or 4 bytes).

Use the first digit of the parameter to specify byte length.

- 1: one byte
 - 2: two bytes
 - 4: four bytes

- (b) Input data address

Specify the address containing the input data address.

- (c) Address for the conversion result output

Specify the address to output the data converted to BCD or binary format.



Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

W1=0: Correct conversion

W1=1: Abnormally

The data to be converted is specified as BCD data but is found to be binary data, or the specified number of bytes cannot contain (and hence an overflow occurs) the BCD data into which a binary data is converted.

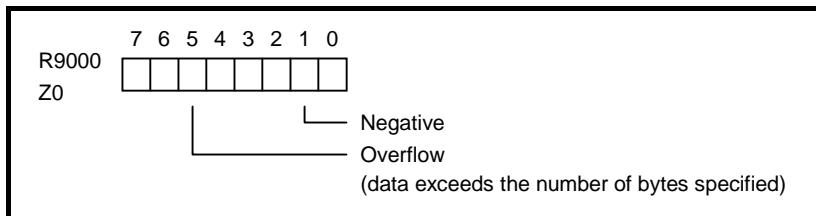
CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Operation output register (R9000, Z0)

This register is set with data on operation. If register bit 1 is on, they signify the following.

For the positive/negative signs when binary data is converted into BCD data, see R9000 or Z0.



4.9.5 DEC (Decode: SUB 4)

Outputs 1 when the two-digit BCD code signal is equal to a specified number, and 0 when not. Is used mainly to decode M or T function. The value type in this instruction is BCD.

Format

Fig. 4.9.5 (a) shows the ladder format and Table 4.9.5 (a) shows the mnemonic format.

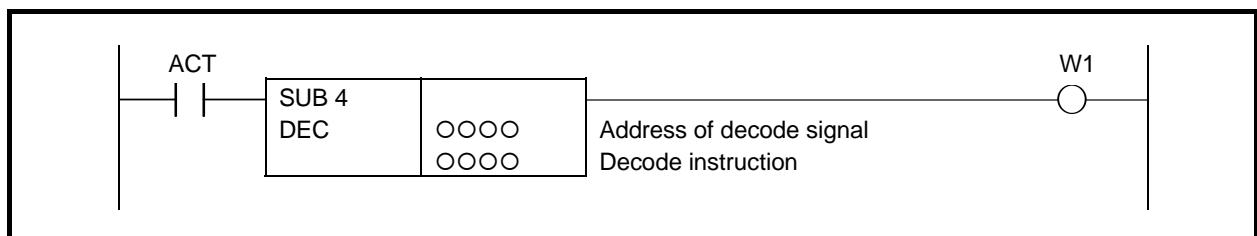


Fig. 4.9.5 (a) Format of DEC instruction

Table 4.9.5 (a) Mnemonic of DEC instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	DEC	0000		Code signal address				
3	(PRM)	0000		Decode specification				↓
4	WRT	0000 .0		W1, decoding result output				W1

The mnemonic-format instruction name "DEC" for step number 2 above may be abbreviated as "D".

Control condition

ACT=0: Turns the decoding result output off (W1).

ACT=1: Performs decoding.

When the specified number is equal to the code signal, W1=1; when not, W1=0.

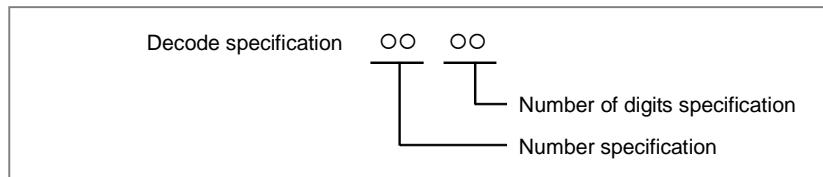
Parameters

(a) Code signal address

Specify the address containing two-digit BCD code signals.

(b) Decode specification

There are two paths, the number and the number of digits.



(i) Number:

Specify the decode number.

Must always be decoded in two digits.

(ii) Number of digits:

01: The high-order digit of two decimal digits is set to 0 and only the low-order digit is decoded.

10: The low-order digit is set to 0 and only the high-order digit is decoded.

11: Two decimal digits are decoded.

W1 (decoding result output)

W1 is 1 when the status of the code signal at a specified address is equal to a specified number, 0 when not. The address of W1 is determined by designer.

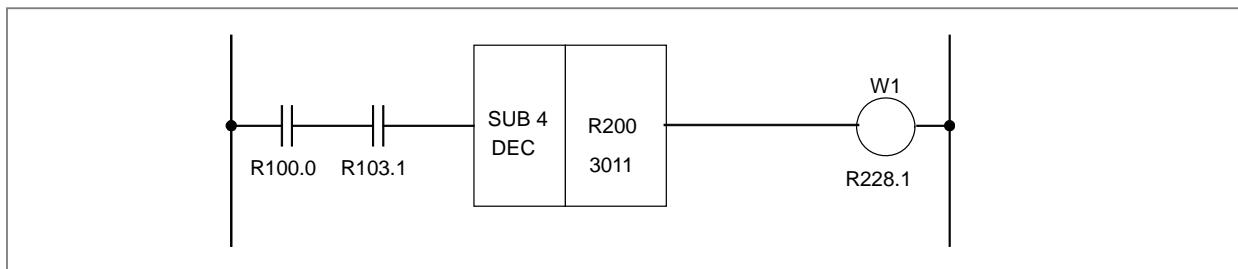


Fig. 4.9.5 (b) Ladder diagram using the DEC instruction

Table 4.9.5 (b) Mnemonic for Fig. 4.9.5 (b)

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	R100 .0		
2	AND	R103 .1		
3	DEC	R200		
4	(PRM)	3011		
5	WRT	R228 .1		

4.9.6 DECB (Binary Decoding: SUB 25)

DECB decodes one, two, or four-byte binary code data. When one of the specified eight consecutive numbers matches the code data, a logical high value (value 1) is set in the output data bit which corresponds to the specified number. When these numbers do not match, a logical low value (value 0) is set.

Use this instruction for decoding data of the M or T function.

There are two specifications - basic specification and extended specification - for setting the format specification parameter in the DECB instruction. The extended specification allows $8n$ consecutive values to be decoded at a time. For the details of the setting of a format specification parameter, see the description of parameters.

Format

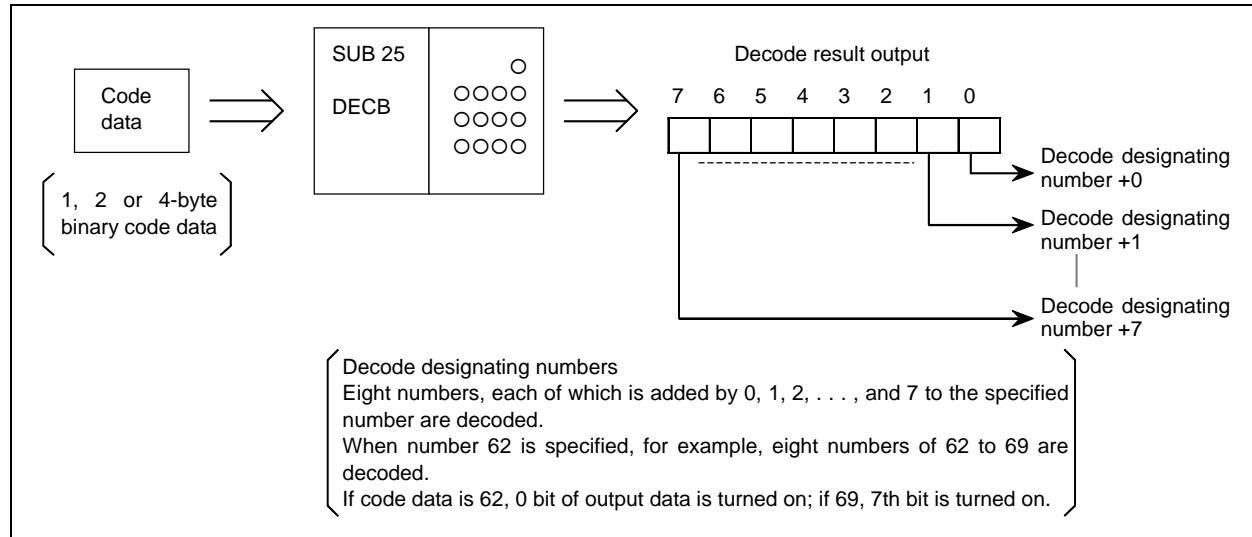


Fig. 4.9.6 (a) Function of DECB instruction (basic specification)

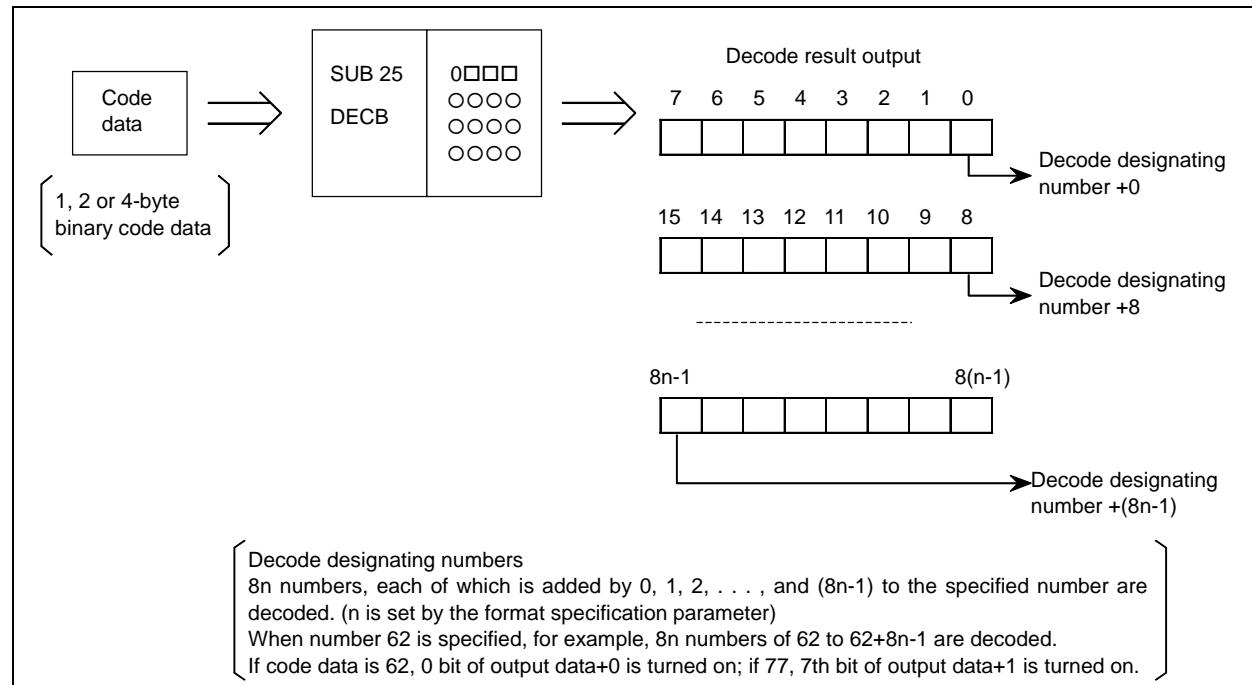


Fig. 4.9.6 (b) Function of DECB instruction (extended specification)

Figs. 4.9.6 (c) and (d) show the ladder formats and Tables 4.9.6 (a) and (b) show the mnemonic formats.

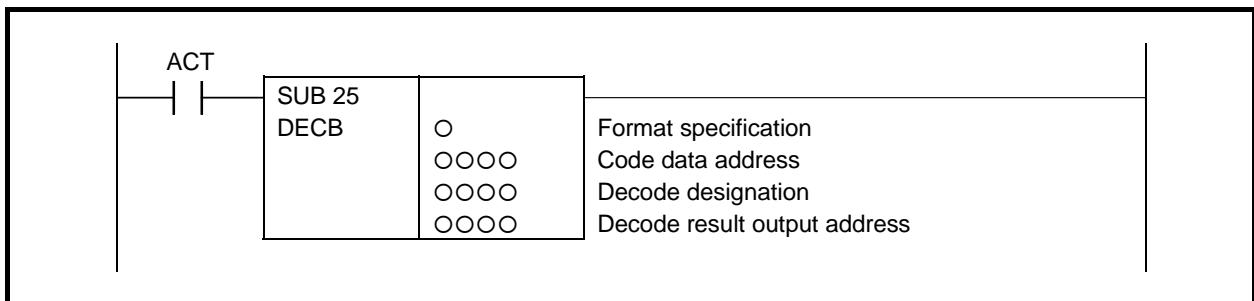


Fig. 4.9.6 (c) Format of DECB instruction (basic specification)

Table 4.9.6 (a) Mnemonic of DECB instruction (basic specification)

Step number	Instruction	Mnemonic format		Remarks	Memory status of control condition			
		Address No.	Bit No.		ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		25	DECB instruction				
3	(PRM)	O		Format specification				
4	(PRM)	0000		Code data address				
5	(PRM)	0000		Decode designation				
6	(PRM)	0000		Decode result output address				

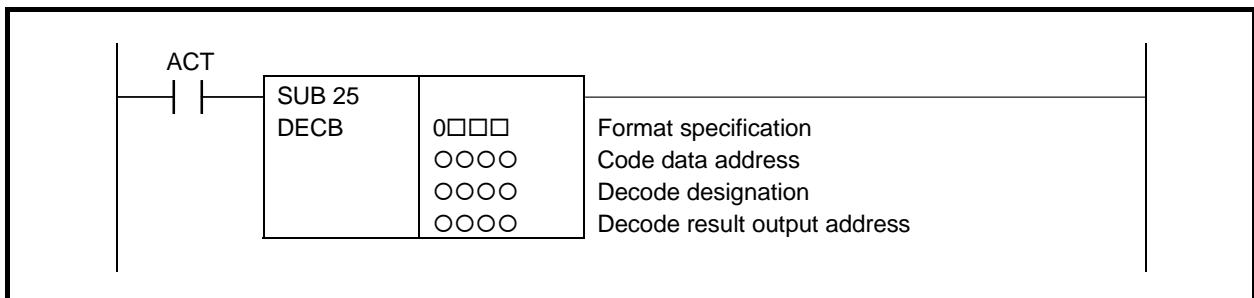


Fig. 4.9.6 (d) Format of DECB instruction (extended specification)

Table 4.9.6 (b) Mnemonic of DECB instruction (extended specification)

Step number	Instruction	Mnemonic format		Remarks	Memory status of control condition			
		Address No.	Bit No.		ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		25	DECB instruction				
3	(PRM)	0□□□		Format specification				
4	(PRM)	0000		Code data address				
5	(PRM)	0000		Decode designation				
6	(PRM)	0000		Decode result output address				

Control conditions

(a) Command (ACT)

ACT=0: Resets all the output data bits.

ACT=1: Decodes data. Results of processing is set in the output data address.

Parameters

(a) Format specification

Set the size of code data to the 1st digit of the parameter.

0001: Code data is in binary format of 1 byte length

0002: Code data is in binary format of 2 bytes length

0004: Code data is in binary format of 4 bytes length

When setting format specification in the following extended format, DECB can decode multiple ($8 \times n$) bytes by 1 instruction.

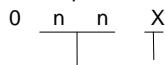
0nn1: In case of decoding multiple ($8 \times nn$) bytes and code data is binary format of 1 byte length

0nn2: In case of decoding multiple ($8 \times nn$) bytes and code data is binary format of 2 bytes length

0nn4: In case of decoding multiple ($8 \times nn$) bytes and code data is binary format of 4 bytes length

The nn is the numerical value from 02 to 99. When setting 00 or 01, it works for decoding 8 numbers.

Format specification (extended specification) :



The byte length setting of code data

1: 1 byte length

2: 2 byte length

4: 4 byte length

The multiple decoding number setting

00-01:

It decodes 8 continuous numbers.

The decode result output address needs a memory of 1 byte length.

02-99:

It decodes multiple ($8 \times nn$) continuous numbers.

The decode result output address needs a memory of nn bytes length.

(b) Code data address

Specifies the numbers to be decoded.

(c) Number specification decode designation

Specifies the numbers to be decoded.

(d) Decode result address

Specifies an address where the decoded result shall be output.

A one-byte area is necessary in the memory for the output.

When executing this instruction in extended specification, the area of setting by the format specification for the nn bytes is necessary.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

4.9.7 TBCDB (Binary to BCD Conversion (1 Byte Length) : SUB 313) TBCDW (Binary to BCD Conversion (2 Bytes Length) : SUB 314) TBCDD (Binary to BCD Conversion (4 Bytes Length) : SUB 315)

The Binary to BCD conversion instruction converts binary data to BCD format data.

As indicated below, three types of Binary to BCD conversion instructions are available according to the type of data to be converted.

Table 4.9.7 (a) Kinds of Binary to BCD conversion instruction

	Instruction name	SUB No.	Data type	
			Source	Destination
1	TBCDB	313	1 byte length signed binary	2-digit BCD
2	TBCDW	314	2 bytes length signed binary	4-digit BCD
3	TBCDD	315	4 bytes length signed binary	8-digit BCD

If conversion source binary data is not within the valid BCD format data range (if source binary data is a negative value or is greater than the maximum allowable value), the result of conversion is not output, and W1=0 is set.

Format

Fig. 4.9.7 shows the ladder format and Table 4.9.7(b) shows the mnemonic format.

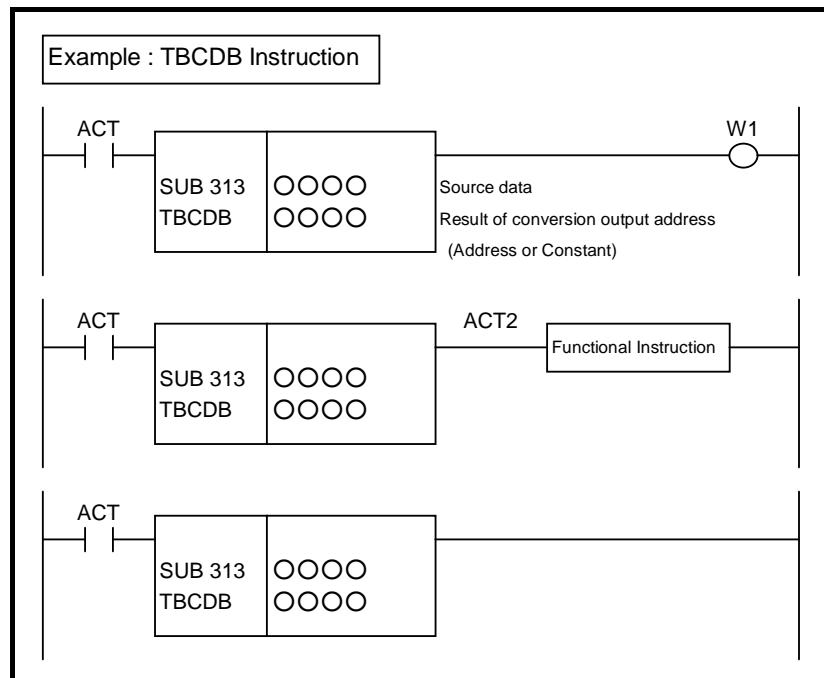


Fig. 4.9.7 Format of TBCDB, TBCDW, TBCDD instruction

Table 4.9.7(b) Mnemonic of TBCDB, TBCDW, TBCDD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		313	SUB No. (TBCDB instruction)				
3	(PRM)	0000		Source data				
4	(PRM)	0000		Result of conversion output address (Address or Constant)				
5	WRT	0000 .0		Normal end output				W1

Control conditions

- (a) Input signal (ACT)
 ACT = 0: Instruction not executed.
 ACT = 1: Executed.

Parameters

- (a) Source data
 Specify conversion source binary data. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Result of conversion output address
 Specify the address to which BCD format data produced as the result of conversion is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when conversion source data is not within the valid BCD format data range.

NOTE

- W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- With the similar functional instruction DCNVB, W1=1 is set when an operation is terminated abnormally. With TBCDB, TBCDW, and TBCDD, W1=1 is set when an operation is terminated normally.
- No data is output to the operation output registers (R9000, Z0).

4.9.8 FBCDB (BCD to Binary Conversion (1 Byte Length) : SUB 313) FBCDW (BCD to Binary Conversion (2 Bytes Length) : SUB 314) FBCDD (BCD to Binary Conversion (4 Bytes Length) : SUB 315)

The BCD to Binary conversion instruction converts BCD format data to binary data.

As indicated below, three types of BCD to Binary conversion instructions are available according to the type of data to be converted.

Table 4.9.8 (a) Kinds of BCD to Binary conversion instruction

	Instruction name	SUB No.	Data type	
			Source	Destination
1	FBCDB	316	2-digit BCD	1 byte length signed binary
2	FBCDW	317	4-digit BCD	2 bytes length signed binary
3	FBCDD	318	8-digit BCD	4 bytes length signed binary

If conversion source data is invalid as BCD format data, the result of conversion is not output, and W1=0 is set.

Format

Fig. 4.9.8 shows the ladder format and Table 4.9.8(b) shows the mnemonic format.

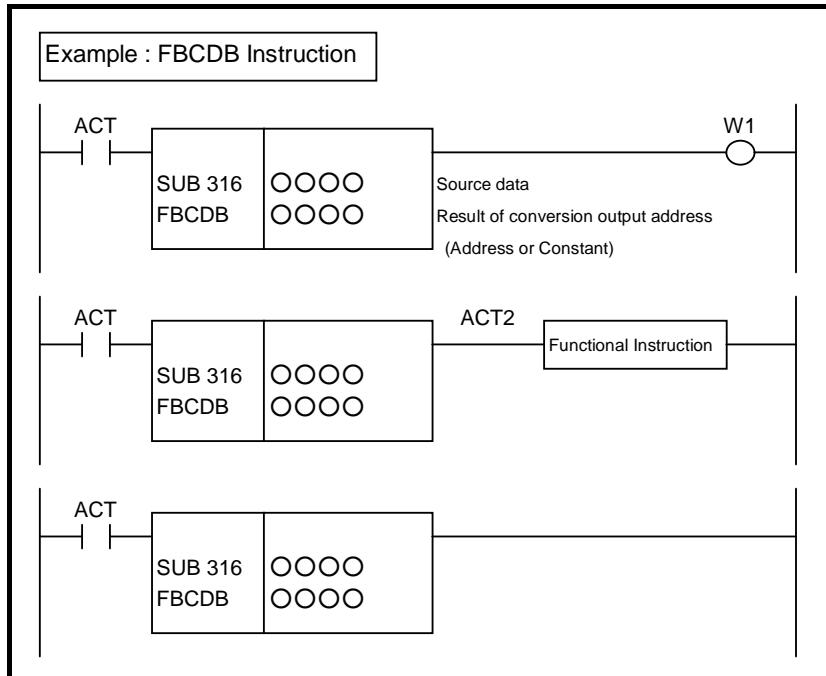


Fig. 4.9.8 Format of FBCDB, FBCDW, FBCDD instruction

Table 4.9.8(b) Mnemonic of FBCDB, FBCDW, FBCDD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB		316	SUB No. (FBCDB instruction)
3	(PRM)		0000	Source data
4	(PRM)		0000	Result of conversion output address (Address or Constant)
5	WRT		0000 .0	Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Source data

Specify conversion source BCD format data. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Result of conversion output address

Specify the address to which binary data produced as the result of conversion is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when conversion source data is invalid as BCD format data.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction DCNVB, W1=1 is set when an operation is terminated abnormally. With FBCDB, FBCDW, and FBCDD, W1=1 is set when an operation is terminated normally.
- 3 No sign is specified for binary data output as the result of conversion. Invert the sign by using NEGSx after conversion if necessary.
- 4 No data is output to the operation output registers (R9000, Z0).

4.10 OPERATION INSTRUCTION

The following types of operation instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	ADDB	36	Binary addition
2	SUBB	37	Binary subtraction
3	MULB	38	Binary multiplication
4	DIVB	39	Binary division
5	ADD	19	BCD addition
6	SUB	20	BCD subtraction
7	MUL	21	BCD multiplication
8	DIV	22	BCD division
9	NUMEB	40	Definition of binary constants
10	NUME	23	Definition of BCD constants
11	ADDSB	319	Addition (1 byte length)
12	ADDSW	320	Addition (2 bytes length)
13	ADDSD	321	Addition (4 bytes length)
14	SUBSB	322	Subtraction (1 byte length)
15	SUBSW	323	Subtraction (2 bytes length)
16	SUBSD	324	Subtraction (3 byte length)
17	MULSB	325	Multiplication (1 byte length)
18	MULSW	326	Multiplication (2 bytes length)
19	MULSD	327	Multiplication (4 bytes length)
20	DIVSB	328	Division (1 byte length)
21	DIVSW	329	Division (2 bytes length)
22	DIVSD	330	Division (4 bytes length)
23	MODSB	331	Remainder (1 byte length)
24	MODSW	332	Remainder (2 bytes length)
25	MODSD	333	Remainder (4 bytes length)
26	INCSB	334	Increment (1 byte length)
27	INCSW	335	Increment (2 bytes length)
28	INCSD	336	Increment (4 bytes length)
29	DECSB	337	Decrement (1 byte length)
30	DECSD	338	Decrement (2 bytes length)
31	DECSD	339	Decrement (4 bytes length)
32	ABSSB	340	Absolute value (1 byte length)
33	ABSSW	341	Absolute value (2 bytes length)
34	ABSSD	342	Absolute value (4 bytes length)
35	NEGSB	343	Sign inversion (1 byte length)
36	NEGSW	344	Sign inversion (2 bytes length)
37	NEGSD	345	Sign inversion (4 bytes length)

4.10.1 ADDB (Binary Addition: SUB 36)

This instruction performs binary addition between 1-, 2-, and 4-byte data. In the operation result register (R9000, Z0), operating data is set besides the numerical data representing the operation results. The required number of bytes is necessary to store each augend, the added, and the operation output data.

Format

Fig. 4.10.1 shows the ladder format and Table 4.10.1 shows the mnemonic format.

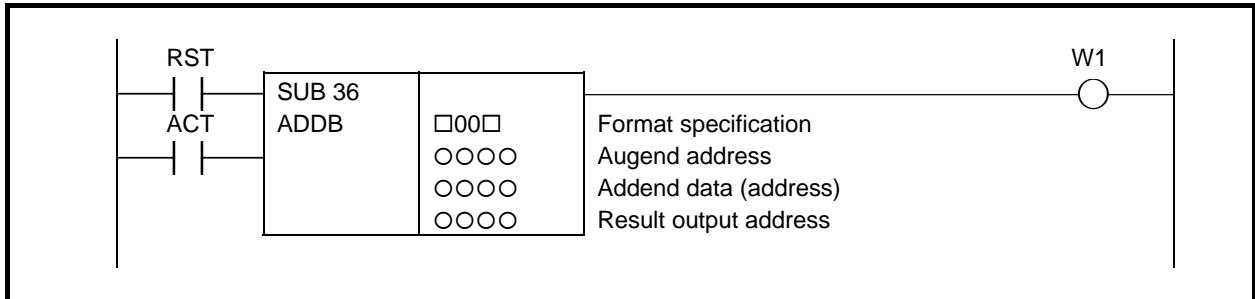


Fig. 4.10.1 Format of ADDB instruction

Table 4.10.1 Mnemonic of ADDB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	OOOO .O		RST				RST
2	RD.STK	OOOO .O		ACT			RST	ACT
3	SUB	36		ADDB instruction				
4	(PRM)	□00□		Format specification				
5	(PRM)	OOOO		Augend address				
6	(PRM)	OOOO		Addend data (address)				
7	(PRM)	OOOO		Result output address				
8	WRT	OOOO .O		Error output				W1

Control conditions

(a) Reset (RST)

RST=0: Release reset

RST=1: Resets error output W1. In other words, makes W1=0.

(b) Command (ACT)

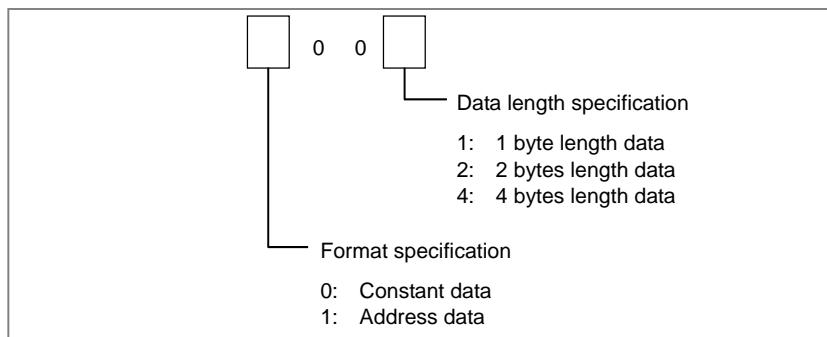
ACT=0: Do not execute ADDB. W1 does not change.

ACT=1: Execute ADDB.

Parameters

(a) Format specification

Specifies data length (1, 2, and 4 bytes) and the format for the addend (constant or address).



(b) Augend address

Address containing the augend.

(c) Addend data (address)

Specification in (a) determines the format of the addend.

(d) Result output address

Specifies the address to contain the result of operation.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

W1=0: Operation correct

W1=1: Operation incorrect

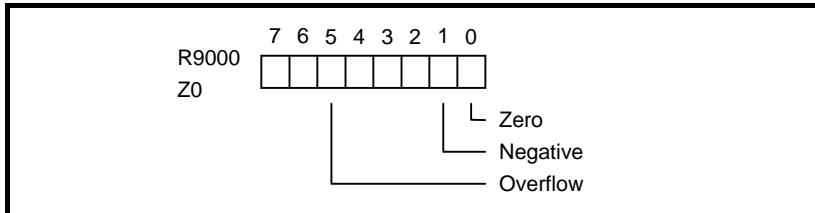
W1 goes on (W1=1) if the result of addition exceeds the specified data length. Then, the result will be output and the overflow flag and other flags will be output to the operation output register.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Operation output register (R9000, Z0)

This register is set with data on operation. If register bit is on, they signify the following operation data:



4.10.2 SUBB (Binary Subtraction: SUB 37)

This instruction subtracts one data from another, both data being in the binary format of 1, 2 or 4 bytes. In the operation result register (R9000, Z0), operation data is set besides the numerical data representing the operation. A required number of bytes is necessary to store the subtrahend, minuend, and the result (difference).

Format

Fig. 4.10.2 shows the ladder format and Table 4.10.2 shows the mnemonic format.

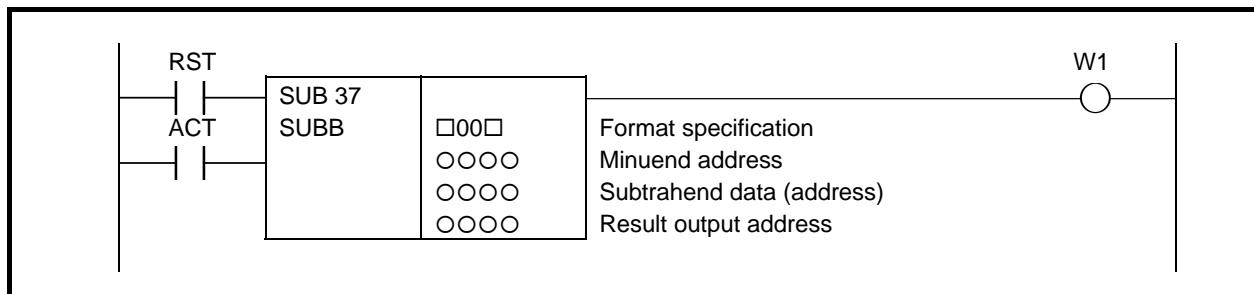


Fig. 4.10.2 Format of SUBB instruction

Table 4.10.2 Mnemonic of SUBB instruction

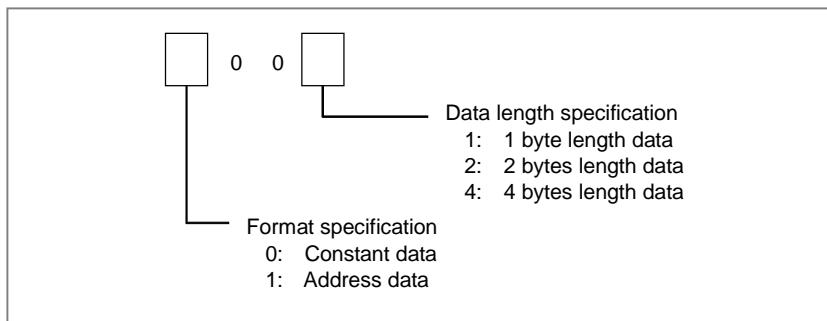
Mnemonic format				Memory status of control condition
Step number	Instruction	Address No.	Bit No.	
1	RD	0000 .O		RST
2	RD.STK	0000 .O		ACT
3	SUB	37		SUBB instruction
4	(PRM)	□00□		Format specification
5	(PRM)	0000		Minuend address
6	(PRM)	0000		Subtrahend data (address)
7	(PRM)	0000		Result output address
8	WRT	0000 .O		Error output

Control conditions

- (a) Reset (RST)
 - RST=0: Release reset
 - RST=1: Resets error output W1. (Set W1 to 0.)
- (b) Command (ACT)
 - ACT=0: Do not execute SUBB. W1 does not change.
 - ACT=1: Execute SUBB.

Parameters

- (a) Format specification
 - Specifies data length (1, 2, and 4 bytes) and the format for the subtrahend (constant or address).



- (b) Minuend address
 - Address containing the minuend.
- (c) Subtrahend data (address)
 - Specification in (a) determines the format of the Subtrahend.
- (d) Result output address
 - Specifies the address to contain the result of operation.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

- W1=0: Operation correct
- W1=1: Operation incorrect

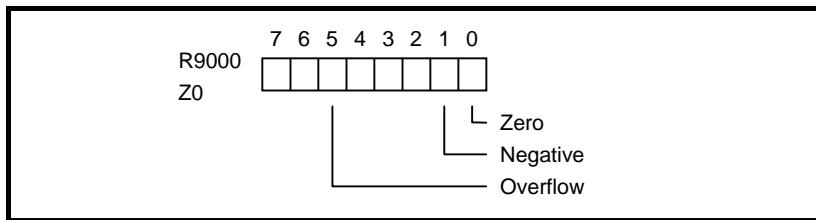
W1 goes on (W1=1) if the result of subtraction exceeds the specified data length. Then, the result will be output and the overflow flag and other flags will be output to the operation output register.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Operation output register (R9000, Z0)

This register is set with data on operation. If register bit is on, they signify the following operation data:



4.10.3 MULB (Binary Multiplication: SUB 38)

This instruction multiplies 1-, 2-, and 4-byte binary data items. In the operation result register (R9000, Z0), operation data is set besides the numerical data representing the operation.

A required number of bytes is necessary to store multiplicand, multiplier, and the result (product).

Format

Fig. 4.10.3 shows the ladder format and Table 4.10.3 shows the mnemonic format.

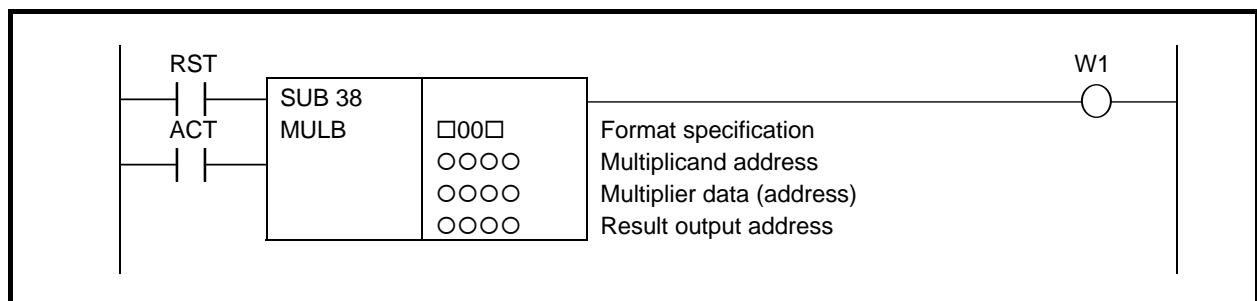


Fig. 4.10.3 Format of MULB instruction

Table 4.10.3 Mnemonic of MULB instruction

Mnemonic format					Memory status of control condition
Step number	Instruction	Address No.	Bit No.	Remarks	
1	RD	0000 .0		RST	
2	RD.STK	0000 .0		ACT	
3	SUB	38		MULB instruction	
4	(PRM)	□00□		Format specification	
5	(PRM)	0000		Multiplicand address	
6	(PRM)	0000		Multiplier data (address)	
7	(PRM)	0000		Result output address	
8	WRT	0000 .0		Error output	

Control conditions

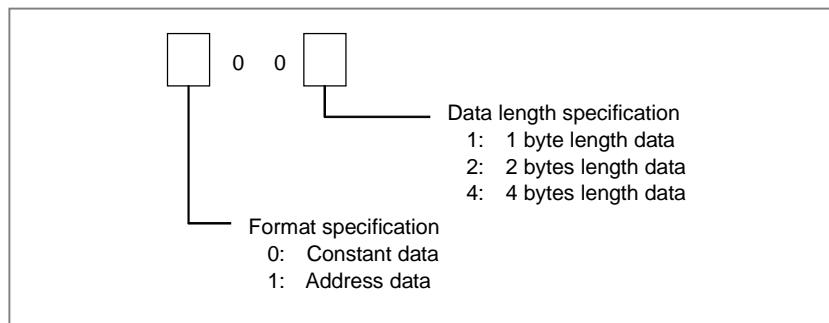
- (a) Reset (RST)
 - RST=0: Release reset
 - RST=1: Resets error output W1. In other words, makes W1=0.

- (b) Command (ACT)
 - ACT=0: Do not execute MULB. W1 does not change.
 - ACT=1: Execute MULB.

Parameters

- (a) Format specification

Specifies data length (1, 2, and 4 bytes) and the format for the multiplier (constant or address).



- (b) Multiplicand address

Address containing the multiplicand.

- (c) Multiplier data (address or constant)

Specification in (a) determines the format of the multiplier.

- (d) Result output address

Specifies the address to contain the result of operation.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

- W1=0: Operation correct
- W1=1: Operation incorrect

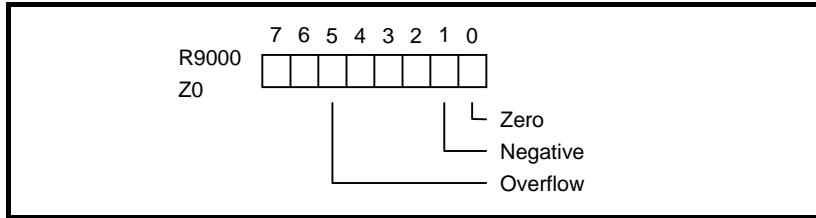
W1 goes on (W1=1) if the result of multiplication exceeds the specified data length. Then, the result will be output and the overflow flag and other flags will be output to the operation output register.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Operation output register (R9000, Z0)

This register is set with data on operation. If register bit is on, they signify the following operation data:



4.10.4 DIVB (Binary Division: SUB 39)

This instruction divides binary data items 1, 2, and 4 bytes in length. In the operation result register (R9000, Z0), operation data is set and remainder is set to R9002 and following addresses.

A required number of bytes is necessary to store the dividend, divisor, and the result (quotient).

Format

Fig. 4.10.4 shows the ladder format and Table 4.10.4 shows the mnemonic format.

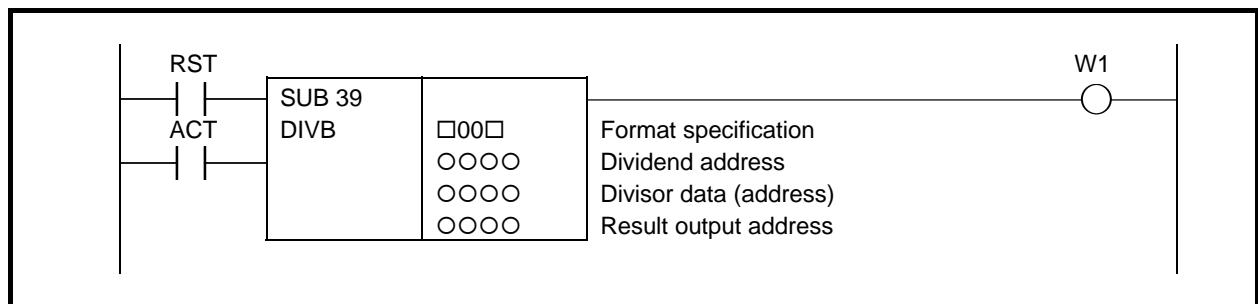


Fig. 4.10.4 Format of DIVB instruction

Table 4.10.4 Mnemonic of DIVB instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		RST				RST
2	RD.STK	0000 .0		ACT			RST	ACT
3	SUB	39		DIVB instruction				
4	(PRM)	□00□		Format specification				
5	(PRM)	○○○○		Dividend address				
6	(PRM)	○○○○		Divisor data (address)				
7	(PRM)	○○○○		Result output address				
8	WRT	0000 .0		Error output				W1

Control conditions

(a) Reset (RST)

RST=0: Release reset

RST=1: Resets error output W1. In other words, makes W1=0.

(b) Command (ACT)

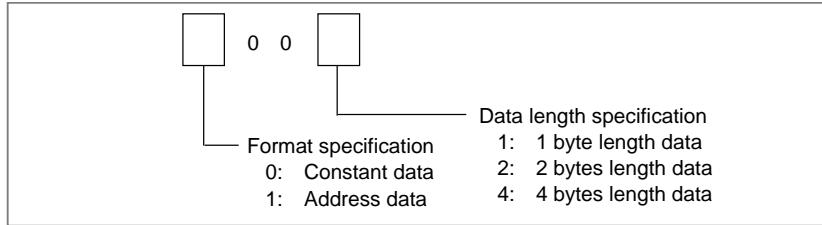
ACT=0: Do not execute DIVB. W1 does not change.

ACT=1: Execute DIVB.

Parameters

(a) Format specification

Specifies data length (1, 2, and 4 bytes) and the format for the divisor (constant or address).



- (b) Dividend address
Address containing the dividend
- (c) Divisor data (address)
Specification in (a) determines the format of the divisor.
- (d) Result output address
Specified the address to contain the result of operation.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Error output (W1)

- W1=0: Operation correct
W1=1: Operation incorrect

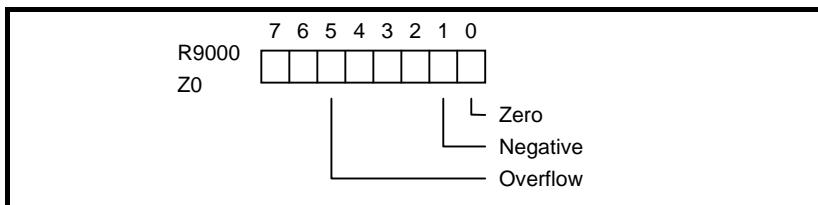
W1 goes on (W1=1) if the divisor is 0. Then, the result will be output and the overflow flag and other flags will be output to the operation output register.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

Operation output register (R9000, Z0)

This register is set with data on operation. If register bit is on, they signify the following operation data:



Remainder output address

Depending on its length, the remainder is stored in one or more of registers R9002 to R9005 or Z2 to Z5.

4.10.5 ADD (BCD Addition: SUB 19)

Adds BCD two- or four-digit data.

Format

Fig. 4.10.5 shows the ladder format and Table 4.10.5 shows the mnemonic format.

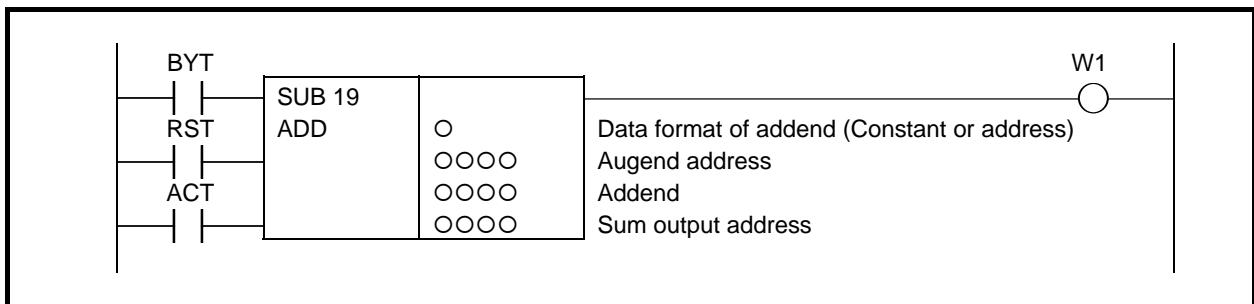


Fig. 4.10.5 Format of ADD instruction

Table 4.10.5 Mnemonic of ADD instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		RST				BYT RST
3	RD.STK	0000 .0		ACT				ACT
4	SUB	19		ADD instruction				
5	(PRM)	O		Addend format				
6	(PRM)	0000		Augend address				
7	(PRM)	0000		Addend (address)				
8	(PRM)	0000		Sum output address				
9	WRT	0000 .0		Error output				W1

Control conditions

- (a) Specify the number of digits of data. (BYT)
 - BYT=0: Data is BCD two digits long.
 - BYT=1: Data is BCD four digits long.
- (b) Reset (RST)
 - RST=0: Release reset.
 - RST=1: Resets error output W1, that is, sets W1 to 0.
- (c) Execution command (ACT)
 - ACT=0: The ADD instruction is not executed. W1 does not change.
 - ACT=1: The ADD instruction is executed.

Parameters

- (a) Data format of addend
 - 0: Specifies addend with a constant.
 - 1: Specifies addend with an address.
- (b) Augend address
 - Set the address storing the augend.
- (c) Addend (address)
 - Addressing of the addend depends on above (a).
- (d) Sum output address
 - Set the address to which the sum is to be output.



Please do not set an illegal value, that is not indicated above, into the "(a) Data format of addend".

Error output

W1=0: Normal operation

W1=1: Abnormal operation. W1 is set to 1 to indicate an error, e.g. if the result of the addition exceeds the data size specified for control condition (a) described above.



Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.10.6 SUB (BCD Subtraction: SUB 20)

Subtracts BCD two- or four-digit data.

Format

Fig. 4.10.6 shows the ladder format and Table 4.10.6 shows the mnemonic format.

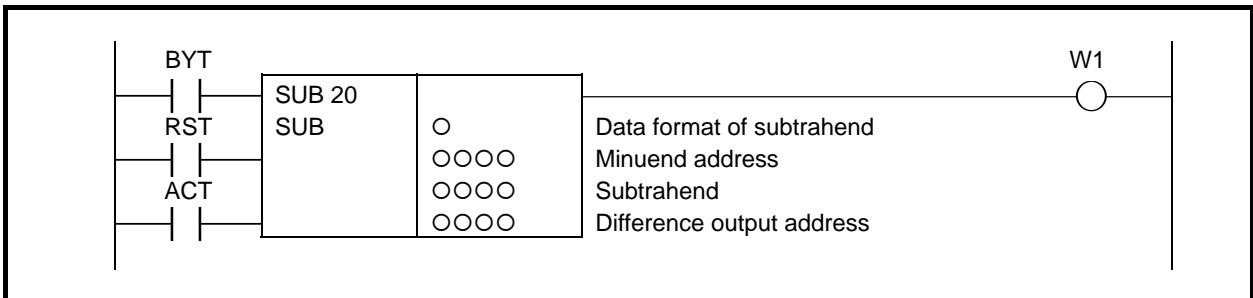


Fig. 4.10.6 Format of SUB instruction

Table 4.10.6 Mnemonic of SUB instruction

Mnemonic format

Memory status of control condition

Condition				
Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		BYT
2	RD.STK	0000 .0		RST
3	RD.STK	0000 .0		ACT
4	SUB	20		SUB instruction
5	(PRM)	0		Data format of subtrahend
6	(PRM)	0000		Minuend address
7	(PRM)	0000		Subtrahend (address)
8	(PRM)	0000		Difference output address
9	WRT	0000 .0		Error output

Control conditions

- (a) Specification of the number of digits of data. (BYT)

BYT=0: Data BCD two digits long

BYT=1: Data BCD four digits long

- (b) Reset (RST)

RST=0: Release reset.
 RST=1: Resets error output W1, that is, sets W1 to 0.

- (c) Execution command (ACT)
 ACT=0: The SUB instruction is not executed. W1 does not change.
 ACT=1: The SUB instruction is executed.

Parameters

- (a) Data format of subtrahend
 0: Specifies subtrahend with a constant.
 1: Specifies subtrahend with an address.
- (b) Minuend address
 Set the address storing the minuend.
- (c) Subtrahend (address)
 Addressing of the subtrahend depends on above (a).
- (d) Difference output address
 Sets the address to which the difference is output.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Data format of subtrahend".

Error output

- W1=0: Normal operation
 W1=1: Abnormal operation. W1 is set 1 to indicate an error if the difference is negative.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.10.7 MUL (BCD Multiplication: SUB 21)

Multiplies BCD two- or four-digit data. The product must also be BCD two- or four-digit data.

Format

Fig. 4.10.7 shows the ladder format and Table 4.10.7 shows the mnemonic format.

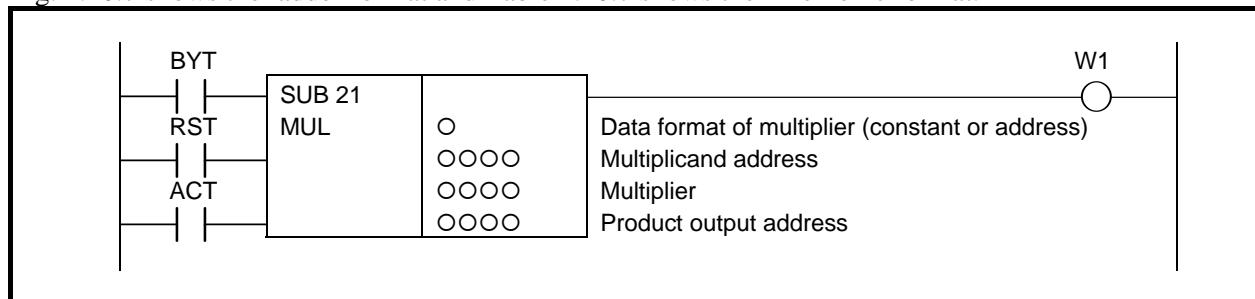


Fig. 4.10.7 Format of MUL instruction

Table 4.10.7 Mnemonic of MUL instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		BYT				BYT
2	RD.STK	0000 .0		RST			BYT	RST
3	RD.STK	0000 .0		ACT	BYT	RST	ACT	
4	SUB		21	MUL instruction				
5	(PRM)	0		Data format of multiplier				
6	(PRM)	0000		Multiplicand address				
7	(PRM)	0000		Multiplier (address)				
8	(PRM)	0000		Product output address				
9	WRT	0000 .0		Error output				W1

Control conditions

- (a) Specify the number of digits of data. (BYT)
 - BYT=0: Data is BCD two digits long.
 - BYT=1: Data is BCD four digits long.
- (b) Reset (RST)
 - RST=0: Releases reset.
 - RST=1: Resets error output W1, that is, sets W1 to 0.
- (c) Execution command (ACT)
 - ACT=0: The MUL instruction is not executed. W1 does not change.
 - ACT=1: The MUL instruction is executed.

Parameters

- (a) Data format of multiplier
 - 0: Specifies multiplier with a constant.
 - 1: Specifies multiplier with an address.
- (b) Multiplicand address
 - Sets the address storing the multiplicand.
- (c) Multiplier (address)
 - Addressing of the multiplier depends on above (a).
- (d) Product output address
 - Set the address to which the product is output.

CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Data format of multiplier".

Error output

- W1=0: Normal operation
- W1=1: Abnormal operation. W1=1 is set to indicate an error if the product exceeds the specified size.

CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.10.8 DIV (BCD Division: SUB 22)

Divides BCD two- or four-digit data. Remainders are discarded.

Format

Fig. 4.10.8 shows the ladder format and Table 4.10.8 shows the mnemonic format.

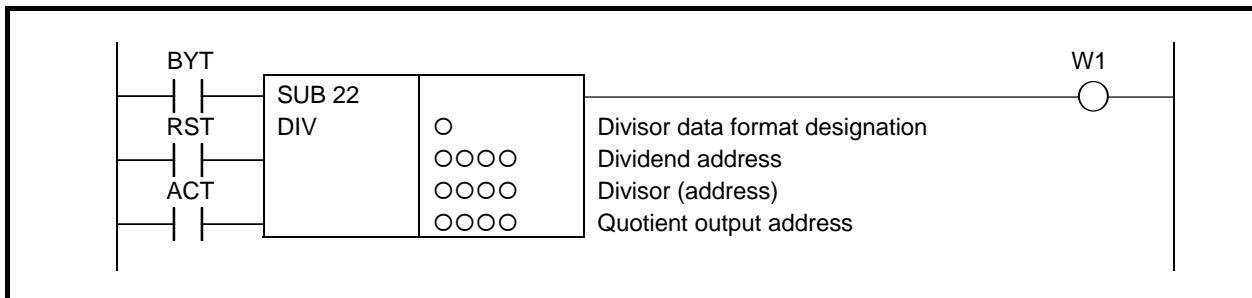


Fig. 4.10.8 Format of DIV instruction

Table 4.10.8 Mnemonic of DIV instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		BYT				BYT
2	RD.STK	0000 .O		RST				BYT RST
3	RD.STK	0000 .O		ACT				BYT RST ACT
4	SUB	22		DIV instruction				
5	(PRM)	O		Divisor data format designation				
6	(PRM)	0000		Dividend address				
7	(PRM)	0000		Divider (address)				
8	(PRM)	0000		Quotient output address				
9	WRT	0000 .O		Error output				

Control conditions

- (a) Specify the number of digits of data. (BYT)
 - BYT=0: Data is BCD two digits long.
 - BYT=1: Data is BCD four digits long.
- (b) Reset (RST)
 - RST=0: Releases reset.
 - RST=1: Resets error output W1, that is, sets W1 to 0.
- (c) Execution command (ACT)
 - ACT=0: The DIV instruction is not executed. W1 does not change.
 - ACT=1: The DIV instruction is executed.

Parameters

- (a) Divisor data format designation
 - 0: Specifies divisor data by constant.

- 1: Specifies divisor data by address.
- (b) Dividend address
Sets the address storing the dividend.
- (c) Divisor (address)
Addressing of the divisor depends on above (a).
- (d) Quotient output address
Sets the address to which the quotient is output.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Divisor data format designation".

Error output

W1=0: Normal operation

W1=1: Abnormal operation. W1=1 is set to indicate an error if the divider is 0.

⚠ CAUTION

Two or more coils, WRT, WRT.NOT, SET or RST, that follow this instruction are prohibited. You have to place a single coil instruction as the output of this instruction.

4.10.9 NUMEB (Definition of Binary Constants: SUB 40)

This instruction defines 1, 2, or 4-bytes long binary constant. Data entered in decimal during programming is converted into binary data during program execution. The binary data is stored in the specified memory address(es).

There are two specifications - basic specification and extended specification - for setting the format specification parameter in the NUMEB instruction. The extended specification allows all the set constants to be defined simultaneously in an array having n elements. This extended specification is effective when initializing a large memory area with value. For the details of the setting of a format specification parameter, see the description of parameters.

Format

Figs. 4.10.9 (a) and (b) show the ladder formats and Tables 4.10.9 (a) and (b) show the mnemonic formats.

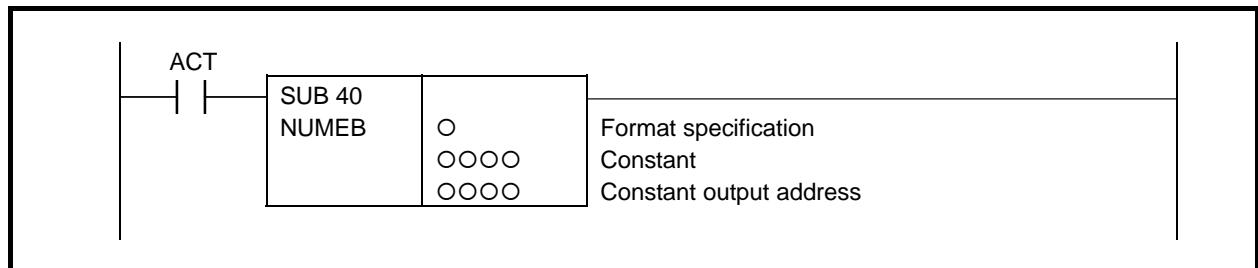
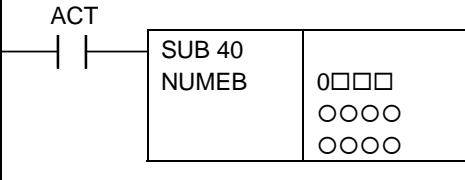


Fig. 4.10.9 (a) Format of NUMEB instruction (basic specification)

Table 4.10.9 (a) Mnemonic of NUMEB instruction (basic specification)

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		40	NUMEB instruction				
3	(PRM)		O	Format specification				
4	(PRM)		0000	Constant				
5	(PRM)		0000	Constant output address				▼



ACT

SUB 40
NUMEB

0□□□
0000
0000

Format specification
Constant
Constant output address

Fig. 4.10.9 (b) Format of NUMEB instruction (extended specification)

Table 4.10.9 (b) Mnemonic of NUMEB instruction (extended specification)

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		ACT				ACT
2	SUB		40	NUMEB instruction				
3	(PRM)		0□□□	Format specification				
4	(PRM)		0000	Constant				
5	(PRM)		0000	Constant output address				▼

Control conditions

(a) Command (ACT)

ACT= 0: Do not execute NUMEB.

ACT=1 : Execute NUMEB.

Parameters

(a) Format specification

Specifies data length (1, 2, or 4 bytes).

Use the first parameter digit to specify byte length:

0001:Binary data of 1 byte length

0002:Binary data of 2 bytes length

0004:Binary data of 4 bytes length

When setting format specification in the following extended format, NUMEB can define all the set constants simultaneously in an array having nn elements.

Specify data length (1, 2, or 4) to the 1st digit as above-mentioned.

Specify the number of the array in which is a constant to the 2nd and 3rd digit is defines.

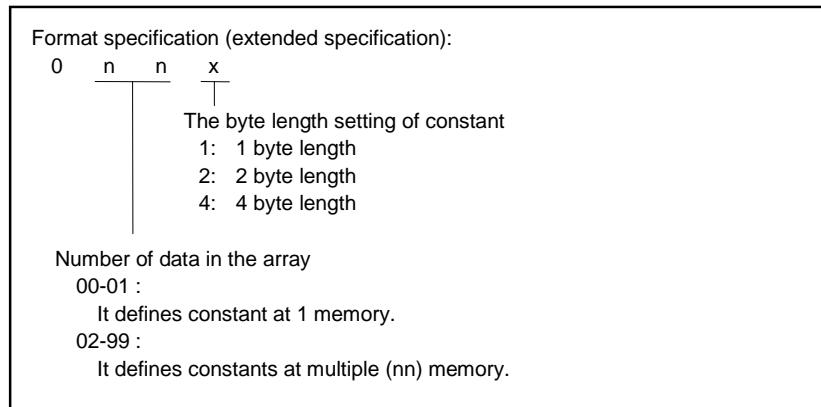
Specify 0 to the 4th digit.

Onn1:In case of defining multiple (nn) data by 1 byte length

Onn2:In case of defining multiple (nn) data by 2 bytes length

Onn4:In case of defining multiple (nn) data by 4 bytes length

The n is the numerical value from 02 to 99. When setting 00 or 01, it works as the basic specification that works for one data.



(b) Constant

Defined constants in decimal format. Set a constant data within the effective range for the byte length which is set in above (a).

(c) Constant output address

Specifies the address of the area for output of the binary data. The memory of the number of bytes which is set in above (a) is necessary.

When setting format specification in the extended format, it is necessary to reserve memory of (byte length) × (number of array elements which define constant) which was set in above (a).

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

4.10.10 NUME (BCD Definition of Constant: SUB 23)

Defines constants, when required. In this case, constants are defined with this instructions. The value type in this instruction is BCD.

Format

Fig. 4.10.10 shows the ladder format and Table 4.10.10 shows the mnemonic format.

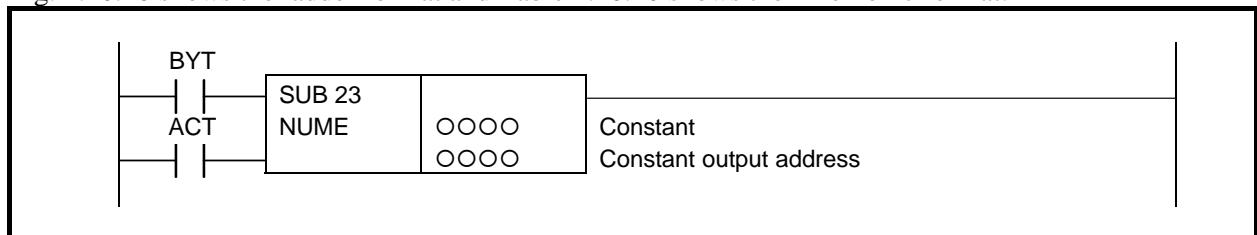


Fig. 4.10.10 Format of NUME instruction

Table 4.10.10 Mnemonic of NUME instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .O		BYT				BYT
2	RD.STK	0000 .O		ACT			BYT	ACT
3	SUB		23	NUME instruction				
4	(PRM)	0000		Constant				
5	(PRM)	0000		Constant output address				

Control conditions

- (a) Specify the number of digits of a constant. (BYT)

BYT=0: Constant is BCD two digits long.

BYT=1: Constant is BCD four digits long.

- (b) Execution command (ACT)

ACT=0: The NUME instruction is not executed.

ACT=1: The NUME instruction is executed.

Parameters

- (a) Constant

Sets the constant as the number of digits specified for control condition (a).

- (b) Constant output address

Sets the address to which the constant defined in parameter (a) is output.

4.10.11 ADDSB (Addition (1 Byte Length) : SUB 319)

ADDSW (Addition (2 Bytes Length) : SUB 320)

ADDSD (Addition (4 Bytes Length) : SUB 321)

The Addition instruction adds signed binary data.

In "Augend data" and "Addend data", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Addition instructions are available according to the type of data to be operated. In each instruction, "Augend data", "Addend data", and the data at "Result output address" are of the same data type.

Table4.10.11 (a) Kinds of Addition instruction

	Instruction name	SUB No.	Data type
1	ADDSB	319	1 byte length signed binary data
2	ADDSW	320	2 bytes length signed binary data
3	ADDSD	321	4 bytes length signed binary data

If an operation results in a positive overflow, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If an operation results in a negative overflow, the minimum value of each data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.11 shows the ladder format and Table 4.10.11(b) shows the mnemonic format.

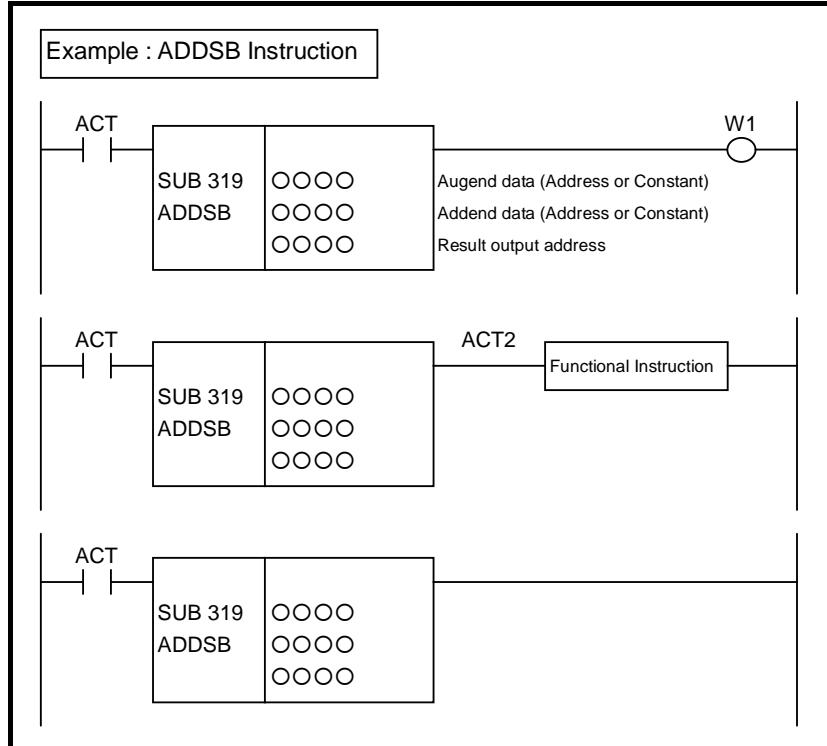


Fig. 4.10.11 Format of ADDSB, ADDSW, ADDSD instruction

Table 4.10.11(b) Mnemonic of ADDSB, ADDSW, ADDSD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	319		SUB No. (ADDSB instruction)
3	(PRM)	0000		Augend data (Address or Constant)
4	(PRM)	0000		Addend data (Address or Constant)
5	(PRM)	0000		Result output address
6	WRT	0000 .0		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Augend data
Specify an augend for addition operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Addend data
Specify an addend for addition operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (c) Result output address
Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction ADDB, W1=1 is set when an operation is terminated abnormally. With ADDSB, ADDSW, and ADDSD, W1=1 is set when an operation is terminated normally.
- 3 No data is output to the operation output registers (R9000, Z0).

4.10.12 SUBSB (Subtraction (1 Byte Length) : SUB 322)**SUBSW (Subtraction (2 Bytes Length) : SUB 323)****SUBSD (Subtraction (4 Bytes Length) : SUB 324)**

The Subtraction instruction subtracts signed binary data.

In "Minuend" and "Subtrahend", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Subtraction instructions are available according to the type of data to be operated. In each instruction, "Minuend", "Subtrahend", and the data at "Result output address" are of the same data type.

Table4.10.12 (a) Kinds of Subtraction instruction

	Instruction name	SUB No.	Data type
1	SUBSB	322	1 byte length signed binary data
2	SUBSW	323	2 bytes length signed binary data
3	SUBSD	324	4 bytes length signed binary data

If an operation results in a positive overflow, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If an operation results in a negative overflow, the minimum value of each data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.12 shows the ladder format and Table 4.10.12(b) shows the mnemonic format.

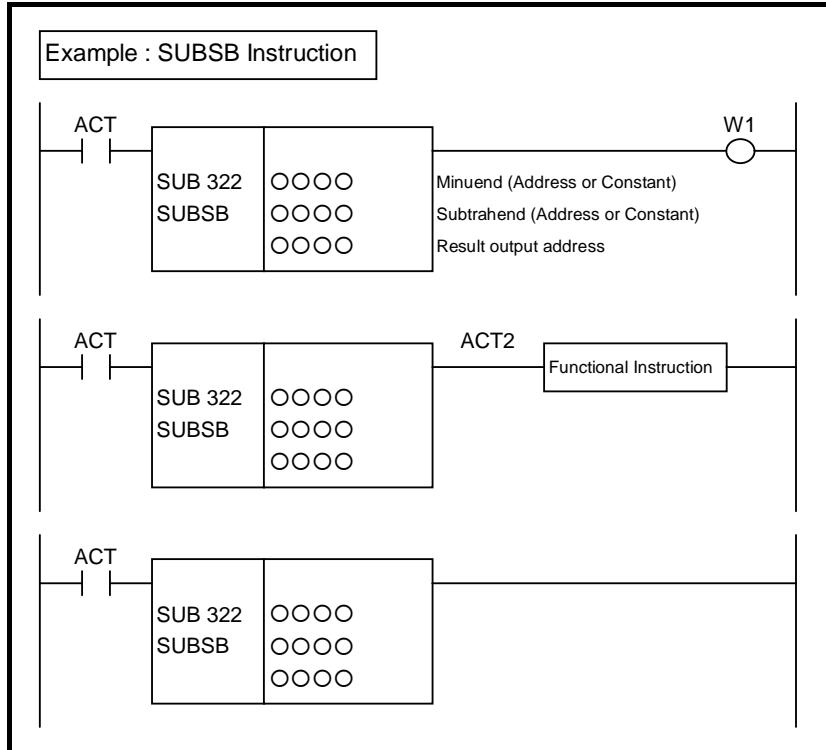


Fig. 4.10.12 Format of SUBSB, SUBSW, SUBSD instruction

Table 4.10.12(b) Mnemonic of SUBSB, SUBSW, SUBSD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	322		SUB No. (SUBSB instruction)
3	(PRM)	0000		Minuend (Address or Constant)
4	(PRM)	0000		Subtrahend (Address or Constant)
5	(PRM)	0000		Result output address
6	WRT	0000 .0		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

(a) Input signal (ACT)

ACT = 0: Instruction not executed.

ACT = 1: Executed.

Parameters

(a) Minuend

Specify a minuend for subtraction operation. In this parameter, a constant or a PMC memory address for storing data can be specified.

(b) Subtrahend

Specify a subtrahend for subtraction operation. In this parameter, a constant or a PMC memory address for storing data can be specified.

(c) Result output address

Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction SUBB, W1=1 is set when an operation is terminated abnormally. With SUBSB, SUBSW, and SUBSD, W1=1 is set when an operation is terminated normally.
- 3 No data is output to the operation output registers (R9000, Z0).

4.10.13 MULSB (Multiplication (1 Byte Length) : SUB 325) MULSW (Multiplication (2 Bytes Length) : SUB 326) MULSD (Multiplication (4 Bytes Length) : SUB 327)

The Multiplication instruction multiplies signed binary data.

In "Multiplicand" and "Multiplier", a constant or a PMC memory address for storing data can be specified. As indicated below, three types of Multiplication instructions are available according to the type of data to be operated. In each instruction, "Multiplicand", "Multiplier", and the data at "Result output address" are of the same data type.

Table 4.10.13 (a) Kinds of Multiplication instruction

	Instruction name	SUB No.	Data type
1	MULSB	325	1 byte length signed binary data
2	MULSW	326	2 bytes length signed binary data
3	MULSD	327	4 bytes length signed binary data

If an operation results in a positive overflow, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If an operation results in a negative overflow, the minimum value of each data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.13 shows the ladder format and Table 4.10.13(b) shows the mnemonic format.

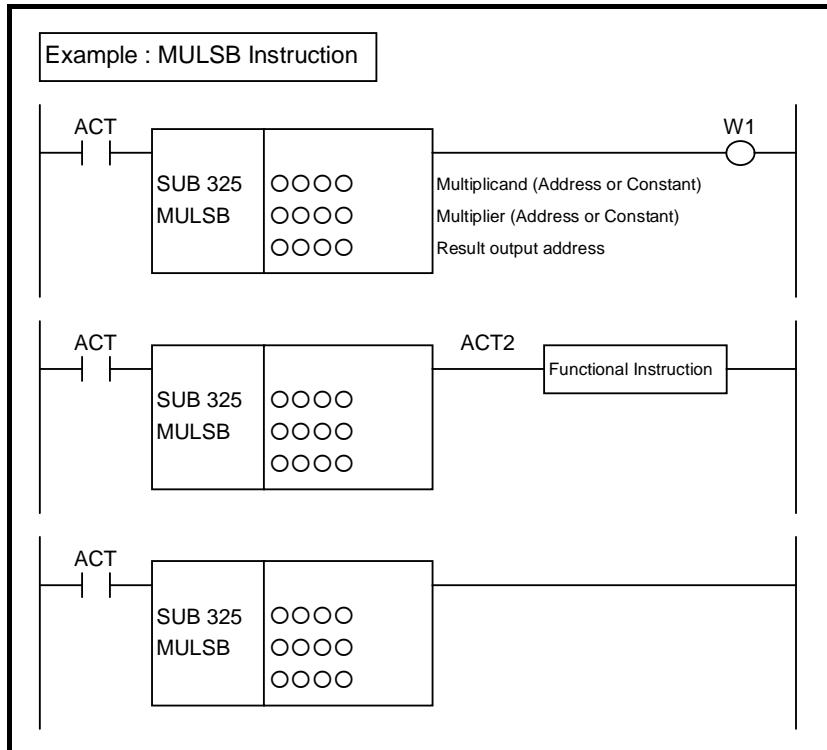


Fig. 4.10.13 Format of MULSB, MULSW, MULSD instruction

Table 4.10.13(b) Mnemonic of MULSB, MULSW, MULSD instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB		325	SUB No. (MULSB instruction)
3	(PRM)		0000	Multiplicand (Address or Constant)
4	(PRM)		0000	Multiplier (Address or Constant)
5	(PRM)		0000	Result output address
6	WRT		0000 .0	Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Multiplicand
Specify a multiplicand for multiplication operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Multiplier
Specify a multiplier for multiplication operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (c) Result output address
Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction MULB, W1=1 is set when an operation is terminated abnormally. With MULSB, MULSW, and MULSD, W1=1 is set when an operation is terminated normally.
- 3 No data is output to the operation output registers (R9000, Z0).

4.10.14 DIVSB (Division (1 Byte Length) : SUB 328) DIVSW (Division (2 Bytes Length) : SUB 329) DIVSD (Division (4 Bytes Length) : SUB 330)

The Division instruction divides signed binary data.

In "Dividend" and "Divisor", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Division instructions are available according to the type of data to be operated. In each instruction, "Dividend", "Divisor", and the data at "Result output address" are of the same data type.

Table 4.10.14 (a) Kinds of Division instruction

	Instruction name	SUB No.	Data type
1	DIVSB	328	1 byte length signed binary data
2	DIVSW	329	2 bytes length signed binary data
3	DIVSD	330	4 bytes length signed binary data

If an operation results in an overflow, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If the divisor is 0, and the dividend is 0 or a positive value, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If the divisor is 0, and the dividend is a negative value, the minimum value of each data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.14 shows the ladder format and Table 4.10.14(b) shows the mnemonic format.

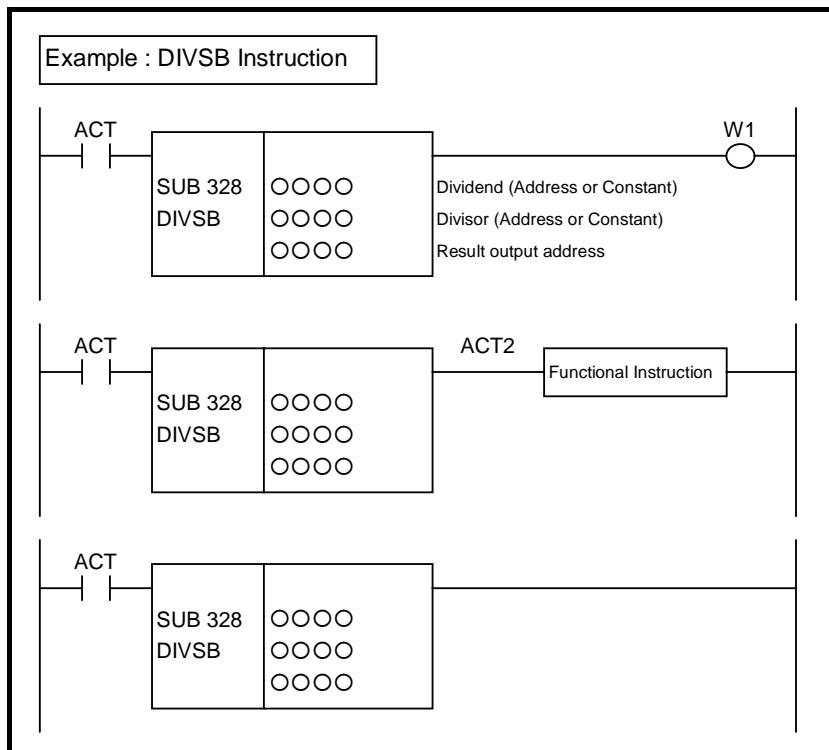


Fig. 4.10.14 Format of DIVSB, DIVSW, DIVSD instruction

Table 4.10.14(b) Mnemonic of DIVSB, DIVSW, DIVSD instruction

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	328		SUB No. (DIVSB instruction)
3	(PRM)	0000		Dividend (Address or Constant)
4	(PRM)	0000		Divisor (Address or Constant)
5	(PRM)	0000		Result output address
6	WRT	0000 .0		Normal end output

Memory status of control condition

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Dividend
Specify a dividend for division operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
 - (b) Divisor
Specify a divisor for division operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
 - (c) Result output address
Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0), when the divisor is 0, or when an operation results in an overflow.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction DIVB, W1=1 is set when an operation is terminated abnormally. With DIVSB, DIVSW, and DIVSD, W1=1 is set when an operation is terminated normally.
- 3 No data is output to the operation output registers (R9000, Z0).
- 4 No data is output to the remainder output addresses (R9002-R9005, Z2-Z5). To calculate remainder data, use the MODSB, MODSW, or MODSD instruction.

Operation

The result of each operation depends on the signs of the dividend and divisor as indicated below.

Table 4.10.14 (c) State of sign in division operation (example)

Dividend	Divisor	Result of DIVSx Instruction	Result of MODSx Instruction
20	3	6	2
20	-3	-6	2
-20	3	-6	-2
-20	-3	6	-2

4.10.15 MODSB (Remainder (1 Byte Length) : SUB 331) MODSW (Remainder (2 Bytes Length) : SUB 332) MODSD (Remainder (4 Bytes Length) : SUB 333)

The Remainder instruction divides signed binary data and calculates remainder data.

In "Dividend" and "Divisor", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Remainder instructions are available according to the type of data to be operated. In each instruction, "Dividend", "Divisor", and the data at "Result output address" are of the same data type.

Table 4.10.15 (a) Kinds of Remainder instruction

	Instruction name	SUB No.	Data type
1	MODSB	331	1 byte length signed binary data
2	MODSW	332	2 bytes length signed binary data
3	MODSD	333	4 bytes length signed binary data

If the quotient of a division operation results in an overflow (if "Dividend" is the minimum value of each data type or the divisor is -1), 0 is output to "Result output address", and W1=1 is set.

If "Divisor" is 0, and "Dividend" is 0 or a positive value, the maximum value of each data type is output to "Result output address", and W1=0 is set.

If "Divisor" is 0, and "Dividend" is a negative value, the minimum value of each data type is output to "Result output address", and W1=0.

Format

Fig. 4.10.15 shows the ladder format and Table 4.10.15(b) shows the mnemonic format.

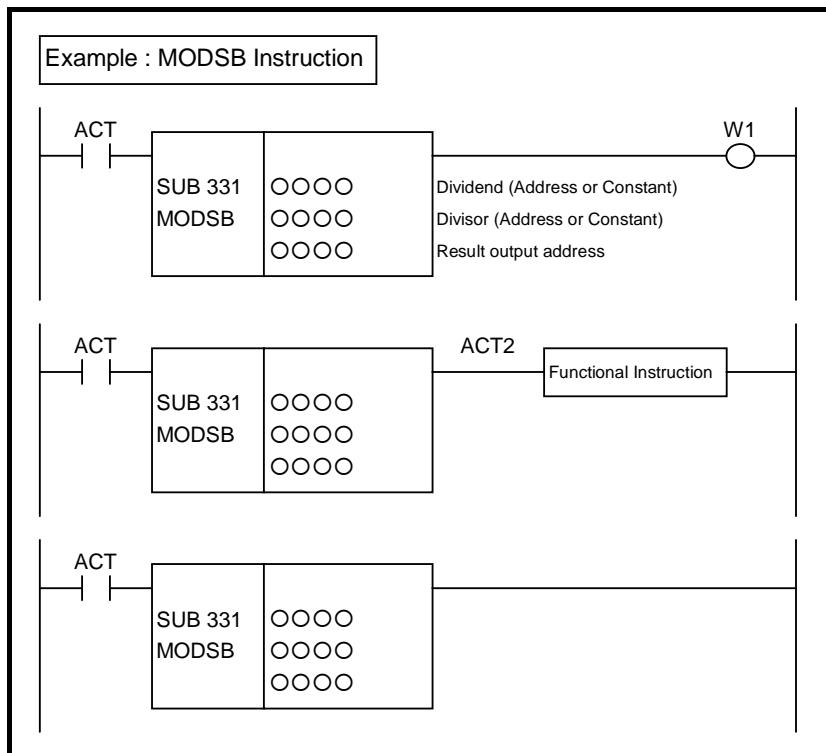


Fig. 4.10.15 Format of MODSB, MODSW, MODSD instruction

Table 4.10.15(b) Mnemonic of MODSB, MODSW, MODSD instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	331		SUB No. (MODSB instruction)
3	(PRM)	0000		Dividend (Address or Constant)
4	(PRM)	0000		Divisor (Address or Constant)
5	(PRM)	0000		Result output address
6	WRT	0000 .0		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
ACT = 0: Instruction not executed.
ACT = 1: Executed.

Parameters

- (a) Dividend
Specify a dividend for remainder operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Divisor
Specify a divisor for remainder operation. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (c) Result output address
Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when the divisor is 0.

NOTE

- 1 W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.
- 2 With the similar functional instruction DIVB, W1=1 is set when an operation is terminated abnormally. With MODSB, MODSW, and MODSD, W1=1 is set when an operation is terminated normally.
- 3 No data is output to the operation output registers (R9000, Z0).
- 4 No data is output to the remainder output addresses (R9002-R9005, Z2-Z5).

Operation

The result of each operation depends on the signs of the dividend and divisor as indicated below.

Table 4.10.15 (c) State of sign in division operation (example)

Dividend	Divisor	Result of DIVSx Instruction	Result of MODSx Instruction
20	3	6	2
20	-3	-6	2
-20	3	-6	-2
-20	-3	6	-2

4.10.16 INCSB (Increment (1 Byte Length) : SUB 334) INCSW (Increment (2 Bytes Length) : SUB 335) INCSD (Increment (4 Bytes Length) : SUB 336)

The Increment instruction increments signed binary data by 1.

As indicated below, three types of Increment instructions are available according to the type of data to be operated.

Table 4.10.16 (a) Kinds of Increment instruction

	Instruction name	SUB No.	Data type
1	INCSB	334	1 byte length signed binary data
2	INCSW	335	2 bytes length signed binary data
3	INCSD	336	4 bytes length signed binary data

If the Increment instruction is executed when data to be operated is the maximum value of a data type, the data to be operated remains unchanged from the maximum value, and W1=0 is set.

For example, if data to be operated by the INCSW instruction is 32767, the data remains unchanged from 32767 as the result of operation, and W1=0 is set.

Format

Fig. 4.10.16 shows the ladder format and Table 4.10.16(b) shows the mnemonic format.

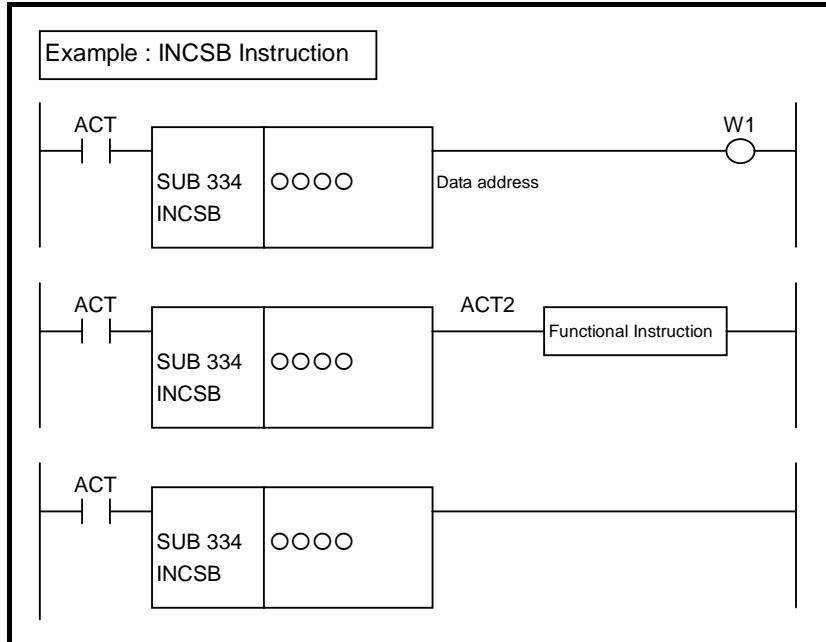


Fig. 4.10.16 Format of INCSB, INCSW, INCSD instruction

Table 4.10.16(b) Mnemonic of INCSB, INCSW, INCSD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	OOOO .O		ACT
2	SUB		334	SUB No. (INCSB instruction)
3	(PRM)		OOOO	Data address
4	WRT		OOOO .O	Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data address
 - Specify the PMC memory address the value at which is to be incremented.

Output (W1)

- W1=1 is set when an operation is terminated normally.
- W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.10.17 DECSB (Decrement (1 Byte Length) : SUB 337) DECSW (Decrement (2 Bytes Length) : SUB 338) DECSD (Decrement (4 Bytes Length) : SUB 339)

The Decrement instruction decrements signed binary data by 1.

As indicated below, three types of Decrement instructions are available according to the type of data to be operated.

Table 4.10.17 (a) Kinds of Decrement instruction

	Instruction name	SUB No.	Data type
1	DECSB	337	1 byte length signed binary data
2	DECSW	338	2 bytes length signed binary data
3	DECSD	339	4 bytes length signed binary data

If the Decrement instruction is executed when data to be operated is the minimum value of a data type, the data to be operated remains unchanged from the minimum value, and W1=0 is set.

For example, if data to be operated by the DECSW instruction is -32768, the data remains unchanged from -32768 as the result of operation, and W1=0 is set.

Format

Fig. 4.10.17 shows the ladder format and Table 4.10.17(b) shows the mnemonic format.

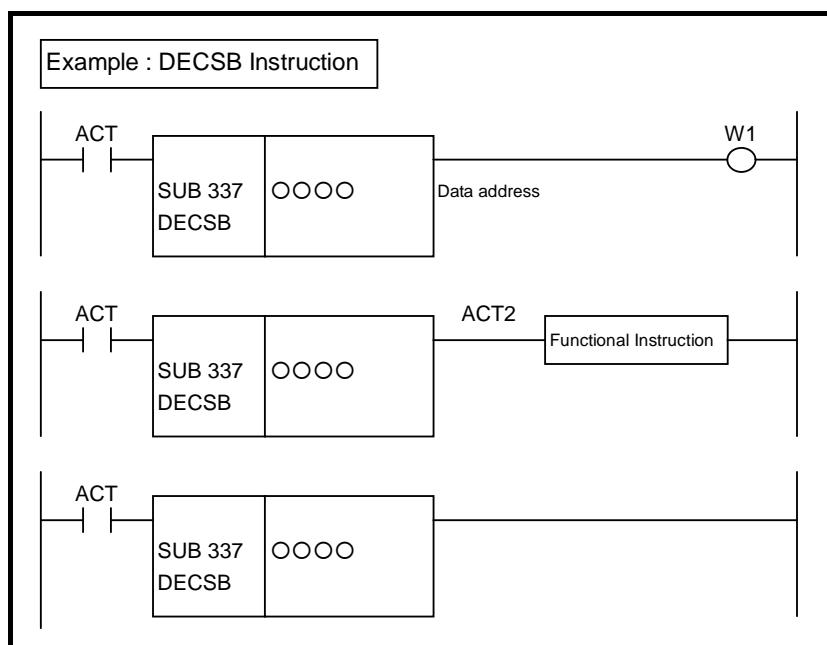


Fig. 4.10.17 Format of DECSB, DECSW, DECSD instruction

Table 4.10.17(b) Mnemonic of DECSB, DECSW, DECSD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	OOOO .O		ACT				ACT
2	SUB	337		SUB No. (DECSB instruction)				
3	(PRM)	OOOO		Data address				
4	WRT	OOOO .O		Normal end output				W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Data address
 - Specify the PMC memory address the value at which is to be decremented.

Output (W1)

W1=1 is set when an operation is terminated normally.
W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.10.18 ABSSB (Absolute Value (1 Byte Length) : SUB 340) ABSSW (Absolute Value (2 Bytes Length) : SUB 341) ABSSD (Absolute Value (4 Bytes Length) : SUB 342)

The Absolute value instruction calculates the absolute value of signed binary data.

In "Source data", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Absolute value instructions are available according to the type of data to be operated. In each instruction, "Source data" and the data at "Result output address" are of the same data type.

Table 4.10.18 (a) Kinds of Absolute value instruction

	Instruction name	SUB No.	Data type
1	ABSSB	340	1 byte length signed binary data
2	ABSSW	341	2 bytes length signed binary data
3	ABSSD	342	4 bytes length signed binary data

If an operation results in an overflow (if the minimum value of a data type is converted), the maximum value of the data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.18 shows the ladder format and Table 4.10.18(b) shows the mnemonic format.

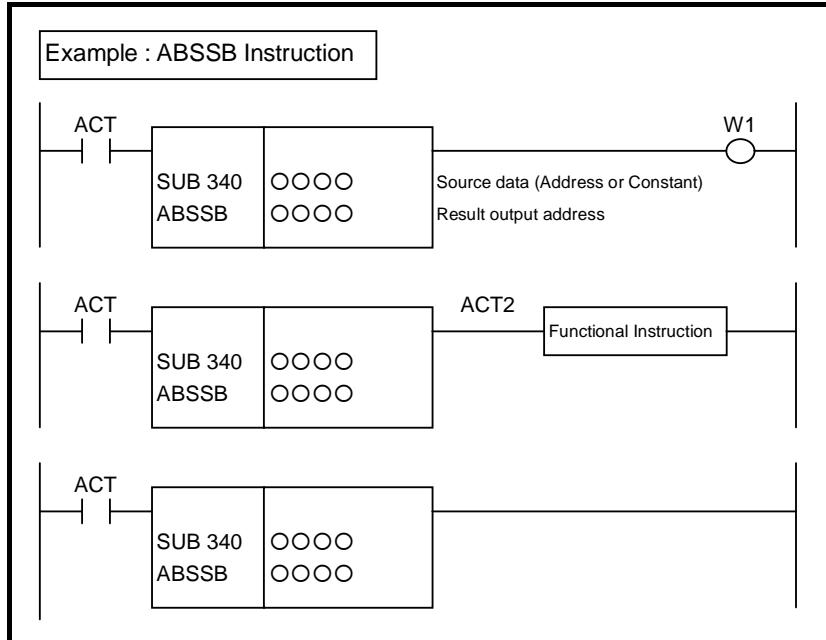


Fig. 4.10.18 Format of ABSSB, ABSSW, ABSSD instruction

Table 4.10.18(b) Mnemonic of ABSSB, ABSSW, ABSSD instruction
Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks
1	RD	0000 .0		ACT
2	SUB	340		SUB No. (ABSSB instruction)
3	(PRM)	0000		Source data (Address or Constant)
4	(PRM)	0000		Result output address
5	WRT	0000 .0		Normal end output

ST3	ST2	ST1	ST0
			ACT
			W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Source data

Specify source data to be converted to an absolute value. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Result output address

Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.
W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.10.19 NEGSB (Sign Inversion (1 Byte Length) : SUB 343) NEGSW (Sign Inversion (2 Bytes Length) : SUB 344) NEGSD (Sign Inversion (4 Bytes Length) : SUB 345)

The Sign inversion instruction inverts the sign of signed binary data.

In "Source data", a constant or a PMC memory address for storing data can be specified.

As indicated below, three types of Sign inversion instructions are available according to the type of data to be operated. In each instruction, "Source data" and the data at "Result output address" are of the same data type.

Table 4.10.19 (a) Kinds of Sign inversion instruction

	Instruction name	SUB No.	Data type
1	NEGSB	343	1 byte length signed binary data
2	NEGSW	344	2 bytes length signed binary data
3	NEGSD	345	4 bytes length signed binary data

If an operation results in an overflow (if the minimum value of a data type is converted), the maximum value of the data type is output to "Result output address", and W1=0 is set.

Format

Fig. 4.10.19 shows the ladder format and Table 4.10.19(b) shows the mnemonic format.

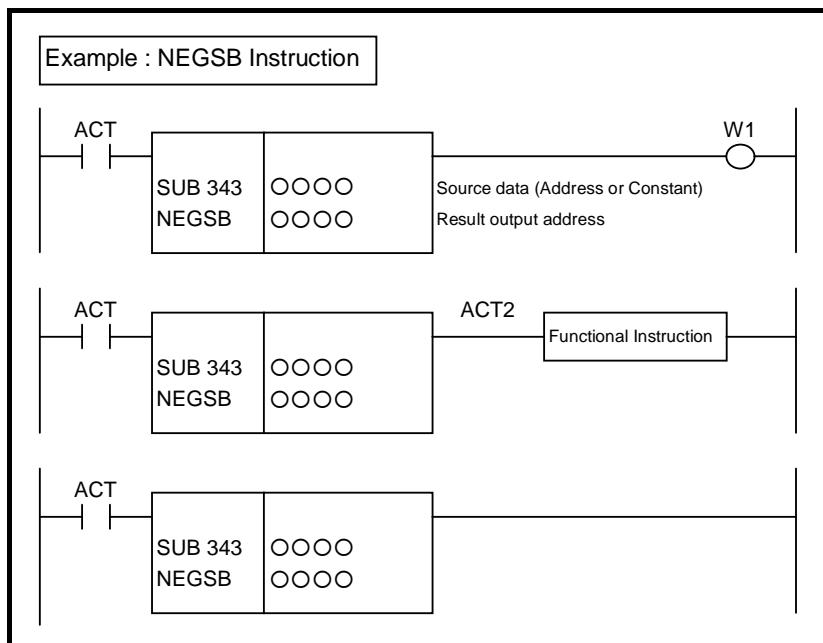


Fig. 4.10.19 Format of NEGSB, NEGSW, NEGSD instruction

Table 4.10.19(b) Mnemonic of NEGSB, NEGSW, NEGSD instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	OOOO .O		ACT				ACT
2	SUB		343	SUB No. (NEGSB instruction)				
3	(PRM)	OOOO		Source data (Address or Constant)				
4	(PRM)	OOOO		Result output address				
5	WRT	OOOO .O		Normal end output				W1

Control conditions

- (a) Input signal (ACT)
 - ACT = 0: Instruction not executed.
 - ACT = 1: Executed.

Parameters

- (a) Source data

Specify source data whose sign is to be inverted. In this parameter, a constant or a PMC memory address for storing data can be specified.
- (b) Result output address

Specify the address to which the result of operation is to be output.

Output (W1)

W1=1 is set when an operation is terminated normally.

W1=0 is set when no operation is executed (ACT=0) or when an operation results in an overflow.

NOTE

W1 may be omitted. Moreover, an other functional instruction can be connected instead of a coil.

4.11 PROGRAM CONTROL

The following types of program control instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	COM	9	Common line control
2	COME	29	Common line control end
3	JMP	10	Jump
4	JMPE	30	Jump end
5	JMPB	68	Label jump 1
6	JMPC	73	Label jump 2
7	LBL	69	Label
8	CALL	65	Conditional subprogram call
9	CALLU	66	Unconditional subprogram call
10	SP	71	Subprogram
11	SPE	72	End of a subprogram
12	END1	1	End of a first level program
13	END2	2	End of a second level program
14	END3	48	End of a third level program
15	END	64	End of a ladder program
16	NOP	70	No operation
17	CS	74	Case call
18	CM	75	Sub program call in case call
19	CE	76	End of case call

4.11.1 COM (Common Line Control: SUB 9)

The coils in a region up to the common line control end instruction (COME) are controlled. Set 0 for the number of coils, and specify the range to be controlled using the common line control end instruction.

If the common line control end instruction is not specified, the "COM FUNCTION MISSING" error results.

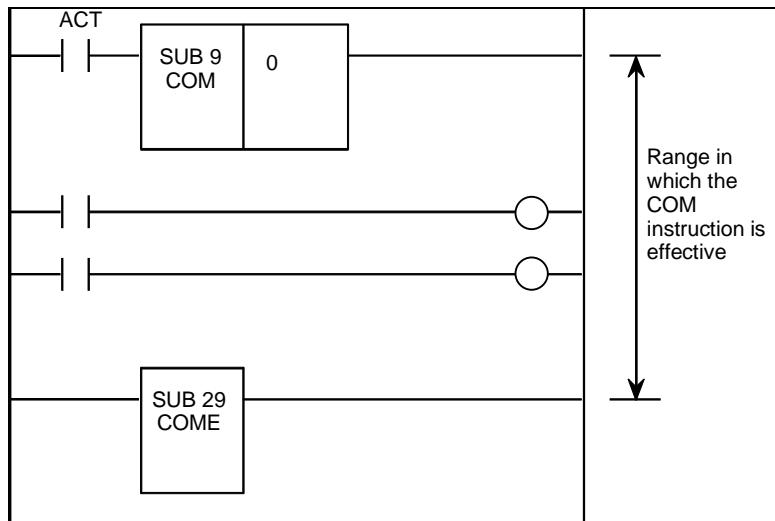


Fig. 4.11.1 (a) Function of COM instruction

Format

Fig. 4.11.1 (b) shows the ladder format and Table 4.11.1 shows the mnemonic format.

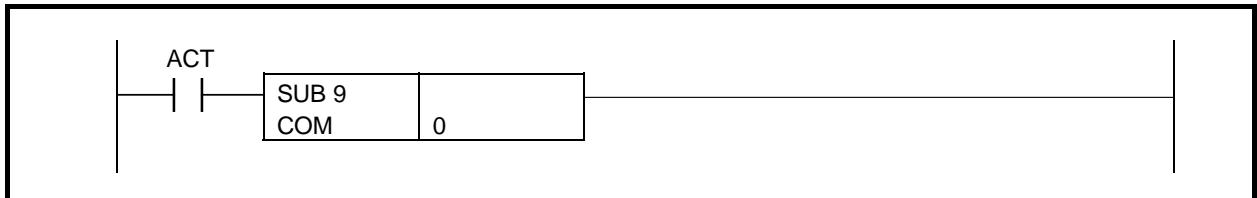


Fig. 4.11.1 (b) Format of COM instruction

Table 4.11.1 Mnemonic of COM instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		9	COM instruction				
3	(PRM)		0	Specify 0.				↓

Control conditions

ACT=0: The coils within the region specified are unconditionally turned off (set to 0).

ACT=1: The program operates in the same way as when COM is not used.

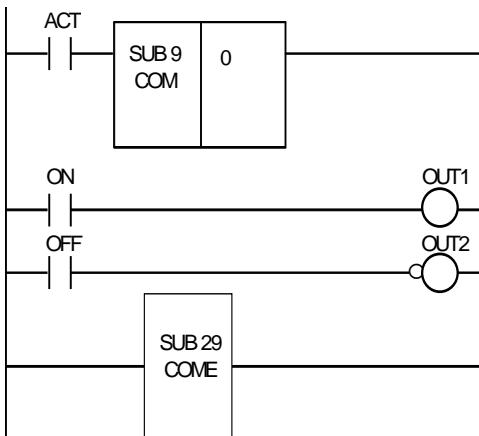
Parameter

(a) Specify 0. (Range specification only)

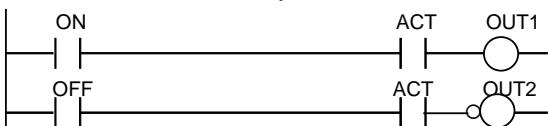
⚠ CAUTION

1 Operation of the COM instruction

Suppose a ladder diagram that includes the COM instruction, as shown below.



For the "OUTx" coils, the COM instruction makes the above ladder diagram similar to the ladder description shown below.

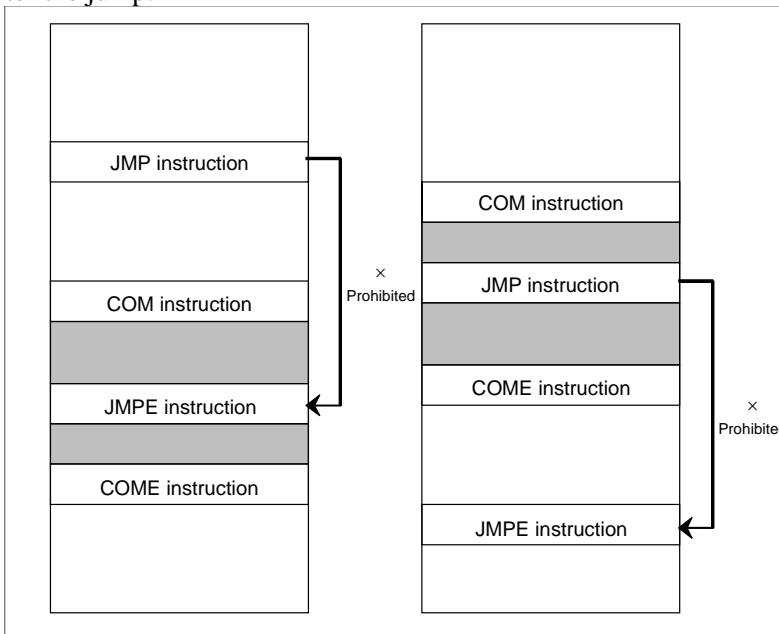


A functional instruction in a range specified by COM executes processing, regardless of COM ACT. However, if COM ACT=0, the coil of the execution result becomes 0.

- 2 Another COM instruction cannot be specified in the range by the COM instruction.
- 3 If COM ACT=0, the coil of a WRT.NOT instruction in the range specified by COM becomes 1 unconditionally as described in 1 above.

Caution

Do not create a program in which a combination of JMP and JMPE instructions is used to cause a jump to and from a sequence between the COM and COME instructions; the ladder sequence may not be able to operate normally after the jump.



4.11.2 COME (Common Line Control End: SUB 29)

This instruction indicates the division in the region specification of the common line control instruction (COM).

This instruction cannot be used alone. It must be used together with the COM instruction.

Format

Fig. 4.11.2 shows the ladder format and Table 4.11.2 shows the mnemonic format.

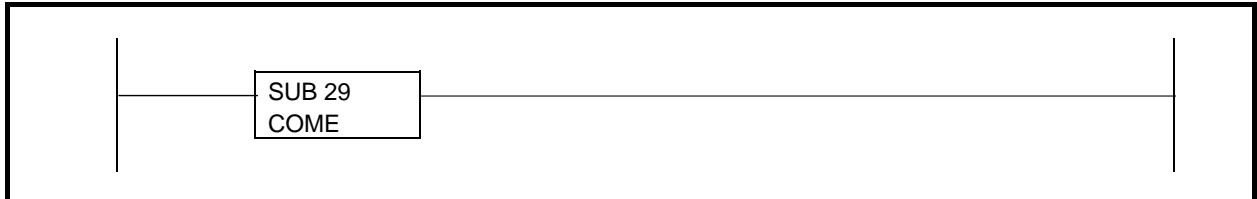


Fig. 4.11.2 Format of COME instruction

Table 4.11.2 Mnemonic of COME instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	29		COME instruction				

4.11.3 JMP (Jump: SUB 10)

The JMP instruction causes a departure from the normal sequence to executing instructions. When a JMP instruction is specified, processing jumps to a jump end instruction (JMPE) without executing the logical instructions (including functional instructions) in the range delimited by a jump end instruction (JMPE). (See Fig. 4.11.3 (a).) Specify a range to be skipped using the jump end instruction.

When the jump end instruction is not specified, the message JUMP FUNCTION MISSING is displayed.

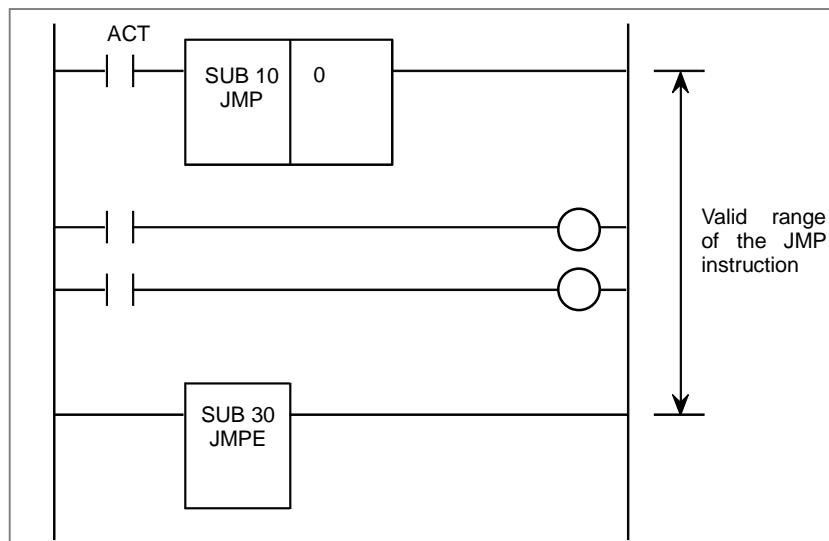


Fig. 4.11.3 (a) Function of JMP instruction

Format

Fig. 4.11.3 (b) shows the ladder format and Table 4.11.3 shows the mnemonic format.

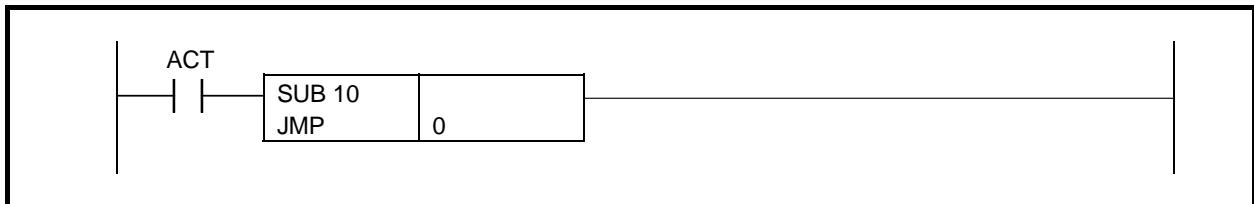


Fig. 4.11.3 (b) Format of JMP instruction

Table 4.11.3 Mnemonic of JMP instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		10	JMP instruction				
3	(PRM)		0	Specify 0.				↓

Control conditions

ACT=1: The logical instructions (including functional instructions) in the specified range are skipped; program execution proceeds to the next step.

ACT=0: The same operation as when JMP is not used is performed.

Parameters

(a) Specify 0. (Range specification only)

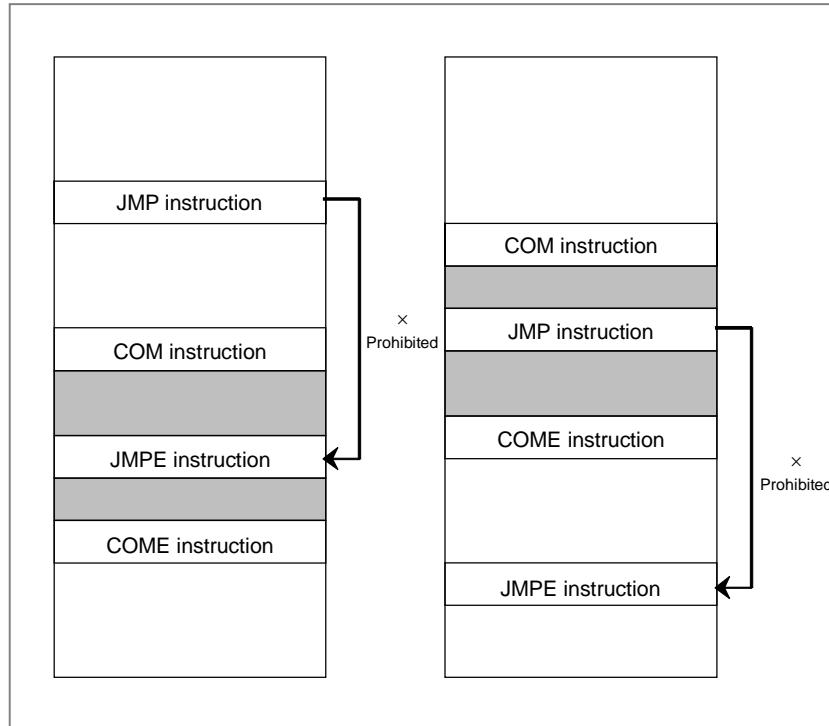
NOTE

JMP instruction operation

When ACT = 1, processing jumps to a jump end instruction (JMPE); the logical instructions (including functional instructions) in the specified jump range are not executed. This instruction can reduce the Ladder execution period (scan time).

Caution

Do not create a program in which a combination of JMP and JMPE instructions is used to cause a jump to and from a sequence between the COM and COME instructions; the ladder sequence may not be able to operate normally after the jump.



4.11.4 JMPE (Jump End: SUB 30)

This instruction indicates the division in the region specification of the jump instruction (JMP). It cannot be used alone. It must be used together with the JMP instruction.

Format

Fig. 4.11.4 shows the ladder format and Table 4.11.4 shows the mnemonic format.

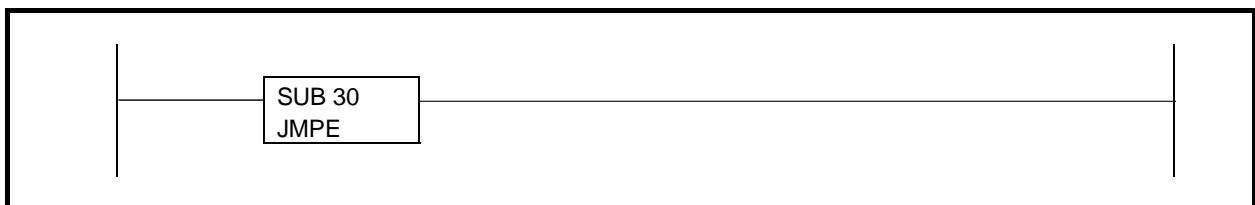


Fig. 4.11.4 Format of JMPE instruction

Table 4.11.4 Mnemonic of JMPE instruction

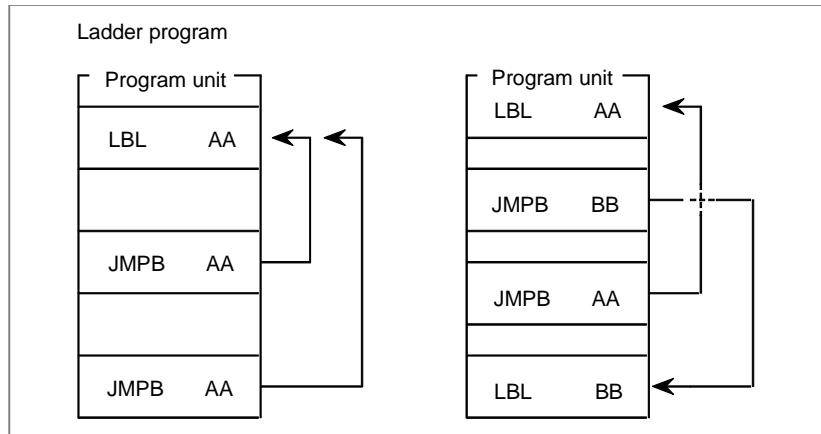
Mnemonic format				Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks			
1	SUB	30		JMPE instruction	ST3	ST2	ST1
							ST0

4.11.5 JMPB (Label Jump 1: SUB 68)

The JMPB functional instruction transfers control to a Ladder immediately after the label set in a Ladder program. The jump instruction can transfer control freely before and after the instruction within the program unit (main program or subprogram) in which the instruction is coded. (See the description of the LBL functional instruction, which is explained later.)

As compared with the conventional JMP functional instruction, JMPB has the following additional functions:

- More than one jump instruction can be coded for the same label.
- Jump instructions can be nested.



Format

Fig. 4.11.5 shows the ladder format and Table 4.11.5 shows the mnemonic format.

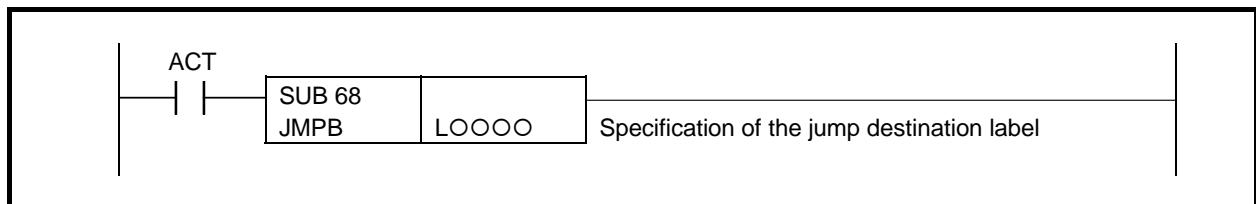


Fig. 4.11.5 Format of JMPB instruction

Table 4.11.5 Mnemonic of JMPB instruction

Mnemonic format					Memory status of control condition
Step number	Instruction	Address No.	Bit No.	Remarks	
1	RD	0000 .0		ACT	
2	SUB	68		JMPB instruction	
3	(PRM)	LOOOO		Specification of the jump destination label	
					ST3 ST2 ST1 ST0
					ACT
					↓

Control conditions

ACT=0: The next instruction after the JMPB instruction is executed.

ACT=1: Control is transferred to the Ladder immediately after the specified label.

Parameters

(a) Label specification

Specifies the label of the jump destination. The label number must be specified in the L address form. A value from L1 to L9999 can be specified.

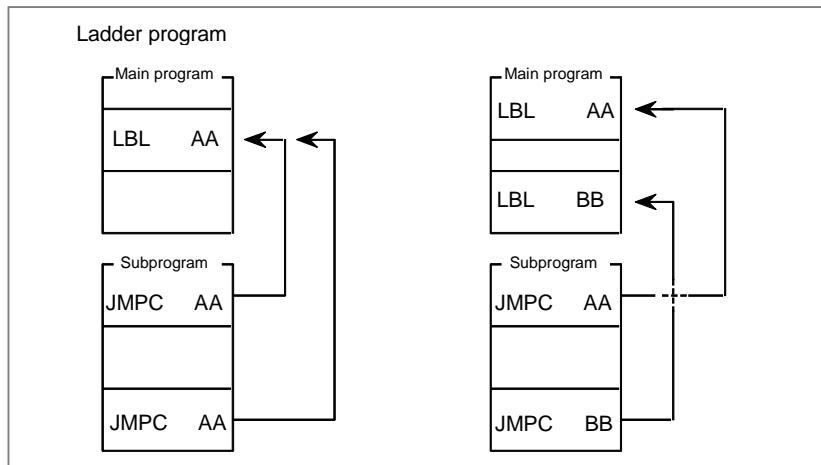
⚠ CAUTION

- 1 For the specifications of this instruction, see the description of functional instruction JMP.
- 2 When this instruction is used to jump back to a previous instruction, care must be taken not to cause an infinite loop.

4.11.6 JMPC (Label Jump 2: SUB 73)

The JMPC functional instruction returns control from a subprogram to the main program. Be sure to code the destination label in the main program. The specifications of this JMPC functional instruction are the same as those of the JMPB functional instruction, except that JMPC always returns control to the main program.

- More than one jump instruction can be coded for the same label.



Format

Fig. 4.11.6 shows the ladder format and Table 4.11.6 shows the mnemonic format.

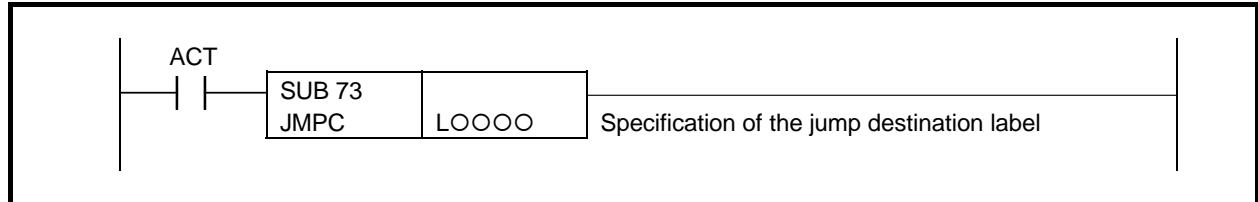


Fig. 4.11.6 Format of JMPC instruction

Table 4.11.6 Mnemonic of JMPC instruction

Mnemonic format					Memory status of control condition
Step number	Instruction	Address No.	Bit No.	Remarks	
1	RD	0000 .0		ACT	
2	SUB		73	JMPC instruction	
3	(PRM)		LOOOO	Specification of the jump destination label	
					ST3 ST2 ST1 ST0
					ACT
					↓

Control conditions

ACT=0: The instruction after the JMPC instruction is executed.

ACT=1: Control is transferred to the Ladder after the specified label.

Parameters

(a) Label specification

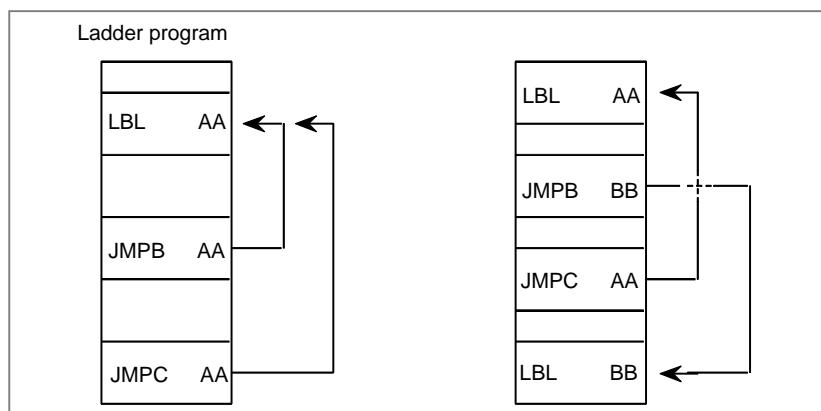
Specifies the label of the jump destination. The label number must be specified in the L address form. A number from L1 to L9999 can be specified.

⚠ CAUTION

- 1 For the specifications of this instruction, see the description of functional instruction JMP.
- 2 When this instruction is used to jump back to a previous instruction, care must be taken not to cause an infinite loop.

4.11.7 LBL (Label: SUB 69)

The LBL functional instruction specifies a label in a Ladder program. It specifies the jump destination for the JMPB and JMPC functional instructions. (See the explanation of the JMPB and JMPC functional instructions.)



Format

Fig. 4.11.7 shows the ladder format and Table 4.11.7 shows the mnemonic format.

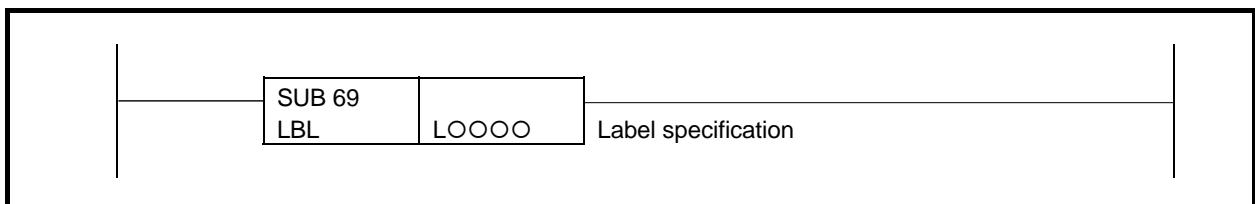


Fig. 4.11.7 Format of LBL instruction

Table 4.11.7 Mnemonic of LBL instruction

Mnemonic format

Memory status of control condition

Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	69		LBL instruction				
2	(PRM)	LOOOO		Label specification				

Parameters

(a) Label specification

Specifies the jump destination for the JMPB and JMPC functional instructions. The label number must be specified in the L address form. A label number from L1 to L9999 can be specified. Up to 256 labels can be used in the main programs (level 1, level 2 and level3). Up to 256 labels can be used in a sub program. The label number may be used to overlap in between the main program and sub program, or in some sub programs.

NOTE

For the use of this instruction, see the description of functional instruction JMPB and JMPC.

4.11.8 CALL (Conditional Subprogram Call: SUB 65)

The CALL functional instruction calls a subprogram. When a subprogram number is specified in CALL, a jump occurs to the subprogram if a condition is satisfied.

Format

Fig. 4.11.8 shows the ladder format and Table 4.11.8 shows the mnemonic format.

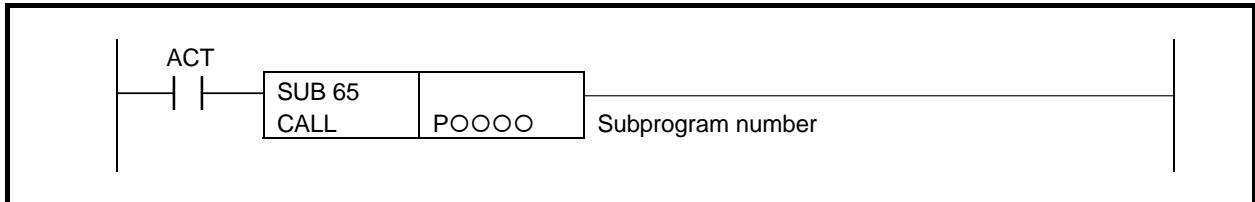


Fig. 4.11.8 Format of CALL instruction

Table 4.11.8 Mnemonic of CALL instruction

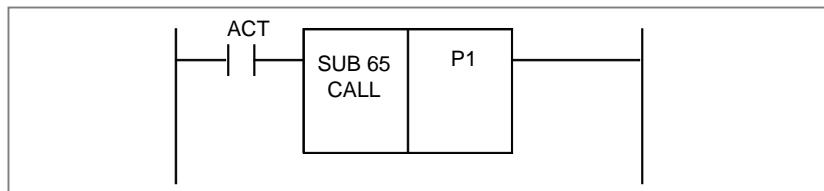
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	RD	0000 .0		ACT				ACT
2	SUB		65	CALL instruction				
3	(PRM)	P0000		Subprogram number				↓

Control conditions

- (a) Input signal
ACT=0: The CALL instruction is not executed.
ACT=1: The CALL instruction is executed.

Parameters

- (a) Subprogram number
Specifies the subprogram number of a subprogram to be called. The subprogram number must be specified in the P address form. **Example:** To call subprogram 1


CAUTION

Be careful when using the CALL instruction with the COM, COME, JMP, or JMPE functional instruction.

For details, see Subsection 1.4.4.

4.11.9 CALLU (Unconditional Subprogram Call: SUB 66)

The CALLU functional instruction calls a subprogram. When a subprogram number is specified, a jump occurs to the subprogram.

Format

Fig. 4.11.9 shows the ladder format and Table 4.11.9 shows the mnemonic format.

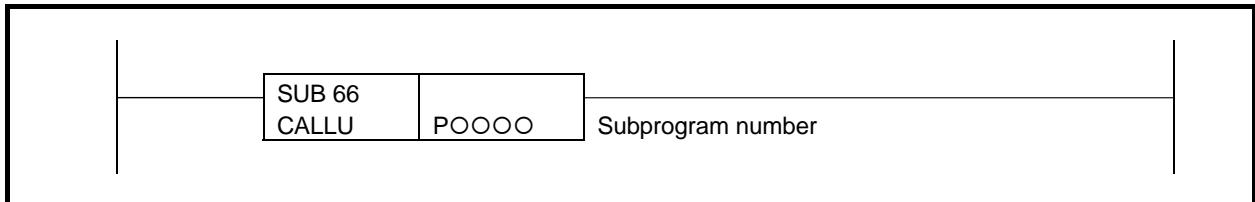


Fig. 4.11.9 Format of CALLU instruction

Table 4.11.9 Mnemonic of CALLU instruction

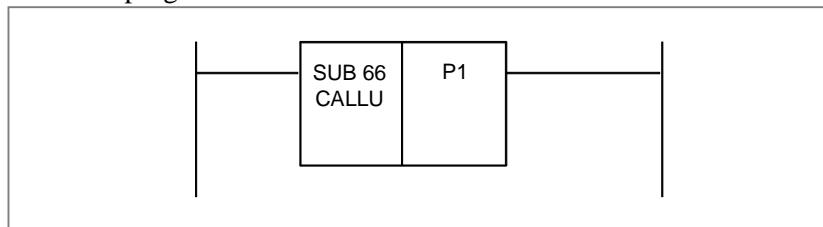
Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	66		CALLU instruction				
2	(PRM)	POOOO		Subprogram number				

Parameters

- (a) Subprogram number

Specifies the subprogram number of a subprogram to be called. The subprogram number must be specified in the P address form.

Example: To call subprogram 1



4.11.10 SP (Subprogram: SUB 71)

The SP functional instruction is used to create a subprogram. A subprogram number is specified as a subprogram name. SP is used with the SPE functional instruction (mentioned later) to specify the subprogram range.

Format

Fig. 4.11.10 shows the ladder format and Table 4.11.10 shows the mnemonic format.

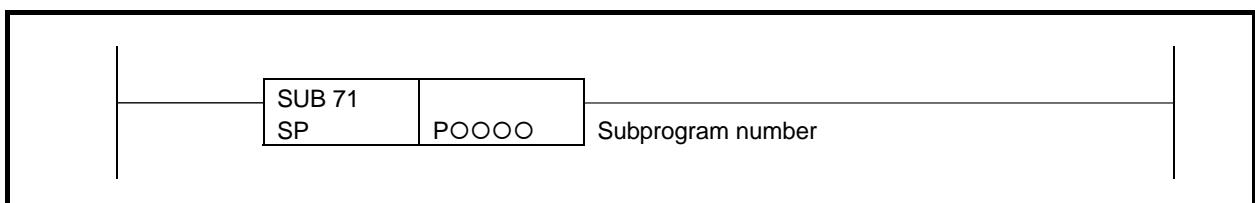


Fig. 4.11.10 Format of SP instruction

Table 4.11.10 Mnemonic of SP instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	71		SP instruction				
2	(PRM)	POOOO		Subprogram number				

Parameters

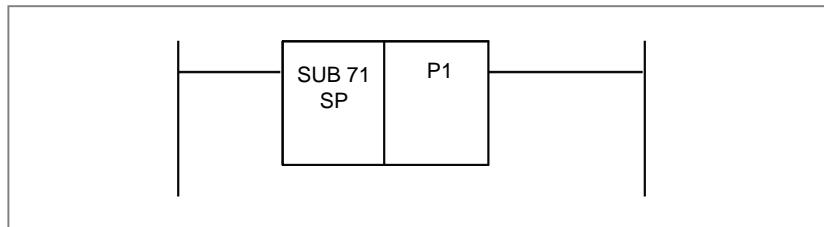
- (a) Subprogram number

Specifies the subprogram number of a subprogram to be coded following this instruction. The subprogram number must be specified in the P address form.

1st to 5th path PMC				Dual check safety PMC
PMC Memory-A	PMC Memory-B	PMC Memory-C	PMC Memory-D	
P1 to P512	P1 to P5000	P1 to P5000	P1 to P5000	P1 to P512

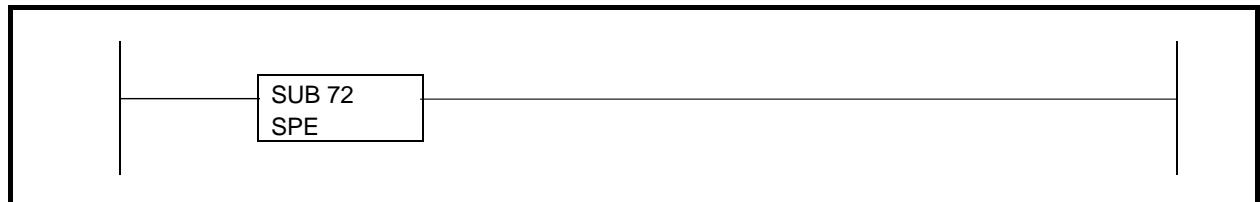
The specified subprogram number must be unique within the sequence program.

Example: When the subprogram number is set to 1



4.11.11 SPE (End of a Subprogram: SUB 72)

The SPE functional instruction is used to create a subprogram. SPE is used with the SP functional instruction. It specifies the range of a subprogram. When this functional instruction has been executed, control is returned to the functional instruction that called the subprogram.

**Fig. 4.11.11 Format of SPE instruction****Table 4.11.11 Mnemonic of SPE instruction**

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	72		SPE instruction				

4.11.12 END1 (1st Level Sequence Program End: SUB 1)

Must be specified once in a sequence program, either at the end of the 1st level sequence, or at the beginning of the 2nd level sequence when there is no 1st level sequence.

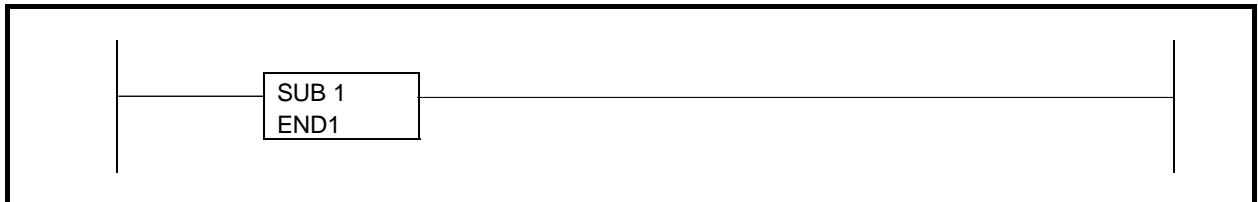


Fig. 4.11.12 Format of END1 instruction

Table 4.11.12 Mnemonic of END1 instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB		1	END1 instruction				

4.11.13 END2 (2nd Level Sequence Program End: SUB 2)

Specify at the end of the 2nd level sequence.

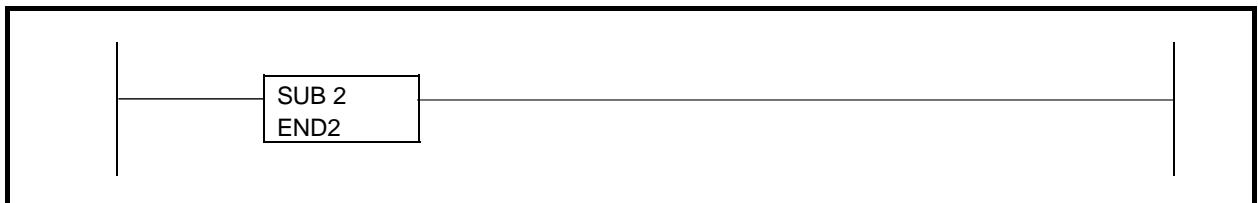


Fig. 4.11.13 Format of END2 instruction

Table 4.11.13 Mnemonic of END2 instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB		2	END2 instruction				

4.11.14 END3 (3rd Level Sequence Program End: SUB 48)

Specify this command at the end of the 3rd level sequence program. If there is no 3rd level sequence program, this instruction need not be specified.

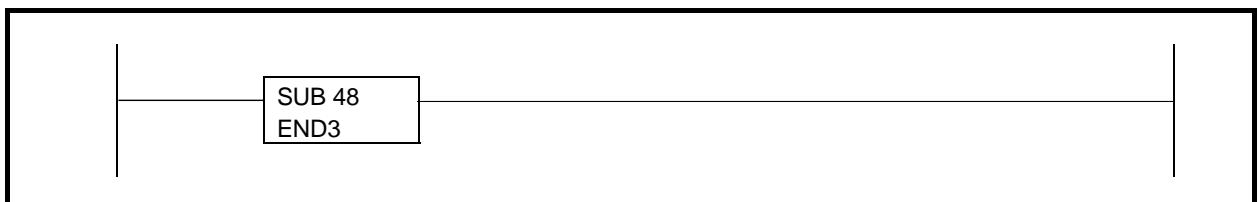


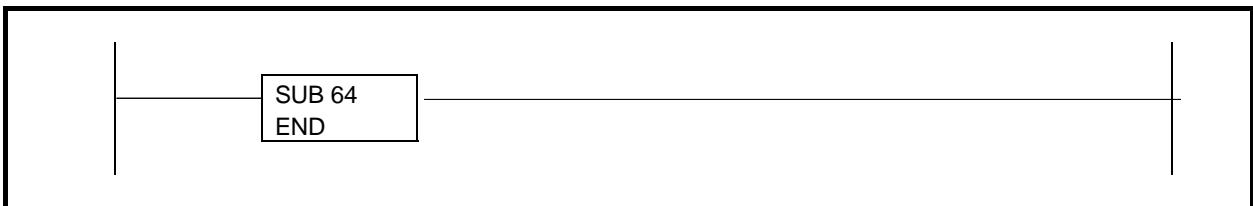
Fig. 4.11.14 Format of END3 instruction

Table 4.11.14 Mnemonic of END3 instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	48		END3 instruction				

4.11.15 END (End of a Ladder Program: SUB 64)

The END functional instruction designates the end of a ladder program. END must be placed at the end of the ladder program.

**Fig. 4.11.15 Format of END instruction****Table 4.11.15 Mnemonic of END instruction**

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	64		END instruction				

4.11.16 NOP (No Operation: SUB 70)

During creation of a ladder program using the programmer, if the program is compiled with specifying the setting with which a net comment or form feed code is used and the point of the net comment is output, position information of the net comment or form feed code is output as the NOP instruction. This instruction performs no operation during execution of the ladder.

4.11.17 CS (Case Call: SUB 74)

The combination of one CS, one or more CM and one CE is used to construct a case call block.

The CS starts the case call block and the CE ends the block. Each CM that should be located between the CS and CE specifies a sub program to be called in each case.

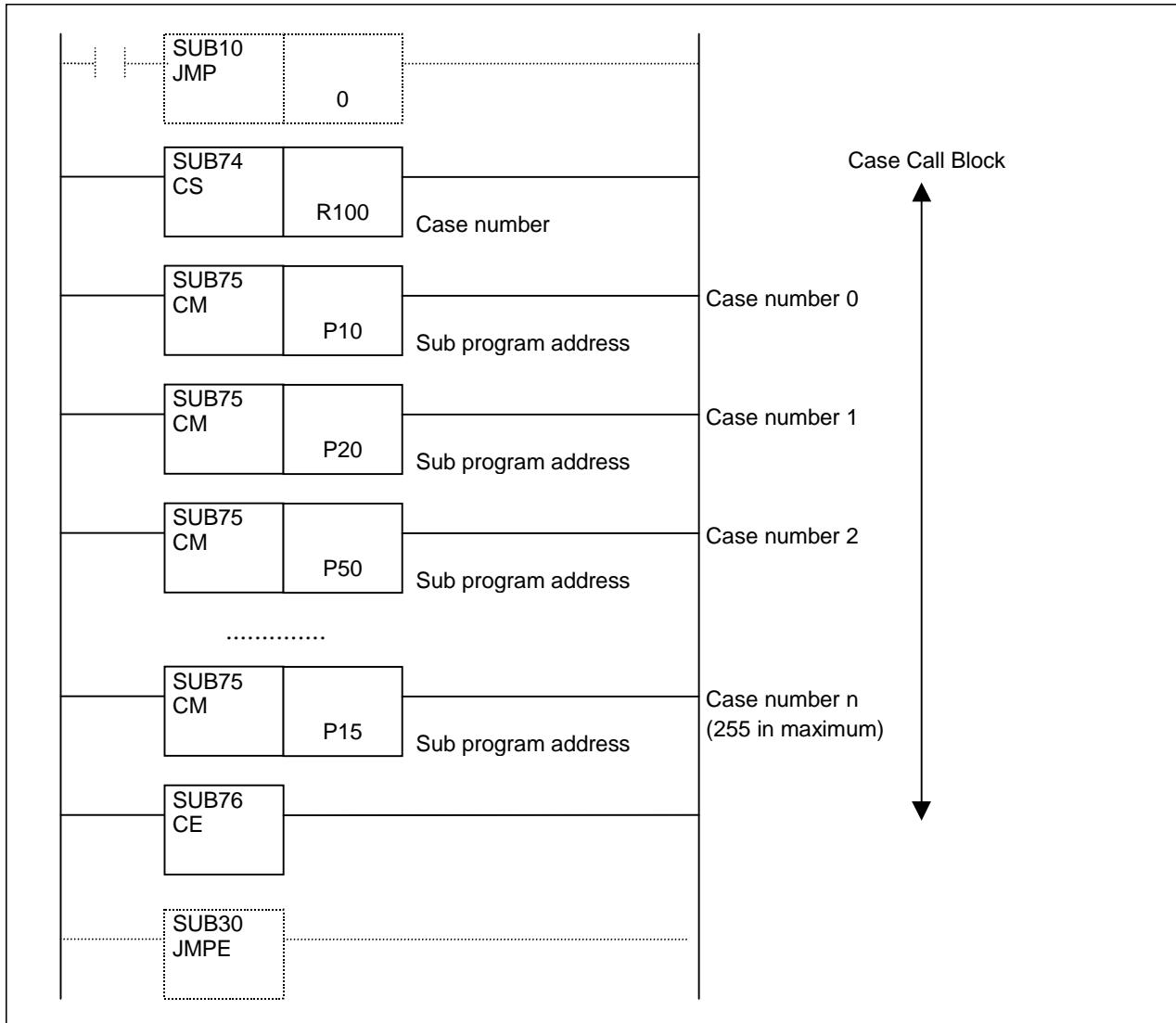
Executing case call block, the CS instruction evaluates the case number from its 1st parameter and only one of CMs that is selected by the case number is activated and calls its associated sub program. When the case number is 0, the 1st CM immediately after CS is executed and certain sub program is called. When the case number is 1, the 2nd CM after CS is executed. The number from 0 to 255 is allowed as the case number. When the case number except 0 through 255 is detected on CS, no sub program is called.

The CM instructions should be programmed immediately after the CS. Other functions except CM must not be programmed between CS and CE. If not so, an error will be detected in compiling process.

The case call block is available only in LEVEL2 and outside of a COM and COME block where normal subprogram call instructions such as CALL and CALLU are allowed. The case call block can be programmed wherever normal subprogram call instructions can be programmed.

In the following example program, the sub program corresponding to the case number is called.

- R100 = 0 The sub program P10 is called.
- R100 = 1 The sub program P20 is called.
- R100 = 2 The sub program P50 is called.
- R100 = n The sub program P15 is called.



Format

Fig. 4.11.17 shows the ladder format and Table 4.11.17 shows the mnemonic format.

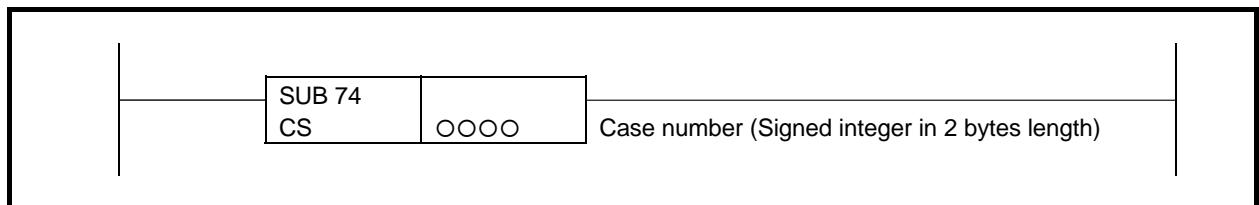


Fig. 4.11.17 Format of CS instruction

Table 4.11.17 Mnemonic of CS instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	74		CS instruction				
2	(PRM)	0000		Case number (Address)				

Parameters

- (a) Case number

Set the address or symbol of the variable in which the case number is stored and commanded. The data type is signed integer in 2 bytes length.

NOTE

Case number is evaluated by CS only once in every cycle. Even if you change the case number in the sub program which is called by the case call block, this change becomes effective in next cycle. This means that only one or no sub program is called in each case call block in each cycle.

4.11.18 CM (Sub Program Call in Case Call: SUB 75)

The combination of one CS, one or more CM and one CE is used to construct a case call block.

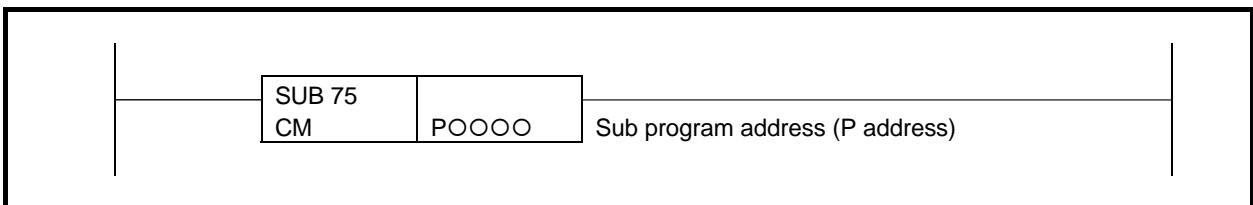
The CM that should be located between the CS and CE is used to specify a sub program to be called when the case number meets the condition.

See the section 4.11.17 CS (Case Call: SUB 74) in details.

Format

Fig. 4.11.18 shows the ladder format and Table 4.11.18 shows the mnemonic format.

The CM should be programmed immediately after the CS .Other functions except CM must not be programmed between CS and CE.

**Fig. 4.11.18 Format of CM instruction****Table 4.11.18 Mnemonic of CM instruction**

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	75		CM instruction				
2	(PRM)	POOOO		Sub program address (P address)				

Parameters

- (a) Sub program address

Set a P address or symbol of a sub program that is call in the case.

4.11.19 CE (End of Case Call: SUB 76)

The combination of one CS, one or more CM and one CE is used to construct a case call block.

The CE ends the case call block.

See the section 4.11.17 CS (Case Call: SUB 74) in details.

Format

Fig.4.11.19 shows the ladder format and Table 4.11.19 shows the mnemonic format.

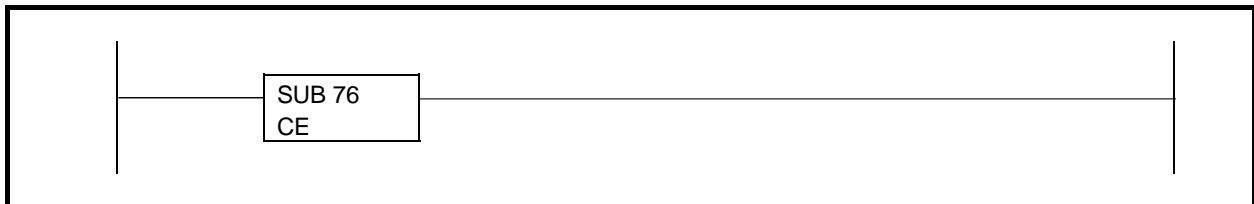


Fig. 4.11.19 Format of CE instruction

Table 4.11.19 Mnemonic of CE instruction

Mnemonic format					Memory status of control condition			
Step number	Instruction	Address No.	Bit No.	Remarks	ST3	ST2	ST1	ST0
1	SUB	76		CE instruction				

4.12 ROTATION CONTROL

The following types of rotation control instruction are available. Use any of these instructions as appropriate for your purpose.

	Instruction name	Sub number	Processing
1	ROT	6	Rotation control
2	ROTB	26	Binary rotation control

4.12.1 ROT (Rotation Control: SUB 6)

Controls rotors, such as the tool post, ATC, rotary table, etc., and is used for the following functions.

- (a) Selection of the rotation direction via the shorter path
- (b) Calculation of the number of steps between the current position and the goal position
- (c) Calculation of the position one position before the goal or of the number of steps up to one position before the goal

Format

Fig. 4.12.1 (a) shows the ladder format and Table 4.12.1 shows the mnemonic format.

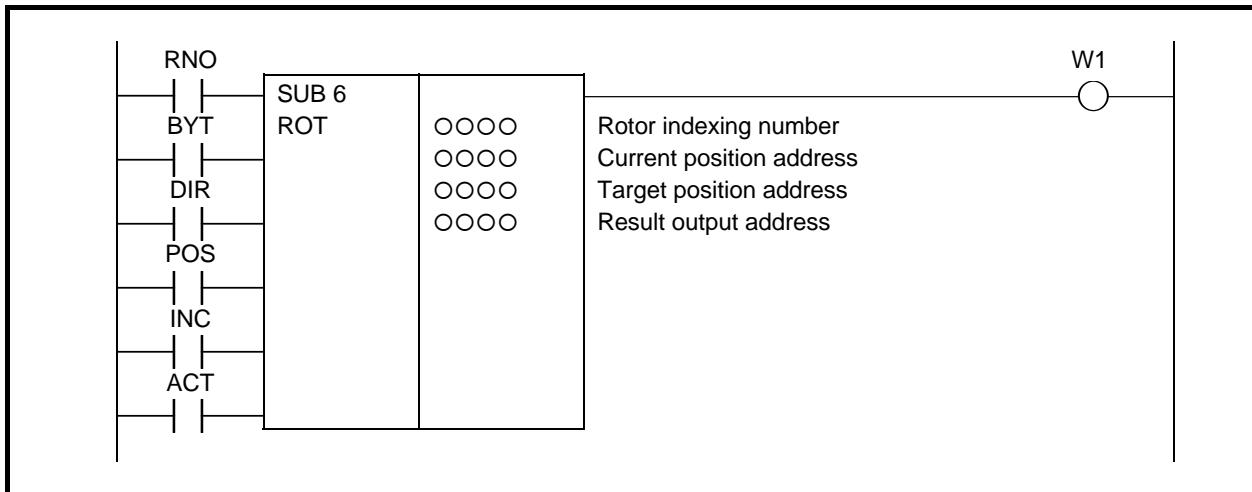


Fig. 4.12.1 (a) Format of ROT instruction

Table 4.12.1 Mnemonic of ROT instruction

Mnemonic format					Memory status of control condition					
Step number	Instruction	Address No.	Bit No.	Remarks	ST5	ST4	ST3	ST2	ST1	ST0
1	RD	0000 .O		RNO						RNO
2	RD.STK	0000 .O		BYT						RNO BYT
3	RD.STK	0000 .O		DIR						RNO BYT DIR
4	RD.STK	0000 .O		POS						RNO BYT DIR POS
5	RD.STK	0000 .O		INC						RNO BYT DIR POS INC
6	RD.STK	0000 .O		ACT						RNO BYT DIR POS INC ACT
7	SUB	6		ROT						
8	(PRM)	0000		Rotor indexing number						
9	(PRM)	0000		Current position address						
10	(PRM)	0000		Target position address						
11	(PRM)	0000		Result output address						
12	WRT	0000 .O		Output of rotation direction						

Control conditions

- (a) Specify the starting number of the rotor. (RN)
 - RNO=0: Begins the number of the position of the rotor with 0.
 - RNO=1: Begins the number of the position of the rotor with 1.
- (b) Specify the number of digits of the process data (position data). (BYT)
 - BYT=0: BCD two digits
 - BYT=1: BCD four digits
- (c) Select the rotation direction via the shorter path or not. (DIR)
 - DIR=0: No direction is selected. The direction of rotation is only forward.
 - DIR=1: Selected. See rotating direction output (W1) described below for details on the rotation direction.
- (d) Specify the operating conditions. (POS)
 - POS=0: Calculates the goal position.
 - POS=1: Calculates the position one position before the goal position.

(e) Specify the position or the number of steps. (INC)

INC=0: Calculates the number of the position. If the position one position before the goal position is to be calculated, specify INC=0 and POS=1

INC=1: Calculates the number of steps. If the difference between the current position and the goal position is to be calculated, specify INC=1 and POS=0.

(f) Execution command (ACT)

ACT=0: The ROT instruction is not executed. W1 does not change.

ACT=1: Executed. Normally, set ACT=0. If the operation results are required, set ACT=1.

Parameters

(a) Rotor indexing number

Specify the rotor indexing number.

(b) Current position address

Specify the address storing the current position.

(c) Target position address

Specify the address storing the target position (or command value).

(d) Result output address

Calculate the number of steps for the rotor to rotate, the number of steps up to the position one position before, or the position before the goal. When the calculating result is to be used, always check that ACT=1.

Rotating direction output (W1)

The direction of rotation for control of rotation via the shorter path is output to W1. When W1=0, the direction is forward (FOR) when 1, reverse (REV). The definition of FOR and REV is shown in Fig. 4.12.1 (b). If the number given to the rotor is ascending, the rotation is FOR; if descending, REV. The address of W1 can be determined arbitrarily. When, however, the result of W1 is to be used, always check that ACT=1.

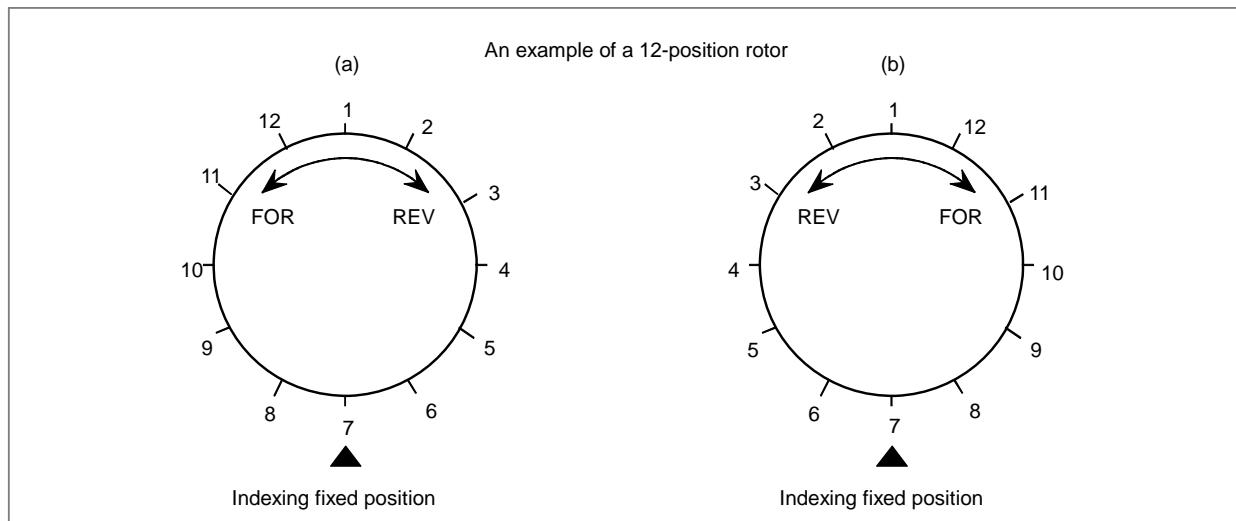


Fig. 4.12.1 (b) Rotation direction

4.12.2 ROTB (Binary Rotation Control: SUB 26)

This instruction is used to control rotating elements including the tool post, ATC (Automatic Tool Changer), rotary table, etc. In the ROT command a parameter indicating the number of rotating element indexing positions is a fixed data in programming. For ROTB, however, you can specify an address for the number of rotating element index positions, allowing change even after programming. The data handled are all in the binary format. Otherwise, ROTB is coded in the same way as ROT.

Format

Fig. 4.12.2 (a) shows the ladder format and Table 4.12.2 shows the mnemonic format.

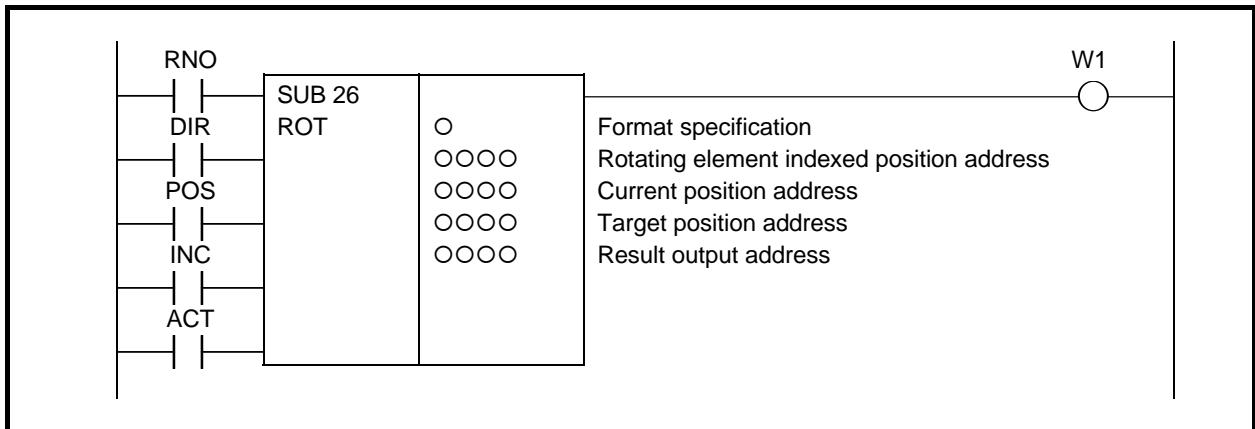


Fig. 4.12.2 (a) Format of ROTB instruction

Table 4.12.2 Mnemonic of ROTB instruction

Step number	Instruction	Address No.	Bit No.	Mnemonic format					Memory status of control condition
				ST4	ST3	ST2	ST1	ST0	
1	RD	0000 .O		RNO					RNO
2	RD.STK	0000 .O		DIR					RNO DIR
3	RD.STK	0000 .O		POS					RNO DIR POS
4	RD.STK	0000 .O		INC					RNO DIR POS INC
5	RD.STK	0000 .O		ACT					RNO DIR POS INC ACT
6	SUB	26		ROTB					
7	(PRM)	O		Format specification					
8	(PRM)	0000		Rotating element indexed position address					
9	(PRM)	0000		Current position address					
10	(PRM)	0000		Target position address					
11	(PRM)	0000		Result output address					
12	WRT	0000 .O		Output of rotation direction					
									W1

Control conditions

The control conditions do not differ basically from those for ROT command. However, BYT has been eliminated from ROTB (it forms part of the ROTB parameters).

For the reset, see ROT.

Parameters

(a) Format specification

Specifies data length (1, 2, or 4 bytes). Use the first digit of the parameter to specify the number of bytes.

1: 1 byte

- 2: 2 bytes
- 4: 4 bytes

All numerical data (number of indexed positions for the rotating elements, current address, etc.) are in the binary format. Therefore, they require the memory space specified by data length.

- (b) Rotating element indexed position address
Specifies the address containing the number of rotary element positions to be indexed.
- (c) Other parameters
For the functions and use of the other parameters, see the ROT instruction.

⚠ CAUTION

Please do not set an illegal value, that is not indicated above, into the "(a) Format specification".

Output for rotational direction (W1)

See the ROT instruction.

Example of using the ROTB instruction

Fig. 4.12.2 (b) illustrates a ladder diagram for a 12-position rotor to be controlled for rotation via the shorter path and for deceleration at the position one position before the goal.

- The goal position is specified with Robot 32B of binary code (address F26 to F29).
- The current position is entered with the binary code signal (address X41) from the machine tool.
- The result of calculating the position one position before the goal is output to address R230 (work area).
- Operation starts with the output TF (address F7.3) from the Robot.
- The binary compare instruction (COMPB) is used to detect the deceleration and stop positions.

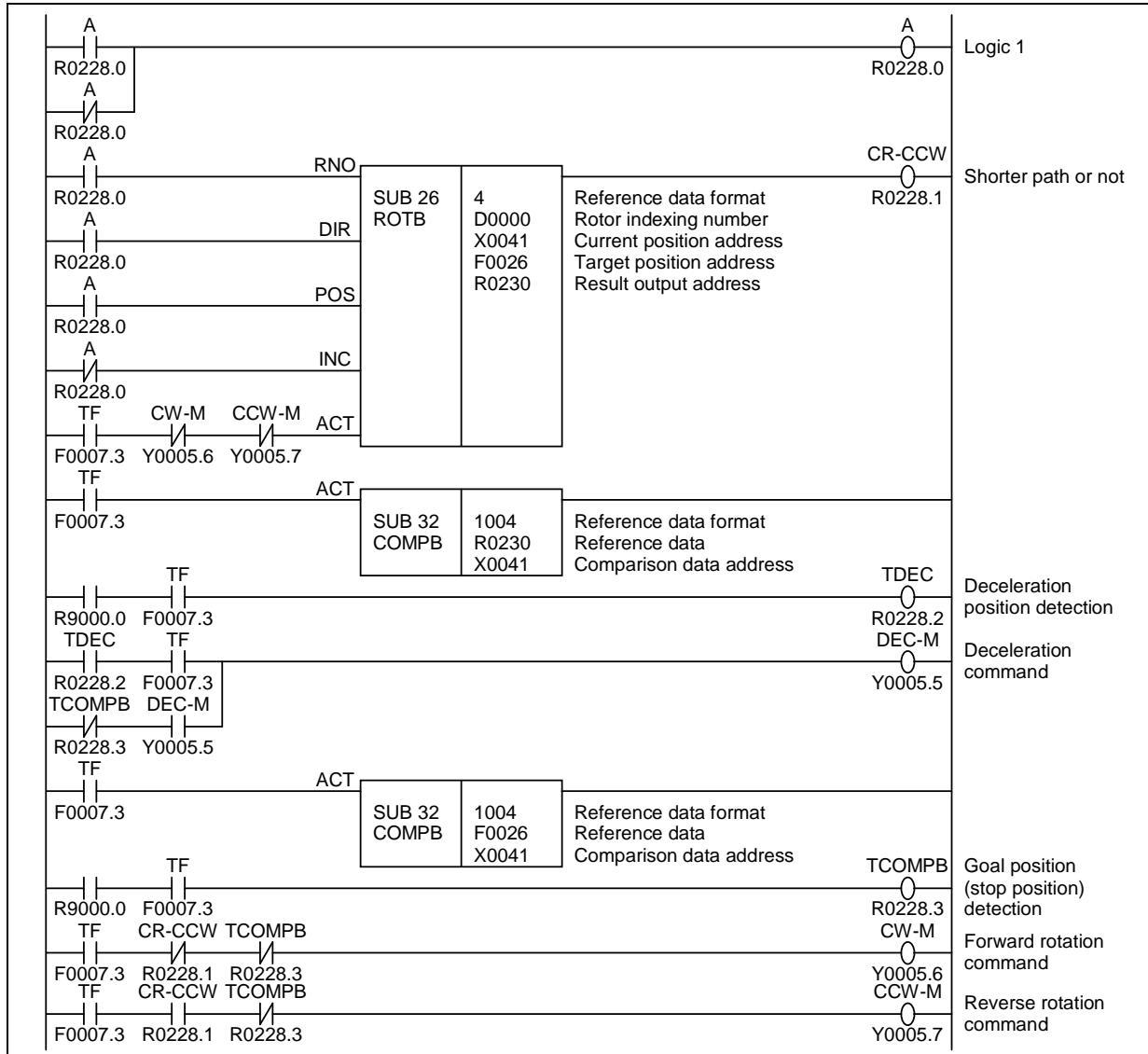


Fig. 4.12.2 (b) Example of a ladder diagram for the ROTB instruction

4.13 INVALID INSTRUCTIONS

The instructions listed below are invalid.

They cause no error but are treated as NOP instructions (which perform no operation when the ladder program is executed).

Instruction name	Sub number	Processing
SPCNT	46	Spindle control
DISP	49	Message display
MMCWR	98	Reading of MMC window data
MMCWW	99	Writing of MMC window data
FNC90	90	Auxiliary functional instruction 1
FNC91	91	Auxiliary functional instruction 2
FNC92	92	Auxiliary functional instruction 3
FNC93	93	Auxiliary functional instruction 4
FNC94	94	Auxiliary functional instruction 5
FNC95	95	Auxiliary functional instruction 6
FNC96	96	Auxiliary functional instruction 7
FNC97	97	Auxiliary functional instruction 8

5 FANUC LADDER-III FOR ROBOT PROGRAMMING

FANUC LADDER-III for Robot is the programming software for developing sequence programs for FANUC PMCs.

This software runs in the Microsoft® software Windows® operating system environment. This manual does not cover basic Windows® operations. If you are a beginner to Windows®, read the Windows® manual first to learn the basic operations.

This manual describes the programming items of FANUC LADDER-III for Robots with respect to the ROBOT CONTROLLER. These items include how to create a new sequence program, how to transfer it to the ROBOT CONTROLLER, how to execute it and how to write it into ROBOT CONTROLLER ROM.

Please order following software to arrange "FANUC LADDER-III for Robot".

Software Name	Order
FANUC LADDER-III for Robot	A08B-9410-J574
FANUC LADDER-III for Robot Upgrade	A08B-9410-J590

"A08B-9410-J590" is software for upgrade. It upgrades previous "FANUC LADDER-III for Robot" to latest software for "R-30iB ROBOT PMC" and "DCS Safety PMC" type. Please order it if previous one is already using.

NOTE

V6.80 or later of "FANUC LADDER-III for Robot" is needed to use "R-30iB ROBOT PMC".

V6.81 or later of "FANUC LADDER-III for Robot" is needed to use "DCS Safety PMC".

NOTE

There is "FANUC LADDER-III" and "FANUC LADDER-III for Robot". Use "FANUC LADDER-III for Robot" to program the Robot Integrated PMC.

Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the United States of America.

Some functions of FANUC LADDER-III for Robot are not supported by the robot controller. The following is the list of menu items not supported.

Menu items not supported

- [Signal Trace] is not available.
- [Online Editor] is not available.
- [IO GROUP SELECTION] cannot be displayed.
- [I/O Module Edit] is not supported.
- [Clear PMC Memory] is not supported.

5.1 CONNECTING FANUC LADDER-III for Robot TO ROBOT CONTROLLER

ROBOT CONTROLLER and FANUC LADDER-III for Robot are connected by RS-232-C or Ethernet to exchange PMC program and parameters.

5.1.1 RS-232-C Connection

Cable

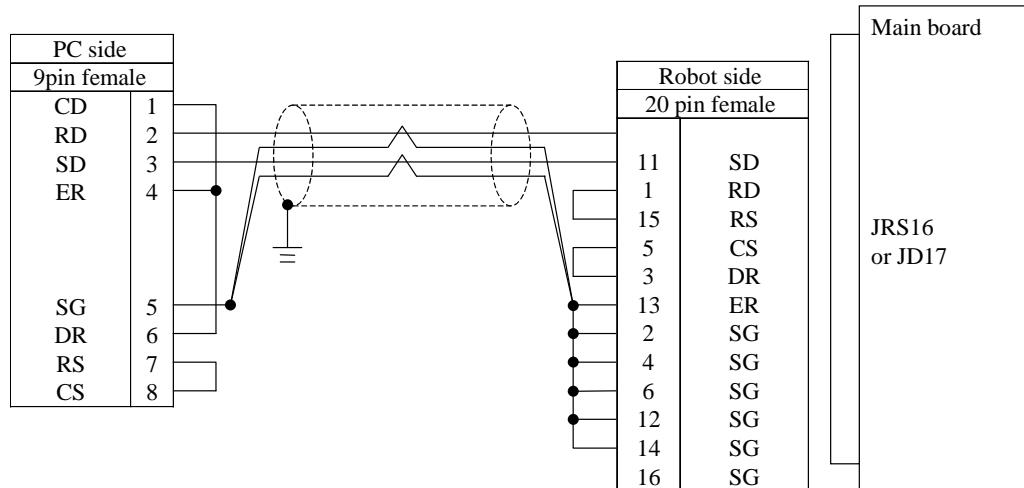
The following diagram shows the connection of RS-232-C.

In case that a RS-232-C port does not exist, please use Ethernet.

A-cabinet

The customer needs to make the cable to connect between ROBOT CONTROLLER (A-cabinet) and FANUC LADDER-III for Robot.

The cable of the ROBOT CONTROLLER side is connected to JRS16 or JD17 connector on the main board.



Specification of the connectors and the cable are as following:

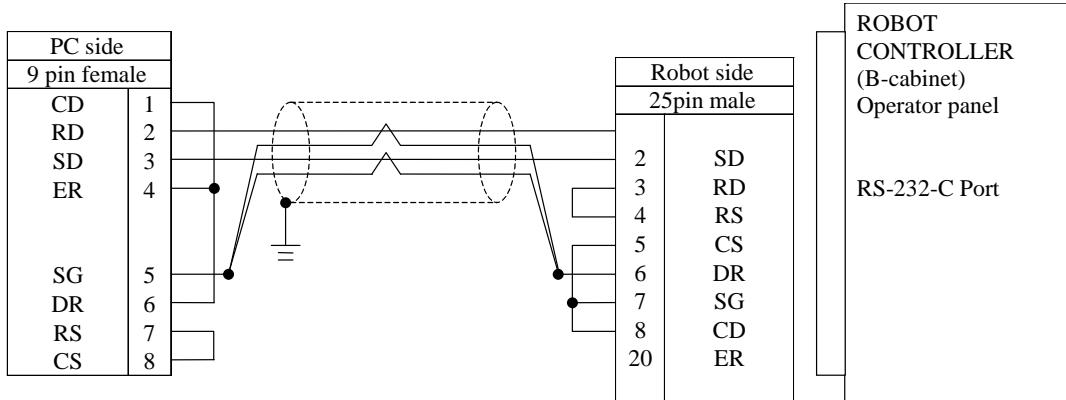
- Connector PCR-E20FS (Honda Tsushin Kogyo)
- Housing PCR-V20LA (Honda Tsushin Kogyo), or compatible connector
- Shielding quality Entirely shielded twisted pair cable is recommended.
For protection against the noise, the twisted pair wire should be used for pairs of RD and SG, SD and SG.

B-cabinet

Use the following cable to connect ROBOT CONTROLLER (B-cabinet) and FANUC LADDER-III for Robot.

PCAT RS-232-C	A02B-0200-K814
---------------	----------------

The D-Sub 25pin is connected to the RS-232-C port in front side of ROBOT CONTROLLER operator panel.



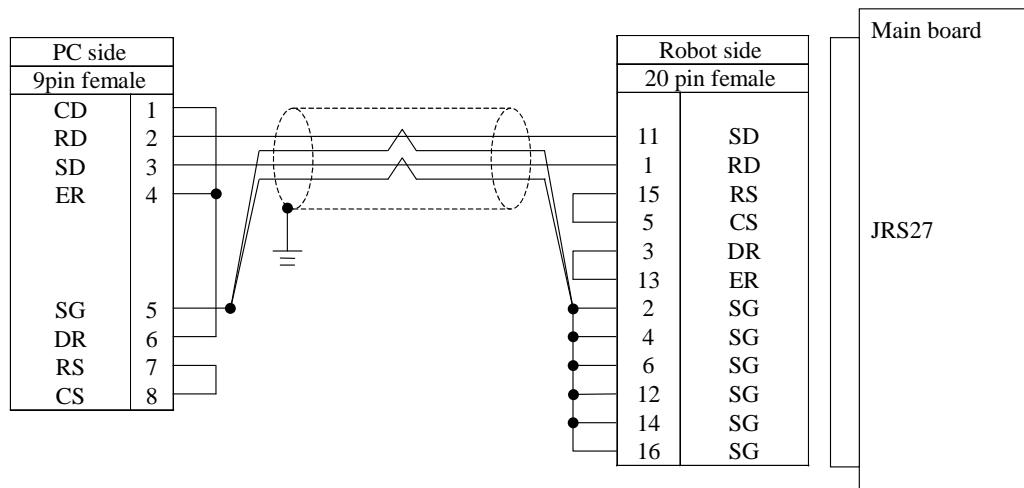
Specification of the connectors and the cable are as following:

- Connector D-Sub25pin male DBM-25P (ANSI/EIA-232)
- Housing DB-C2-J9 (ANSI/EIA-232)
- Shielding quality Entirely shielded twisted pair cable is recommended.
For protection against the noise, the twisted pair wire should be used for pairs of RD and SG, SD and SG.

R-30iB Mate

The customer needs to make the cable to connect between ROBOT CONTROLLER (R-30iB Mate) and FANUC LADDER-III for Robot.

The cable of the ROBOT CONTROLLER side is connected to JRS27 connector on the main board.



Specification of the connectors are as following:

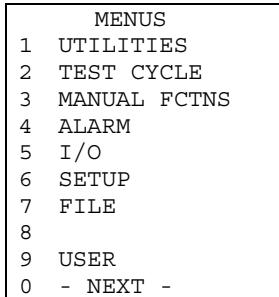
- Connector PCR-E20FS (Honda Tsushin Kogyo)
- Housing PCR-V20LA (Honda Tsushin Kogyo), or compatible connector
- Shielding quality Entirely shielded twisted pair cable is recommended.
For protection against the noise, the twisted pair wire should be used for pairs of RD and SG, SD and SG.

Setting of Serial Port

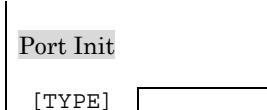
Choose PMC programmer as a port from Teach pendant before using FANUC LADDER-III for Robot online.

For example, followings show the way to setup PMC programmer to port1.

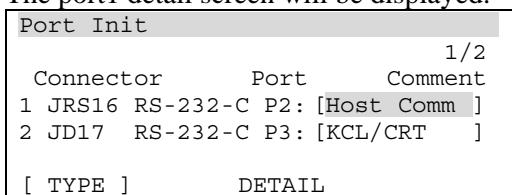
- 1 Press MENU key, select "6.SETUP".



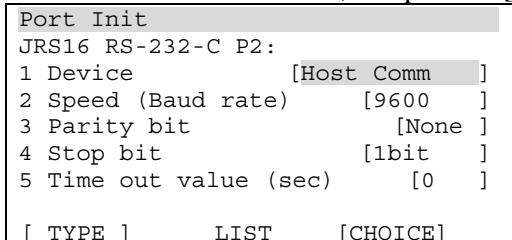
- 2 Press F1[TYPE] select "Port Init".



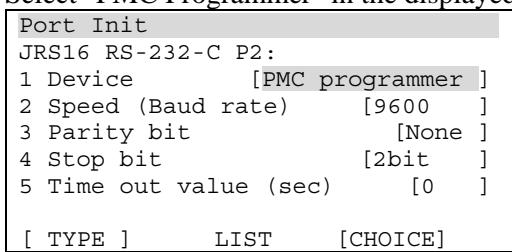
- 3 Move the cursor to "P2" .Press F3(DETAIL).
The port1 detail screen will be displayed.



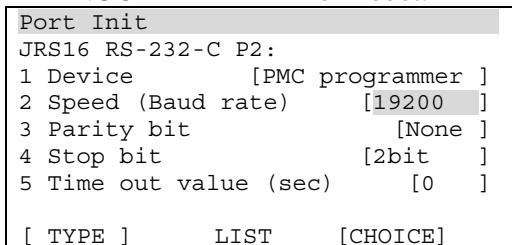
- 4 Move the cursor to 1 Device, and press F4[CHOICE].



- 5 Select "PMC Programmer" in the displayed menu.



- 6 Move the cursor to 2 Speed (Baud rate), and press F4[CHOICE]. Select the same speed as the setting in FANUC LADDER-III for Robot.

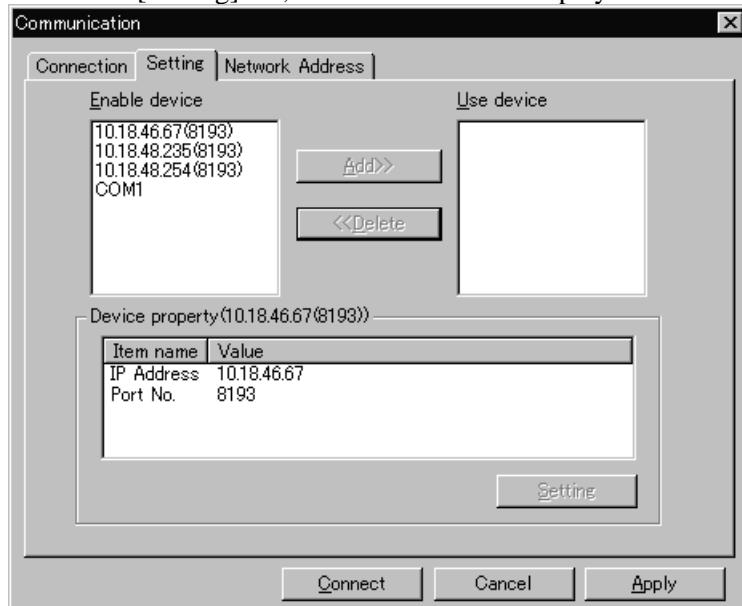


Serial Port communication

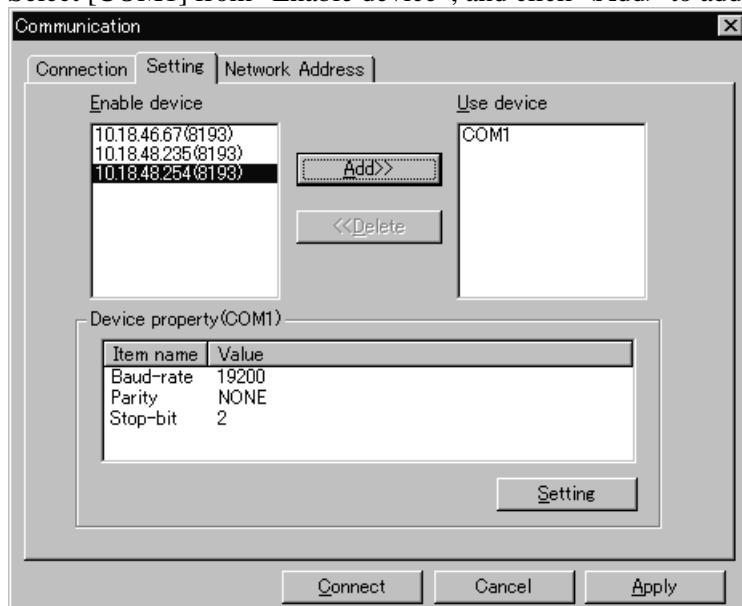
After completing the setup of the Serial Port, Robot PMC can connect with FANUC LADDER-III for Robot.

- 1 Execute the FANUC LADDER-III for Robot and select [Tool]-[Communication].

Next click [Setting] tab, Enable devices are displayed



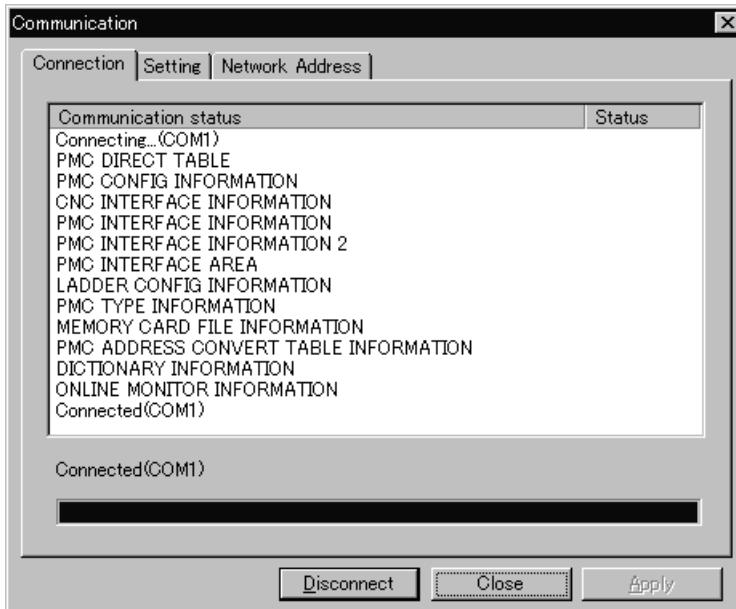
- 2 Select [COM1] from “Enable device”, and click <Add> to add “COM1” to “Use device”.



- 3 Click <Connect> to communicate with the Robot PMC by RS-232-C.
- 4 If “Multi-Path PMC (J763)” or “DCS Safety PMC (J764)” is ordered, PMC type selection screen is displayed. So select PMC type to communicate and click <OK>. (If not ordered Multi-Path PMC (J763) and DCS Safety PMC (J764), skip to process 5.)



- 5 When the connection is completed, "Connected (COM1)" is displayed on the screen.
Click <Close>.



5.1.2 Ethernet Connection

Setting for Ethernet communication

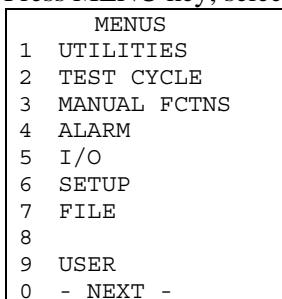
To use FANUC LADDER-III for Robot Online function with Ethernet communication, you must set the Robot IP address in advance.

To set the Robot IP address, refer to "SETTING UP TCP/IP" in "Ethernet function OPERATOR'S MANUAL (B-82974EN)"

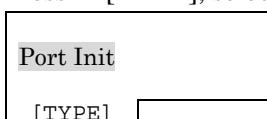
Setting of Serial Port

If the "PMC programmer" is set as a serial port, remove PMC programmer from all ports on the Teach pendant before using FANUC LADDER-III for Robot online.

- 1 Press MENU key, select "6.SETUP".

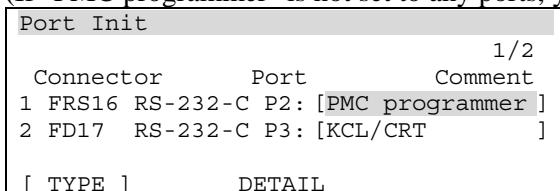


- 2 Press F1[TYPE], select "Port Init".

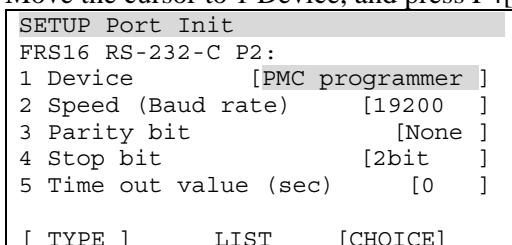


- 3 If a port is set as [PMC programmer], move the cursor to the port. And press F3(DETAIL). So the port detail screen will be displayed.

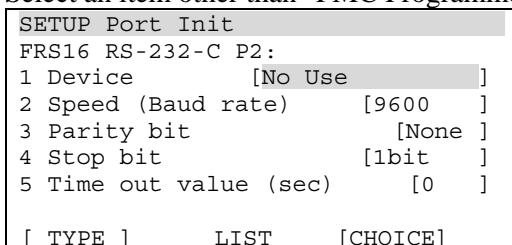
(If "PMC programmer" is not set to any ports, you are finished in the "Port Init" setup screen.)



- 4 Move the cursor to 1 Device, and press F4[CHOICE].



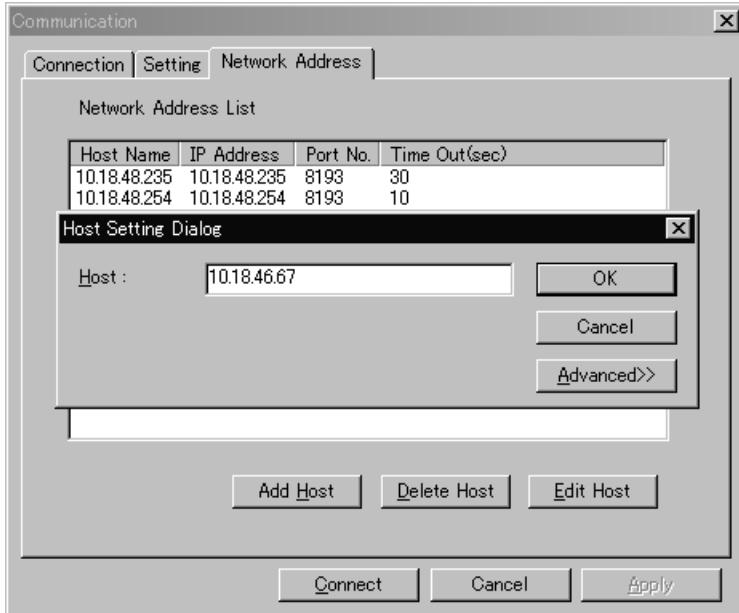
- 5 Select an item other than "PMC Programmer" in the displayed menu (such as "No Use").



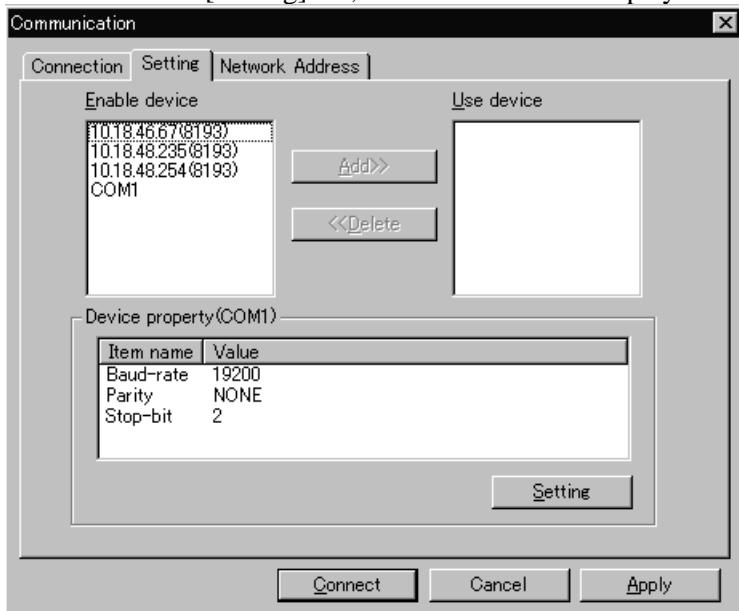
Ethernet connection

When setting of Ethernet connection and serial port are completed, Robot PMC can communicate with FANUC LADDER-III for Robot by Ethernet.

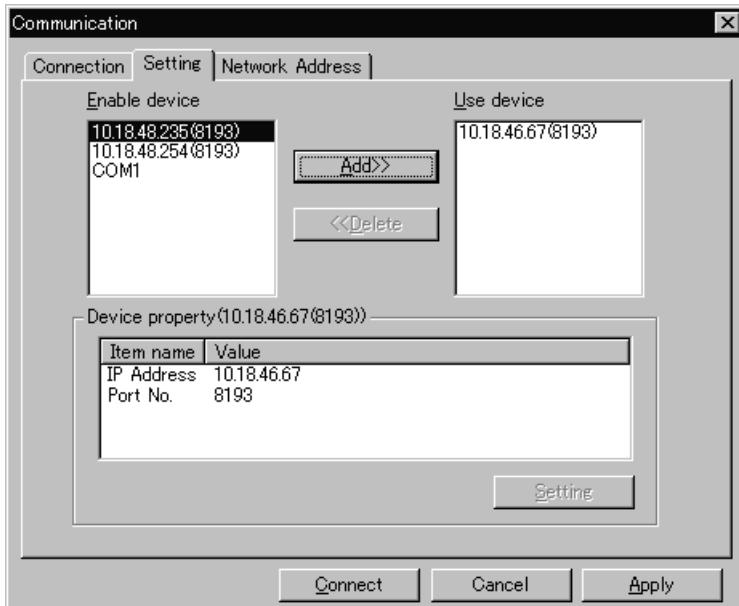
- 1 Execute the FANUC LADDER-III for Robot and select [Tool]-[Communication]. When click [Network Address] tab, Enable devices are displayed
Click <Add Host>, put the Robot IP address, and click <OK>.



- 2 Next when click [Setting] tab, Enable devices are displayed.



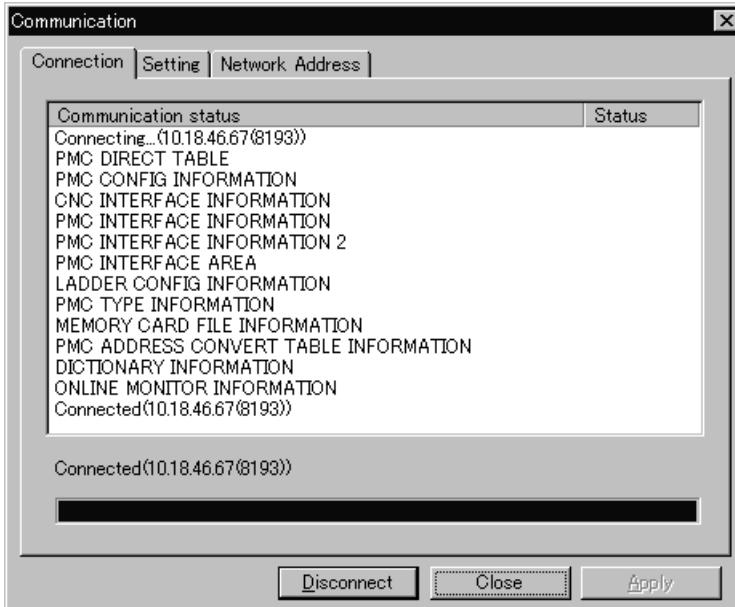
- 3 Select the Robot IP address from Enable devices, and click <Add> to add the IP address to "Use device".



- 4 Click <Connect> to communicate with the Robot PMC by Ethernet.
- 5 If “Multi-Path PMC (J763)” or “DCS Safety PMC (J764)” is ordered, PMC type selection screen is displayed. So select PMC type to communicate and click <OK>. (If not ordered Multi-Path PMC (J763) and DCS Safety PMC (J764), skip this process.)



- 6 When the connection is completed, “Connected (IP address)” is displayed on the screen. Click <Close>.



5.1.3 Connection with ROBOGUIDE

ROBOGUIDE and FANUC LADDER-III for Robot are connected by Ethernet to exchange PMC program and parameters.

Setting for Ethernet communication

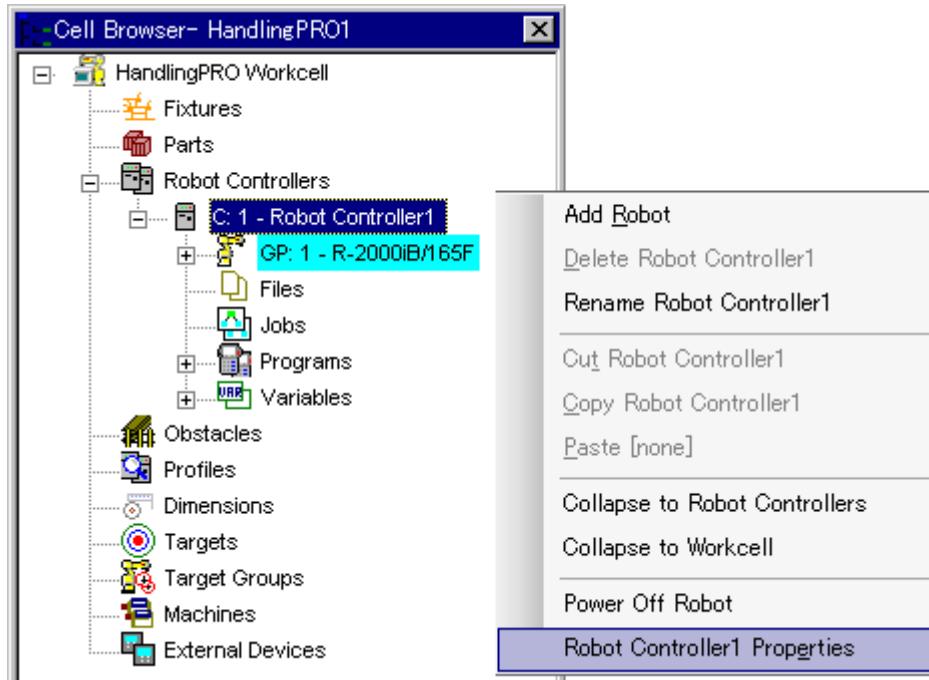
To use FANUC LADDER-III for Robot Online function with ROBOGUIDE by Ethernet communication, you don't have to set the Robot IP address to the Robot controller on ROBOGUIDE. The Robot IP address is the same as the PC that ROBOGUIDE is running on.

Obtain the Port Number

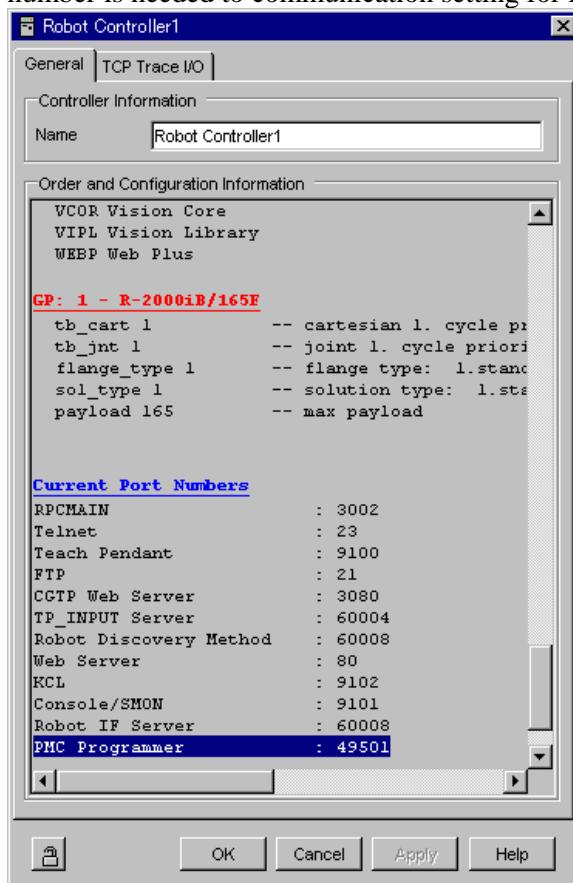
Multiple robots can be run dynamically on ROBOGUIDE. FANUC LADDER-III for Robot can connect with each robot by Ethernet. At this time, the IP addresses of these robots are same as the PC that ROBOGUIDE is running on. So port numbers are needed to connect ROBOGUIDE with FANUC LADDER-III for Robot besides IP address. Setting port number to FANUC LADDER-III for Robot is described in [Ethernet connection] later. Followings are how to get the port number for setting.

To obtain the port number, perform the following procedures on ROBOGUIDE.

- 1 Open the ROBOGUIDE Cell Browser, and click the right mouse button on “Robot Controller*” to open the robot controller Properties



- 2 The number of “PMC Programmer” in Current Port Numbers is the port number of the robot. This number is needed to communication setting for FANUC LADDER-III for Robot.



Setting of Serial Port

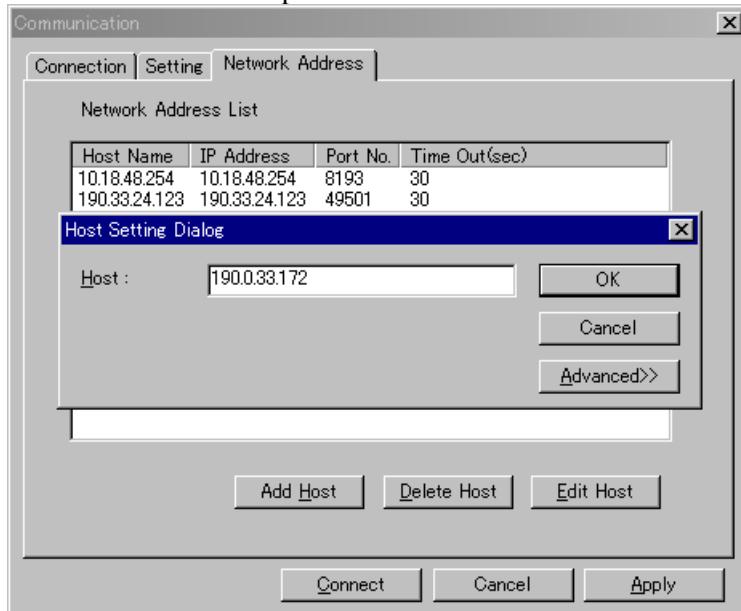
If the "PMC programmer" is set as a serial port, remove PMC programmer from all ports on the Teach pendant before using FANUC LADDER-III for Robot online.

For the way to set the serial port, refer to “Setting of Serial Port” in “5.1.2 Ethernet Connection”.

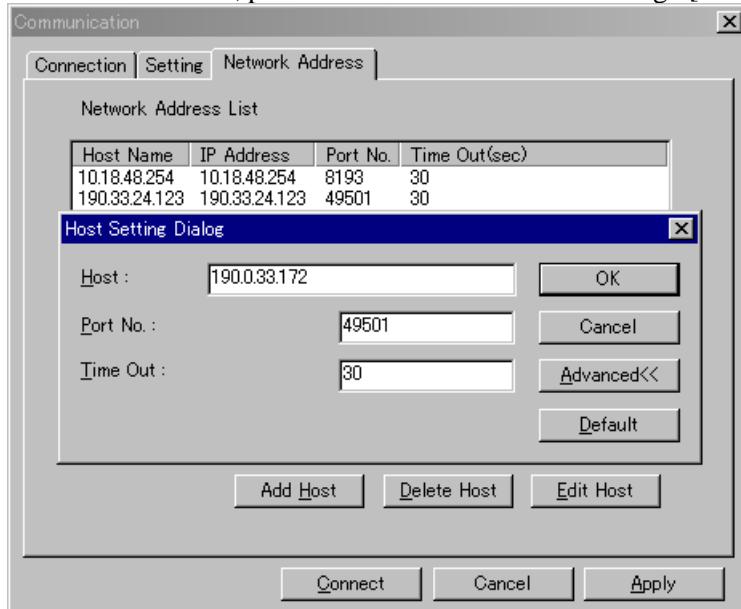
Ethernet connection

When setting of Ethernet connection and serial port are completed, Robot PMC can communicate with FANUC LADDER-III for Robot by Ethernet.

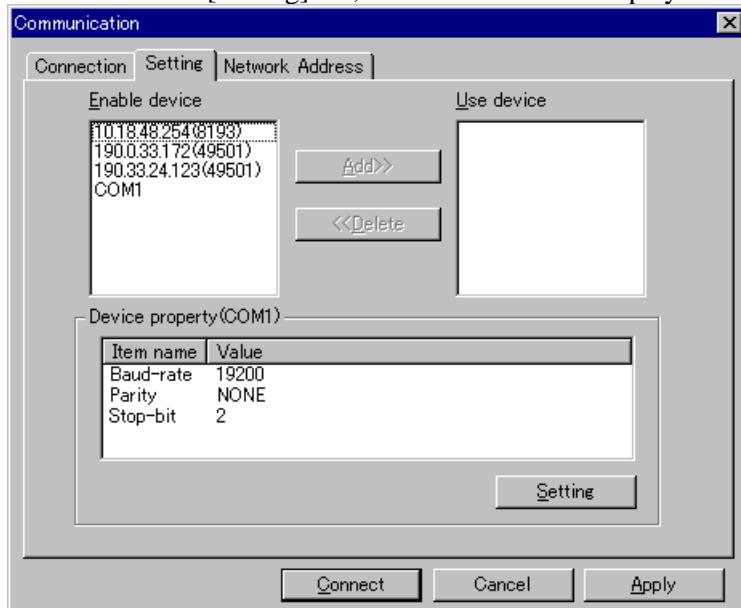
- 1 Execute the FANUC LADDER-III for Robot and select [Tool]-[Communication]. When click [Network Address] tab, Enable devices are displayed
Click <Add Host> and put the Robot IP address.



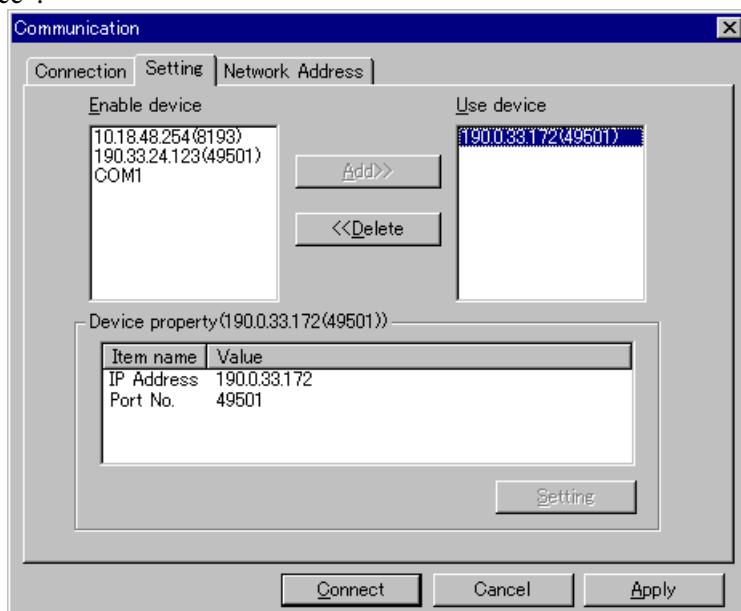
- 2 Click <Advanced>, put the Port number obtained through [Obtain the Port Number], and click <OK>.



- 3 Next when click [Setting] tab, Enable devices are displayed.



- 4 Select the Robot IP address from Enable devices, and click <Add> to add the IP address to “Use device”.

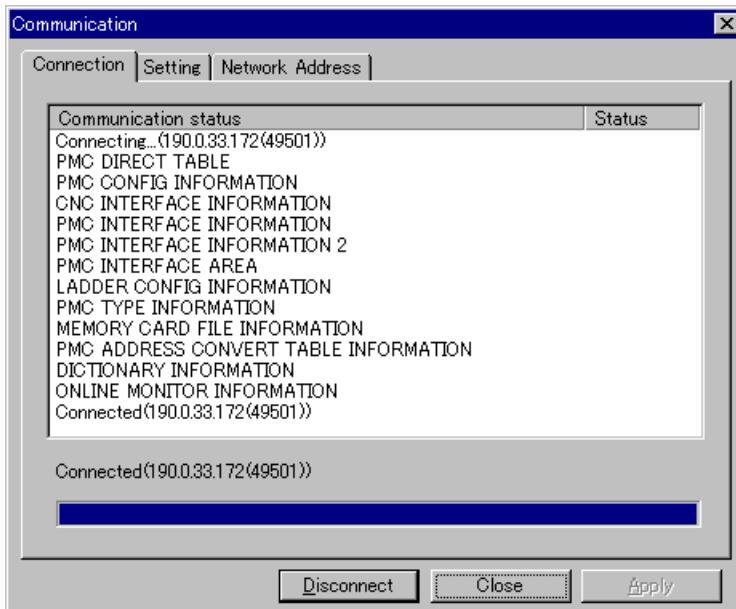


- 5 Click <Connect> to communicate with the Robot PMC by Ethernet.

- 6 If “Multi-Path PMC (J763)” or “DCS Safety PMC (J764)” is ordered, PMC type selection screen is displayed. So select PMC type to communicate and click <OK>. (If not ordered Multi-Path PMC (J763) and DCS Safety PMC (J764), skip this process.)



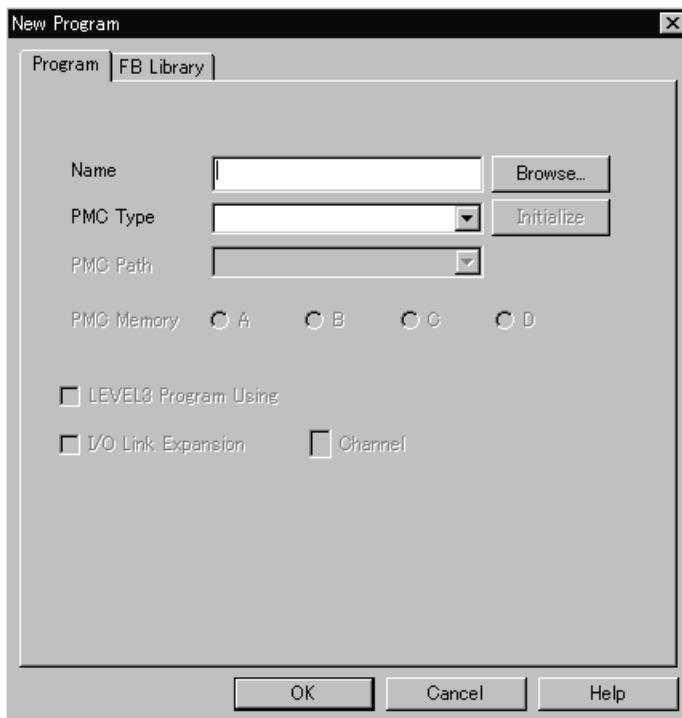
- 7 When the connection is completed, “Connected (IP address)” is displayed on the screen.
Click <Close>.



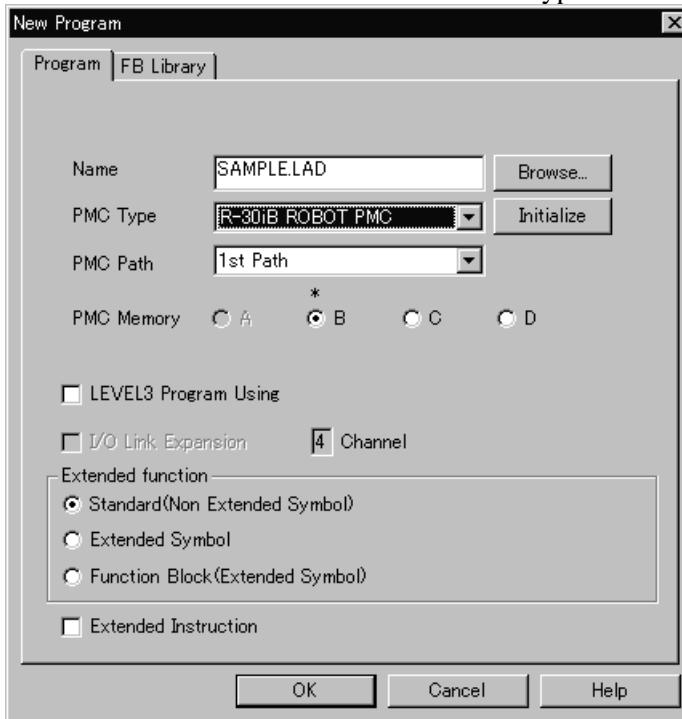
5.2 CREATING PMC PROGRAM

In FANUC LADDER-III for Robot, the PMC program is stored as a .LAD file.
To create a new PMC program, perform the following procedure below.

- From FANUC LADDER-III for Robot screen, select [File]-[New Program]
The [New Program] screen appears.



- 2 Enter the name of a program file you want to create. Specify "LAD" in the extension, and the extension cab be omitted.
- 3 Select "R-30iB ROBOT PMC" at the "PMC type".



- 4 Select the number of PMC Path at the "PMC path". To create DCS Safety PMC program, select "DCS".
- 5 Select PMC memory type at the "PMC Memory"
- 6 If use ladder program of level 3, check the "LEVEL3 Program Symbol". (Do not check it for DCS Safety PMC.)
- 7 Check the "Extended Symbol" box to use the extended symbol function.
The extended symbol/comment function allows to use the following features.

- Ladder edit by using symbol
 - Symbol effective in each sub-program
 - Symbol type setting
 - Automatic assignment function for R,D,E address
- 8 Check “Function Block (Extended Symbol)” box to use the Function Block function. (Do not check it for DCS Safety PMC.)
The Function Block function allows to use the following features.
- Creating/Editing/Printing of function blocks
 - Calling function blocks
 - Creating function block library
- 9 Check the “Extended Instruction” box to use the Extended PMC Ladder Instruction Function. (Do not check it for DCS Safety PMC.)
The Extended PMC Ladder Instruction Function allows to use the following features.
- Continuous operation by cascading functional instructions within a net
 - Positive and Negative transition contact
 - Functional instructions are added
 - Logical operations after branch of circuit
- 10 Click <OK>, then the program is created and program list is displayed.

NOTE

To use the multi path PMC, the option “Multi-Path PMC(J763)” is needed, in addition to the option “Integrated PMC(J760)”.

To use the DCS Safety PMC, the option “DCS Safety PMC(J764)” is needed, in addition to the option “Integrated PMC(J760)”.

5.3 EDITTING A PMC PROGRAM

PMC program editor is off-line function of FANUC LADDER-III for Robot.

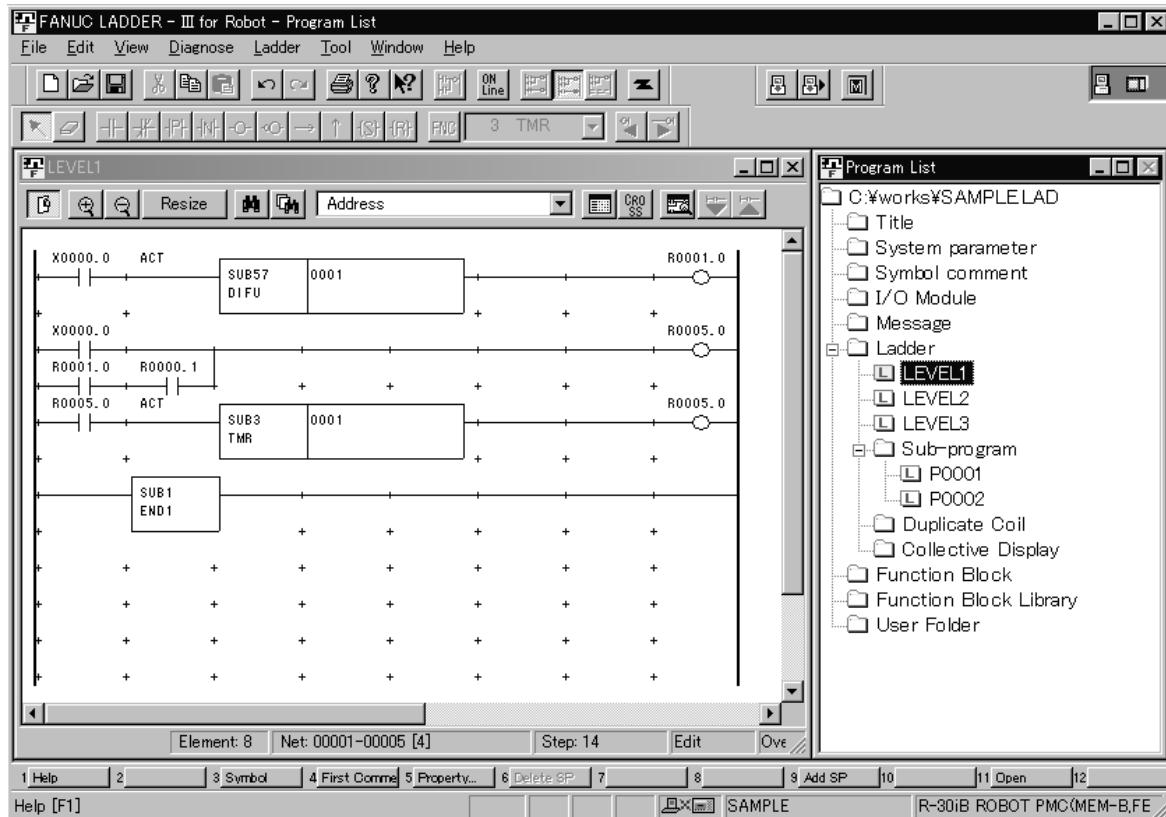
Refer to "Help" on toolbar with FANUC LADDRE-III for Robot to learn the basic FANUC LADDER-III for Robot operation.

NOTE

The robot controller Integrated PMC does not support the On-Line edit function.
PMC programs must be edited with the Off-Line Function. However, the robot controller Integrated PMC does support PMC program transfer during PMC program execution.

Procedure

- 1 In the [Program List] screen, double click LEVEL1, LEVEL2, or subprograms, such as P0001. Edit screen of selected program appears.



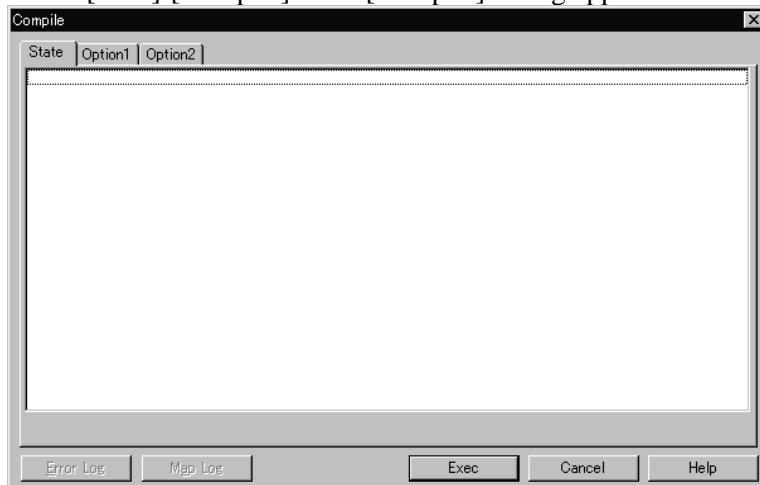
- 2 Edit ladder diagram.
- 3 Select [File]-[Save] to save program.

5.4 COMPIILING A PMC PROGRAM

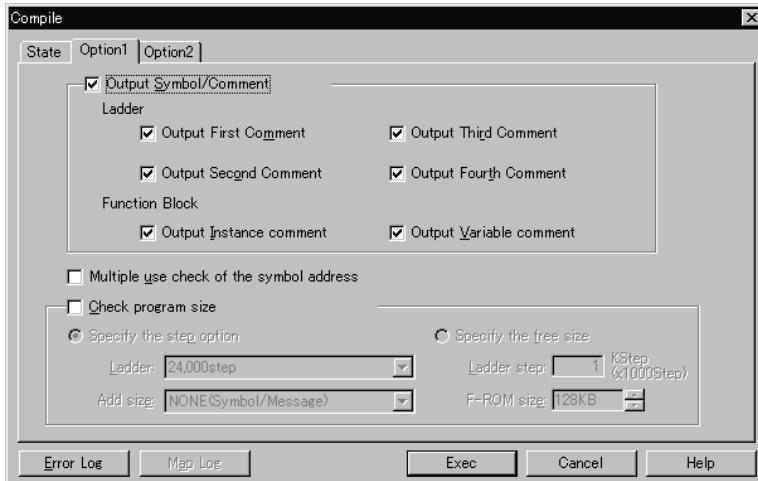
You must compile a PMC program to execute it on the robot controller.

Procedure

- 1 Select [Tool]-[Compile]. The [Compile] dialog appears.



- 2 Click [Option] tab to set the compile options.
(Refer to [Help]-[Tool]-[Compile] on toolbar with FANUC LADDER-III for Robot to learn the compiling a PMC program.)



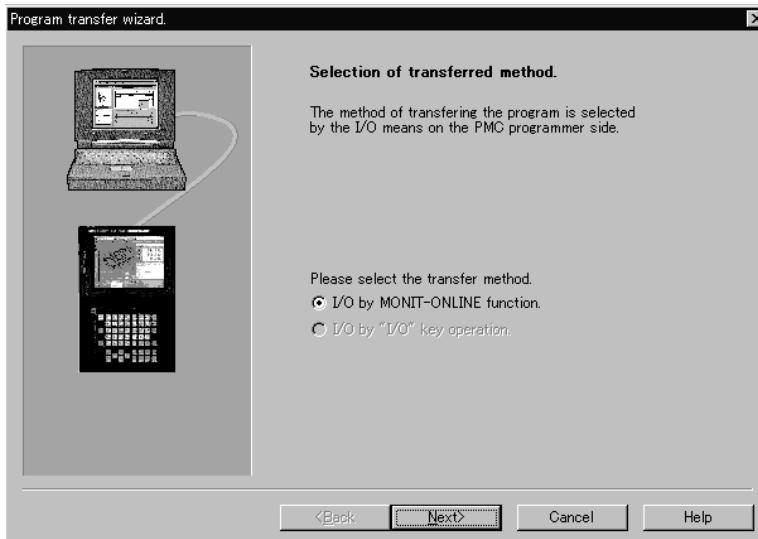
- 3 To start compilation, click the <Exec> button.
- 4 During compilation, the processing state screen appears. When completed, the number of errors and warnings appears.
- 5 Click <Close> button and close the window after compile finished.

5.5 TRANSFERRING PMC PROGRAM

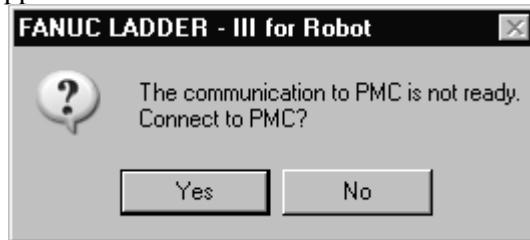
After the PMC program is edited and compiled by the On-Line function of FANUC LADDER-III for Robot, there are two ways to send the PMC program to the robot controller. One way is the On-Line Function of FANUC- LADDER-III for Robot, and the other is by the LADDER*.PMC file. This chapter explains the On-Line Function transfer of the PMC program. The way to save LADDER*.PMC is explained in Section 5.10.1, "Exporting LADDER*.PMC using FANUC LADDER-III for Robot."

Procedure

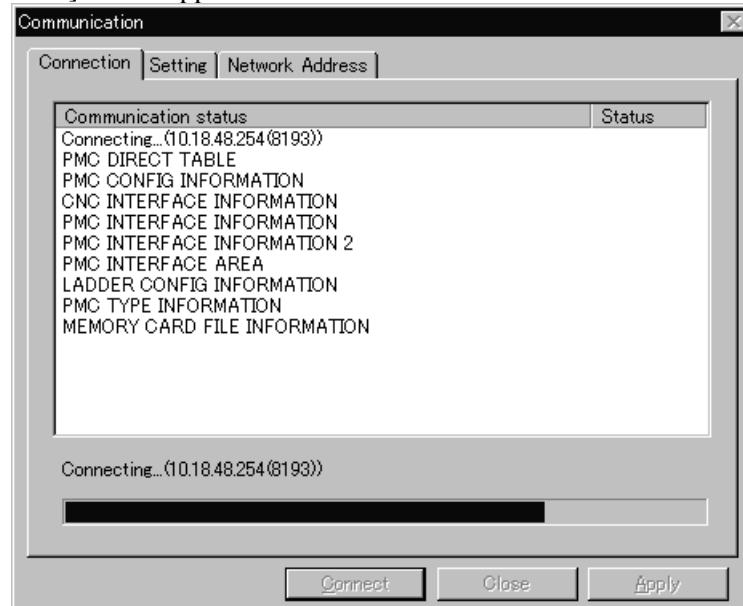
- 1 Connect ROBOT CONTROLLER and FANUC LADDER-III for Robot by cable, according to "5.1 CONNECTING FANUC LADDER-III for Robot".
- 2 Select [Tool]-[Store to PMC].
- 3 Select <I/O by MONIT-ONLINE function>, and then click the <Next> button. (Skip to process (7), if ROBOT CONTROLLER and FANUC LADDER-III for Robot are on communication.)



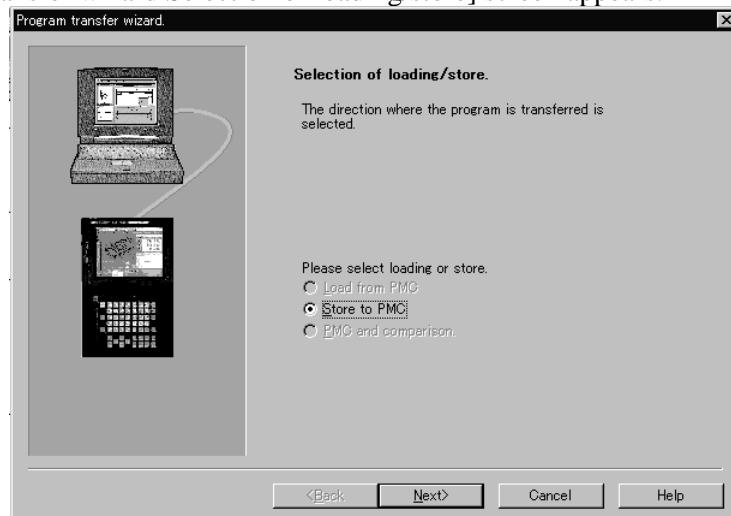
- 4 The following message appears. Click <Yes> button.



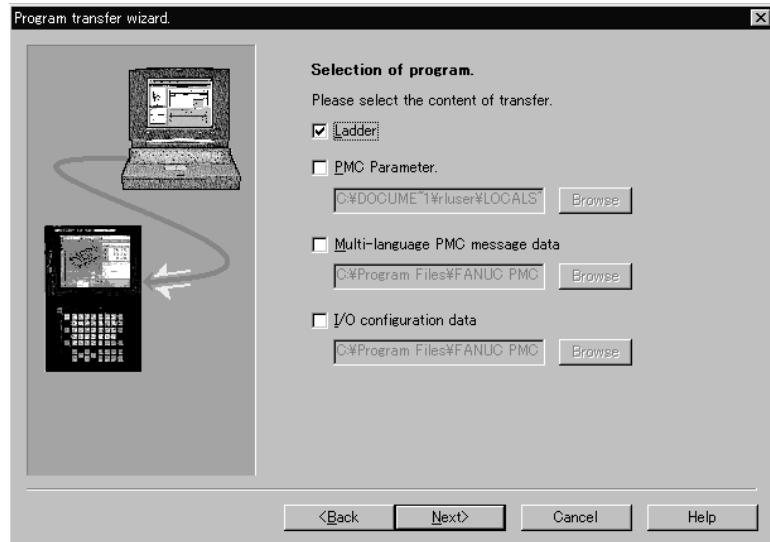
- 5 The [Communication] screen appears. Then communication with ROBOT CONTROLLER is started.



- 6 The [Program transfer wizard Selection of loading/store] screen appears.



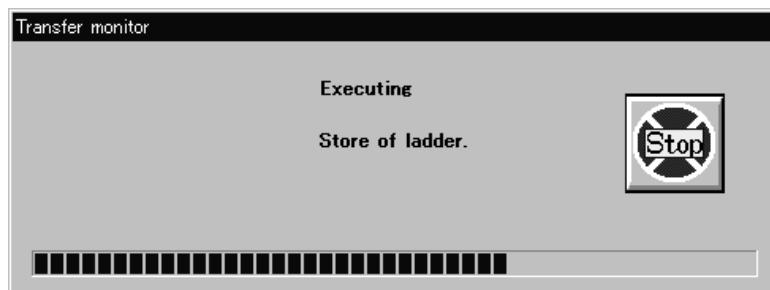
- 7 Click the <Next> button. The [Program transfer wizard Selection of program] screen appears.



- 8 Click the <Next> button. The [Program transfer wizard Configuration of processing] screen appears.



- 9 Click the <Finish> button. Then, the [Transfer monitor] screen appears, and then the program is transferred.



- 10 After transferring to RAM is finished, the program is backup to ROM automatically.

NOTE

For 7DC1 and 7DD0 software, RAM program is not written to ROM program automatically. When controller power is cycled, the original PMC program is loaded back into RAM memory, thus overwriting the PMC program that was recently transferred. To keep the new PMC program after power is cycled, the program must be transferred to ROM memory. To write the PMC program to ROM, refer to Section 5.8, "WRITING PMC PROGRAM TO F-ROM".

- 11 After the process finished, the screen was closed.

NOTE

You can store only "Ladder" program. Store of "PMC parameter", "Multi-language PMC message data" and "I/O configuration data" are not supported.

If a PMC program is running during this procedure, the PMC program continues to run during transfer. The executing PMC program is switched to the new program when the transfer is complete. With this procedure, you can send a PMC program without stopping execution.

NOTE

When a PMC program is transferred during execution, the value of internal relays, internal variables, and so forth are not initialized. This might cause unexpected output. For example, DIFU turns on when the PMC program is switched. Be careful when making large changes to the PMC program.

NOTE

When a DCS Safety PMC program is transferred, "SYST-212 Need to apply to DCS param" occurs. Please apply to DCS parameter in DCS screen, then the new PMC program is executed.

Please refer to "FANUC Robot Series R-30iB/ R-30iB Mate Controller Dual Check Safety Operator's Manual (B-83184EN)" for apply procedure.

When transferring a program, if an error occurs, the following dialog box will be displayed on FANUC LADDER-III for Robot.

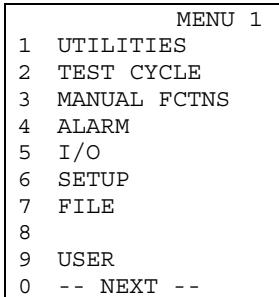


In this case PMC is still executing the previous PMC program. An error message is also displayed on the robot controller teach pendant.

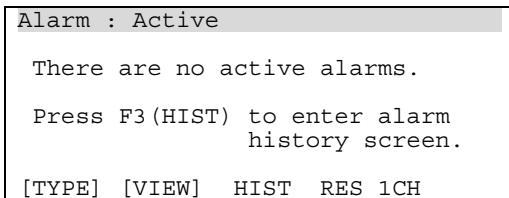
The teach pendant displays the last error message. To read all error messages, display the alarm history screen with the following procedure.

Procedure to check errors caused by PMC programs

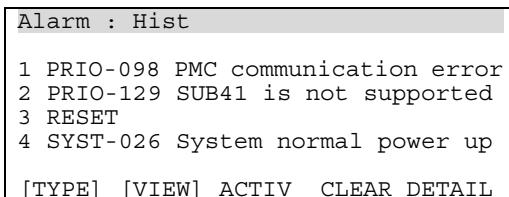
- 1 Press MENU key.



- 2 Select "4. ALARM". The Active Alarm screen will be displayed.



- 3 Press F3 in Active Alarm screen. The History Alarm screen will be displayed.



NOTE

The PMC program error is a 'warning', and it is not displayed in the active alarm screen. Please check the Alarm History screen to see this error displayed.

In the Alarm History screen, the errors are listed in the order they occurred. The last alarm is on the top line. Refer to "Alarm Code List OPERATOR'S MANUAL" for details of all PMC errors. When an alarm occurs during program transfer, the new PMC program can't run. If a previous PMC program is running during PMC program transfer, the previous PMC program continues to run. The new PMC program cannot be written to ROM until it is correctly transferred.

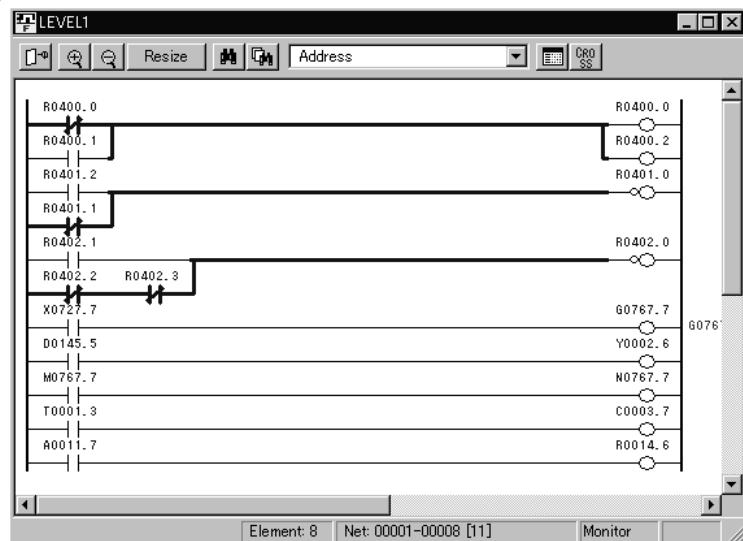
If a PMC program has more than 10 errors, the 11th and higher alarms are not displayed in the alarm history screen and "PRIO-140 can't display all PMC errors" is displayed. The first ten alarms must be corrected in order to see the remaining alarms.

5.6 PMC PROGRAM MONITORING (ONLINE MONITOR)

LADDER MONITORING is online monitor function for the PMC program.

Refer to [Help]-[Diagnosis]-[Ladder Monitoring] on FANUC LADDER-III for Robot toolbar for detailed information.

LADDER MONITOR



Signal Status Monitor

	Address	b7	b6	b5	b4	b3	b2	b1	b0	
		Sym	BDL	Byte	Wdg	D.Wdg	Bit	Dec	Hex	Bcd
Signal Status	R0000	0	0	0	0	0	0	0	0	0
A	R0001	0	0	0	0	0	1	1	1	1
R	R0002	0	0	0	0	0	0	0	0	0
R0	R0003	1	0	0	0	0	1	1	1	1
R9000	R0004	0	1	0	0	0	1	1	1	1
T	R0005	1	0	0	0	1	0	0	0	0
K	R0006	0	0	0	0	0	1	0	1	1
C	R0007	0	0	0	0	0	0	0	0	0
D0	R0008	0	0	0	0	0	0	0	0	0
S1	R0009	0	1	0	1	0	1	0	1	1
E0	R0010	0	0	0	0	0	0	0	0	0
Y	R0011	0	0	0	0	0	0	0	0	0
X	R0012	0	0	0	0	0	0	0	0	0
Synchronous input memory	R0013	0	0	0	0	0	0	0	0	0
N0	R0014	0	0	0	0	0	0	0	0	0
M0	R0015	0	0	0	0	0	0	0	0	0
G	R0016	0	0	0	0	0	0	0	0	0
F	R0017	0	0	0	0	0	0	0	0	0
	R0018	0	0	0	0	0	0	0	0	0
	R0019	0	0	0	0	0	0	0	0	0
	R0020	0	0	0	0	0	0	0	0	0
	R0021	0	0	0	0	0	0	0	0	0

Function Instruction display Format

The data types of the displayed parameters are shown in the following table.

No.	Name	Parameter	Monitor format
1	END1	—	—
2	END2	—	—
3	TMR	1	special
4	DEC	1	2-digit BCD
		2	constant
5	CTR	1	special
6	ROT	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
7	COD *	1	constant
		2	2-digit BCD
		3	4-digit BCD
8	MOVE	1	constant
		2	constant
		3	2-digit HEX
		4	2-digit HEX
9	COM	1	constant
10	JMP	1	constant
11	PARI	1	1-byte binary
14	DCNV	1	no monitor
		2	no monitor
15	COMP	1	constant
		2	4-digit BCD
		3	4-digit BCD
16	COIN	1	constant
		2	4-digit BCD
		3	4-digit BCD
17	DSCH	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD

No.	Name	Parameter	Monitor format
18	XMOV	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
19	ADD	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
20	SUB	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
21	MUL	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
22	DIV	1	constant
		2	4-digit BCD
		3	4-digit BCD
		4	4-digit BCD
23	NUME	1	constant
		2	4-digit BCD
24	TMRB	1	special
		2	constant
25	DECB	1	constant
		2	variable binary
		3	constant
		4	2-digit HEX
26	ROTB	1	constant
		2	variable binary
		3	variable binary
		4	variable binary
		5	variable binary

No.	Name	Parameter	Monitor format
27	CODB *	1	constant
		2	constant
		3	1-byte binary
		4	variable binary
28	MOVOR	1	2-digit HEX
		2	2-digit HEX
		3	2-digit HEX
29	COME	—	—
30	JMPE	—	—
31	DCNVB	1	constant
		2	no monitor
		3	no monitor
32	COMPB	1	constant
		2	constant or variable binary
		3	variable binary
33	SFT	1	4-digit HEX
34	DSCHB	1	constant
		2	variable binary
		3	variable binary
		4	variable binary
		5	variable binary
35	XMOVB	1	constant
		2	variable binary
		3	variable binary
		4	variable binary
		5	variable binary
36	ADDB	1	constant
		2	variable binary
		3	constant or variable binary
		4	variable binary
37	SUBB	1	constant
		2	variable binary
		3	constant or variable binary
		4	variable binary
		5	variable binary
38	MULB	1	constant
		2	variable binary
		3	constant or variable binary
		4	variable binary

No.	Name	Parameter	Monitor format
39	DIVB	1	constant
		2	variable binary
		3	constant or variable binary
		4	variable binary
40	NUMEB	1	constant
		2	constant
		3	variable binary
43	MOVB	1	1-byte binary
		2	1-byte binary
44	MOVW	1	2-byte binary
		2	2-byte binary
45	MOVN	1	constant
		2	4-byte binary
		3	4-byte binary
47	MOVD	1	4-byte binary
		2	4-byte binary
48	END3	—	—
54	TMRC	1	constant
		2	special
		3	special
55	CTRC	1	2-byte binary
56	CTRIB	1	constant
		2	special
57	DIFU	1	constant
58	DIFD	1	constant
59	EOR	1	constant
		2	variable HEX
		3	constant or variable HEX
		4	variable HEX

No.	Name	Parameter	Monitor format
60	AND	1	constant
		2	variable HEX
		3	constant or variable HEX
		4	variable HEX
61	OR	1	constant
		2	variable HEX
		3	constant or variable HEX
		4	variable HEX
62	NOT	1	constant
		2	variable HEX
		3	variable HEX
64	END	-	-
65	CALL	1	no monitor
66	CALLU	1	no monitor
68	JMPB	1	no monitor
69	LBL	1	no monitor
70	NOP	1	constant
71	SP	1	no monitor
72	SPE	-	-
73	JMPC	1	no monitor
74	CS	1	2-byte binary
75	CM	1	no monitor
76	CE	-	-
77	TMRBF	1	special
		2	constant
200	EQB	1	constant or 1-byte binary
		2	constant or 1-byte binary
		-	-
201	EQW	1	constant or 2-byte binary
		2	constant or 2-byte binary
202	EQD	1	constant or 4-byte binary
		2	constant or 4-byte binary

No.	Name	Parameter	Monitor format
203	NEB	1	constant or 1-byte binary
		2	constant or 1-byte binary
204	NEW	1	constant or 2-byte binary
		2	constant or 2-byte binary
205	NED	1	constant or 4-byte binary
		2	constant or 4-byte binary
206	GTB	1	constant or 1-byte binary
		2	constant or 1-byte binary
207	GTW	1	constant or 2-byte binary
		2	constant or 2-byte binary
208	GTD	1	constant or 4-byte binary
		2	constant or 4-byte binary
209	LTB	1	constant or 1-byte binary
		2	constant or 1-byte binary
210	LTW	1	constant or 2-byte binary
		2	constant or 2-byte binary
211	LTD	1	constant or 4-byte binary
		2	constant or 4-byte binary
212	GEB	1	constant or 1-byte binary
		2	constant or 1-byte binary

No.	Name	Parameter	Monitor format
213	GEW	1	constant or 2-byte binary
		2	constant or 2-byte binary
214	GED	1	constant or 4-byte binary
		2	constant or 4-byte binary
215	LEB	1	constant or 1-byte binary
		2	constant or 1-byte binary
216	LEW	1	constant or 2-byte binary
		2	constant or 2-byte binary
217	LED	1	constant or 4-byte binary
		2	constant or 4-byte binary
218	RNGB	1	constant or 1-byte binary
		2	constant or 1-byte binary
		3	constant or 1-byte binary
219	RNGW	1	constant or 2-byte binary
		2	constant or 2-byte binary
		3	constant or 2-byte binary
220	RNGD	1	constant or 4-byte binary
		2	constant or 4-byte binary
		3	constant or 4-byte binary

No.	Name	Parameter	Monitor format
221	TMRST	1	Constant or Special
		2	Special
		3	No monitor
222	TMRSS	1	Constant or Special
		2	Special
		3	No monitor
223	CTRД	1	4-byte binary
		2	4-byte binary
224	MOVBT	1	No monitor
		2	4-byte HEX
		3	Constant
		4	4-byte HEX
		5	Constant
225	SETNB	1	No monitor
		2	Constant or 1-byte binary
		3	1-byte binary
226	SETNW	1	No monitor
		2	Constant or 2-byte binary
		3	2-byte binary
227	SETND	1	No monitor
		2	Constant or 4-byte binary
		3	4-byte binary
228	XCHGB	1	1-byte binary
		2	1-byte binary
229	XCHGW	1	2-byte binary
		2	2-byte binary
230	XCHGD	1	4-byte binary
		2	4-byte binary
231	SWAPW	1	No monitor
		2	2-byte binary
		3	2-byte binary
232	SWAPD	1	No monitor
		2	4-byte binary
		3	4-byte binary

No.	Name	Parameter	Monitor format
233	TBLRB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	1-byte binary
234	TBLRW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	2-byte binary
235	TBLRD	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	4-byte binary
236	TBLRN	1	No monitor
		2	No monitor
		3	4-byte binary
		4	Constant or 2-byte binary
		5	4-byte binary
237	TBLWB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	1-byte binary
238	TBLWW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	2-byte binary
239	TBLWD	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	4-byte binary
240	TBLWN	1	No monitor
		2	No monitor
		3	4-byte binary
		4	Constant or 2-byte binary
		5	4-byte binary

No.	Name	Parameter	Monitor format
241	DSEQB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
242	DSEQW	5	2-byte binary
		1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
243	DSEQD	5	2-byte binary
		1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary
244	DSNEB	5	2-byte binary
		1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
245	DSNEW	5	2-byte binary
		1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
246	DSNED	5	2-byte binary
		1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary

No.	Name	Parameter	Monitor format
247	DSGTB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
		5	2-byte binary
248	DSGTW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
		5	2-byte binary
249	DSGTD	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary
		5	2-byte binary
250	DSLTB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
		5	2-byte binary
251	DSLTW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
		5	2-byte binary
252	DSLTD	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary
		5	2-byte binary

No.	Name	Parameter	Monitor format
253	DSGEB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
		5	2-byte binary
254	DSGEW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
		5	2-byte binary
255	DSGED	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary
		5	2-byte binary
256	DSLEB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	Constant or 1-byte binary
		5	2-byte binary
257	DSLEW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	Constant or 2-byte binary
		5	2-byte binary
258	DSLED	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	Constant or 4-byte binary
		5	2-byte binary

No.	Name	Parameter	Monitor format
259	DMAXB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	1-byte binary
		5	2-byte binary
260	DMAXW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	2-byte binary
		5	2-byte binary
261	DMAXD	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	4-byte binary
		5	2-byte binary
262	DMINB	1	No monitor
		2	1-byte binary
		3	Constant or 2-byte binary
		4	1-byte binary
		5	2-byte binary
263	DMINW	1	No monitor
		2	2-byte binary
		3	Constant or 2-byte binary
		4	2-byte binary
		5	2-byte binary
264	DMIND	1	No monitor
		2	4-byte binary
		3	Constant or 2-byte binary
		4	4-byte binary
		5	2-byte binary

No.	Name	Parameter	Monitor format
265	EORB	1	Constant or 1-byte HEX
		2	Constant or 1-byte HEX
		3	1-byte HEX
266	EORW	1	Constant or 2-byte HEX
		2	Constant or 2-byte HEX
		3	2-byte HEX
267	EORD	1	Constant or 4-byte HEX
		2	Constant or 4-byte HEX
		3	4-byte HEX
268	ANDB	1	Constant or 1-byte HEX
		2	Constant or 1-byte HEX
		3	1-byte HEX
269	ANDW	1	Constant or 2-byte HEX
		2	Constant or 2-byte HEX
		3	2-byte HEX
270	ANDD	1	Constant or 4-byte HEX
		2	Constant or 4-byte HEX
		3	4-byte HEX
271	ORB	1	Constant or 1-byte HEX
		2	Constant or 1-byte HEX
		3	1-byte HEX
272	ORW	1	Constant or 2-byte HEX
		2	Constant or 2-byte HEX
		3	2-byte HEX

No.	Name	Parameter	Monitor format
273	ORD	1	Constant or 4-byte HEX
		2	Constant or 4-byte HEX
		3	4-byte HEX
274	NOTB	1	Constant or 1-byte HEX
		2	1-byte HEX
275	NOTW	1	Constant or 2-byte HEX
		2	2-byte HEX
276	NOTD	1	Constant or 4-byte HEX
		2	4-byte HEX
277	SHLB	1	Constant or 1-byte HEX
		2	Constant or 1-byte binary
		3	1-byte HEX
278	SHLW	1	Constant or 2-byte HEX
		2	Constant or 2-byte binary
		3	2-byte HEX
279	SHLD	1	Constant or 4-byte HEX
		2	Constant or 4-byte binary
		3	4-byte HEX
280	SHLN	1	No monitor
		2	Constant or 4-byteS HEX
		3	Constant or 4-byte binary
		4	4-byte HEX
281	SHRB	1	Constant or 1-byte HEX
		2	Constant or 1-byte binary
		3	1-byte HEX

No.	Name	Parameter	Monitor format
282	SHRW	1	Constant or 2-byte HEX
		2	Constant or 2-byte binary
		3	2-byte HEX
283	SHRD	1	Constant or 4-byte HEX
		2	Constant or 4-byte binary
		3	4-byte HEX
284	SHRN	1	No monitor
		2	Constant or 4-byte HEX
		3	Constant or 4-byte binary
		4	4-byte HEX
285	ROLB	1	Constant or 1-byte HEX
		2	Constant or 1-byte binary
		3	1-byte HEX
286	ROLW	1	Constant or 2-byte HEX
		2	Constant or 2-byte binary
		3	2-byte HEX
287	ROLD	1	Constant or 4-byte HEX
		2	Constant or 4-byte binary
		3	4-byte HEX
288	ROLN	1	No monitor
		2	Constant or 4-byte HEX
		3	Constant or 4-byte binary
		4	4-byte HEX
289	RORB	1	Constant or 1-byte HEX
		2	Constant or 1-byte binary
		3	1-byte HEX

No.	Name	Parameter	Monitor format
290	RORW	1	Constant or 2-byte HEX
		2	Constant or 2-byte binary
		3	2-byte HEX
291	RORD	1	Constant or 4-byte HEX
		2	Constant or 4-byte binary
		3	4-byte HEX
292	RORN	1	No monitor
		2	Constant or 4-byte HEX
		3	Constant or 4-byte binary
		4	4-byte HEX
293	BSETB	1	1-byte HEX
		2	Constant or 1-byte binary
294	BSETW	1	2-byte HEX
		2	Constant or 2-byte binary
295	BSETD	1	4-byte HEX
		2	Constant or 4-byte binary
296	BSETN	1	No monitor
		2	4-byte HEX
		3	Constant or 4-byte binary
297	BRSTB	1	1-byte HEX
		2	Constant or 1-byte binary
298	BRSTW	1	2-byte HEX
		2	Constant or 2-byte binary
299	BRSTD	1	4-byte HEX
		2	Constant or 4-byte binary
300	BRSTN	1	No monitor
		2	4-byte HEX
		3	Constant or 4-byte binary

No.	Name	Parameter	Monitor format
301	BTSTB	1	1-byte HEX
		2	Constant or 1-byte binary
302	BTSTW	1	2-byte HEX
		2	Constant or 2-byte binary
303	BTSTD	1	4-byte HEX
		2	Constant or 4-byte binary
304	BTSTN	1	No monitor
		2	4-byte HEX
		3	Constant or 4-byte binary
305	BPOSB	1	1-byte HEX
		2	1-byte binary
306	BPOSW	1	2-byte HEX
		2	2-byte binary
307	BPOSD	1	4-byte HEX
		2	4-byte binary
308	BPOSN	1	No monitor
		2	4-byte HEX
		3	4-byte binary
309	BCNTB	1	1-byte HEX
		2	1-byte binary
310	BCNTW	1	2-byte HEX
		2	2-byte binary
311	BCNTD	1	4-byte HEX
		2	4-byte binary
312	BCNTN	1	No monitor
		2	4-byte HEX
		3	4-byte binary
313	TBCDB	1	Constant or 1-byte binary
		2	1-byte HEX
314	TBCDW	1	Constant or 2-byte binary
		2	2-byte HEX
315	TBCDD	1	Constant or 4-byte binary
		2	4-byte HEX

No.	Name	Parameter	Monitor format
316	FBCDB	1	Constant or 1-byte HEX
		2	1-byte binary
317	FBCDW	1	Constant or 2-byte HEX
		2	2-byte binary
318	FBCDD	1	Constant or 4-byte HEX
		2	4-byte binary
319	ADDSSB	1	Constant or 1-byte binary
		2	Constant or 1-byte binary
		3	1-byte binary
320	ADDSSW	1	Constant or 2-byte binary
		2	Constant or 2-byte binary
		3	2-byte binary
321	ADDSD	1	Constant or 4-byte binary
		2	Constant or 4-byte binary
		3	4-byte binary
322	SUBSB	1	Constant or 1-byte binary
		2	Constant or 1-byte binary
		3	1-byte binary
323	SUBSW	1	Constant or 2-byte binary
		2	Constant or 2-byte binary
		3	2-byte binary
324	SUBSD	1	Constant or 4-byte binary
		2	Constant or 4-byte binary
		3	4-byte binary

No.	Name	Parameter	Monitor format
325	MULSB	1	Constant or 1-byte binary
		2	Constant or 1-byte binary
		3	1-byte binary
326	MULSW	1	Constant or 2-byte binary
		2	Constant or 2-byte binary
		3	2-byte binary
327	MULSD	1	Constant or 4-byte binary
		2	Constant or 4-byte binary
		3	4-byte binary
328	DIVSB	1	Constant or 1-byte binary
		2	Constant or 1-byte binary
		3	1-byte binary
329	DIVSW	1	Constant or 2-byte binary
		2	Constant or 2-byte binary
		3	2-byte binary
330	DIVSD	1	Constant or 4-byte binary
		2	Constant or 4-byte binary
		3	4-byte binary
331	MODSB	1	Constant or 1-byte binary
		2	Constant or 1-byte binary
		3	1-byte binary
332	MODSW	1	Constant or 2-byte binary
		2	Constant or 2-byte binary
		3	2-byte binary

No.	Name	Parameter	Monitor format
333	MODSD	1	Constant or 4-byte binary
		2	Constant or 4-byte binary
		3	4-byte binary
334	INCSB	1	1-byte binary
335	INCSW	1	2-byte binary
336	INCSD	1	4-byte binary
337	DECSB	1	1-byte binary
338	DECSW	1	2-byte binary
339	DECSD	1	4-byte binary
340	ABSSB	1	Constant or 1-byte binary
		2	1-byte binary
341	ABSSW	1	Constant or 2-byte binary
		2	2-byte binary
342	ABSSD	1	Constant or 4-byte binary
		2	4-byte binary
343	NEGSB	1	Constant or 1-byte binary
		2	1-byte binary
344	NEGSW	1	Constant or 2-byte binary
		2	2-byte binary
345	NEGSD	1	Constant or 4-byte binary
		2	4-byte binary

NOTE

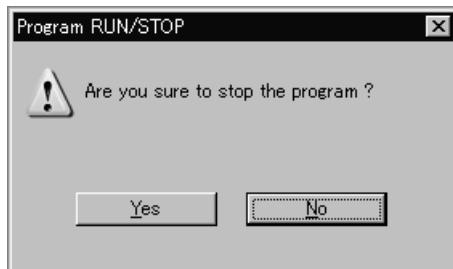
When data type is BCD, invalid values for BCD (10, 11, 12, 13, 14, 15) are not displayed correctly.

5.7 RUNNING OR STOPPING PMC PROGRAM

This section describes how to execute and stop PMC program from FANUC LADDER-III for Robot.

Procedure

- 1 Select [Tool]-[Program Run/Stop]. The message similar to the following appears.



- 2 Click <Yes> button.
The PMC program will stop.

If PMC program is not running, the message "Are you sure to run the program ?" appears. Follow the same procedure to run the program.

NOTE

Program RUN/STOP cannot be executed for DCS Safety PMC.

5.8 WRITING PMC PROGRAM TO F-ROM

NOTE

Following description is operation for 7DC1 and 7DD0 software.
For 7DC2 or later software, RAM program is written to ROM program automatically after "transferring PMC program". It is not needed to backup the PMC programs after transferring.

The PMC program is stored in ROM on the robot controller. It is loaded to RAM on the robot controller when the controller is turned on. The PMC program in RAM is executed.

When the PMC program is transferred by On-Line function of the FANUC LADDER-III for Robot, the PMC program in RAM is changed. But for 7DC1 and 7DD0 software, the PMC program in ROM is not changed. If you cycle power in this situation, the PMC program in ROM is reloaded to RAM, and is executed.

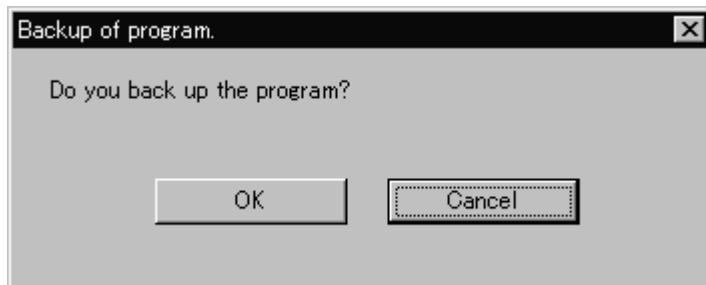
For 7DC1 and 7DD0 software, to change the PMC program in ROM, you need to backup the PMC program. This means writing the PMC program in RAM to ROM. By PMC program backup, the PMC program in ROM is changed, and a new PMC program is executed at the next power up.

NOTE

For 7DC1 and 7DD0 software, before backing up the PMC program, press the E-STOP on the teach pendant or Operator's panel, or the robot controller must be in Controlled start mode. If the robot controller is not in this condition, a dialog box with "N : E-3007 "Flash ROM Not EMG stop" error code" is displayed on FANUC LADDER-III for Robot, and "PRIO-142 Need E-STOP or CTRL start" is displayed on the robot controller teach pendant.

Procedure

- 1 Make sure the TP program on ROBOT CONTROLLER program is stopped.
- 2 Press EMERGENCY STOP on the operator panel or teach pendant. (When ROBOT CONTROLLER is at a Controlled Start, step 2 is not required.)
- 3 Select [Tool]-[Back Up] on FANUC LADDER-III for Robot. The following message appears.



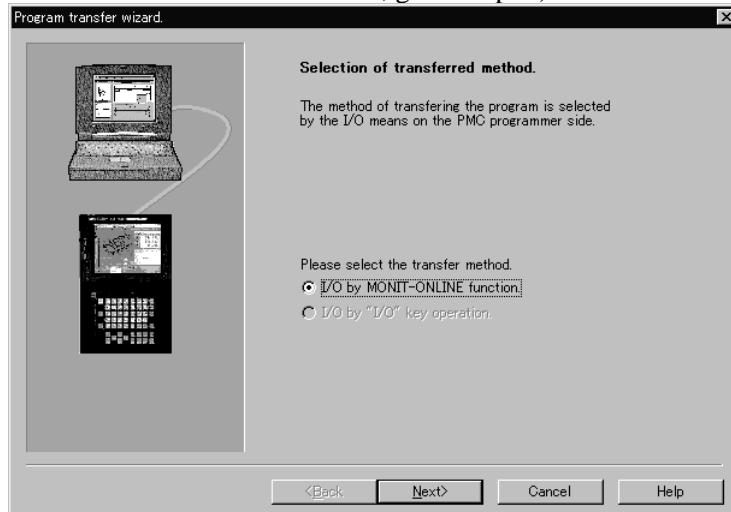
- 4 Click <OK>. Then Back Up starts. Don't turn off the ROBOT CONTROLLER during Back Up. The following message is displayed when it finishes.



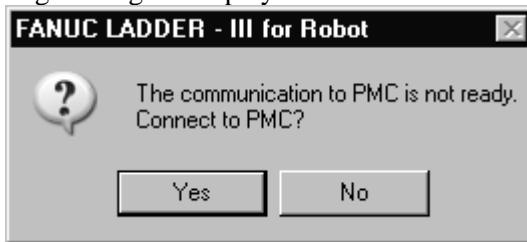
5.9 MODIFYING THE PMC PROGRAM IN THE ROBOT CONTROLLER

Use the following procedure to modify the PMC program running on the Robot Controller if you do not have a copy of the program on the FANUC LADDER-III for Robot.

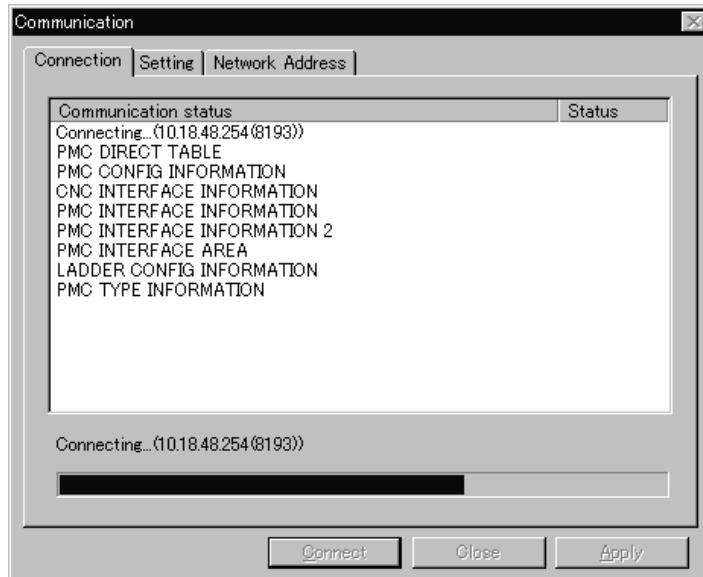
- 1 According to "4.1 CONNECTING FANUC LADDER-III for Robot", connect ROBOT CONTROLLER and FANUC LADDER-III for Robot.
- 2 According to "4.2 CREATING PMC", create new PMC program.
- 3 Select [Tool]-[Load from PMC] to open the program. (If FANUC LADDER-III for Robot and ROBOT CONTROLLER are on communication, go to step 8.)



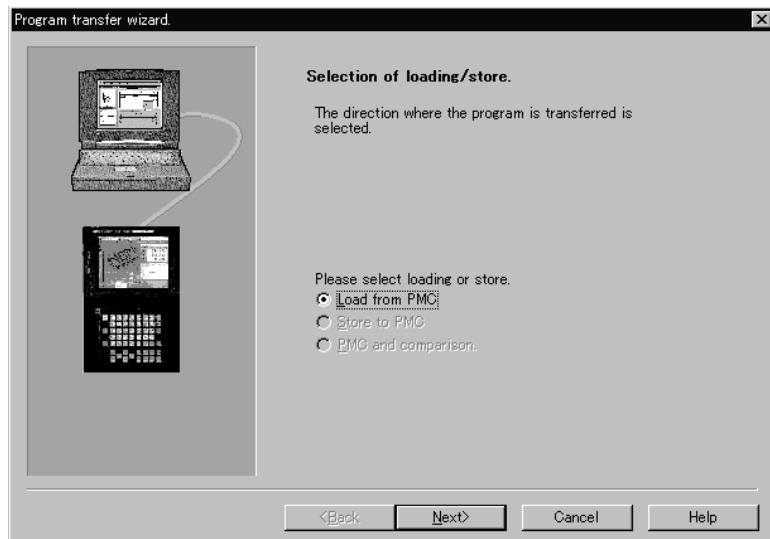
- 4 Click <Next>. The following message is displayed.



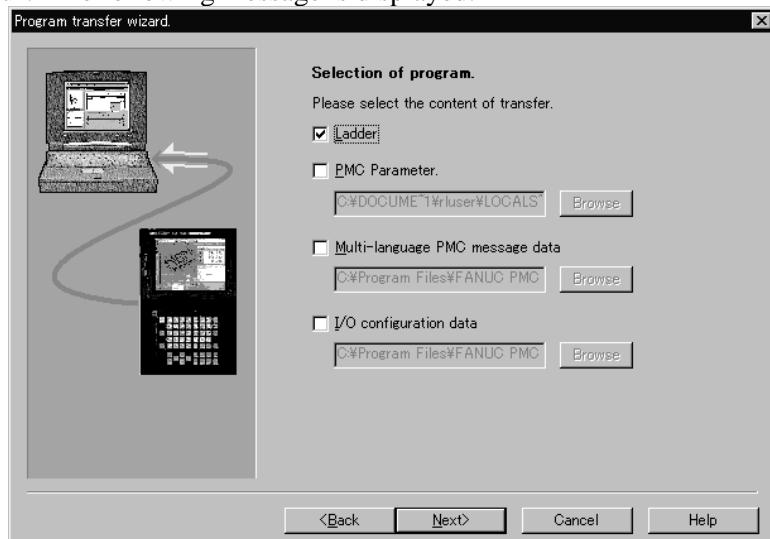
- 5 Click <Yes>.
- 6 The [Communication] screen appears and the communication with ROBOT CONTROLLER starts.



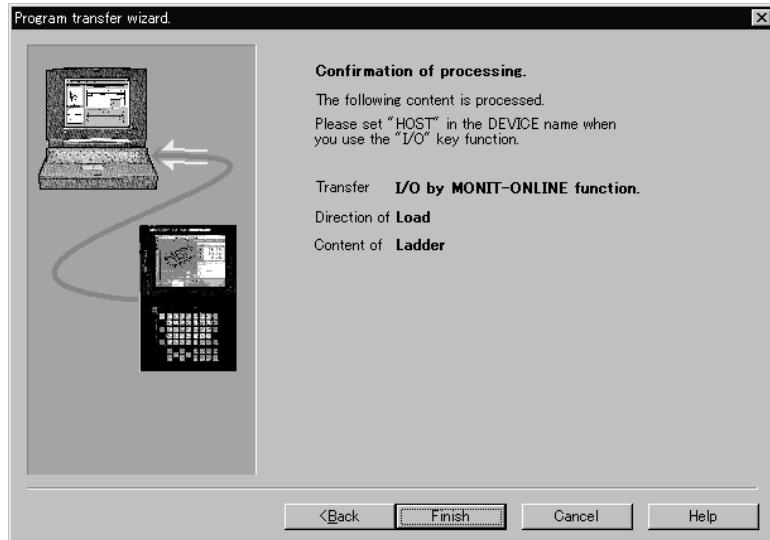
- 7 After communication is established, the [Program transfer wizard Selection of loading/store] screen appears.



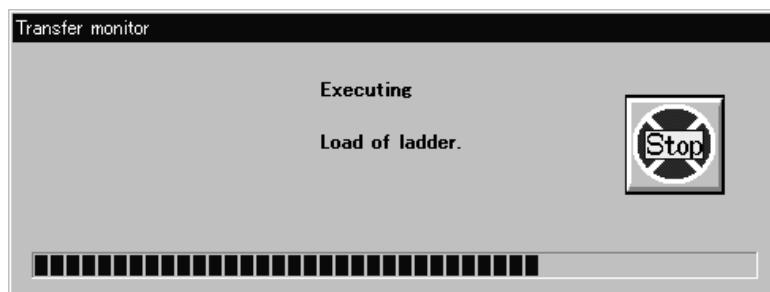
- 8 Click the <Next>. The following message is displayed.



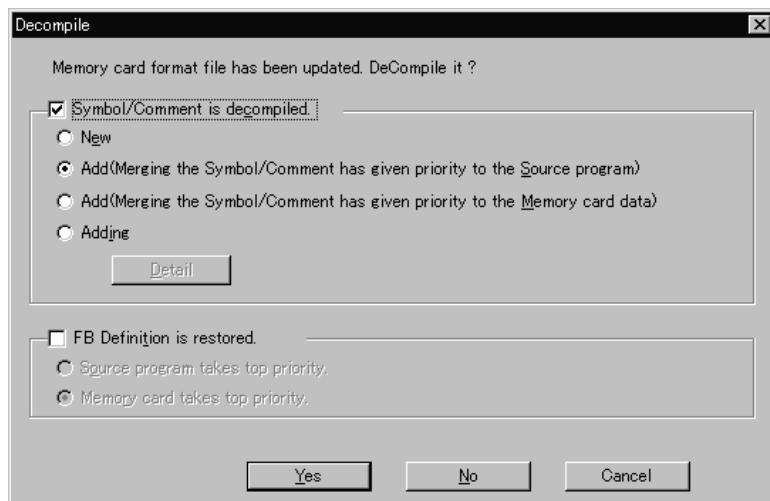
- 9 Click <Next>. The following message is displayed.



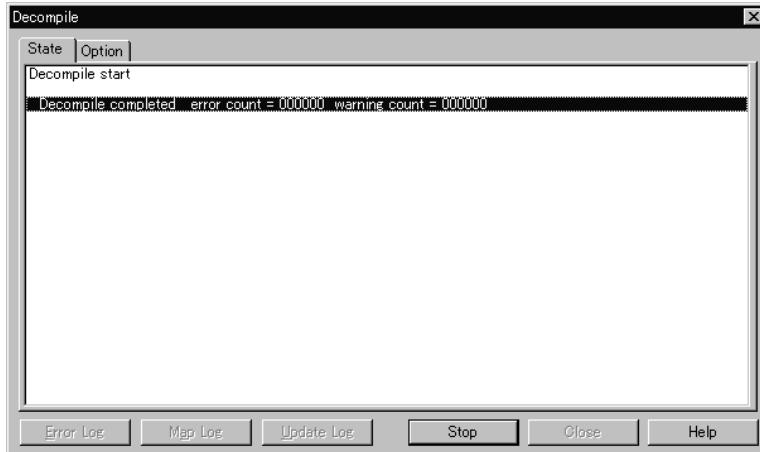
- 10 Click <Finish>. Then the program is transferred.
11 The [Transfer monitor] screen is displayed during program transfer.



- 12 When the load is finished, the following dialog is displayed.



- 13 Click <Yes>. Then decompile starts.



14 Edit PMC program Off-line.

NOTE

You can load only "Ladder" program. Load of "PMC parameter", "Multi-language PMC message data" and "I/O configuration data" are not supported.

5.10 STORE/LOAD PMC PROGRAM

The robot Controller can save the PMC program to a file named LADDER1.PMC, LADDER2.PMC, LADDER3.PMC, LADDER4.PMC, LADDER5.PMC, LADDERS.PMC.

FANUC LADDER-III for Robot also can save/load these files.

You can transfer a PMC program between the robot controller and FANUC LADDER-III for Robot without the online connection using the LADDER.PMC file.

This chapter explains the procedure to load/save LADDER*.PMC using FANUC LADDER-III for Robot.

NOTE

When Multi-path PMC option (J763) is not loaded, LADDER2.PMC, LADDER3.PMC, LADDER4.PMC and LADDER5.PMC are not saved.

When DCS Safety PMC option (J764) is not loaded, LADDERS.PMC is not saved.

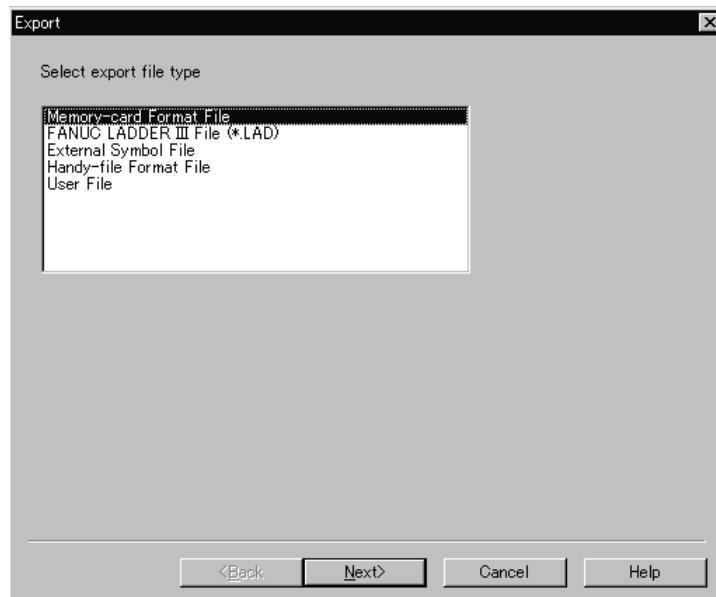
5.10.1 Exporting LADDER*.PMC using FANUC LADDER-III for Robot

FANUC LADDER-III for Robot can export LADDER*.PMC directly.

You can transfer a PMC program from FANUC LADDER-III for Robot to the robot controller without connecting the robot controller and FANUC LADDER-III for Robot by reading the LADDER*.PMC that was exported by FANUC LADDER-III for Robot.

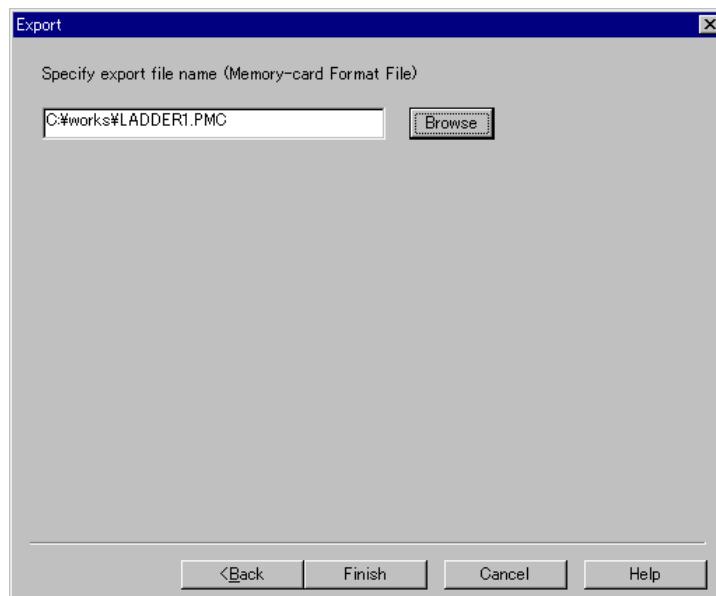
Procedure to export LADDER*.PMC

- 1 Compile the PMC program by FANUC LADDER-III for Robot.
- 2 Select [File]-[Export].
- 3 Choose "Memory-card Format File" and click <Next>.

**NOTE**

The robot controller supports "Memory-card Format File" format only. Do not select any other format.

- 4 Specify directory and file name to export.
File name must be followings for each PMC type.
 - PMC1 : LADDER1.PMC
 - PMC2 : LADDER2.PMC
 - PMC3 : LADDER3.PMC
 - PMC4 : LADDER4.PMC
 - PMC5 : LADDER5.PMC
 - DCS(Safety PMC) : LADDERS.PMC
 ex) C:\works\LADDER1.PMC



- 5 Click <Finish> to export LADDER*.PMC.
- 6 When LADDER*.PMC is exported, the following message is displayed.



- 7 Click <OK>.

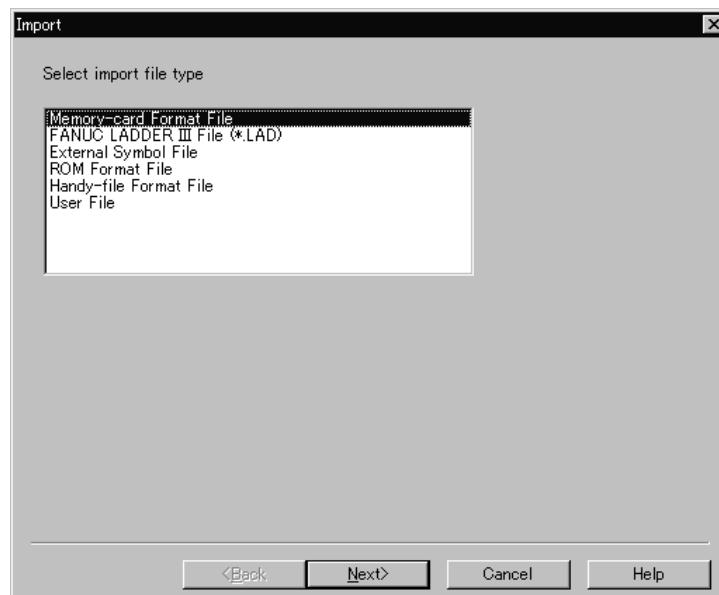
5.10.2 Importing LADDER*.PMC using FANUC LADDER-III for Robot

FANUC LADDER-III for Robot can import LADDER*.PMC directly.

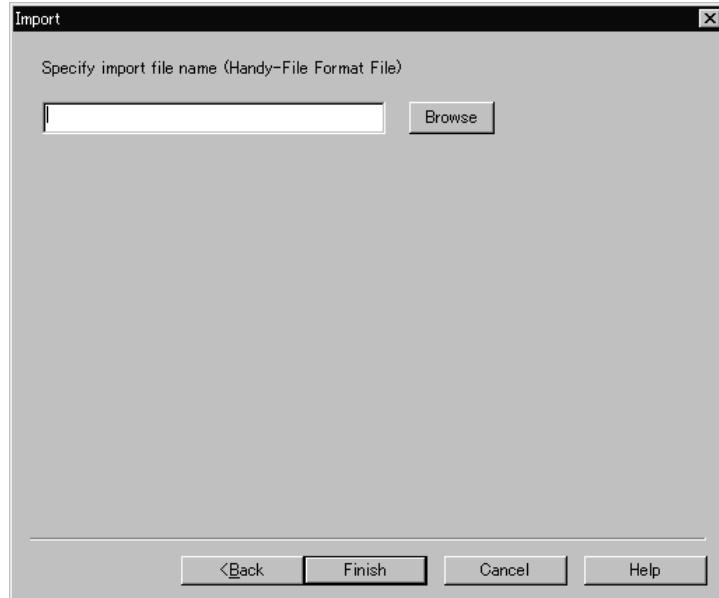
You can transfer a PMC program from the robot controller to FANUC LADDER-III for Robot without connecting the robot controller and FANUC LADDER-III for Robot by importing the LADDER*.PMC that was saved by the robot controller.

Procedure to import LADDER*.PMC

- 1 According to "5.2 CREATING PMC PROGRAM", create new PMC program.
- 2 Select [File]-[Import].



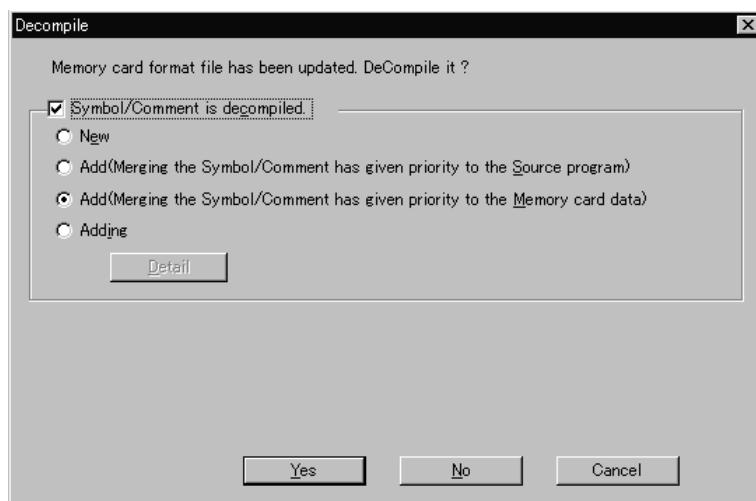
- 3 Select "Memory-card Format File".
Click <Next>.



- 4 Specify LADDER*.PMC to import by directory and file name.
Click <Finish>. The following message appears.



- 5 Click <OK>. The following message appears.



- 6 Click <Yes>. The [Decompile] screen appears.



- 7 Edit the exported PMC program by Off-line function.

6 TEACH PENDANT OPERATION

This chapter explains the PMC operations available on the Teach Pendant.

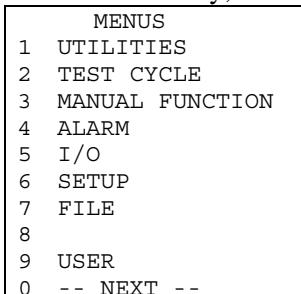
6.1 PMC MENUS

PMC menus have the following functions.

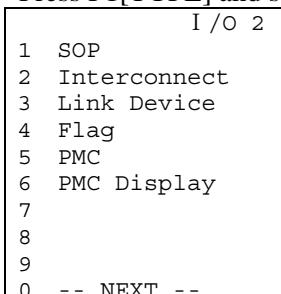
1. Display and set the PMC address (Bit menu, Byte menu)
2. Display and set the Timer value (Timer menu)
3. Display and set the Counter value (Counter menu)
4. Display and set the Data Table Control Data (Data Table control data menu)
5. Display and set the Data Table value (Data Table menu)
6. Display and set the PMC setting parameters (Parameters menu)
7. Display the PMC program size and scan time (Status menu)
8. Display the Title data (Title menu)
9. Setting of PMC I/O assignment (Internal I/O assignment menu and External I/O assignment menu)
10. Search
11. Run PMC and the stop PMC.
12. Add Data Table Group and delete Data Table Group
13. Multi-Path PMC option (A05B-2600-J763) is needed to perform following operation.
14. Setting of Multi-Path PMC (PMC Setting menu)
15. Display of status and executed time of Steps for Step Sequence function (Step Status menu)

Procedure to display PMC menus

- 1 Press MENU key, and select 5 I/O in the displayed menu.

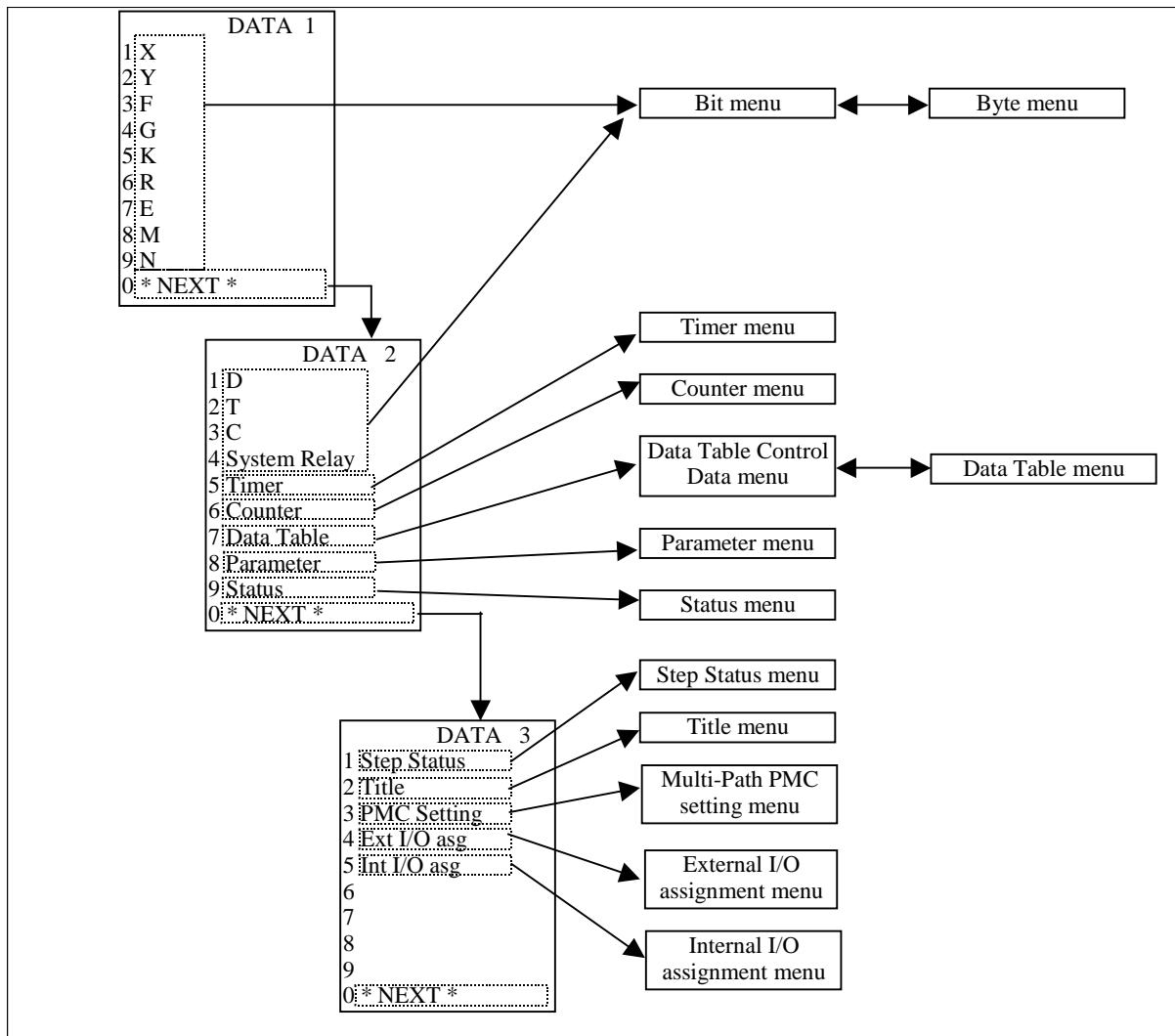


- 2 Press F1[TYPE] and select PMC.



Construction of PMC menus

PMC menus are made up of sub menus. You can choose between the sub menus by pressing F2[DATA].



6.1.1 Byte Menu

The Byte menu displays the value, symbol and comment of the byte address.

PMC 1 <RUNNING>		JOINT 100%								X0000.0	
X BYTE		X0000.0									
Addr.	Symbol	7	6	5	4	3	2	1	0	Hex	Dec
X0000		0	0	0	0	0	0	0	:	00	0
X0001		0	0	0	0	0	0	0	:	00	0
X0002		0	0	0	0	0	0	0	:	00	0
X0003		0	0	0	0	0	0	0	:	00	0
X0004		0	0	0	0	0	0	0	:	00	0
X0005		0	0	0	0	0	0	0	:	00	0
X0006		0	0	0	0	0	0	0	:	00	0
X0007		0	0	0	0	0	0	0	:	00	0
X0008		0	0	0	0	0	0	0	:	00	0
Comment:	[TYPE] [DATA] [FUNC]									BIT	

Addr.

The Addr. column displays the byte address.

Symbol

The Symbol column displays the symbol of the byte address.

7, 6, 5, 4, 3, 2, 1, 0

The 7-0 columns display the value of every bit of the address.

When the cursor is on these columns, you can set the bit to 1 by pressing the 1 key and to 0 by pressing the 0 key.

Hex

The Hex column displays the value of the byte address hexadecimal format.

When the cursor is on this column, you can set a new value with the numeric keys and the ENTER key. To enter the A-F characters, press ↑ or ↓ key to select a character and press → to decide.

Dec

The Dec column displays the value of the byte address in decimal format.

When the cursor is on this column, you can set a new value with the numeric keys and the ENTER key.

NOTE

For DCS Safety PMC, address value is displayed, but writing value is protected in this screen.

Comment

The Comment line displays the coil comment of the byte address.

Display another address

To display another address, press ITEM and type the numeric part of the PMC address at the "Address:" prompt.

To display the address that has different alphabetic part, press F2[DATA] and select the alphabetic character in the displayed menu.

Go to bit menu

To go to the bit menu press F5(BIT).

Conditions for changing value

You can change an address value whether the PMC is running or stopped.

You can not change a value when the "Change address value" item in the Parameters menu is DISABLED (K900.4 is 0).

Setting for PMC Override

When the Override function is enabled, function menu can be change if a F→ key is pressed.

			SIMULATE	UNSIM>
F1	F2	F3	F4	F5

To set override, move the cursor to the address and press F4(SIMULATE). Then the address is set as override. '*' is displayed at left side of values if the address is set as override. Similarly, to release override setting, press F5(UNSIM).

When the cursor is on the value of the BIT address column, override is set to the bit address. When the cursor is on the value of the Hex or Dec, override is set to the all bit address on the line.

When "Use Override mode" in the parameter menu (K906.0 = 0) is DISABLE, override cannot be used.

NOTE

All override address settings are cleared after power is cycled.

NOTE

For DCS Safety PMC, override function cannot be used.

6.1.2 Bit Menu

The Bit menu displays the value, symbol and comment of bit addresses. The corresponding port name is also displayed.

PMC 1 <RUNNING>		
X BIT	Symbol	Prot name
Address		Value
X0000.0	DI[1]	: 0
X0000.1	DI[2]	: 0
X0000.2	DI[3]	: 0
X0000.3	DI[4]	: 0
X0000.4	DI[5]	: 0
X0000.5	DI[6]	: 0
X0000.6	DI[7]	: 0
X0000.7	DI[8]	: 0
X0001.0	DI[9]	: 0
Comment:		
[TYPE][DATA] [FUNC]		BYTE

Address

The Address column displays bit address.

Symbol

The Symbol column displays symbol of the bit address.

Port name

The Port name column displays the corresponding I/O port name of the bit address. The System interface name is also displayed in this column.

If an internal relay is assigned to an I/O port, the assigned I/O port name is also displayed in this column.

If there are multiple corresponding I/O ports, they are displayed side by side. If the length is too long to display, the last character becomes ">".

Example: "DO[10001], GO[1]-0, DO>"

In this case, you can scroll through the port names by pressing the ← key and the → key.

Value

The "Value" column displays the value of the bit address.

When the cursor is on this column, you can set the bit to 1 by pressing the 1 key to 0 by pressing the 0 key.

NOTE

For DCS Safety PMC, address value is displayed, but writing value is protected in this screen.

Comment

The Comment line displays the coil comment of the byte address.

Display another address

To display another address, press ITEM and type the numeric part of the PMC address at the "Address:" prompt.

To display an address that begins with a different letter, press F2[DATA] and select the alphabetic character in the displayed menu.

Go to byte menu

To go to the byte menu, press F5(BYTE).

Conditions for changing value

You can change an address value whether the PMC is running or stopped.

You can not change a value when the Change address value item in the Parameters menu is DISABLED (K900.4 is 0).

Setting for PMC Override

When Override function is enabled, function menu can be change if a F→ key is pressed.

		SIMULATE	UNSIM	>
F1	F2	F3	F4	F5

To set override, move the cursor to the address and press F4(SIMULATE). Then the address is set as override. '*' is displayed at left side of values if the address is set as override. Similarly, to release override setting, press F5(UNSIM).

When "Use Override mode" in the parameter menu (K906.0 = 0) is DISABLE, override cannot be used.

NOTE

All override address settings are cleared after power is cycled.

NOTE

For DCS Safety PMC, override function cannot be used.

6.1.3 Timer Menu

The Timer menu displays and sets the timer value for the function command TMR (SUB3).

PMC 1 <RUNNING>					
No.	Addr.	Symbol	Data	Accuracy	1 / 250
1	T0000		2016msec	48msec	
2	T0002		0msec	100msec	
3	T0004		0sec	1sec	
4	T0006		0min	1min	
5	T0008		0msec	48msec	
6	T0010		0msec	48msec	
7	T0012		0msec	48msec	
8	T0014		0msec	48msec	
9	T0016		0msec	8msec	
10	T0018		0msec	8msec	
Comment:					
[TYPE] [DATA] [FUNC]					

No.

The No. column displays the timer number. This number is specified in the TMR function command as the "timer number" parameter.

Addr.

The Addr. column displays the corresponding PMC address. One timer uses two bytes of the T area.

Symbol

The Symbol column displays the symbol of the byte address.

Data

The Data column displays the timer value .

If accuracy is set as 48msec, 8msec, 1msec, 10msec or 100msec, unit of date is ‘msec’.

If accuracy is set as 1sec, unit of date is ‘sec’.

If accuracy is set as 1min, unit of date is ‘min’.

You can set a new value using the numeric keys and the ENTER key.

- Timers have a resolution of accuracy value.
- If you enter a preset value that is not divisible by the resolution, the remainder is omitted.

Example: If you enter 60 when accuracy value is 48, the value becomes 48.

Setting Data for each accuracy value are as following.

Timer No.	Accuracy	Min. Time	Max Time
1~8	48msec(default)	48msec	1572.8sec
9~500	8msec(default)	8msec	262.1sec
9~500	1msec	1msec	32.7sec
9~500	10msec	10msec	327.7sec
9~500	100msec	100msec	54.6min
9~500	1sec	1sec	546min
9~500	1min	1min	546hour

Accuracy

The Accuracy column displays the timer accuracy value.

You can set a new value using the F4[CHOISE] key.

For Timer 1-8, ‘48msec’, ‘1msec’, ‘10msec’, ‘100msec’, ‘1sec’ and ‘1min’ can be chosen. ‘8msec’ cannot be used.

For Timer 9 and above, ‘8msec’, ‘1msec’, ‘10msec’, ‘100msec’, ‘1sec’ and ‘1min’ can be chosen. ‘48msec’ cannot be used.

Comment

The Comment column displays the coil comment of the byte address.

Condition for changing the value

You can not change a timer value when the PMC is running. You must stop the PMC to change the value.

6.1.4 Counter Menu

The Counter menu displays and sets the preset value and current value of the CTR (SUB5) function command.

PMC 1 <RUNNING>				
Counter	Data type:	Binary	1 / 100	
No.	Addr.	Symbol	Preset	Current
1	C0000	:	4	/ 1
2	C0004	:	0	/ 0
3	C0008	:	0	/ 0
4	C0002	:	0	/ 0
5	C0006	:	0	/ 0
6	C0000	:	0	/ 0
7	C0004	:	0	/ 0
8	C0028	:	0	/ 0
9	C0032	:	0	/ 0
10	C0036	:	0	/ 0
Comment:				
[TYPE][DATA][FUNC]				

No.

The No. column displays the counter number. This number is specified in the CTR function command as the "counter number" parameter.

Addr.

The Addr. column displays the corresponding PMC address. One counter uses four bytes of the C area.

Symbol

The Symbol column displays symbol of the byte address.

Preset

The Preset column displays preset value of the counter.

You can set a new value using the numeric keys and the ENTER key.

Current

The Current column displays current value of the counter.

You can set a new value using the numeric keys and the ENTER key.

Comment

The Comment column displays coil comment of the byte address.

Condition for changing the value

You can not change preset value and current value when PMC is running. You must stop the PMC to change the value.

6.1.5 Data Table Control Data Menu

The Data Table Control Data menu displays and sets the Data Table Control Data.

PMC 1 <RUNNING>					
Grp	Address	Number	Access	Type	Protect
1	D0000	200	Word	Dec	On
2	D0200	1000	Byte	Hex	Off
3	D1200	100	Word	BCD	Off

Press F5 to monitor data table
[TYPE] [DAT] [FUNC] DETAIL

Grp

The Grp column displays the Data Table group number.

Address

The Address column displays the starting address of the Data Table group.

You can set a new value with the numeric keys and the ENTER key.

Number

The Number column displays the number of elements in the Data Table group.

You can set a new value with the numeric keys and ENTER key.

Access

The Access column displays the accessing mode of the Data Table group.

Byte: One element uses one byte of D area.

Word: One element uses two bytes of D area.

DWord: One element uses four bytes of D area.

To change the access mode, press the F4[CHOICE] key and select an item from the displayed menu.

Type

The Type column displays the data type of the Data Table group.

Dec: Data is displayed as decimal format.

Hex: Data is displayed as hexadecimal format.

BCD: Data is displayed as BCD format.

To change the data type press F4[CHOICE], and select an item from the displayed menu.

Protect

The Protect column displays the protection setting of the Data Table group.

Off: You can change the value of the Data Table group.

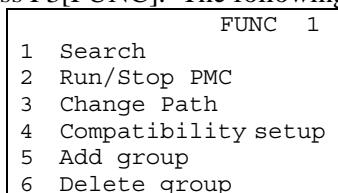
On: You can not change the value of the Data Table group.

To change the protection setting press F4[CHOICE] and select an item from the displayed menu.

The protection setting is used only in the Data Table menu of the PMC menus and the Data Table monitor in PMC programmer. You can change the value of the D area from the Byte menu or Bit menu even though the protection setting is On.

To add a new Data Table group

Press F3[FUNC]. The following menu is displayed.



Select Add group. A new Data table group is inserted under the current Data Table group.

The new Data Table group has the same setting as the current line.

The Data Table groups under the new Data Table group are moved down.

To delete a Data Table group

Select Delete group in the menu displayed by F3[FUNC] to delete the current Data Table group.

The Data Table groups under the deleted Data Table group are moved up.

Conditions for changing value

You can not change Data Table Control Data when PMC is running. You must stop the PMC to change the Data Table Control Data.

You can not change Data Table Control Data when the "Protect data tbl ctl" item in the Parameters menu is ENABLED (K900.7 is 1).

To go to the Data Table menu

To display the Data Table data of the current Data Table group, press F5(DETAIL) or ENTER.

6.1.6 Data Table Menu

The Data Table menu displays and sets the data of the Data Table for every Data Table Group.

PMC 1 <RUNNING>				
No.	Addr.	Symbol	Data(Dec)	
1	D0000	:	0	
2	D0002	:	0	
3	D0004	:	0	
4	D0006	:	0	
5	D0008	:	0	
6	D0010	:	0	
7	D0012	:	0	
8	D0014	:	0	
9	D0016	:	0	
10	D0018	:	0	

Comment:
[TYPE] [DATA] [FUNC] NEXT LIST

Data table grp.

The selected Data Table group number is displayed in the upper line of the screen as Data table grp.1.

Protect

The Protection setting of the selected Data Table group is displayed in the upper line of the screen as Protect:Off.

You can not change the protection setting in this menu.

No.

The No. column displays element number.

NOTE

The first element number is 1. The first number of data in Table of function commands DSCH, DSCHB, XMOV and XMOVB is 0.
(Data number in Table) = (Element number) - 1

Addr.

The Addr. column displays the PMC address in the D area.

Symbol

The Symbol column displays the symbol of the byte address.

Data

The Data column displays the data of the element.

Data type is displayed in right side of "Data" header as "Data(Dec)".

Data is displayed in the selected Data type format.

You can set a new value with the numeric keys and the ENTER key.

When the data type is Hex, enter A-F characters by pressing the ↑ or ↓ keys to select the character and pressing → to decide.

Comment

The Comment column displays the coil comment of the byte address.

To display the next Data Table group

To display the next Data Table group, press F4(NEXT).

When the last Data Table group is displayed, Data Table group 1 is displayed by pressing F4(NEXT).

To go to the Data Table Control Data menu

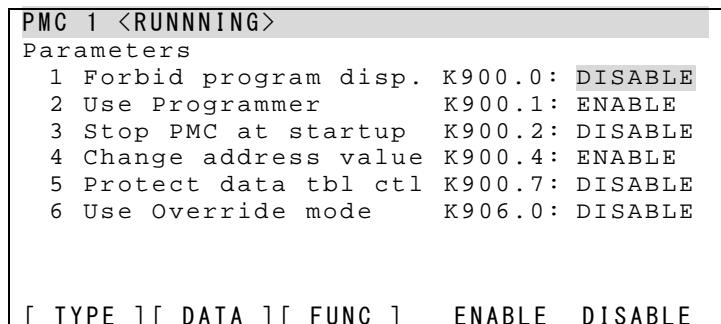
To display Data Table Control Data menu, press F5(LIST) or PREV.

Conditions for changing data

You can not change the Data Table data when the PMC is running. You must stop the PMC to change data. When the protection setting is On, you can not change the Data Table data even though PMC is stopped.

6.1.7 Parameters Menu

The Parameters menu displays and sets the PMC setting parameters.



Forbid program disp.

DISABLE : PMC program can be displayed by PMC monitor function.

Ladder monitor in PMC programmer can be displayed.

ENABLE : PMC monitor cannot be displayed by PMC monitor function.

Ladder monitor in PMC programmer can not be displayed.

Use Programmer

DISABLE : PMC programmer is not available. ("Load program", "Store program", "Backup program" and "Run/Stop the program" are not available in PMC programmer.)

ENABLE : PMC programmer is available.

Stop PMC at startup

DISABLE : PMC runs at power up automatically.

ENABLE : PMC does not run at power up automatically.

Change address value

DISABLE : You can not change values in the Byte menu and Bit menu of the PMC menus.

You can not change values in "Signal status" of the PMC programmer.

ENABLE : You can change values in the Byte menu and Bit menu of the PMC menus.

You can change values in "Signal status" of the PMC programmer.

Protect data tbl ctl

DISABLE : You can change Data Table Control Data in the Data Table Control Data menu.

The Data Table Control Data menu in PMC programmer can be displayed.

ENABLE : You can not change Data Table Control Data in the Data Table Control Data menu.

The Data Table Control Data menu in PMC programmer can not be displayed.

Use Override mode

DISABLE : Override mode is not available.

ENABLE : Override mode is available.

Override mode is a function for checking of PMC program operation. Please set DISABLE in production.

NOTE

For DCS Safety PMC, only value of "K900.0 Forbid program disp", "K900.1 Use Programmer", "K900.4 Change address value" and "K900.7 Protect data tbl ctl" can be changed. Other items cannot be changed.

6.1.8 Status Menu

Status menu displays PMC program size and scan time.

```
PMC 1 <RUNNING>
Status

Program size(step)
Level 1:      142
Level 2:     2986
Level 3:       0
Total:        3128 ( 40 / 128KB )

Scan time(ms)
Current:      8
Maximum:      8
Minimum:      8

[ TYPE ][ DATA ][ FUNC ]
```

Program size

The Program size area displays the number of steps in the PMC program. The number of steps of level 1, level 2, level 3 and the total of level 1, level 2, level 3 and all sub programs are displayed. The used program size and capacity are displayed in the right of "Total" line as K Byte unit.

Scan time

The Scan time area displays the scan time of level2. Current scan time, maximum scan time and minimum scan time are displayed. Maximum scan time and minimum scan time are the maximum and minimum since the last PMC program was started.

6.1.9 Step Status Menu

The Step Status menu displays Step Status data for Step Sequence Program.

NOTE

The Step Status menu is not display without Multi-Path PMC option (A05B-2600-J763).

```
PMC 1 <RUNNING>
Step Sequence: Step Status      1 / 2000
No.   Step Symbol  Status    Time(ms)
  1 S0000      :          5168
  2 S0001      : ACTIVE    2125
  3 S0002      :          0
  4 S0003      :          2168
  6 S0004      :          0
  7 S0005      :          0
  8 S0006      :          0
  9 S0007      :          0
 10 S0008      :          0

[ TYPE ][ DATA ][ FUNC ]
```

No.

The No. column displays step element number.

Step

The Step column displays S address of the Step.

Symbol

The Symbol column displays the symbol of the byte address.

Status

The Status column displays the status of the step. When the step is activated, ACTIVE is displayed on the column.

Time

The Time column displays elapsed time since the step is activated. When the step is inactive, timer count is stopped.

6.1.10 Title Menu

The Title menu displays title data of the PMC program.

Title data is edited with the PMC programmer.

PMC 1 <RUNNING>	
Title	
Company: FANUC	
Machine: R-30iB	
PMC: R-30iB ROBOT PMC	
Program: 1	
Edition: 10	
Drawing:	
Date: 2011/6/1	
Author:	
Install:	
remarks:	
[TYPE]	[DATA]
[FUNC]	

Corresponding title data

The following is the corresponding item in the title menu of PMC programmer for every line.

Corresponded item in FAPT LADDER-II for Robot	
Company	MACHINE TOOL BUILDER NAME
Machine	MACHINE TOOL NAME
PMC	PMC & NC NAME
Program	PMC PROGRAM NO
Edition	EDITION NO
Drawing	PROGRAM DRAWING NO
Date	DATE OF PROGRAMMING
Author	PROGRAM DESIGNED BY
Install	ROM WRITTEN BY
Remarks	REMARKS

6.1.11 PMC Setting Menu

The PMC Setting menu displays and changes Multi-PMC setting.

NOTE

The PMC Setting menu is not display without Multi-Path PMC option (A05B-2600-J763).

```
PMC 1 <RUNNING>
PMC Setting
Maximum PMC path : 3
Level 1 Cycle Time: 8ms
Execution Type : Interlock
PMC interface(M,N) : PMC1-PMC2

PMC STATUS Type SEQ Rate Memory
PMC1 RUNNING C 1 50% 256KB
PMC2 RUNNING B 2 20% 256KB
PMC3 STOP A 3 10% 256KB
PMC4 NONE A 4 10% 256KB
PMC5 NONE A 5 10% 256KB

[ TYPE ][ DATA ][ FUNC ]
```

Maximum PMC path

The maximum number of used PMC path is displayed.

PMC paths over this value cannot be used.

To change maximum PMC path, move cursor to the item ‘Maximum PMC path’, input a path number from 1 to 5.

NOTE

If setting of Maximum PMC path is changed to a lower number, programs of path over this setting are deleted at next power up.

Level 1 Cycle Time

Cycle Time of Level1 is displayed.

To the change cycle time, move cursor on the item of ‘Level 1Cycle Time’, press F4[CHOICE] and select cycle time from ‘8ms’ and ‘4ms’.

Execution Type

PMC execution type is displayed.

To the change type, move cursor to the item of ‘Execution Type’, press F4[CHOICE] and select execution type form ‘Interlock’ and ‘Independent’.

Interlock: When user runs/stops one of PMCs execution, all other loaded PMC path programs are run/stop execution.

Independent: When user runs/stops PMC program, only selected PMC program starts/stops execution.

PMC Interface (M,N)

Interface between PMC-path (M,N address) setting is displayed.

To change setting, move cursor to the item ‘PMC Interface’, press F4[CHOICE] and select setting type.

None : Interface between PMC path is not used.

PMC1-PMC2 : Interface between PMC path is used by 1st PMC and 2nd PMC.

PMC1-PMC3 : Interface between PMC path is used by 1st PMC and 3rd PMC.

PMC2-PMC3 : Interface between PMC path is used by 2nd PMC and 3rd PMC.

PMC

The column displays PMC path number.

STATUS

The column displays status of PMC.

RUNNING :PMC program is running.

STOP :PMC program is stopped

NONE :PMC program is not loaded or invalid PMC program is exist.

Type

The column displays memory type of PMC.

To change memory type move cursor to the column 'Type', press F4[CHOICE] and select setting type.

1st PMC : Memory type B, C and D can be selected.

2nd PMC : Memory type A, B, C and COM can be selected.

3rd PMC : Memory type A, B, C and COM can be selected.

4th PMC : Memory type A and COM can be selected.

5th PMC : Memory type A and COM can be selected.

When memory type of 1st PMC is D, 2nd-5th PMC can be selected only COM (Common Memory Mode).

SEQ

The column displays execution sequence.

To change sequence number, move cursor to the column 'SEQ', input sequence number from 1 to maximum PMC path number.

Rate

The column displays execution rate.

To change rate, move cursor to the column 'Rate', input number from 0% to 100%.

The total of all PMC execution rate have to be less or equal to 100%.

When execution rate is 0%, the PMC cannot run.

Memory

The column displays maximum memory capacity for PMC program.

When size of PMC program is over this value, its PMC program cannot be loaded.

To change the maximum capacity, move cursor to the column 'Memory', input maximum memory size in multiples of 128KB.

The total of all PMC memory capacity have to be less or equal to 3MB (3072KB).

6.1.12 External I/O Assignment Menu

The External I/O Assignment menu displays and sets the PMC external I/O assignment. Please refer "3.2 PMC External I/O Assignment" for the detail.

NOTE

For DCS Safety PMC, assignment between PMC address and safety I/O are fixed.
External I/O assignment cannot be used for safety I/O.

PMC 1 <RUNNING>						
External I/O assignment						
Type	Rack	Slot	Size	Address	Occpy	
1 DI	0	1	12	1:X00000	NO	
2 DO	0	1	12	1:Y00000	YES	
3 RI	0	0	1	1:X01000	NO	
4 RO	0	0	1	1:Y01000	NO	
5 ---	0	0	0	1:-00000	NO	
6 ---	0	0	0	1:-00000	NO	
7 ---	0	0	0	1:-00000	NO	
8 ---	0	0	0	1:-00000	NO	
9 ---	0	0	0	1:-00000	NO	

[TYPE] [DATA] [FUNC] [CHOICE] ALL_CLR

Type

The signal type of the assigned external I/O device is displayed.

To change setting, move cursor to the column 'Type', press F4[CHOICE] and select signal type.

Rack

The rack number of the assigned external I/O device is displayed.

To change the rack number, move cursor to the column 'Rack', input rack number.

Slot

The slot number of the assigned external I/O device is displayed.

To change the slot number, move cursor to the column 'Slot', input slot number.

Size

The signal size of the connected external I/O device is displayed as BYTE size.

When the assigned external I/O device is connected, the size based on the number of the signals is displayed.

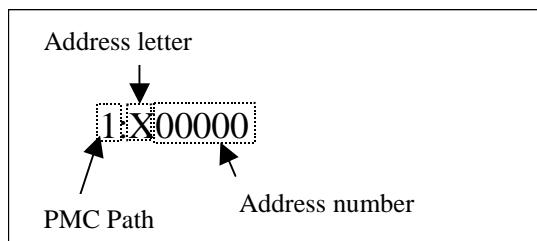
When the device is not connected, 0 is displayed.

The size cannot be changed.

Address

The external I/O device is assigned to PMC address by the setting of the top PMC address in "Address" column.

The PMC address is indicated as the format like "1:X00000". The PMC address consists of "PMC path", "Address letter" and "Address number", and each element can be changed.



- When cursor is on PMC path, the PMC path can be changed by pressing F4[CHOICE]. However, PMC path cannot be changed when address letter is not set.
- When cursor is on address letter, the address letter can be changed by pressing F4[CHOICE]. X or R can be selected for input signal, Y or R can be selected for output signal. When address letter is set, address number will be set automatically as the minimum available address number.
- When cursor is on address number, the address number can be changed by pressing ENTER key, and enter number. However, address number cannot be changed when address letter is not set.

The "*" mark is displayed on the left of address in some external I/O device. For this external I/O device, the address must be multiple of 32.

Occpy

Occupancy setting of external I/O device on this setting line can be changed. Occupancy setting is enabled when 'OCCPY' item is set 'YES'. For occupancy mode, external I/O device cannot be accessed by except PMC functions.

It can be set 'YES', when 'TYPE' is DI, DO, AI or AO. It cannot be set enable if 'TYPE' is RI, RO, SI, SO, WI, WO, WSI or WSO.

Please refer "3.2.3 Occupancy Setting" for detail of this function.

NOTE

Power failure recovery of all I/O ports is not performed if any occupancy setting is changed.

ALL_CLR

All setting of PMC external I/O assignment is cleared by this function.

When F5 (ALL_CLR) is pushed, "Clear all external I/O assignment?" is displayed.

All PMC external I/O assignment is cleared by selecting F4 (YES).

6.1.13 Internal I/O Assignment Menu

The Internal I/O Assignment menu displays and sets the PMC internal I/O assignment. Please refer "3.3 PMC Internal I/O Assignment" for the detail.

NOTE

For DCS Safety PMC, assignment between PMC address and safety I/O are fixed.
Internal I/O assignment cannot be used for safety I/O.

PMC 1 <RUNNING>			
Internal I/O assignment 2 msec			
	Robot data	Size	Address
1	DO[1- 128]	16	1:F00100
2	DI[1- 128]	16	1:G00100
3	GO[1- 4]	8	1:D00000
4	GI[1- 4]	8	1:R00000
5	UO[1- 20]	8	1:F00000
6	UI[1- 18]	8	1:G00000
7	----[0- 0]	0	1:-00000
8	----[0- 0]	0	1:-00000
9	----[0- 0]	0	1:-00000
10	----[0- 0]	0	1:-00000
[TYPE] [DATA] [FUNC] [CHOICE] ALL_CLR			

The processing period of the transference and the synchronization is displayed on the title line (right of "Internal I/O assignment"). The processing period affects the response time of the following internal I/O assignment.

- Robot: Register → PMC: F Transference
- Robot: Register ← PMC: G Synchronization
- Robot: GI,AI → PMC: F Transference
- Robot: GO,AO ← PMC: G Synchronization

Robot data

In "Robot data" column, I/O type, start index and end index of the assigned robot data is displayed. I/O type and start index can be changed. End index is displayed automatically when the size is set.

The signal type of the assigned external I/O device is displayed.

- When cursor is on I/O type, the I/O type can be changed by pressing F4[CHOICE].
- When cursor is on top index, the top index can be changed by pressing ENTER key and enter the number.

Size

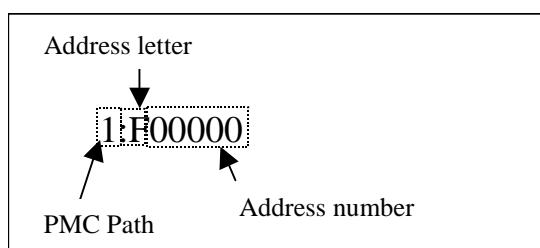
The PMC address size to be assigned to the robot data is displayed as BYTE size.

The size can be changed by pressing ENTER key and enter the number. When robot data and size are set, PMC address is set automatically to the recommended address.

Address

Top PMC address to be assigned is displayed in "Address" column.

The PMC address is indicated as the format like "1:F00000". The PMC address consists of "PMC path", "Address letter" and "Address number", and each element can be changed.



- When cursor is on PMC path, the PMC path can be changed by pressing F4[CHOICE]. However, PMC path cannot be changed when address letter is not set.
- When cursor is on address letter, the address letter can be changed by pressing F4[CHOICE]. When address letter is set, address number will be set automatically as the minimum available address number.
- When cursor is on address number, the address number can be changed by pressing ENTER key and enter number. However, address number cannot be changed when address letter is not set.

ALL_CLR

All setting of PMC internal I/O assignment is cleared by this function.

When F5 [ALL_CLR] is pushed, "Clear all internal I/O assignment?" is displayed.

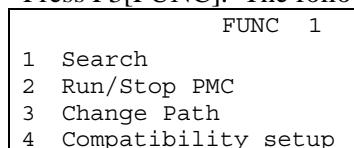
All PMC internal I/O assignment is cleared by selecting F4 [YES].

6.1.14 Search

Search procedure

Use the following procedure to search for PMC addresses, symbols, comments and port names.

- 1 Press F3[FUNC]. The following menu is displayed.



- 2 Select "1 Search", and type the search word at the "Address/Symbol/Port name:" prompt.

Search PMC address

To search for a PMC address, type the PMC address like "X1.0" or "X12".

If you type a bit address such as "X1.0", the bit menu is displayed, and the cursor points to the specified address.

If you enter a byte address such as "X12", the byte menu is displayed, and the cursor points to the "Hex" column of the specified address.

Search symbol and comment

To search for a symbol or comment, type characters contained in the symbol or comment.

Search port name

To search for a port name, type the port name such as "DO[1]".

To search for group I/O or analog I/O, you can specify a bit number such as "GO[1]-3". If you do not specify a bit number, bit 0 is returned.

Found in multiple locations

If the searched word is found in multiple locations, the function key label is changed as follow.

	CANCEL		PREV	NEXT
F 1	F 2	F 3	F 4	F 5

Press F5(NEXT) to display next item.

Press F4(PREV) to display previous item.

Press F2(CANCEL) to cancel searching.

6.1.15 Run/Stop PMC

Display PMC execution status

The selected PMC execution status is always displayed in upper line of all PMC menus.

PMC is running



PMC is stopped



PMC execution status can be seen in the PMC setting menu.

NOTE

PMC status in the upper line of PMC menu is for the selected PMC path. To see status of other PMC paths, change the PMC path or switch menu to PMC setting menu.

When Safety PMC is selected in screen, 'S' is displayed as PMC path. When "Safety I/O process" is "Safety PMC" in DCS Safety I/O device screen, <RUNNING> is displayed as execution status. When it is "Safety I/O connect", <STOP> is displayed as execution status.



Procedure to run/stop PMC

You can run or stop the selected PMC by the following operation.

- 1 Press F3[FUNC], the following menu is displayed.

FUNC 1	
1	Search
2	Run/Stop PMC
3	Change Path
4	Compatibility setup

- 2 Select 2 Run/Stop PMC,
- When PMC is running, the prompt "Stop PMC program?" is displayed, if you press F4(YES) key, then PMC is stopped.
 - When PMC is stopped, the prompt "Run PMC program?" is displayed, if you press F4(YES) key, then PMC is started.

NOTE

Selected PMC is run by this operation.

When PMC Execution Type is 'Interlock', all loaded PMC program are run/stopped.

When Multi-path PMC setting or I/O assignment are invalid, PMC programs cannot be run.

NOTE

When DCS Safety PMC is selected in the screen, RUN/STOP cannot be executed.

6.1.16 Change PMC Path

Display PMC path

The selected PMC path is always displayed in upper line of all PMC menus.

PMC 1 <RUNNING>

Procedure to run/stop PMC

You can change PMC path by the following operation.

- 1 Press F3[FUNC], the following menu is displayed.

FUNC 1	
1	Search
2	Run/Stop PMC
3	Change Path
4	Compatibility setup

- 2 Select 3 Change path,

The prompt " Enter PMC number?" is displayed, input PMC path number.

Any number over the Maximum PMC path number cannot be input.

When "DCS Safety PMC (J764)" is ordered, the prompt "Enter PMC number (DCS:0)" is displayed.

When 0 is input, Safety PMC is selected.

NOTE

'Change path' is not displayed without "Multi-Path PMC (J763)" or "DCS Safety PMC (J764)" option.

NOTE

Different PMC path menu cannot be displayed by multi screen. When PMC path is changed in a screen, PMC path of other screens is also changed.

6.1.17 Save LADDER*.PMC and PARAM*.PMC

You can save LADDER1.PMC-LADDER5.PMC, LADDERS.PMC, PARAM1.PMC-PARAM5.PMC and PARAMS.PMC in PMC menus.

NOTE

LADDER2.PMC-LADDER5.PMC PARAM2.PMC- PARAM5.PMC are not saved without Multi-Path PMC option (A05B-2600-J763).

LADDER*.PMC and PARAM*.PMC of the PMC path is not saved when the sequence program does not exits.

LADDER*.PMC and PARAM*.PMC of the PMC path over maximum PMC path number can not be saved.

NOTE

LADDERS.PMC and PARAMS.PMC are not saved without "DCS Safety PMC (J764)" option.

Procedure to save LADDER*.PMC and PARAM*.PMC

NOTE

You must select an external file device in file menu before this procedure. Refer to Section 5.2, "Operation in File Menu," for information on how to select an external file device.

- 1 Press FCTN while in the PMC menus. The following menu is displayed.

FUNCTION 1
1 ABORT (ALL)
2 Disable FWD/BWD
3
4
5
6
7 RELEASE WAIT
8
9
0 -- NEXT --

- 2 Select – NEXT – to display next page, and select SAVE.

LADDER*.PMC and PARAM.P*MC of the selected PMC path are saved to the selected external file device. For example, 1st PMC path is selected and MC is selected as external file device, LADDER1.PMC and PARAM1.PMC are saved to MC:

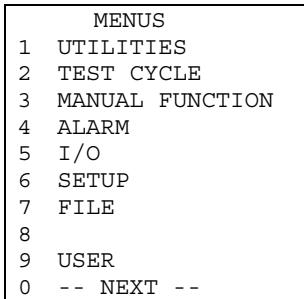
FUNCTION 2
1 QUICK/FULL MENUS
2 SAVE
3 PRINT SCREEN
4 PRINT
5
6 UNSIM ALL I/O
7
8 CYCLE POWER
9 ENABLE HMI MENUS
0 -- NEXT --

6.2 FILE MENU OPERATIONS

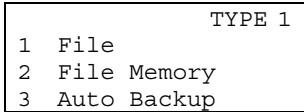
You can save and load LADDER1.PMC- LADDER5.PMC and PARAM1.PMC- PARAM5.PMC from the teach pendant FILE menu.

Operation to display File menu

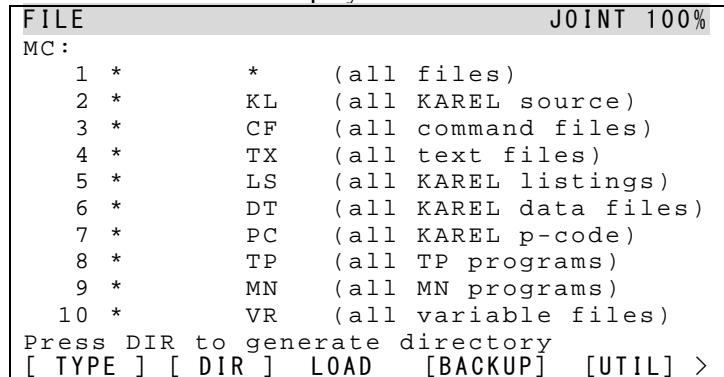
- 1 Press MENU, the following menu is displayed.



- 2 Press F1 [TYPE], and select "File" in the displayed menu".

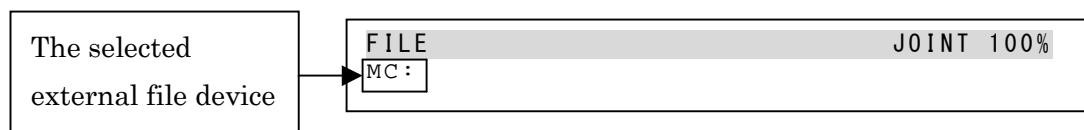


- 3 The File menu is displayed.



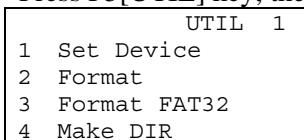
Select external file device

To perform operations on files in the File menu, you must select the target external file device. The selected external file device is displayed in left side of upper line in File menu.



Use the following procedure to select the external file device.

- 1 Press F5[UTIL] key, the following menu is displayed.



- 2 Select Set device, the following menu is displayed.

1
1 FROM Disk (FR:)
2 Backup (FRA:)
3 RAM Disk (RD:)
4 Mem Card (MC:)
5 Mem Device (MD:)
6 Console (CONS:)
7 USB Disk (UD1:)
8 -- next page --

- 3 Select external file device.

6.2.1 Save LADDER*.PMC, PARAM*.PMC, PMCCFG.SV

Save LADDER*.PMC, PARAM*.PMC and PMCCFG.SV

You can save LADDER*.PMC, PARAM*.PMC and PMCCFG.SV individually with the following procedure.

- 1 Press F4[BACKUP] on the File menu, the following menu is displayed.

BACKUP 1
1 System files
2 TP programs
3 Application
4 Aplic.-TP
5 Error log
6 Diagnostic
7 Vision data
8 All of above
9 Maintenance data
0 - NEXT --

- 2 Select System files, the following message is displayed.

Save MC:DIOCFGSV.IO?				
EXIT	ALL	YES	NO	
F1	F2	F3	F4	F5

- 3 Press F5(NO). Similar messages are displayed repeatedly. Select F5(NO) until LADDER*.PMC, PARAM*.PMC or PMCCFG.SV is displayed.

Save MC:LADDER1.PMC?				
EXIT	ALL	YES	NO	
F1	F2	F3	F4	F5

- 4 If LADDER*.PMC, PARAM*.PMC or PMCCFG.SV is displayed, press F4, YES.

- 5 After LADDER*.PMC, PARAM*.PMC and PMCCFG.SV are saved, press F2, EXIT.

NOTE

'LADDER2.PMC-LADDER5.PMC PARAM2.PMC- PARAM5.PMC are not saved without Multi-Path PMC option (A05B-2600-J763).
 LADDER*.PMC and PARAM*.PMC of the PMC path is not saved when the sequence program does not exits.
 LADDER*.PMC and PARAM*.PMC of the PMC path over maximum PMC path number are not saved.

NOTE

LADDERS.PMC and PARAMS.PMC are not saved without “DCS Safety PMC (J764)” option.

All backup

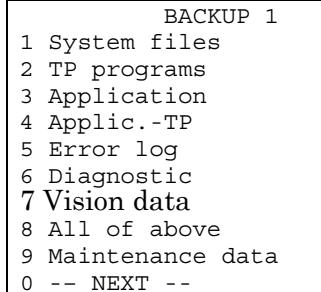
The "All backup" function in the File menu can back up all files to recover the current system. Then following PMC backup files are also saved with the "All backup" function.

- LADDER*.PMC, PARAM*.PMC : Files of PMC paths that were loaded.
- PMCCFG.SV

NOTE

All files in the selected external file device are deleted by the following procedure.

- 1 Press F4[BACKUP] in File menu. The following menu is displayed.



- 2 Select All of above. The following message is displayed.

Delete MC:\ before backup files??				
		YES	NO	
F1	F2	F3	F4	F5

- 3 Press F4(YES).

NOTE

‘LADDER2.PMC-LADDER5.PMC PARAM2.PMC- PARAM5.PMC are not saved without Multi-Path PMC option (A05B-2600-J763).

LADDER*.PMC and PARAM*.PMC of the PMC path is not saved when the sequence program does not exits.

LADDER*.PMC and PARAM*.PMC of the PMC path over maximum PMC path number are not saved.

NOTE

LADDERS.PMC and PARAMS.PMC are not saved without “DCS Safety PMC (J764)” option.

6.2.2 Load LADDER*.PMC, PARAM*.PMC, PMCCFG.SV

Load a selected file

- 1 Press F2[DIR]. The following menu is displayed.

```

FILE                               JOINT 100%
MC :
 1 *      *      (all files)
 2 *      KL     (all KAREL source)
 3 *      CF     (all command files)
 4 *      TX     (all text files)
 5 *      LS     (all KAREL listings)
 6 *      DT     (all KAREL data files)
 7 *      PC     (all KAREL p-code)
 8 *      TP     (all TP programs)
 9 *      MN     (all MN programs)
10 *      VR     (all variable files)
Press DIR to generate directory
[ TYPE ] [ DIR ] LOAD [BACKUP] [UTIL] >

```

- 2 Select "*.*". The file list is displayed.
(If there are many files, select "*.PMC" to display PMC files only.)

```

FILE                               JOINT 100%
MC :
 1 LADDER1  PMC   94524
 2 PARAM1   PMC   17831
 3 *      *      (all files)
 4 *      KL     (all KAREL source)
 5 *      CF     (all command files)
 6 *      TX     (all text files)
 7 *      LS     (all KAREL listings)
 8 *      DT     (all KAREL data files)
 9 *      PC     (all KAREL p-code)
10 *      TP     (all TP programs)
Press DIR to generate directory
[ TYPE ] [ DIR ] LOAD [ BACKUP ][ UTIL ] >

```

- 3 Move the cursor on the file to load, and press F3(LOAD). The following message is displayed.

Load MC:LADDER1.PMC?				
		YES	NO	
F1	F2	F3	F4	F5

- 4 Press F4 , YES, the file is loaded.

Notes on loading LADDER.PMC

If LADDER*.PMC is loaded, the previous PMC program is lost. The running PMC program is not changed by loading LADDER*.PMC. To run the loaded PMC program, you must cycle the robot controller power. LADDER*PMC of PMC path over maximum PMC path number can be loaded, but the programs are deleted at next power up. To use the programs, change setting of Maximum PMC path.

NOTE

For 7DC1 and 7DD0 software, The teach pendant E-STOP button or operator panel E-STOP button must be pressed to load LADDER*.PMC, if the controller is not in controlled start mode.

Notes on loading PARAM.PMC

If the controller is not in Controlled Start mode, the PMC program must be stopped to load PARAM*.PMC. If PARAM*.PMC is loaded, the previous value of backed up internal relays is lost. If the controller is not in Controlled Start mode, PARAM*.PMC of PMC path over maximum PMC path number can not be loaded.

Notes on loading PMCCFG.SV

If PMCCFG.SV is loaded, the previous PMC setting, maximum PMC number, multi-path PMC setting, External IO assignment and Internal IO assignment, is lost.

All restore

The "All restore" function in the File menu of the Controlled Start mode can load all files in the selected external file device. LADDER*.PMC and PARAM*.PMC are also loaded by the "All restore" function.

NOTE

All robot programs and settings are lost by the following procedure. The programs and settings become that of the files that are restored.

- 1 Turn the robot controller off. Then press and hold PREV and NEXT keys on teach pendant, and turn on the robot controller. The following menu is displayed.
- 2 Select "Controlled start", and wait a moment. The following menu is displayed.
- 3 Press MENU key and select "File". The File menu is displayed.
- 4 Press F4[RESTOR]. Select "All of above" in the displayed menu.

NOTE

LADDER*PMC and PARAM*.PMC of PMC path over maximum PMC path number are loaded, but the programs are deleted at next power up. To use the programs, change setting of Maximum PMC path.

Loading of LADDERS.PMC

LADDERS.PMC can be loaded as same as other LADDER*.PMC by file screen and all restore. When the Safety PMC program for editing and the Safety PMC program for execution are different by loading, "SYST-212 Need to apply to DCS param" occurs. This alarm cannot be reset until the "Apply to DCS parameter" procedure is done. Please apply to DCS parameter in DCS screen.

Please refer to "FANUC Robot Series R-30iB/ R-30iB Mate Controller Dual Check Safety Operator's Manual (B-83184EN)" for apply procedure.

6.3 PMC MONITOR FUNCTION

The PMC monitor function enables the program of the built-in PMC to be displayed on the teach pendant.

PMC ladder display

It is possible to display a PMC ladder program using the procedure below.

- 1 Press Screen select (menu).
- 2 Select I/O.
- 3 Press F1 [TYPE].
- 4 Select PMC Display and press the ENTER key.
A screen shows as that shown in the figure below appears.

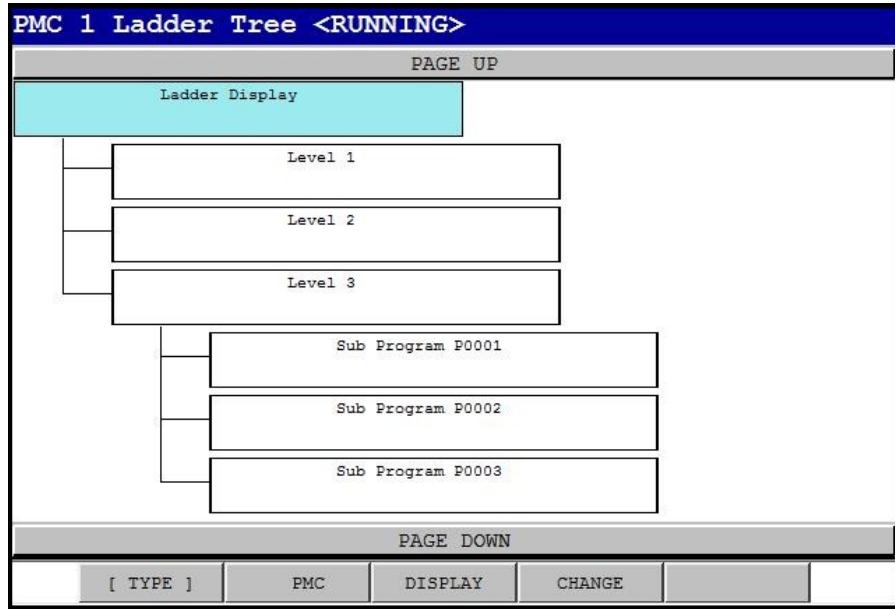


Fig. 6.3 (a) PMC Ladder Tree view

- 5 Use the PMC key to select the PMC path and a screen such as that shown in Fig. 6.3 (a) appears for the specific PMC path.
- 6 Use the \uparrow or \downarrow key to move the cursor to the desired level or sub-program to be displayed. Use SHIFT- \uparrow or SHIFT- \downarrow to page up and down in the Ladder Tree View.
- 7 Press F3 [DISPLAY] or the ENTER key, and a screen such as that shown in Fig. 6.3 (b) appears.

Touch Screen operation

- 8 Touching the “PAGE UP” or “PAGE DOWN” buttons at the top and bottom of the screens to page up and down in the Ladder Tree View.
- 9 Touch the screen on a particular level or subprogram to move the cursor.
- 10 Touch the screen in the same level or subprogram a second time and a screen such as that shown in Fig. 6.3 (b) appears.

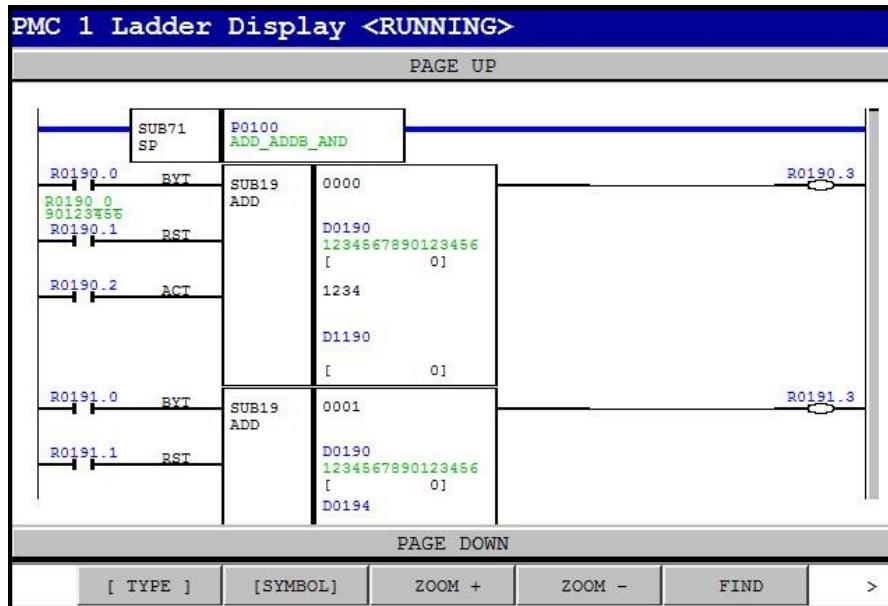


Fig. 6.3 (b) PMC sub-program

Touch Screen operation

- 11 Touching the “PAGE UP” or “PAGE DOWN” buttons at the top and bottom of the screens to page up and down in the Ladder Display.
- 12 Touch the top or bottom net in the display will decrement or increment the screen by one net.

⚠ CAUTION

If a specific level or sub-program is selected, the selected program is displayed, but it is possible to display all consecutive levels or sub programs by scrolling down.

Switching between address and symbol displays

By pressing F2 [SYMBOL] and selecting one from 1 to 4 below, it is possible to switch between address and symbol displays.

- | | |
|--------------------|---|
| 1 Address | - Displays only addresses above relays or coils. (Example: X1000.2) |
| 2 Symbol | - Displays only symbols above relays or coils. (Example: U13) |
| 3 Address & symbol | - Displays addresses above relays or coils and symbols below them. |
| 4 Symbol & address | - Displays symbols above relays or coils and addresses below them. |

Zooming IN and OUT

By pressing F3 ZOOM+ the screen will zoom in making the elements on the screen larger with less elements displayed on the screen.

By pressing F4 ZOOM- the screen will zoom out making the elements on the screen smaller with more elements displayed on the screen.

Address and function command search

- 1 Press F5 [FIND] and a screen such as that shown in Fig. 6.3 (c) appears.

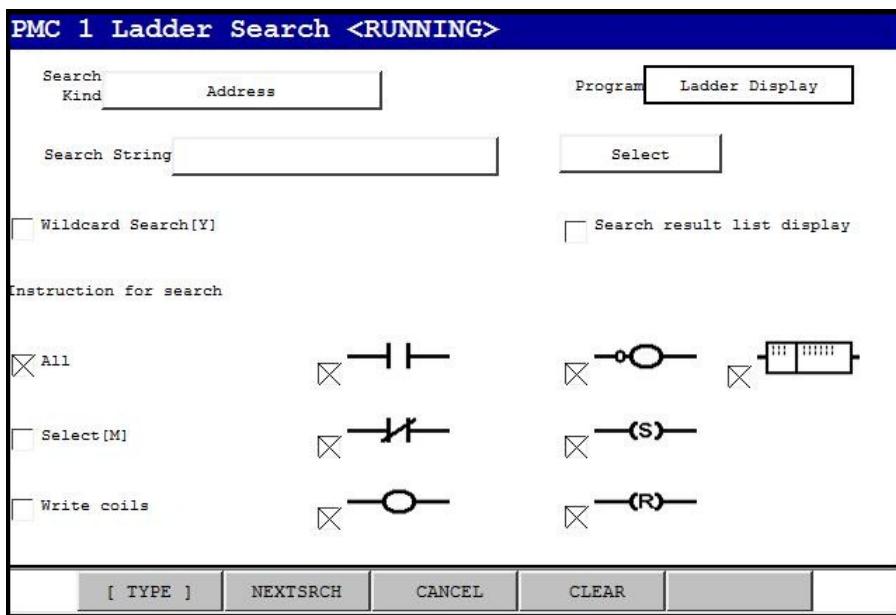


Fig. 6.3 (c) Search screen

The following explains how to use the menu.

- 1 Select Search kind and press the ENTER key, and a popup window appears. Select which of an address or function command to search for.
- 2 Move the cursor to the Search string field and press the ENTER key, and a popup keyboard appears. Input the character string to search for.

- 3 Use the Select button to specify a search start position. The search starts at the selected level or sub-program.
- 4 By selecting Wildcard search (a check mark is placed), it is possible to make a wildcard search, using a wildcard (*) in the string to search for (for example, F1008.*). Only one wildcard character can be used.
- 5 If a search is made by selecting Search result list display (a check mark is placed), a list of all nets containing the search string is created. If a check mark is not placed, a net containing the search string appears. Press F5 [FIND] to search for another net containing the search string.
- 6 If the cursor is moved to the location in which to specify a search range, the selections below can be used.
All - All relays, coils, and function commands
Select - Selected relays, coils, and function commands
Write coils - Coil only
- 7 Press F3 [CANCEL] or the [PREV] key to return to the PMC ladder display screen.
- 8 Press F4 [CLEAR] and the search string is deleted.
- 9 Press F2 [NEXTSRCH] to start a search and display the results.

Touch Screen operation

- 10 Touch any button or check box on the screen will select the item.

6.4 PMC EDIT FUNCTION

The PMC edit function enables editing of PMC ladder programs on teach pendant. It enables editing of relay, coil, and function instructions in all levels and all subprograms of the PMC ladder programs located on the controller.

NOTE

To use this function, PMC change mode option (A05B-2600-R652) is necessary.

It enables changing of addresses and all nets in a ladder program. The edit screen can be used from all three window frames of teach pendant, and can also be used via remote communication by accessing CGTP.HTM or ECHO.HTM. By protecting the edit function with a password, it is possible to make it unusable via remote communication. Like the PMC display function, this function enables address and symbol searching.

The PMC edit function does not permit the following:

- Change of symbol and comment
- Addition and deletion of nets
- Addition and deletion of relays, coils, and connection lines
- Changing of parameter #1 of the function instruction COM
- Changing of the parameters of the function instructions SP, SPE, JMP, JMPE, JMPB, LBL, JMPC, COME, CALL, and CALLU

Using the PMC edit function

- 1 Press the menu.
- 2 Select I/O.
- 3 Press F1 [TYPE].
- 4 Select PMC Display, and a screen such as that shown in Fig. 6.4 (a) appears.

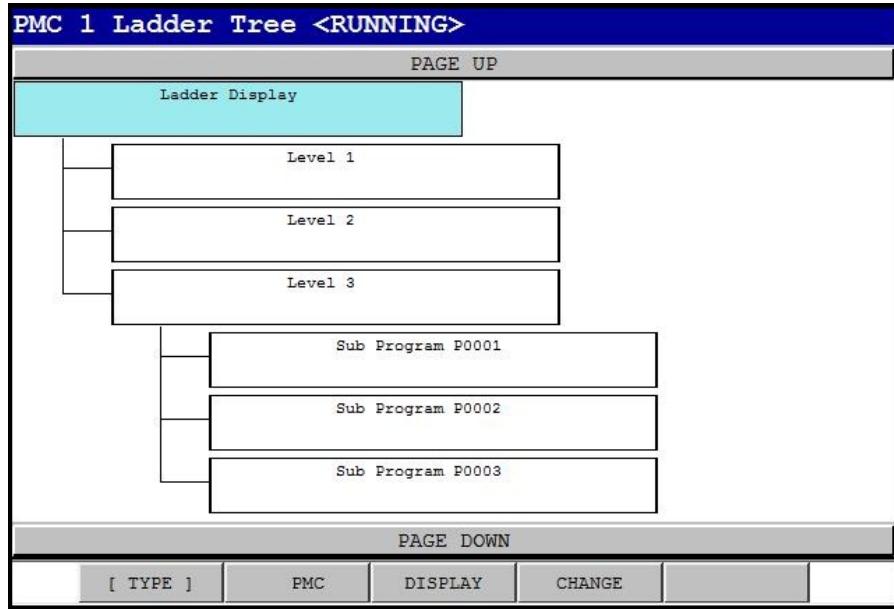


Fig. 6.4 (a) PMC ladder tree view

For example, when a ladder program at level 1 or 2 is selected, ladder components are displayed.

- 5 Use the PMC key to select the PMC path and a screen such as that shown in Fig. 6.4 (a) appears for the specific PMC path.
- 6 Use the ↑ or ↓ key to move the cursor to the level or subprogram to be displayed. Use SHIFT-↑ or SHIFT-↓ to page up and down in the Ladder Tree View.
- 7 Press the F4 [CHANGE] or ENTER key, and a screen such as that shown in Fig. 6.4 (b) appears.

Touch Screen operation

- 8 Touch the "PAGE UP" or "PAGE DOWN" buttons at the top and bottom of the screens to page up and down in the Ladder Tree View.
- 9 Touch the screen on a particular level or subprogram to move the cursor.

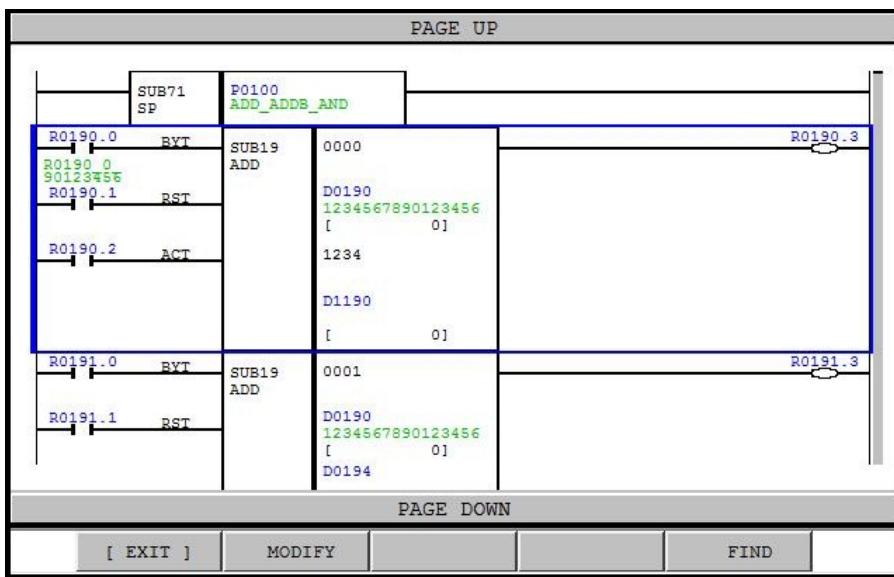


Fig. 6.4 (b) PMC edit screen

CAUTION

It is possible to start at a specific level or subprogram by selecting it, but the cursor passes through all ladder programs and as you move down.

- 10 The cursor changes to a blue one with a 2-pixel width, enclosing a net. The cursor can be used with the arrow keys.

Touch Screen operation

- 11 Touch the “PAGE UP” or “PAGE DOWN” buttons at the top and bottom of the screens to page up and down in the Ladder Editor.
 12 The cursor can be moved by touching any part of the screen.

⚠ CAUTION

To search for an address or a function instruction, refer to the section on address and function instruction search.

- 13 To edit a net, position the cursor on the net and press the F2 [MODIFY] or ENTER key. A screen such as that shown in Fig. 6.4 (c) appears.

Touch Screen operation

- 14 Touch the screen on a particular net to move the cursor. Touching in the same position on the screen will select the net to be edited.

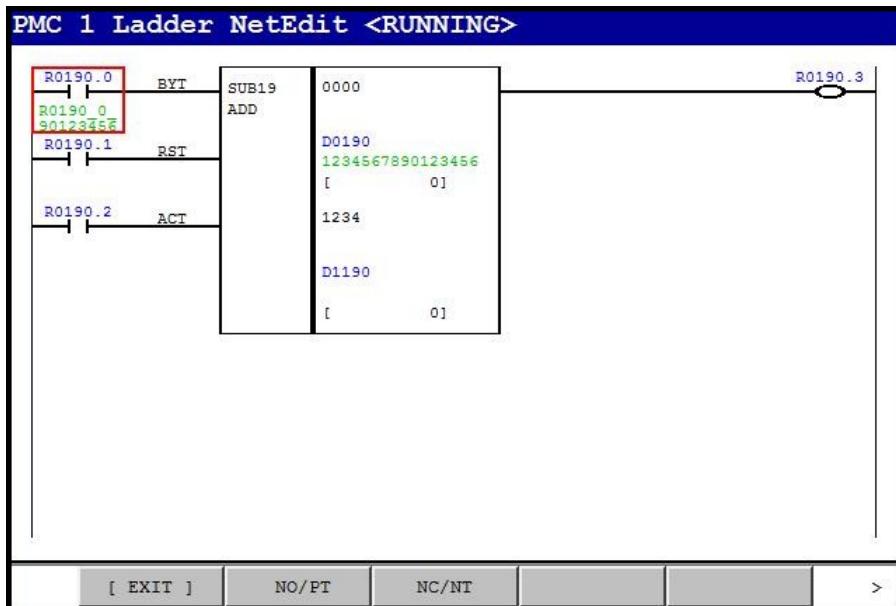


Fig. 6.4 (c) PMC ladder net editing

- 15 To set an element on contact A, press F2 [NO].
 16 To set an element on contact B, press F3 [NC].
 17 To modify a parameter of a function instruction, move the cursor to the upper left corner of the function instruction and press the ENTER key. The function instruction parameter screen appears. Use the arrow keys to position the cursor and press the ENTER key, so that parameters can be edited.
 18 When the cursor is at the rightmost position, the following can be performed:
 - Press F2 [WRT] to write a coil.
 - Press F3 [WRTNT] to write an inverted coil.
 - Press F3 [SET] to set a coil.
 - Press F4 [RESET] to reset a coil.
 19 To change a specific address or symbol, position the cursor on a relay and press the ENTER key. The PMC connection setup screen appears, enabling editing of addresses and symbols.
 20 When editing is finished, press F1 [EXIT] and select SUBMIT (which transfers the changes to the editor buffer). To cancel the changes, press F1 [EXIT] and select CANCEL. To continue editing, select CONTINUE.

- 21 When editing of all necessary nets is finished, press F1 [EXIT] on the PMC ladder edit screen and select SUBMIT. The PMC ladder tree screen reappears, and the changes made to the PMC ladder program are saved.
* In 7DC1 and 7DD0 software, if neither teach pendant E-STOP button nor operator panel E-STOP button is pressed at this time, the changed PMC ladder is not written to FROM, and the change is lost by power off/on.

Touch Screen operation

- 22 Touch the screen on a particular element to move the cursor. Touching in the same position on the screen will select the element to be edited.

7 STEP SEQUENCE FUNCTION

7.1 OVERVIEW

NOTE

To use the step sequence function, Multi-path integrated PMC option is necessary.

7.1.1 Step Sequence Method

The ladder method is most often used for programming the sequence control governed by a programmable controller. This method, shown in Fig.7.1.1(a), was derived from relay-panel control circuits. Since it has been in use for years, many sequence control engineers are already familiar with it. This method is also used in PMC sequence programming.

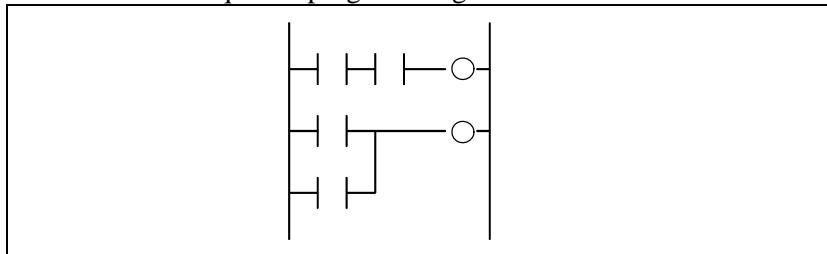


Fig.7.1.1(a) Ladder method

The greater the number of functions implemented by the PMC, the larger and the more complicated the sequence program becomes. A large-scale system requires a larger program and a greater number of processes, making it hard for the ladder method to control the overall process. This is because the ladder method does not describe the order of control. While the ladder method is suitable for describing partial control, it is hard to apply it to the description of the flow of control overall.

To overcome this problem, structured programming has been introduced into sequence control. A PMC that supports the subprogram function enables the use of modular programs. As shown in Fig.7.1.1(b), a large-scale program is divided into subprograms for each function, simplifying the unit of processing. Since the programmer determines how to divide the main program into subprograms and the control flow used to call the subprograms, however, the programs are not necessarily easy-to-understand by other programmers.

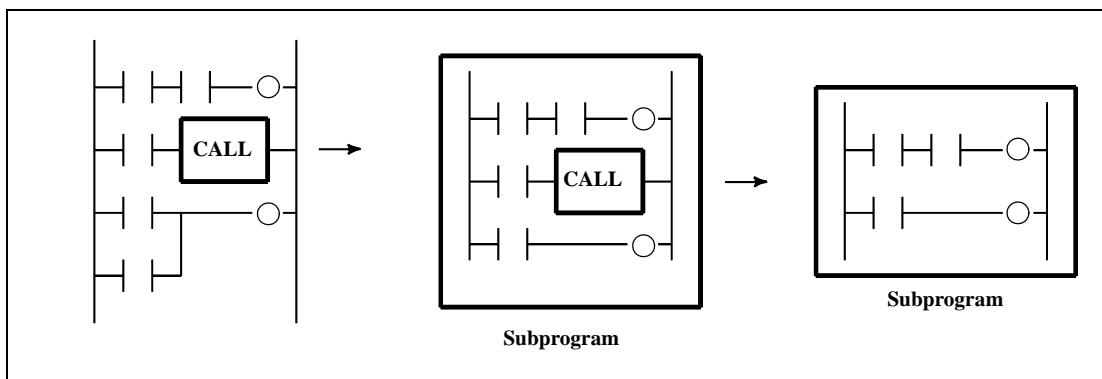


Fig.7.1.1(b) Module method

Given these conditions, a step sequence method has been created to describe programs structurally. It is well-suited to the control of entire processes and provides an easy-to-understand visualized flow of the

process. The step sequence programming features the direct representation of the control flow on a flow chart, as shown in Fig.7.1.1(c).

Each block of processing is described as a subprogram, using the ladder method. The entire program is then created by combining these subprograms.

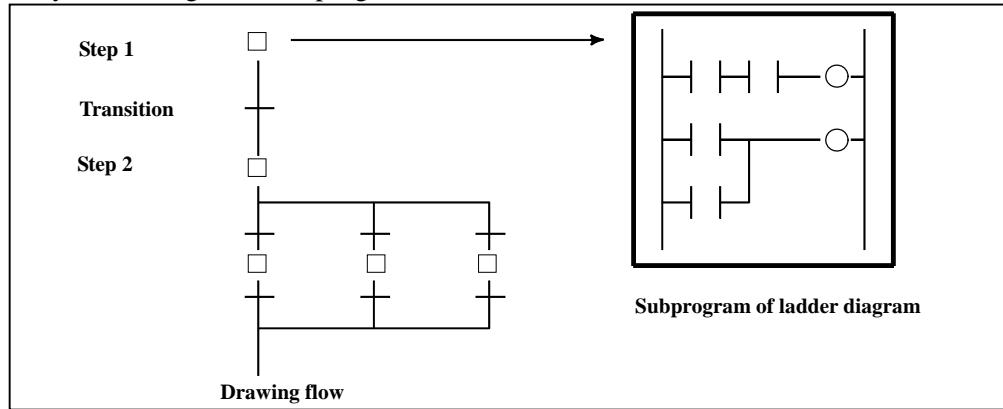


Fig.7.1.1(c) Step sequence method

The step sequence method has the following features:

(1) Increased programming efficiency

- Since the flow of processes can be programmed directly, simple, correct programming is enabled, reducing the time required for programming.
- Even for complicated control, programming proceeds from the main flow to detailed flow in each process, creating a structured, top-down program, which is easy-to-understand by persons other than the original creator.
- Structured modules can be used again easily.

(2) Easy debugging and maintenance

- Graphical display enables the operator to easily understand the execution state of a program visually.
- Erroneous steps in a program can be found easily.
- A part of a program can be easily modified.

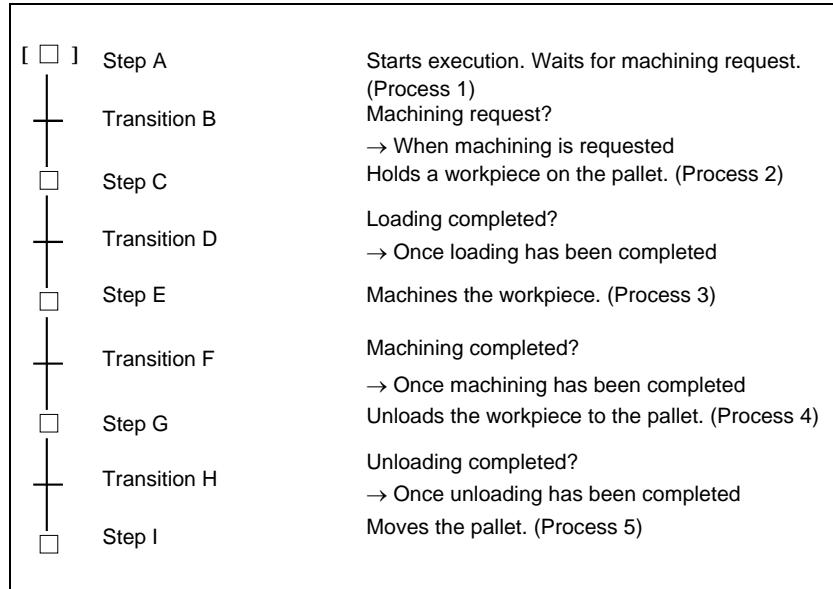
(3) High-speed program

- Since only the subprograms required for a certain process are executed, the cycle time is reduced.

(4) Transition from ladder programs

- Since steps and transitions consist of conventional ladder programs, conventional ladder programs can be converted to new step sequence programs, without discarding ladder-program resources.

In step sequence programming, a sequence control program is divided into two types of subprograms, steps and transitions. Steps describe processes. Transitions connect steps and determine whether the transition conditions from one step to another evaluate true. As shown in Fig.7.1.1(d), a step sequence program is described using graphical symbols.

**Fig.7.1.1(d) Example of machining the workpiece**

As shown in this example, the program flow from process 1 through process 5 is expressed visually. Detailed programs related to the movements performed as part of each process, and the signals used for determining whether transition conditions for proceeding to the next step are satisfied, are not described here. To program complicated control flows, many other functions are supported, such as divergence, jump, and nesting functions. The details of these functions are described later.

Step sequence programming is suitable for creating programs which control processes sequentially. Programs used for controlling a unit which operates according to a certain sequence, such as a loader, ATC, and other peripheral units, are best suited to step sequence programming. For programs which control units with no particular sequence, such as that of the operator's panel which is always monitoring the emergency stop signal or mode signals, however, are not well-suited to step sequence programming. The PMC supports the advantages of both methods, ladder and step sequence programming, by calling subprograms written according to a step sequence and those written as a ladder, from the main program.

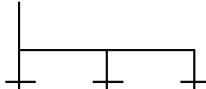
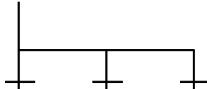
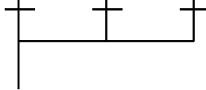
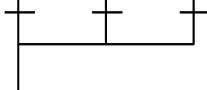
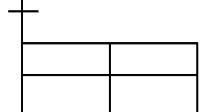
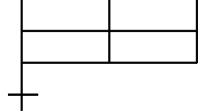
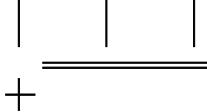
7.1.2 Graphical Symbols

This manual uses the graphical symbols listed in Table 7.1.2 to describe step sequence flowcharts. Depending on the character font being used, the actually displayed symbols may differ slightly from those listed here.

These graphical symbols are described in the subsequent chapters.

Table 7.1.2 List of graphical symbols

Contents	Display of programming manual	FANUC LADDER-III for Robot
Step	 □ Sn 	 □ Sn
Initial Step	 [□] Sn 	 [□] Sn
Transition	+ Pn	+ Pn

Contents	Display of programming manual	FANUC LADDER-III for Robot
Divergence of Selective Sequence		
Convergence of Selective Sequence		
Divergence of Simultaneous Sequence		
Convergence of Simultaneous Sequence		
Jump		
Label		
Block Step		
Initial Block Step		
End of Block Step		

7.1.3 Editing and Debugging Step Sequence Programs

The personal computer programmer "FANUC LADDER-III for Robot" is used to edit a step sequence program.

For details of transferring and writing a step sequence program to the PMC, see Subsection 5, " FANUC LADDER-III for Robot Programming".

A step sequence program is executed and debugged on the CNC.

For details of debugging a step sequence program, see Subsection 6, "Teach Pendant Operation".

Table 7.1.3 indicates the step sequence functions usable on FANUC LADDER-III for Robot and the Robot controller.

Table 7.1.3 Step sequence functions

	Robot controller	FANUC LADDER-III for Robot
Display and edit of a program		
• Display of subprogram list	○	○
• Create a new subprogram		○
• Delete a subprogram		○
• Edit a subprogram of Step Sequence form		○
• Edit a subprogram of ladder diagram	○	○
Input and output		
• Input and output with a memory card/USB memory	○	○
• Write to a Flash ROM	○	○
Execution of program		
• Execution of a ladder diagram	○	○
Diagnosis and debugging		
• Diagnosis of Step Sequence program		
• Diagnosis of a ladder diagram	○	○

7.2 STEP SEQUENCE BASICS

7.2.1 Terminology

A step sequence program is created using a variety of graphical symbols, as shown in Fig.7.2.1(a). The main terms used in the step sequence are described below.

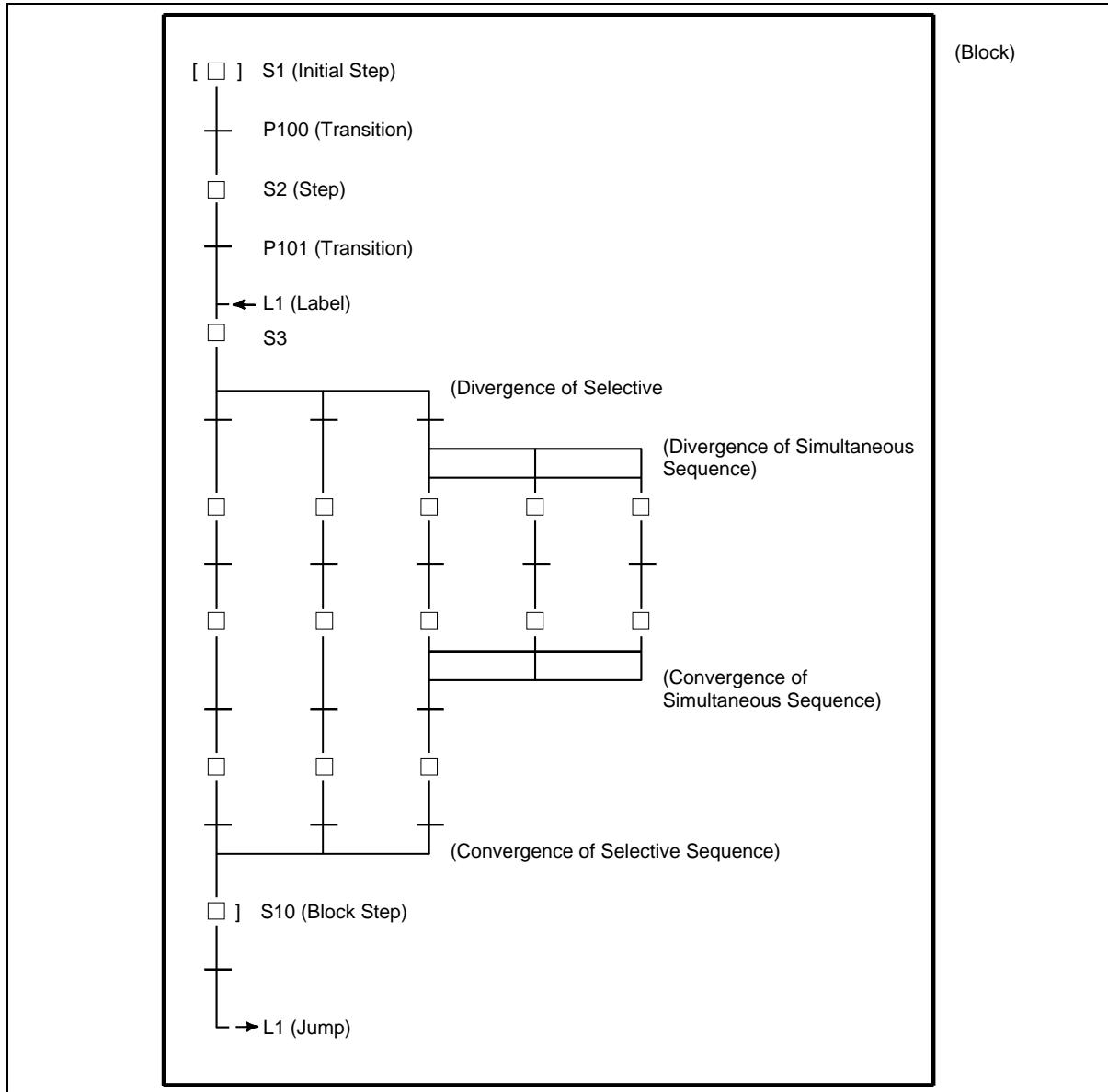


Fig.7.2.1(a) Step sequence elements

(1) Step



A step indicates a process, which is the basic processing unit in a step sequence program. In a step, specify the S address (Sn), which is a step number, and P address (Pm), which indicates a subprogram (action program) specifying the details of processing in each step.

(2) Step state transition

When a step sequence program is executed, the process proceeds as program processing advances, the state of each step changes accordingly. Each step can assume any of the logical states listed in Table 7.2.1, its state changes as shown in Fig.7.2.1(b). Activation refers to the changing of a step from the inactive state to the active state.

Inactivation refers to the changing of a step from the active state to the inactive state.

Table 7.2.1 Step state

State		Processing	Display
Active	Execution	Activated step. The action program (subprogram) is being executed.	 ■ Sn
Inactive	Transition to halt	Transition from execution to halt. The action program (subprogram) is executed once only, then the step automatically transits to halt.	
	Halt	Not activated state. The action program (subprogram) has not yet been executed.	 □ Sn

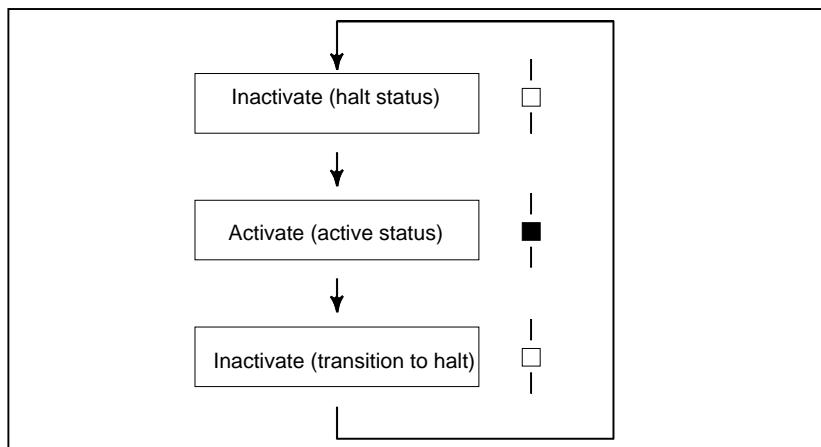
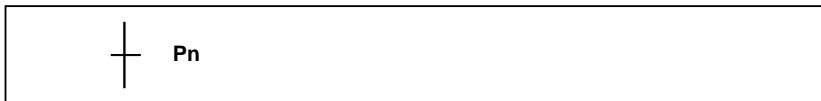


Fig.7.2.1(b) Step state transition

(3) Transition



A transition denotes the transition conditions. When these evaluate true, the step of the corresponding state changes from the inactive to active state or vice the reverse. Specify the P address (Pn), which indicates a subprogram describing the transition conditions in detail.

As shown in Fig.7.2.1(c), step S2 changes its state from inactive to active when the conditions described in transition P10 evaluate true, while step S2 changes its state from active to inactive when the conditions described in transition P20 evaluate true.

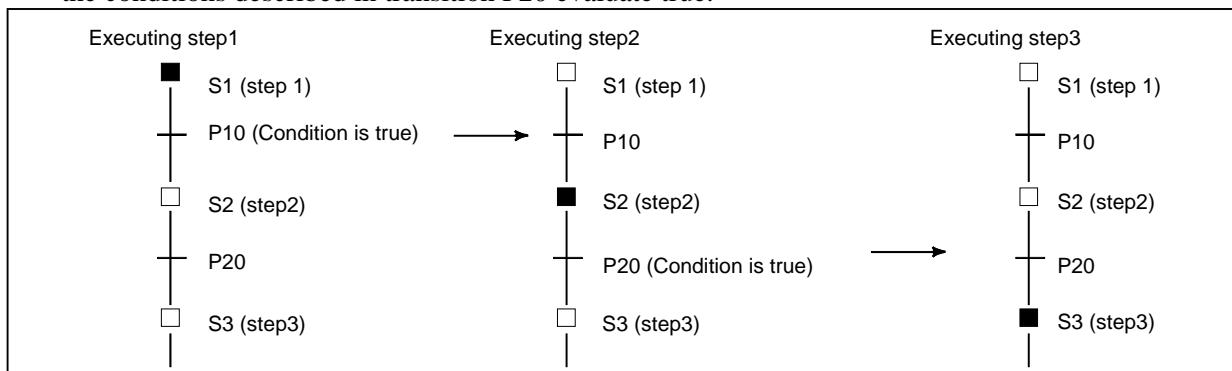


Fig.7.2.1(c) Transition of step state by the transition

Note that the step immediately before a transition must be active in order to switch the next step from inactive to active when the conditions specified in the transition evaluate true. As shown in Fig.7.2.1(d), step S3 does not change to the active state, even when transition P20 evaluates true, if step S1 is active and step S2 is inactive. An active state passes from a certain step to the next step

when the corresponding transition conditions evaluate true, the execution of the step sequence program advancing one step.

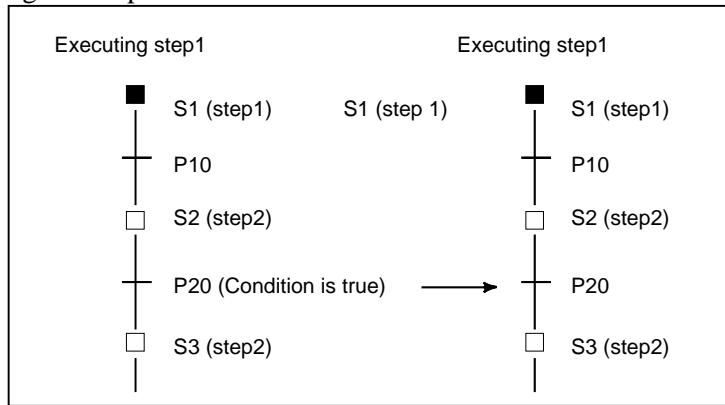
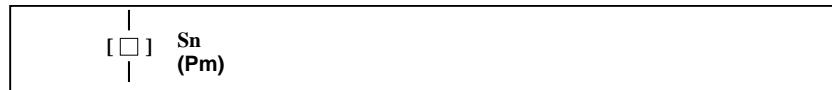


Fig.7.2.1(d) Transition of step state by transition

(4) Initial Step



While a normal step can be activated by a transition, the initial step is activated automatically when execution of the program starts, as shown in Fig.7.2.1(e).

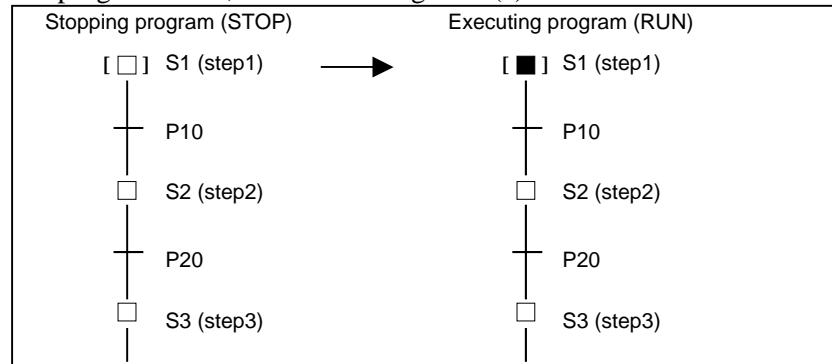


Fig.7.2.1(e) Activate of initial step

Although the initial step, which is usually executed first, is often placed at the top of a program, it can also be specified at some point within a program. It is always activated first. After being deactivated once, it can be subsequently be activated again. In this case, it acts in the same way as a normal step.

(5) Divergence and Convergence of Selective Sequence

To describe a complicated sequence, selective sequences can be used.

A selective sequence offers multiple choices, from among which the condition becomes true first activates the corresponding step, as shown in Fig.7.2.1(f). The divergent paths join to generate the main sequence.

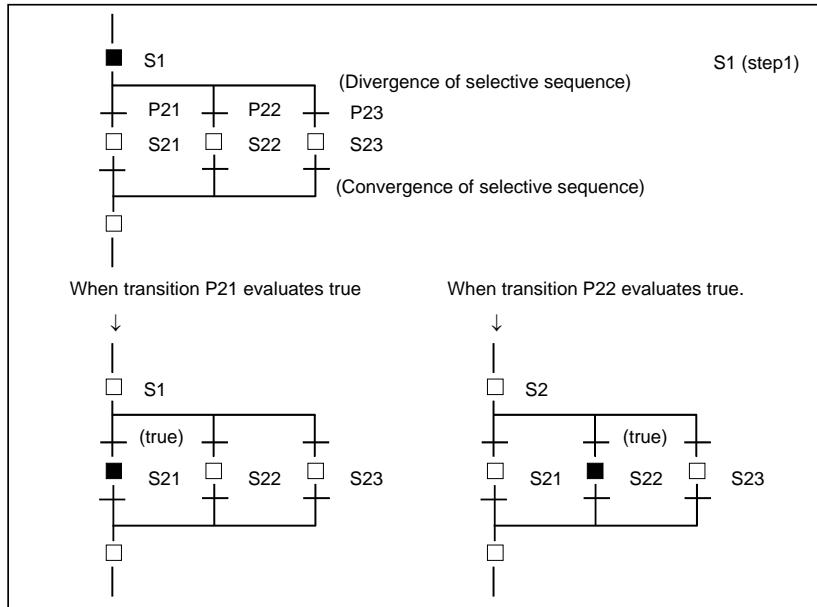


Fig.7.2.1(f) Selective sequence

(6) Divergence and Convergence of Simultaneous Sequence

A Simultaneous sequence can be used to execute multiple processes simultaneously. In a Simultaneous sequence, as shown in Fig.7.2.1(g), one transition activates multiple steps. The activated multiple steps are executed independently. Once all steps along the multiple paths have been completed, the divergent paths join to generate the main sequence.

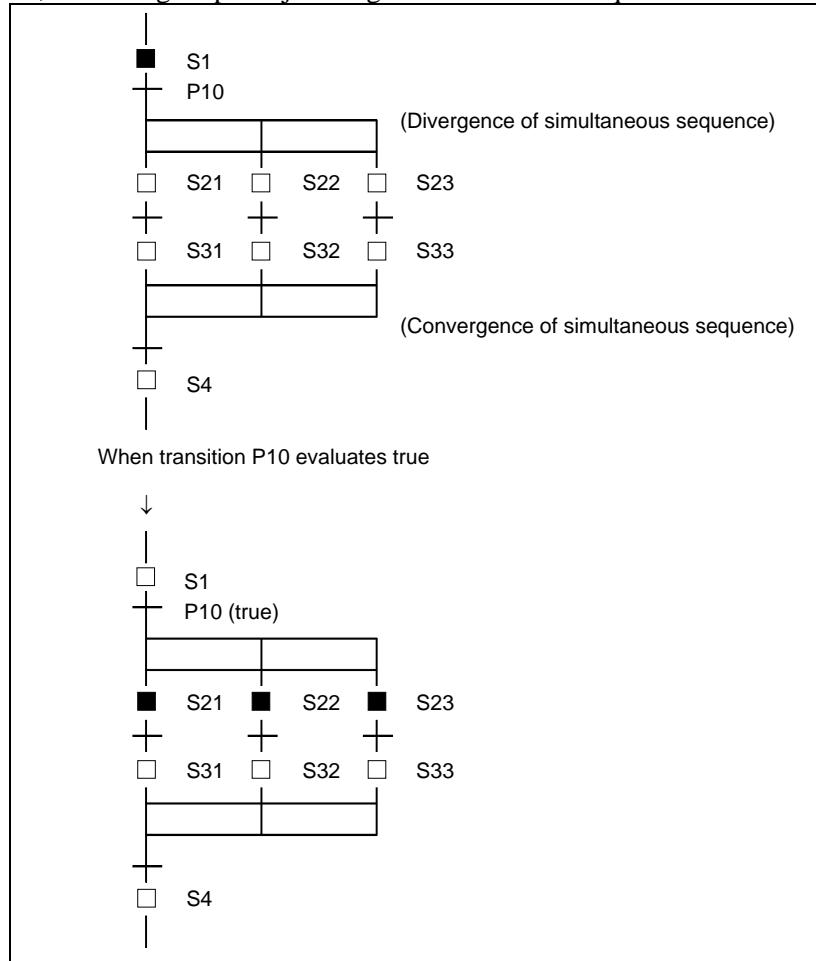


Fig.7.2.1(g) Simultaneous sequence

(7) Jump and Label

The jump function is used to describe a non-serial sequence, such as a repeated loop. As shown in Fig.7.2.1(h), when a jump designation is activated, the sequence jumps to the step having the corresponding jump destination label, after which that step is activated. To specify a label number, the L address is used in the same way as a jump instruction in ladder programming. A jump can be made to a previous or subsequent step.

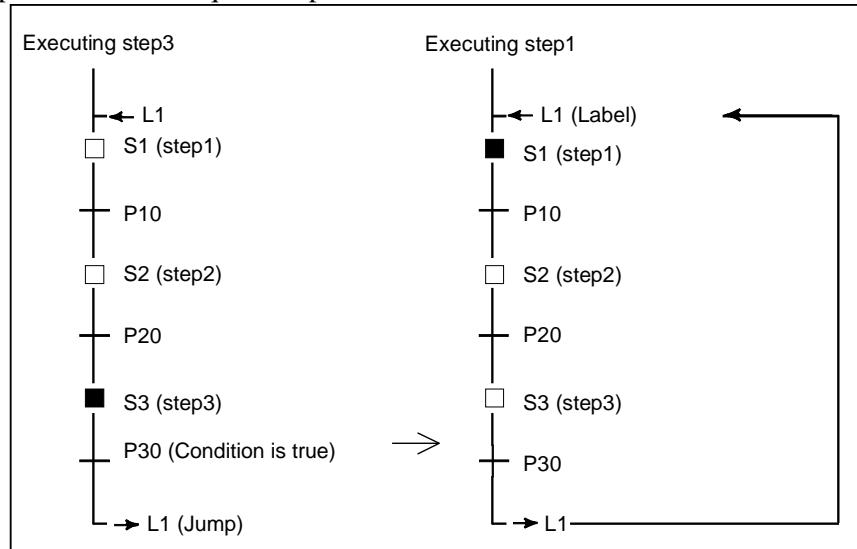


Fig.7.2.1(h) Jump and Label

(8) Block

A block refers to a group of consecutive steps and transitions. A block can be a step sequence program. The more complicated the sequence becomes, the larger and more complex the block is. A program can be divided into multiple blocks in the same way as for subprograms in ladder programming, based on the concept of modular programming. Each block is identified by a P address, which corresponds to the subprogram number in ladder programming.

A block is executed as the main program in a step sequence, or called from another step sequence program as a subprogram.

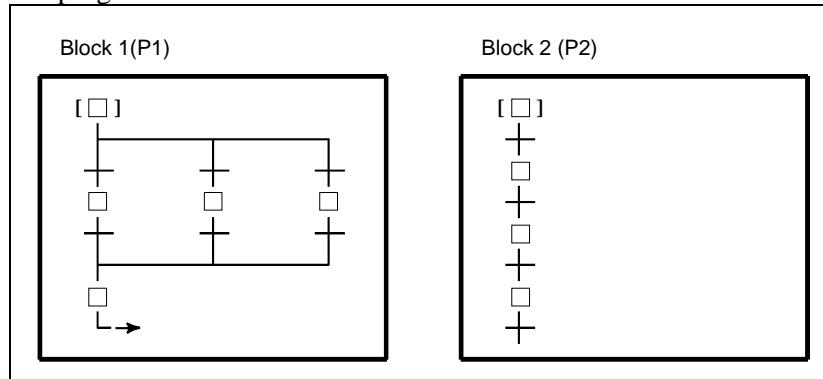


Fig.7.2.1(i) Block

(9) Calling block

To execute a block as the main program in a step sequence, call the block with the CALLU (SUB 66) or CALL (SUB65) instruction in the same way as for ladder subprogram calling from the second level ladder program.

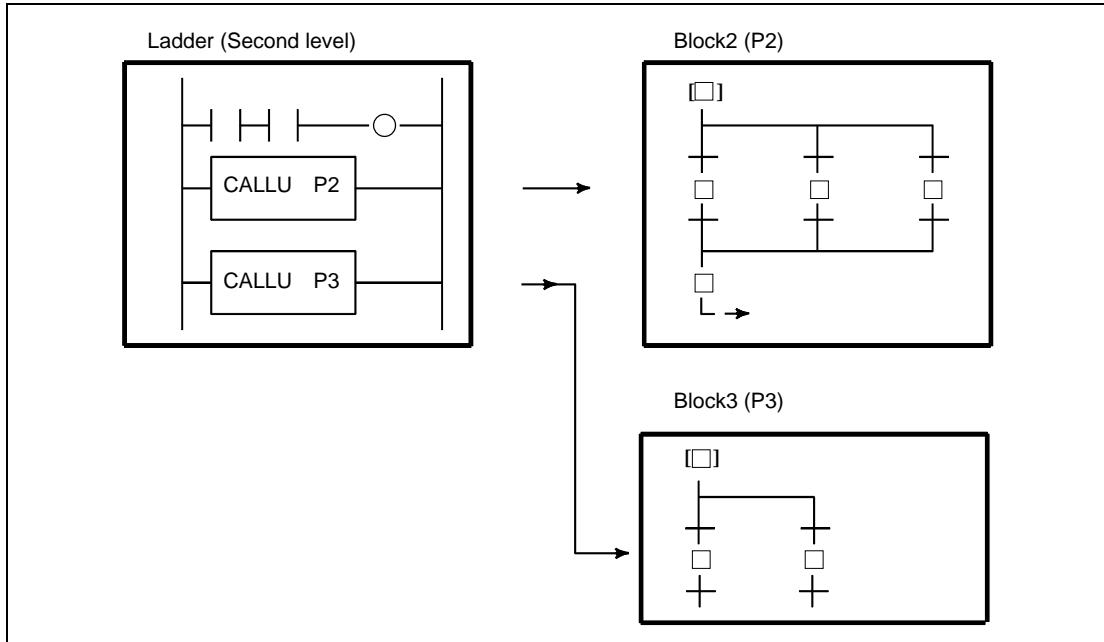


Fig.7.2.1(j) Calling block

(10) Block step (calling step sequence program)



To call a block from the step sequence program as a subprogram, specify a block step in the step sequence program which calls the block, as shown in Fig.7.2.1 (k). This is called bloc nesting.

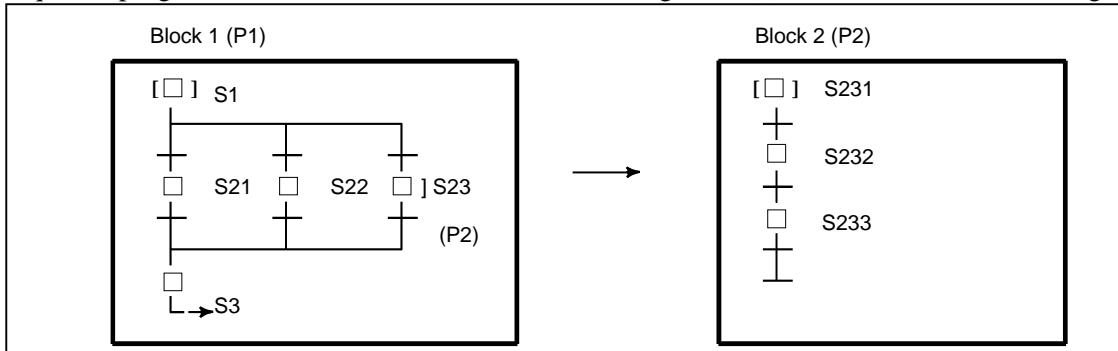


Fig.7.2.1(k) Block nesting

The program shown in Fig.7.2.1(k) is equivalent to in Fig.7.2.1(l) which does not use a block step.

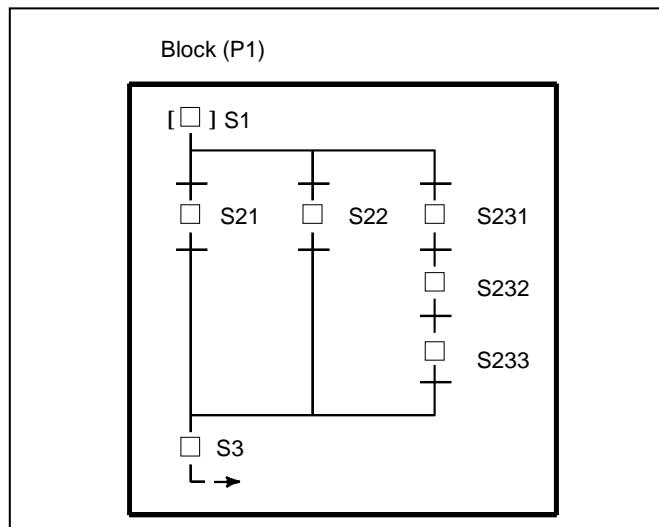


Fig.7.2.1(l) Program without block step

(11) End of block step



Use an end block step to terminate nested-block-step calling and to return to the calling sequence.

7.2.2 Execution of Step Sequence

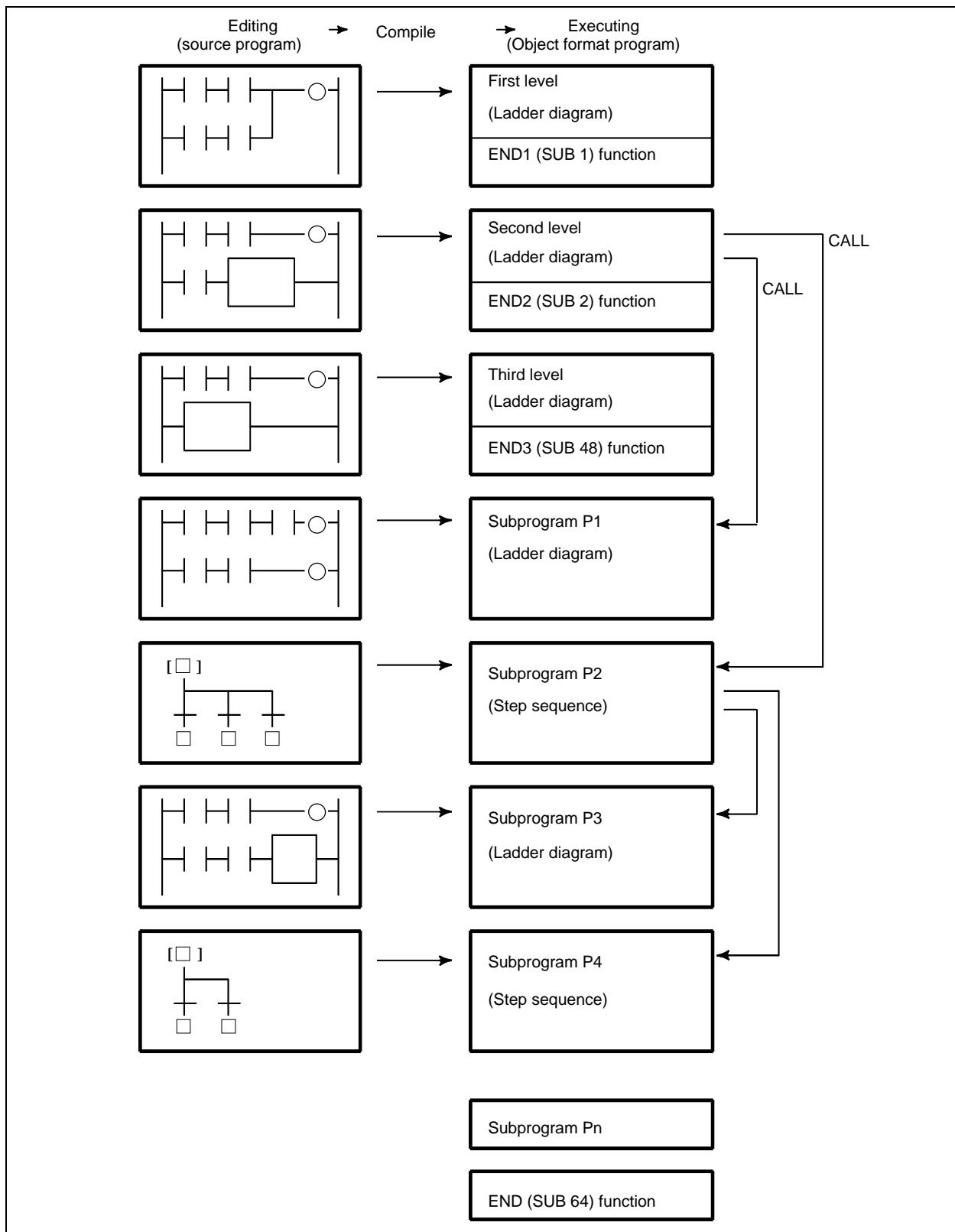


Fig.7.2.2(a) Structure of program

In the step sequence method, a program is created (edited) in units of subprograms. The edited source program is compiled and converted to an executable ROM-format program, then linked, as shown in Fig.7.2.2(a).

A ROM-format program is a kind of a modular program, created using conventional subprograms. A step sequence block is also a type of a subprogram. Step sequence blocks are linked to the end of the first level to third level ladder programs, together with other ladder subprograms.

In the same way as in the ladder method, a program is activated at certain intervals.

Refer to section 1.4.3 "Processing Priority (1st Level, 2nd Level, and 3rd Level)" for details

All subprograms, created using either the ladder or step sequence method, are called from the second level ladder. Hence, the execution time of the second level ladder includes those of ladder subprograms, step sequence programs (blocks), steps, and transitions. Since only the activated step and the transition which checks the transition condition from the step to the next step are executed in a step sequence program, the second level ladder is executed much more frequently than may be expected from the total number of steps.

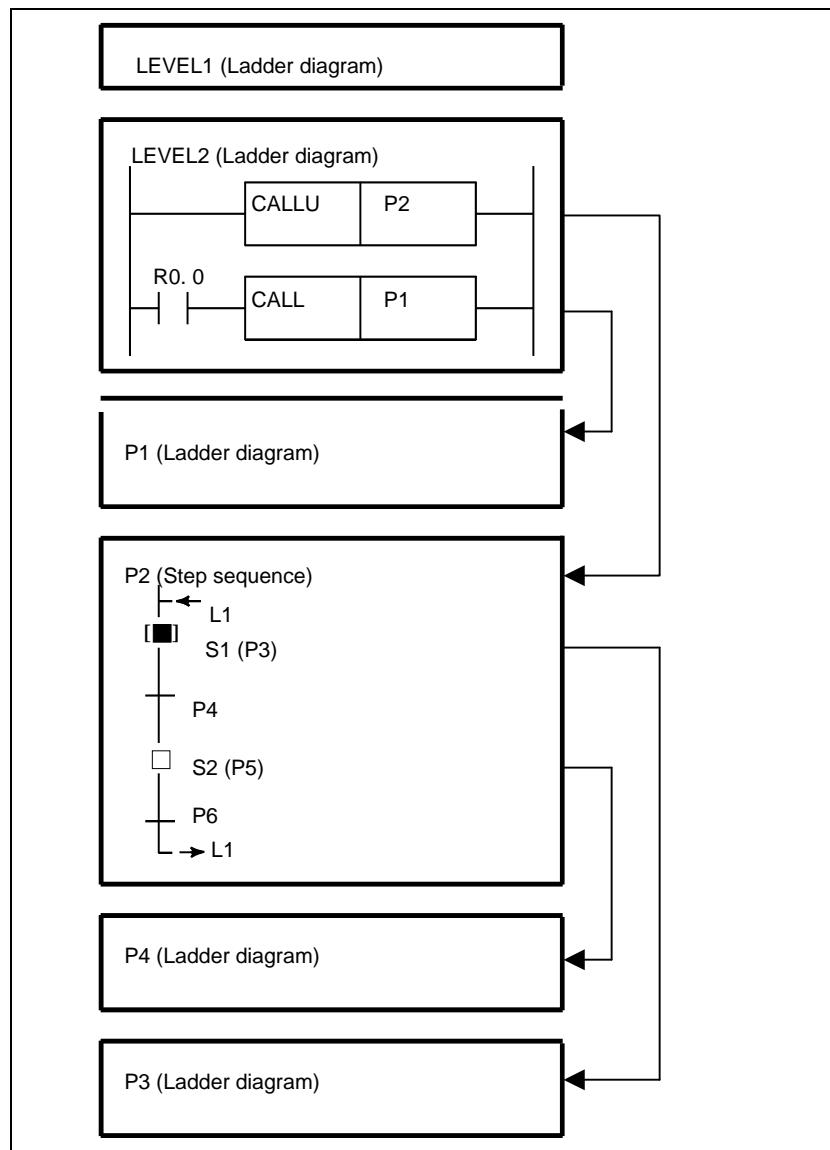


Fig.7.2.2(b) Execution of step sequence

In the step sequence program shown in Fig.7.2.2(b), when step S1 is activated, subprograms are executed according to the timing illustrated in Fig.7.2.2(c).

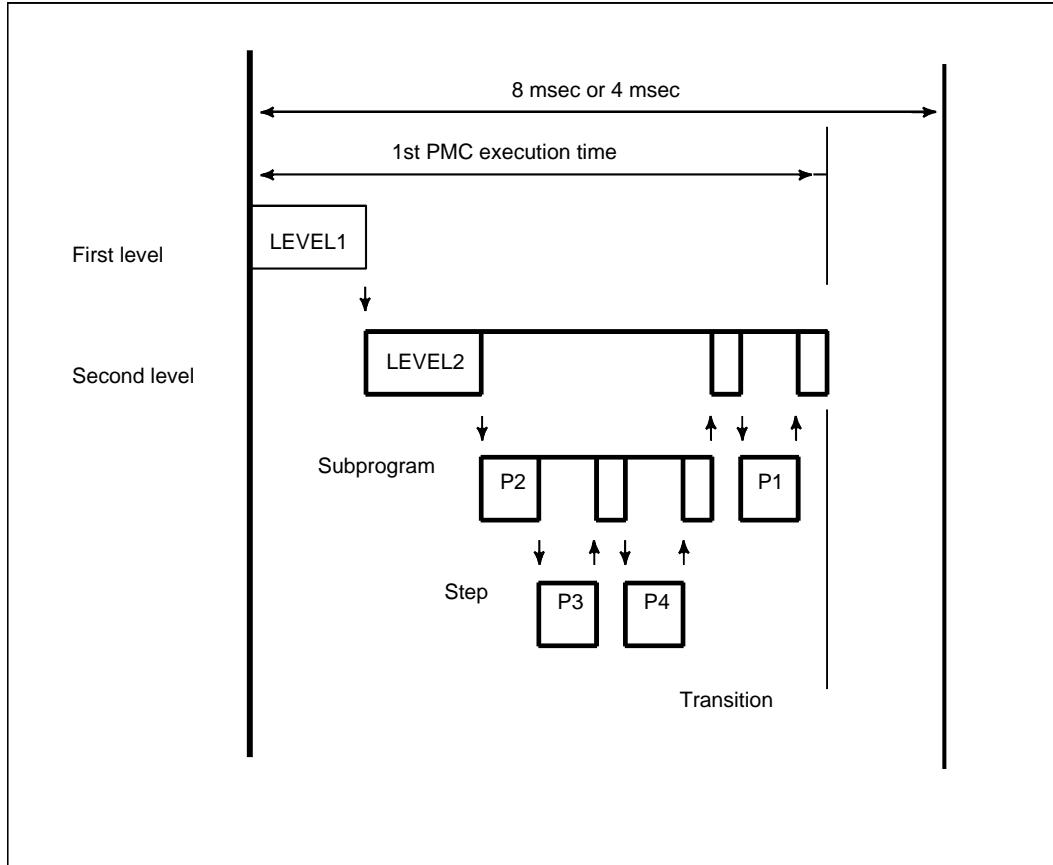


Fig.7.2.2(c) Timing of execution of step sequence program

In this case, step sequence program P2, step P3, transition P4, and ladder subprogram P1 are executed. Step P5 and transition P6 are not executed.

7.3 CONFIGURATION AND OPERATION OF STEP-SEQUENCE PROGRAMS

7.3.1 Step

A step is a unit of processing in a program.

Display

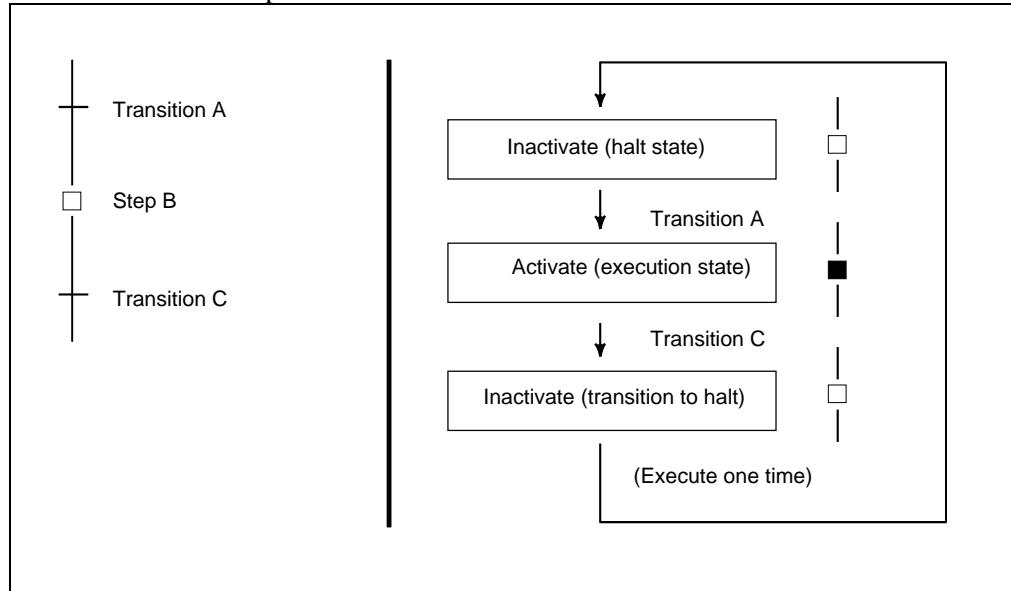


Contents

- Define a step number (Sn), necessary for controlling execution, and subprogram number (Pm) specifying actual processing, for a step.
- Assign a step number to a step.
- The same step number cannot be used twice in a program.
- A step has three logical states: the execution, transition to halt, and halt states. The execution state is also called the active state. The transition to halt and halt states are collectively called the inactive state.

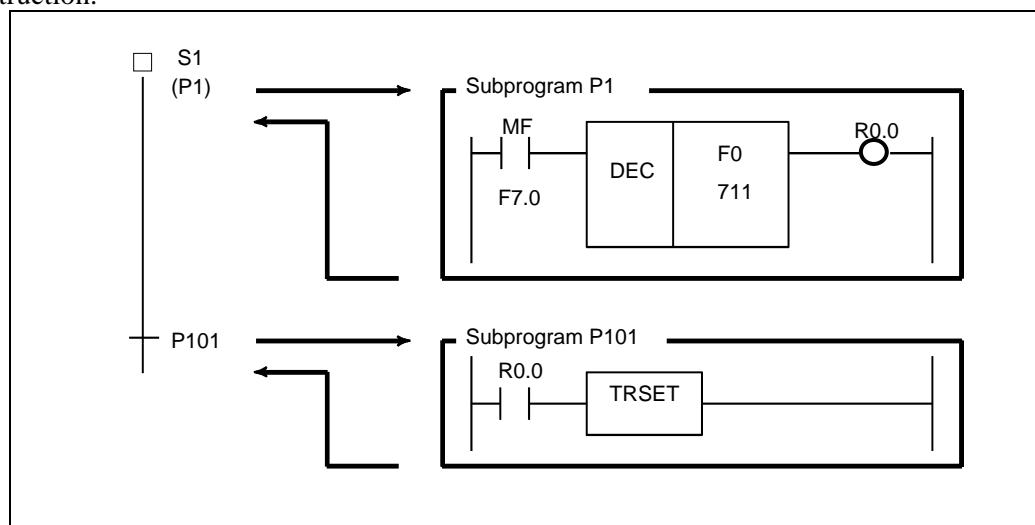
State		Contents of operation	Display	Sn.0
Activate	Execution	Activated step. The action program (subprogram) is being executed.	■ Sn	1
Inactivate	Transition to halt	Transition from execution to halt. The action program (subprogram) is executed once only, then the step automatically transits to halt.	□ Sn	0
	Stop	Not activated state. The action program (subprogram) has not yet been executed.	□ Sn	0

Example) State transition of Step B



Example

After the M7 code is decoded, control is transferred to the next step using a DEC functional instruction.



7.3.2 Initial Step

An initial step is automatically activated when execution of the program starts. Once it has been activated, it operates in the same way as a normal step. The program can be returned to this step through other steps.

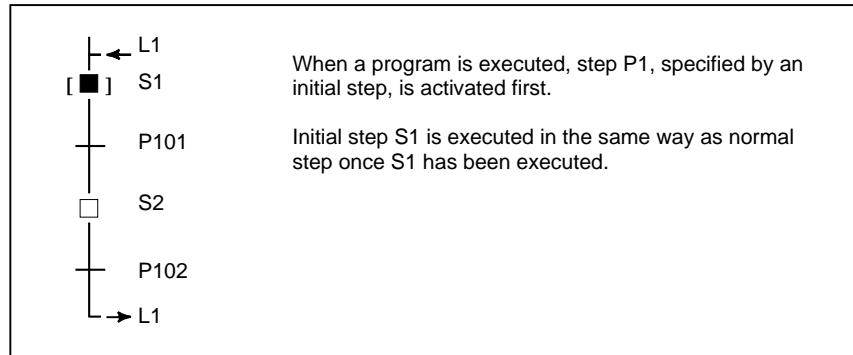
Display



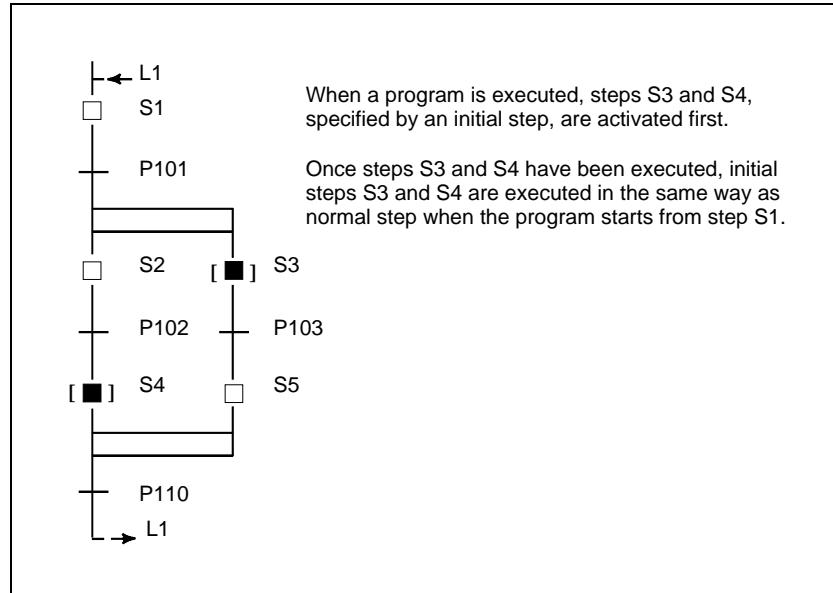
Contents

- Define a step number (Sn), necessary for controlling execution, and subprogram number (Pm) specifying the actual processing, for an initial step.
- All initial steps are activated when the other steps are not activated.
- Each block must contain at least one initial step. No limit is applied to the number of initial steps contained in a block.
- A block having no initial step cannot be executed if called.
- Assign a step number to an initial step.
- The same step number cannot be used more than once in a program.
- In parallel branch, one initial step is required for each path. (See example 2.)

Example 1



Example 2



7.3.3 Transition

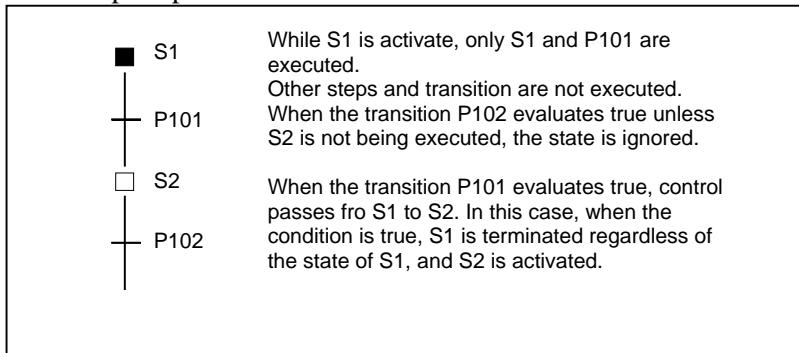
A transition specifies the conditions governing the transition from the step to the next step.

Display



Contents

- Only one transition is required between steps.
- Transition between steps is performed as described below.



- When a signal is set to 1 in a transition, it remains the state even if the control is transferred to the subsequent step. To set the signal to 0, use another subprogram to do so.

Example

Refer an example described on the Step function (Sub sec. 10.3.1).

7.3.4 Divergence of Selective Sequence

A selective sequence branches to two or more sequences. When the transition evaluates true, the corresponding step is activated.

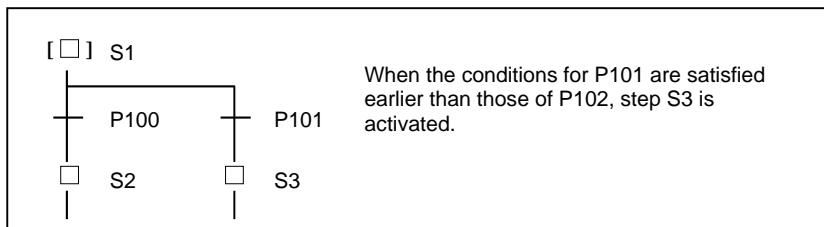
Display



Contents

- Transitions are placed after a divergence of selective sequence.
- The step connected to the transition for which the conditions are true is first activated.
- When the conditions for any transition are true simultaneously, the leftmost step is activated.
- A selective sequence can create up to 16 paths.

Example



7.3.5 Convergence of Selective Sequence

It combines two or more divergent paths to the main sequence.

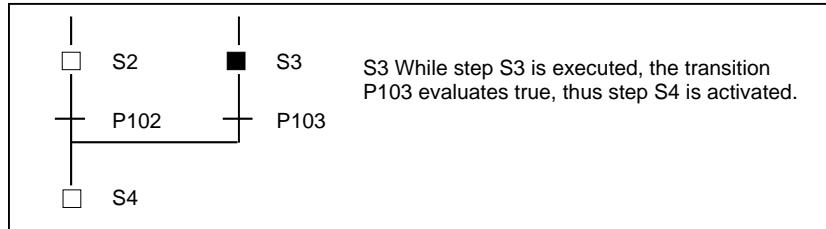
Display



Contents

The number of divergent paths must match that of the convergent paths.

Example



7.3.6 Divergence of Simultaneous Sequence

A simultaneous sequence branches to two or more sequences, and all steps are activated simultaneously.

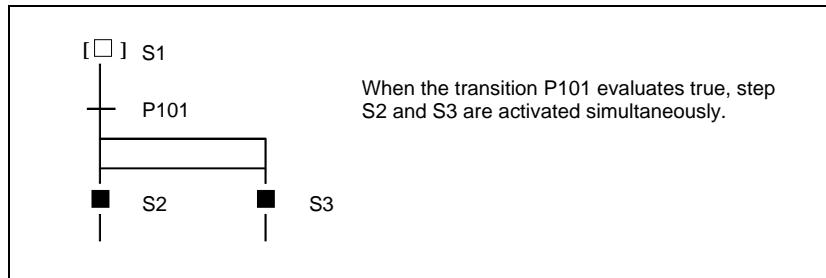
Display



Contents

- A transition must be placed before a divergence of simultaneous sequence.
- All branched steps are activated simultaneously, then executed.
- A simultaneous sequence can create up to 32 paths.

Example



7.3.7 Convergence of Simultaneous Sequence

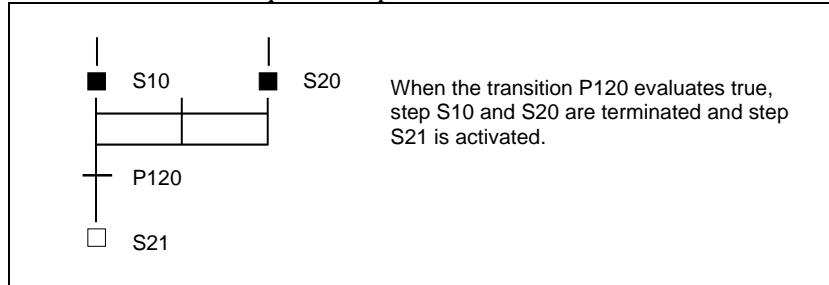
It combines two or more divergent paths to the main sequence.

Display



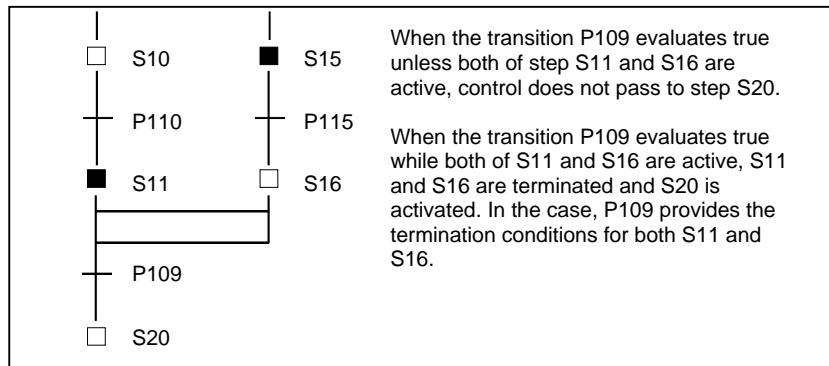
Contents

- A convergence of simultaneous sequence is processed as follows.

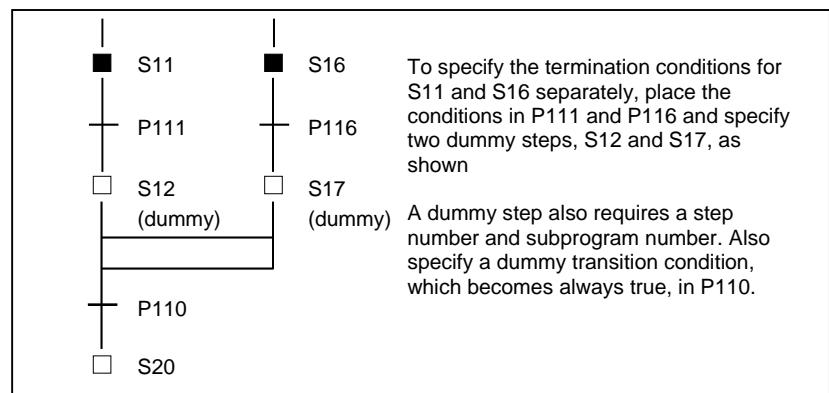


- Wait processing is processed as follows.

Case 1)



Case 2)



7.3.8 Jump

A jump controls the execution of steps non-sequentially, together with a transition.

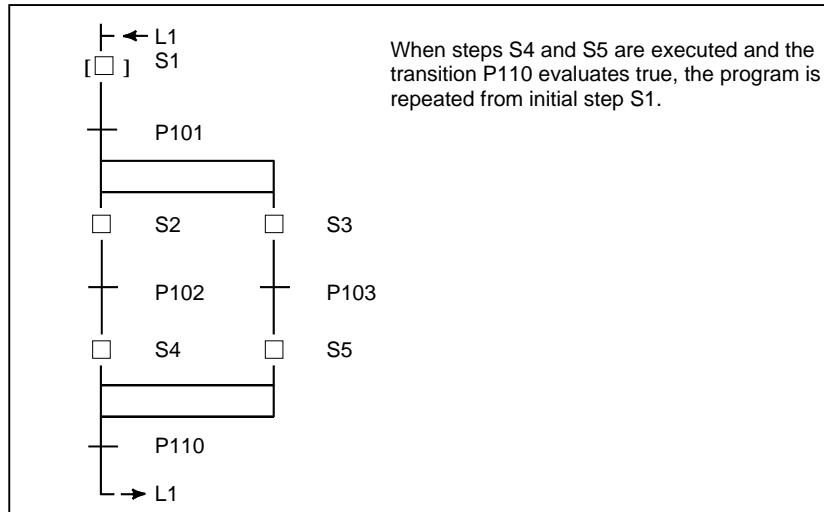
Display



Contents

- Specify a jump destination label (Ln).
- The step to which control is transferred (jumped) is activated.
- The jump destination must be within the same program.
- A jump cannot be performed from outside a simultaneous sequence to within the simultaneous sequence, or from within a simultaneous sequence to outside.
- A jump cannot be performed between parallel-branched paths.

Example



7.3.9 Label

A label specifies the jump destination.

Display



Contents

- Specify the jump destination label (Ln).

Example

Refer to an example described on the jump function (Subsec. 10.3.8).

7.3.10 Block Step

A block step specifies the step sequence subprogram to be executed.

Display



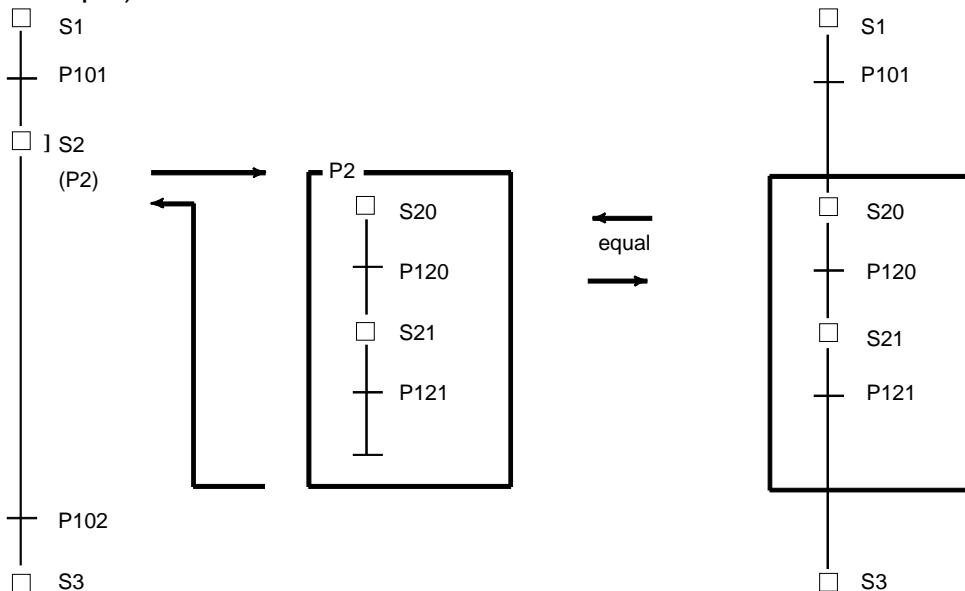
Contents

Define a step number (Sn), which controls the execution of a block step, and a subprogram (Pm) specifying the actual process, for a block step.

CAUTION

- 1 Assign a step number to a block step.
- 2 The same step number cannot be used twice in a program.
- 3 A transition must be placed after a block step.

Example)



- 4 Transition P102 cannot be omitted due to the syntax of the step sequence method. Specify a dummy transition, which becomes always true, for transition P102.
- 5 Transition P121 must specify the transition condition for the termination of the step S21.
- 6 When the conditions of transitions P102 and P121 are switched, step S21 will not be correctly executed.

7.3.11 Initial Block Step

This is an initial step on the block step.

Display



Contents

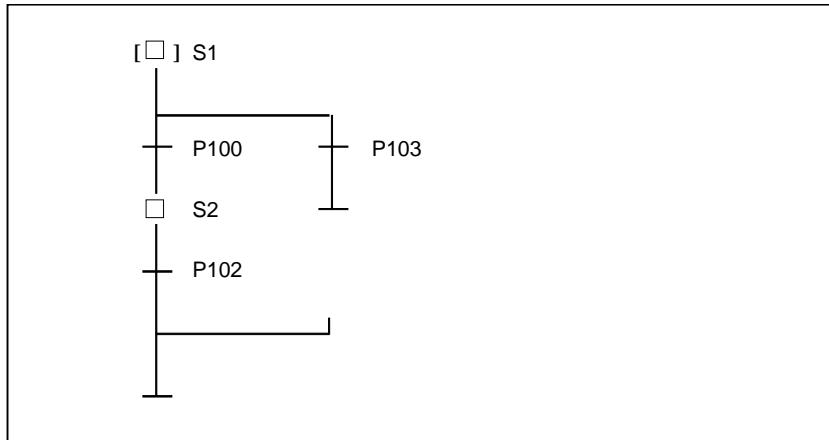
- Define a step number (Sn), necessary for controlling execution, and subprogram number (Pm) specifying the actual processing, for an initial step.
- This step has the same function and graphical symbol as an initial step.

7.3.12 End Of Block Step

This terminates a block step.

Display**Contents**

- Use this step to terminate a block step.
- Each block requires at least one end block step. No limit is applied to the number of end block steps.

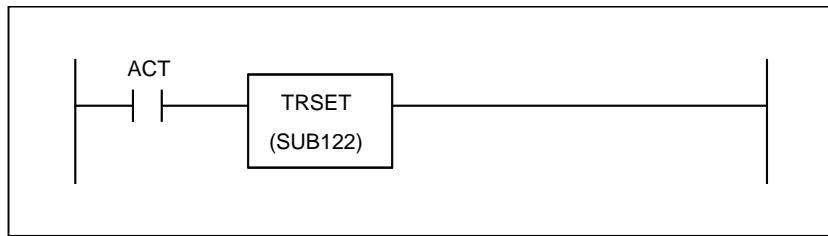
Example**7.4 EXTENDED LADDER INSTRUCTIONS**

To enable the specification of steps and transitions, the components of a step sequence program, by means of the ladder method, the following signals and functional instructions are provided. These signals and instructions can only be used in subprograms in which step sequence step and transitions are specified.

7.4.1 Functional Instruction TRSET**Function**

This instruction describes that the conditions for a transition have been true.

This instruction is used in a subprogram which is called from a transition.

Format

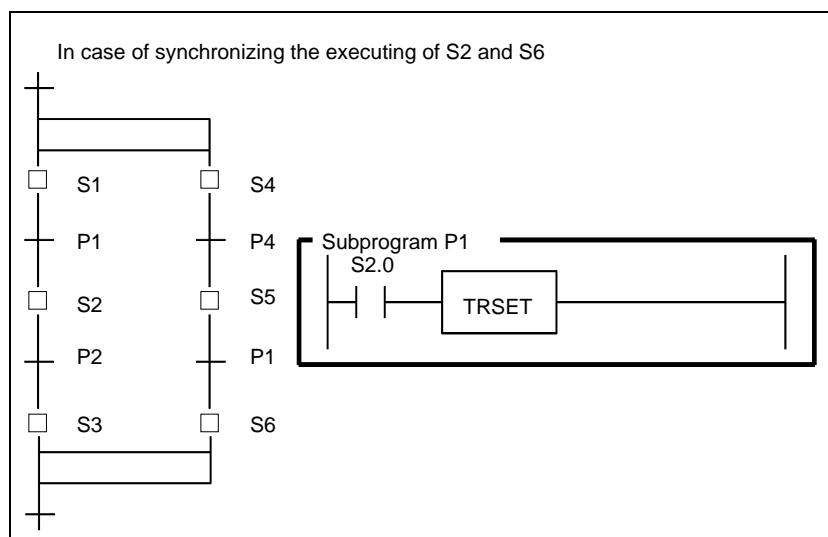
7.4.2 PMC Address (S Address)

Contents

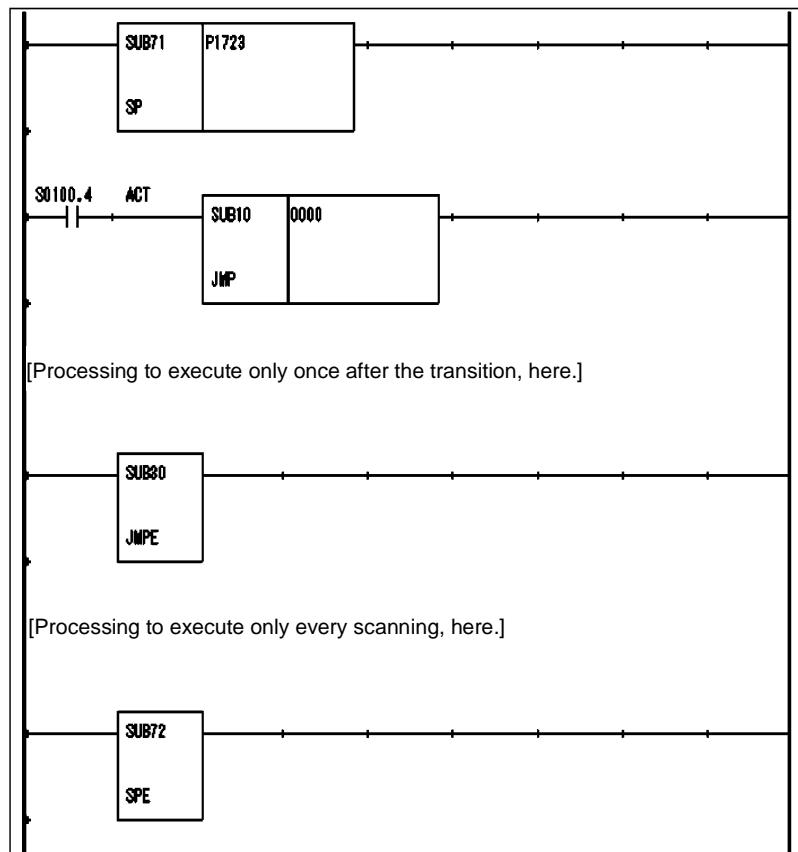
- An S address is created to end block step processing.
- The meaning of each bit of the step number (S address) is shown in the following.
 $Sn.0\ 0$: Transition to halt state, or halt state
 1 : Execution state
 $Sn.4\ 0$: Transition to halt state, or halt state, or the scanning execute for the first time.
 1 : Execution state (Turns ON 1 scan delayed from $Sn.0$)
- This address allows any subprogram to reference the state of any step.
- When 0 is written in the S address with byte size, the Execution State of step that is specified can be initialized. When LADDER was stopped, or the step which is no longer being called while in the activated state, etc, the step sequence program can be executed from the beginning when it is activated next time. The initialization of Execution State of step should be carried out with state that step sequence program which contains this step isn't called ($ACT=0$). To initialize a step sequence program, writes 0 in all the S addresses included within the program.
- A ladder for the TRSET transition instruction can be programmed using each bits of S address. Referring to S address, however, adversely affects the portability and comprehensibility. Use this feature sparingly.

Example 1

This address is used to reference the activation states of steps in a step in which this address has been specified, and performs complicated wait processing in a program including a simultaneous sequence.

**Example 2**

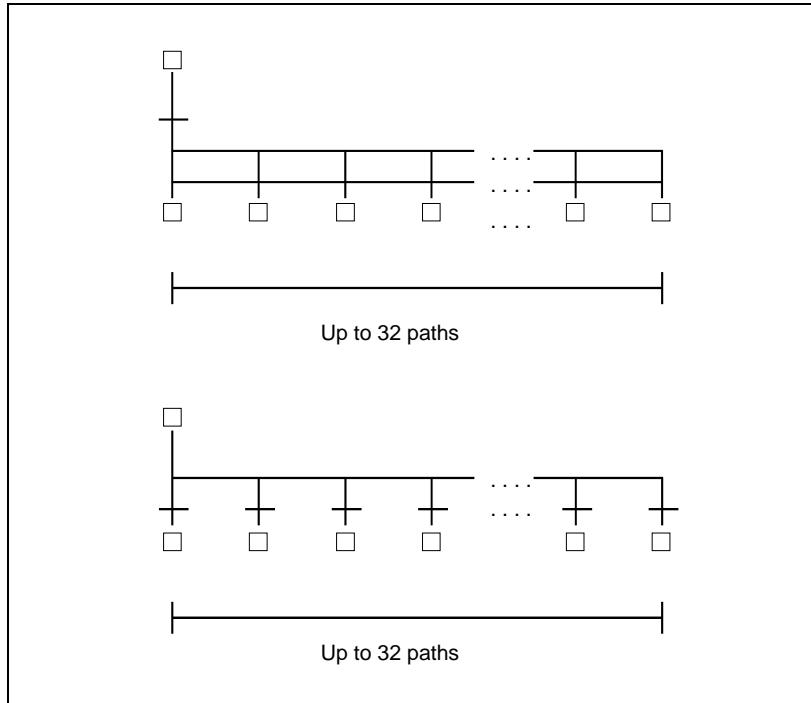
The section between JMP and JMPE in the following example is executed only once after the specific step (The following example is in the case of S100.) transits in the activated state from the inactivated state.



7.5 SPECIFICATION OF STEP SEQUENCE

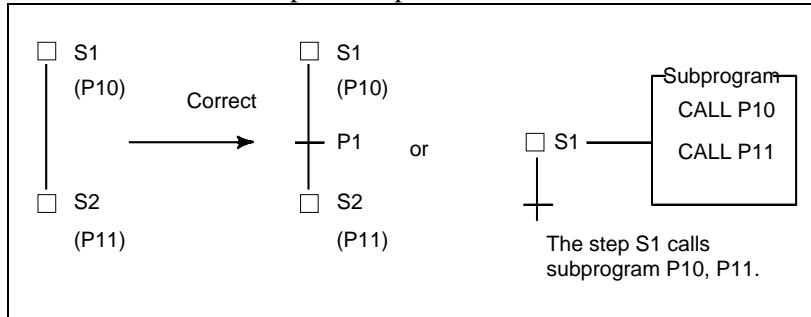
7.5.1 Specification

Item	Description
Number of subprogram	Up to 5000 (P1 to P5000)
Number of step	Up to 2000 (S1 to S2000)
Number of label	Up to 9999 (L1 to L9999)
Maximum number of jumps per block	Up to 256
Nesting depth of block step	Up to 8 levels
Size of block	192 lines × 48 columns
Number of paths	Up to 32 paths

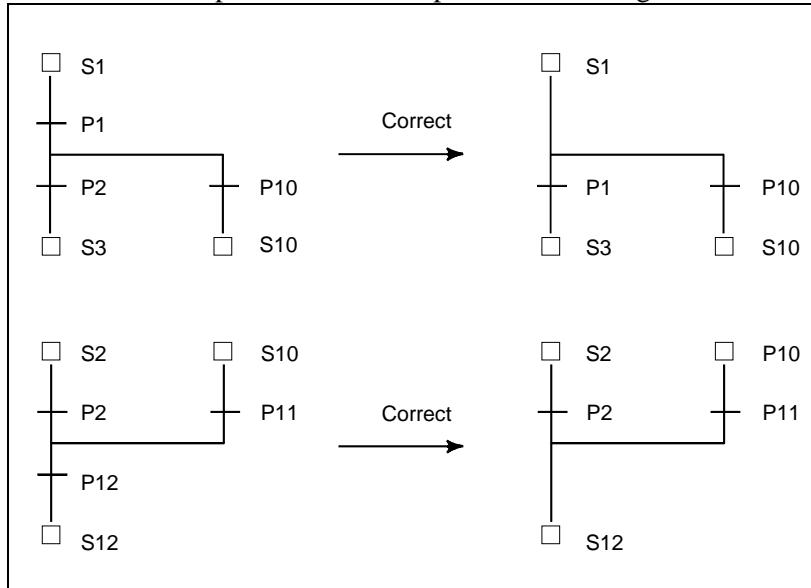


7.5.2 General Rules

- One transition must exist between step and step.



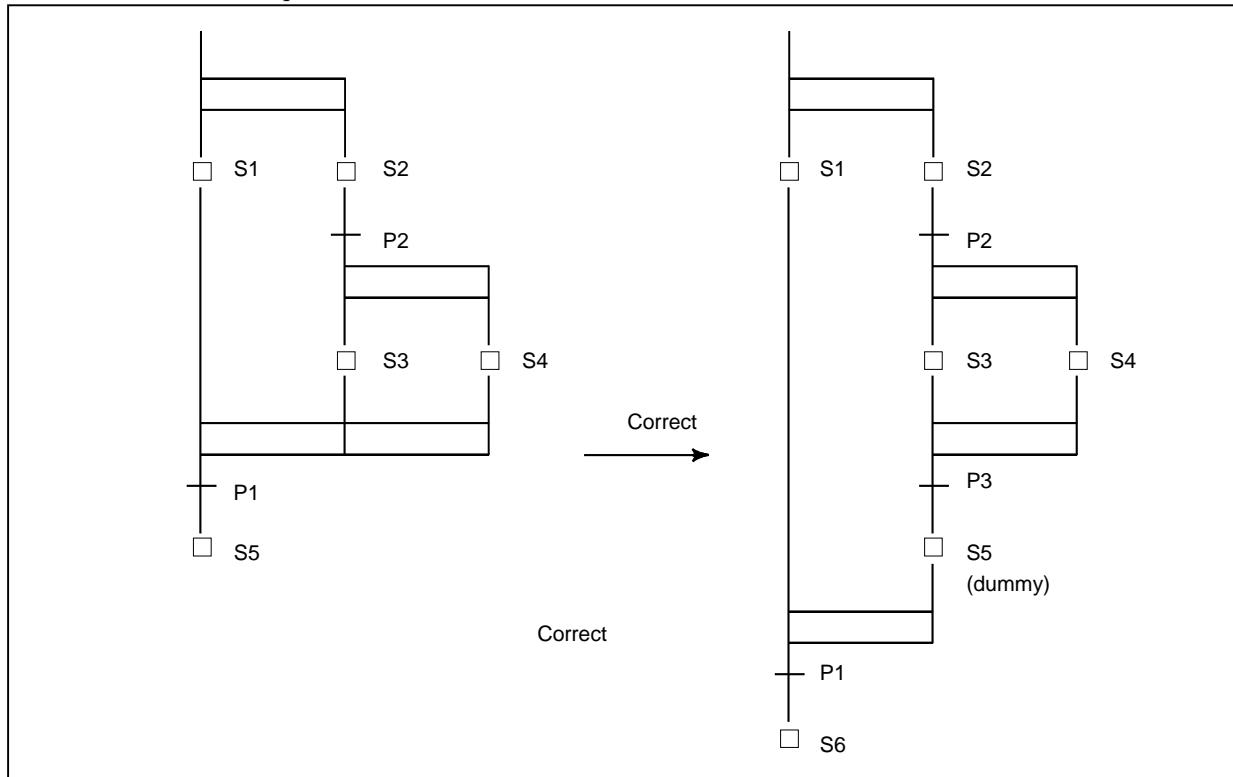
- The transition shall never be repeated even at the point of the divergence and the convergence.



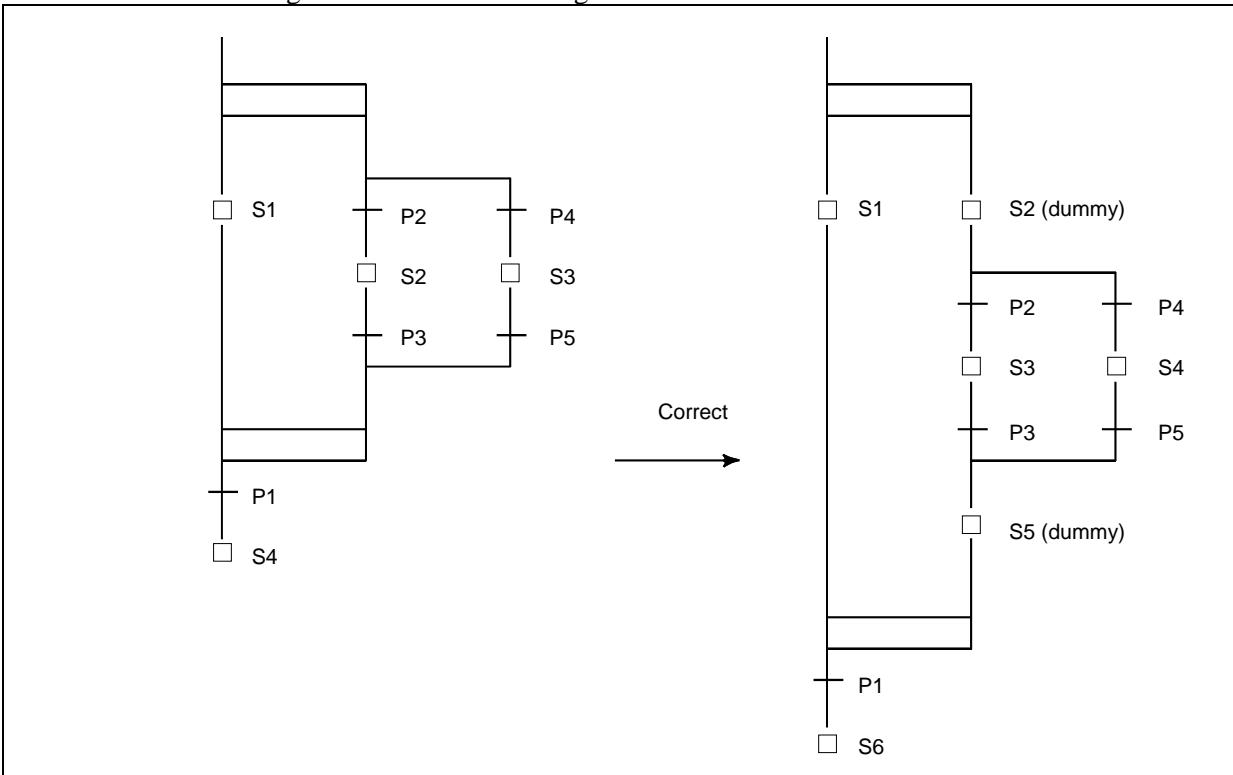
7. STEP SEQUENCE FUNCTION

B-83254EN/02

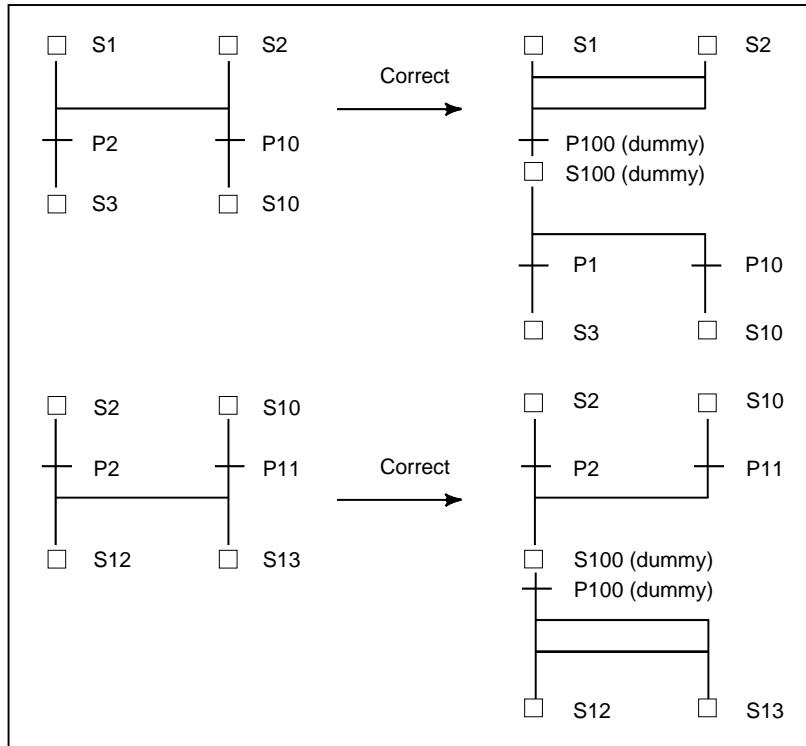
- When a simultaneous sequence is specified in another simultaneous sequence, one convergence must not be used for each sequence.



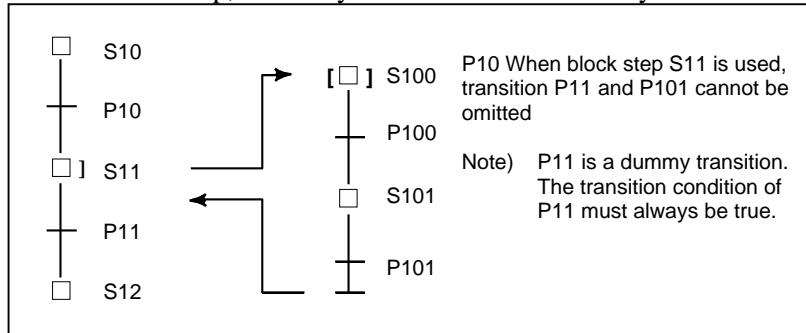
- When a selective sequence is specified in a simultaneous sequence, dummy steps must be required both after the divergence and before convergence.



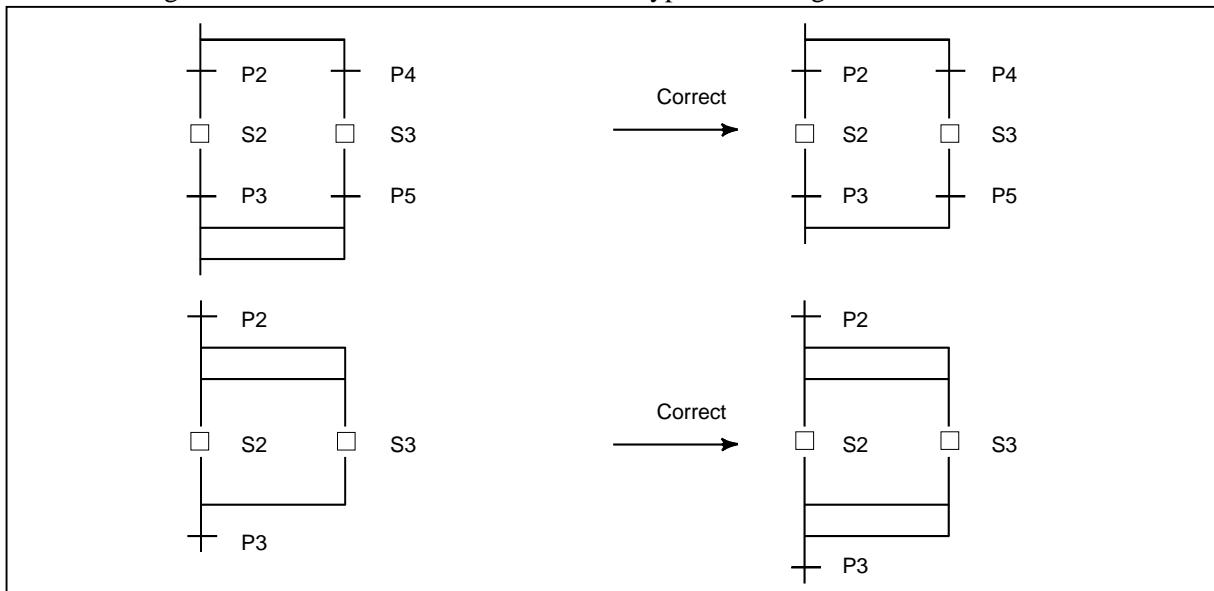
- In case of branching again immediately after the convergence, a step/transition is required between the divergence and convergence.



- Immediately after the block step, a dummy transition which is always true is needed.



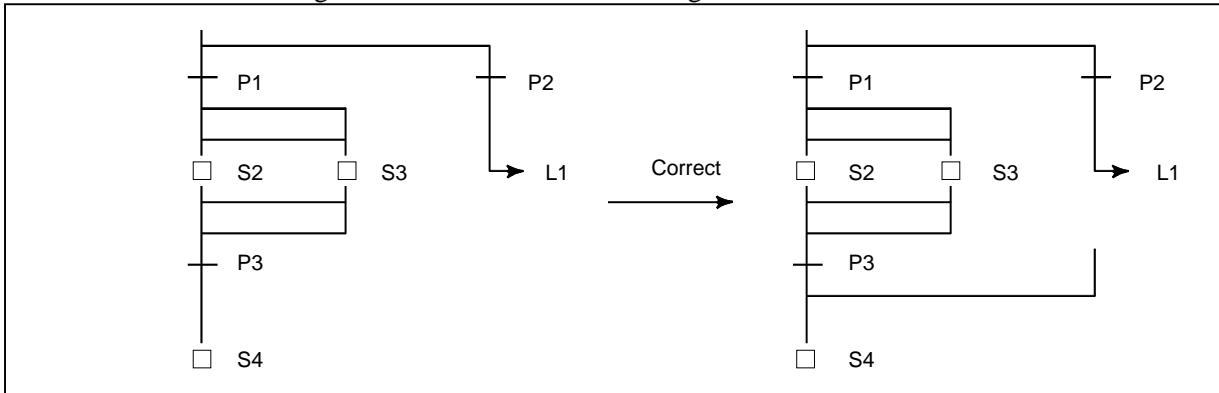
- The divergence must be terminated with the same type of convergence.



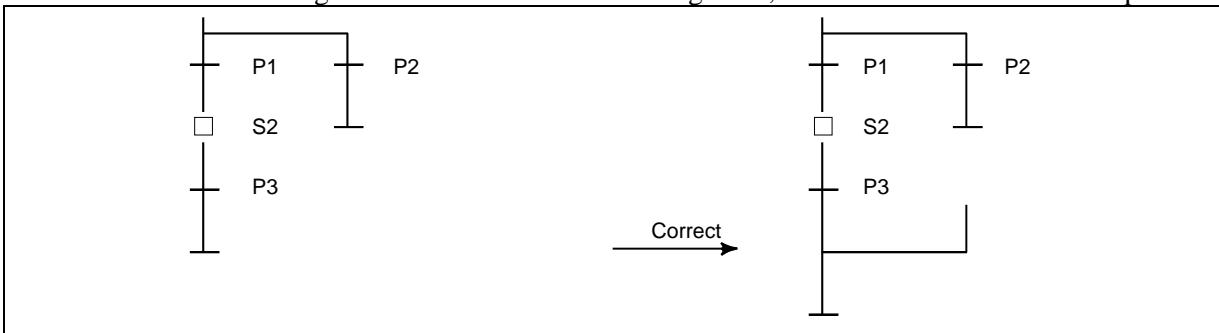
7. STEP SEQUENCE FUNCTION

B-83254EN/02

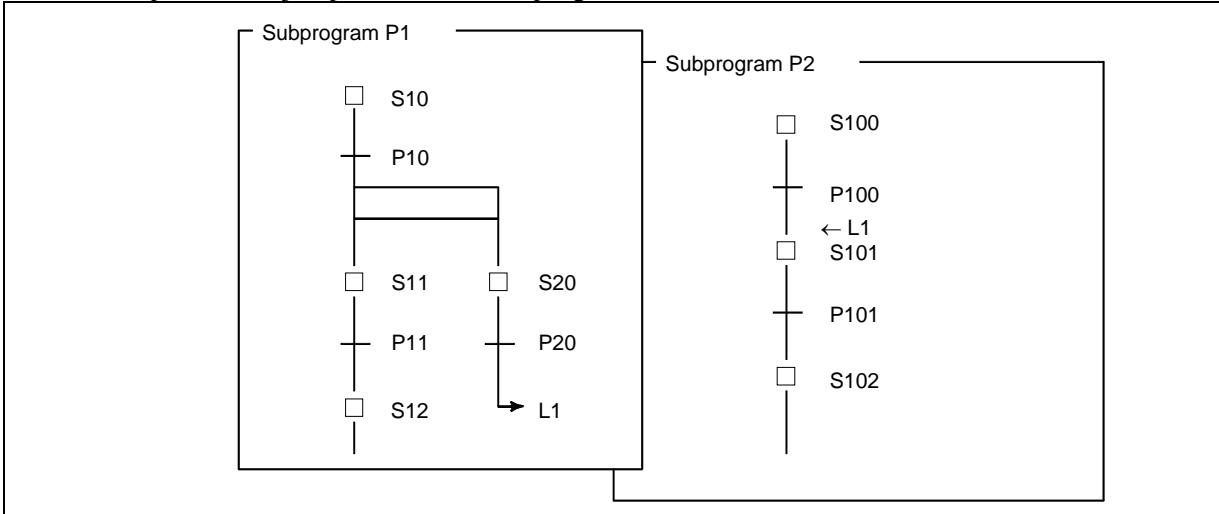
- The number of convergences must match that of divergences.



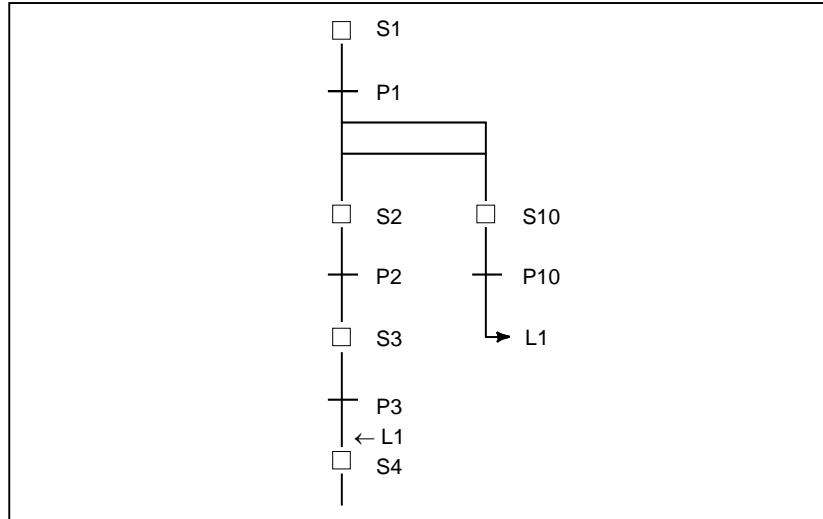
- The number of convergences must match that of divergences, even at the end of a block step.



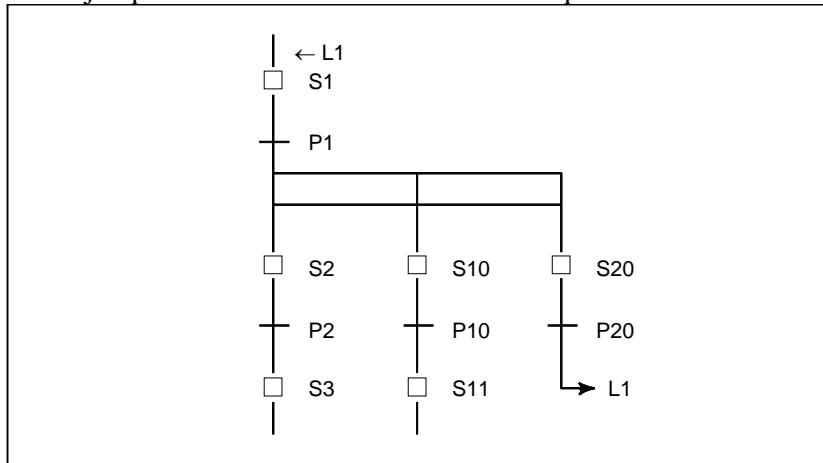
- It is not possible to jump to the other subprogram.



- It is not possible to jump from a simultaneous sequence to another simultaneous sequence.



- It is not allowed to jump from inside of the simultaneous sequence to outside.



7.5.3 Exclusive Control for Functional Instructions

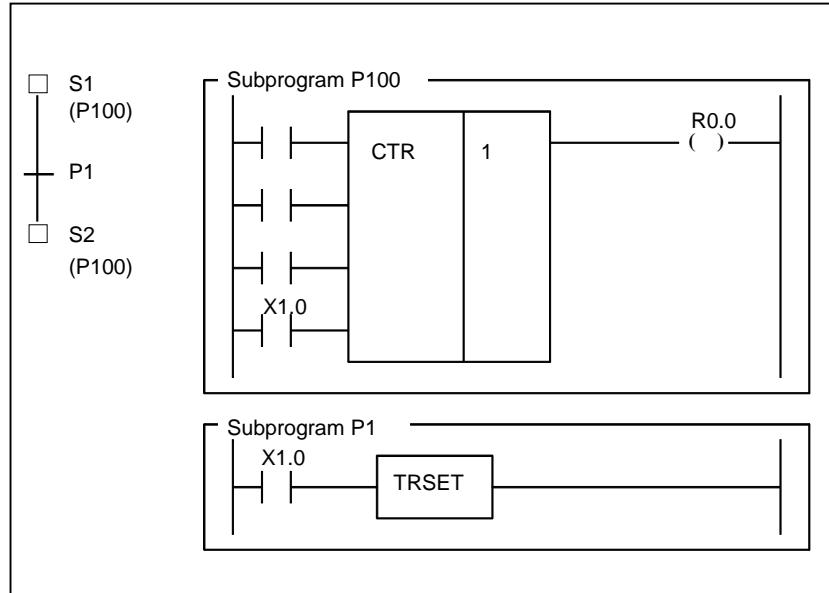
The use of the following basic/functional instructions is restricted in steps and transitions.

Description		Basic instructions	Functional instructions
The instructions operate when a signal changes.		RDPT	CTR (SUB5)
Condition	Multiple functional instructions having the same number are used.	ANDPT ORPT RDPT.STK RDNT	CTRC (SUB60) TMR (SUB3) TMRB (SUB24)
Problem	Not activated. Correct operation cannot be guaranteed.	ANDNT ORNTP RDNT.STK	TMRC (SUB54) DIFU (SUB57) DIFD (SUB58)

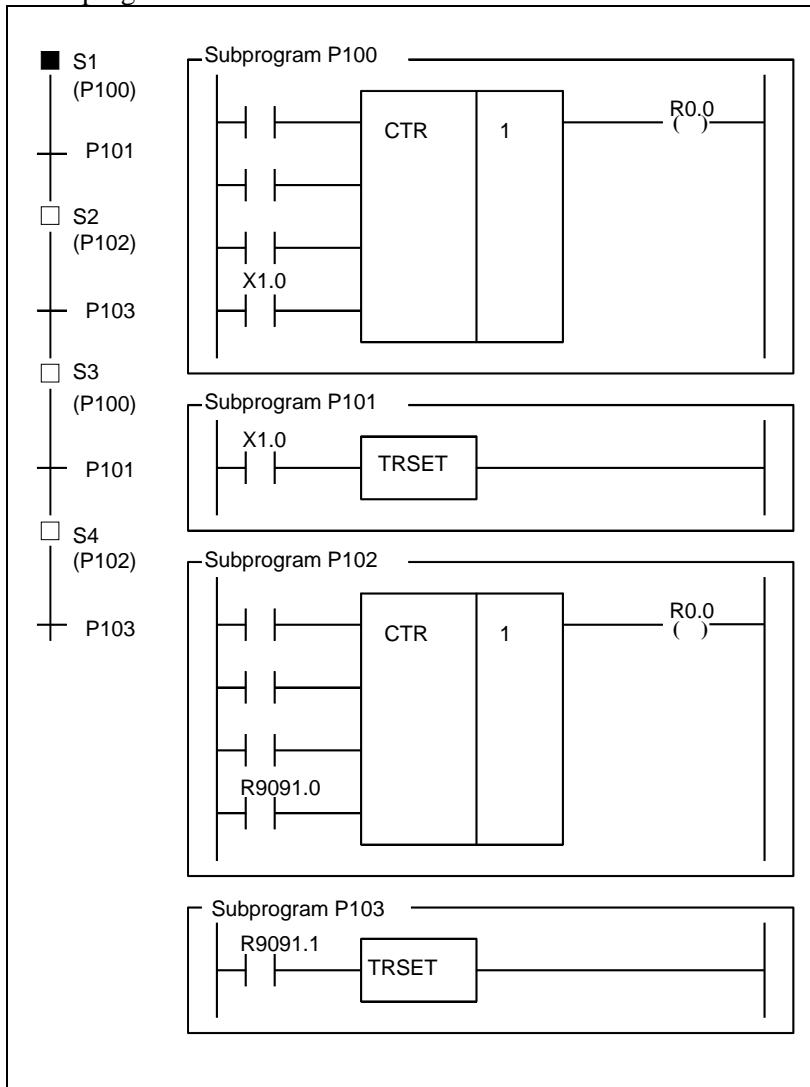
Since these functional instructions operate when the corresponding signals change, they may not operate correctly when called from multiple steps.

Example

While multiple CTR functional instructions are used, when control passes from S1 to S2 with ACT of CTR not set to off, CTR is not counted when called from step S2.

**Correct program**

Divide the subprogram so that ACT of CTR is called after it is set to off.



8 FUNCTION BLOCK FUNCTION

8.1 OVERVIEW

A “function block” is a block of a ladder program defined in advance that implements a particular process (function).

You can place a defined function block in other ladder program and set required input/output parameters to execute the function.

By defining a frequently used function as a function block, you can reuse the function easily, and can reduce the time required for programming and increase the development efficiency.

In addition, program diagnosis can be performed without displaying the detailed program in the function block, which is also effective to decreases the amount of the printed maintenance ladder diagram.

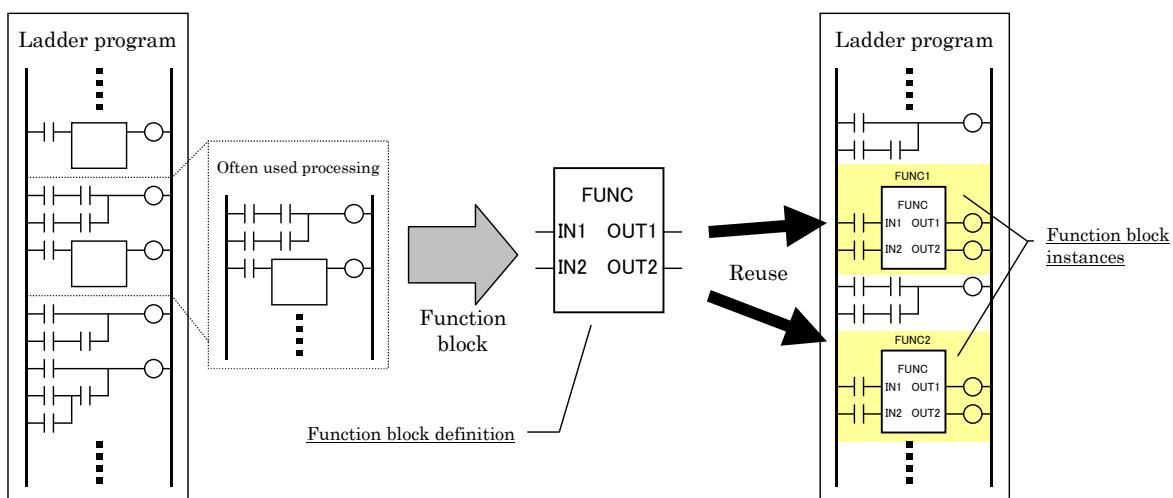


Fig. 8.1 Reusing a program using a function block

NOTE

To use the Function Block function, Multi-path integrated PMC option is necessary.

Definition and instance

To create a function block, you need ladder program to implement the function and the input and output signals for the program. These are called as “function block definition.” You can paste the defined function block into an actual program and specify the input and output signals to call and execute the function. Each function block pasted into a program is called a “function block instance.” You can create more than one instances of the same function block in a program.

NOTE

Programming using function blocks requires FANUC LADDER-III for Robot, a PMC programmer that runs on PC.

Assignment of addresses to parameters and variables

Program a ladder program in a function block definition using variables (symbols) to which specific addresses are not assigned (symbol programming). When a program containing a function block instance is compiled, specific addresses are assigned to the parameters and variables used in the program in the function block. Different addresses are assigned to different function block instances and individual instances operate independently.

8.1.1 Item Names

A function block is represented by a rectangle as shown below.

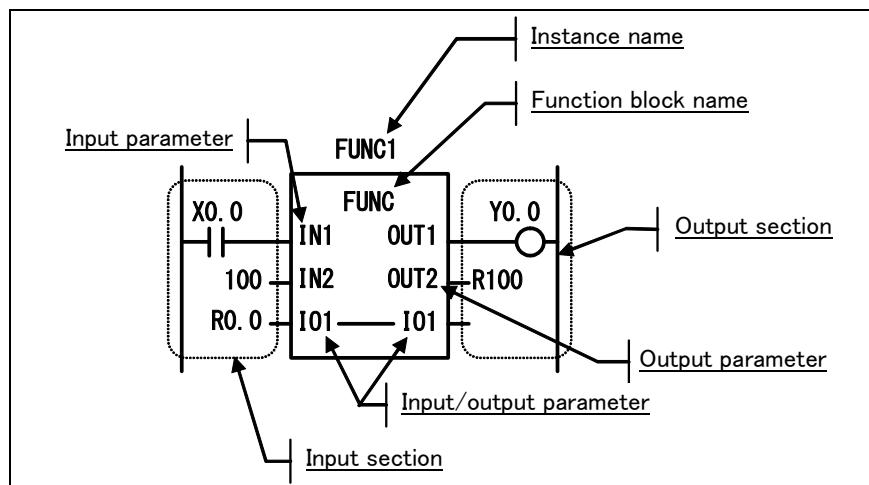


Fig. 8.1.1 Function block

An “**instance name**” is a name that uniquely identifies each instance of a function block. Each instance has different name with each other.

A “**function block name**” is the name of the source function block definition of each instance. The instances that call the same function block have the same function block name.

“**Input parameters**” receive input signals to a function block. Specify the value passed to each input parameter with an address or constant in the “input section”. For a bit signal, specify its address on the relevant contact.

“**Output parameters**” are output signals from a function block. In the “output section”, specify the address for receiving each output parameter value to fetch the output of the function block. For a bit signal, specify its address on the relevant coil.

“**Input/output parameters**” functions as both input and output of a function block. It is represented as the two same parameter names in the input and output parameter positions connected by a line.

8.1.2 Overview of Specifications

(1) Function block definition

Item	Specification	Remarks
Function block name	Identifier consisting of up to 40 characters	Conforms to IEC61131-3.
Comment	Character string consisting of up to 255 characters x 4 (Japanese characters available)	
Parameter	Up to 64 parameters in total of input and output	

Item	Specification	Remarks
Data protection	For each function block, "editing protection" or "browsing and editing protection" using a password can be specified.	
Other information	Version information	

(2) Parameter specifications

Item	Specification	Remarks
Types of parameters	Input parameter Input/output parameter (NOTE 1) Output parameter	The EN input and ENO output are also supported.
Maximum number of parameters	64 parameters in total	
Name (symbol)	Identifier consisting of up to 40 characters	Conforms to IEC61131-3.
Comment	Character string consisting of up to 255 characters x 4 (Japanese characters available)	
Data types	BOOL SINT, USINT, INT, UINT, DINT, UDINT BYTE, WORD, DWORD	Conforms to IEC61131-3.
Count specification (NOTE 2)	1 to 32	Integer parameters other than BOOL only
Displaying Internal and External Variables in the Monitor (FB Instance Monitor Display)	BOOL SINT, USINT, INT, UINT, DINT, UDINT BYTE, WORD, DWORD 8-bit bit string, 16-bit bit string	Can be specified up to 16 parameters in each function block.

CAUTION

When editing FB definitions and FB instances by the offline editing function on FANUC LADDER-III for Robot, the addresses assigned to the parameters and variables of function block will be changed. Therefore, the parameters and variables of all function blocks will be initialized by 0 when the sequence program is inputted into Robot controller. (See "8.1.4 Assignment of FB variable" for details.)

NOTE

- 1 While the data of input/output parameters are passed by reference, the data of other parameters are passed by value.
- 2 A value of 2 or larger can be specified to pass multiple contiguous data items of the same data type.

(3) Variable specifications

Item	Specification	Remarks
Types of variables	Internal variable External variable	
Maximum number of variables	1024 in total	
Name (symbol)	Identifier consisting of up to 40 characters	Conforms to IEC61131-3.
Comment	Character string consisting of up to 255 characters x 4 (Japanese characters available)	
Data types	BOOL SINT, USINT, INT, UINT, DINT, UDINT BYTE, WORD, DWORD	Conforms to IEC61131-3.

Item	Specification	Remarks
Count specification (NOTE 1)	1 to 1000	Can be specified only for non-bool internal variables.
Nonvolatile memory type specification	Available	Can be specified only for internal variables.
Memory allocation of internal variables	Contiguously allocated in the order in which they are defined.	Divided into nonvolatile and volatile types and arranged in different areas.
Displaying Internal and External Variables in the Monitor (FB Instance Monitor Display)	BOOL SINT, USINT, INT, UINT, DINT, UDINT BYTE, WORD, DWORD 8-bit bit string, 16-bit bit string	Can be specified up to 16 variables in each function block.

 **CAUTION**

When editing FB definitions and FB instances by the offline editing function on FANUC LADDER-III for Robot, the addresses assigned to the parameters and variables of function block will be changed. Therefore, the parameters and variables of all function blocks will be initialized by 0 when the sequence program is inputted into Robot controller. (See "8.1.4 Assignment of FB variable" for details.)

NOTE

- 1 A value of 2 or larger can be specified to allocate contiguous areas for multiple data items of the same data type.

(4) Program in a function block(FB body program)

Item	Specification	Remarks
Programming language	Ladder language can call another function block (up to 4 nested levels).	
Maximum number of steps	8000 steps per function block	
Available addresses	Defined parameters, and internal and external variables (NOTE 1) Fixed PMC addresses (NOTE 2)	
Available instructions	Basic and functional instructions available with the PMC. The following instructions cannot be used, however: END1, END2, END3, END SP, SPE, CALL, CALLU, JMPC CS, CM, CE The following instructions cannot be used in any function block for which more than one instance is to be created: TMR CTR, CTRB For the following instructions, the automatic number assignment function must be used: TMRB, TMRBF DIFU, DIFD	
Call of another function block	Other function block can be called up to 4 levels deep.	

NOTE

- 1 They are programmed not with actual addresses, but with symbols.
- 2 Any addresses (including X, Y, F, G, R, D, and so on) available in the ladder language of the PMC can be directly specified.

(5) Function block call

Item	Specification	Remarks
Instance name	Identifier consisting of up to 40 characters	Conforms to IEC61131-3.
Comment	Character string consisting of up to 255 characters x 4 (Japanese characters available)	
Parameter specification	For a BOOL parameter, connect basic instructions. For an integer parameter, specify an address or constant. For an integer input/output parameter, specify an address only.	
Program level to call function block	Can be called from level 1 to 3 or subprogram. Placed as a net in ladder program.	
Number of function block calls	Up to 1024 types of function blocks Up to 5000 calls (instances) (NOTE)	Function block instance called from a function block also included

NOTE

In PMC Memory-B/C/D up to 5000 instances can be used. In PMC Memory-A up to 512 instances can be used.

8.1.3 Memory Usage Related to Function Blocks

The following table lists memory usage related to programming using function blocks.

Table 8.1.3 (a) Memory usage related to function blocks

Category	Item	Memory usage (NOTE 1)
Function block definition information (NOTE 2)	One function block (including name and comment character string)	55 to 148 bytes
	One parameter information item (including symbol and comment character string)	14 to 91 bytes
	Program section	Varies depending on the program (NOTE 3)
Function block call (instance)	One call	76 bytes
	BOOL type parameter	12 bytes + Input/Output circuit(NOTE 5)
	Parameter other than BOOL type	24 bytes (NOTE 5)
	FB body program section (NOTE 4)	Equivalent to a conventional ladder program(NOTE 6)
Symbol and comment (Extended function format)	One definition	16 to 23 bytes (NOTE 8)
	One symbol character	1 byte
	One comment character (single-byte character)	1 byte (NOTE 7)
	One function block (NOTE 9)	8 bytes

NOTES

- 1 In addition to the memory usage listed in the table, some amount of memory may be used to adjust the memory allocation.
- 2 These items are required for each type of function block used in the program.
- 3 To enable function block definitions to be restored at decompilation, include the function block definition data in the object. In this case, the memory usage varies depending on the contents of the function block definition. Generally the memory usage of a function block consisting of 8000 steps may be about 7K to 10K bytes.
- 4 The size of FB body program is added for each instance.
- 5 The memory usage in following cases is 8 bytes.
 - Case of input side of input/output parameter
 - Case of omitted output side of output parameter and input/output parameter.
- 6 The size of FB body program is calculated in the same way as for conventional ladder programs as the memory usage listed in the table below.

Type of instruction	Memory usage
Basic instruction	4 bytes
Functional instruction	4 bytes
Functional instruction parameter	4 bytes

See "2.1.3 Sequence Program Memory Capacity" for details.

- 7 One double-byte character uses 2 bytes.
- 8 One definition of symbol and comment data uses 16 to 23 bytes of memory. In addition, memory is used based on the lengths of the symbol and comment character strings.
- 9 This memory is required for each function block call.

8.1.4 Assignment of FB Variable

An address of FB variable is assigned at compiling on FANUC LADDER-III for Robot and the assigned address depends on the arrangement of FB instance in the ladder program. Therefore, when FB definition and FB instance are edited, the assignment of address may be changed.

When the sequence program being executed is updated to the sequence program whose FB variables are assigned to the different addresses, value of the variables may be unsuitable. For this reason, when updating sequence program to the one whose FB variables are assigned to the different addresses, the PMC system software will initialize FB variable area by 0. Therefore, you should design your function block to operate safely when updating sequence program to the one whose FB variables are assigned to the different addresses. The initialization range of FB variable area is not only actually assigned address for variables but all addresses specified by setting of "Assignment Address of FB" on FANUC LADDER-III for Robot.

When updating sequence program to the one by the following operations, FB variable area will be initialized.

- (a) When changing a FB definition (except for editing FB body program only)
- (b) When adding / deleting / moving a FB instance
- (c) When changing an address of input / output parameter
- (d) When changing a symbol / comment data referred as an external variable
- (e) When changing the setting of "Assignment Address of FB" in the system parameter

NOTE

- 1 Depending on how you modify the sequence program, the updated sequence program may run safely without initializing FB variable area.
- 2 By setting 1 to K903.4 of system keep relay, you can choose not initialize FB variable area when changing the address of FB variable. (See "2.2.10 System Keep Relay Addresses (K)" for details)

8.2 FUNCTION BLOCK DEFINITION

The definition section of a function block consists of the following information:

- Function block name
- Information of variables (including parameters and internal variables)
- FB body program
- Other information

The following sections explain the above items.

8.2.1 Function Block Name

A function block name is a character string used to identify a function block.

A character string consisting of the following characters (identifier conforming to IEC61131-3) can be used as the name of a function block:

- Alphabetic characters (A to Z)
- Numeric characters (0 to 9)
- Underscore (_)

NOTE

- 1 A function block name must not begin with a numeric character. When an underscore is specified as the first character, it must be followed by an alphanumeric character.
- 2 A name character string can consist of up to 40 characters.

In addition to the name, you can define an arbitrary character string as a comment for a function block.

8.2.2 Variable Information

Variables used in the FB body program must be declared in advance.

The following types of variables are available in the program:

- Parameter
- Internal variable
- External variable

⚠ CAUTION

When editing FB definitions and FB instances by the offline editing function on FANUC LADDER-III for Robot, the addresses assigned to the parameters and variables of function block will be changed. Therefore, the parameters and variables of all function blocks will be initialized by 0 when the sequence program is inputted into Robot controller. (See “8.1.4 Assignment of FB variable” for details.)

The following table lists the maximum number of variables of each type that can be used in a function block.

Type	Maximum number
Parameter	64 in total
Internal and external variables	1024 in total

NOTE

- 1 Different addresses are assigned for parameters and internal variables in different function block instances.
- 2 You can directly specify an actual address in the FB body program. In this case, the address has an effect equivalent to an external variable. The address is not included in the above number because it is not assumed to be an external variable.

Each type of variable definition consists of the following information. Each variable type has its features and restrictions. For details, see the explanation of each type of variable.

(a) Symbol

Each variable is identified by a symbol represented by a character string consisting of the following characters (identifier conforming to IEC61131-3):

- Alphabetic characters (A to Z)
- Numeric characters (0 to 9)
- Underscore (_)

NOTE

- 1 A symbol must not begin with a numeric character. When an underscore is specified as the first character, it must be followed by an alphanumeric character.
- 2 A symbol character string can consist of up to 40 characters.
- 3 The following symbols are reserved and not available for other purpose:
 - EN
 - ENO

For details of these symbols, see “(1) EN input and ENO output”.

In addition to the symbol, you can define an arbitrary character string as a comment for each variable.

You cannot use the same symbol for more than one variable in a function block definition.

(b) Basic data type

A defined variable must have one of the following data types conforming to IEC61131-3.

Type name	Data type	Monitor format
BOOL	1-bit bool value	ON/OFF
SINT	8-bit signed integer value	Signed decimal number
USINT	8-bit unsigned integer value	Unsigned decimal number
INT	16-bit signed integer value	Signed decimal number
UINT	16-bit unsigned integer value	Unsigned decimal number
DINT	32-bit signed integer value	Signed decimal number
UDINT	32-bit unsigned integer value	Unsigned decimal number
BYTE	8-bit bit string	Hexadecimal number
WORD	16-bit bit string	Hexadecimal number
DWORD	32-bit bit string	Hexadecimal number

NOTE

- 1 A constant is also displayed in the monitor format listed above if given to an input parameter.
- 2 BCD data is correctly displayed in hexadecimal notation.

(c) Count specification

For input and output parameters and internal variables of the data types that occupy 1 byte or more such as INT, you can specify the number of data items to allocate their area. For example, when you specify 3 for the number of an INT internal variable, 6-byte area is allocated as the area for the variable.

Type of variable	Count specification range
Input or output parameter	1 to 32
Internal variable	1 to 1000

NOTE

The larger value is specified as the number of input or output parameters, the larger amount of data must be copied during the execution of each relevant function block instance, resulting in worse performance. In this case, memory allocated for each instance is also increased. If you require input or output parameters that use a large amount of PMC memory (R, D), you can use input/output parameters to efficiently pass the large data.

Parameter

Parameters are variables used to exchange values between a function block and the circuit outside the function block.

Parameters are divided into the following types:

- Input parameter
- Output parameter
- Input/output parameter

In addition, there are the following two special parameters:

- EN input
- ENO output

The EN input and ENO output are special input and output parameters that control the execution of the function block. For details, see “(1) EN input and ENO output” below.

For each parameter, specify an address for exchanging a value or a constant. While a constant or address can be specified for an input parameter, only an address can be specified for an input/output parameter or output parameter.

Each type of parameter is explained below.

(1) EN input and ENO output

The EN input is an input parameter which controls execution of the function block itself. The ENO output is an output parameter which indicates whether the function block terminates normally when the execution of the function block itself is completed.

The EN input and ENO output may or may not be specified. When defining a function block, specify whether to use each of the EN input and ENO output.

NOTE

A parameter having the name of EN or ENO is always treated as the EN input or ENO output. You cannot define a parameter or variable other than the EN input or ENO output with the name of EN or ENO.

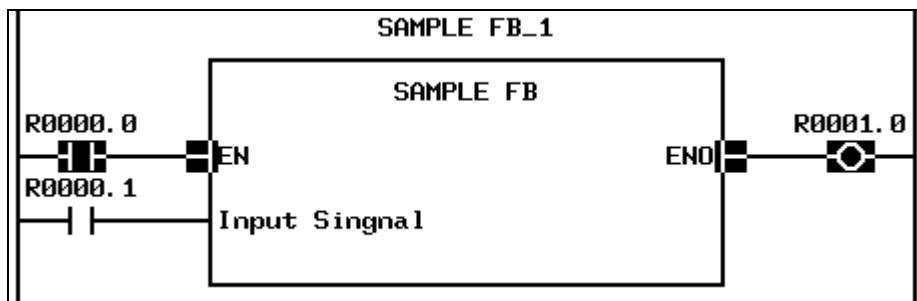


Fig.8.2.2 (a) EN input and ENO output

(a) EN input

The EN input controls whether to execute the function block. It functions as follows:

- When the EN input is ON, the FB body program is executed. When the FB has ENO output, the ENO is set to ON before the program is executed.
- When the EN input is OFF, the FB body program is not executed and control is passed to the execution of the subsequent program with the status at that point kept. When the FB has ENO output, the ENO is set to OFF.

When the FB has no EN input, the FB body program is executed in the same way as when it is ON.

NOTE

You can also use common line control (COM instruction) to control execution, which is similar to using the EN input. Common line control is also valid for a function block for without EN input.

(b) ENO output

The ENO output indicates whether operation of a function block terminates normally. The value of the ENO output is set to ON before the FB body program is executed. If an error occurs in the FB body program and the output is invalid, the ENO output should be set to OFF. When the EN input is OFF or when ACT of common line control (COM instruction) is OFF, the ENO output is automatically set to OFF.

(2) Input parameter

An input parameter is a variable which receives the input to the FB body program. It is read only in the FB body program. The EN input is a kind of input parameter.

Input parameters are displayed at the left side of a function block instance.

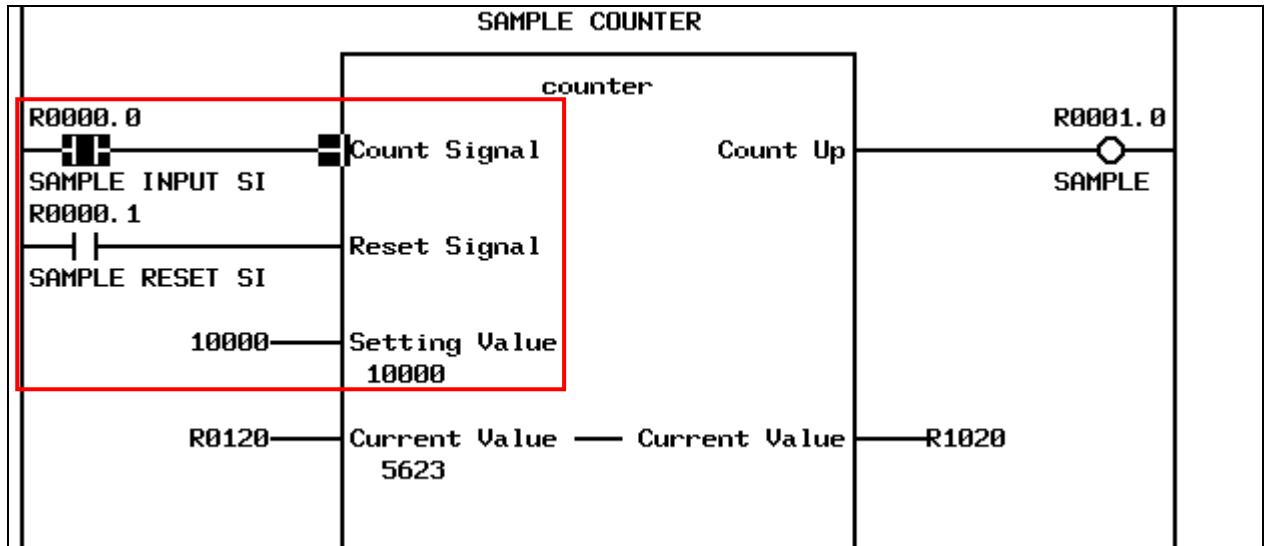


Fig.8.2.2 (b) Input parameters and input section

NOTE

You cannot write to an input parameter in the FB body program.

(3) Output parameter

An output parameter is a variable to pass the output from the FB body program. A value should be set to it by the FB body program. The ENO output is a kind of output parameter.

If you do not have to fetch any output value, you can leave an output section without specifying an address.

Output parameters are displayed at the right side of a function block instance in the output section.

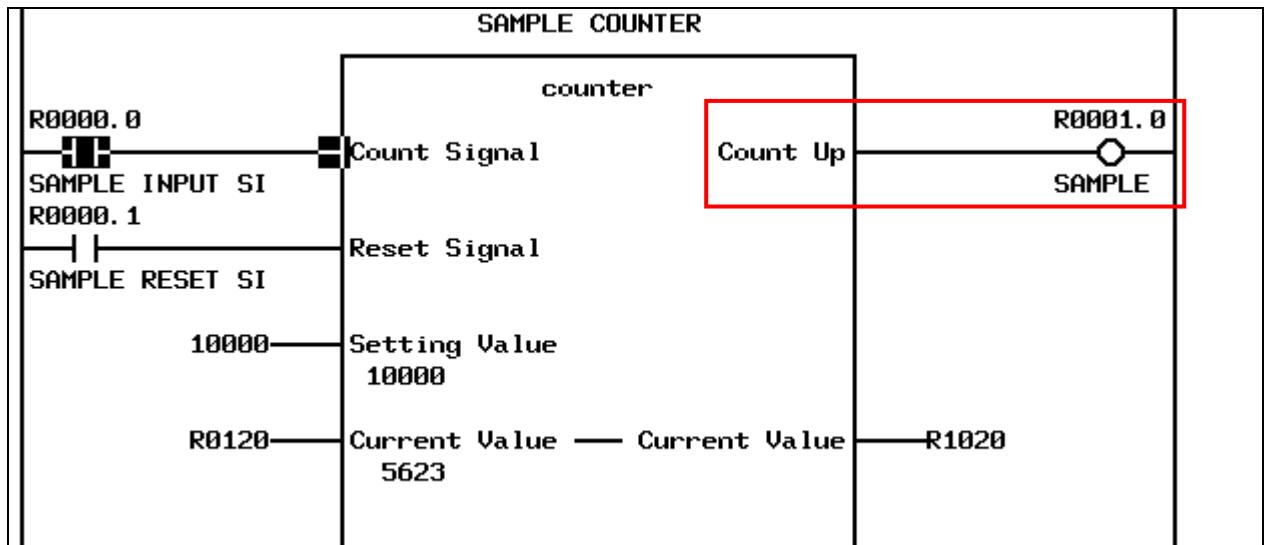


Fig.8.2.2 (c) Output parameters and output section

NOTE

If a value is not set for an output parameter in the FB body program, the previous value is remained.

(4) Input/output parameter

An input/output parameter is handled as a variable which receives the input to the FB body program and of which value can be changed by the FB body program.

It can be read and written by the FB body program without restrictions.

An input/output parameter is displayed at both sides of a function block connected by a line.

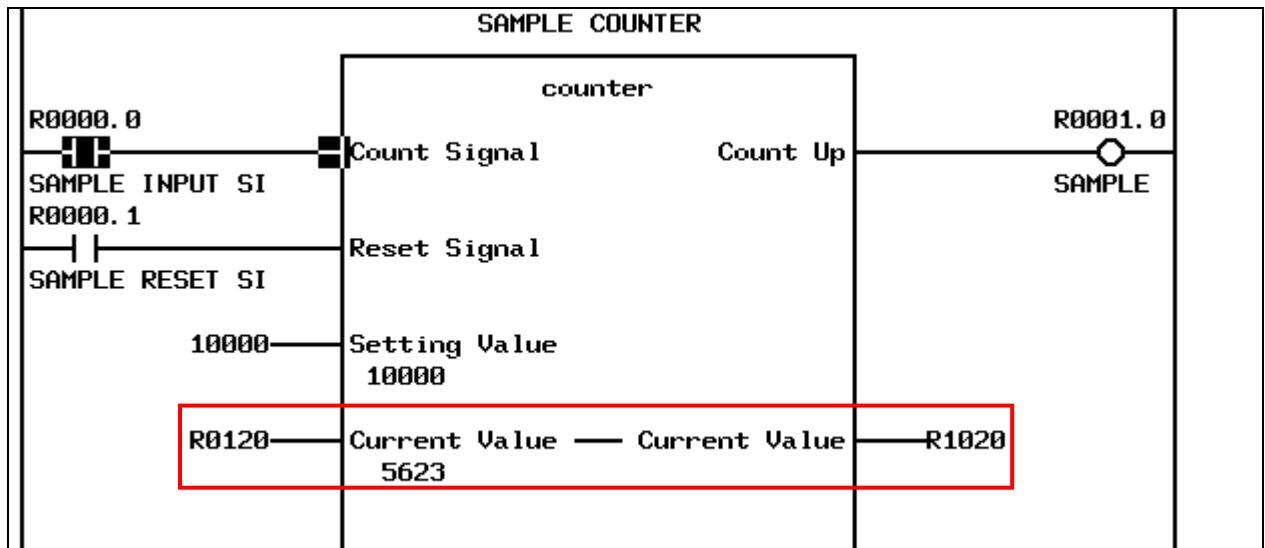


Fig.8.2.2 (d) Input/output parameter display

For an input/output parameter, you can specify an address in each of the left input and right output sections. You can omit an address in the output section, but cannot omit one in the input section. When an input/output parameter is accessed in the FB body program, the address specified in the input section is referenced directly. The value of the input/output parameter is copied to the address specified in the output section after function block processing terminates.

For input/output parameter "Current Value" in the figure above, the address specified in the input section is "R0120" and that specified in the output section is "R1020."

(a) Feature of input/output parameters

For an input parameter, the given constant or the value at the specified address is stored in the variable assigned as the input parameter before the start of FB body program processing. For an output parameter, the value of the output parameter is written at the specified address after the termination of function block processing.

In contrast, for an input/output parameter, the program in the function block directly accesses the address specified in the input section. Therefore, writing an input/output parameter by the FB body program means directly writing a value at the address specified in the input section for that input/output parameter.

You can use this feature of input/output parameters when the order to access signals must strictly be controlled in a function block or when a large amount of data such as table data needs to be passed.

(b) Notes on input/output parameters

Note the following points when using input/output parameters:

- No constant can be specified for an input/output parameter.
- The value of an input/output parameter may change during the execution of the FB body program.
- The address assigned in the input section for an input/output parameter cannot be changed by the PMC edit function.
- When the same address is set to different input/output parameters of a function block, or when an address used as an external variable in the function block is set for an

input/output parameter of the function block, a correct symbol may not be displayed for the address in the FB body program when displayed.

Internal variables

An internal variable is used only in the function block.

(1) Nonvolatile memory type

The nonvolatile memory type can be specified for an internal variable. In this case, the variable is allocated in the nonvolatile memory type area (D address).

(2) Arrangement

Internal variables are arranged in contiguous memory areas in the order in which they are defined. Nonvolatile and volatile variables are arranged in different areas.

NOTE

- 1 When internal variables of different data types are arranged, any variable of a data type such as INT or DWORD that occupies at least 2 bytes is not aligned based on the data type, but is arranged on a byte boundary. To avoid affect performance, try to adjust the order of variable definitions so that variables of these data types are arranged at even addresses. (The start of an internal variable is always arranged at an even address.)
- 2 When a non-BOOL variable is arranged following a BOOL variable, any unused bit address is not assigned to other BOOL variable after that. For example, BOOL, SINT, BOOL, and SINT variables defined in this order use 4 byte area. A used memory can be reduced to define the BOOL variables contiguously.

#7	#6	#5	#4	#3	#2	#1	#0
				Unused			
							BOOL uses 1 bit.
							SINT uses 1 byte.
				Unused			BOOL uses 1 bit.
							SINT uses 1 byte.

(3) Input parameter

When a function block is displayed on the screen, any internal variable is not displayed on that screen. If you want to display and monitor the value of an internal variable in the function block, you can specify the “monitor display” attribute with FANUC LADDER-III for Robot to display and monitor the value of the variable in the function block. For details of monitor display of internal variables,

External variables

An external variable is used in a function block to refer a symbol defined in advance in ladder program outside the function block. The entity (address) to be accessed is the same even from different function block instances.

If a symbol definition of the same name as an external variable is not found during compilation, an error occurs.

NOTE

- 1 Details (including the real address) of an external variable are defined not by an external variable declaration in the function block definition, but by symbol data of the used ladder program. An external variable declaration in a function block definition is used for referencing a variable defined in the ladder program.
- 2 The name of an external variable is an identifier conforming to IEC61131-3, so only a symbol defined as an identifier conforming to IEC61131-3 can be referenced. For details of a variable name, see Section 8.2.2, "Variable Information"
- 3 The symbol and data type of a declared external variable must be consistent within the whole program. For example, if a program registers symbol "ABC" of the bit type, and a function block declares "ABC" as a byte external variable, a compilation error occurs because the data type of the variable indicated by symbol "ABC" is inconsistent.
- 4 A fixed address can be referenced by writing not a symbol, but a specific address directly in the FB body program. In this case, the address does not need to be counted as an external variable.

(1) Monitor display

The "monitor display" attribute can be specified for an external variable like an internal variable. Specifying this attribute with FANUC LADDER-III for Robot will display and monitor an external variable, which is normally not displayed in the function block.

8.2.3 FB Body Program

The function of each function block is defined by ladder program programmed using symbols. All symbols that appear in the program must be declared as variables in advance. You can also specify an address directly in a program to always access a fixed address.

NOTE

Creating more than one instance of a function block which writes data at a fixed address causes duplicate writing.

(1) Levels of nested function block calls

From the FB body program, other function block can be called. Calling function block can be nested up to 4 levels deep. If calling function block is nested more than 4 levels deep, a compilation error occurs on FANUC LADDER-III for Robot.

NOTE

- 1 An ordinary subprogram cannot be called from a function block.
- 2 A function block call is independent of the nesting of subprogram call using the CALL or CALLU instruction in the ladder language. Therefore, you do not have to count a function block call in the number of nested subprogram call levels, or a subprogram call in the number of nested function block call levels.
- 3 The depth of nested function block calls is not determined based on not the number of nested function blocks actually called during execution, but the number of nested function block calls defined in the program. That is, a function block call that is programmed not to call actually is also counted. Therefore, any function block cannot be called recursively.
- 4 For each of function block calls (instances) in a function block, a number is automatically added to its instance name during compilation on FANUC LADDER-III for Robot so that they have different names.

(2) Restrictions

There are some restrictions at creating the FB body program comparing with an ordinary ladder program.

(a) Functional instructions

There are restrictions and notes on some functional instructions.

The following functional instructions cannot be used in the FB body program:

- END1, END2, END3, END
- SP, SPE
- CALL, CALLU
- JMPC
- CS, CM, CE

Do not use the following functional instructions in a function block of which more than one instance is to be created because they cannot perform independent operations for different function block instances:

- TMR
- CTR, CTRB

When the following functional instructions are used in a function block, set 0 to the timer number, and rising and falling numbers with FANUC LADDER-III for Robot so that the automatic number assignment function assigns different numbers for different function block instances:

- TMRB, TMRBF
- DIFU, DIFD

NOTE

If these functional instructions are used in a program without using the automatic number assignment function, these instructions may not work correctly because more than one functional instruction having the same number may operate simultaneously.

When the following functional instructions are used in a FB body program, time is integrated only while the FB body program is called:

- TMRST, TMRSS

(b) Other restrictions

In addition, the following restrictions apply on the FB body program:

- A value cannot be written to an input parameter.
- The JMP and JMPE instructions and the COM and COME instructions must be paired within a function block.
- The JMPB instruction can jump only to the LBL instruction within the function block.
- A program consisting of up to 8000 steps can be created in a function block.
- Because a FB body program is not executed when the EN input is OFF, it is referred as always ON in the case of it is used in the FB body program. So, the EN input can not be used as the input signal of DIFU/DIFD, -|P|-, -|N|- and counter instructions to catch rising and falling edge.

8.2.4 Other Information

A function block definition also contains the following information:

- Version information
- Protection information

The following explains the above information.

(1) Version information

The following information is included as version information in a function block definition:

- Character string indicating the user definition version (character string consisting of up to 16 desired characters)
- Last update time stamp

These information items are used for managing the function block in a library. They are also used as criteria at an identity check function for function blocks during recompilation on FANUC LADDER-III for Robot.

NOTE

FANUC LADDER-III for Robot automatically records the last update time stamp.
You do not need specific setting or operation.

(2) Protection information

Protection information is used to protect a function block definition from editing or browsing with a password.

After a password for protection is set, the password is required when the function block definition is to be edited or the FB body program is to be browsed (displayed in the monitor).

Protection information set in a function block definition is inherited to each function block instance generated from the function block definition.

There are the following two types of password protection:

- Editing protection
- Browsing and editing protection

Select editing or browsing and editing protection and set a password to protect the function block definition. You can use a character string consisting of up to any 16 desired characters for the password.

According to the selected type of protection, the relevant operations are prohibited as listed in the table below.

Type of protection	Browsing	Editing
No protection	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Editing protection	<input checked="" type="radio"/> Enabled	<input type="checkbox"/> Disabled
Browsing and editing protection	<input type="checkbox"/> Disabled	<input type="checkbox"/> Disabled

Each protection setting prohibits the following operations.

Type of protection	Example of prohibited operation
Editing protection	Editing of the function block definition (Deletion of the function block definition itself is possible.)

Type of protection	Example of prohibited operation
Browsing and editing protection	Display and monitor display of the FB body program

NOTE

The FB body program can be displayed and monitored on FANUC LADDER-III for Robot, and can be edited only on FANUC LADDER-III for Robot in the offline mode.

Protection with a password can be released by entering the password to enable the relevant operation temporarily.

8.3 FUNCTION BLOCK CALL

To use a defined function block actually, insert an instruction (instance) which calls the function block in a program and set signals and other items in the input and output section to complete the calling section. An object code which calls the specified function block processing is generated based on the information at compilation on FANUC LADDER-III for Robot.

⚠ CAUTION

When editing FB definitions and FB instances by the offline editing function on FANUC LADDER-III for Robot, the addresses assigned to the parameters and variables of function block will be changed. Therefore, the parameters and variables of all function blocks will be initialized by 0 when the sequence program is inputted into Robot controller. (See "8.1.4 Assignment of FB variable" for details.)

8.3.1 Function Block Call Positions

This section explains about difference by the positions of function block call.

(1) Program levels

A function block can be called from any position in level 1 to 3 ladder programs and subprograms.

(2) Common line control

When the ACT condition of the COM instruction is OFF, a function block call between COM and COME is not called and the processing in the function block is not executed.

This is the same effect as when the EN input is set to OFF. For a function block without EN input, you can use common line control to control a conditional function block call, which is similar to using the EN input.

8.3.2 Creating a Function Block Call Section

Follow the procedure below to create a function block call section:

1. Enable the reference to the definition of a function block to be called.
2. Choose the function block definition and create a function block call section in the program.
3. Assign a name (instance name) to the function block call section.
4. Set a value, address, or symbol for each parameter.

NOTE

A total of function block instance which can be created in a program is as follows.

- Up to 5000, in the case of PMC Memory-B,C and D
- Up to 512, in the case of PMC Memory-A

This number includes function blocks called from other function blocks.

The following explains the above procedure in detail.

(1) Name of a function block instance

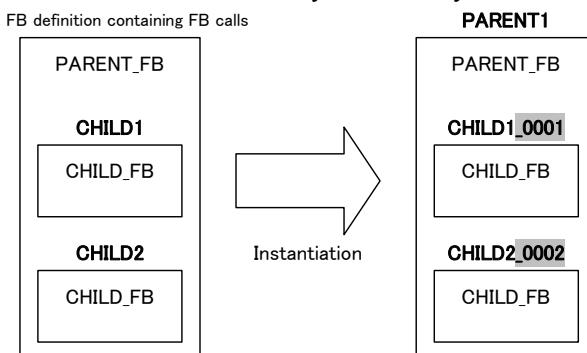
To insert an instruction which calls a function block in a ladder program, a name must be assigned to the instance to be created at that time. Instance names are assigned to distinguish individual instances when the same function block is called plurally in a program. The same name cannot be assigned to more than one instance.

For the name of a function block instance, specify a character string consisting of the following characters (identifier conforming to IEC61131-3):

- Alphabetic characters (A to Z)
- Numeric characters (0 to 9)
- Underscore (_)

NOTE

- 1 A function block instance name must not begin with a numeric character. When an underscore is specified as the first character, it must be followed by an alphanumeric character.
- 2 A name character string can consists of up to 40 characters.
- 3 When a function block contains a function block call instance, underscore (_) + 4-digit number is automatically added to the instance name in the function block definition during compilation on FANUC LADDER-III for Robot so that the name uniquely identifies the instance. For this reason, 5 characters ("_nnnn") are added to a function block instance name in a function block definition and the maximum number of characters of the instance name becomes 35. If a name to which a number is added is the name of another symbol, a compilation error occurs. Do not use any similar symbols.



(2) Setting data in the input and output section

After inputting a function block instance, set a numeric value or address to each parameter of the function block in the input or output section.

The available range differs depending on the type of parameter. For details, see "Parameters" in Subsection 11.2.2.

For a non-BOOL parameter, specify an address to the right or left side of the parameter name. For an input parameter, you can also specify a constant.

R0120	SETTING VALUE1 0
10000	SETTING VALUE2 10000

Fig.8.3.2 (a) Specifying an address (upper) and a constant (lower) for parameters

For a BOOL parameter, a contact is displayed in the input section. Specify an address on the contact. And, you can add coils, contacts and connection lines if needed.



Fig.8.3.2 (b) Specifying a contact and a coil for BOOL parameters

For an input/output parameter, no contact is displayed in the input section even when the data type is BOOL. Directly specify an address in the same way as for other data types of parameters.

For the output section for an output parameter or input/output parameter, you can omit the address specification if the output value does not need to be saved.

NOTE

The consistency of the data type between the symbol set to the parameter and the parameter itself is basically not checked. Combination of the BOOL and non-BOOL types causes an error. Any combination of a numeric type (such as INT or DINT) and a bit set type (such as BYTE) is available.

When a symbol of a different type is set for an input or output parameter, data of the size which suites to the type of parameter is actually input or output. Note that if data is input and output in different sizes, the program may not work as expected.

8.4 EXECUTING A FUNCTION BLOCK

A function block call section is executed in the following three steps:

1. Input process
2. Execution of the FB body program
3. Output process

The following explains the processing performed in each step in detail.

(1) Input process

In input process, given signals and numeric values are set to input parameters. The values are sequentially set for the input parameters from the top to the bottom.

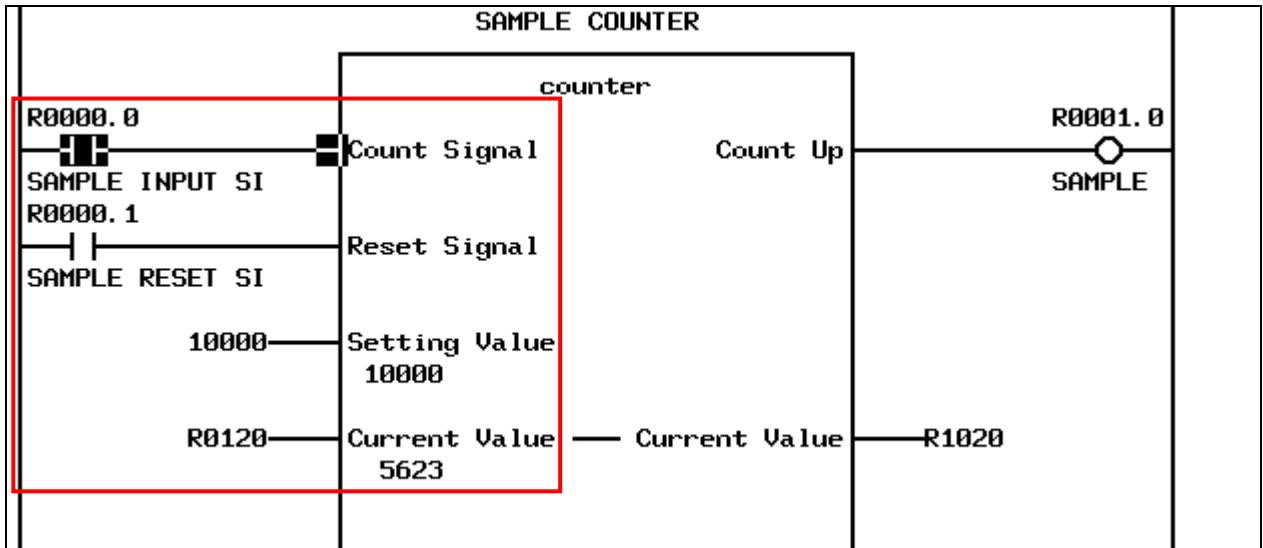


Fig.8.4 (a) Input process

In the example in this figure, input process will be performed as follows:

1. The signal status of R0000.0 is copied into input parameter “count signal”.
2. The signal status of R0000.1 is copied into input parameter “reset signal”.
3. The value 10000 is copied into input parameter “setting value”.

The address of input/output parameter “current value” itself will be R0120 and the value is not copied.

NOTE

When a function block to be executed has the EN input and the EN input is OFF, value is set for the subsequent input parameters but the subsequent execution step of the FB body program is skipped. For details of the EN input, see Section 8.2.2, “Variable Information”.

- (2) Execution of the FB body program

After values are set to all input parameters by input process, the FB body program is executed.

NOTE

When a function block to be executed has the EN input and the EN input is OFF, the FB body program is not executed. For details of the EN input, see Section 8.2.2, “Variable Information”.

- (3) Output process

After the FB body program has been executed, output process is performed.

In output processing, the values of the output parameters are set to the addresses connected to these output parameters. The values of the output parameters are sequentially set from the top to the bottom.

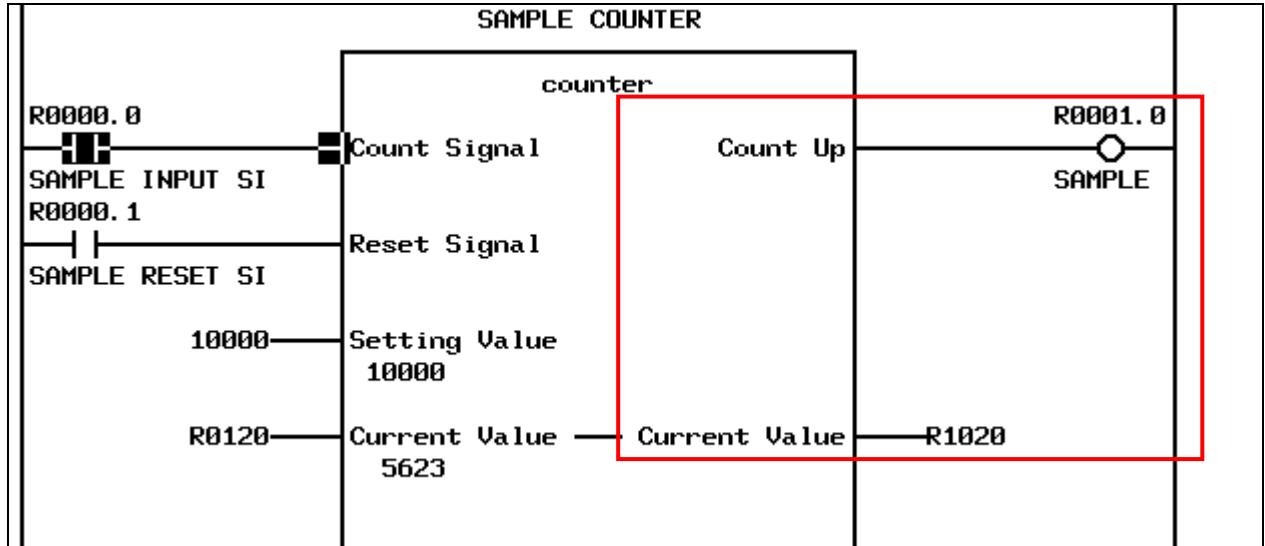


Fig.8.4 (b) Output process

In the example in this figure, output process will be performed as follows:

1. The signal status of output parameter “count up” is copied into R0001.0.
2. The value of input/output parameter “current value” is copied into R1020.

NOTE

- 1 Output process is performed in the order in which parameters are arranged. When the order in which values are set is important, change the order of parameters or use input/output parameters to adjust the timing to set values.
- 2 When a value is set to an input/output parameter in the FB body program, the value is set directly to the address specified in the input section of the input/output parameter. Then the value of output and input/output parameter is set to an address in the output section at output process.

INDEX

<A>

ABSSB (Absolute Value (1 Byte Length) : SUB 340)	341
ABSSD (Absolute Value (4 Bytes Length) : SUB 342).....	341
ABSSW (Absolute Value (2 Bytes Length) : SUB 341)	341
ADD (BCD Addition: SUB 19)	319
ADDB (Binary Addition: SUB 36).....	312
Addresses for Multi-path PMC Interface (M, N)	67
Addresses for Signals Between the PMC and ROBOT (F, G)	52
Addresses of Signals Between the PMC and External I/O Device (X, Y).....	52
ADDSB (Addition (1 Byte Length) : SUB 319)	328
ADDSD (Addition (4 Bytes Length) : SUB 321).....	328
ADDSW (Addition (2 Bytes Length) : SUB 320)	328
AND (Logical AND: SUB 60).....	241
AND Instruction.....	126
AND.NOT Instruction.....	127
AND.STK Instruction	132
ANDB (Logical AND (1 Byte Length) : SUB 268)	251
ANDD (Logical AND (4 Bytes Length) : SUB 270)	251
ANDNT Instruction	141
ANDPT Instruction	137
ANDW (Logical AND (2 Bytes Length) : SUB 269)	251
Assignment of FB Variable	476

Basic Configuration of PMC.....	1
Basic Instructions.....	38
BASIC INSTRUCTIONS	119
Basic Specifications	33
BCNTB (Bit Count (1 Byte Length) : SUB 309)	292
BCNTD (Bit Count (4 Bytes Length) : SUB 311)	292
BCNTN (Bit Count (Arbitrary Bytes Length) : SUB 312)	293
BCNTW (Bit Count (2 Bytes Length) : SUB 310)	292
Bit Menu	412
BIT OPERATION.....	237
Block Step.....	461
BPOSB (Bit Search (1 Byte Length) : SUB 305)	288
BPOSD (Bit Search (4 Bytes Length) : SUB 307).....	288

BPOSN (Bit Search (Arbitrary Bytes Length) : SUB 308)	290
BPOSW (Bit Search (2 Bytes Length) : SUB 306)	288
BRSTB (Bit Reset (1 Byte Length) : SUB 297)	281
BRSTD (Bit Reset (4 Bytes Length) : SUB 299).....	281
BRSTN (Bit Reset (Arbitrary Bytes Length) : SUB 300)	283
BRSTW (Bit Reset (2 Bytes Length) : SUB 298)	281
BSETB (Bit Set (1 Byte Length) : SUB 293)	277
BSETD (Bit Set (4 Bytes Length) : SUB 295).....	277
BSETN (Bit Set (Arbitrary Bytes Length) : SUB 296).....	279
BSETW (Bit Set (2 Bytes Length) : SUB 294)	277
BTSTB (Bit Test (1 Byte Length) : SUB 301)	285
BTSTD (Bit Test (4 Bytes Length) : SUB 303).....	285
BTSTN (Bit Test (Arbitrary Bytes Length) : SUB 304)	286
BTSTW (Bit Test (2 Bytes Length) : SUB 302)	285
Byte Menu.....	410

<C>

CALL (Conditional Subprogram Call: SUB 65).....	353
CALLU (Unconditional Subprogram Call: SUB 66)	354
Cautions for Reading from/Writing to Nonvolatile Memory	70
CE (End of Case Call: SUB 76)	360
Change PMC Path.....	427
Checking Sequence Program	11
CM (Sub Program Call in Case Call: SUB 75)	359
COD (Code Conversion: SUB 7)	295
CODB (Binary Code Conversion: SUB 27).....	298
CODE CONVERSION	295
COIN (Coincidence Check: SUB 16)	235
COM (Common Line Control: SUB 9)	345
COME (Common Line Control End: SUB 29)	347
Comment.....	5
Common PMC Memory Mode of Multi-Path PMC	29
COMP (Comparison: SUB 15)	234
COMPARISON	221
COMPATIBILITY BETWEEN PMC MEMORY TYPE.....	76
Compatibility between PMC Memory-A and PMC Memory-B	76
Compatibility between PMC Memory-B and PMC Memory-C/D	76

Compatibility Setup Function	115	DIV (BCD Division: SUB 22)	324
COMPATIBILITY WITH CONVENTIONAL MODELS	113	DIVB (Binary Division: SUB 39)	318
Compatibility with PMC Memory-C and PMC Memory-D	77	Divergence of Selective Sequence	457
COMPB (Comparison Between Binary Data: SUB 32)	232	Divergence of Simultaneous Sequence	458
COMPILE A PMC PROGRAM	382	DIVSB (Division (1 Byte Length) : SUB 328)	334
CONFIGURATION AND OPERATION OF STEP-SEQUENCE PROGRAMS	454	DIVSD (Division (4 Bytes Length) : SUB 330)	334
CONNECTING FANUC LADDER-III for Robot TO ROBOT CONTROLLER	367	DIVSW (Division (2 Bytes Length) : SUB 329)	334
Connection with ROBOGUIDE	375	DMAXB (Maximum Data (1 Byte Length): SUB 259)	215
Convergence of Selective Sequence	458	DMAXD (Maximum Data (4 Bytes Length) : SUB 261)	215
Convergence of Simultaneous Sequence	459	DMAXW (Maximum Data (2 Bytes Length) : SUB 260)	215
COUNTER	160	DMINB (Minimum Data (1 Byte Length): SUB 262)	218
Counter Addresses (C)	59	DMIND (Minimum Data (4 Bytes Length): SUB 264)	218
Counter Menu	414	DMINW (Minimum Data (2 Bytes Length): SUB 263)	218
Creating a Function Block Call Section	487	DSCH (Data Search: SUB 17)	198
Creating Ladder Diagram	9	DSCHB (Binary Data Search: SUB 34)	195
CREATING PMC PROGRAM	379	DSEQB (Searching Data from Table(=)(1 Byte Length):SUB 241)	211
CS (Case Call: SUB 74)	357	DSEQD (Searching Data from Table(=)(4 Bytes Length):SUB 243)	211
CTR (Counter: SUB 5)	160	DSEQW (Searching Data from Table(=)(2 Bytes Length):SUB 242)	211
CTRB (Fixed Counter: SUB 56)	164	DSGEB (Searching Data from Table(\geq)(1 Byte Length):SUB 253)	211
CTRC (Counter: SUB 55)	166	DSGED (Searching Data from Table(\geq)(4 Bytes Length) :SUB 255)	211
CTRD (Counter (4 Bytes Length) : SUB 223)	168	DSGEW (Searching Data from Table(\geq)(2 Bytes Length):SUB 254)	211
<D>		DSGTB (Searching Data from Table(>)(1 Byte Length):SUB 247)	211
Data Table Addresses (D)	63	DSGTD (Searching Data from Table(>)(4 Bytes Length):SUB 249)	211
Data Table Control Data Menu	415	DSGTW (Searching Data from Table(>)(2 Bytes Length):SUB 248)	211
Data Table Menu	416	DSLEB (Searching Data from Table(\leq)(1 Byte Length) :SUB 256)	211
DATA TRANSFER	171	DSLED (Searching Data from Table(\leq)(4 Bytes Length) :SUB 258)	211
DCNV (Data Conversion: SUB 14)	300	DSLEW (Searching Data from Table(\leq)(2 Bytes Length) :SUB 257)	211
DCNVB (Extended Data Conversion: SUB 31)	301	DSLTB (Searching Data from Table(<)(1 Byte Length):SUB 250)	211
DEC (Decode: SUB 4)	303		
DECB (Binary Decoding: SUB 25)	305		
DECSD (Decrement (1 Byte Length) : SUB 337)	340		
DECSD (Decrement (4 Bytes Length) : SUB 339)	340		
DECSW (Decrement (2 Bytes Length) : SUB 338)	340		
Details of the Basic Instructions	120		
Determination of PMC Memory Type	35		
Determining Specification	9		
DIFD (Falling Edge Detection: SUB 58)	239		
Difference Between Relay Sequence Circuit and Ladder Sequence Program	6		
DIFU (Rising Edge Detection: SUB 57)	238		

DSLTD (Searching Data from Table(<)(4 Bytes Length):SUB 252)	211
DSLTW (Searching Data from Table(<)(2 Bytes Length):SUB 251)	211
DSNEB (Searching Data from Table(≠)(1 Byte Length):SUB 244)	211
DSNED (Searching Data from Table(≠)(4 Bytes Length):SUB 246)	211
DSNEW (Searching Data from Table(≠)(2 Bytes Length):SUB 245)	211

<E>

Editing and Debugging Step Sequence Programs	443
Editing Sequence Program	10
EDITTING A PMC PROGRAM	381
END (End of a Ladder Program: SUB 64).....	357
End Of Block Step	462
END1 (1st Level Sequence Program End: SUB 1).....	356
END2 (2nd Level Sequence Program End: SUB 2).....	356
END3 (3rd Level Sequence Program End: SUB 48).....	356
EOR (Exclusive OR: SUB 59).....	240
EORB (Exclusive OR (1 Byte Length) : SUB 265)	249
EORD (Exclusive OR (4 Bytes Length) : SUB 267)	249
EORW (Exclusive OR (2 Bytes Length) : SUB 266)	249
EQB (1 Byte Length: SUB 200)	222
EQD (4 Bytes Length: SUB 202).....	222
EQW (2 Bytes Length: SUB 201)	222
Ethernet Connection.....	371
Example 1 : UOP is Controlled by External I/O Device	104
Example 2 : UOP is controlled by PMC	108
EXAMPLE OF PMC I/O ASSIGNMENT	104
Exclusive Control for Functional Instructions.....	469
EXECUTING A FUNCTION BLOCK.....	489
EXECUTION OF SEQUENCE PROGRAM.....	11
Execution of Step Sequence	452
Execution Order and Execution Time Percentage.....	27
Execution Procedure of Sequence Program	12
Exporting LADDER*.PMC using FANUC LADDER-III for Robot	404
EXTENDED LADDER INSTRUCTIONS	462
External I/O Assignment Menu.....	422
Extra Relay Addresses (E)	57

<F>

FANUC LADDER-III FOR ROBOT PROGRAMMING.....	366
FB Body Program	484
FBCDB (BCD to Binary Conversion (1 Byte Length) : SUB 313)	309
FBCDD (BCD to Binary Conversion (4 Bytes Length) : SUB 315).....	309
FBCDW (BCD to Binary Conversion (2 Bytes Length) : SUB 314)	309
FILE MENU OPERATIONS	429
Format of the Functional Instructions	145
FUNCTION BLOCK CALL.....	487
Function Block Call Positions.....	487
FUNCTION BLOCK DEFINITION.....	477
FUNCTION BLOCK FUNCTION	471
Function Block Name	477
Functional Instruction TRSET	462
FUNCTIONAL INSTRUCTIONS.....	145
Functional Instructions (Arranged in Sequence of Instruction Group)	39
Functional Instructions (Arranged in Sequence of SUB No.)	46

<G>

GEB (1 Byte Length: SUB 212)	228
GED (4 Bytes Length: SUB 214).....	228
General Rules.....	465
GEW (2 Bytes Length: SUB 213)	228
Graphic Symbols of Relays and Coils.....	5
Graphical Symbols.....	442
GTB (1 Byte Length: SUB 206)	225
GTD (4 Bytes Length: SUB 208).....	225
GTW (2 Bytes Length: SUB 207)	225

</>

I/O ASSIGNMENT ON ROBOT CONTROLLER	78
I/O Signals of PMC.....	2
Implementation	15
Importing LADDER*.PMC using FANUC LADDER-III for Robot	406
INCSB (Increment (1 Byte Length) : SUB 334)	338
INCSD (Increment (4 Bytes Length) : SUB 336)	338
INCSW (Increment (2 Bytes Length) : SUB 335)	338
Initial Block Step	461
Initial Step.....	456

Interlock	26	MOVW (Transfer of 2 Bytes: SUB 44)	172
Internal I/O Assignment Menu	424	MUL (BCD Multiplication: SUB 21)	322
Internal Relay Addresses (R)	52	MULB (Binary Multiplication: SUB 38)	316
INVALID INSTRUCTIONS.....	365	MULSB (Multiplication (1 Byte Length) : SUB 325)	332
Item Names	472	MULSD (Multiplication (4 Bytes Length) : SUB 327).....	332
		MULSW (Multiplication (2 Bytes Length) : SUB 326)	332
<J>		MULTI-PATH PMC FUNCTION	26
JMP (Jump: SUB 10)	347	Multi-Path PMC Interface.....	28
JMPB (Label Jump 1: SUB 68).....	349		
JMPC (Label Jump 2: SUB 73).....	351		
JMPE (Jump End: SUB 30).....	349		
Jump.....	460		
<K>			
Keep Relay Addresses (K)	61	NEB (1 Byte Length: SUB 203)	224
<L>		NED (4 Bytes Length: SUB 205).....	224
Label	460	NEGSB (Sign Inversion (1 Byte Length) : SUB 343)	343
Label Number Addresses (L)	68	NEGSD (Sign Inversion (4 Bytes Length) : SUB 345).....	343
Ladder Diagram Format	4	NEGSW (Sign Inversion (2 Bytes Length) : SUB 344)	343
LADDER LANGUAGE.....	119	NEW (2 Bytes Length: SUB 204)	224
LBL (Label: SUB 69).....	352	NOP (No Operation: SUB 70)	357
LEB (1 Byte Length: SUB 215)	229	NOT (Logical NOT: SUB 62).....	244
LED (4 Bytes Length: SUB 217)	229	NOTB (Logical NOT (1 Byte Length) : SUB 274)	255
LEW (2 Bytes Length: SUB 216)	229	NOTD (Logical NOT (4 Bytes Length) : SUB 276).....	255
Line Number and Net Number	5	NOTW (Logical NOT (2 Bytes Length) : SUB 275)	255
Load LADDER*.PMC, PARAM*.PMC, PMCCFG.SV	431	NUME (BCD Definition of Constant: SUB 23).....	327
LTB (1 Byte Length: SUB 209)	226	NUMEB (Definition of Binary Constants: SUB 40).....	325
LTD (4 Bytes Length: SUB 211)	226		
LTW (2 Bytes Length: SUB 210)	226		
<M>			
Memory Usage Related to Function Blocks.....	475	Occupancy Setting	85
Message Display Addresses (A).....	58	OPERATION INSTRUCTION.....	312
MODIFYING THE PMC PROGRAM IN THE ROBOT		OR (Logical OR: SUB 61).....	243
CONTROLLER.....	401	OR Instruction.....	128
MODSB (Remainder (1 Byte Length) : SUB 331)	336	OR.NOT Instruction.....	129
MODSD (Remainder (4 Bytes Length) : SUB 333).....	336	OR.STK Instruction	133
MODSW (Remainder (2 Bytes Length) : SUB 332)	336	ORB (Logical OR (1 Byte Length) : SUB 271)	253
MOVB (Transfer of 1 Byte: SUB 43).....	171	ORD (Logical OR (4 Bytes Length) : SUB 273)	253
MOVBT (Bit Transfer: SUB 224).....	186	ORNT Instruction	142
MOVD (Transfer of 4 Bytes: SUB 47)	173	ORPT Instruction	138
MOVE (Logical Product Transfer: SUB 8).....	175	ORW (Logical OR (2 Bytes Length) : SUB 272)	253
MOVN (Transfer of an Arbitrary Number of Bytes:		Other Information	486
SUB 45).....	174	Override Function	83
MOVOR (Data Transfer After Logical Sum: SUB 28).....	176	OVERVIEW	440,471
		OVERVIEW OF PMC.....	1
		Overview of Specifications	472
<P>			
PARAMETERS FOR THE PMC SYSTEM	75		
Parameters Menu.....	418		
PARI (Parity Check: SUB 11)	246		

PMC Address (S Address)	463	Robot : GO → PMC : F	91
PMC Addresses	37	Robot : GO,AO ← PMC : G	102
PMC EDIT FUNCTION	436	Robot : GO[10001-11500] → PMC1: D0-2999	103
PMC EXTERNAL I/O ASSIGNMENT	78	Robot : INFO → PMC : F	94
PMC I/O ASSIGNMENT	78	Robot : Register → PMC : F	96
PMC INTERNAL I/O ASSIGNMENT	87	Robot : Register ← PMC : G	97
PMC Menus	409	Robot : UALM ← PMC : G	95
PMC MONITOR FUNCTION	433	Robot : UI ← PMC : G	93
PMC Parameter Format	70	Robot : UO → PMC : F	92
PMC PARAMETERS	68	ROLB (Bit Rotation Left (1 Byte Length) : SUB 285)	267
PMC PROGRAM MONITORING (ONLINE MONITOR)	387	ROLD (Bit Rotation Left (4 Bytes Length) : SUB 287)	267
PMC Setting Menu	420	ROLN (Bit Rotation Left (Arbitrary Bytes Length) : SUB 288)	270
PMC Signal Addresses	2	ROLW (Bit Rotation Left (2 Bytes Length) : SUB 286)	267
PMC SIGNAL ADDRESSES	52	RORB (Bit Rotation Right (1 Byte Length) : SUB 289)	272
PMC SPECIFICATIONS	33	RORD (Bit Rotation Right (4 Bytes Length) : SUB 291)	272
Processing Priority (1st Level, 2nd Level, and 3rd Level)	13	RORN (Bit Rotation Right (Arbitrary Bytes Length) : SUB 292)	275
Program Capacity	35	RORW (Bit Rotation Right (2 Bytes Length) : SUB 290)	272
PROGRAM CONTROL	344	ROT (Rotation Control: SUB 6)	360
PUSH Instruction / POP Instruction	144	ROTATION CONTROL	360
<R>		ROTB (Binary Rotation Control: SUB 26)	363
RD Instruction	122	RS-232-C Connection	367
RD.NOT Instruction	123	RST Instruction	135
RD.NOT.STK Instruction	131	Run/Stop PMC	426
RD.STK Instruction	130	RUNNING OR STOPPING PMC PROGRAM	399
RDNT Instruction	140		
RDNT.STK Instruction	143		
RDPT Instruction	136		
RDPT.STK Instruction	139		
Relationship with I/O Assignment (Robot)	82		
Repetitive Operation	13		
RNGB (1 Byte Length: SUB 218)	230		
RNGD (4 Bytes Length: SUB 220)	230		
RNGW (2 Bytes Length: SUB 219)	230		
Robot : DI ← PMC : G	90		
Robot : DI,RI,WI,WSI,SI,*UI → PMC : X	98		
Robot : DO → PMC : F	89		
Robot : DO,RO,WO,WSO,SO,*UO ← PMC : Y	99		
Robot : DO[10001-10136] → PMC1: K0-17	103		
Robot : DO[10137-10160] → PMC1: K900-902	103		
Robot : DO[11001-23000] → PMC1: R0-1499	103		
Robot : GI ← PMC : G	91		
Robot : GI,AI → PMC : F	101		
<S>		SAFETY PRECAUTIONS	s-1
Save LADDER*.PMC and PARAM*.PMC	428	Save LADDER*.PMC, PARAM*.PMC, PMCCFG.SV	430
SAVE/LOAD PMC RELATED DATA	31	SEQUENCE PROGRAM CREATION PROCEDURE	9
Search	425	SET Instruction	134
SETNB (Data Setting (1 Byte Length) : SUB 225)	189	SETND (Data Setting (4 Bytes Length) : SUB 227)	189
SETNW (Data Setting (2 Bytes Length) : SUB 226)	189	SETNW (Data Setting (2 Bytes Length) : SUB 226)	189
Setting Parameters	75	SFT (Shift Register: SUB 33)	247
SHLB (Bit Shift Left (1 Byte Length) : SUB 277)	257	SHLB (Bit Shift Left (1 Byte Length) : SUB 277)	257

SHLD (Bit Shift Left (4 Bytes Length) : SUB 279).....	257
SHLN (Bit Shift Left (Arbitrary Bytes Length) : SUB 280)	260
SHLW (Bit Shift Left (2 Bytes Length) : SUB 278)	257
SHRB (Bit Shift Right (1 Byte Length) : SUB 281)	262
SHRD (Bit Shift Right (4 Bytes Length) : SUB 283)	262
SHRN (Bit Shift Right (Arbitrary Bytes Length) : SUB 284)	265
SHRW (Bit Shift Right (2 Bytes Length) : SUB 282)	262
Signal Name (Symbol Name).....	5
Signed Binary Comparison (\neq)	224
Signed Binary Comparison ($<$)	226
Signed Binary Comparison ($=$)	222
Signed Binary Comparison ($>$)	225
Signed Binary Comparison (\leq)	229
Signed Binary Comparison (\geq)	228
Signed Binary Comparison (Range)	230
SP (Subprogram: SUB 71)	354
SPE (End of a Subprogram: SUB 72)	355
Specification.....	464
Specification of Extended Symbol and Comment.....	7
SPECIFICATION OF STEP SEQUENCE	464
SPECIFICATIONS	33
Status Menu.....	419
Step	454
STEP SEQUENCE BASICS	444
STEP SEQUENCE FUNCTION	440
Step Sequence Method.....	440
Step Status Menu.....	419
Storage and Management of Sequence Program	11
STORE/LOAD PMC PROGRAM	404
Structured Sequence Program	15
SUB (BCD Subtraction: SUB 20)	321
SUBB (Binary Subtraction: SUB 37).....	314
Subprogram Number Addresses (P).....	68
Subprogramming and nesting.....	19
SUBSB (Subtraction (1 Byte Length) : SUB 322)	330
SUBSD (Subtraction (4 Bytes Length) : SUB 324)	330
SUBSW (Subtraction (2 Bytes Length) : SUB 323)	330
SWAPD (Data Swap (4 Bytes Length) : SUB 232)	193
SWAPW (Data Swap (2 Bytes Length) : SUB 231)	193
Synchronization Processing of I/O Signals	23
System Keep Relay Addresses (K)	62
System Relay Addresses (R9000, Z0).....	53

<T>

TABLE DATA.....	200
TBCDB (Binary to BCD Conversion (1 Byte Length) : SUB 313)	308
TBCDD (Binary to BCD Conversion (4 Bytes Length) : SUB 315).....	308
TBCDW (Binary to BCD Conversion (2 Bytes Length) : SUB 314)	308
TBLRB (Reading Data from Table (1 Byte Length) : SUB 233)	201
TBLRD (Reading Data from Table (4 Bytes Length) : SUB 235).....	201
TBLRN (Reading Data from Table (Arbitrary Bytes Length) : SUB 236)	203
TBLRW (Reading Data from Table (2 Bytes Length) : SUB 234)	201
TBLWB (Writing Data to Table (1 Byte Length) : SUB 237)	206
TBLWD (Writing Data to Table (4 Bytes Length) : SUB 239)	206
TBLWN (Writing Data to Table (Arbitrary Bytes Length) : SUB 240)	208
TBLWW (Writing Data to Table (2 Bytes Length) : SUB 238)	206
TEACH PENDANT OPERATION	409
Terminology.....	444
The Convert Method of Source Program Using FANUC LADDER-III for Robot	114
TIMER	150
Timer Addresses (T)	58
Timer Menu	413
Title Menu.....	420
TMR (On-delay Timer: SUB 3)	150
TMRB (Fixed On-delay Timer: SUB 24)	151
TMRBF (Fixed Off-delay Timer: SUB 77)	153
TMRC (On-delay Timer: SUB 54)	154
TMRSS (Stop Watch Timer (1sec Accuracy) : SUB 222)	156
TMRST (Stop Watch Timer (1ms Accuracy) : SUB 221)	156
Transferring and Writing Sequence Program to PMC	11
TRANSFERRING PMC PROGRAM.....	383
Transition	457

<U>

Used Memory Size of Sequence Program.....36

<V>

Variable Information.....477

<W>

WHAT IS LADDER LANGUAGE?4

WHAT IS PMC?1

WRITING PMC PROGRAM TO F-ROM.....400

WRT Instruction.....124

WRT.NOT Instruction125

<X>

XCHGB (Data Exchange (1 Byte Length) : SUB 228)191

XCHGD (Data Exchange (4 Bytes Length) : SUB 230)191

XCHGW (Data Exchange (2 Bytes Length) : SUB 229)

.....191

XMOV (Indexed Data Transfer: SUB 18)184

XMOVB (Binary Index Modifier Data Transfer: SUB

35)177

REVISION RECORD

Edition	Date	Contents
02	Sep., 2013	Addition of R-30iB Mate.
01	Apr., 2012	

B-83254EN/02



* B - 8 3 2 5 4 E N / 0 2 *