TOSHIBA
Leading Innovation

UM-V200-E002

Programmable Logic Controllers

# USER'S MANUAL
# LADDER LOGIC

V200 Series PLC & OIS PLUS

**Toshiba International Corporation**

Thank you for purchasing the V200 Series PLC (Programmable Logic Controller) product from Toshiba International Corp. V200 Series products are versatile PLCs which are configured with Microsoft Windows® based software.

# Manual’s Purpose and Scope

This manual provides information on how to safely install, operate, and maintain your TIC V200 Series PLC. This manual includes a section of general safety instructions that describes the warning labels and symbols that are used throughout the manual. Read the manual completely before installing, operating, or performing maintenance on this equipment.

This manual and the accompanying drawings should be considered a permanent part of the equipment and should be readily available for reference and review. Dimensions shown in the manual are in metric and/or the English equivalent.

Toshiba International Corporation reserves the right, without prior notice, to update information, make product changes, or to discontinue any product or service identified in this publication.

TOSHIBA is a registered trademark of the Toshiba Corporation. All other product or trade references appearing in this manual are registered trademarks of their respective owners.

**Toshiba International Corporation (TIC) shall not be liable for technical or editorial omissions or mistakes in this manual, nor shall it be liable for incidental or consequential damages resulting from the use of information contained in this manual.**

# Important Notice

The instructions contained in this manual are not intended to cover all details or variations in equipment types, nor may it provide for every possible contingency concerning the installation, operation, or maintenance of this equipment. Should additional information be required contact your Toshiba representative.

The contents of this manual shall not become a part of or modify any prior or existing agreement, commitment, or relationship. The sales contract contains the entire obligation of Toshiba International Corporation. The warranty contained in the contract between the parties is the sole warranty of Toshiba International Corporation and any statements contained herein do not create new warranties or modify the existing warranty.

Any electrical or mechanical modifications to this equipment without prior written consent of Toshiba International Corporation will void all warranties and may void the 3rd party (CE, UL, CSA, etc) safety certifications. Unauthorized modifications may also result in a safety hazard or equipment damage.

# Contacting Toshiba's Customer Support Center

Toshiba's Customer Support Center may be contacted to obtain help in resolving any system problems that you may experience or to provide application information.

The center is open from 8 a.m. to 5 p.m. (CST), Monday through Friday. The Support Center's toll free number is US 800-231-1412 Fax 713-466-8773 — Canada 800-527-1204 — Mexico 01-800-527-1204.

You may also contact Toshiba by writing to:

Toshiba International Corporation
13131 West Little York Road
Houston, Texas 77041-9990
Attn: PLC Marketing
Or email
plc@tic.toshiba.com.

For further information on Toshiba's products and services, please visit our website at **www.toshiba.com/ind/**.

# Manual Revisions

Please have the following information available when contacting Toshiba International Corp. about this manual.

**Name:** V200 User's Manual
**Document:** UM-V200-E001
**Revision:**

| Rev No. | Date | Description |
|---|---|---|
| 0 | 2012/01/1 | Initial Issue (for V200) |

# TABLE OF CONTENTS

# General Safety Instructions and Information

## 0.1 Warning Labels Within Manual

**DO NOT** attempt to install, operate, maintain, or dispose of this equipment until you have read and understood all of the product warnings and user directions that are contained in this instruction manual.

Listed below are the signal words that are used throughout this manual followed by their descriptions and associated symbols. When the words **DANGER**, **WARNING**, and **CAUTION** are used in the manual, they will be followed by important safety information that must be carefully adhered to.

**DANGER** — The danger symbol is an exclamation mark enclosed in a triangle that precedes the word DANGER. The danger symbol is used to indicate an imminently hazardous situation that will result in serious injury, possible severe property and equipment damage, or death if the instructions are not followed.

⚠ DANGER

**WARNING** – The warning symbol is an exclamation mark enclosed in a triangle that precedes the word WARNING. The warning symbol is used to indicate a potentially hazardous situation that can result in serious injury, or possibly severe property and equipment damage, or death, if the instructions are not followed.

⚠ WARNING

**CAUTION** – The caution symbol is an exclamation mark enclosed in a triangle that precedes the word CAUTION. The caution symbol is used to indicate situations that can result in minor or moderate operator injury, or equipment damage if the instructions are not followed.

⚠ CAUTION

To identify special hazards, other symbols may appear in conjunction with the **DANGER**, **WARNING**, and **CAUTION** symbols. These warnings describe areas that require special care and/or strict adherence to the procedures to prevent serious injury and possible death.

**Electrical Hazard —** The electrical hazard symbol is a lightning bolt enclosed in a triangle. The electrical hazard symbol is used to indicate high voltage locations and conditions that may cause serious injury or death if the proper precautions are not observed.

ELECTRICAL
HAZARD

**Explosion Hazard** — The explosion hazard symbol is an explosion image enclosed in a triangle. The explosion hazard symbol is used to indicate locations and conditions where molten exploding parts may cause serious injury or death if the proper precautions are not observed.

EXPLOSION
HAZARD

## 0.2 Equipment Warning Labels.

**DO NOT** attempt to install, operate, maintain, or dispose of this equipment until you have read and understood all of the product warnings and user directions that are contained in this instruction manual.

Shown below are examples of warning labels that may be found attached to the equipment. **DO NOT** remove or cover any of the labels. If the labels are damaged or if additional labels are required, contact your Toshiba representative for additional labels.

The following are examples of the warning labels that may be found on the equipment and are there to provide useful information or to indicate an imminently hazardous situation that may result in serious injury, severe property and equipment damage, or death if the instructions are not followed.

**Examples of labels that may be found on the equipment.**

## 0.3 Preparation

### Qualified Person

A **Qualified Person** is one that has the skills and knowledge relating to the construction, installation, operation, and maintenance of the electrical equipment and has received safety training on the hazards involved (Refer to the latest edition of NFPA 70E for additional safety requirements).

**Qualified Personnel** shall:

- Have carefully read the entire operation manual.
- Be trained and authorized to safely energize, de-energize, ground, lockout and tag circuits and equipment, and clear faults in accordance with established safety practices.
- Be trained in the proper care and use of protective equipment such as safety shoes, rubber gloves, hard hats, safety glasses, face shields, flash clothing, etc., in accordance with established safety practices.
- Be trained in rendering first aid.

**For further information on workplace safety visit www.osha.gov**.

### Equipment Inspection

- Upon receipt of the equipment inspect the packaging and equipment for shipping damage.
- Carefully unpack the equipment and check for parts that were damaged from shipping, missing parts, or concealed damage. If any discrepancies are discovered, it should be noted with the carrier prior to accepting the shipment, if possible. File a claim with the carrier if necessary and immediately notify your Toshiba representative.
- **DO NOT** install or energize equipment that has been damaged. Damaged equipment may fail during operation resulting in further equipment damage or personal injury.
- Check to see that the model number specified on the nameplate conforms to the order specifications.
- Modification of this equipment is dangerous and must not be performed except by factory trained representatives. When modifications are required contact your Toshiba representative.
- Inspections may be required before and after moving installed equipment.
- Keep the equipment in an upright position as indicated on the shipping carton.
- Contact your Toshiba representative for assistance if required.

### Handling and Storage

- Use proper lifting techniques when moving the OIS; including properly sizing up the load, and getting assistance if required.
- Store in a well-ventilated covered location and preferably in the original carton if the equipment will not be used upon receipt.
- Store in a cool, clean, and dry location. Avoid storage locations with extreme temperatures, rapid temperature changes, high humidity, moisture, dust, corrosive gases, or metal particles.
- Do not store the unit in places that are exposed to outside weather conditions (i.e., wind, rain, snow, etc.).
- Store in an upright position as indicated on the shipping carton.
- Include any other product-specific requirements.

### Disposal

Never dispose of electrical components via incineration. Contact your state environmental agency for details on disposal of electrical components and packaging in your area.

## 0.4 Installation Precautions

### Location and Ambient Requirements

- Adequate personnel working space and adequate illumination must be provided for adjustment, inspection, and maintenance of the equipment (refer to NEC Article 110-34).
- Avoid installation in areas where vibration, heat, humidity, dust, fibers, steel particles, explosive/corrosive mists or gases, or sources of electrical noise are present.
- The installation location shall not be exposed to direct sunlight.
- Allow proper clearance spaces for installation. Do not obstruct the ventilation openings. Refer to the recommended minimum installation dimensions as shown on the enclosure outline drawings.
- The ambient operating temperature shall be between 0° and 50° C (32° and 122° F).

### Mounting Requirements

- Only **Qualified Personnel** should install this equipment.
- Install the unit in a secure upright position in a well-ventilated area.
- A noncombustible insulating floor or mat should be provided in the area immediately surrounding the electrical system at the place where maintenance operations are to be performed.
- As a minimum, the installation of the equipment should conform to the NEC Article 110 Requirements For Electrical Installations, OSHA, as well as any other applicable national, regional, or industry codes and standards.
- Installation practices should conform to the latest revision of NFPA 70E Electrical Safety Requirements for Employee Workplaces.

Conductor Routing and Grounding

- Use separate metal conduits for routing the input power, and control circuits.
- A separate ground cable should be run inside the conduit with the input power, and control circuits.
- **DO NOT** connect control terminal strip return marked CC to earth ground.
- Always ground the unit to prevent electrical shock and to help reduce electrical noise.

**The Metal Of Conduit Is Not An Acceptable Ground.**

## 0.5 Connection, Protection & Setup

Personnel Protection

- Installation, operation, and maintenance shall be performed by **Qualified Personnel Only**.
- A thorough understanding of the OIS will be required before the installation, operation, or maintenance of the OIS.
- Rotating machinery and live conductors can be hazardous and shall not come into contact with humans. Personnel should be protected from all rotating machinery and electrical hazards at all times. Depending on its program, the OIS can initiate the start and stop of rotating machinery.
- Insulators, machine guards, and electrical safeguards may fail or be defeated by the purposeful or inadvertent actions of workers. Insulators, machine guards, and electrical safeguards are to be inspected (and tested where possible) at installation and periodically after installation for potential hazardous conditions.
- Do not allow personnel near rotating machinery. Warning signs to this effect shall be posted at or near the machinery.
- Do not allow personnel near electrical conductors. Human contact with electrical conductors can be fatal. Warning signs to this effect shall be posted at or near the hazard.
- Personal protection equipment shall be provided and used to protect employees from any hazards inherent to system operation or maintenance.

System Setup Requirements

- When using the OIS as an integral part of a larger system, it is the responsibility of the OIS installer or maintenance personnel to ensure that there is a fail-safe in place (i.e., an arrangement designed to switch the system to a safe condition if there is a fault or failure).
- System safety features should be employed and designed into the integrated system in a manner such that system operation, even in the event of system failure, will not cause harm or result in personnel injury or system damage (i.e., E-Off, Auto-Restart settings, System Interlocks, etc.).
- The programming setup and system configuration of the OIS may allow it to start a motor unexpectedly. A familiarity with Auto-restart settings is a requirement to use this product.
- Improperly designed or improperly installed system interlocks may render the motor unable to start or stop on command.

The failure of external or ancillary components may cause intermittent system operation, i.e., the system may start a motor without warning or may not stop on command.

- There may be thermal or physical properties, or ancillary devices integrated into the overall system that may allow the OIS to start a motor without warning. Signs at the equipment installation must be posted to this effect.
- The operating controls and system status indicators should be clearly readable and positioned where the operator can see them without obstruction.
- Additional warnings and notifications shall be posted at the equipment installation location as deemed required by **Qualified Personnel**.

## 0.6 System Integration Precautions

The following precautions are provided as general guidelines for using an OIS in an industrial or process control system.

- The Toshiba PLC is a general-purpose product. It is a system component and is used in conjunction with other items of industrial equipment such as PLCs, Loop Controllers, Adjustable Speed Drives, etc.
- **A detailed system analysis and job safety analysis should be performed by the systems designer or systems integrator before including the OIS in any new or existing system**. Contact Toshiba for options availability and for application-specific system integration information if required.
- The PLC may be used to control an adjustable speed drive connected to high voltage sources and rotating machinery that is inherently dangerous if not operated safely. Interlock all energy sources, hazardous locations, and guards in order to restrict the exposure of personnel to hazards. The adjustable speed drive may start the motor without warning. Signs at the equipment installation must be posted to this effect. A familiarity with Auto-restart settings is a requirement when controlling adjustable speed drives. Failure of external or ancillary components may cause intermittent system operation, i.e., the system may start the motor without warning or may not stop on command. **Improperly designed or improperly installed system interlocks and permissives may render a motor unable to start or stop on command**
- Control through serial communications can fail or can also override local controls, which can create an unsafe condition. System safety features should be employed and designed into the integrated system in a manner such that system operation, even in the event of system failure, will not cause harm or result in personnel injury or system damage. Use of the built-in system protective features and interlocks of the equipment being controlled is highly recommended (i.e., emergency-off, overload protection, etc.)
- **Never use the PLC units to perform emergency stops**. Separate switches outside the OIS, the PLC, and the ASD should be used for emergency stops.
- Changes or modifications to the PLC program should not be made without the approval of the system designer or systems integrator. Minor changes or modifications could cause the defeat of safety interlocks and permissives. Any changes or modifications should be noted and included with the system documentation.

# Instruction Overview

## 1.1 Instruction Specifications

In this section, each instruction mentioned in section 1.1 is described in detailed. For each instruction, the following items are explained:

Expression: Shows the operands required for the instruction as marked.

Function: Explains the function of the instruction with referring the operands shown on the expression box.

Execution Condition:
Shows the execution condition of the instruction and the instruction's output status.

Operand: Shows available register, device or constant value for each operand. For constant operand, available value range is described. If the constant column is just marked (√), it means normal value range (-32768 to 32767 in 16-bit integer or -2147483648 to 2147483647 in 32-bit integer) is available. Whether index modification for a register operand is usable or not is also shown for each operand.

Example: Explains the operation of the instruction by using a typical example.

Note: Explains supplementary information, limitations, etc. for the instruction.

For a quick reference, table given in next section will describe you the purpose of each instruction, instruction timings and number of steps for each instruction.

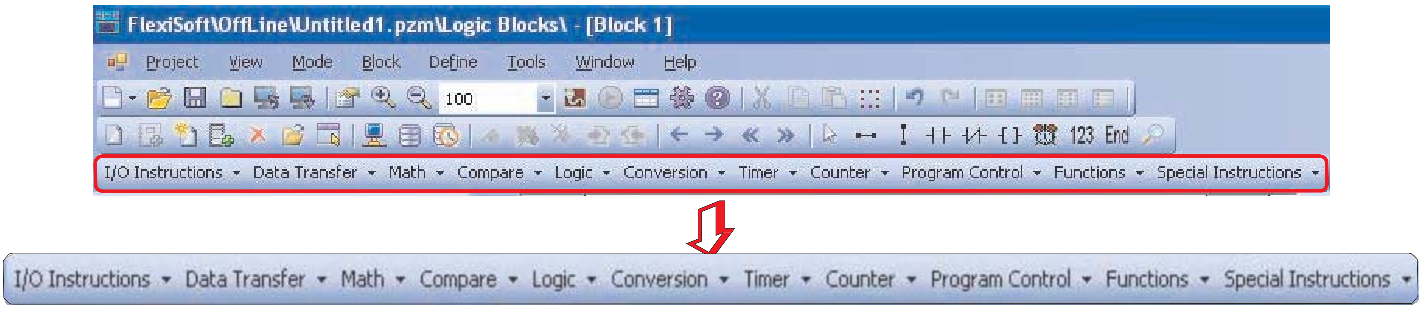
About RAM registers, EEPROM registers and Instruction Timings:
Register 'D', 'BW', 'MW', 'SW', 'T', 'C' are allocated memory in RAM for all models. 'R' are the retentive registers which retain their values after power cycle. 'R' registers are allocated memory in EEPROM for V200, OIS45/55/65/7 PLUS series models. For OIS10 PLUS and OIS40 PLUS series models a battery back up for RAM is used as 'R' memory.

When retentive registers are used in the ladder, a call to EEPROM is invoked. As the EEPROM access is slow, the execution time is higher if retentive registers are used in the instructions. So separate execution timings are mentioned for instructions where 'R' registers are used. Retentive register 'R' in OIS10 PLUS & OIS40 PLUS are stored in Battery backup RAM. So execution time for retentive register operation is same as RAM registers ('D', 'BW' etc.) User should be careful while using 'R' registers in destination as the number of write operations to EEPROM is limited to 10,000,000 operations only. After that the EEPROM may become unusable.

Data retention validity for EEPROM is more than 200 years. Data retention validity for battery backup RAM is dependent on Battery life which is published by the battery manufacturer.

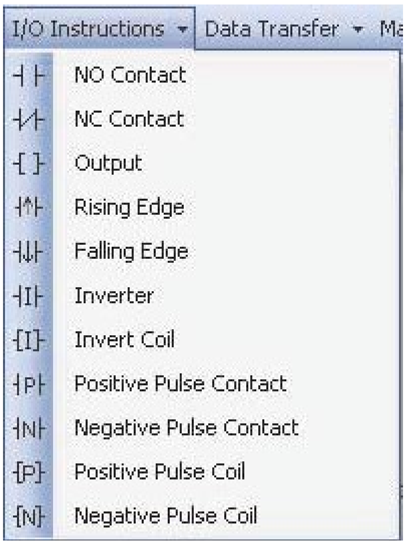
## 1.2 List of Instructions

The Flexi Panel series units has 113 types of ladder instructions as listed below.



The specifications of each instruction will be described in detail later.

### 1.2.1 I/O Instructions

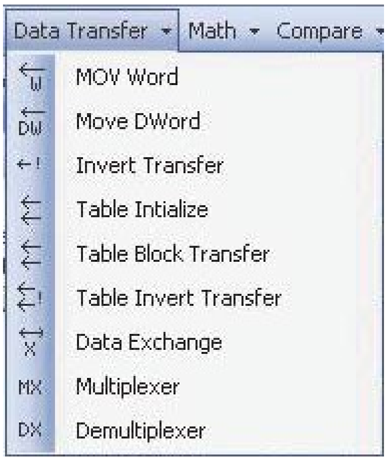


For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM andOIS10/ 20/40PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | NO Contact | NO (Normally open) contact | 1.0333 | NA | 371.988 | NA |
| 2. | NC Contact | NC (Normally Closed) contact | 1.0472 | NA | 376.992 | NA |
| 3. | Output | Relay Coil | 1.0889 | NA | 392.004 | NA |
| 4. | Transitional Contact (rising edge) | Turns ON output for 1 scan when input changes from OFF to ON | 1.0055 | NA | 361.98 | NA |
| 5. | Transitional Contact (falling edge) | Turns ON output for 1 scan when input changes from ON to OFF | 1.0194 | NA | 366.984 | NA |

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 | |
| | | | RAM and OIS10/ 20/40PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 7. | Inverter | Inverts the input state | 0.8250 | NA | 297 | NA |
| 8. | Inverter Coil | Stores the invers state of input input into device A | 1.1167 | NA | 402.012 | NA |
| 9. | Positive Pulse Contact | Turns ON output for 1 scan when input is ON and device A changes from OFF to ON. | 1.2833 | NA | 461.988 | NA |
| 10. | Negative Pulse Contact | Turns ON output for 1 scan when input is ON and device A changes from ON to OFF | 1.3389 | NA | 482.004 | NA |
| 11. | Positive Pulse Coil | Turns ON device A for 1 scan when input changes from OFF to ON | 1.3250 | NA | 477 | NA |
| 12. | Negative Pulse Coil | Turns ON device A for 1 scan when input changes from ON to OFF | 1.2972 | NA | 466.992 | NA |

### 1.2.2 Data Transfer

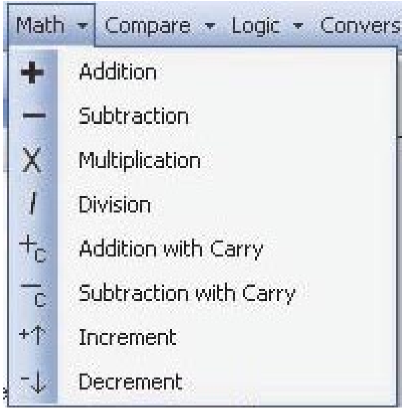


For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | MOV Word | Transfers data of A to B | 1.85278 | 1.10300 | 667.0008 | 0.6 |
| 2. | MOV Dword | Transfers double-word data of (A+1)-A to (B+1)-B | 2.22700 | 2.15709 | 801.7200 | 1.2 |
| 3. | Invert Transfer | Transfers bit-inverted data of A to B | 1.85278 | 1.12806 | 667.0008 | 0.6 |

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 4. | Table Initialize | Transfers data of A to n registers starting with B | 1.81110 | 1.10309 | 651.996 | 0.6 |
| | | | 205.25600 | 547.06731 | 73892.16 | 0.3 |
| 5. | Table Block Transfer | Transfers data n registers starting with A to n registers starting with B | 1.65833 | 1.09168 | 596.9988 | 0.6 |
| | | | 271.39440 | 1093.62762 | 97701.98401 | 600 |
| 6. | Table Invert Transfer | Transfers bit-inverted data of n registers starting with A to n registers starting with B | 1.64444 | 1.10842 | 591.9984 | 0.6 |
| | | | 316.25000 | 1095.56357 | 113850 | 600 |
| 7. | Data Exchange | Exchanges data of A with B | 2.08890 | 10.27224 | 752.004 | 6 |
| 8. | Multiplexer | Transfers data from the register specified by B in table, size n starting with A, to C | 2.68611 | 1.62344 | 966.9996001 | 0.9 |
| 9. | Demultiplexer | Transfers data from A to the register specified by B in the table, size n starting with C | 2.54722 | 1.64176 | 916.9992001 | 0.9 |

### 1.2.3 Math (Arithmatic Instructions)

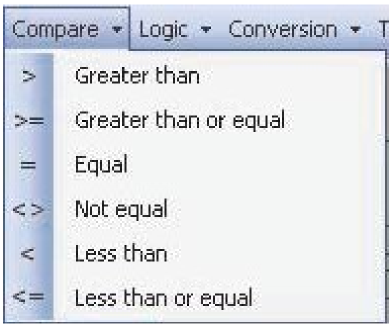


For a quick reference, below given table will describe you the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Addition<br>(i) Signed Word | Adds data of *A* & *B* and stores the result in *C* | 3.2833 | 1.6473 | 1181.988 | 0.9 |
| | (ii) Signed D-Word | | 2.9083 | 3.2323 | 1046.9988 | 1.8 |
| | (iii) Float | | | | | |
| 2. | Subtraction<br>(i) Signed Word | Subtracts data *B* from *A*, and stores result in *C* | 3.5056 | 1.6437 | 1262.0016 | 0.9 |
| | (ii) Signed D-Word | | 2.9222 | 3.2183 | 1051.992 | 1.8 |
| | (iii) Float | | | | | |
| 3. | Multiplication<br>(i) Signed | Multiplies data of *A* & *B*, and stores the result in double-length register *C+1.C* | 1.9917 | 2.1840 | 717.0012 | 1.2 |
| | (ii) Unsigned | | 2.8389 | 2.1716 | 1022.004 | 1.2 |
| | (iii) Float | | | | | |

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 4. | Division (i) Signed | Divides data of *A* by *B*, & stores the quotient in *C* and reminder in *C+1* | 9.5056 | 2.1488 | 3422.0016 | 1.2 |
| | (ii) Unsigned | | 8.8250 | 2.1524 | 3177 | 1.2 |
| | (iii) Unsigned D-Word | | 9.0300 | 2.7109 | 3250.8 | 1.4 |
| | (iv) float | | | | | |
| 5. | Addition with carry | Adds data of *A, B* & the carry, and stores result in *C*. The carry flag changes accordingly to the result. | 3.5055 | 1.6483 | 1261.98 | 0.9 |
| 6. | Subtraction with carry | Subtracts data of *B* & the carry from *A*, and stores the result in *C*. The carry flag changes accordingly to the result. | 3.4916 | 1.6475 | 1256.976 | 0.9 |
| 7. | Increment | Increaments data of *A* by 1 | 1.6444 | 5.0850 | 591.984 | 2.6 |
| 8. | Decrement | Decrements data of *A* by 1 | 1.6167 | 5.0850 | 582.0012 | 2.6 |

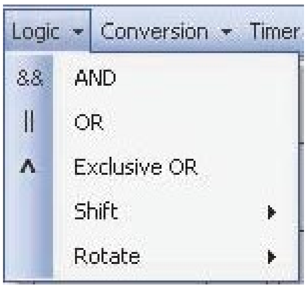
### 1.2.4 Compare Instructions



For a quick reference, below given table will describe you the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Greater than<br>(i) Signed Word | Turns ON output if $A > B$ | 2.4222 | 1.0975 | 871.9920 | 0.6 |
| | (ii) Unsigned Word | | 2.1583 | 1.1175 | 776.9880 | 0.6 |
| | (iii) Signed D-Word | | 2.6444 | 2.1814 | 951.9840 | 1.2 |
| 2. | Greater than or equal (i) Signed | Turns ON output if $A \geq B$ | 2.4222 | 1.1028 | 871.9920 | 0.6 |
| | (ii) Unsigned Word | | 2.1861 | 1.1074 | 786.9960 | 0.6 |
| | (iii) Signed D-Word | | 2.5472 | 2.1763 | 916.9992 | 1.2 |
| 3. | Equal<br>(i) Signed Word | Turns ON output if $A = B$ | 2.3111 | 1.1027 | 831.9960 | 0.6 |
| | (ii) Unsigned Word | | 2.4306 | 1.1076 | 875.0001 | 0.6 |
| | (iii) Signed D-Word | | 2.5472 | 2.1624 | 916.9992 | 1.2 |
| 4. | Not Equal<br>(i) Signed Word | Turns ON output if $A = B$ | 2.3389 | 1.1022 | 842.0004 | 0.6 |
| | (ii) Unsigned Word | | 2.1583 | 1.1062 | 776.9880 | 0.6 |
| | (iii) Signed D-Word | | 2.5889 | 2.1766 | 932.0040 | 1.2 |
| 5. | Less Than<br>(i) Signed Word | Turns ON output if $A < B$ | 2.3667 | 1.1081 | 852.0120 | 0.6 |
| | (ii) Unsigned Word | | 2.1306 | 1.1098 | 766.9980 | 0.6 |
| | (iii) Signed D-Word | | 2.5472 | 2.1757 | 916.9920 | 1.2 |
| 6. | Less than or equal<br>(i) Signed Word | Turns ON output if $A \leq B$ | 2.3520 | 1.1027 | 846.7200 | 0.6 |
| | (ii) Unsigned Word | | 2.1306 | 1.1065 | 767.0001 | 0.6 |
| | (iii) Signed D-Word | | 2.6444 | 2.1717 | 951.9840 | 1.2 |

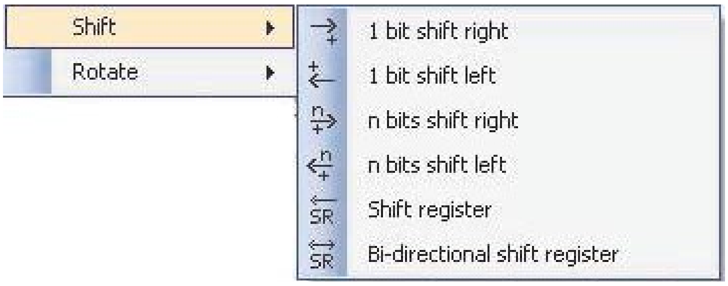
### 1.2.5 Logic Instructions



For a quick reference, below given table will describe you the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | AND | Finds logical AND of *A* & *B*, and stores it in *C*. | 2.7000 | 1.6382 | 972.0000 | 0.9 |
| 2. | OR | Finds logical OR of *A* & *B*, and stores it in *C*. | 2.6722 | 1.6373 | 961.9920 | 0.9 |
| 3. | Exclusive OR | Finds logical exclusive OR of *A* & *B*, and stores it in *C*. | 2.7417 | 1.6485 | 987.0001 | 0.9 |

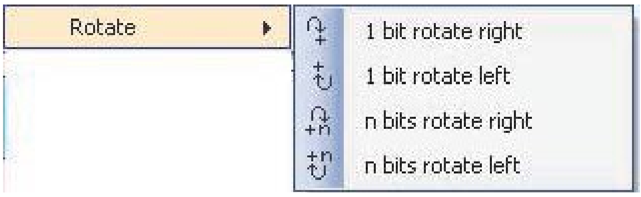
### 1.2.5.1 Shift Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | 1 bit shift right | Shifts data of *A* 1 bit to the right (LSB). The carry flag changes accordingly to the result. | 1.9778 | 1.1017 | 712.008 | 0.6 |
| 2. | 1 bit shift left | Shifts data of *A* 1 bit to the left (MSB). The carry flag changes accordingly to the result. | 2.0333 | 1.1026 | 731.988 | 0.6 |
| 3. | n bits shift right | Shifts data of *A* n bits to the right (LSB) and stores result in *B*. The carry flag changes accordingly to the result. | 2.4361 | 1.1082 | 876.9996 | 0.6 |
| 4. | n bits shift left | Shifts data of *A* n bits to the left (MSB) and stores result in *B*. The carry flag changes accordingly to the result. | 2.4639 | 1.0989 | 887.0004 | 0.6 |
| 5. | Shift register | When shift input (S) comes ON, shifts the data of specified shift register 1 bit to the left, and stores data input (D) state into *A*. This operation is enabled while enable input (E) is ON. The carry flag changes according to the result.<br>Shift register: *n* devices starting with device *A*. | 15.4500 | NA | 5562 | NA |
| | | | 36.6444 | NA | 13191.984 | NA |
| 6. | Bi-directional shift register | When shift input (S) comes ON, shifts the data of specified shift register 1 bit to the left or to the right depending on direction input (L). This operation is enabled while enable input (E) is ON. The carry flag changes according to the result.<br>Shift register: *n* devices starting with device *A*.<br>Direction: Left when L is ON, right when L is OFF | 21.6861 | NA | 7806.9996 | NA |
| | | | 42.2972 | NA | 15226.992 | NA |

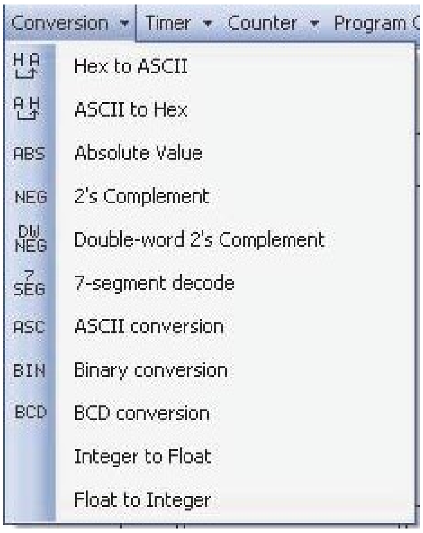
### 1.2.5.2 Rotate Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | 1 bit rotate right | Rotates data of *A* 1 bit to the right (LSB direction).The carry flag changes according to the result. | 2.0750 | 5.1967 | 747 | 2.6 |
| 2. | 1 bit rotate left | Rotates data of *A* 1 bit to the left (MSB direction). The carry flag changes according to the result. | 2.0611 | 5.1758 | 741.996 | 2.6 |
| 3. | n bit rotate right | Rotates data of *A* *n* bits to the right (LSB direction) and stores the result in *B*. The carry flag changes according to the result | 2.4222 | 1.1168 | 871.9920 | 0.6 |
| 4. | n bits rotate left | Rotates data of *A* *n* bits to the left (MSB direction) and stores the result in *B*. The carry flag changes according to the result | 2.5750 | 1.1065 | 927.0000 | 0.6 |

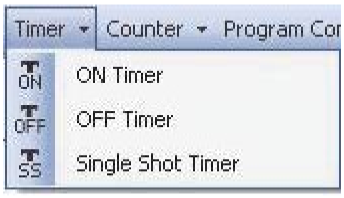
### 1.2.6 Conversion Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Hex to ASCII | Converts the hexadecimal data of *n* words stating with *A* into ASCII characters, and sto- res them in nx2 registers starting with *B* | 5.8389<br>87.1167 | NA<br>NA | 2102.0004<br>31362.012 | NA<br>NA |
| 2. | ASCII to Hex | Converts the ASCII chara- cters stored in *n* registers stating with *A* into hexa- decimal data, & stores them in *n*/2 registers starting with *B*. | 6.5333<br>64.8667 | NA<br>NA | 2351.988<br>23352.012 | NA<br>NA |
| 3. | Absolute Value | Stores absolute value of A in B. | 1.3389 | NA | 482.0004 | NA |
| 4. | 2’s Complement | Stores the 2’s complement value of A in B | 1.1306 | NA | 407.00016 | NA |
| 5. | Double-word 2’s Complement | Stores the 2’s complement value of *A+1.A* in *B+1.B* | 1.5889 | NA | 572.004 | NA |
| 6. | 7-segment decode | Converts lower 4 bits of A into7 segment code and stores in B | 1.2556 | NA | 452.016 | NA |
| 7. | ASCII Conversion | Converts the alphanumeri- cs (max. 16 characters) of *A* into ASCII codes, and stores them in registers starting with *B*. | 1.6583<br>5.7694 | NA<br>NA | 596.9988<br>2076.9984 | NA<br>NA |
| 8. | Binary conversion | Converts the BCD data in *A* into binary data, and stores it in *B* | 1.7417 | NA | 627.00012 | NA |
| 9. | BCD Conversion | Converts the binary data in *A* into BCD data, & stores in *B* | 11.3667 | NA | 4092.012 | NA |
| 10. | Integer to Float | Converts the integer data from A into float format, and stores it in B. | | | | |
| 11. | Float to Integer | Converts the float data from A into interger format, and stores it in B. | | | | |

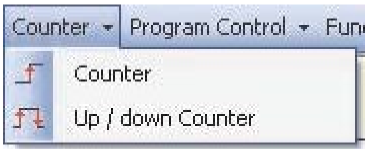
### 1.2.7 Timer Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | ON Timer | Turns ON output when the time specified by *A* has elapsed after the input came ON. *B* is a timer register | 6.7278 | NA | 2422.008 | NA |
| 2. | OFF Timer | Turns OFF output when the time specified by *A* has elapsed after the input came OFF. *B* is a timer register | 6.7833 | NA | 2441.988 | NA |
| 3. | Single Shot Timer | Turns ON output for the time specified by *A* when the input comes ON. *B* is a timer register | 7.0889 | NA | 2552.004 | NA |

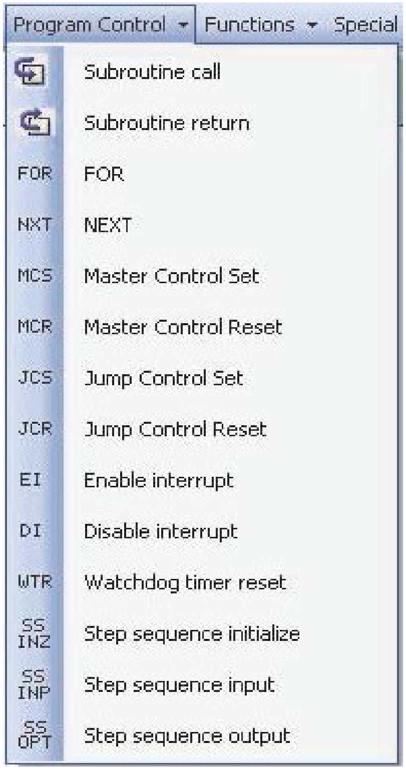
### 1.2.8 Counter Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Counter | Counts the number of cycles the count input (C) comes ON while the enable input (E) is ON, and turns ON output (Q) when the count reaches to the value specified by *A. B* is a counter register | 4.3944 | NA | 1581.984 | |
| 2. | Up / down Counter | While enable input (E) is ON, counts up or down the number of cycles the count input (C) comes ON, depending on the up/down select input (U).<br>Up when U is ON, down when U is OFF | 1.3528 | NA | 486.9972 | |

### 1.2.9 Program Control Instructions:



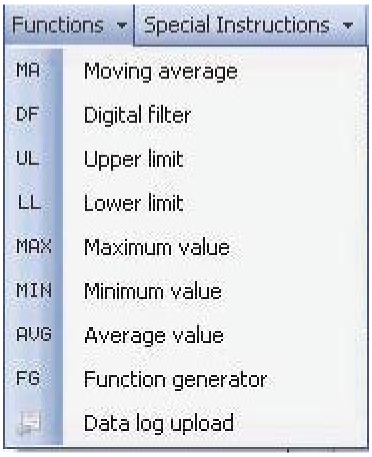
For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Subroutine call | Calls the subroutine number *n* | 2.7000 | NA | 2.7000 | NA |
| 2. | Subroutine return | Indicates the end of a subroutine | | | | |
| 3. | FOR | When the input of FOR is ON, executes the segment from FOR to NEXT the number of times specified by *n*. | 3.2694 | NA | 3.2694 | NA |
| 4. | NEXT | | | | | |
| 5. | Master Control Set | Turns OFF power rail between MCS and MCR when MCS input is OFF | 2.3111 | NA | 2.3111 | NA |
| 6. | Master Control Reset | | | | | |
| 7. | Jump Control Set | Jumps from JCS to JCR when JCS input is ON | 1.8111 | NA | 1.8111 | NA |
| 8. | Jump Control Reset | | | | | |
| 9. | Enable interrupt | Enables execution of interrupt program. | 5.1861 | NA | 5.1861 | NA |
| 10. | Disable interrupt | Disables execution of interrupt program. | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11. | Watchdog timer reset | Extends the scan time over detection time. | 0.9917 | NA | 0.9917 | NA |
| 12. | *Step sequence Initialize | Resets OFF the *n* devices stating with *A*, and sets | 3.4500<br>86.8389<br>ON *A*. | NA<br>NA | 3.4500<br>86.8389 | NA<br>NA |
| 13. | *Step sequence input | Turns ON output if input is ON and *A* is ON. | 1.2139 | NA | 1.2139 | NA |
| 14. | *Step sequence output | When input is ON, resets OFF the devices of STIN on the same rung, and sets ON *A* | 1.852778 | NA | 1.852778 | NA |

**: These Configure  a series of step sequences.*

### 1.2.10 Functions:



For a quick reference, below given table will describe the purpose of each instruction.

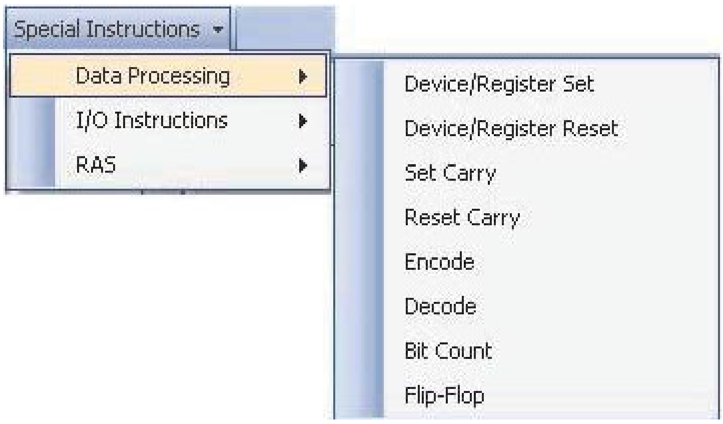
| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Moving average | Calculates the average value of latest *n* scan values of *A*, and stores the result in *C* | 5.6583<br>45.5333 | NA<br>NA | 5.6583<br>45.5333 | NA<br>NA |
| 2. | Digital Filter | Filters the value of *A* by filter constant specified by *B*, and stores the result in *C* | 28.3528 | NA | 28.3528 | NA |
| 3. | PID (1,4) | Performs PID control.(pre-derivative real PID algorithm)<br>Process value (PV): *A*<br>Set value (SV): *A+1*<br>PID parameters: *B*<br>& after Manipulation value (MV): *C* | 35.8805<br>44.7000 | NA<br>NA | 35.8805<br>44.7000 | NA<br>NA |
| 4. | Upper limit | Upper limits the value of *A* by B, and stores the result in *C*. | 2.3389 | NA | 2.3389 | NA |
| 5. | Lower limit | lower limits the value of *A* by B, and stores the result in C. | 2.0889 | NA | 2.0889 | NA |
| 6. | Maximum Value | Finds the maximum value of *n* registers data starting with *A*, and stores the value in *C* and the pointer in *C+1* | 3.9917<br>64.5611 | NA<br>NA | 3.9917<br>64.5611 | NA<br>NA |
| 7. | Minimum Value | Finds the minimum value of *n* registers data starting with *A*, and stores the value in *C* and the pointer in *C+1* | 3.9361<br>61.0611 | NA<br>NA | 3.9361<br>61.0611 | NA<br>NA |

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 8. | Average Value | Calculates the average value of *n* registers data starting with *A*, and stores the result in *C* | 12.5472 | NA | 12.5472 | NA |
| | | | 39.7556 | NA | 39.7556 | NA |
| 9. | Function generator | Finds f(x) for given x=*A*, & stores it in *C*. The function f(x) is defined by parameters stored in a table 2×*n* registers starting with *B* | 5.2417 | NA | 5.2417 | NA |
| | | | 68.7694 | NA | 68.7694 | NA |

### 1.2.11 Special Instructions

Special Instructions consist of Data Processing Instructions, I/O Instructions, and RAS Instructions

#### 1.2.11.1 Data Processing Instructions:

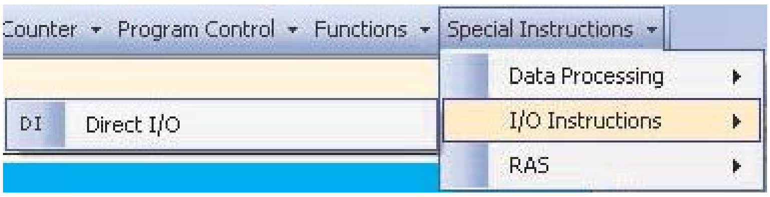


For a quick reference, below given table will describe the purpose of each instruction.

| Sr. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| No. | | | OIS10/20/40 PLUS | V200 | OIS45/55/60/70 PLUS | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Device Set<br>Register Set | If *A* is a device: Sets device *A* to ON If *A* is a register: Stores HFFFF in register *A* | 1.0889<br>1.0472 | NA<br>NA | 1.0889<br>1.0472 | NA<br>NA |
| 2. | Device Reset<br>Register Reset | If *A* is a device: Resets device *A* to OFF If *A* is a register: Stores 0 in register *A* | 1.0750<br>0.9778 | NA<br>NA | 1.0750<br>0.9778 | NA<br>NA |
| 3. | Set Carry | Sets the carry flag to ON. | 1.0194 | NA | 1.0194 | NA |
| 4. | Reset Carry | Resets the carry flag to OFF | 1.0056 | NA | 1.0056 | NA |
| 5. | Encode | Finds the uppermost ON bit position in the bit file of size 2n bits starting with register *A*, and stores it in *B*. | 4.6861<br>99.7000 | NA<br>NA | 4.6861<br>99.7000 | NA |

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 6. | Decode | In the bit file of size 2n bits starting with register *B*, sets ON the bit position indicated by lower *n* bits of *A*, and resets OFF all other bits | 4.2833<br>46.8389 | NA<br>NA | 4.2833<br>46.8389 | NA<br>NA |
| 7. | Bit Count | Counts the number of ON bits of *A* and stores it in *B* | 4.2273 | NA | 4.2278 | NA |
| 8. | Flip-Flop | Sets ON device *A* when set input (S) is ON, and resets OFF device *A* when reset input (R) is ON.<br>(Reset takes priority) | 1.5890 | NA | 1.5890 | NA |

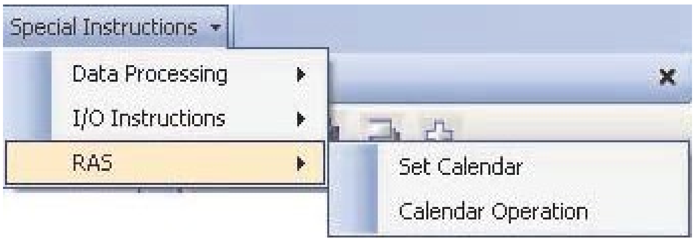
### 1.2.11.2 I/O Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | OIS10/20/40 PLUS V200 | | OIS45/55/60/70 | |
| | | | RAM and OIS10/20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | FP4035 / FP4057 (uSec) | On Retentive Register (mSec) |
| 1. | Direct I/O | i) Immediate update of inputs and outputs of base registers (Local I/O) | 1.5889 | 1.5889 | 0.57 | NA |
| | | ii) Immediate update of inputs and outputs of expansion registers (Expansion I/O) | 176.8667 | 2000 | 2000 | NA |

### 1.2.11.3 RAS Instructions:



For a quick reference, below given table will describe the purpose of each instruction.

| Sr. No. | Name of Instruction | Description | Execution Speed | | | |
|---|---|---|---|---|---|---|
| | | | FP4020/FP4030/V200 | | FP4035/FP4057 | |
| | | | RAM and OIS10/ 20/40 PLUS R Registers (uSec) | V200 Retentive Registers (mSec) | On RAM (nS) | On Retentive Register (mSec) |
| 1. | Set Calender | Sets 6 registers data starting with A into clock/calendar. | 785.2694 | NA | 785.2694 | NA |
| 2. | Calendar Operation | Calculates difference between present date & time and past date & time stored in 6 registers starting with *A*, and stores the result in 6 registers starting with *B*. | 748.9222 | NA | 748.9222 | NA |

# Instruction Details

## Instruction-1: NO Contact

Expression:

A<br>
Input ┤ ├ Output

Function:

NO (normally open) contact of device *A*.
When the input is ON and the device *A* is ON, the output is turned ON.

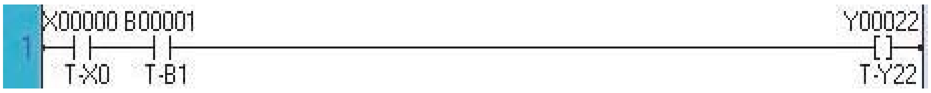
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Regardless of the state of device *A* | OFF |
| ON | When device *A* is OFF | OFF |
|  | When device *A* is ON | ON |

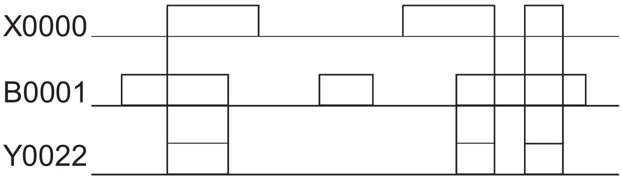
Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| *A* | Device | √ | √ | √ | √ | √ | √ | √ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Example:



Coil Y0022 comes ON when the devices X0000 and B0001 are both ON.



## Instruction-2: NC Contact

Expression:

A
Input —|/|— Output

Function:

NC (normally closed) contact of device *A.*
When the input is ON and the device *A* is OFF, the output is turned ON.

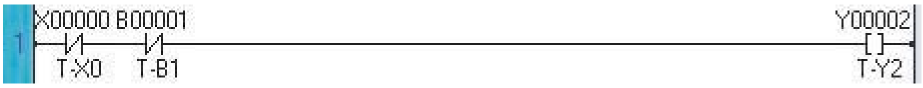
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Regardless of the state of device *A* | OFF |
| ON | When device *A* is OFF | ON |
| | When device *A* is ON | OFF |

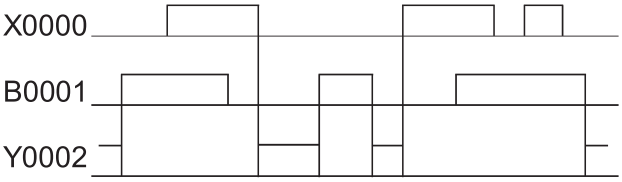
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Device | √ | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | | |

Example:



Coil Y0002 comes ON when the devices X0000 and B0001 are both OFF.



## Instruction-3: Output

Expression:

```
                A
Input      ─(   )─|
```

Function:

Output coil of device *A*.
When the input is ON, the device *A* is ON.

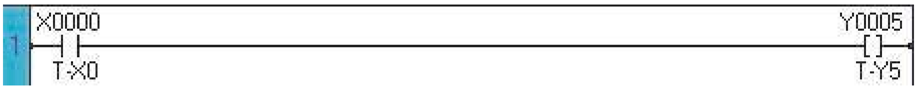
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Sets device *A* to OFF | — |
| ON | Sets device *A* to ON | — |

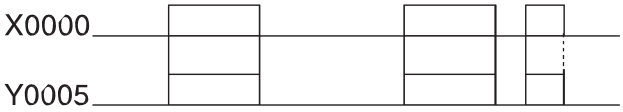
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



Coil Y0005 comes ON when the device X0000 is ON.



## Instruction-4: Rising Edge (Transitional Contact)

Expression:

A
Input ┤↑├ Output

Function:

When the input at last scan is OFF and the input at this scan is ON, the output is turned ON.
This instruction is used to detect the input changing from OFF to ON

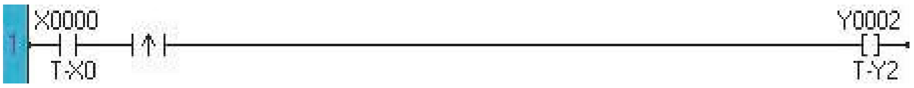
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Regardless of the input state at last scan | OFF |
| ON | When the input state at last scan is OFF | ON |
| | When the input state at last scan is ON | OFF |

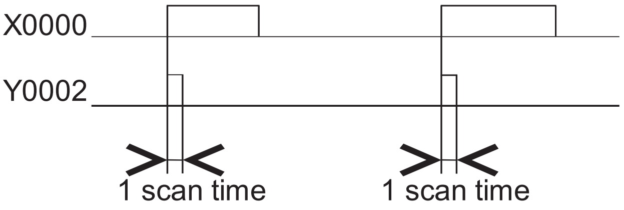
Operand:

No operand is required.

Example:



Coil Y0002 comes ON for only 1 scan when the device X0000 comes ON.



## Instruction-5: Falling Edge (Transitional Contact)

Expression:

A

Input ┤↓├ Output

Function:

When the input at last scan is ON and the input at this scan is OFF, the output is turned ON.
This instruction is used to detect the input changing from ON to OFF.

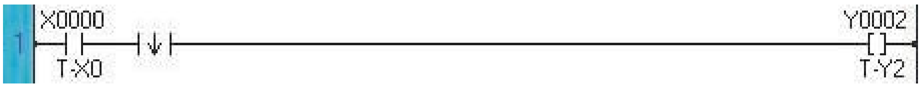
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | When the input state at last scan is OFF | OFF |
| | When the input state at last scan is ON | ON |
| ON | Regardless of the input state at last scan | OFF |

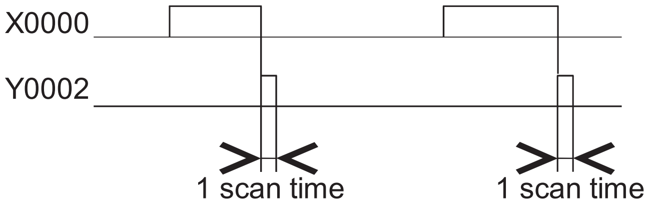
Operand:

No operand is required.

Example:



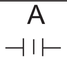
Coil Y0002 comes ON for only 1 scan when the device X0000 comes OFF.



## Instruction-6: Inverter

Expression:

Input ⊣|⊢ Output



Function:

When the input is OFF, the output is turned ON, and when the input is ON, the output is turned OFF. This instruction inverts the link state.

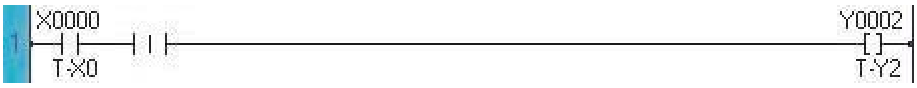
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Inverts the input state | ON |
| ON | Inverts the input state | OFF |

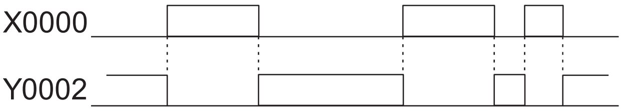
Operand:

No operand is required

Example:



Device Y0002 comes ON when X0000 is OFF, and Y0002 comes OFF when X0000 is ON.



## Instruction-7: Inverter Coil

Expression:

| | A |
|---|---|
| Input | —( I )— |

Function:

When the input is OFF, the device *A* is set to ON, and when the input is ON, the device *A* is set to OFF. This instruction inverts the input state and store it in the device *A.*

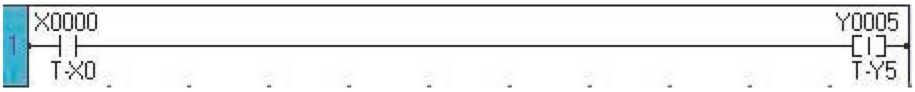
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Sets device A to ON | — |
| ON | Sets device A to OFF | — |

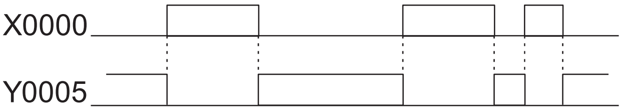
Operand:

| | Name | Device | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



Device Y0005 comes ON when X0000 is OFF, and Y0005 comes OFF when X0000 is ON.



## Instruction-8: Positive Pulse Contact

Expression:

```
            A
Input     ⊣P⊢     Output
```

Function:

When the input is ON and the device *A* is changed from OFF to ON (OFF at last scan and ON at this scan), the output is turned ON.
This instruction is used to detect the device changing from OFF to ON.

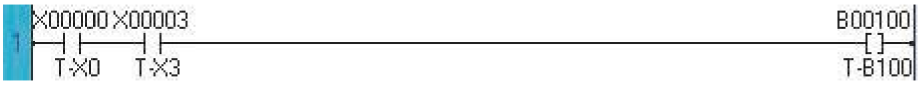
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | Regardless of the state of device A | | OFF |
| ON | State of device A is OFF | | OFF |
| | State of device A is ON | A is OFF at last scan | ON |
| | | A is ON at last scan | OFF |

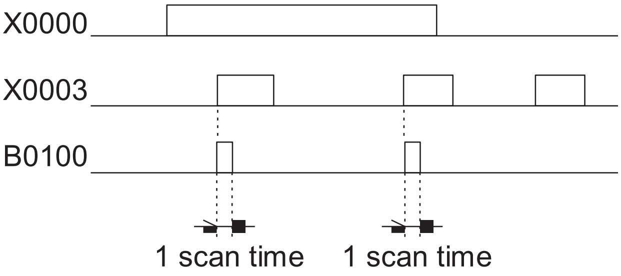
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | √ | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | |

Example:



B0100 comes ON for only 1 scan when X0000 is ON and X0003 changes to ON.



## Instruction-9: Negative Pulse Contact

Expression:

A
Input ⊣N⊢ Output

Function:

When the input is ON and the device *A* is changed from ON to OFF (ON at last scan and OFF at this scan), the output is turned ON.
This instruction is used to detect the device changing from ON to OFF.

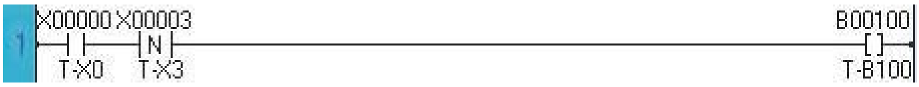
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | Regardless of the state of device A | | OFF |
| ON | State of device A is OFF | A is OFF at last scan | OFF |
| | | A is ON at last scan | ON |
| | State of device A is ON | | OFF |

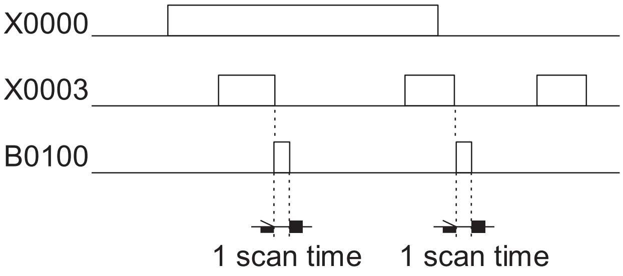
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | √ | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | |

Example:



B0100 comes ON for only 1 scan when X0000 is ON and X0003 changes to OFF.



## Instruction-10: Positive Pulse Coil

Expression:

```
            A
Input     -( P)-|
```

Function:

When the input is changed form OFF to ON, the device *A* is set to ON for 1 scan time.
This instruction is used to detect the input changing from OFF to ON.

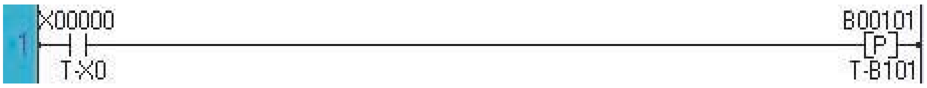
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Sets device A to OFF | — |
| ON | When the input at last scan is OFF, sets A to ON | — |
| | When the input at last scan is OFF, sets A to OFF | — |

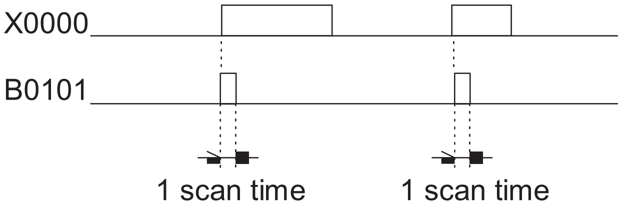
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



B0101 comes ON for only 1 scan when X0000 is changed from OFF to ON.



## Instruction-11: Negative Pulse Coil

Expression:

```
                A
Input         ─( N)─
```

Function:

When the input is changed from ON to OFF, the device *A* is set to ON for 1 scan time.
This instruction is used to detect the input changing from ON to OFF.

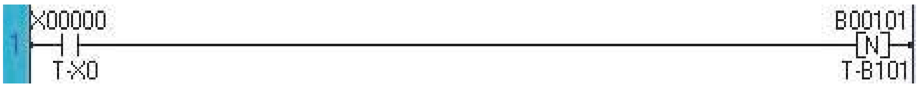
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | When the input at last scan is OFF, sets *A* to OFF | — |
| | When the input at last scan is ON, sets *A* to ON | — |
| ON | Sets device *A* to OFF | — |

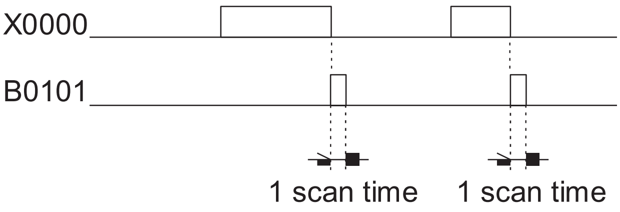
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



B0101 comes ON for only 1 scan when X0000 is changed from ON to OFF.



## Instruction-12: MOV WORD

Expression:

Input ⊣ A MOV B ⊢

Function:

When the input is ON, the data of A is stored in B.

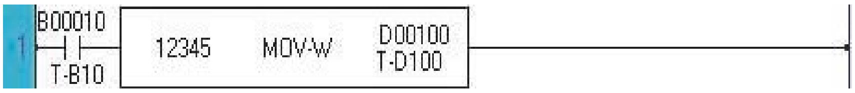
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

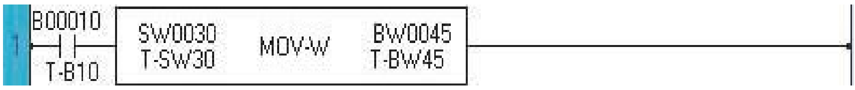
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | |

Example-1: (constant to register)



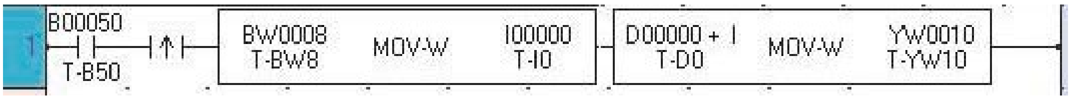
B0010 is ON, a constant data (12345) is stored in D0100 and the output is turned ON.

Example-2: (register to register)



When B00010 is ON, the data of SW030 is stored in BW045 and the output is turned ON. If SW030 is 500, the data 500 is stored in BW045.

Example-3: (index modification)



When B050 is changed from OFF to ON, the data of BW008 is stored in the index register I and the data of D(0000+I) is stored in YW010. If BW008 is 300, the data of D0300 is stored in YW010.

## Instruction-13: MOV DWord

Expression:

Input ⊣[ A+1.A MOV B+1.B ]–

Function:

When the input is ON, the double-word (32-bit) data of A+1× A is stored in double-word register B+1× B. The data range is -2147483648 to 2147483647.

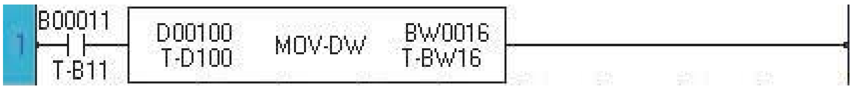
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B011 is ON, a double-word data of D0101×D0100 is stored in BW17×BW16 and the output is turned ON. If D0101×D0100 is 1234567, the data 1234567 is stored in BW17×BW16.

## Instruction-14: Invert Transfer

Expression:

Input ⊣[ A NOT B ]⊢

Function:

When the input is ON, the bit-inverted data of A is stored in B.

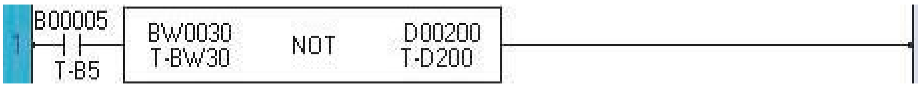
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B005 is ON, the bit-inverted data of BW30 is stored in D0200 and the output is turned ON.
If BW30 is H4321, the bit-inverted data (HBCDE) is stored in D0200.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW30 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

4 3 2 1

↓ Bit-Invert

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0200 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

B C D E

## Instruction-15: Table Initialize

Expression:

Input ⊣[ A TINZ (n) B ]⊢ Output

Function:

When the input is ON, the data of A is stored in n registers starting with B.
The allowable range of the table size n is 1 to 1024 words.

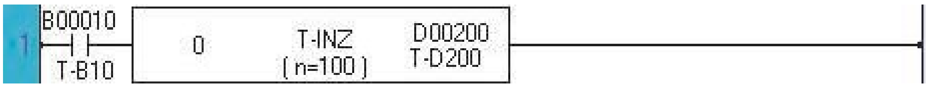
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

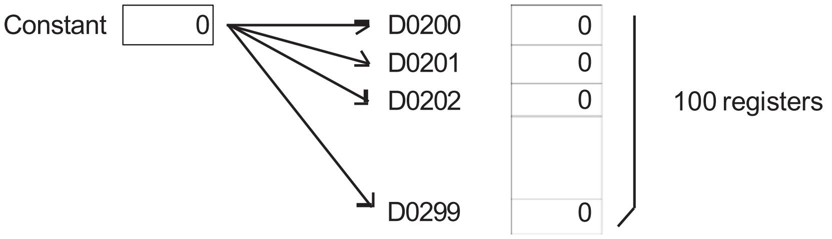
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | ource | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | |
| n | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 1024 | |
| B | Start of estination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B010 is ON, a constant data (0) is stored in 100 registers starting with D0200 (D0200 to D0299) and the output is turned ON.



## Instruction-16: Table Block Transfer

Expression:

Input ─[ A TMOV (n) B ]─ Output

Function:

When the input is ON, the data of n registers starting with A are transferred to n registers starting with B in a block. The allowable range of the table size n is 1 to 1024 words.

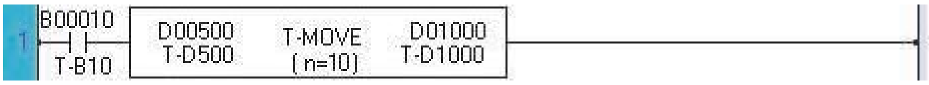
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 1024 | |
| *B* | Start of Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B010 is ON, the data of D0500 to D0509 (10 registers) are block transferred to D1000 to D1009, and the output is turned ON.

| | | | | | |
|---|---|---|---|---|---|
| D0500 | 1111 | | D1000 | 1111 | 10 registers |
| D0501 | 2222 | Block transfer → | D1001 | 2222 | |
| D0502 | 3333 | | D1002 | 3333 | |
| | | | | | |
| D0509 | 12345 | | D1009 | 12345 | |

Note:
The source and destination tables can be overlapped.

# Instruction-17: Table Invert Transfer

Expression:

Input -[ A TNOT (n) B ]- Output

Function:

When the input is ON, the data of n registers starting with A are bit-inverted and transferred to n registers starting with B in a block. The allowable range of the table size n is 1 to 1024 words.

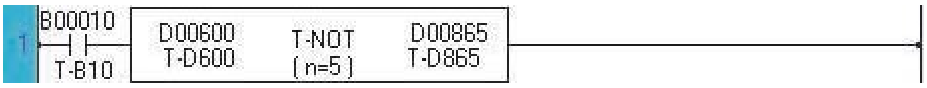
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 1024 | |
| *B* | Start of Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B010 is ON, the data of D0600 to D0604 (5 registers) are bit-inverted and transferred to D0865 to D0869, and the output is turned ON.

| | | | | | |
|---|---|---|---|---|---|
| D0600 | H00FF | | D0865 | HFF00 | |
| D0601 | H0000 | Bit Invert & | D0866 | HFFFF | |
| D0602 | H1234 | Transfer | D0867 | HEDCB | 5 registers |
| D0603 | H5555 | → | D0868 | HAAAA | |
| D0604 | H89AB | | D0869 | H7654 | |

Note:
The source and destination tables can be overlapped.

## Instruction-18: Data Exchange

Expression:

Input ┤[ A XCHG B ]├ Output

Function:

When the input is ON, the data of A and the data of B is exchanged.

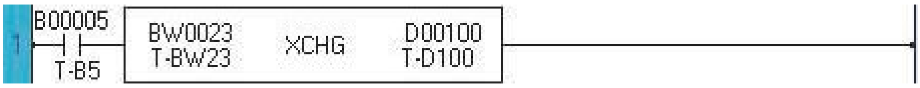
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Operation Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |
| *B* | Operation Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



1 B00005 T-B5 BW0023 T-BW23 XCHG D00100 T-D100

When B005 is ON, the data of BW23 and D0100 is exchanged. If the original data of BW23 is 23456 and that of D0100 is 291, the operation result is as follows.

| | | | | |
|---|---|---|---|---|
| BW023 | 23456 | ↗ | BW023 | 291 |
| D0100 | 291 | ↘ | D0100 | 23456 |

Before Operation After Operation

## Instruction-19: Multiplexer

Expression:

Input ⊣[ A MPX (n) B ⟶ C ]⊢ Output

Function:

When the input is ON, the data of the register which is designated by B in the table, size n starting with A, is transferred to C.

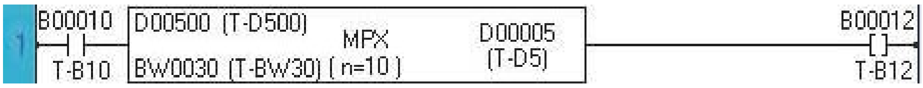
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | OFF |
| | Pointer over (no execution) | ON |

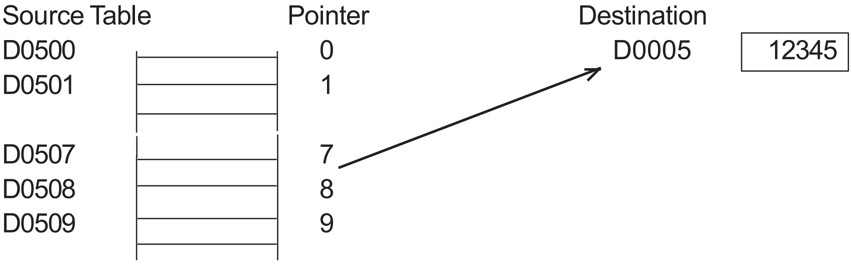
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of table | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| B | Pointer | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | 0 - 63 | |
| *C* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | |

Example:



When B010 is ON, the register data which is designated by BW30 is read from the table D0500 to D0509 (10 registers size), and stored in D0005.
If the data of BW30 is 7, D0507 data is transferred to D0005.



Note:
If the pointer data designates outside the table (10 or more in the above example), the transfer is not executed and the output comes ON.
The table must be within the effective range of the register address.

## Instruction-20: Demultiplexer

Expression:

Input ⊣ A DMPX (n) B⟶C ⊢ Output

Function:

When the input is ON, the data of A is transferred to the register which is designated by B in the table, size n starting with C.

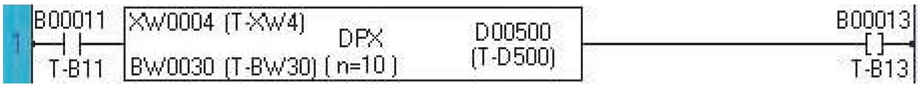
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | OFF |
| | Pointer over (no execution) | ON |

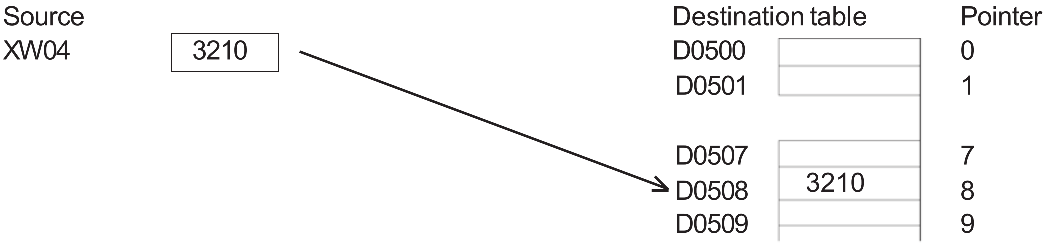
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| B | Pointer | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | 0 - 63 | |
| *C* | Start of table | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B011 is ON, the data of XW04 is transferred to the register which is designated by BW30 in the table D0500 to D0509 (10 registers size).
If the data of BW30 is 8, XW04 data is transferred to D0508.



Note:
If the pointer data designates outside the table (10 or more in the above example), the transfer is not executed and the output comes ON.
The table must be within the effective range of the register address.

## Instruction-21: Addition

Expression:

Input ⊣[ A + B⟶C ] Output

Function:

When the input is ON, the data of A and the data of B are added, and the result is stored in C.
If the result is greater than 32767, the upper limit value 32767 is stored in C, and the output is turned ON. If the result is smaller than -32768, the lower limit value -32768 is stored in C, and the output is turned ON.

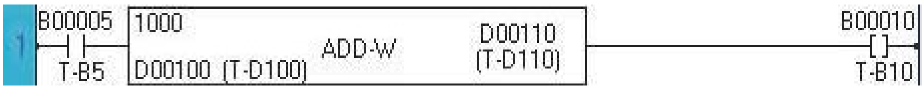
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
| | Execution (overflow or underflow occurred) | ON |

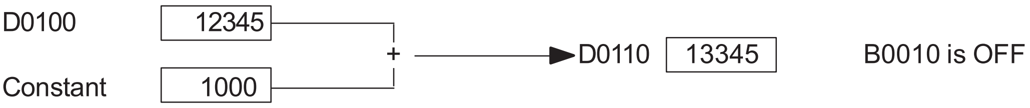
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Augend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Addent | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Sum | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

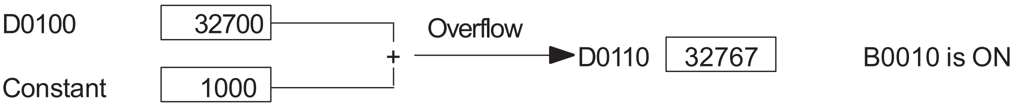
Example:



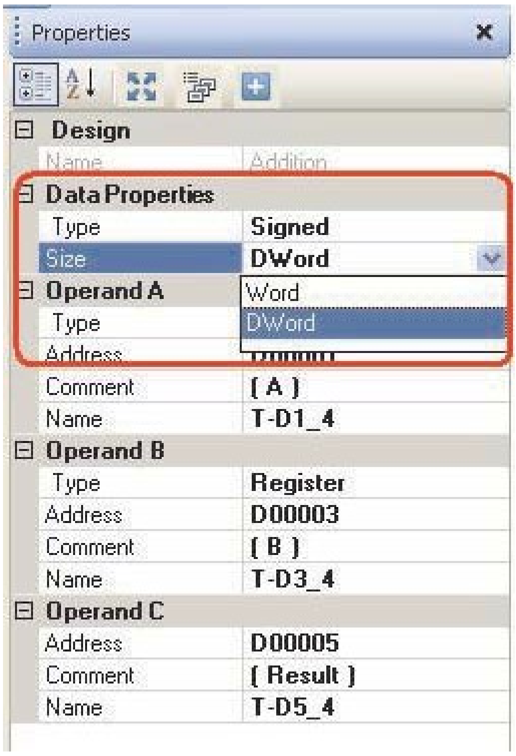
When B005 is ON, the data of D0100 and the constant data 1000 is added, and the result is stored in D0110.
If the data of D0100 is 12345, the result 13345 is stored in D0110, and B010 is turned OFF.



If the data of D0100 is 32700, the result exceeds the limit value, therefore 32767 is stored in D0110, and B010 is turned ON.



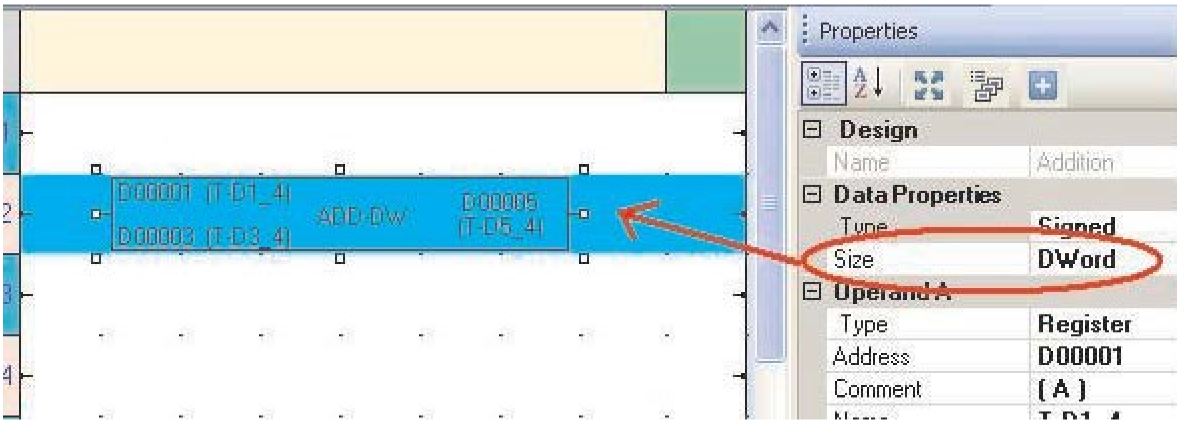
When user select “Addition” function and place it in logic block, “Property” docker window occurs to the right side of the application window.
Where user can select “Addition” to “Double-word” addition from the Data Proprty selection tab as shown below:



Properties

| | |
|---|---|
| **Design** | |
| Name | Addition |
| **Data Properties** | |
| Type | Signed |
| Size | DWord |
| **Operand A** | Word<br>DWord |
| Type | |
| Address | D00001 |
| Comment | ( A ) |
| Name | T-D1_4 |
| **Operand B** | |
| Type | Register |
| Address | D00003 |
| Comment | ( B ) |
| Name | T-D3_4 |
| **Operand C** | |
| Address | D00005 |
| Comment | ( Result ) |
| Name | T-D5_4 |

Thus by selcting “Size” type,”Addition” entry can be changed to “Double-word Addition” entry as shown below:



Properties

| | |
|---|---|
| **Design** | |
| Name | Addition |
| **Data Properties** | |
| Type | Signed |
| Size | DWord |
| **Operand A** | |
| Type | Register |
| Address | D00001 |
| Comment | ( A ) |

## Instruction-22: Double-word Addition

Expression:

Input ─[ A+1.A D + B+1.B ⟶ C+1.C ]─ Output

Function:

When the input is ON, the double-word data of A+1× A and B+1× B are added, and the result is stored in C+1× C. The data range is -2147483648 to 2147483647.
If the result is greater than 2147483647, the upper limit value 2147483647 is stored in C+1× C, and the output is turned ON. If the result is smaller than -2147483648, the lower limit value -2147483648 is stored in C+1× C, and the output is turned ON.

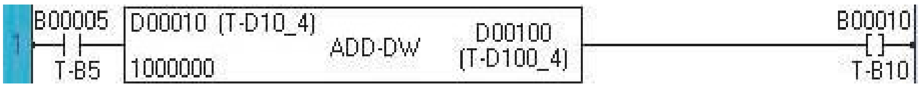
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
|  | Execution (overflow or underflow occured) | ON |

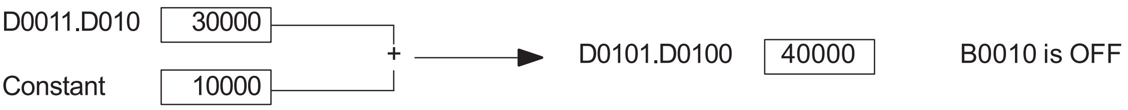
Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| A | Augend |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ |  |  |  | √ | √ | √ |  |
| B | Addent |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ |  |  |  | √ | √ | √ |  |
| C | Sum |  |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ |  |  |  | √ | √ |  |  |

Example:



When B005 is ON, the data of D0011×D0010 and the constant data 100000 is added, and the result is stored in D0101×D0100.
If the data of D0011×D0010 is 300000, the result 400000 is stored in D0101×D0100, and B010 is turned OFF. (No overflow/underflow).



## Instruction-23: Float Addition

Expression:

Input ⫞ A+1.A ADD-F B+1.B⟶ C+1.C ⊢ Output

Function:

When the input is ON, the float data of A+1× A and B+1× B are added, and the result is stored in C+1× C. The data range is -3.4e+38 to +3.4e+38.
If the result is greater than 3.4e+38, the upper limit value is stored in C+1× C, and the output is turned ON. If the result is smaller than -3.4e+38, the lower limit value is stored in C+1× C, and the output is turned ON.

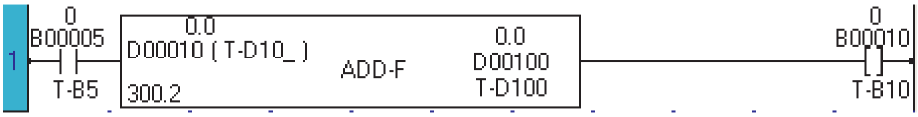
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
| | Execution (overflow or underflow occured) | ON |

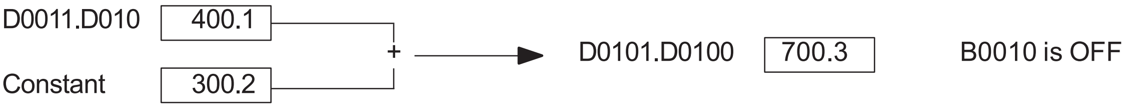
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Augend | | | | | | | | | | √ | | | | | | | | | √ | √ | |
| B | Addent | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| *C* | Sum | | | | | | | | | | √ | | | | √ | | | | | √ | | |

Example:



When B005 is ON, the float data of D0011×D0010 and the float data 300.2 is added, and the result is stored in D0101×D0100.
If the data of D0011×D0010 is 400.1, the result is stored in D0101×D0100, and B010 is turned OFF. (No overflow/underflow).



# Instruction-24: Subtraction

Expression:

Input ─[ A - B ⟶C ] Output

Function:

When the input is ON, the data of B is subtracted from the data of A, and the result is stored in C.
If the result is greater than 32767, the upper limit value 32767 is stored in C, and the output is turned ON. If the result is smaller than -32768, the lower limit value -32768 is stored in C, and the output is turned ON.

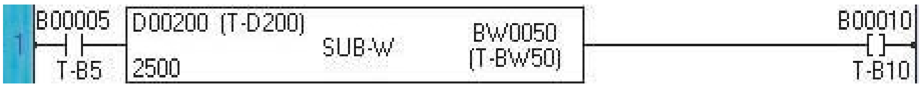
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
| | Execution (overflow or underflow occured) | ON |

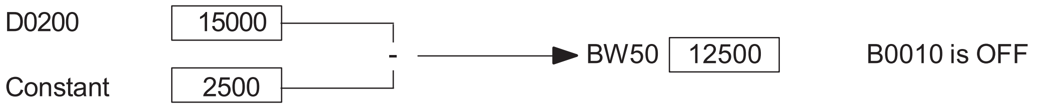
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Minuend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Subtrahend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *C* | Difference | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

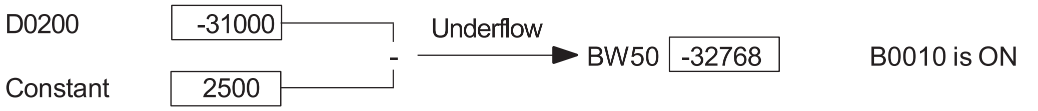
Example:



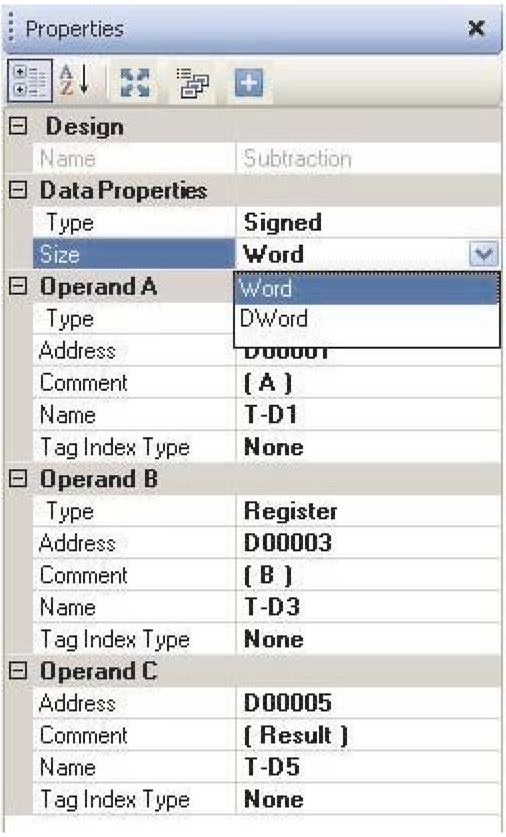
When B005 is ON, the constant data 2500 is subtracted from the data of D0200, and the result is stored in BW50.
If the data of D0200 is 15000, the result 12500 is stored in BW50, and B010 is turned OFF.



If the data of D0200 is -31000, the result is smaller than the limit value, therefore -32768 is stored in BW50, and B010 is turned ON.



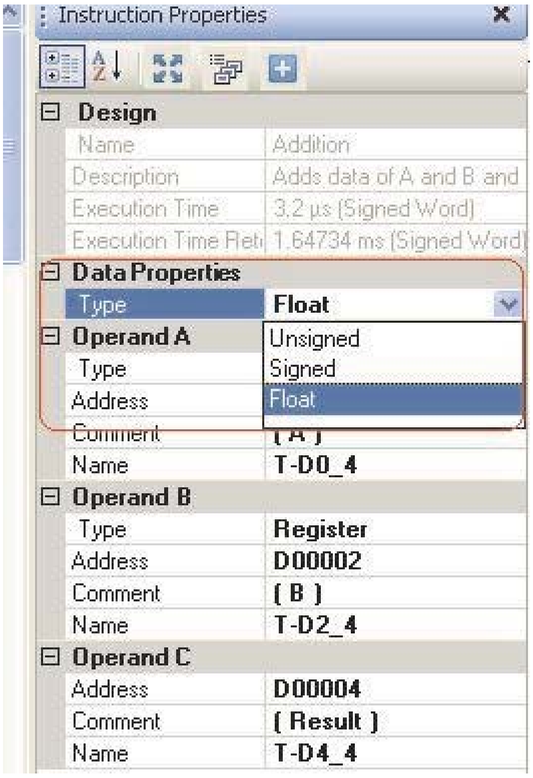
When user select “Subtraction” function and place it in logic block, “Property” docker window occurs to the right side of the application window.
Where user can select Size type to “Double-word” addition from the Data Proprty selection tab as shown below:



Properties

| | |
|---|---|
| **Design** | |
| Name | Subtraction |
| **Data Properties** | |
| Type | Signed |
| Size | Word |
| **Operand A** | Word<br>DWord |
| Type | |
| Address | D00001 |
| Comment | ( A ) |
| Name | T-D1 |
| Tag Index Type | None |
| **Operand B** | |
| Type | Register |
| Address | D00003 |
| Comment | ( B ) |
| Name | T-D3 |
| Tag Index Type | None |
| **Operand C** | |
| Address | D00005 |
| Comment | ( Result ) |
| Name | T-D5 |
| Tag Index Type | None |

Also user can change “Type” of the data entry to “Signed”, “Unsigned” or “Float” type.



Instruction Properties

| | |
|---|---|
| **Design** | |
| Name | Addition |
| Description | Adds data of A and B and |
| Execution Time | 3.2 µs (Signed Word) |
| Execution Time Ret | 1.64734 ms (Signed Word) |
| **Data Properties** | |
| Type | Float |
| **Operand A** | Unsigned<br>Signed<br>Float |
| Type | |
| Address | |
| Comment | ( A ) |
| Name | T-D0_4 |
| **Operand B** | |
| Type | Register |
| Address | D00002 |
| Comment | ( B ) |
| Name | T-D2_4 |
| **Operand C** | |
| Address | D00004 |
| Comment | ( Result ) |
| Name | T-D4_4 |

## Instruction-25: Double-Word Subtraction

Expression:

Input –[ A+1.A D- B+1.B ⟶ C+1.C ]– Output

Function:

When the input is ON, the double-word data of B+1× B is subtracted from A+1× A, and the result is stored in C+1× C. The data range is -2147483648 to 2147483647.
If the result is greater than 2147483647, the upper limit value 2147483647 is stored in C+1× C, and the output is turned ON. If the result is smaller than -2147483648, the lower limit value -2147483648 is stored in C+1× C, and the output is turned ON.

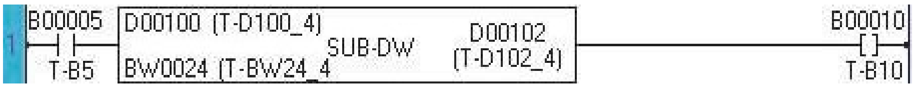
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
| | Execution (overflow or underflow occured) | ON |

Operand:

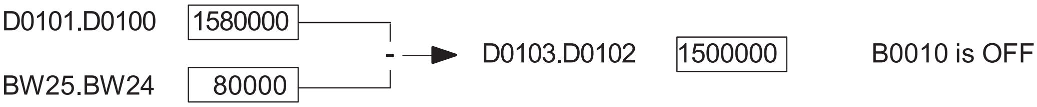
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Minuend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| B | Subtrahend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| C | Difference | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B005 is ON, the double-word data of BW25×BW24 is subtracted from the double-word data of D0101×D0100, and the result is stored in D0103×D0102.

If the data of D0101×D0100 is 1580000 and the data of BW25×BW24 is 80000, the result 1500000 is stored in D0103×D0102, and B010 is turned OFF. (No overflow/underflow)



## Instruction-26: Float Subtraction

Expression:

Input ┤ A+1.A SUB-F B+1.B ⟶ C+1.C ├ Output

Function:

When the input is ON, the double-word data of B+1× B is subtracted from A+1× A, and the result is stored in C+1× C. The data range is +/- 3.4e + 38.
If the result is greater than +3.4e+38, the upper limit value is stored in C+1× C, and the output is turned ON. If the result is smaller than -3.4e+38, the lower limit value is stored in C+1× C, and the output is turned ON.

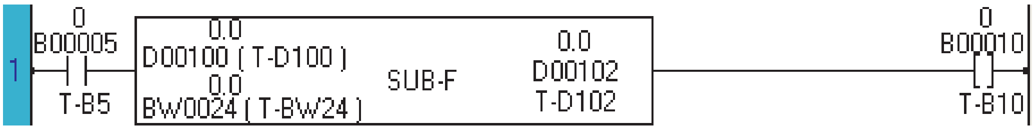
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (Normal) | OFF |
| | Execution (overflow or underflow occurred) | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Minuend | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| B | ubtrahend | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| C | Difference | | | | | | | | | | √ | | | | √ | | | | | √ | | |

Example:



When B005 is ON, the float data of BW25×BW24 is subtracted from the float data of D0101×D0100, and the result is stored in D0103×D0102.

If the data of D0101×D0100 is 700.12 and the data of BW25×BW24 is 300.02, the result 400.1 is stored in D0103×D0102, and B010 is turned OFF. (No overflow/underflow)

D0101.D0100 700.12

- ⟶ D0103.D0102 400.1 B0010 is OFF

BW25.BW24 300.02

## Instruction-27: Multiplication

Expression:

Input ┤[ A * B ⟶C+1.C ]├ Output

Function:

When the input is ON, the data of A is multiplied by the data of B, and the result is stored in double length register C+1×C.

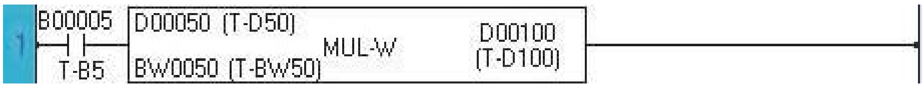
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

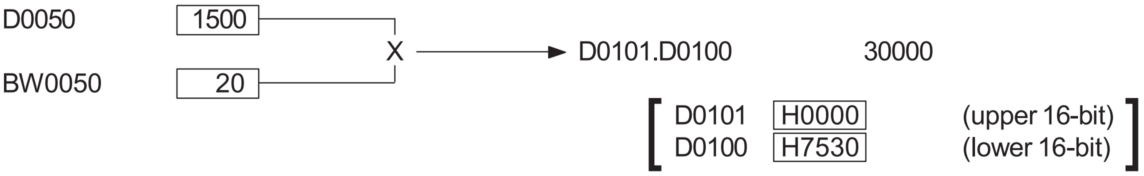
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Multiplicand | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Multiplier | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Product | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:

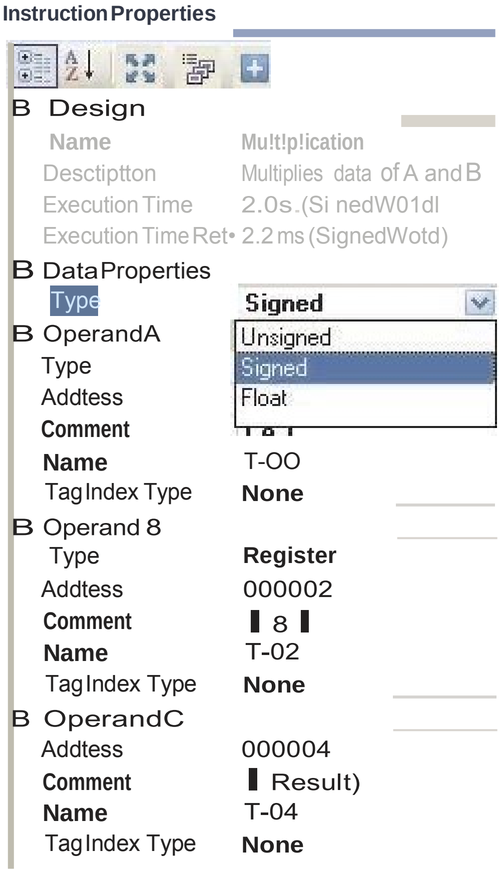


```
1  B00005   D00050 (T-D50)              D00100
   ─┤ ├─              MUL-W            (T-D100)
    T-B5    BW0050 (T-BW50)
```

When B005 is ON, the data of D0050 is multiplied by the data of BW050, and the result is stored in double-length register D0101×D0100 (upper 16-bit in D0101 and lower 16-bit in D0100).
If the data of D0050 is 1500 and the data of BW05 is 20, the result 30000 is stored in D0101×D0100.



When user select "Multiplication" function and place it in logic block,"Property" docker window occurs to the right side of the application window.
Where user can select data type to "Signed","Unsigned" or "Float" muhiplication from the Data Proprty selection tab as shown below:



Instruction Properties

B Design
Name Mu!t!p!ication
Desctiptton Multiplies data of A and B
Execution Time 2.0s.(Si nedW01dl
Execution Time Ret• 2.2 ms (SignedWotd)

B Data Properties
Type Signed

Unsigned
Signed
Float

B OperandA
Type
Addtess
Comment
Name T-OO
Tag Index Type None

B Operand 8
Type Register
Addtess 000002
Comment ▌8▌
Name T-02
Tag Index Type None

B OperandC
Addtess 000004
Comment ▌ Result)
Name T-04
Tag Index Type None

# Instruction-28: Unsigned Multiplication

Expression:

Input ⊣[ A U* B⟶C+1.C ]⊢ Output

Function:

When the input is ON, the unsigned data of A and B are multiplied, and the result is stored in double-length register C+1×C. The data range of A and B is 0 to 65535 (unsigned 16-bit data).

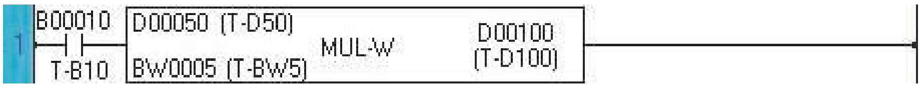
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

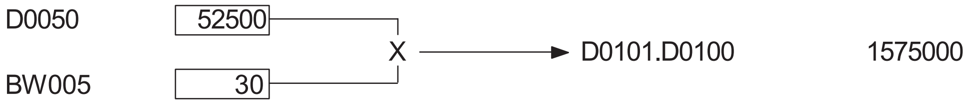
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Multiplicand | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Multiplier | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Product | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B010 is ON, the data of D0050 is multiplied by the data of BW05, and the result is stored in double-length register D0101×D0100 (upper 16-bit in D0101 and lower 16-bit in D0100).
If the data of D0050 is 52500 and the data of BW05 is 30, the result 1575000 is stored in D0101×D0100.



Note: This instruction handles the register data as unsigned integer.

## Instruction-29: Float Multiplication

Expression:

Input ─[ A+1.A MUL-F B+1.B→C+1.C ]─ Output

Function:

When the input is ON, the data of A is multiplied by the data of B, and the result is stored in double length register C+1×C.

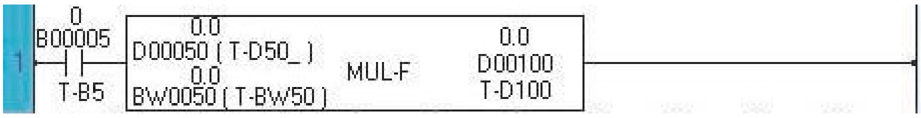
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

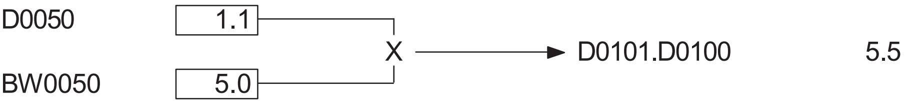
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Multiplicand | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| B | Multiplier | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| C | Product | | | | | | | | | | √ | | | | √ | | | | | √ | | |

Example:



When B005 is ON, the data of D0050 x D0051 is multiplied by the data of BW050 X BW0051, and the result is stored in double-length register D0101×D0100 (upper 16-bit in D0101 and lower 16-bit in D0100).

If the data of D0050 x D0051 is 1.1 and the data of BW05 is 5.0, the result 5.5 is stored in D0101×D0100.



## Instruction-31: Division

Expression:

Input ─[ A / B ⟶C ]─ Output

Function:

When the input is ON, the data of A is divided by the data of B, and the quotient is stored in C and the remainder in C+1.

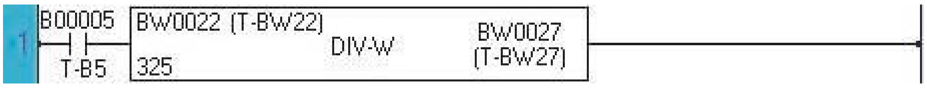
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No execution | OFF | - |
| ON | Normal execution (B ≠ 0) | ON | - |
|  | No execution (B = 0) | OFF | - |

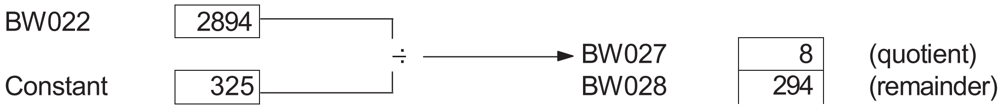
Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| A | Dividend |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Divisor |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Quotient |  |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |  | √ |

Example:



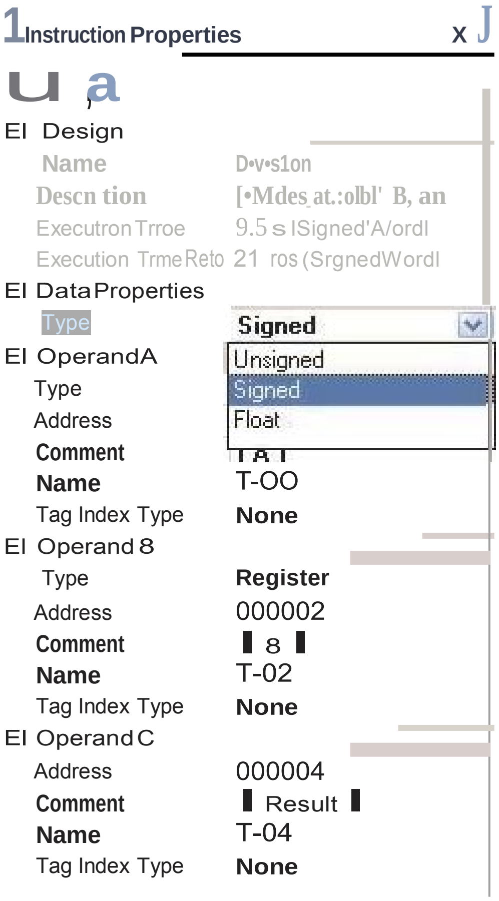
When B005 is ON, the data of BW22 is divided by the constant data 325, and the quotient is stored in BW27 and the remainder is stored in BW28.
If the data of BW22 is 2894, the quotient 8 is stored in BW27 and the remainder 294 is stored in BW28.



Note

- If divisor (operand B) is 0, ERF (instruction error flag = S0034) is set to ON.
  The ERF (S0034) can be reset to OFF by user program, e.g. Ä[ RST S0034 ]Ä.
- If the index register K is used as operand C, the remainder is ignored.
- If operand A is -32768 and operand B is -1, the data -32768 is stored in C and 0 is stored in C+1.

When user select "Division" function and place it in logic block,"Property" dockerwindow occurs to the right side of the application window.
Where user can select Data type as"Signed","Unsigned" dwision or"Fioat" division from the Data Proprty selection tab as shown below:



1Instruction Properties x J

| | |
|---|---|
| EI Design | |
| Name | D•v•s1on |
| Descn tion | [•Mdes at.:olbl' B, an |
| Executron Trroe | 9.5 s lSigned'A/ordl |
| Execution TrmeReto | 21 ros (SrgnedWordl |
| EI Data Properties | |
| Type | Signed<br>Unsigned<br>Signed<br>Float |
| EI OperandA | |
| Type | |
| Address | |
| Comment | TAT |
| Name | T-OO |
| Tag Index Type | None |
| EI Operand 8 | |
| Type | Register |
| Address | 000002 |
| Comment | ▌8▌ |
| Name | T-02 |
| Tag Index Type | None |
| EI Operand C | |
| Address | 000004 |
| Comment | ▌Result▌ |
| Name | T-04 |
| Tag Index Type | None |

# Instruction-32: Unsigned Division

Expression:

Input ⊣[ A U/ B ⟶C ]⊢ Output

Function:

When the input is ON, the unsigned data of A is divided by the unsigned data of B, and the quotient is stored in C and the remainder in C+1. The data range of A and B is 0 to 65535 (unsigned 16-bit data).

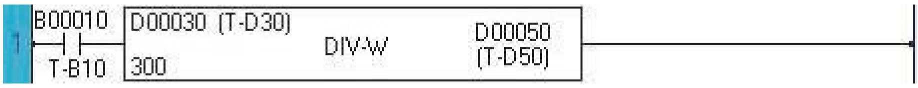
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No execution | OFF | - |
| ON | Normal execution (B ≠ 0) | ON | - |
| | No execution (B = 0) | OFF | Set |

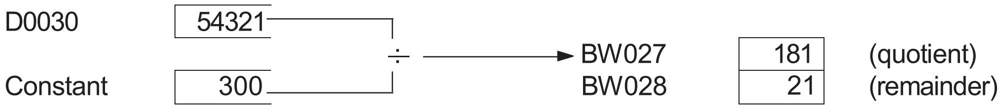
Operand:

| | Name | Device | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Dividend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Divisor | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Quotient | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B010 is ON, the data of D0030 is divided by the constant data 300, and the quotient is stored in D0050 and the remainder is stored in D0051.
If the data of D0030 is 54321, the quotient 181 is stored in D0050 and the remainder 21 is stored in D0051.



Note

- ◆ If divisor (operand B) is 0, ERF (instruction error flag = S0034) is set to ON.
  The ERF (S0034) can be reset to OFF by user program, e.g. -[ RST S0034 ]-.
- ◆ If the index register K is used as operand C, the remainder is ignored.
- ◆ This instruction handles the register data as unsigned integer.

## Instruction-33: Unsigned Double / Single Division

Expression:

Input ─[ A+1.A / B ⟶ C ]─ Output

Function:

When the input is ON, the double-word data of A+1× A is divided by the data of B, and the quotient is stored in C and the remainder in C+1. The data range of A+1× A is 0 to 4294967295, and the data range of B and C is 0 to 65535.
If the quotient is greater than 65535 (overflow), the limit value 65535 is stored in C, 0 is stored in C+1, and the instruction error flag (ERF = S051) is set to ON.

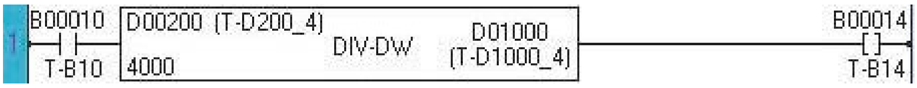
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No execution | OFF | - |
| ON | Normal execution (B ≠ 0) | ON | - |
| | Overflow (B ≠ 0) | ON | Set |
| | No execution (B = 0) | OFF | Set |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Dividend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| B | Divisor | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| C | Quotient | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | |

Example:



When B010 is ON, the double-word data of D0201×D0200 is divided by the constant data 4000, and the quotient is stored in D1000 and the remainder is stored in D1001.
If the data of D0201×D0200 is 332257, the quotient 83 is stored in D1000 and the remainder 257 is stored in D1001.

| | | | | | |
|---|---|---|---|---|---|
| D0201.D0200 | 332257 | ÷ ⟶ | D1000 | 83 | (quotient) |
| Constant | 4000 | | D1001 | 257 | (remainder) |

Note
- ◆ If divisor (operand B) is 0, ERF (instruction error flag = S051) is set to ON.
  The ERF (S051) can be reset to OFF by user program, e.g. -[ RST S051 ]-.
- ◆ If the index register K is used as operand C, the remainder is ignored.
- ◆ This instruction handles the register data as unsigned integer.

## Instruction-34: Float Division

Expression:

Input ⊣[ A+1.A DIV-F B+1.B⟶C+1.C ]— Output

Function:

When the input is ON, the double-word data of A+1× A is divided by the data of B+1xB, and the result is stored in C.1+C. The data range of A, B and C is 3.4e+38 to 3.4e-38.
If the result is greater than 3.4e+38 (overflow), the limit value 3.4e+38 is stored in C.1+C, and the instruction error flag (ERF = S034) is set to ON.

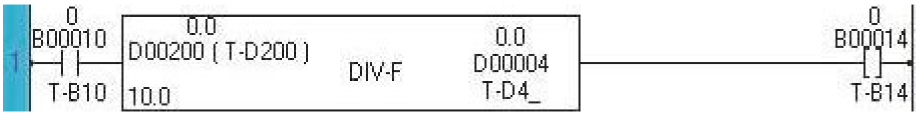
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No execution | OFF | - |
| ON | Normal execution (B ≠ 0) | ON | - |
| | Overflow (B ≠ 0) | ON | Set |
| | No execution (B = 0) | OFF | Set |

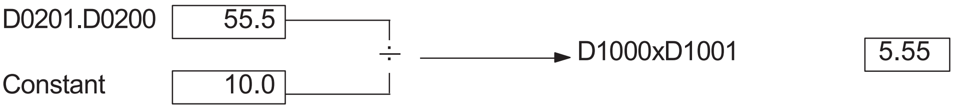
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Dividend | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| B | Divisor | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| C | Quotient | | | | | | | | | | √ | | | | √ | | | | | √ | | |

Example:



When B010 is ON, the float data of D0201×D0200 is divided by the constant data 10.0, and the result is stored in D1000xD1001.
If the data of D0201×D0200 is 55.5, the result is stored in D1000xD1001.



Note

◆ If divisor (operand B) is 0, ERF (instruction error flag = S034) is set to ON.
The ERF (S034) can be reset to OFF by user program, e.g. -[ RST S034 ]-.

## Instruction-35: Addition with Carry

Expression:

Input ─[ A +C B⟶C ]─ Output

Function:

When the input is ON, the data of A, B and the carry flag (CF = S0) are added, and the result is stored in C. If carry is occurred in the operation, the carry flag is set to ON. If the result is greater than 32767 or smaller than -32768, the output is turned ON.
This instruction is used to perform unsigned addition or double-length addition.

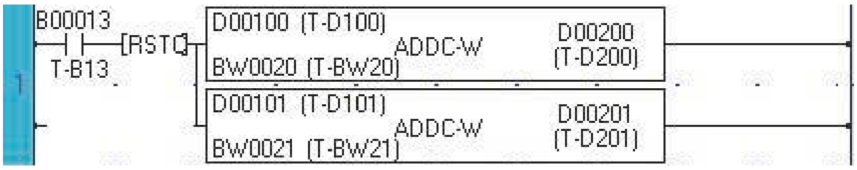
Execution condition:

| | Input | Operation | | | Output | CF |
|---|---|---|---|---|---|---|
| | OFF | No execution | | | OFF | - |
| | ON | Execution | Normal | No Carry | OFF | Reset |
| | | | | Carry Occured | OFF | Set |
| | | | Overflow / Underflow | No carry | ON | Reset |
| | | | | Carry Occured | ON | Set |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Augend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Addend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | Sum | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | √ |

Example:



When B013 is ON, the data of double-length registers D0100×D0101 and BW20×BW21 are added, and the result is stored in D0201×D0200. The RSTC is a instruction to reset the carry flag before starting the calculation.
If the data of D0100×D0101 is 12345678 and BW20×BW21 is 54322, the result 12400000 is stored in D0201×D0200.

D0101.D0100 12345678
+ ⟶ D0201.D0200 12400000
BW21.RW20 54322

## Instruction-36: Subtraction with Carry

Expression:

Input ─[ A -C B →C ]─ Output

Function:

When the input is ON, the data of B and the carry flag (CF = S0) are subtracted from A, and the result is stored in C. If borrow is occurred in the operation, the carry flag is set to ON. If the result is greater than 32767 or smaller than -32768, the output is turned ON.
This instruction is used to perform unsigned subtraction or double-length subtraction.

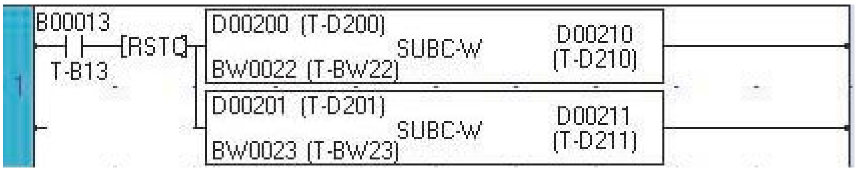
Execution condition:

| Input | Operation | | | Output | CF |
|---|---|---|---|---|---|
| OFF | No execution | | | OFF | - |
| ON | Execution | Normal | No Borrow | OFF | Reset |
| | | | Borrow Occured | OFF | Set |
| | | Overflow / Underflow | No Borrow | ON | Reset |
| | | | Borrow Occured | ON | Set |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Minuend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Subtrahend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *C* | Difference | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | √ | | √ |

Example:



```
B00013         D00200 (T-D200)              D00210
─┤ ├──[RSTC]─  BW0022 (T-BW22)  SUBC-W      (T-D210)
T-B13
               D00201 (T-D201)              D00211
               BW0023 (T-BW23)  SUBC-W      (T-D211)
```

When B013 is ON, the data of double-length register BW23×BW22 is subtracted from the data of D0201×D0200, and the result is stored in D0211×D0210. The RSTC is a instruction to reset the carry flag before starting the calculation.
If the data of D0200×D0201 is 12345678 and BW22×BW23 is 12340000, the result 5678 is stored in D0210×D0211.

D0200.D0201 12345678
-
BW22.BW23 12340000
→ D0210.D0211 5678

# Instruction-37: Increment

Expression:

Input ─[ +1 A ]─ Output

Function:

When the input is ON, the data of A is increased by 1 and stored in A.

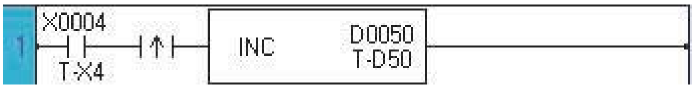
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Operation Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



At the rising edge of X004 changes from OFF to ON, the data of D0050 is increased by 1 and stored in D0050.
If the data of D0050 is 750 before the execution, it will be 751 after the execution.

D0050 750 +1 ⟶ D0050 751

Note
There is no limit value for this instruction. When the data of operand A is 32767 before the execution, it will be -32768 after the execution.

## Instruction-38: Decrement

Expression:

Input ─[ -1 A ]─ Output

Function:

When the input is ON, the data of A is decreased by 1 and stored in A.

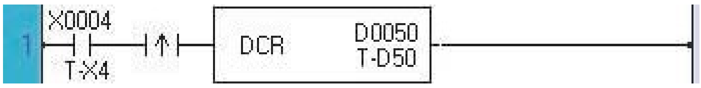
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | peration Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



At the rising edge of X005 changes from OFF to ON, the data of D0050 is decreased by 1 and stored in D0050.
If the data of D0050 is 1022 before the execution, it will be 1021 after the execution.

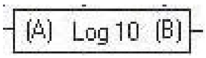
D0050 | D0050

1022 -1 ⟶ 1021

Note
There is no limit value for this instruction. When the data of operand A is -32768 before the execution, it will be 32767 after the execution.

## Instruction-39: Log (10)

Symbol

Expression: 

Function:

This instruction calculates the Log to the base 10 value of the Operand A.1+A and stores the result in Operand in B.1+B. Both the operands are float.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | ON |

Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| *A* | Source |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ | √ | √ |
| *B* | Destination |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ |  | √ |

Example :

When B020 is ON, the data of D0100.D0101 is calculated as Log to the base 10, and the result is stored in BW020.BW021

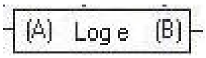
For example, if D0100.D0101 is having value 100, then its Log to the base 10, value 2 will be stored in BW020.BW021.

| D0100.D0101 | Log 10 | BW020.BW021 |
|---|---|---|
| 100 \| |  | 2 |

# Instruction-40: Log (e)

Symbol

Expression:



Function:

This instruction calculates the Log to the base e value of the Operand A.1+A and stores the result in Operand in B.1+B. Both the operands are float.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Source | | | | | | | | | | √ | | | | √ | | | | | √ | | √ | √ |
| *B* | Destination | | | | | | | | | | √ | | | | √ | | | | | √ | | | √ |

Example :

When B020 is ON, the data of D0100.D0101 is calculated as Log to the base "e", and the result is stored in BW020.BW021.

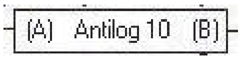
For example, if D0100.D0101 is having value 10, then its Log to the base "e", value 2.3026 will be stored in BW020.BW021.

| D0100.D0101 | Log e | BW020.BW021 |
|---|---|---|
| 10 | | 2.3026 |

## Instruction-41: Antilog (10)

Symbol

Expression:



Function:

This instruction calculates the Antilog to the base 10 value of the Operand A.1+A and stores the result in Operand in B.1+B. Both the operands are float.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | ON |

Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |  |
| *A* | Source |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ |  | √ | √ |
| *B* | Destination |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ |  |  | √ |

Example :

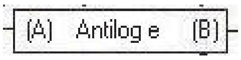
When B020 is ON, the data of D0100.D0101 is calculated as Antilog to the base "10", and the result is stored in BW020.BW021.

For example, if D0100.D0101 is having value 2, then its Antilog to the base "10", value 100 will be stored in BW020.BW021.

| D0100.D0101 | Antilog 10 | BW020.BW021 |
|---|---|---|
| 2 \| |  | 100 \| |

## Instruction-42: Antilog (e)

Symbol

Expression: 

Function:

This instruction calculates the Antilog to the base "e" value of the Operand A.1+A and stores the result in Operand in B.1+B. Both the operands are float.

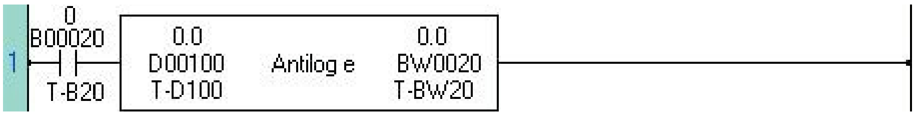
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | | | √ | | | | √ | | | | | √ | √ | √ |
| *B* | Destination | | | | | | | | | | √ | | | | √ | | | | | √ | | √ |

Example :



When B020 is ON, the data of D0100.D0101 is calculated as Antilog to the base "e", and the result is stored in BW020.BW021.

For example, if D0100.D0101 is having value 1, then its Antilog to the base "e", value 2.7183 will be stored in BW020.BW021.

| D0100.D0101 | Antilog e | BW020.BW021 |
|---|---|---|
| 1 | | 2.7183 |

## Instruction-43: Square Root

Symbol

Expression: -[ A Square root B ]-

Function:

This instruction calculates the Square root value of the Operand A.1+A and stores the result in Operand in B.1+B. Both the operands are float. If source value is negative, the result will be “0” and output will be turned OFF.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Normal Execution | ON |
| | Source value is negative (No execution) | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | | | √ | | | | √ | | | | | √ | √ | √ |
| *B* | Destination | | | | | | | | | | √ | | | | √ | | | | | √ | | √ |

Example :

When B020 is ON, the square root of the flaoting point value in D100.D101 is aclculated, and the result is stored in BW020.BW021.

For example, if D0100.D0101 is having value 25, then its square root value 5.0 will be stored in BW020.BW021.

| D0100.D0101 | Square root | BW020.BW021 |
|---|---|---|
| 25.0 | | 5.0 |

## Instruction-44: Greater Than

Expression:

Input ┤[ A > B ]├ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is greater than B, the output is turned ON.

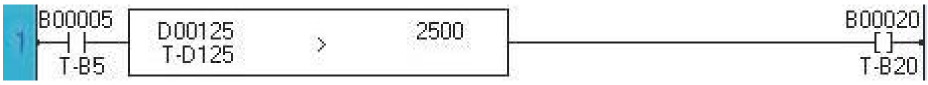
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A > B | ON |
| | | A ≤ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data 2500, and if the data of D0125 is greater than 2500, R0020 is turned ON.
If the data of D0125 is 3000, the comparison result is true. Consequently, B0020 is turned ON.

D0125 3000 > Constant 2500 → B0020 is ON

If the data of D0125 is -100, the comparison result is false. Consequently, B0005 is turned OFF.

D0125 -100 ≤ Constant 2500 → B0020 is OFF

Note
This instruction deals with the data as signed integer (-32768 to 32767).

## Instruction-45: Double Word Greater Than

Expression:

Input –[ A D> B ]– Output

Function:

When the input is ON, the double-word data of A+1× A and B+1× B are compared, and if A+1× A is greater than B+1× B, the output is turned ON.

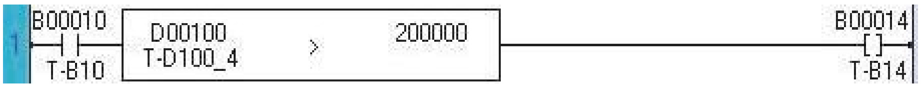
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A > B+1.B | ON |
| | | A+1.A ≤ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |

Example:



When B010 is ON, the data of D0101×D0100 is compared with the constant data 200000, and if the data of D0101×D0100 is greater than 200000, B014 is turned ON.
If the data of D0101×D0100 is 250000, the comparison result is true. Consequently, B014 is turned ON.

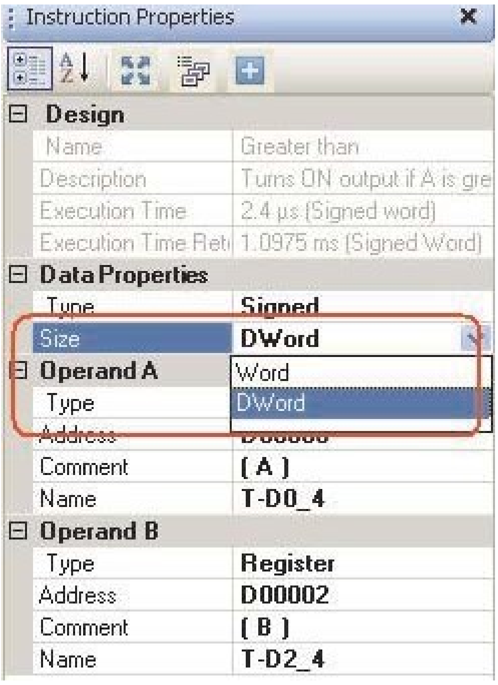
D0101.D0100 [250000] > Constant [200000] → B0014 is ON

If the data of D0101×D0100 is -100, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 [-100] ≤ Constant [200000] → B0014 is OFF

Note
This instruction deals with the data as double word integer (-2147483648 to 2147483648).

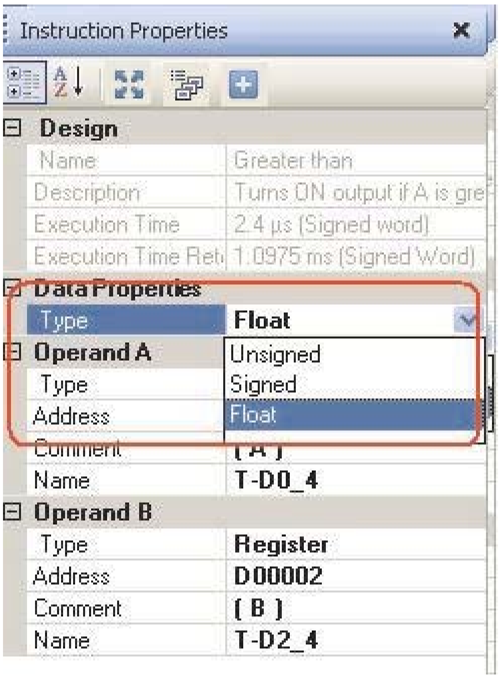
When user select “Double Word greater Than” function and place it in logic block, “Property” docker window occurs to the right side of the application window; where user can select “Size” Proprty and change “Word” to “DWord” as shown below:



Instruction Properties

| | |
|---|---|
| **Design** | |
| Name | Greater than |
| Description | Turns ON output if A is gre |
| Execution Time | 2.4 µs (Signed word) |
| Execution Time Ret | 1.0975 ms (Signed Word) |
| **Data Properties** | |
| Type | Signed |
| Size | DWord |
| **Operand A** | Word<br>DWord |
| Type | |
| Address | D00000 |
| Comment | ( A ) |
| Name | T-D0_4 |
| **Operand B** | |
| Type | Register |
| Address | D00002 |
| Comment | ( B ) |
| Name | T-D2_4 |

Then by selecting “Size” property entry can be changed to “Signed”, “Unsigned” or “Float” as shown below:



Instruction Properties

| | |
|---|---|
| **Design** | |
| Name | Greater than |
| Description | Turns ON output if A is gre |
| Execution Time | 2.4 µs (Signed word) |
| Execution Time Ret | 1.0975 ms (Signed Word) |
| **Data Properties** | |
| Type | Float |
| **Operand A** | Unsigned<br>Signed<br>Float |
| Type | |
| Address | |
| Comment | ( A ) |
| Name | T-D0_4 |
| **Operand B** | |
| Type | Register |
| Address | D00002 |
| Comment | ( B ) |
| Name | T-D2_4 |

## Instruction-46: Unsigned Greater Than

Expression:

Input ─[ A > B ]─ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is greater than B, the output is turned ON.

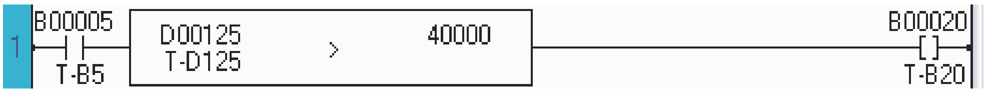
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A > B | ON |
| | | A ≤ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data 40000, and if the data of D0125 is greater than 40000, B0020 is turned ON.
If the data of D0125 is 52000, the comparison result is true. Consequently, B0020 is turned ON.

D0125 52000 > Constant 40000 ──► B0020 is ON

If the data of D0125 is 21000, the comparison result is false. Consequently, B0005 is turned OFF.

D0125 21000 ≤ Constant 40000 ──► B0020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

## Instruction-47: Float Greater Than

Expression:

Input ─[ A > B ]─ Output

Function:

When the input is ON, the float data of A+1× A and B+1× B are compared, and if A+1× A is greater than B+1× B, the output is turned ON.

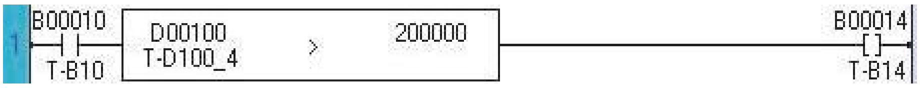
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A > B+1.B | ON |
| | | A+1.A ≤ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| *B* | Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |

Example:



When B010 is ON, the data of D0101×D0100 is compared with the constant data 200000.467, and if the data of D0101×D0100 is greater than 200000.467, B014 is turned ON.
If the data of D0101×D0100 is 250000.123, the comparison result is true. Consequently, B014 is turned ON.

D0101.D0100 250000.123 > Constant 200000.467 ⟶ B0014 is ON

If the data of D0101×D0100 is -100, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 -100.012 ≤ Constant 200000.467 ⟶ B0014 is OFF

Note
This instruction deals with the data as float (-3.4e + 38 to 3.4e + 38).

## Instruction-48: Greater Than or Equal To

Expression:

Input ⊣[ A >= B ]⊢ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is greater than or equal to B, the output is turned ON.

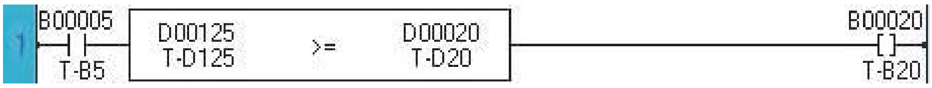
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≥ B | ON |
| | | A < B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the data of D0020, and if the data of D0125 is greater than or equal to the data of D0020, B020 is turned ON.
If the data of D0125 is 3000 and that of D0020 is 3000, the comparison result is true.
Consequently, B020 is turned ON.

D0125 $\boxed{3000}$ ≥ D0020 $\boxed{3000}$ ⟶ B020 is ON

If the data of D0125 is -1500 and that of D0020 is 0, the comparison result is false. Consequently, B020 is turned OFF.

D0125 $\boxed{-1500}$ < D0020 $\boxed{0}$ ⟶ B020 is OFF

Note
This instruction deals with the data as signed integer (-32768 to 32767).

## Instruction-49: Double Word Greater Than or Equal To

Expression:

Input ─[ A D>= B ]─ Output

Function:

When the input is ON, the data of A+1 X A and the data of B+1 X B are compared, and if A+1.A is greater than or equal to B+1.B, the output is turned ON.

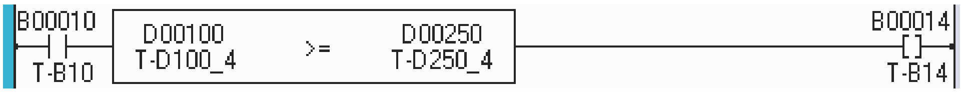
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A ≥ B+1.B | ON |
| | | A+1.A < B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |

Example:



When B010 is ON, the double-word data of D0101×D0100 is compared with the double-word data of D0251×D0250, and if the data of D0101×D0100 is greater than or equal to the data of D0251×D0250, B014 is turned ON.
If the data of D0101×D0100 is 250000 and D0251×D0250 is 200000, B014 is turned ON.

D0101.D100 250000 ≥ D0251.D0250 200000 → B014 is ON

If the data of D0101xD100 is -100 and that of D0251xD0250 is 0, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 -100 < D0251.D0250 0 → B014 is OFF

Note
This instruction deals with the data as double word integer (-2147483648 to 2147483648).

# Instruction-50: Unsigned Greater Than or Equal To

Expression:

Input ⊣[ A >= B ]⊢ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is greater than or equal to B, the output is turned ON.

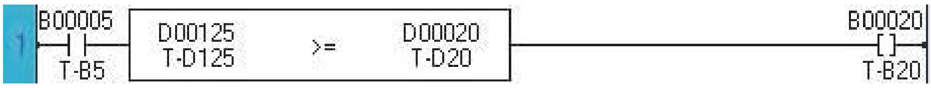
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≥ B | ON |
| | | A < B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the data of D0020, and if the data of D0125 is greater than or equal to the data of D0020, B020 is turned ON.
If the data of D0125 is 40000 and that of D0020 is 40000, the comparison result is true. Consequently, B020 is turned ON.

D0125 [40000] ≥ D0020 [40000] ⟶ B020 is ON

If the data of D0125 is 15000 and that of D0020 is 20000, the comparison result is false. Consequently, B020 is turned OFF.

D0125 [15000] < D0020 [20000] ⟶ B020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

# Instruction-51: Float Greater Than or Equal To

Expression:

Input ┤[ A >= B ]├ Output

Function:

When the input is ON, the float data of A and the float data of B are compared, and if A is greater than or equal to B, the output is turned ON.

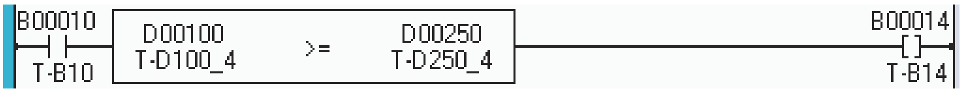
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≥ B | ON |
| | | A < B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| *B* | Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |

Example:



When B010 is ON, the double-word data of D0101×D0100 is compared with the double-word data of D0251×D0250, and if the data of D0101×D0100 is greater than or equal to the data of D0251×D0250, B014 is turned ON.

If the data of D0101×D0100 is 250000.123 and D0251×D0250 is 200000.123, B014 is turned ON.

D0101.D100 250000.123 ≥ D0251.D0250 200000.123 → B014 is ON

If the data of D0101xD100 is -100.467 and that of D0251xD0250 is 0.123, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 -100.467 < D0251.D0250 0.123 → B014 is OFF

# Instruction-52: Equal

Expression:

Input ⊣[ A = B ]⊢ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is equal to B, the output is turned ON.

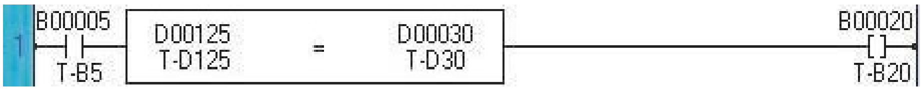
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A = B | ON |
| | | A ≠ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | ompared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | eference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the data of D0030, and if the data of D0125 is equal to the data of D0030, B020 is turned ON.

If the data of D0125 is 3000 and that of D0020 is 3000, the comparison result is true. Consequently, B020 is turned ON.

D0125 $\boxed{3000}$ = D0030 $\boxed{3000}$ ——→ B020 is ON

If the data of D0125 is -1500 and that of D0020 is 0, the comparison result is false. Consequently, B020 is turned OFF.

D0125 $\boxed{-1500}$ ≠ D0030 $\boxed{0}$ ——→ B020 is OFF

Note

This instruction deals with the data as signed integer (-32768 to 32767).

## Instruction-53: Double Word Equal

Expression:

Input ─[ A D= B ]─ Output

Function:

When the input is ON, the data of A+1.A and the data of B+1.B are compared, and if A+1.A is equal to B+1.B, the output is turned ON.

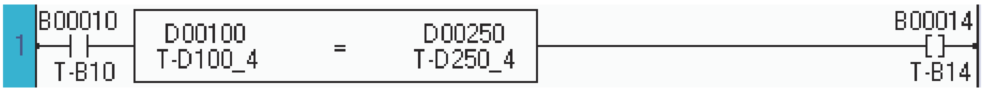
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A = B+1.B | ON |
| | | A+1.A ≠ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |

Example:



When B010 is ON, the double-word data of D0101×D0100 is compared with the double-word data of D0251×D0250, and if the data of D0101×D0100 is equal to the data of D0251×D0250, B014 is turned ON.

If the data of D0101XD0100 is 250000 and that of D0251XD0250 is 250000, the comparison result is true. Consequently, B014 is turned ON.

D0101.D0100 $\boxed{250000}$ = D0251.D0250 $\boxed{250000}$ ──► B014 is ON

If the data of D0101x D0100 is -100 and that of D0251xD0250 is 0, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 $\boxed{-100}$ ≠ D0251.D0250 $\boxed{0}$ ──► B014 is OFF

Note

This instruction deals with the data as double word integer (-2147483648 to 2147483648).

## Instruction-54: Float Equal

Expression:

Input ─[ A D= B ]─ Output

Function:

When the input is ON, the float data of A+1.A and the float data of B+1.B are compared, and if A+1.A is equal to B+1.B, the output is turned ON.

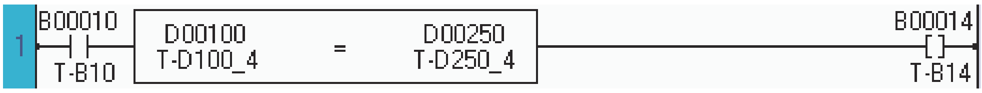
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A = B+1.B | ON |
| | | A+1.A ≠ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | | √ | |
| *B* | Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | | √ | |

Example:



When B010 is ON, the float data of D0101×D0100 is compared with the float data of D0251×D0250, and if the data of D0101×D0100 is equal to the data of D0251×D0250, B014 is turned ON.

If the data of D0101XD0100 is 250000.123 and that of D0251XD0250 is 250000.123, the comparison result is true. Consequently, B014 is turned ON.

D0101.D0100 $\boxed{250000.123}$ = D0251.D0250 $\boxed{250000.123}$ → B014 is ON

If the data of D0101x D0100 is -100 and that of D0251xD0250 is 0, the comparison result is false. Consequently, B014 is turned OFF.

D0101.D0100 $\boxed{-100.123}$ ≠ D0251.D0250 $\boxed{0.467}$ → B014 is OFF

## Instruction-55: Unsigned Equal

Expression:

Input ┤[ A = B ]├ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is equal to B, the output is turned ON.

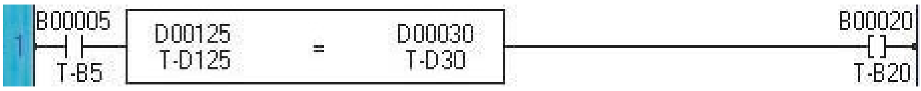
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A = B | ON |
| | | A ≠ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the data of D0030, and if the data of D0125 is equal to the data of D0030, B020 is turned ON.

If the data of D0125 is 35000 and that of D0020 is 35000, the comparison result is true. Consequently, B020 is turned ON.

D0125 $\boxed{35000}$ = D0030 $\boxed{35000}$ ──► B020 is ON

If the data of D0125 is 1500 and that of D0020 is 4000, the comparison result is false. Consequently, B020 is turned OFF.

D0125 $\boxed{1500}$ ≠ D0030 $\boxed{4000}$ ──► B020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

## Instruction-56: Not equal

Expression:

Input ─[ A <> B ]─ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is not equal to B, the output is turned ON.

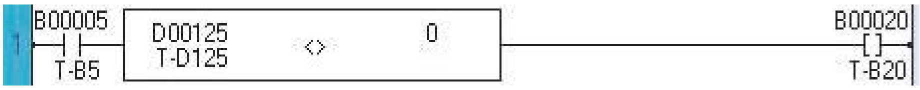
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≠ B | ON |
| | | A = B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A ( | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B ( | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data 0, and if the data of D0125 is not 0, B0020 is turned ON.

If the data of D0125 is 10, the comparison result is true. Consequently, B0020 is turned ON.

D0125 | 10 | ≠ Constant | 0 | ⟶ B0020 is ON

If the data of D0125 is 0, the comparison result is false. Consequently, B0020 is turned OFF.

D0125 | 0 | = Constant | 0 | ⟶ B0020 is OFF

> Note
> This instruction deals with the data as signed integer (-32768 to 32767).

# Instruction-57: Double Word Not Equal

Expression:

Input ─[ A D<> B ]─ Output

Function:

When the input is ON, the data of A+1.A and the data of B+1.B are compared, and if A+1.A is not equal to B+1.B, the output is turned ON.

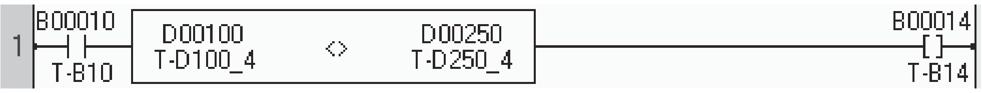
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A ≠ B+1.B | ON |
| | | A+1.A = B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | √ | |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | | √ | |

Example:



When B010 is ON, the double-word data of D0101×D0100 is compared with the double-word data of D0251×D0250, and if the data of D0101×D0100 is not equal to the data of D0251×D0250, B014 is turned ON.

If the data of D0101.D0100 is 250000 and D0251xD0250 is 200000, B014 is turned ON.

D0101.D0100 250000 ≠ D0251.D0250 250000 → B014 is ON

If the data of D0101.D0100 is -100 and D0251.D0250 is -100, B014 is turned OFF.

D0101.D0100 -100 = D0251.D0250 -100 → B014 is OFF

Note
This instruction deals with the data as double word integer (-2147483648 to 2147483648).

## Instruction-58: Unsigned Not Equal

Expression:

Input ─[ A <> B ]─ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is not equal to B, the output is turned ON.

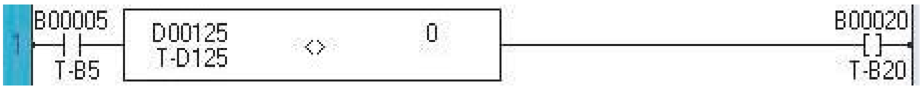
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≠ B | ON |
| | | A = B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data 0, and if the data of D0125 is not 0, B0020 is turned ON.

If the data of D0125 is 41000, the comparison result is true. Consequently, B0020 is turned ON.

D0125 | 41000 | ≠ Constant | 0 | ⟶ B0020 is ON

If the data of D0125 is 0, the comparison result is false. Consequently, B0020 is turned OFF.

D0125 | 0 | = Constant | 0 | ⟶ B0020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

## Instruction- 59: Float Not Equal

Expression:

Input ┤[ A D<> B ]├ Output

Function:

When the input is ON, the float data of A+1.A and the float data of B+1.B are compared, and if A+1.A is not equal to B+1.B, the output is turned ON.

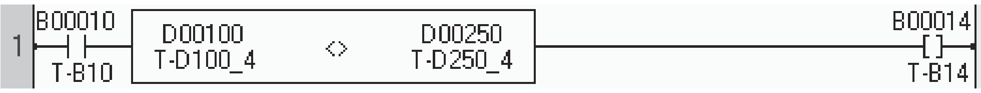
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A ≠ B+1.B | ON |
| | | A+1.A = B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | (Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| B | (Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |

Example:



When B010 is ON, the float data of D0101×D0100 is compared with the float data of D0251×D0250, and if the data of D0101×D0100 is not equal to the data of D0251×D0250, B014 is turned ON.

If the data of D0101.D0100 is 250000 and D0251xD0250 is 200000, B014 is turned ON.

D0101.D0100 $\boxed{250000.123}$ ≠ D0251.D0250 $\boxed{200000.467}$ ⟶ B014 is ON

If the data of D0101.D0100 is -100 and D0251.D0250 is -100, B014 is turned OFF.

D0101.D0100 $\boxed{-100.123}$ = D0251.D0250 $\boxed{-100.123}$ ⟶ B014 is OFF

## Instruction-60: Less than

Expression:

Input ─[ A < B ]─ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is less than B, the output is turned ON.

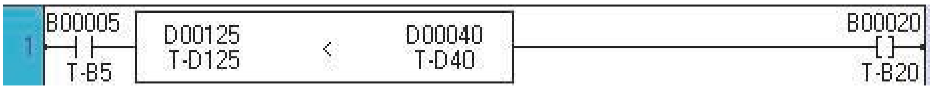
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A < B | ON |
| | | A ≥ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B005 is ON, the data of D0125 is compared with the data of D0040, and if the data of D0125 is less than the data of D0040, B020 is turned ON.

If the data of D0125 is 10 and that of D0040 is 15, the comparison result is true. Consequently, B020 is turned ON.

D0125 10 < D0040 15 ──→ B020 is ON

If the data of D0125 is 0 and that of D0040 is -50, the comparison result is false. Consequently, B020 is turned OFF.

D0125 0 ≤ D0040 0 ──→ B020 is OFF

Note
This instruction deals with the data as signed integer (-32768 to 32767).

### Instruction-61: Double Word Less Than

Expression:

Input ⊣[ A D< B ]⊢ Output

Function:

When the input is ON, the data of A+1.A and the data of B+1.B are compared, and if A+1.A is less than B+1.B, the output is turned ON.

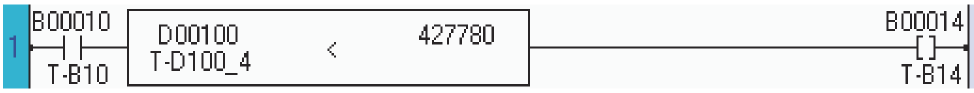
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A < B+1.B | ON |
| | | A+1.A ≥ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |

Example:


```
  B00010   D00100              427780                          B00014
1 ─┤ ├─   T-D100_4    <                                    ─[ ]─
   T-B10                                                        T-B14
```

When B010 is ON, the data of D0101.D0100 is compared with the constant data 427780, and if the data of D0101.D0100 is less than the data 427780, B014 is turned ON.

If the data of D0101.D0100 is 250000 B014 is turned ON.

D0101.D100 $\boxed{250000}$ < Constant $\boxed{427780}$ ⟶ B014 is ON

If the data of D0101Xd100 is 430000, B014 is turned OFF.

D0101.d0100 $\boxed{430000}$ ≤ Constant $\boxed{427780}$ ⟶ B014 is OFF

Note
This instruction deals with the data as double word integer (-2147483648 to 2147483648).

# Instruction-62: Unsigned Less Than

Expression:

Input −[ A < B ]− Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is less than B, the output is turned ON.

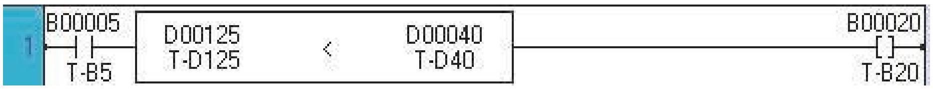
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A < B | ON |
| | | A ≥ B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B005 is ON, the data of D0125 is compared with the data of D0040, and if the data of D0125 is less than the data of D0040, B020 is turned ON.

If the data of D0125 is 43000 and that of D0040 is 45000, the comparison result is true. Consequently, B020 is turned ON.

D0125 43000 < D0040 45000 → B020 is ON

If the data of D0125 is 50000 and that of D0040 is 50000, the comparison result is false. Consequently, B020 is turned OFF.

D0125 50000 ≥ D0040 50000 → B020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

## Instruction-63: Float Less Than

Expression:

Input ┤[ A D< B ]├ Output

Function:

When the input is ON, the float data of A+1.A and the float data of B+1.B are compared, and if A+1.A is less than B+1.B, the output is turned ON.

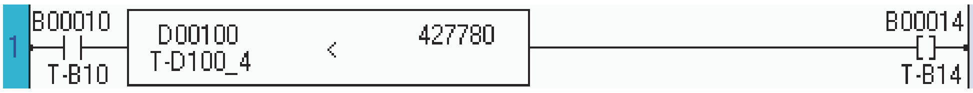
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A < B+1.B | ON |
| | | A+1.A ≥ B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| *B* | Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |

Example:



When B010 is ON, the data of D0101.D0100 is compared with the constant data 427780, and if the data of D0101.D0100 is less than the data 427780, B014 is turned ON.

If the data of D0101.D0100 is 250000 B014 is turned ON.

D0101.D100 250000.123 < Constant 427780.467 → B014 is ON

If the data of D0101Xd100 is 430000, B014 is turned OFF.

D0101.d0100 430000.123 ≤ Constant 427780.467 → B014 is OFF

# Instruction-64: Less Than or Equal

Expression:

Input –[ A <= B ]– Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is less than or equal to B, the output is turned ON.

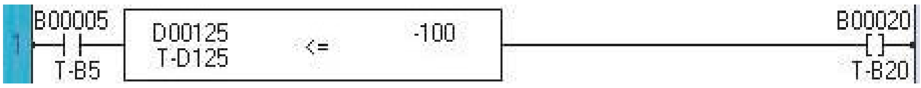
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≤ B | ON |
| | | A > B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data -100, and if the data of D0125 is less than or equal to -100, B020 is turned ON.

If the data of D0125 is -150, the comparison result is true. Consequently, B020 is turned ON..

D0125 $\boxed{-150}$ < Constant $\boxed{-100}$ ⟶ B0020 is ON

If the data of D0125 is 0, the comparison result is false. Consequently, B0020 is turned OFF.

D0125 $\boxed{0}$ ≥ Constant $\boxed{-100}$ ⟶ B0020 is OFF

Note
This instruction deals with the data as signed integer (-32768 to 32767).

# Instruction-65: Double Word Less Than or Equal

Expression:

Input ─[ A D<= B ]─ Output

Function:

When the input is ON, the data of A+1.A and the data of B+1.B are compared, and if A+1.A is less than or equal to B+1.B, the output is turned ON.

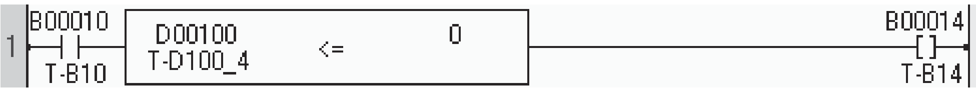
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A ≤ B+1.B | ON |
| | | A+1.A > B+1.B | OFF |

Operand:

| | Name | Device | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |
| B | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | √ | √ | |

Example:



When B010 is ON, the data of D0101xD100 is compared with the constant data 0, and if the data of D0101xD0100 is less than or equal to 0, B014 is turned ON.

If the data of D0101xD0100 is -1, the comparison result is true. Consequently, B014 is turned ON.

D0101.D0100 -1 < Constant 0 → B014 is ON

If the data of D0101.D0100 is 10000, B014 is turned OFF.

D0101.D0100 10000 ≤ Constant 0 → B014 is OFF

Note
This instruction deals with the data as double word integer (-2147483648 to 2147483648).

# Instruction-66: Unsigned Less Than or Equal

Expression:

Input ─┤ A <= B ├─ Output

Function:

When the input is ON, the data of A and the data of B are compared, and if A is less than or equal to B, the output is turned ON.

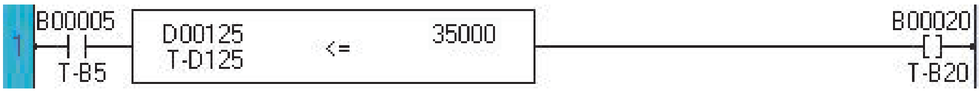
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A ≤ B | ON |
| | | A > B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Compared Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Reference Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Example:



When B0005 is ON, the data of D0125 is compared with the constant data 35000, and if the data of D0125 is less than or equal to 35000, B020 is turned ON.

If the data of D0125 is 35000, the comparison result is true. Consequently, B020 is turned ON..

D0125 $\boxed{35000}$ ≤ Constant $\boxed{35000}$ ⟶ B0020 is ON

If the data of D0125 is 0, the comparison result is false. Consequently, B0020 is turned OFF.

D0125 $\boxed{38000}$ > Constant $\boxed{35000}$ ⟶ B0020 is OFF

Note
This instruction deals with the data as unsigned integer (0 to 65535).

## Instruction-67: Float Less Than or Equal

Expression:

Input ─[ A D<= B ]─ Output

Function:

When the input is ON, the float data of A+1.A and the float data of B+1.B are compared, and if A+1.A is less than or equal to B+1.B, the output is turned ON.

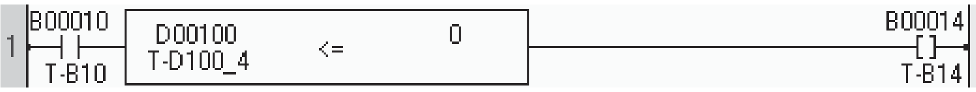
Execution condition:

| Input | Operation | | Output |
|---|---|---|---|
| OFF | No execution | | OFF |
| ON | Execution | A+1.A ≤ B+1.B | ON |
| | | A+1.A > B+1.B | OFF |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Compared Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |
| B | Reference Data | | | | | | | | | | √ | | | | √ | | | | | √ | √ | |

Example:



When B010 is ON, the data of D0101xD100 is compared with the constant data 0, and if the data of D0101xD0100 is less than or equal to 0, B014 is turned ON.

If the data of D0101xD0100 is -1, the comparison result is true. Consequently, B014 is turned ON.

D0101.D0100 -1.123 ≤ Constant 0 → B014 is ON

If the data of D0101.D0100 is 10000, B014 is turned OFF.

D0101.D0100 10000.123 ≤ Constant 0 → B014 is OFF

## Instruction-68: Logic AND

Expression:

Input -[ A AND B ——■C ] Output

Function:

When the input is ON, this instruction finds logical AND of A and B, and stores the result in C.

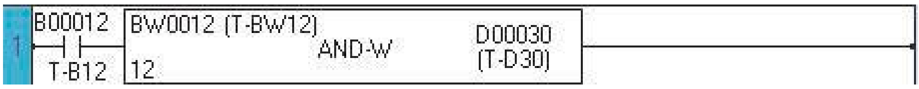
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| B | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| C | ND | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B0012 is ON, logical AND operation is executed for the data of BW012 and the constant data 12, and the result is stored in D0030.

If the data of BW012 is 140, the result 1680 is stored in D0030.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW012 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| | 3 | | | | 4 | | | | 5 | | | | 6 | | | |

AND

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Constant | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | F | | | | F | | | | 0 | | | | 0 | | | |

↓

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0030 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | | | | 4 | | | | 0 | | | | 0 | | | |

## Instruction-69: Logic OR

Expression:

Input ⊣[ A OR B ——■C ] Output

Function:

When the input is ON, this instruction finds logical OR of A and B, and stores the result in C.

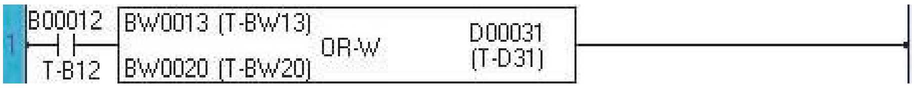
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *B* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *C* | OR | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When B012 is ON, logical OR operation is executed for the data of BW13 and BW20, and the result is stored in D0031.

If the data of BW13 is H5678 and BW20 is H4321, the result H5779 is stored in D0031.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW013 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | 5 | | | | 6 | | | | 7 | | | | 8 | | | |

OR

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW020 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| | 4 | | | | 3 | | | | 2 | | | | 1 | | | |

↓

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0031 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| | 5 | | | | 7 | | | | 7 | | | | 9 | | | |

## Instruction-70: Logic Exclusive OR

Expression:

Input ┤[ A EOR B ——■C ] Output

Function:

When the input is ON, this instruction finds logical exclusive OR of A and B, and stores the result in C.

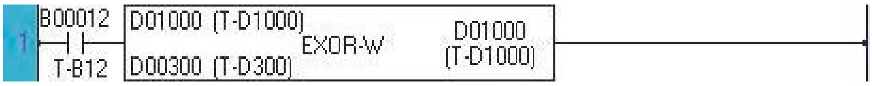
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | √ |
| *B* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | √ |
| *C* | OR | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ |

Example:


B00012 T-B12 D01000 (T-D1000) D00300 (T-D300) EXOR-W D01000 (T-D1000)

When B012 is ON, exclusive OR operation is executed for the data of D1000 and D0300, and the result is stored in D1000.

If the data of D1000 is H5678 and D0300 is H4321, the result H1559 is stored in D1000.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D1000 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

5 6 7 8

OR

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0300 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

4 3 2 1

↓

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D1000 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

1 5 5 9

## Instruction-71: Logic Shift - 1 bit Shift Right

Expression:

Input ⊣[ SHR-1 A ]⊢ Output

Function:

When the input is ON, the data of register A is shifted 1 bit to the right (LSB direction). 0 is stored in the left most bit (MSB). The pushed out bit state is stored in the carry flag (CF = S0). After the operation, if the right most bit (LSB) is ON, the output is turned ON.

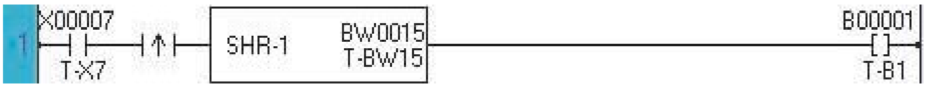
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No execution | | OFF | --- |
| ON | Execution | When LSB = 1 | ON | Set or reset |
| | | When LSB = 0 | OFF | Set or reset |

Operand:

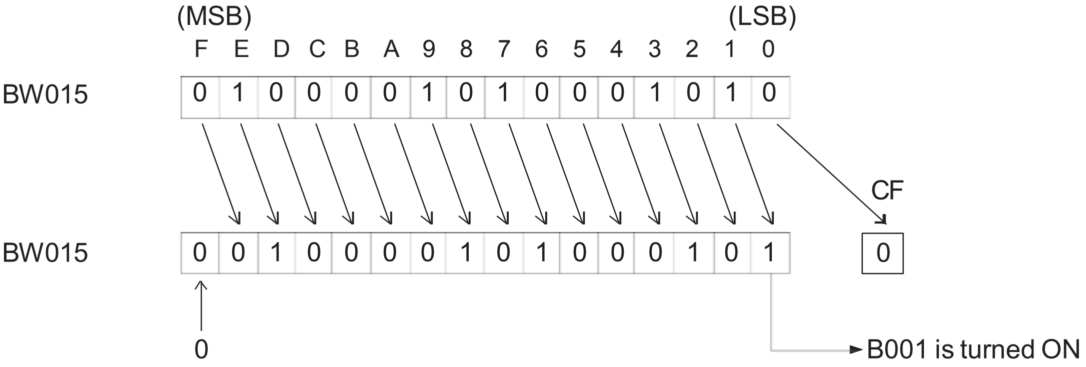
| | Name | Device | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Operation Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | |

Example:



When X007 is changed from OFF to ON, the data of BW15 is shifted 1 bit to the right.

The figure below shows an operation example.



| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | CF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW015 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| BW015 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

## Instruction-72: Logic Shift - 1 bit Shift Left

Expression:

Input ⊣[ SHL-1 A ]⊢ Output

Function:

When the input is ON, the data of register A is shifted 1 bit to the left (MSB direction). 0 is stored in the right most bit (LSB). The pushed out bit state is stored in the carry flag (CF = S0). After the operation, if the left most bit (MSB) is ON, the output is turned ON.

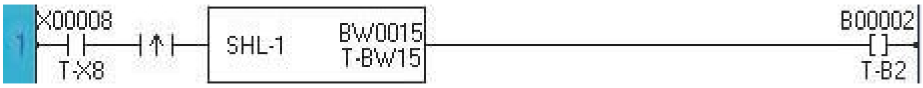
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No execution | | OFF | --- |
| ON | Execution | When MSB = 1 | ON | Set or reset |
| | | When MSB = 0 | OFF | Set or reset |

Operand:

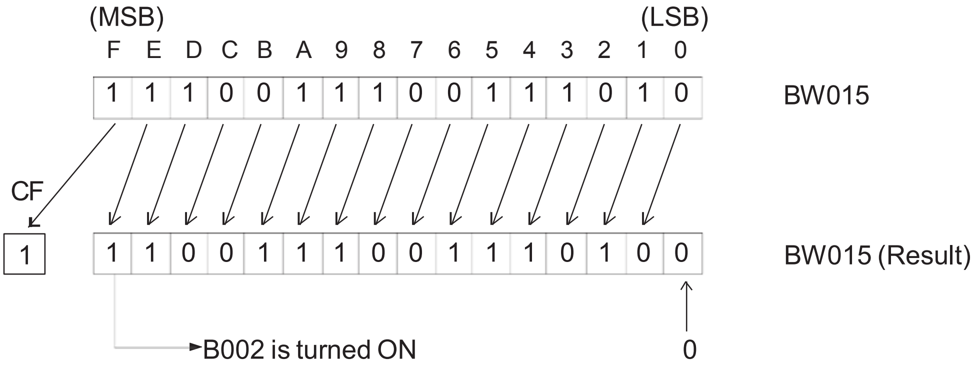
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | peration Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X008 is changed from OFF to ON, the data of BW15 is shifted 1 bit to the left.

The figure below shows an operation example.



## Instruction-73: Logic Shift - n bits Shift Right

Expression:

Input ─[ A SHR n ──➘B ]─ Output

Function:

When the input is ON, the data of register A is shifted n bits to the right (LSB direction) including the carry flag (CF = S0), and stored in B. 0 is stored in upper n bits. After the operation, if the right most bit (LSB) is ON, the output is turned ON.

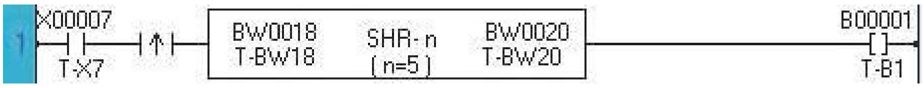
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No execution | | OFF | --- |
| ON | Execution | When LSB = 1 | ON | Set or reset |
| | | When LSB = 0 | OFF | Set or reset |

Operand:

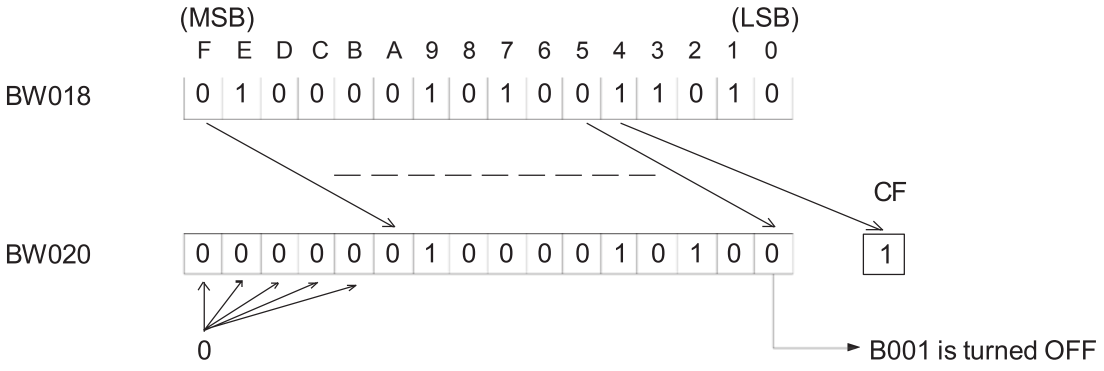
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | |
| *n* | Shift bits | | | | | | | | | | | | | | | | | | | | 1 - 16 | |
| *B* | estination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X007 is changed from OFF to ON, the data of BW18 is shifted 5 bits to the right and the result is stored in BW20.

The figure below shows an operation example.



# Instruction-74: Logic Shift - n bits Shift Left

Expression:

Input ─[ A SHL n ──▸B ]─ Output

Function:

When the input is ON, the data of register A is shifted n bits to the left (MSB direction) including the carry flag (CF = S0), and stored in B. 0 is stored in lower n bits. After the operation, if the left most bit (MSB) is ON, the output is turned ON.

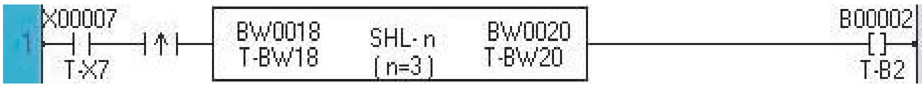
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No execution | | OFF | --- |
| ON | Execution | When MSB = 1 | ON | Set or reset |
| | | When MSB = 0 | OFF | Set or reset |

Operand:

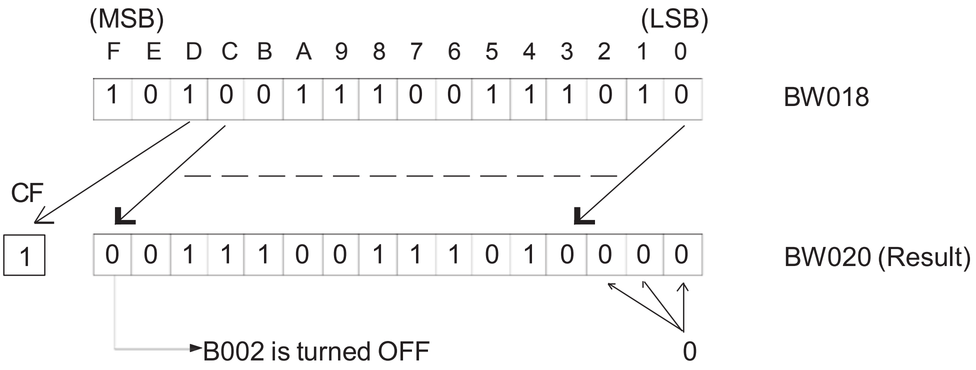
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *n* | Shift bits | | | | | | | | | | | | | | | | | | | | 1 - 16 | |
| *B* | estination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X007 is changed from OFF to ON, the data of BW18 is shifted 3 bits to the left and the result is stored in BW20.

The figure below shows an operation example.



| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | BW018 |
| CF 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | BW020 (Result) |

## Instruction-75: Shift Register

Expression:

```
        Data input    -|D   SR   Q|-      Output
        Shift input   -|S         |
        Enable input  -|E       A_|
```

Function:

While the enable input is ON, this instruction shifts the data of the bit table, size n starting with A, 1 bit to the left (upper address direction) when the shift input is ON. The state of the data input is stored in A. The pushed out bit state is stored in the carry flag (CF = S0).
When the enable input is OFF, all bits in the table and the carry flag are reset to OFF.

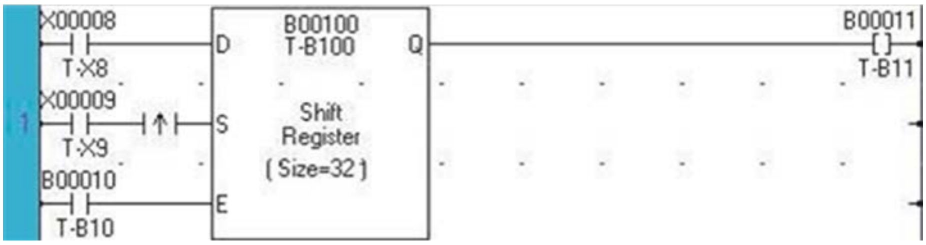
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | Resets all bits in the bit table | | OFF | Reset |
| ON | When the shift input is ON | Shift execution | ON | Set or reset |
| | When the shift input is OFF | No execution | OFF | --- |

Operand:

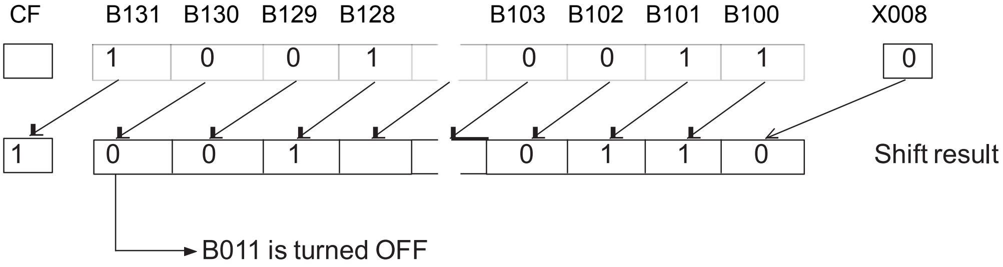
| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Leading Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | | |
| *n* | Device Size | | | | | | | | | | | | | | | | | | | | | 1 - 64 | |

Example:

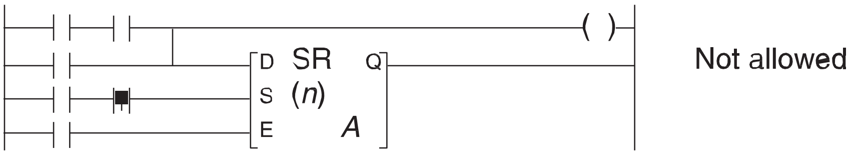


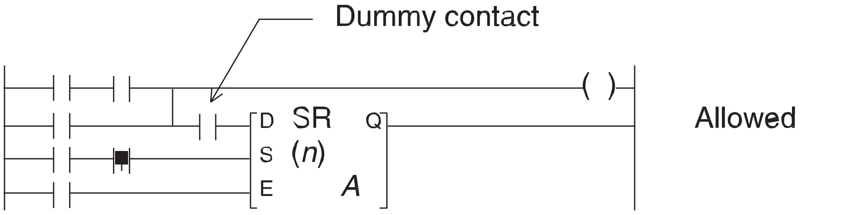
32 devices starting with B100 (B100 to B131) is specified as a shift register.
When B010 is OFF, the data of the shift register is reset to 0. (B100 to B131 are reset to OFF). The carry flag (CF = S0) is also reset to OFF.
While B010 is ON, the data of the shift register is shifted 1 bit to the upper address direction when X009 is changed from OFF to ON. At the same time, the state of X008 is stored in the leading bit (B100).
The output (B011) indicates the state of the last bit (B131).

The figure below shows an operation example. (When X009 is changed from OFF to ON).



*Note*
*When the shift input is ON, the shift operation is performed every scan. Use a transitional contact for the shift input to detect the state changing.*
*For the data input and the shift input, direct linking to a connecting point is not allowed. In this case, insert a dummy contact (always ON special device = S04F, etc.) just before the input.*





# Instruction-76: Bi-directional Shift Register

Expression:

```
Data input      ─┤D  DSR  Q├─      Output
Shift input     ─┤S  (n)   │
Enable input    ─┤E        │
Direction input ─┤L       A│
```

Function:

While the enable input (E) is ON, this instruction shifts the data of the bit table, size n starting with A, 1 bit when the shift input (S) is ON. The shift direction is determined by the state of the direction input (L).
When L is OFF, the direction is right (lower address direction).
When L is ON, the direction is left (upper address direction).
The state of the data input (D) is stored in the highest bit if right shift, and stored in the lowest bit A if left shift. The pushed out bit state is stored in the carry flag (CF = S0).
When the enable input (E) is OFF, all bits in the table and the carry flag are reset to OFF.

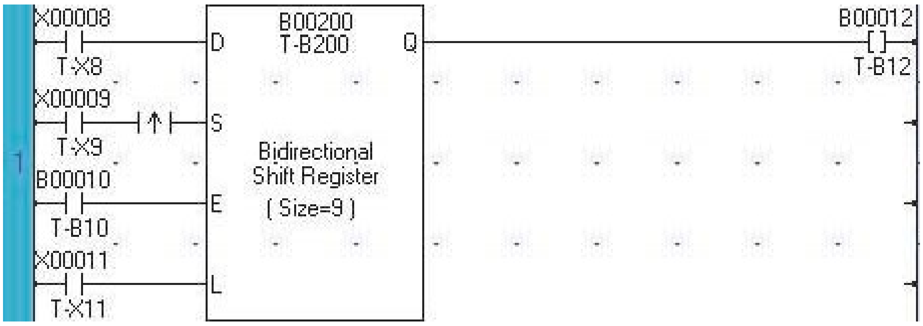
Execution condition:

| Input | Operation | | | Output | CF |
|---|---|---|---|---|---|
| OFF | Resets all bits in the bit table | | | OFF | Reset |
| ON | S = ON | L = ON | Shift left execution | Highest bit state | Set or reset |
| | | L = OFF | Shift right execution | Lowest bit state | Set or reset |
| | S = OFF | No execution | | Highest bit state | --- |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Leading Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |
| *n* | Device Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |

Example:



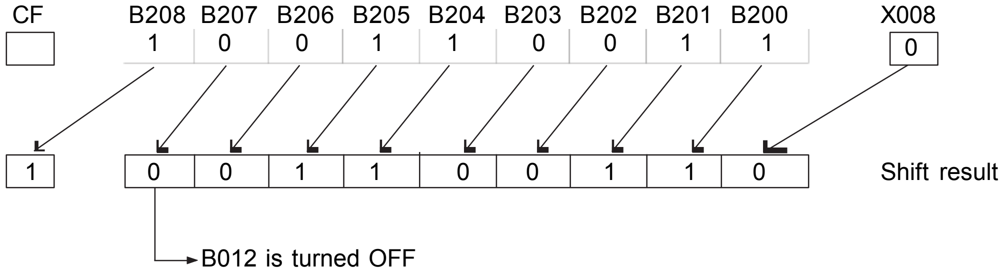
9 devices starting with B200 (B200 to B208) is specified as a shift register.
When B010 is OFF, the data of the shift register is reset to 0. (B200 to B208 are reset to OFF)
The carry flag (CF = S0) is also reset to OFF.

While B010 is ON the following operation is enabled.

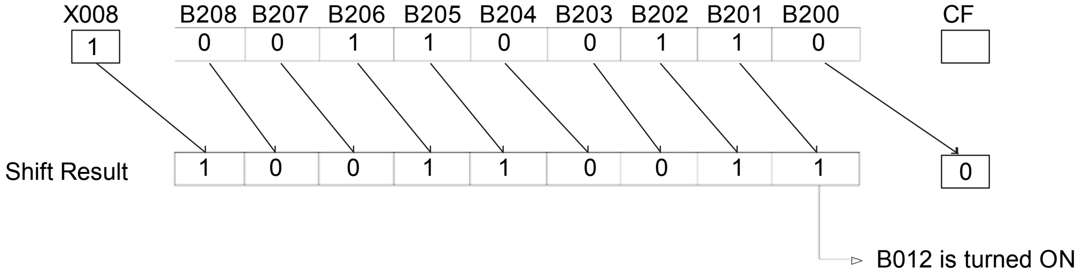
- When X0011 is ON (shift left), the data of the shift register is shifted 1 bit to the upper address direction when X009 is changed from OFF to ON. At the same time, the state of X008 is stored in the leading bit (B200). The output (B012) indicates the state of the highest bit (B208).
- When X0011 is OFF (shift right), the data of the shift register is shifted 1 bit to the lower address direction when X009 is changed from OFF to ON. At the same time, the state of X008 is stored in the highest bit (B208). The output (B012) indicates the state of the lowest bit (B200).

The figure below shows an operation example.

(When X0011 is ON and X009 is changed from OFF to ON).



(When X0011 is OFF and X009 is changed from OFF to ON)



Note:

When the shift input is ON, the shift operation is performed every scan. Use a transitional contact for the shift input to detect the state changing.

For the data input, the shift input and the enable input, direct linking to a connecting point is not allowed. In this case, insert a dummy contact (always ON special device = S04F, etc.) just before the input.

## Instruction-77: 1 bit Rotate Right

Expression:

Input ─[ RTR1 A ]─ Output

Function:

When the input is ON, the data of register A is rotated 1 bit to the right (LSB direction). The pushed out bit state is stored in the left most bit (MSB) and in the carry flag (CF = S0). After the operation, if the right most bit (LSB) is ON, the output is turned ON.

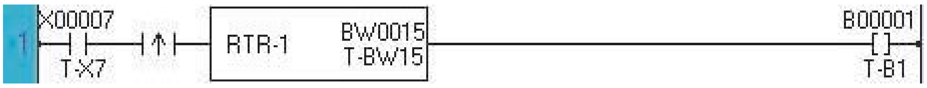
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No Execution | | OFF | --- |
| ON | Execution | When LSB = 1 | ON | Set or reset |
| | | When LSB = 0 | OFF | Set or reset |

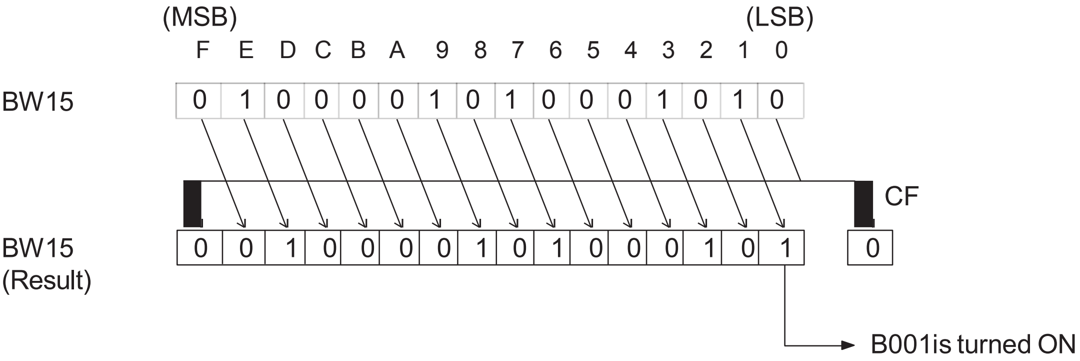
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | ( peration<br>Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X007 is changed from OFF to ON, the data of BW15 is rotated 1 bit to the right.
The figure below shows an operation example.



| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | CF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW15 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| BW15<br>(Result) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

## Instruction-78: 1 bit Rotate Left

Expression:

Input ─[ RTL1 A ]─ Output

Function:

When the input is ON, the data of register A is rotated 1 bit to the left (MSB direction). The pushed out bit state is stored in the right most bit (LSB) and in the carry flag (CF = S0). After the operation, if the left most bit (MSB) is ON, the output is turned ON.

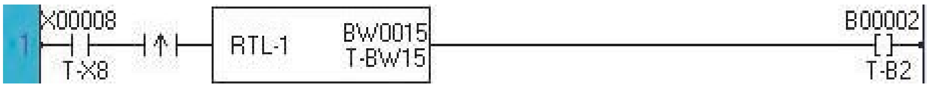
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No Execution | | OFF | --- |
| ON | Execution | When MSB = 1 | ON | Set or reset |
| | | When MSB = 0 | OFF | Set or reset |

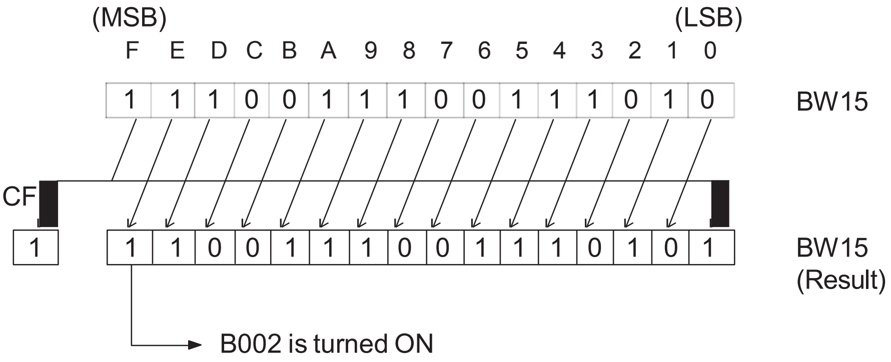
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Operation<br>Data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X008 is changed from OFF to ON, the data of BW15 is rotated 1 bit to the left.
The figure below shows an operation example.



| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | BW15 |
| CF 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | BW15 (Result) |

## Instruction-79: n bit Rotate Right

Expression:

Input ─[ A RTR n ⟶ B ]─ Output

Function:

When the input is ON, the data of register A is rotated n bits to the right (LSB direction), and stored in B. After the operation, if the right most bit (LSB) is ON, the output is turned ON.

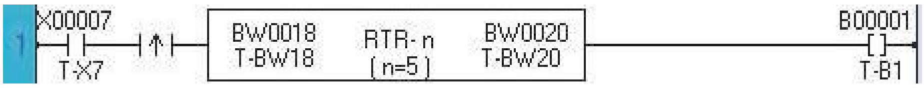
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No Execution | | OFF | --- |
| ON | Execution | When LSB = 1 | ON | Set or reset |
| | | When LSB = 0 | OFF | Set or reset |

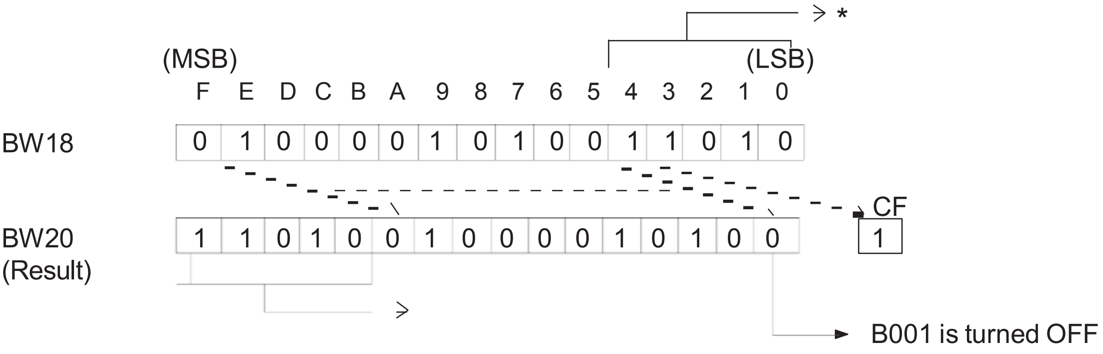
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *n* | Shift bits | | | | | | | | | | | | | | | | | | | | 1 - 16 | |
| *B* | estination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X007 is changed from OFF to ON, the data of BW18 is rotated 5 bits to the right and the result is stored in BW20.
The figure below shows an operation example.



## Instruction-80: n bit Rotate Left

Expression:

Input ─[ A RTL n ──→B ]─ Output

Function:

When the input is ON, the data of register A is rotated n bits to the left (MSB direction), and stored in B. After the operation, if the left most bit (MSB) is ON, the output is turned ON.

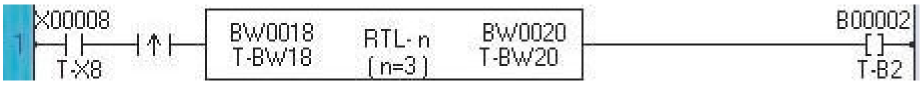
Execution condition:

| Input | Operation | | Output | CF |
|---|---|---|---|---|
| OFF | No Execution | | OFF | --- |
| ON | Execution | When MSB = 1 | ON | Set or reset |
| | | When MSB = 0 | OFF | Set or reset |

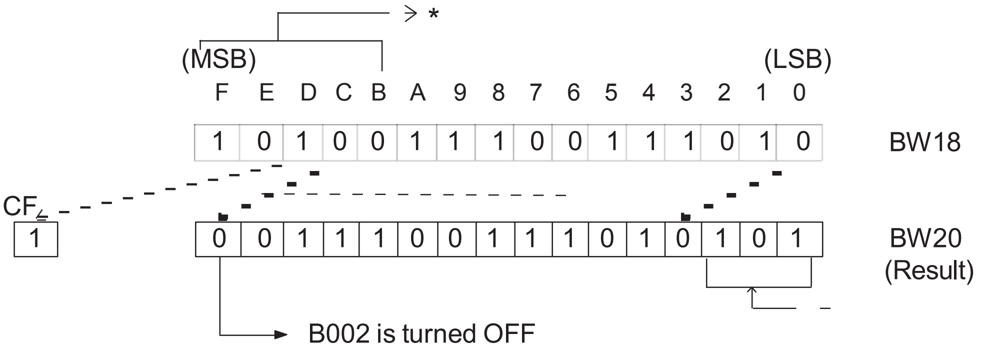
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| *n* | Shift bits | | | | | | | | | | | | | | | | | | | | 1 - 16 | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ |

Example:



When X008 is changed from OFF to ON, the data of BW18 is rotated 3 bits to the left and the result is stored in BW20.
The figure below shows an operation example.



# Instruction-81: Hex to ASCII Conversion

Expression:

Input ─[ A HTOA (n) B ]─ Output

Function:

When the input is ON, the hexadecimal data of n registers starting with A is converted into ASCII characters and stored in B and after. The uppermost digit of source A is stored in lower byte of destination B, and followed in this order. The allowable range of n is 1 to 32.

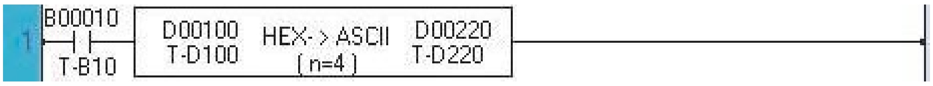
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| *n* | Data Size | | | | | | | | | | | | | | | | | | | | 1 - 32 | |
| *B* | Destinatio n | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When B010 is ON, 4 words data of D0100 to D0103 are converted into ASCII characters, and stored in 8 words registers starting with D0220.

| | F 0 |
|---|---|
| D0100 | H0125 |
| D0101 | H4567 |
| D0102 | H89AB |
| D0103 | HCDEF |

Converted →

| | F 8 | 7 0 |
|---|---|---|
| D0220 | “1” (H31) | “0” (H30) |
| D0221 | “3” (H33) | “2” (H32) |
| D0222 | “5” (H35) | “4” (H34) |
| D0223 | “7” (H37) | “6” (H36) |
| D0224 | “9” (H39) | “8” (H38) |
| D0225 | “B” (H42) | “A” (H41) |
| D0226 | “D” (H44) | “C” (H43) |
| D0227 | “F” (H46) | “E” (H45) |

Note:
If index register (I, J or K) is used for the operand A, only n = 1 is allowed.

# Instruction-82: ASCII to Hex Conversion

Expression:

Input −[ A ATOH (n) B ]− Output

Function:

When the input is ON, the ASCII characters stored in n registers starting with A is converted into hexadecimal data and stored in B and after. The lower byte of source A is stored as uppermost digit of destination B, and followed in this order. The allowable ASCII character in the source table is “0” (H30) to “9” (H39) and “A” (H41) to “F” (H46). The allowable range of n is 1 to 64.

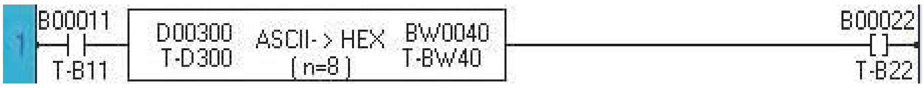
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No Execution | OFF | — |
| ON | Normal Execution | ON | — |
| | Conversion Data Error (no execution) | OFF | Set |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | |
| *n* | Data Size | | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | | |

Example:



When B011 is ON, the ASCII characters stored in 8 words of D0300 to D0307 are converted into hexadecimal data, and stored in 4 words registers starting with BW040.

| | F — 8 | 7 — 0 |
|---|---|---|
| D0300 | “1” (H31) | “0” (H30) |
| D0301 | “3” (H33) | “2” (H32) |
| D0302 | “5” (H35) | “4” (H34) |
| D0303 | “7” (H37) | “6” (36) |
| D0304 | “9” (H39) | “8” (H38) |
| D0305 | “B” (H42) | “A” (H41) |
| D0306 | “D” (H44) | “C” (H43) |
| D0307 | “F” (H46) | “E” (H45) |

| | F — 0 |
|---|---|
| BW040 | H0123 |
| BW041 | H4567 |
| BW042 | H89AB |
| BW043 | HCDEF |

Note:
- If index register (I, J or K) is used for the operand A, only n = 1 is allowed.
- If n is odd number, lower 2 digits of the last converted data will not be fixed, Use even for n.

## Instruction-83: Absolute Value

Expression:

Input ─[ A ABS B ]─ Output

Function:

When the input is ON, this instruction finds the absolute value of operand A, and stores it in B.

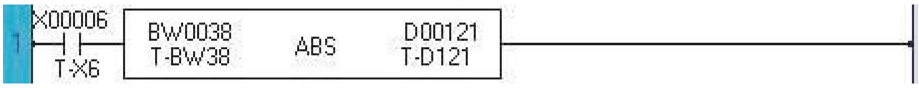
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

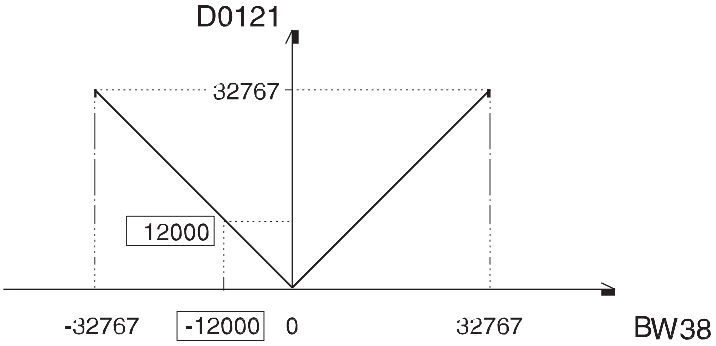
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When X006 is ON, the absolute value of BW38 is stored in D0121.
For example, if BW38 is -12000, the absolute value 12000 is stored in D0121.



Note:
- The data range of A is -32768 to 32767. If the data of A is -32768, 32767 is stored in B.

## Instruction-84: 2’s Compliment

Expression:

Input ⊣[ A NEG B ]⊢ Output

Function:

When the input is ON, this instruction finds the 2’s compliment value of A, and stores it in B.

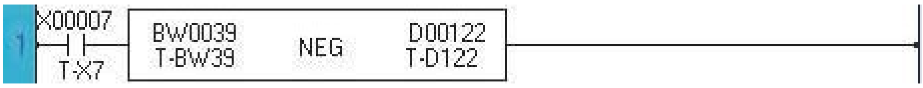
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When X007 is ON, the 2’s complement value (sign inverted data) of BW39 is stored in D0122.
For example, if BW38 is 4660, the 2’s complement value -4660 is stored in D0122.

2’s complement data is calculated as follows.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW39 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | (4660) |
| Bit Inverse | | | | | | | | | | | | | | | | | |
| | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | (-4661) |
| +1 | | | | | | | | | | | | | | | | | |
| D0122 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | (-4660) |

Note:
- The data range of A is -32768 to 32767. If the data of A is -32768, the same data -32768 is stored in B.

## Instruction-85: Double-Word 2’s Compliment

Expression:

Input ─[ A+1.A DNEG B+1.B ]─ Output

Function:

When the input is ON, this instruction finds the 2’s complement value of double-word data A+1×A, and stores it in B+1×B.

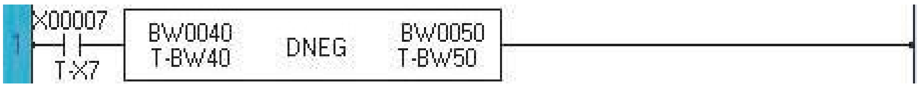
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | √ | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When X007 is ON, the 2’s complement value (sign inverted data) of double-word register BW41×BW40 is stored in double-word register BW0051×BW0050.

For example, if BW41×BW40 is -1234567890, the 2’s complement value 1234567890 is stored in BW0051×BW0050.

Note:
- The data range of A+1× A is -2147483648 to 2147483647. If the data of A+1× A is -2147483648, the same data -2147483648 is stored in B+1× B.

## Instruction-86: 7 Segment Decode

Expression:

Input ─[ A 7SEG B ]─ Output

Function:

When the input is ON, this instruction converts the lower 4 bits data of A into the 7 segment code, and stores it in B. The 7 segment code is normally used for a numeric display LED.

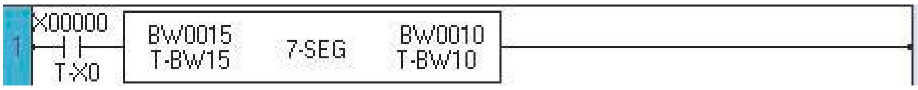
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| *B* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



1 X00000 T-X0 BW0015 T-BW15 7-SEG BW0010 T-BW10

When X000 is ON, the lower 4 bits data of BW15 is converted into the 7 segment code, and the result is stored in lower 8 bits of BW10. 0 is stored in upper 8 bits of BW10.
For example, if BW15 is H0009, the corresponding 7 segment code H006F is stored in BW10.

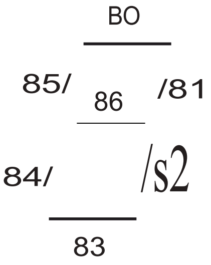
| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | (H0009) |
| BW10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | (H006F) |

Upper 12 bits are ignored

7 Segment decode

0 is stored in upper 8 bits.

*The 7 segment code conversion table is shown on the next page*.

| Operand *A* (lower 4 bits} | | 7 segment LED | OJerand *B*!ower 8 bits} | | | | | | | | Display |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hex | Binary | composition | 87 | 86 | 85 | 84 | B3 | 82 | 81 | BO | |
| 0 | 0000 |  | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | n<br>u |
| 1 | 0001 | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ¦ |
| 2 | 0010 | | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | :ı<br>L |
| 3 | 0011 | | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 3 |
| 4 | 0100 | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | Ч |
| 5 | 0101 | | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | c<br>...J |
| 6 | 0110 | | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 5 |
| 7 | 0111 | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | n |
| 8 | 1000 | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 9 | 1001 | | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | g |
| A | 1010 | | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | R |
| B | 1011 | | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | b |
| C | 1100 | | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | r<br>L |
| D | 1101 | | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | ú |
| E | 1110 | | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | E |
| F | 1111 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | F |

## Instruction-87: ASCII Conversion

Expression:

| |
|---|
| Input ⊣[ A ASC B ]— Output |

Function:

| |
|---|
| When the input is ON, this instruction converts the alphanumeric characters into the ASCII codes, and stores them in the register table starting with B. (16 characters maximum). |

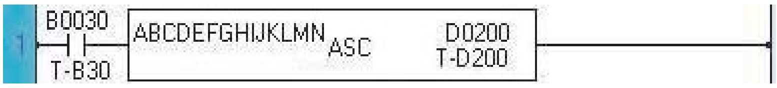
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Characters | | | | | | | | | | | | | | | | | | | | √ | |
| *B* | Start of Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When B030 is ON, the characters 'ABCDEFGHIJKLMN' is converted into the ASCII codes, and the result is stored in 8 registers starting with lower 8 bits (byte) of D0200 (D0200 to D0207).

| | High | Low | |
|---|---|---|---|
| | F 8 | 7 0 | |
| D0200 | H42 (B) | H41 (A) | |
| D0201 | H44 (D) | H43 (C) | |
| D0202 | H46 (F) | H45 (E) | |
| D0203 | H48 (H) | H47 (G) | |
| D0204 | H4A (J) | H49 (I) | |
| D0205 | H4C (L) | H4B (K) | |
| D0206 | H4E (N) | H4D (M) | |
| D0207 | | | ← Previous data is remained |

Note:
Only the number of bytes converted are stored. The rest are not changed. In the above example, 14 characters are converted into 14 bytes of ASCII code, and these ASCII codes are stored in 7 registers (D0200 to D0206). The data of D0207 remains unchanged.

# Instruction-88: Binary Conversion

Expression:

Input ─[ A BIN B ]─ Output

Function:

When the input is ON, this instruction converts the 4 digits of BCD data of A into binary, and stores in B. If any digit of A contains non-BCD code (other than H0 through H9), the conversion is not executed and the instruction error flag (ERF = S0034) is set to ON.

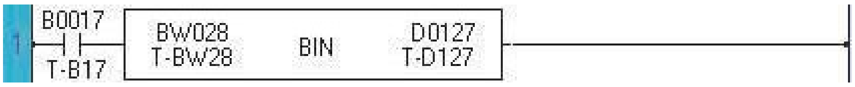
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No Execution | OFF | — |
| ON | Normal Execution | ON | — |
| | BCD data error | OFF | Set |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source (BCD) | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | H000-H9999 | |
| *B* | Destinatio n (Binary) | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When B017 is ON, the BCD data of BW28 is converted into binary data, and the result is stored in D0127.

For example, if BW28 is H1234, the binary data 1234 is stored in D0127.

| BW28 | BCD to Binary | D0127 |
|---|---|---|
| H1234 | ──┼─▶ | 1234 |

Note:
If any digit of operand A contains non-BCD data, e.g. H13A6, the conversion is not executed and the instruction error flag (ERF = S0034) is set to ON.

## Instruction-89: BCD Conversion

Expression:

Input ┤[ A BCD B ]├ Output

Function:

When the input is ON, this instruction converts the binary data of A into BCD, and stores in B. If the data of A is not in the range of 0 to 9999, the conversion is not executed and the instruction error flag (ERF = S0034) is set to ON.

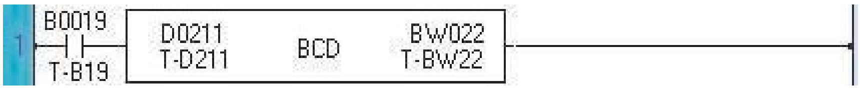
Execution condition:

| Input | Operation | Output | ERF |
|---|---|---|---|
| OFF | No Execution | OFF | — |
| ON | Normal Execution | ON | — |
|  | Binary data error | OFF | Set |

Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| *A* | Source<br>(Binary) |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |  | 0 - 9999 |  |
| *B* | Destinatioh<br>(BCD) |  |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |  |  |  |

Example:



When B019 is ON, the data of D0211 is converted into 4-digit BCD, and the result is stored in BW22.

For example, if D0211 is 5432, the BCD data H5432 is stored in BW22.

D0211 Binary to BCD BW22

5432 ────┼──► H5432

Note:
If the data of A is smaller than 0 or greater than 9999, the conversion is not executed and the instruction error flag (ERF = S0034) is set to ON.

## Instruction-90: Integer to Float

Expression:

Input ⊣[ A INT -> FLOAT B ]— Output

Function:

This instruction converts integer of double word type data into floating point data.

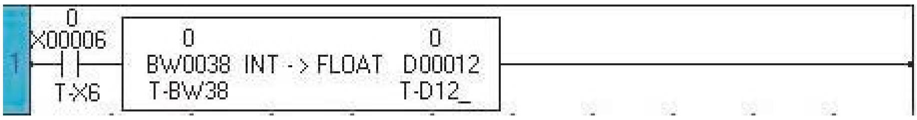
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | | | √ | | | | √ | | | | | √ | | |
| *B* | Destination | | | | | | | | | | √ | | | | √ | | | | | √ | | |

Example:



When X006 is ON, the integer value of BW38, BW39 will be converted into float fomat and will be stored in D0012.
For example, if BW38, BW39 is 12 then it will become 12.0.

## Instruction-91: Float to Integer

Expression:

Input –[ A FLOAT -> INT B ]– Output

Function:

This instruction converts floating point data into double word integer.

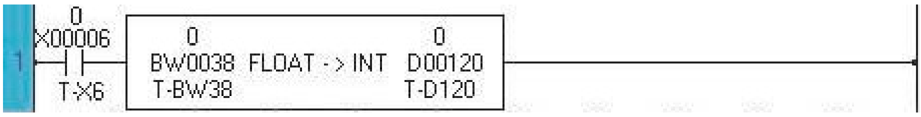
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution | ON |

Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| *A* | Source |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ |  |  |
| *B* | Destination |  |  |  |  |  |  |  |  |  | √ |  |  |  | √ |  |  |  |  | √ |  |  |

Example:



When X006 is ON, the floating point value of BW38 will be converted into integer fomat and will be stored in D00120, D00121.
For example, if BW38 is 12.7 then it will become 13.
If the value is 12.3, then it becomes 12.

## Instruction-92: ON Timer

Expression:

Input ─[ A TON B ]─ Output

Function:

When the input is changed from OFF to ON, timer updating for the timer register B is started. The elapsed time is stored in B. When the specified time by A has elapsed after the input came ON, the output and the timer device corresponding to B are turned ON. (Timer updating is stopped)
When the input is changed from ON to OFF, B is cleared to 0, and the output and the timer device are turned OFF.
The available data range for operand A is 0 to 32767.

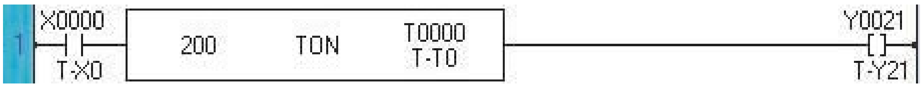
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No operation (timer is not updating) | OFF |
| ON | Elapsed time < preset time (timer is updating) | ON |
| | Elapsed time ≥ preset time (timer is not updating) | OFF |

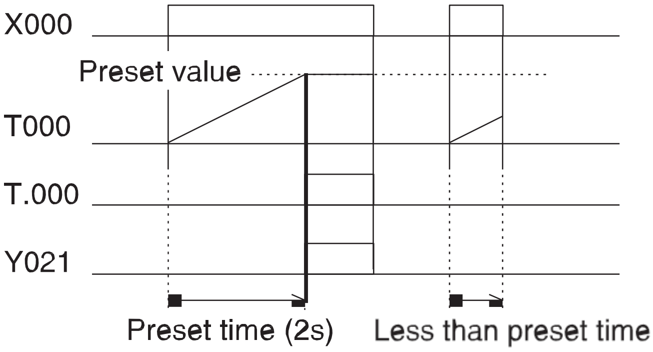
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Preset Time | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | 0 - 32767 | |
| *B* | Elapsed time | | | | | | | | | | | | √ | | | | | | | | | | |

Example:



Y021 (and the timer device T.000) is turned ON 2 seconds after X000 came ON.



Note
Time is set in 10 ms units for;
RMP10: T000 to T060 (0 to 327.67 s)

Time is set in 100 ms units for;
RMP10: T061 to T190 (0 to 3276.7 s)

Time is set in 1 s units for;
RMP10: T191 to T255 (0 to 32767 s)

Multiple timer instructions (TON, TOF or TSS) with the same timer register are not allowed.

Note:
Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

## Instruction-93: OFF Timer

Expression:

Input ─[ A TOFF B ]─ Output

Function:

When the input is changed from OFF to ON, the output and the timer device corresponding to the timer register B are set to ON. When the input is changed from ON to OFF, timer updating for B is started. The elapsed time is stored in B. When the specified time by A has elapsed after the input came OFF, the output and the timer device are turned OFF. (Timer updating is stopped)
The available data range for operand A is 0 to 32767.

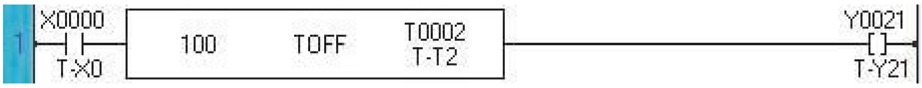
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Elapsed time < preset time (timer is updating) | ON |
| | Elapsed time ≥ preset time (timer is not updating) | OFF |
| ON | No operation (timer is not updating) | ON |

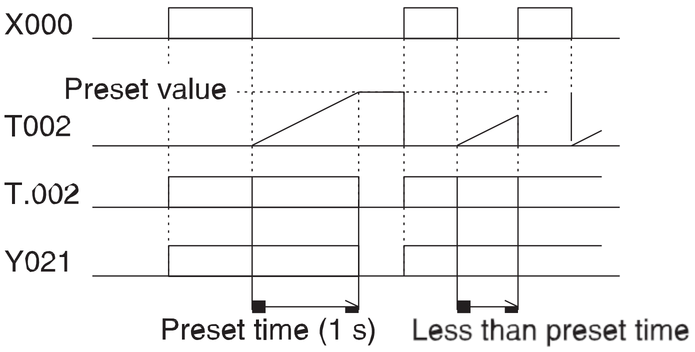
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Preset Time | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | 0 - 32767 | |
| *B* | Elapsed time | | | | | | | | | | | | √ | | | | | | | | | |

Example:



Y021 (and the timer device T.002) is turned OFF 1 second after X000 came ON.



Note
Time is set in 10 ms units for;
RMP10: T000 to T060 (0 to 327.67 s)

Time is set in 100 ms units for;
RMP10: T061 to T190 (0 to 3276.7 s)

Time is set in 1 s units for;
RMP10: T191 to T255 (0 to 32767 s)

Multiple timer instructions (TON, TOF or TSS) with the same timer register are not allowed.

Note:
Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

## Instruction-94: Single Shot Timer

Expression:

Input ─[ A TSS B ]─ Output

Function:

When the input is changed from OFF to ON, the output and the timer device corresponding to the timer register B are set to ON, and timer updating for B is started. The elapsed time is stored in B.
When the specified time by A has elapsed after the input came ON, the output and the timer device are turned OFF. (Timer updating is stopped)
The available data range for operand A is 0 to 32767.

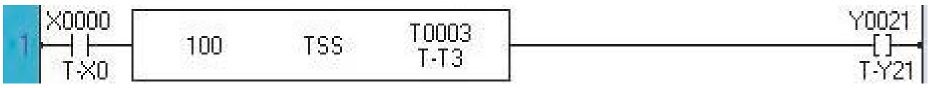
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Elapsed time < preset time (timer is updating) | ON |
| | Elapsed time ≥ preset time (timer is not updating) | OFF |
| ON | Elapsed time < preset time (timer is updating) | ON |
| | Elapsed time ≥ preset time (timer is not updating) | OFF |

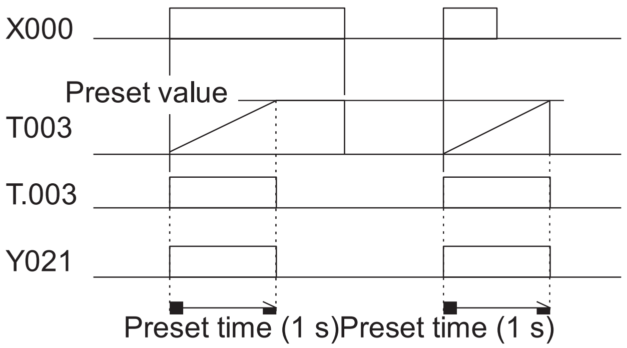
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Preset Time | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | 0 - 32767 | |
| B | Elapsed time | | | | | | | | | | | | √ | | | | | | | | | |

Example:



Y021 (and the timer device T.003) is turned OFF 1 second after X000 came ON.



Note
Time is set in 10 ms units for;
RMP10: T000 to T060 (0 to 327.67 s)

Time is set in 100 ms units for;
RMP10: T061 to T190 (0 to 3276.7 s)

Time is set in 1 s units for;
RMP10: T191 to T255 (0 to 32767 s)

Multiple timer instructions (TON, TOF or TSS) with the same timer register are not allowed.

Note:
Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

# Instruction-95: Counter

Expression:

```
Count Input   ─┌c  CNT   Q┐─   Output

Enable Input  ─└E   A   B ┘
```

Function:

While the enable input is ON, this instruction counts the number of the count input changes from OFF to ON. The count value is stored in the counter register B. When the count value reaches the set value A, the output and the counter device corresponding to B are turned ON. When the enable input comes OFF, B is cleared to 0 and the output and the counter device are turned OFF.
The available data range for operand A is 0 to 65535.

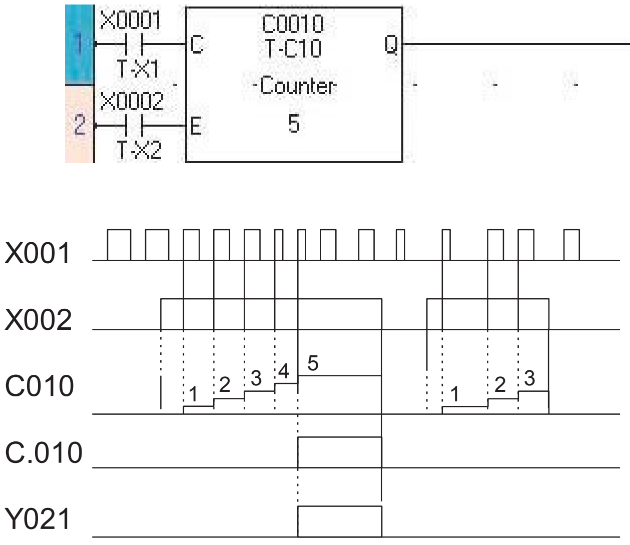
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No operation ( B is cleared to 0) | OFF |
| ON | Count value ( B) < set value ( A) | OFF |
| | Count value ( B) ≥ set value ( A) | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Set Value | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | 0 - 65535 | |
| *B* | Count Value | | | | | | | | | | | | | √ | | | | | | | | |

Example:



Note:
No transitional contact is required for the count input. The count input rising edge is detected by this instruction.

For the count input, direct linking to a connecting point is not allowed. In this case, insert a dummy contact (always ON = S04F, etc.) just before the input.
Refer to Note of Shift register FUN 074.

Multiple counter instructions (CNT) with the same counter register are not allowed.

## Instruction-96: Up / Down Counter

Expression:

```
Direction Input ┤U   A   Q ├ Output
Count Input      C
Enable Input    ┤E         ┘
```

Function:

While the enable input is ON, this instruction counts the number of the count input changes from OFF to ON. The count direction (up count or down count) is selected by the state of the direction input. The count value is stored in the counter register A. The count value range is 0 to 65535.

Up count when the direction input is ON

Down count when the direction input is OFF

When the enable input is OFF, the counter register A is cleared to 0.

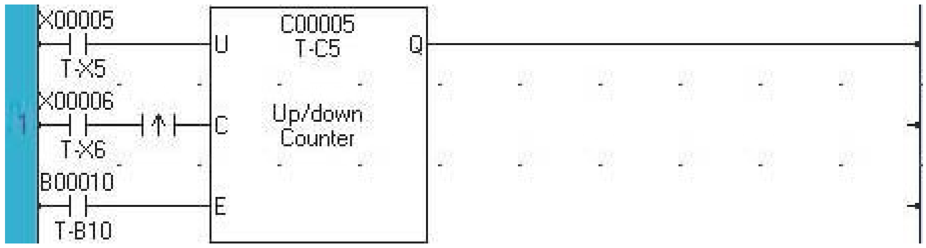
Execution condition:

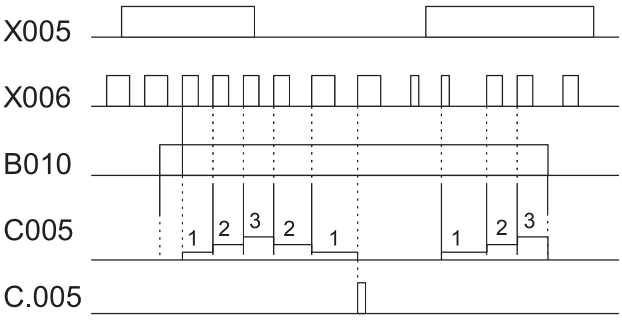
| Input | Operation | Output |
|---|---|---|
| OFF | No operation ( A is cleared to 0) | OFF |
| ON | Count value is not limit value (0 or 65535) | OFF |
| | Count value is limit value and count input is ON | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Count Value | | | | | | | | | | | | | √ | | | | | | | | |

Example:





Note:
The transitional contact is required for the count input. Otherwise, counting is executed every scan during X005 is ON in this example.

For the direction input and the count input, direct linking to a connecting point is not allowed. Refer to Note of Shift register Function.

## Instruction-97: Subroutine Call

Expression:

Input –[CALL N. n]– Output

Function:

When the input is ON, this instruction calls the subroutine number n..

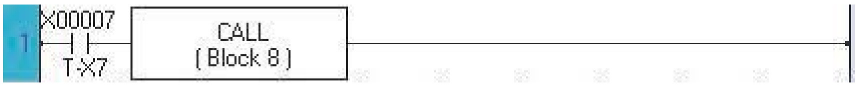
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

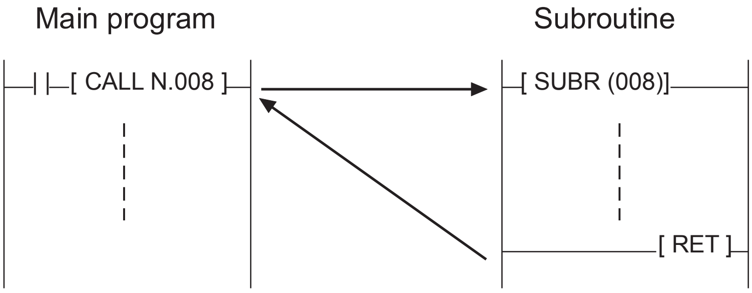
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *n* | Subroutine Number | | | | | | | | | | | | | | | | | | | | √(Note) | |

Example:



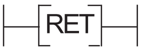
When X007 is ON, the subroutine number 8 is called. When the program execution is returned from the subroutine, the output is turned ON.



Note:
The possible subroutine number is 0 to 255.
Refer to the SUBR instruction.
The CALL instruction can be used in an interrupt program. However, it is not allowed that the same subroutine is called from an interrupt program and from main program.

## Instruction-98: Subroutine Return

Expression:



Function:

This instruction indicates the end of a subroutine. When program execution is reached this instruction, it is returned to the original CALL instruction.

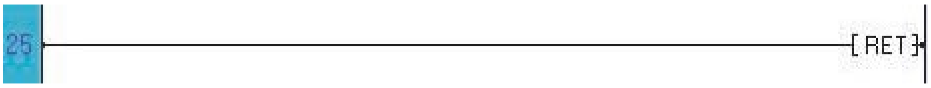
Execution condition:

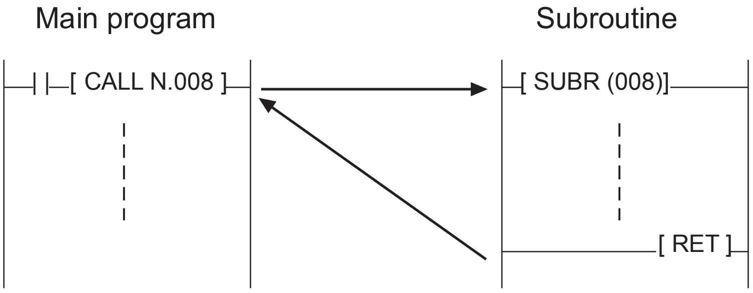
| Input | Operation | Output |
|---|---|---|
| – | Execution | – |

Operand:

No operand is required.

Example:





Note:
Refer to the SUBR instruction.
The RET instruction can be programmed only in the program type ‘Subroutine’.
The RET instruction must be connected directly to the left power rail.

## Instruction-99: FOR (For next loop)

Expression:

Input —[FOR n]— Output

Function:

When the input is ON, the program segment between FOR and NEXT is executed n times repeatedly in a scan.
When the input is OFF, the repetition is not performed. (the segment is executed once).

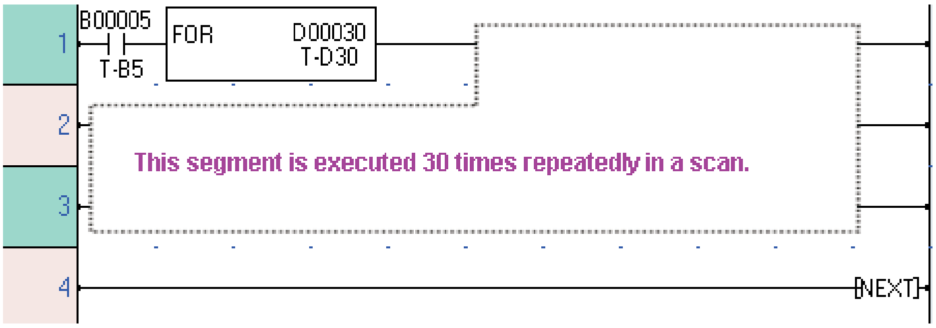
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Repetition | OFF |
| ON | Repetition | ON |

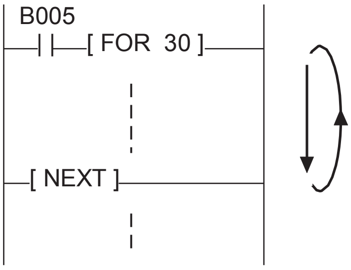
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *n* | epetation Times | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | 1-32767 | |

Example:



When B005 is ON, the program segment between FOR and NEXT is executed 30 times in a scan.



Executed 30 times in a scan when B005 is ON.

# Instruction-100: NEXT (FOR-NEXT loop)

Expression:

Input —[NEXT]— Output

Function:

This instruction configures a FOR-NEXT loop.
If the input is OFF, The repetition is forcibly broken. and the program execution is moved to the next instruction.

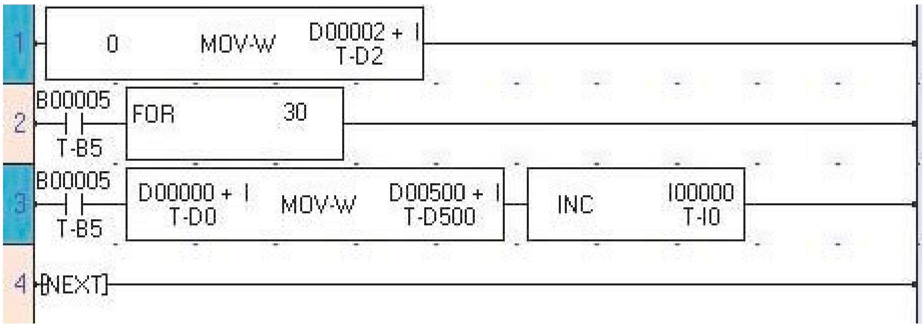
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Forcibly breaks the repetition | OFF |
| ON | Repetition | ON |

Operand:

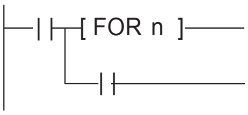
No operand is required.

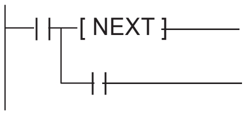
Example:



When B005 is ON, the program segment between FOR and NEXT is executed 30 times in a scan.
In the above example, the rung 3 is executed 30 times. As a result, the data of D0000 to D0029 are transferred to D0500 to D0529. (Block transfer)

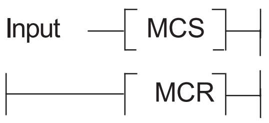
Note
The FOR instruction must be used with a corresponding NEXT instruction one by one.
Nesting of the FOR-NEXT loop is not allowed. That is, the FOR instruction cannot be used in a FOR-NEXT loop.
The FOR and NEXT instructions cannot be programmed on the same rung.
The following connection is not allowed.





# Instruction-101: Master Control Set / Reset

Expression:



Function:

When the MCS input is ON, ordinary operation is performed. When the MCS input is OFF, the state of left power rail between MCS and MCR is turned OFF.

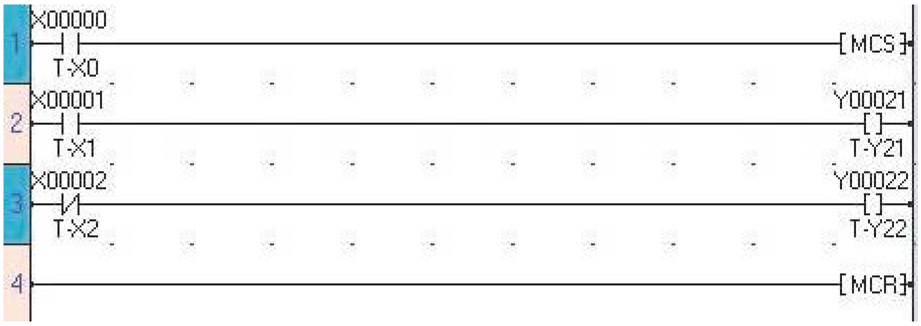
Execution condition:

| MCS Input | Operation | Output |
|---|---|---|
| OFF | Sets OFF the left power rail until MCR | — |
| ON | Ordinary operation | — |

Operand:

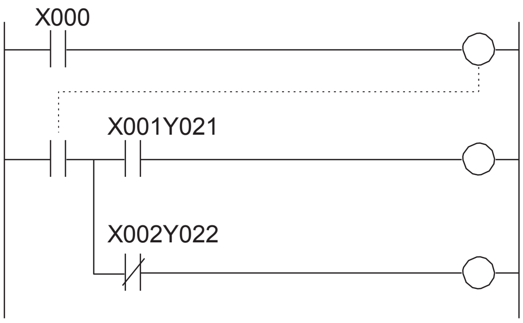
No operand is required.

Example:



When X000 is OFF, Y021 and Y022 are turned OFF regardless of the states of X001 and X002.

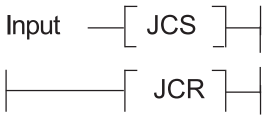
Equivalent circuit



Note
MCS and MCR must be used as a pair.
Nesting is not allowed.

## Instruction-102: Jump Control Set / Reset

Expression:



Function:

When the JCS input is ON, instructions between JCS and JCR are skipped (not executed). When the JCS input is OFF, ordinary operation is performed.

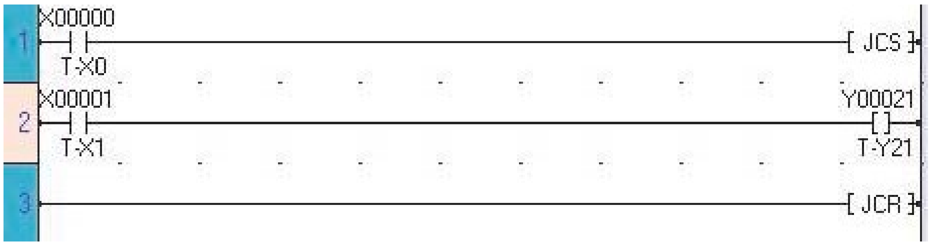
Execution condition:

| JCS Input | Operation | Output |
|---|---|---|
| OFF | Ordinary operation | — |
| ON | Skip until JCR | — |

Operand:

No operand is required.

Example:



When X000 is ON, the rung 2 circuit is skipped, therefore Y021 is not changed its state regardless of the X001 state. When X000 is OFF, Y021 is controlled by the X001 state.

Note
JCS and JCR must be used as a pair.
Nesting is not allowed.

## Instruction-103: Enable Interrupt

Input ├─[ EI ]─┤ Output

Function:

When the input is ON, this instruction enables the execution of user designated interrupt operation, i.e. timer interrupt program and I/O interrupt programs.

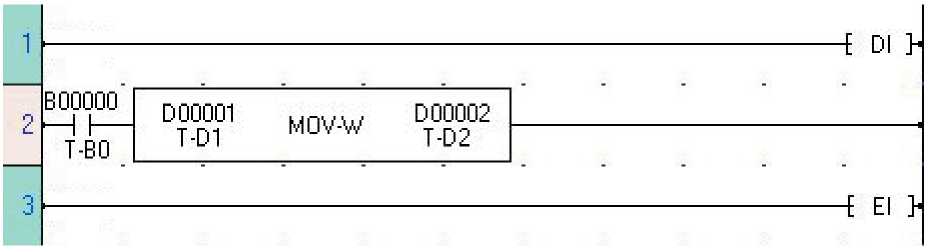
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

No operand is required.

Example:



In the above example, the DI instruction disables the interrupt. Then the EI instruction enables the interrupt again. As a result, the rung 2 instructions can be executed without interruption between each instructions.

Note

· Refer to the DI instruction.

· If an interrupt factor is occurred during the interrupt disabled state, the interrupt is kept waiting and it will be executed just after the EI instruction is executed.

· The EI instruction can be used only in the main program.

# Instruction-104: Disable Interrupt

Input ├[ DI ]┤ Output

Function:

When the input is ON, this instruction disables the execution of user designated interrupt operation, i.e. timer interrupt program and I/O interrupt programs.

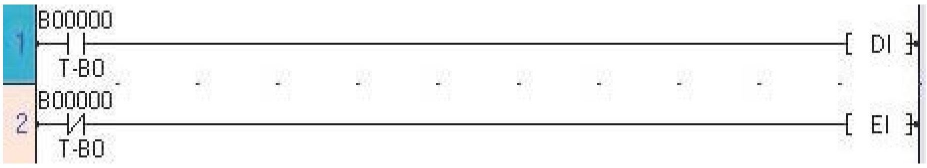
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

No operand is required.

Example:



In the above example, the interrupt is disabled when B000 is ON, and it is enabled when B000 is OFF.

Note
· Refer to the EI instruction.
· If an interrupt factor is occurred during the interrupt disabled state, the interrupt is kept waiting and it will be executed just after the EI instruction is executed.
· The DI instruction can be used only in the main program.

## Instruction-105: Watchdog Timer Reset

Expression:

Input ─┤[ WDT n ]├─ Output

Function:

When the input is ON, this instruction extend the scan time over detection time by 200 ms. This instruction can be used to extend the detection time by multiple of 1ms.
if n = 1 => 201ms; if n = 100 => 300ms

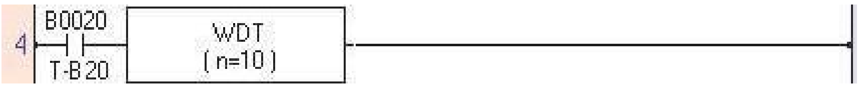
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *n* | Extend time | | | | | | | | | | | | | | | | | | | | 1-100 | |

Example:



When B020 is ON, the scan time detection time is extended by 10x1 ms.

Note
· The operand n specifies the extended time.
· The normal scan time detection is 200 ms
· If the ladder scan time (SW0046) exceeds the detection time, the following error bits are set:
M00018 (MW01_2): Program error
M00033 (MW02_1): Ladder scan time error
· The unit does not restart

## Instruction-106: Step Sequence Initialize

Expression:

Input ├[STIZ (n) A]┤ Output

Function:

When the input is ON, n devices starting with A are reset to OFF, and A is set to ON.
This instruction is used to initialize a series of step sequence. The step sequence is useful to describe a sequential operation.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution at the rising edge of the input | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *n* | Size of step Sequence | | | | | | | | | | | | | | | | | | | | 1-64 | |
| A | Start Device | | | √ | | | | | | | | | | | | | | | | | | |

Example:

When B020 is changed from OFF to ON, B400 is set to ON and subsequent 9 devices (B401 to B409) are reset to OFF.
This instruction initializes a series of step sequence, 10 devices starting with B400.

| B409 | B408 | B407 | B406 | B405 | B404 | B403 | B402 | B401 | B400 |
|---|---|---|---|---|---|---|---|---|---|
| OFF | OFF | OFF | OFF | OFF | OFF | OFF | OFF | OFF | ON |

10 devices staring with B400

Note
· The STIZ instruction is used together with STIN and STOT instructions to configure the step sequence.
· The STIZ instruction is executed only when the input is changed from OFF to ON.

# Instruction-107: Step Sequence Input

Expression:

Input ⊢[ STIN    A ]⊣ Output

Function:

When the input is ON and the device A is ON, the output is set to ON.

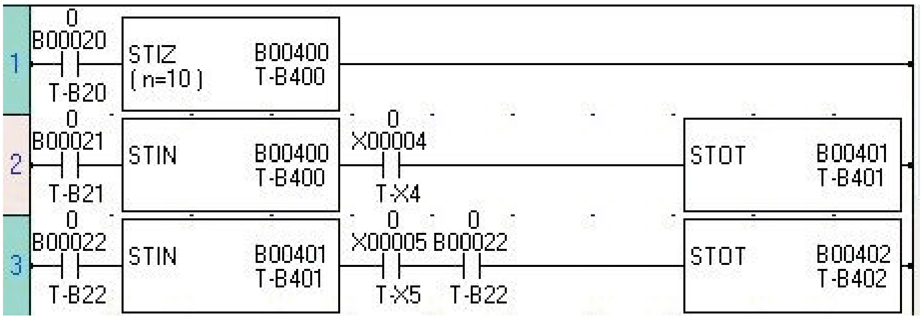
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | When A is ON | ON |
|  | When A is OFF | OFF |

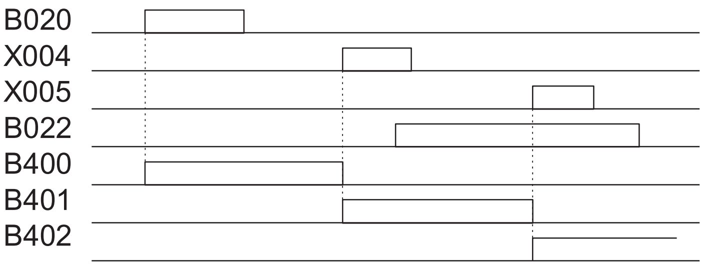
Operand:

|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| *A* | Step Device |  |  | √ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Example:



The following sequential operation is performed.
When B020 is changed from OFF to ON, B400 is set to ON and subsequent 9 devices (B401 to B409) are reset to OFF.
When X004 comes ON, B400 is reset to OFF and B401 is set to ON.
When both X005 and B022 are ON, B401 is reset to OFF and B402 is set to ON.



## Instruction-108: Step Sequence Output

Expression:

Input ├[ STOT A ]┤ Output

Function:

When the input is ON, the device A is set to ON and the devices of STIN instructions on the same rung are reset to OFF.

Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | — |
| ON | Execution | — |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Step Device | | | √ | | | | | | | | | | | | | | | | | | |

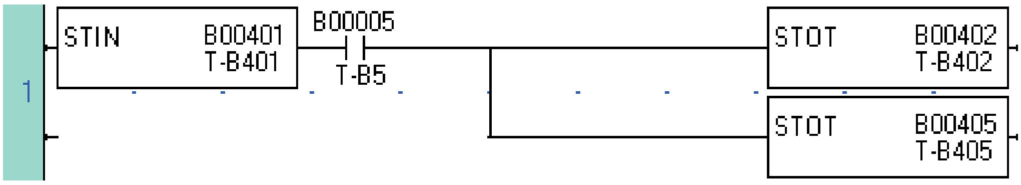
Example:

See example on STIN instruction.

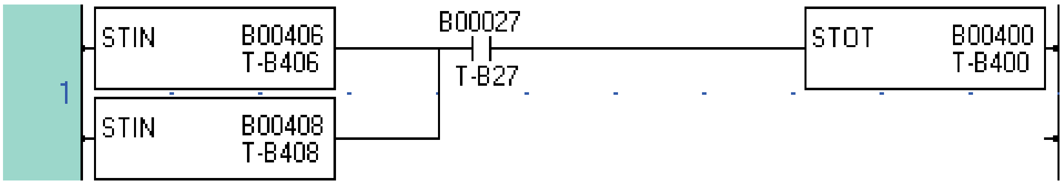
Note:

· The STIZ, STIN and STOT instructions are used together to configure the step sequence.

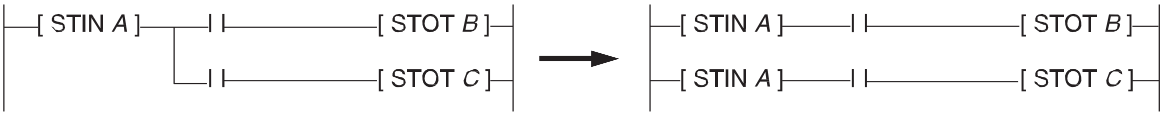
· Two or more STOT instructions can be placed on one rung to perform simultaneous sequences.



· Two or more STIN instructions can be placed on one rung in parallel or in series to perform loop or convergence of sequences. (Max. 11 STIN instructions on one rung)



· To perform the conditional branch (sequence selection), separate the rungs as follows.



Not allowed

Available

## Instruction-109: Moving Average

Expression:

Input ─[ A MAVE (n) B ⟶ C ]─ Output

Function:

When the input is ON, this instruction calculates the average value of the latest n scan's register A data, and stores it in C. The allowable range of n is 1 to 64.
This instruction is useful for filtering the analog input signal.
The latest n scan's data of A are stored in n registers starting with B, and C+1 are used as pointer.

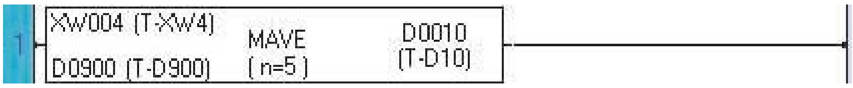
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Input Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| n | Data Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| B | Start of table | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| C | Output data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | | √ | | | |

Example:



The latest 5 scan's data of XW04 is stored in D0900 to D0904 (5 registers), and the average value of them is calculated and stored in D0010.
D0011 is used as internal work data.

| | XW04 | D0010 | |
|---|---|---|---|
| 1st scan | 1000 | 200 | = (1000) / 5 |
| 2nd scan | 1005 | 401 | = (1000 + 1005) / 5 |
| 3rd scan | 1009 | 603 | = (1000 + 1005 + 1009) / 5 |
| 4th scan | 1012 | 805 | = (1000 + 1005 + 1009 + 1012) / 5 |
| 5th scan | 1007 | 1006 | = (1000 + 1005 + 1009 + 1012 + 1007) / 5 |
| 6th scan | 1004 | 1007 | = (1005 + 1009 + 1012 + 1007 + 1004) / 5 |
| 7th scan | 998 | 1006 | = (1009 + 1012 + 1007 + 1004 + 998) / 5 |
| 8th scan | 994 | 1003 | = (1012 + 1007 + 1004 + 998 + 994) / 5 |

⋮

## Instruction-110: Digital Filter

Expression:

Input ─[ A DFL B →C]─ Output

Function:

When the input is ON, this instruction calculates the following formula to perform digital filtering for input data A by filter constant by B, and stores the result in C.
Yn = (1 - FL) * Xn + FL * Yn-1
Here; Xn is input data specified by A
FL is filter constant, 1/10000 of data specified by B (data range: 0 to 9999)
Yn is output data to be stored in C
Yn-1 is output data at last scan
This instruction is useful for filtering the analog input signal. C+1 is used for internal work data.

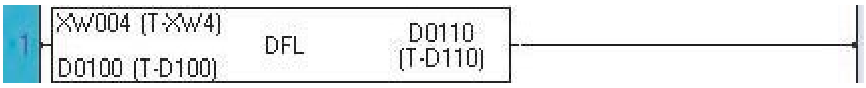
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution (FL is limited within the range of 0 to 9999) | ON |

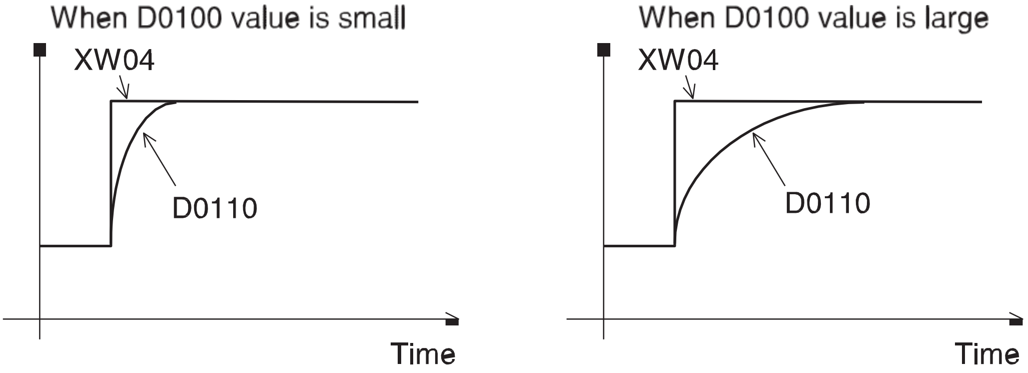
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Input Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| B | Filter Constant | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| C | Output data | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



The filtered data of XW04 is stored in D0110. (D0111 is used for internal work data).



## Instruction-111: Pre-derivative Real PID1

Expression:

Input ─[ A PID1 B →C]─ Output

Function:

Using the parameters stored in the 7 registers starting with the register specified by the operand B and previous values stored in the 4 registers following the register specified by the operand C, the PID calculation is executed as described below on the present value P and the set value S stored in the 2 registers starting with the register specified by the operand A. The increments of manipulation value M is calculated and stored in the register specified by the operand C.

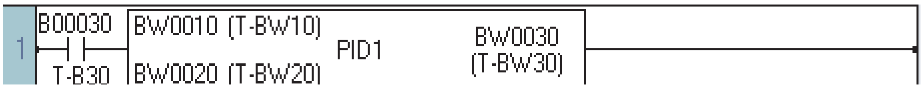
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Execution | OFF |
| ON | Execution KIH and KIL ! = 0 | ON |
| ON | Execution KIH and KIL = 0<br>(only proportional controller ON) | ON |

Operand:

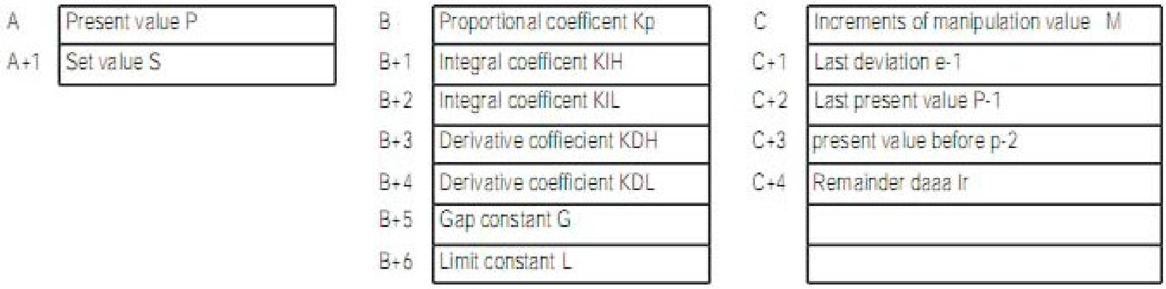
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Top of Input Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| B | Top of Parameter | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| C | Top of output data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



If the NO-contact B0030 in ON, then, using the contents of the 7 registers starting with the register specified by the operand B [i.e. the contents of BW20 (Kp = 1), of RW21 (Kih = 4), of BW22 (KIL = 10), of BW23 (KDH = 20), of BW24 (KDL = 5), of BW25 (G = 0) and BW26 (L = 100)] - plus the contents of the 4 registers (BW31 to BW34) following the register specified by the operand C (BW30) [i.e. the previous deviation e-1 (78), the previous input value P-1 (22), the input before the previous input P-2 (20), and the remainder data Ir (0)] - the PID calculation is executed on the input data consisting of the contents (P = 25) of the register BW10 and the contents (S = 100) of the register BW11 specified by the operand A. The result (M = 180, e-1 = 75, P-1 = 25, P-2 = 22, Ir = 2) are stored in the 5 registers (BW30 - BW34) starting with the register specified by the operand C.
After the calculation, the execution output is switched ON.

If the NO-contact B0030 is OFF, the calculation is not executed and the output is switched OFF. However, M and Ir are set to0, e-1 is set to the value of e (=S-P), ans P-1 and P-2 are set to the value of P.



| | | | | | |
|---|---|---|---|---|---|
| A | Present value P | B | Proportional coefficent Kp | C | Increments of manipulation value M |
| A+1 | Set value S | B+1 | Integral coefficent KIH | C+1 | Last deviation e-1 |
| | | B+2 | Integral coefficent KIL | C+2 | Last present value P-1 |
| | | B+3 | Derivative coffiecient KDH | C+3 | present value before p-2 |
| | | B+4 | Derivative coefficient KDL | C+4 | Remainder daaa Ir |
| | | B+5 | Gap constant G | | |
| | | B+6 | Limit constant L | | |

PID Calculation:

$$M = Kp . [ ( e-e-1 ) + INT ( \frac{|KIL| . e + Ir}{|KIH|} ) + INT [ \frac{|KDH|}{|KDL|} . (2P-1-P-P-2) ]$$

Here, e is the deviation, and is calculated by applying limit and gap for the value of (S-P).
(See diagram below:)

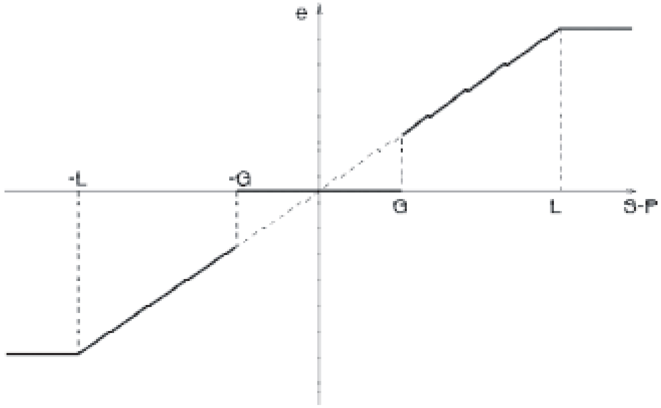
Ir shows the remainder of the following:

$$INT ( \frac{|KIL| . e + Ir}{|KIH|} )$$ (Initial value of Ir is 0)

INT 9a) is the function which produces the quotient from the devision a.

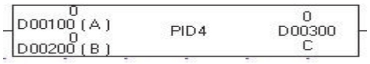
Example: $INT (\frac{50}{3}) = 16$, $INT (\frac{18}{5}) = 3$

* The range of data which can be stored in the register specified by the operand A is from -32768 to 32767.
* When the calculated M>32767, or when M<-32768, the limit value is stored in the register of the operand C, and the execution output is switched ON.
* If KIH = 0, or if KDL = 0, the Integral and derivative calculation is not executed.



## Instruction-112: Pre-derivative Real PID4

Expression:



Function:

Performs PID (Proportional, Integral, Derivative) control which is a fundamental method of feed-back control. The basic idea behind the a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output.

Using the parameters stored in the 6 registers starting with the register specified by the operand B and previous values stored in the 5 registers following the register specified by the operand C, the PID calculation is executed as described below on the present value P and the set value S stored in the 2 registers starting with the register specified by the operand A. The increments of manipulation value M is calculated and stored in the register specified by the operand C.

Algorithm used:

$$MV_n = MV_{n-1} \pm \Delta MV_n$$

$$\Delta MV_n = K_P \bullet (\Delta P_n + \Delta I_n) + K_D(\Delta D_n)$$

*Here,*

$$\Delta P_n = e_n - e_{n-1}$$

$$e_n = SV_n - PV_n \qquad \textit{if reverse action}$$

$$e_n = PV_n - SV_n \qquad \textit{if forward action}$$

$$\Delta I_n = \left(\frac{e_n}{T_I}\right) \text{x} \ (T_S + 1)$$

$$\Delta Dn = e_n - 2e_{n-1} + e_{n-2}$$

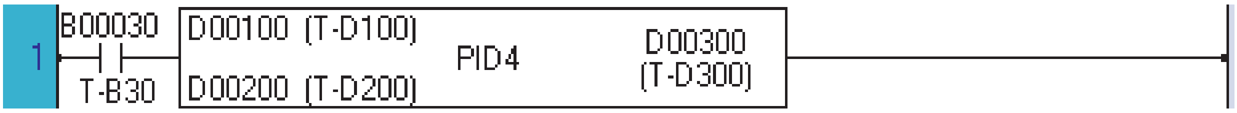
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | Initialization | OFF |
| ON | Execute PID every setting interval | ON when execution |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Top of Input Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| *B* | Top of Parameter | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| *C* | Top of output data | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



For the above shown sample ladder, data register are assigned as given below.

| **Input data** | | | **Control Parameters** | | | **Output data** | | |
|---|---|---|---|---|---|---|---|---|
| A | D100 | Process Input value | B | D200 | Proportional gain (KP) | C | D300 | Manipulation Value (MV) |
| A+1 | D101 | Set Value | B+1 | D201 | Integral time (TI) | C+1 | D301 | Previous error (en-1) |
| | | | B+2 | D202 | Derivative gain (KD) | C+2 | D302 | Previous error (en-2) |
| | | | B+3 | D203 | Gap (dead-band) GP | C+3 | D303 | Previous MV (MVn-1) |
| | | | B+4 | D204 | Not used | | | |
| | | | B+5 | D205 | Action Type | | | |

Parameters Details:

| | | | |
|---|---|---|---|
| A | Process Input Value | Data Range: | -32768 to +32767 |
| A+1 | Set Value | Data Range: | -32768 to +32767 |
| B | Proportional gain | Data Range: | -32768 to +32767 |
| B+1 | Integral time (sec) | Data Range: | 0 to 32767 |
| B+2 | Derivative gain | Data Range: | -32768 to +32767 |

B+3 Dead band (percentage) Data Range: 0 to 100

Dead band value = DB * SV / 100

Dead band value is expressed as *Dead band (DB)* percentage of *set value (SV)* in execution of PID instruction. PID instruction is executed only if error (en) is less than Dead band value.

When PID instruction is not executed MV is set automatically to 0 or 4095 (MVMAX) depending on compari son between SV and PV.

MV = 4095 .... if SV > PV

MV = 0 ..... if PV >= SV

B+4 Not Used

B+5 Action Data Range: 0 to 1

0: Direct Action, MV increases when PV is increased.

1: Reverse Action, MV decreases when PV is increased.

| | | | |
|---|---|---|---|
| C | Manipulation Value | Data Range: | 0 to 4095 |
| C+1 | Previous error Value (en-1) | Data Range: | -32768 to +32767 |
| C+2 | Previous error Value (en-2) | Data Range: | -32768 to +32767 |
| C+3 | Previous Manipulation Value | Data Range: | 0 to 4095 |

**Note -**

Users need to ensure that PID instruction is executed once every scan interval through Ladder Logic.

**Precaution -**

If both normal program and interrupt program contain this instruction, make sure both not executed simultaneously.

## Instruction-113: Upper Limit

Expression:

Input ─[ A UL B → C ]─ Output

Function:

When the input is ON, the following operation is executed. (Upper limit for A by B)
If A ≤ B, then C = A.
If A > B, then C = B.

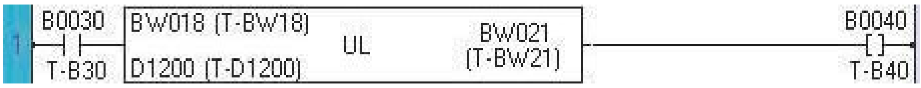
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution: not limited (A ≤ B) | OFF |
| | Execution: limited (A > B) | ON |

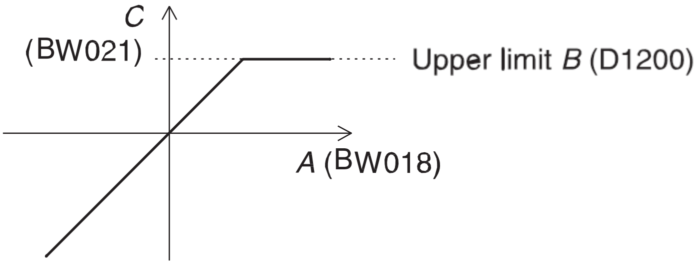
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Operation Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | √ |
| *B* | Upper Limit | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | √ |
| *C* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | | √ |

Example:



When B030 is ON, the upper limit operation is executed for the data of BW018 by the data of D1200, and the result is stored in BW021.



When BW018 is 3000 and D1200 is 4000, 3000 is stored in BW021 and B0040 is OFF.
When BW018 is 4500 and D1200 is 4000, the limit value 4000 is stored in BW021 and B0040 is ON.

Note
· This instruction deals with the data as signed integer (-32768 to 32767).

## Instruction-114: Lower Limit

Expression:

Input ─[ A LL B → C ]─ Output

Function:

When the input is ON, the following operation is executed. (Lower limit for A by B)
If A $\geq$ B, then C = A.
If A < B, then C = B.

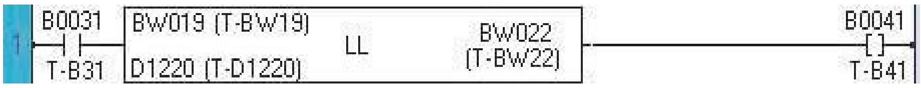
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution: not limited (A $\geq$ B) | OFF |
| | Execution: limited (A < B) | ON |

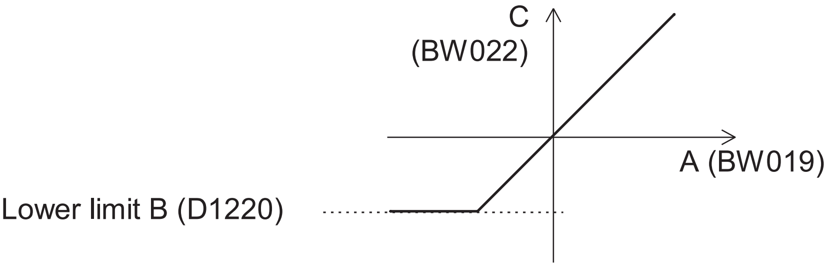
Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Operation Data | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | √ |
| *B* | Lower Limit | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | √ |
| *C* | Destination | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | | √ |

Example:



When B031 is ON, the lower limit operation is executed for the data of BW019 by the data of D1220, and the result is stored in BW022.



When BW019 is -1000 and D1220 is -1800, -1000 is stored in BW022 and B0041 is OFF.
When BW019 is 800 and D1220 is 1200, the limit value 1200 is stored in BW022 and B0041 is ON.

Note
· This instruction deals with the data as signed integer (-32768 to 32767)

## Instruction-115: Maximum Value

Expression:

Input ─[ A MAX (n) B ]─ Output

Function:

When the input is ON, this instruction searches for the maximum value from the table of size n words starting with A, and stores the maximum value in B and the pointer indicating the position of the maximum value in B+1. The allowable range of the table size n is 1 to 64.

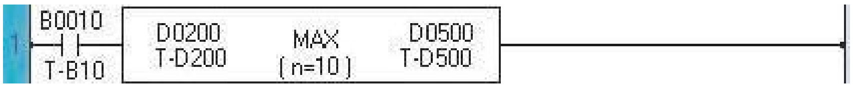
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of table | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| *B* | Result | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When B010 is ON, the maximum value is found from the register table D0200 to D0209 (10 words), and the maximum value is stored in D0500 and the pointer is stored in D0501.

| | | Pointer |
|---|---|---|
| D0200 | 100 | 0 |
| D0201 | 10000 | 1 |
| D0202 | -1000 | 2 |
| D0203 | 10 | 3 |
| D0204 | 0 | 4 |
| D0205 | 200 | 5 |
| D0206 | -300 | 6 |
| D0207 | 20000 | 7 |
| D0208 | -30 | 8 |
| D0209 | 20 | 9 |

→

| | | |
|---|---|---|
| D0500 | 20000 | (Maximum value) |
| D0501 | 7 | (Pointer) |

Note
- This instruction deals with the data as signed integer (-32768 to 32767).
- If there are two or more maximum value in the table, the lowest pointer is stored.
- If Index register K is used as operand B, the pointer data is discarded.

## Instruction-116: Minimum Value

Expression:

Input ─[ A MIN (n) B ]─ Output

Function:

When the input is ON, this instruction searches for the minimum value from the table of size n words starting with A, and stores the minimum value in B and the pointer indicating the position of the minimum value in B+1. The allowable range of the table size n is 1 to 64.

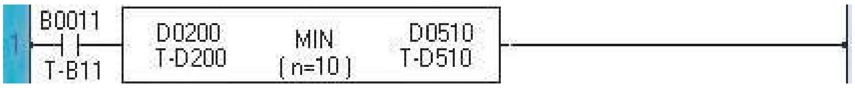
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of table | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| *B* | Result | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When B011 is ON, the minimum value is found from the register table D0200 to D0209 (10 words), and the minimum value is stored in D0510 and the pointer is stored in D0511.

| | | Pointer |
|---|---|---|
| D0200 | 100 | 0 |
| D0201 | 10000 | 1 |
| D0202 | -1000 | 2 |
| D0203 | 10 | 3 |
| D0204 | 0 | 4 |
| D0205 | 200 | 5 |
| D0206 | -300 | 6 |
| D0207 | 20000 | 7 |
| D0208 | -30 | 8 |
| D0209 | 20 | 9 |

| | | |
|---|---|---|
| → D0510 | -1000 | (Maximum value) |
| → D0511 | 2 | (Pointer) |

Note
· This instruction deals with the data as signed integer (-32768 to 32767).
· If there are two or more minimum value in the table, the lowest pointer is stored.
· If Index register K is used as operand B, the pointer data is discarded.

# Instruction-117: Average Value

Expression:

Input ─[ A AVE (n) B ]─Output

Function:

When the input is ON, this instruction calculates the average value of the data stored in the n registers starting with A, and stores the average value in B. The allowable range of the table size n is 1 to 64.

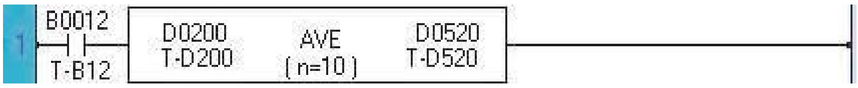
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Start of table | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| *n* | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 64 | |
| *B* | Result | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



When B012 is ON, the average value of the data stored in the register table D0200 to D0209 (10 words), and the average value is stored in D0520.

| | | |
|---|---|---|
| D0200 | 100 | 0 |
| D0201 | 10000 | 1 |
| D0202 | -1000 | 2 |
| D0203 | 10 | 3 |
| D0204 | 0 | 4 |
| D0205 | 200 | 5 |
| D0206 | -300 | 6 |
| D0207 | 20000 | 7 |
| D0208 | -30 | 8 |
| D0209 | 20 | 9 |

→ D0520 2900 (Average value)

# Instruction-118: Function Generator

Expression:

Input ─[ A FG (n) B ]─Output

Function:

When the input is ON, this instruction finds the function value *f(x)* for A as *x*, and stores it in C. The function *f(x)* is defined by the parameters stored in 2 * n registers starting with B.

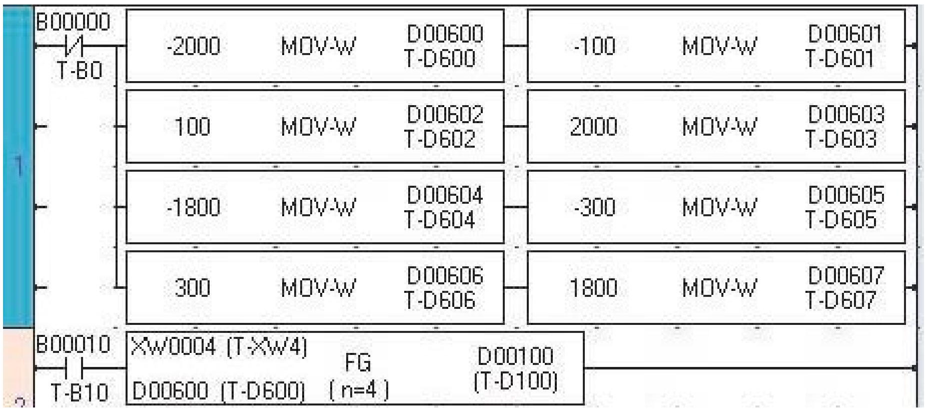
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | | |
| *A* | Input Value x | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | √ | |
| *n* | Parameter Size | | | | | | | | | | | | | | | | | | | | | 1 - 32 | |
| *B* | Starts of Parameters | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | | |
| *C* | Function Value *f(x)* | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | | |

Example:



B00000 T-B0

-2000 MOV-W D00600 T-D600

-100 MOV-W D00601 T-D601

100 MOV-W D00602 T-D602

2000 MOV-W D00603 T-D603

-1800 MOV-W D00604 T-D604

-300 MOV-W D00605 T-D605

300 MOV-W D00606 T-D606

1800 MOV-W D00607 T-D607

B00010 T-B10

XW0004 (T-XW4) FG D00100 (T-D100)

D00600 (T-D600) (n=4)

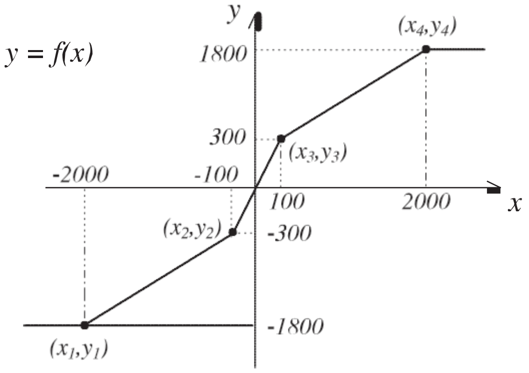
When B010 is ON, the FG instruction finds the function value *f(x)* for *x* = XW004, and stores the result in D0100.

The function *f(x)* is defined by 2 ´ 4 = 8 parameters stored in D0600 to D0607. In this example, these parameters are set at the first scan.

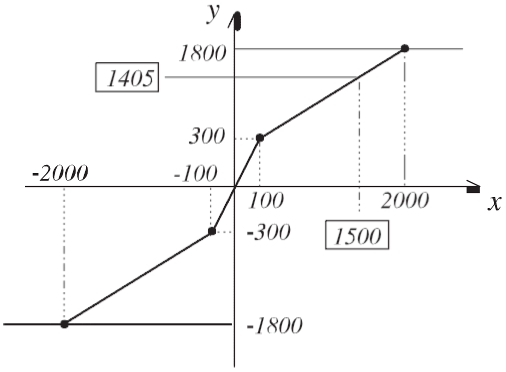
Parameter table

4 registers for *x* parameters and subsequent 4 registers for corresponding *f(x)* parameters

| | | |
|---|---|---|
| D0600 | -2000 | *x1* |
| D0601 | -100 | *x2* |
| D0602 | 100 | *x3* |
| D0603 | 2000 | *x4* |
| D0604 | -1800 | *y1* |
| D0605 | -300 | *y2* |
| D0606 | 300 | *y3* |
| D0607 | 1800 | *y4* |



The FG instruction interpolators *f(x)* value for *x* based on the n parameters of (*xi,yi*).
For example, if XW04 is 1500 (*x = 1500*), the result 1405 (*f(x) = 1405*) is stored in D0100.



Note

- · The order of the *x* parameters should be *x1* < *x2* < ... < *xi* < ... < *xn*. In the above example, the data of D0600 to D0603 should be D0600 < D0601 < D0602 < D0603.
- · If *x* is smaller than *x1*, *y1* is given as *f(x)*. In this example, D0604 data (-1800) is stored in D0100 if XW04 is smaller than D0600 (-2000).
- · If *x* is greater than *xn*, *yn* is given as *f(x)*. In this example, D0607 data (1800) is stored in D0100 if XW04 is greater than D0603 (2000).
- · The valid data range is -32768 to 32767.

## Instruction-119: USB Data log Upload

This ladder instruction is applicable on in FP-HMI with USB port support.
Expression:

```
                 [A]
Input  —             [C] LD-UPLD [D]  —          Output
                 [B]
```

Function:

The output of this instruction is a “*.csv” type file which will be uploaded in USB stick. This ladder supports only those units having, USB functionality.

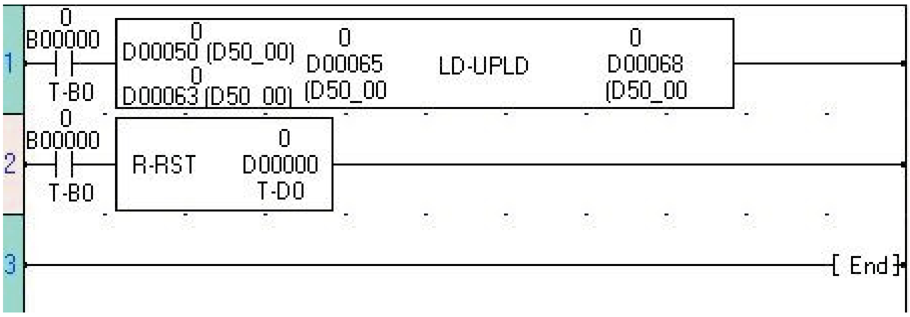
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Date time tag | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| *B* | Group (1-4) | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| *C* | Filename | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| D | Status Register | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |

Example:



Here user needs at least 16 tag registers to execute this task.
In the above shown image, once user defined tag address for “Date Time”, the application automatically considers consecutive 12 registers for date and time.
i.e. If tag address D000 is for Date time, then:
D0001 will be for Start Date
D0002 will be for Start Month
D0003 will be for Start Year
D0004 will be for Start Hour
D0005 will be for Start Minute

D0006 will be for Start Second
D0007 will be for End Date
D0008 will be for End Month
D0009 will be for End Year
D0010 will be for End Hour
D0011 will be for End Minute
D0012 will be for End Second.

Apart from this, user needs tag address for group Number (1 - 4).
User also has to defined another tag address for file name. This file name is for “*.csv” output file which can be in ASCII data entry format.
*User can also defined file name using a string which should be no longer than 8 characters.*
The Status byte will show the respective status code depending on the current status of the Task , like task complete, task is in execution, invalid date, invalid group number, USB stick is absent, invalid entry of File output device etc. etc.
Every time a new file will be created on USB stick. If old file with same name is present it will be overwritten.
The data can be sorted according to group number and the Start-End Date- Time only. e.g. the csv file can open in Windows Excel sheet or in Microsoft Word or in notepad.

This function can also be carried out as an application task in the OIS PLUS dispays.

## Instruction-120: Device Set

Expression:

Input ─[ DSET A ]─ Output

Function:

When the input is ON, the device A is set to ON if A is a device.

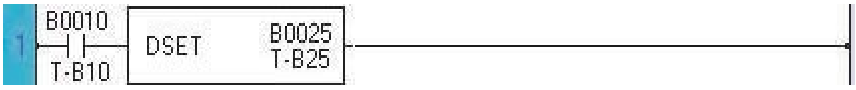
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



When B010 is ON, B025 is set to ON. The state of B025 is remained even if B010 comes OFF.

## Instruction-121: Device Reset

Expression:

Input ─[ D-RST A ]─ Output

Function:

When the input is ON, the device A is reset to OFF if A is a device.

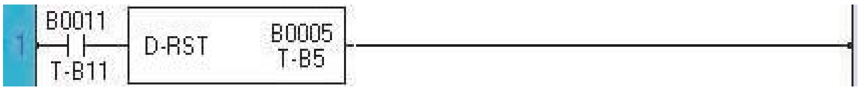
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



When B011 is ON, B005 is reset to OFF. The state of B025 is remained even if B011 comes OFF.

# Instruction-122: Register Set

Expression:

Input —[ R-SET A ]— Output

Function:

When the input is ON, the data HFFFF is stored in the register A if A is a register.

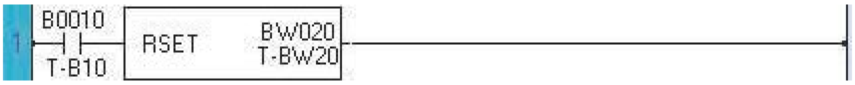
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Register | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | | |

Example:



When B010 is ON, the data HFFFF is stored in BW20. (R320 to R335 are set to ON). The state of BW20 is remained even if B010 comes OFF.

## Instruction-123: Register Reset

Expression:

Input ─[ R-RST A ]─ Output

Function:

When the input is ON, the data 0 is stored in the register A if A is a register.

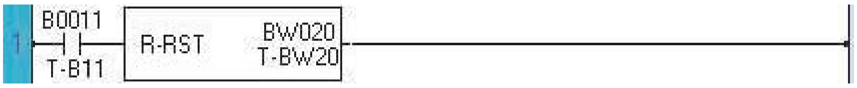
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Register | | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | | |

Example:



When B011 is ON, the data 0 is stored in BW20. (R320 to R335 are reset to OFF). The state of BW20 is remained even if B011 comes OFF.

# Instruction-124: Set Carry

Expression

Input ─[ SETC ]─ Output

Function:

When the input is ON, the carry flag (CF = S0) is set to ON.

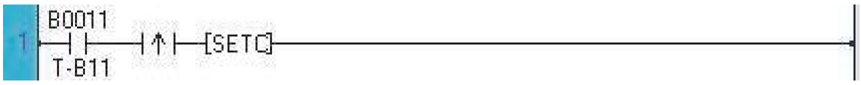
Execution condition:

| Input | Operation | Output | CF |
|---|---|---|---|
| OFF | No execution | OFF | — |
| ON | Execution | ON | Set |

Operand:

No operand is required.

Example:



When B011 is changed from OFF to ON, the carry flag S0 is set to ON.

## Instruction-125: Reset Carry

Expression:

Input —[ RSTC ]— Output

Function:

When the input is ON, the carry flag (CF = S0) is reset to OFF.

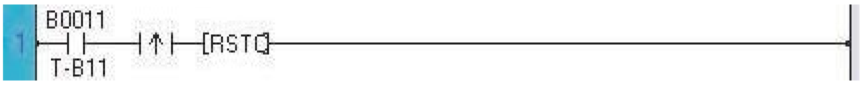
Execution condition:

| Input | Operation | Output | CF |
|---|---|---|---|
| OFF | No execution | OFF | — |
| ON | Execution | ON | Reset |

Operand:

No operand is required.

Example:



When B011 is changed from OFF to ON, the carry flag S0 is reset to OFF.

## Instruction-126: Encode

Expression:

Input —[ A ENC (n) B ]— Output

Function:

When the input is ON, this instruction finds the bit position of the most significant ON bit in the bit table, size 2 n bits starting with 0 bit (LSB) of A, and stores it in B.

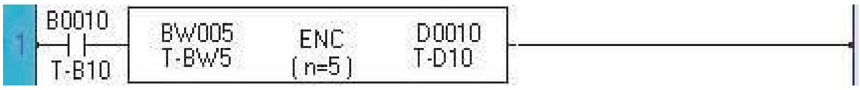
Execution condition:

| Input | Operation | Output | CF |
|---|---|---|---|
| OFF | No execution | OFF | — |
| ON | Normal Execution | ON | — |
|  | There is no ON bit (no execution) | OFF | Set |

Operand:

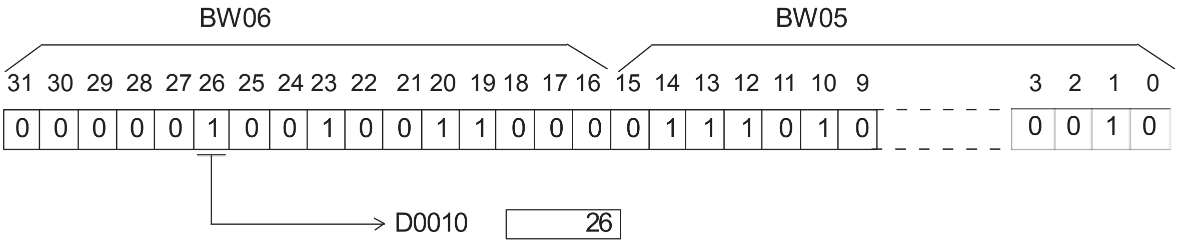
|  | Name | Device |  |  |  |  |  |  | Register |  |  |  |  |  |  |  |  |  |  |  | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R |  |  |
| A | Start of Table |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ |  |  |  | √ |  |  |  |
| n | Table Size |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 - 8 |  |
| B | Encode Result |  |  |  |  |  |  |  |  | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |  |  |  |

Example:



$2^5$ (=32) bits starting with 0 bit of BW05 (B050 to B06F) are defined as the bit table.
When B010 is ON, the most significant ON (1) bit position in the bit table is searched, and the position is stored in D0010.

The following figure shows an operation example.



Note:
· If there is no ON bit in the bit table, the instruction error flag (ERF = S0034) is set to ON.

# Instruction-127: Decode

Expression:

Input ─[ A DEC (n) B ]─Output

Function:

When the input is ON, this instruction sets the bit position which is designated by lower n bits of A to ON in the bit table, size $2^n$ bits starting with 0 bit (LSB) of B, and resets all other bits to OFF.

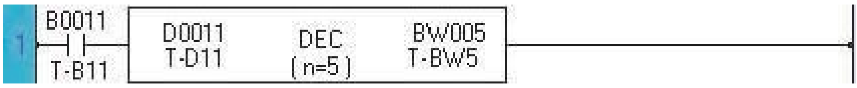
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

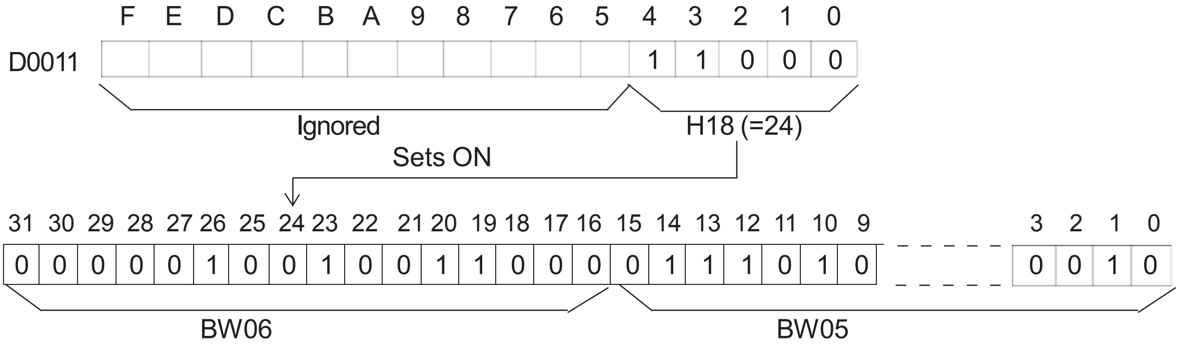
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Decode Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | | |
| n | Table Size | | | | | | | | | | | | | | | | | | | | 1 - 8 | |
| *B* | Start of Table | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



$2^5$ (=32) bits starting with 0 bit of BW05 (B050 to B06F) are defined as the bit table.
When B011 is ON, the bit position designated by lower 5 bits of D0011 in the bit table is set to ON, and all other bits in the table are reset to OFF.

The following figure shows an operation example.



## Instruction-128: Bit Count

Expression:

Input ─[ A BC B ]─ Output

Function:

When the input is ON, this instruction counts the number of ON (1) bits of A, and stores the result in B.

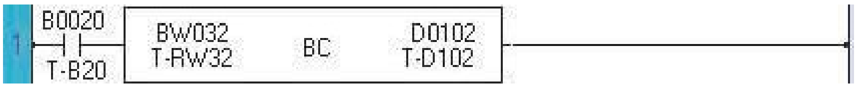
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | Source | | | | | | | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | | √ | |
| *B* | Count Data | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When B020 is ON, the number of ON (1) bits of the register BW032 is counted, and the result is stored in D0102.

The following figure shows an operation example.

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BW032 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

Counts the number of ON (1) bits = 7

| | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

The result data (7) is stored in binary

# Instruction-129: Flip-Flop

Expression:

```
Set Input   -[S   F/F  Q]-  Output
Reset Input-[R    A     ]
```

Function:

When the set input is ON, the device A is set to ON. When the reset input is ON, the device A is reset to OFF. When both the set and reset inputs are OFF, the device A remains the state. If both the set and reset inputs are ON, the device A is reset to OFF.
The state of the output is the same as the device A.

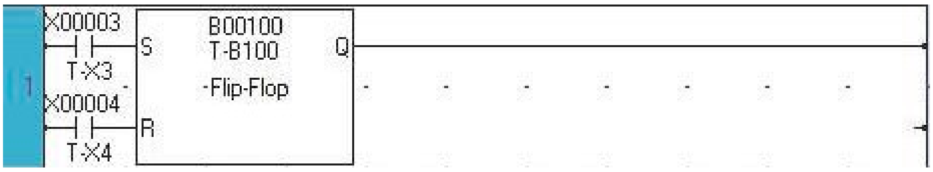
Execution condition:

| Set input | Reset input | Operation | Output |
|---|---|---|---|
| OFF | OFF | No execution ( A remains previous state) | Same as A |
| | ON | Resets A to OFF | |
| ON | OFF | Sets A to ON | |
| | ON | Resets A to OFF | |

Operand:

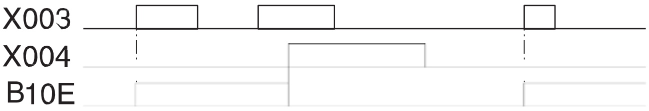
| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Device | | √ | √ | √ | | | √ | | | | | | | | | | | | | | |

Example:



When X003 is ON, B10E is set to ON. When X004 is ON, B0100 is reset to OFF. If both are ON, B0100 is reset to OFF.

An example timing diagram is shown below.



Note:
· For the set input, direct linking to a connecting point is not allowed. In this case, insert a dummy contact (always ON = S04F, etc.) just before the input. Refer to Note of Shift register Function.

## Instruction-130: Direct I/O

Expression:

Input ─[ I/O (n) A ]─ Output

Function:

When the input is ON, this instruction immediately updates all external input (XW) and all output (YW) registers of the slot specified by register.
· For XW register ... reads the data from corresponding slot (Base and expansion)
· For YW register ... writes the data into corresponding slot (Base and expansion).

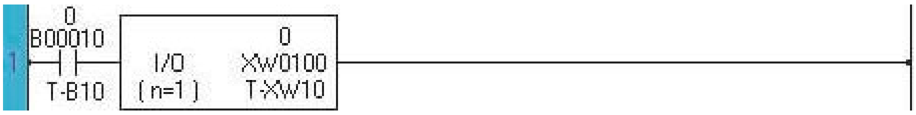
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No execution | OFF |
| ON | Execution | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *n* | Register size | | | | | | | | | | | | | | | | | | | | | |
| *A* | Start of registers | | | | | | | | √ | √ | | | | | | | | | | | | |

Example:



When B010 is ON, all registers of slot1 are updated immediately.

Note1:
· In normal execution XW or YW registers (Input and output registers of base and expansion) are updated / written only once in the main scan. (Refer flow chart). But when direct IO instruction is used reading of physical input and writing to physical outputs is carried out at the time of execution of ladder instruction..

Note2:
· The Direct I/O instruction can be programmed in the main program and in the interrupt program.
If this instruction is programmed in both, the instruction in the main program should be executed in interrupt disable state. Refer to EI (Enable interrupt) and DI (Disable Interrupt) instructions.

## Instruction-131: Set Calendar

Expression:

Input –[A CLND]– Output

Function:

When the input is ON, the built-in clock/calendar is set to the date and time specified by 6 registers starting with A. If an invalid data is contained in the registers, the operation is not executed and the output is turned ON.

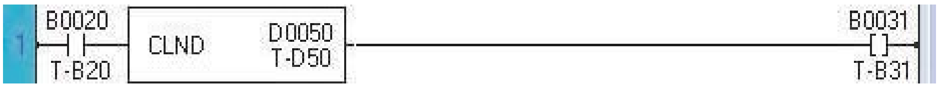
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No Operation | OFF |
| ON | Execution (data is valid) | OFF |
| | No execution (data is not valid) | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| A | Start of table | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When B020 is ON, the clock/calendar is set according to the data of D0050 to D0055, and the output is OFF (B0031 is OFF).
If D0050 to D0055 contains invalid data, the setting operation is not executed and the output is turned ON (B0031 comes ON).

D050 (first) to D055 (last) contains

| F 8 | 7 0 |
|---|---|
| 00 | Year 00 to 99 |
| 00 | Month 01 to 12 |
| 00 | Day 01 to 31 |
| 00 | Hour 00 to 23 |
| 00 | Minute 00 to 59 |
| 00 | Second 00 to 59 |

Year 00 to 99 ⇄ 2000 to 2099

Note
The day of the week is automatically.
Sunday = 0 , Monday = 1 ,Tuesday = 2 ...........Saturday = 6.

Currently following system registers (SW) are updated after 2 sec

| ***Modbus address*** | ***SW*** | |
|---|---|---|
| 420011 | SW10 | Year (00 To 99 <=> 2000 To 2099) |
| 420012 | SW11 | Month (01 To 12) |
| 420013 | SW12 | Date ( 01 To 31) |
| 420014 | SW13 | Hour (00 To 23) |
| 420015 | SW14 | Min (00 To 59) |
| 420016 | SW15 | Sec (00 To 59) |
| 420017 | SW16 | Day (00 To 07) |

If there is any error RTC_Fail Flag is set to ON (SW 03 BIT 02)

## Instruction-132: Calendar Operation

Expression:

Input ─[A CLDS B]─ Output

Function:

When the input is ON, this instruction subtracts the date and time stored in 6 registers starting with A from the current date and time, and stores the result in 6 registers starting with B.
If invalid data is contained in the registers, the operation is not executed and the output is turned ON.

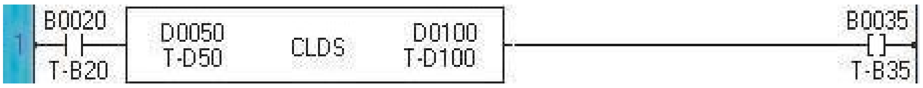
Execution condition:

| Input | Operation | Output |
|---|---|---|
| OFF | No operation | OFF |
| ON | Execution (data is valid) | OFF |
| | No execution (data is not valid) | ON |

Operand:

| | Name | Device | | | | | | | Register | | | | | | | | | | | | Constant | Index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | X | Y | B | S | T. | C. | M | XW | YW | BW | SW | T | C | D | I | J | K | MW | R | | |
| *A* | ubtrahend | | | | | | | | √ | √ | √ | √ | √ | √ | √ | | | | √ | | | |
| *B* | Result | | | | | | | | | √ | √ | √ | √ | √ | √ | | | | √ | | | |

Example:



When B020 is ON, the date and time data recorded in D0050 to D0055 are subtracted from the current date and time of clock/calendar, and the result is stored in D0100 to D0105.
In normal operation, the output is OFF (B0035 is OFF). If D0050 to D0055 contains invalid data, the operation is not executed and the output is turned ON (B0035 comes ON).

Current date & time

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| H0098 | | D0050 | H0097 | | D0100 | H0000 | (Year) |
| H0001 | | D0051 | H0010 | | D0101 | H0003 | (Month) |
| H0015 | minus | D0052 | H0010 | ──→ | D0102 | H0007 | (Day) |
| H0017 | | D0053 | H0015 | | D0103 | H0001 | (Hour) |
| H0000 | | D0054 | H0030 | | D0104 | H0030 | (Minute) |
| H0000 | | D0055 | H0000 | | D0105 | H0000 | (Second) |

Note
· Future date and time cannot be used as subtrahend A.
· In the calculation result, it means that 1 year is 365 days and 1 month is 30 days.

**TOSHIBA INDUSTRIAL PRODUCTS**:

- Adjustable Speed Drives
- Motors
- Motor Controls
- PLCs, DCS, & Instrumentation
- Uninterruptible Power Systems

Industrial Division •
13131 West Little York Road
Houston, Texas 77041 USA •
Tel +713-466-0277  US 1-800-231-1412 •

TOSHIBA
Leading Innovation >>>

www.toshiba.com/ind
plc@tic.toshiba.com