

Introduction

TinyPLC is an advanced type of PLC, which is designed with a different concept from that of the general PLC. Using Relay Ladder Diagram as a programming tools same as the general PLC, TinyPLC has different composition of system conspicuously. The most difference that distinguishes itself from other PLC is that TinyPLC has module type CPU. We, Comfile Technology, provide CPU of TinyPLC in form of module independently. Let us look over the characteristics of TinyPLC.

- Programming with the Relay Ladder Diagram
- User can change freely the input/output direction of the port.
- Been stored in flash memory, a program can be conserved safely without the battery-backup.
- It is possible to create and edit a ladder program without an additional handy-loader and download it via RS232 port.
- Support real-time monitoring.
- Free software for editing ladder program : MPGL2, TinyPLC studio
- Supporting the source protection
- As TinyPLC is semiconductor type, it can be installed on the PCB. You can lessen unnecessary wiring work.
- Economical price cut down manufacturing cost.
- Small footprint: the whole size of the product can be considerably reduced.

In addition, TinyPLC provides various functions that conventional PLCs cannot support without add-on unit.

- Built-in 30 KHz 16 bit high-speed counter
- Built-in A/D converter with 10 bit & 8 channels
- Supporting LCD module(16 X2 to 20 X4)
- Supporting 7 Segment module (up to 8 piece of 5 DIGIT)
- General purpose communication
- Supporting 8 x 8 key matrix
- Supporting Digital Temperature sensors (DS1820)

TinyPLC is, as above, highly efficient one-board processors integrating functions of previous PLC into semiconductor type of a module. Comfile Technology, are very proud of this new product. We hope that you will get a good result out of using our product.

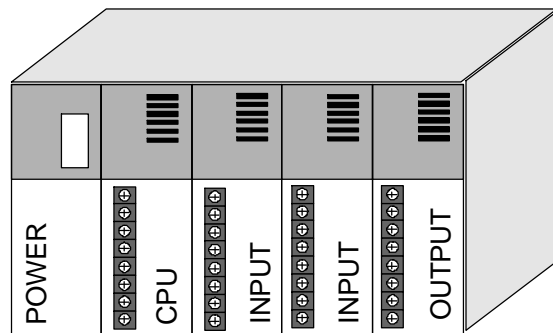
Chapter 1

TinyPLC Overview

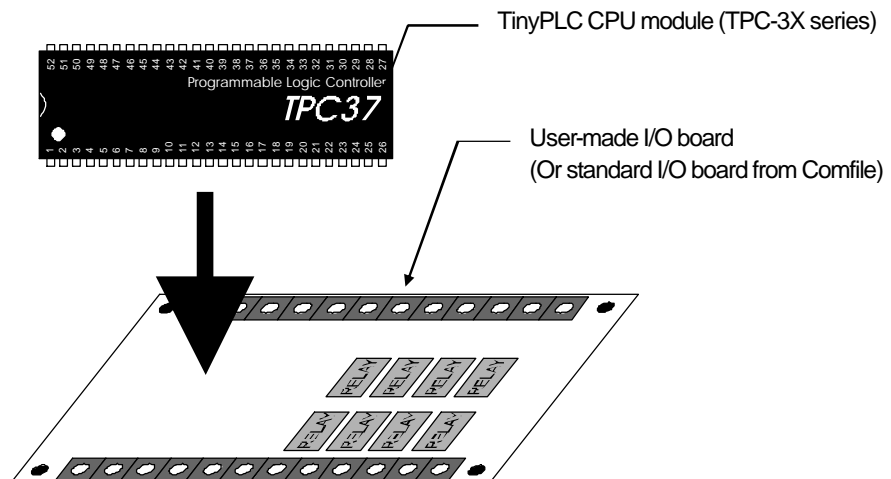
This chapter covers overview and basic specifications of TinyPLC.

1.1. Overview

PLC, Programmable Logic Controller, is a typical controller that is commonly used in industrial field. The picture shown below is general type of PLC which you have been familiar with. Because that its all-in-one case contains power supply, CPU and I/O, you can place it inside of a control box and program various functions by installing extra units where necessary.



TinyPLC has very different composition from the general PLCs. “TinyPLC” is provided in the forms of CHIP (similar to CPU of general PLC). By adding additional device such as I/O circuits, users can organize their own PLC.



TinyPLC gives advantages to user as follows.

- Being able to design and assemble I/O parts except CPU, users can make their own PLC. (Suitable for designing dedicated machines)
- Needing to purchase CPU module only at lower-price than usual PLC, user can reduce the cost of materials. (profitable to mass-production)
- User can freely set direction of I/O ports and it makes the most of I/O port.
- You can reduce cable-wiring work to the minimum, if basic circuits are on PCB.
- As it is small, you can make compact product.

TinyPLC is suitable for the people who mass-produce FA machine. Originally, TinyPLC is designed that users compose I/O board and Power Supply. To make easy the process examining and producing products, Comfile Technology put basic Power Supply Modules and I/O boards (so called baseboard) as standard type on market. If you use Tiny PLC Module, Baseboard and Power Supply Module together, it is same as using common PLC.

The table below lists Baseboard type and their features.

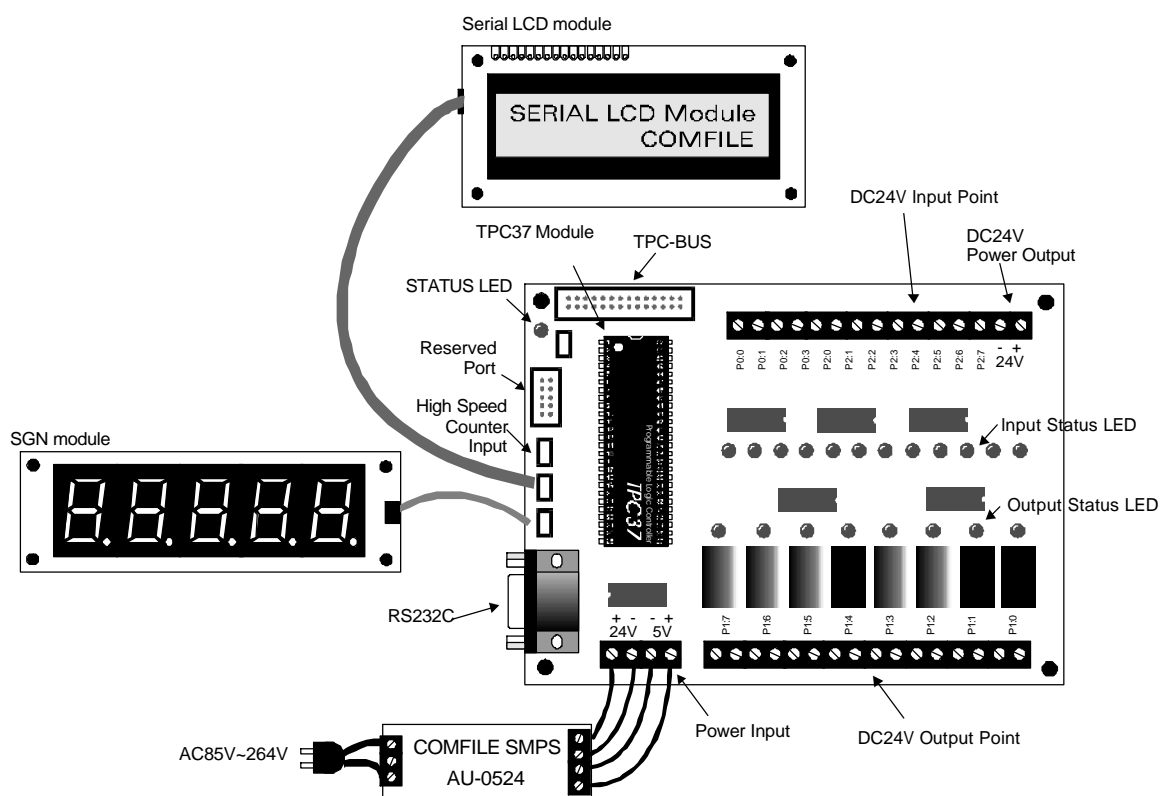
Model Name	Description	Point of Input and Output
BASE-C20R (TPC33, 37)	Basic Baseboard for TPC37 20 points of input and output port General purpose RS232C connect port LCD, SGN connection ports	DC24V Input : 12 Points RELAY Output : 8 Points
EIO-DI8-24A	Expansion I/O Module connecting to TPC-BUS DC24V Input 8 Points	DC24V Input : 8 Points
EIO-RY8A	Expansion I/O Module connecting to TPC-BUS REALY Output 8 Points	Relay Output : 8 Points
EIO-TR8A	Expansion I/O Module connecting to TPC-BUS TR Output 8 Points	TR Output : 8 Points

Model Name	Description	Point of Input and Output
BASE-C64R (TPC38)	Basic Baseboard for TPC38 64 points of input and output port General purpose RS232C connect port LCD, SGN connection ports	DC24V Input : 24 Points RELAY Output : 24 Points I/O CELL point: 16 Points

*You can find more various baseboard and detailed information at www.comfile.co.kr or Product Catalog, Vo.7.

1.2. Baseboard

The picture shown below is the diagram of BASE-C20R that is generally used as basic Baseboard of TinyPLC. You can connect the BASE-C20R to LCD (Serial Display Module provided by Comfile Technology) and SGN (Seven Segment Network module) directly. There are RS232C connector and Level converter Circuit for Communication with PC. Basically, there are 8points of built-in Relay Output and 12points of DC24V Input. Power Supply is available for 5V and 24V (available for connecting to AU-0524, COMFILE SMPS product).

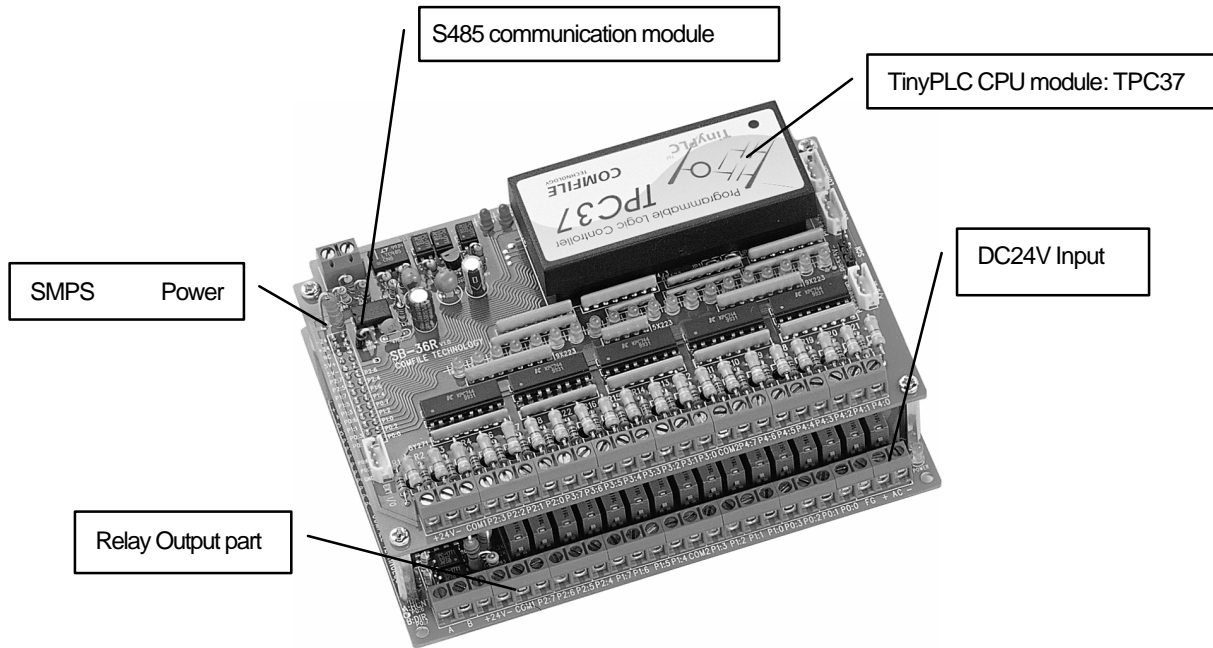


You can add Expansion I/O module (Model no. starts with EIO), if necessary, by connecting to TPC-BUS.

You can use TPC33 and TPC37 as Tiny PLC Module (CPU).

1.3. Block type PLC

Block type PLC consists of CPU module and SMPS (Switching Module Power Supply) Relay Output and DC24V Input Part. The picture below shows composition of SB36.



The table shown below is the representative product line of Single-unit PLC. Comfile Technology is providing more various type of PLC such as Indicator type, Module type and Case type. You can find more various baseboard and detailed information at www.comfile.co.kr or Product Catalog, Vo.7

Model	CPU	Power Input	Input/Output
SB-14R	TPC33	AC85V ~ 264V	DC24V input: 8 points RELAY output: 6 points RS485 communication port
SB-30S	TPC33	AC85V ~ 264V	DC24V input: 8 points RELAY output: 6 points RS485 communication port EIO extension module connection
SB-22R	TPC33	AC85V ~ 264V	DC24V input: 12points RELAY output: 10points RS485 communication port
SB-36R	TPC37	AC85V ~ 264V	DC24V input: 20points RELAY output: 16points RS485 communication port Encoder input port

For detailed information on Single-unit PLC, see appendix.

1.4. Specification

1.4.1. TPC3X series

The table below lists comparison about function and capacity of TPC3X series.

Contents	TPC 26	TPC 33	TPC 37	TPC 38
Program memory	16K byte	16K byte	128K byte	128K byte
Scan-time	2.5mS	2.5 or 5mS	2.5 or 5mS	5mS
Basic Instruction	27	27	27	27
Application Instruction	84	84	84	84
Input/Output Relay (P area)	26 points (Available for setting I/O to all of 26 Points)	24 points (Available for setting I/O to all of 26 Points)	40 points (Available for setting I/O to 24points, 8 point is INPUT only, 8 points is OUTPUT only)	68 points (Available for setting I/O to 24points, 20 point is INPUT only, 24 points is OUTPUT only)
DA Output (DA area)	-	-	-	10bit, 2channel (PWM)
Internal-relay (M area)	256points	256points	1024points	1024points
Step control (S area)	16 pairs, 255 steps	16 pairs, 255 steps	32 pairs, 255 steps	32 pairs, 255 steps
KEEP Relay (K area)	512 points	512 points	512 points	512points
Timer (T area)	64 points(word)	64points(word)	256points(word)	256points(word)
counter (C area)	32 points(word)	32 points(word)	256 points(word)	256 points(word)
Data (D area)	220 word	220 word	1024 word	1024 word
LCD display buffer (CH area)	20 by 4 (80 byte)	20 by 4 (80 byte)	20 by 4 (80 byte)	20 by 4 (80 byte)
SGN display buffer (G area)	5 by 8 (40 byte)	5 by 8 (40 byte)	5 by 8 (40 byte)	5 by 8 (40 byte)
AD converter (AD area)	8 word (10bit, 8channel built-in A/D) (Using P2:X as A/D input)	8 word (10bit, 8channel built-in A/D) (Using P2:X as A/D input)	8 word (10bit, 8channel built-in A/D) (Using P3:X as A/D input)	8 word (10bit, 8channel built-in A/D) (Using P3:X as A/D input)
High-speed timer (CNT area)	1 word (16 BIT)	1 word (16 BIT)	1 word (16 BIT)	1 word (16 BIT)
Package	40pins DIP type (semiconductor chip)	36 pins DIP type	52pins DIP type	80 pins DIP type

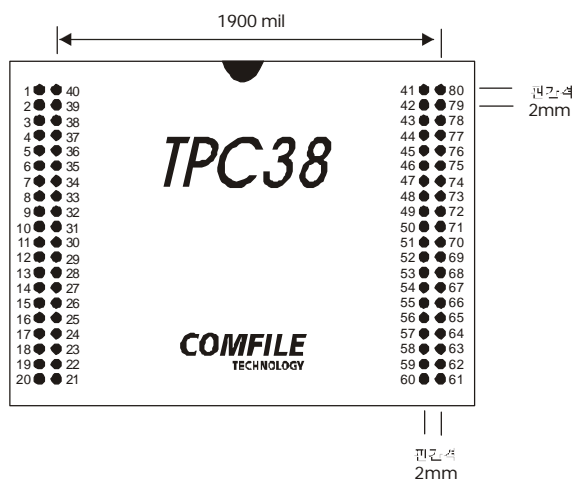
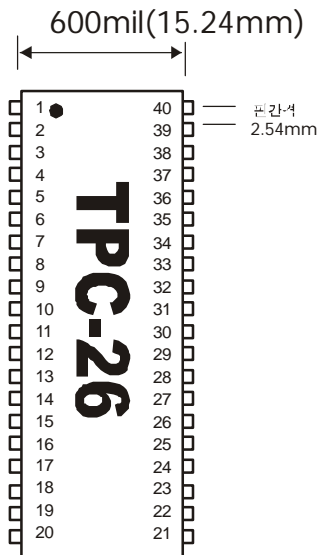
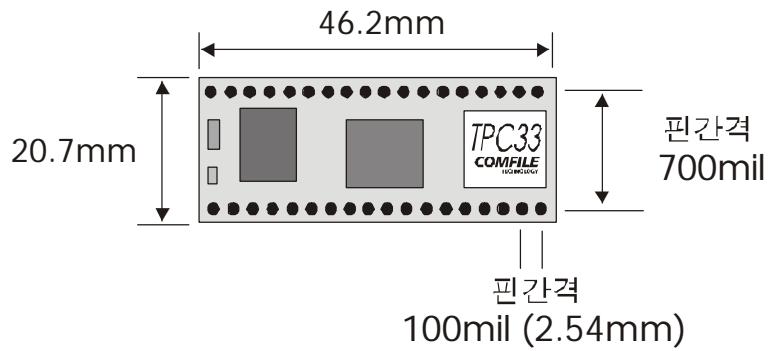
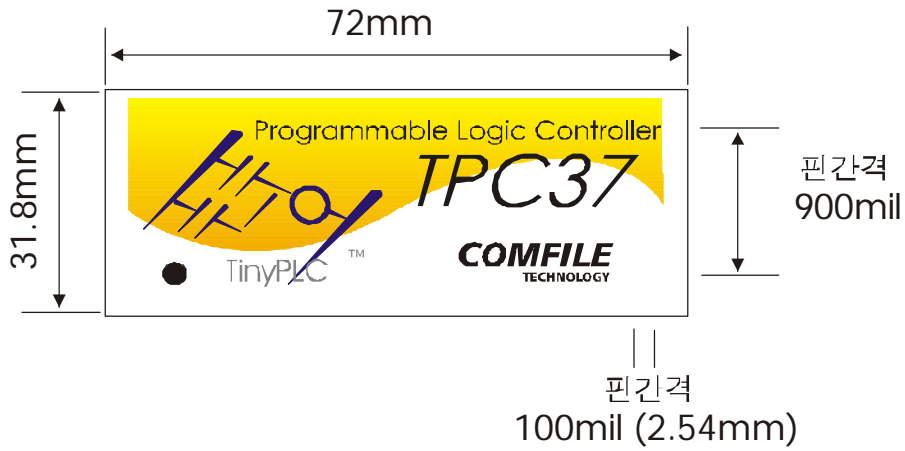
- TPC3X series is TPC26, TPC33, TPC37 and TPC38. (Jan 2003)
- SGN is Product name of Seven Segment Display Module from Comfile Technology.
- In case of connection with LCD, only Serial LCD module provided from Comfile Technology is allowed to connect to TPC3X (1line connection)

1.4.2. TinyPLC

The table below describes general specifications of Tiny PLC.

Item	Specification
SMPS Power Source Input	AC 85V~254V 50~60Hz (in case apply AU-0524)
Tiny PLC Power Source	DC 5V Single Power Source
Power Consumption	10VA
Power Failure Allowance	10mS
Operation Temperature	0~55
Storage Temperature	-10 ~ 70
Working Humidity	5 ~95% RH, no dew form
Working Condition	no dust and gas
Noise Area	1500V Vpp, 1uS
Pressure-proof	AC 1500V / min
Insulation Resistance	Over 10M ohm (DC 500V insulation resistance)
Vibration-proof	16.7Hz complex vibration 2mm, 2 hr
Impact	10 G (3 shock each in 3 axes)
Ground	100 Ohm

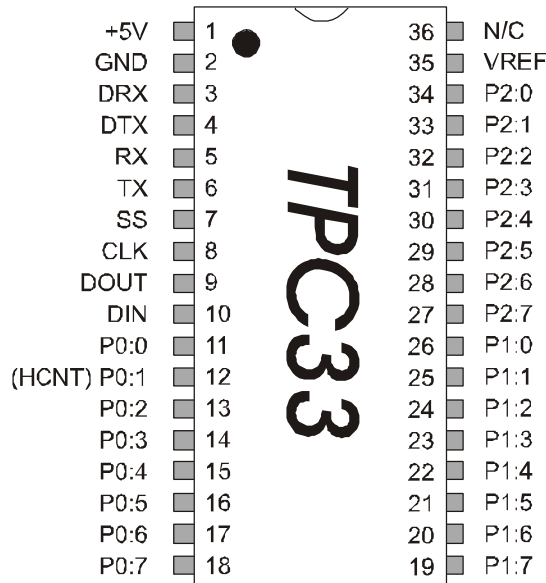
1.5. Outline



1.6. PINOUT

1.6.1. TPC33

This section describes PIN OUT of TPC33.



Pin no.	name	Input /Output*	Description
1	+5V	Power	Power Source Input Terminal (Support 4.5V~5.5V)
2	GND	Power	Ground
3	DRX	Input	RX Terminal for download (Connect to RS232C-TD of PC)
4	DTX	Output	TX Terminal for download (Connect to RS232C-RD of PC)
5	RX	Input	Common communication terminal RX (Connect to TD of Host)
6	TX	Output	Common communication terminal TX (Connect to RD of Host)
7	485TE	Output	RS485 transmission enable signal
8	CLK	Output	For expansion (Clock signal)
9	Dout	Output	For expansion (Data Output signal)
10	Din	Input	For expansion (Data Input signal)
11	P0:0	I/O	I/O port
12	P0:1	I/O	I/O port (High-speed counter input port)
13	P0:2	I/O	I/O port
14	P0:3	I/O	I/O port
15	P0:4	I/O	I/O port
16	P0:5	I/O	I/O port
17	P0:6	I/O	I/O port
18	P0:7	I/O	I/O port
19	P1:7	I/O	I/O port
20	P1:6	I/O	I/O port
21	P1:5	I/O	I/O port
22	P1:4	I/O	I/O port

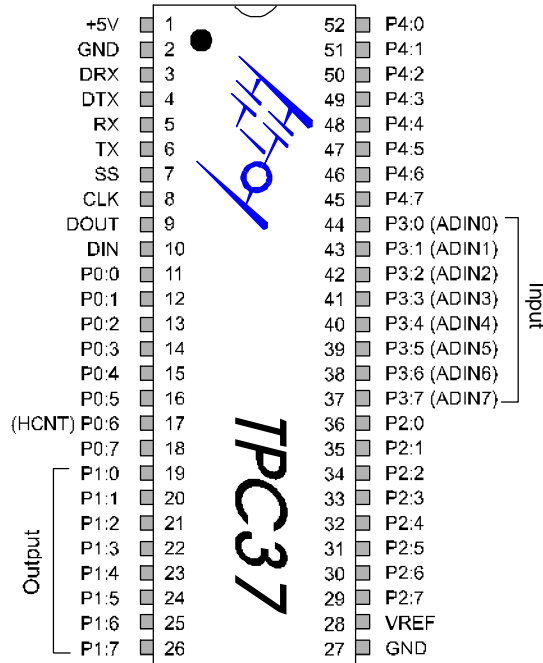
23	P1:3	I/O	I/O port
24	P1:2	I/O	I/O port
25	P1:1	I/O	I/O port
26	P1:0	I/O	I/O port
27	P2:7	I/O	I/O port (Analog Ch7 Input port)
28	P2:6	I/O	I/O port (Analog Ch6 Input port)
29	P2:5	I/O	I/O port (Analog Ch5 Input port)
30	P2:4	I/O	I/O port (Analog Ch4 Input port)
31	P2:3	I/O	I/O port (Analog Ch3 Input port)
32	P2:2	I/O	I/O port (Analog Ch2 Input port)
33	P2:1	I/O	I/O port (Analog Ch1 Input port)
34	P2:0	I/O	I/O port (Analog Ch0 Input port)
35	Vref	I	Analog standard voltage Input
36	N/C		Not connected

* Pin no. 8, 9 and 10 (grey-colored) is not available.

* On installing, take care of location of Pin no. 1.

1.6.2. TPC 37

This section describes PIN OUT of TPC37.



Pin no.	name	Input/ Output*	Function
1	+5V	Power	Power Source Input Terminal (Support 4.5V~5.5V)
2	GND	Power	Ground
3	DRX	Input	RX Terminal for download (Connect to RS232C-TD of PC)
4	DTX	Output	TX Terminal for download (Connect to RS232C-RD of PC)
5	RX	Input	Common communication terminal RX (Connect to TD of Host)
6	TX	Output	Common communication terminal TX (Connect to RD of Host)
7	485TE	Output	RS485 transmission enable signal
8	CLK	Output	For expansion (Clock signal)
9	Dout	Output	For expansion (Data Output signal)
10	Din	Input	For expansion (Data Input signal)
11	P0:0	I/O	I/O port
12	P0:1	I/O	I/O port
13	P0:2	I/O	I/O port
14	P0:3	I/O	I/O port
15	P0:4	I/O	I/O port
16	P0:5	I/O	I/O port
17	P0:6	I/O	I/O port (High-speed counter Input port)
18	P0:7	I/O	I/O port
19	P1:0	Output	Output port only
20	P1:1	Output	Output port only
21	P1:2	Output	Output port only
22	P1:3	Output	Output port only
23	P1:4	Output	Output port only

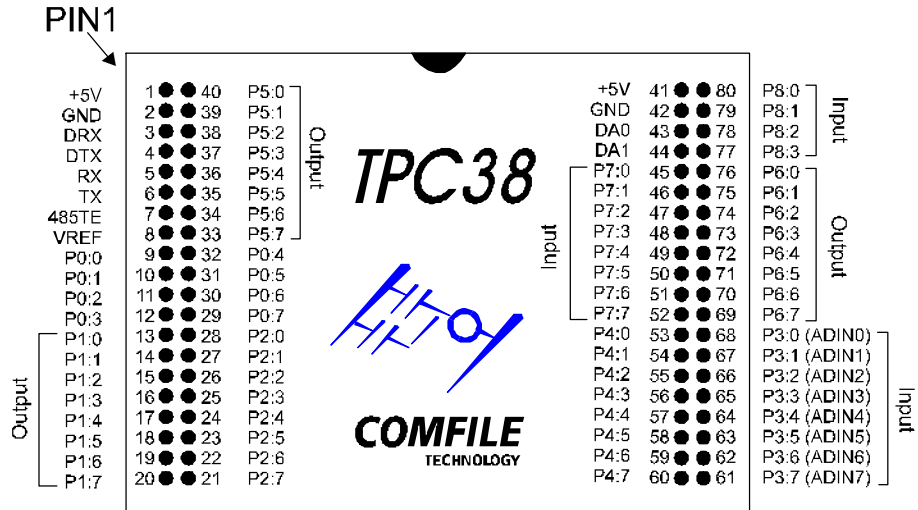
24	P1:5	Output	Output port only
25	P1:6	Output	Output port only
26	P1:7	Output	Output port only
27	GND	Power	Ground terminal
28	VREF	Input	A/D standard voltage Input (AD converting between 0V~VREF. Up to 5V)
29	P2:7	I/O	I/O port
30	P2:6	I/O	I/O port
31	P2:5	I/O	I/O port
32	P2:4	I/O	I/O port
33	P2:3	I/O	I/O port
34	P2:2	I/O	I/O port
35	P2:1	I/O	I/O port
36	P2:0	I/O	I/O port
37	P3:7	Input	Input port only (Analog Ch7 Input port)
38	P3:6	Input	Input port only (Analog Ch6 Input port)
39	P3:5	Input	Input port only (Analog Ch5 Input port)
40	P3:4	Input	Input port only (Analog Ch4 Input port)
41	P3:3	Input	Input port only (Analog Ch3 Input port)
42	P3:2	Input	Input port only (Analog Ch2 Input port)
43	P3:1	Input	Input port only (Analog Ch1 Input port)
44	P3:0	Input	Input port only (Analog Ch0 Input port)
45	P4:7	I/O	I/O port
46	P4:6	I/O	I/O port
47	P4:5	I/O	I/O port
48	P4:4	I/O	I/O port
49	P4:3	I/O	I/O port
50	P4:2	I/O	I/O port
51	P4:1	I/O	I/O port
52	P4:0	I/O	I/O port

* Pin no. 8, 9 and 10 (grey-colored) is not available.

* On installing, take care of location of Pin no. 1.

1.6.3. TPC 38

This section describes PIN OUT of TPC38.



Pin no.	Name	Input/ Output*	Function
1	+5V	Power	Power Source Input Terminal (Support 4.5V~5.5V)
2	GND	Power	Ground
3	DRX	Input	RX Terminal for download (Connect to RS232C-TD of PC)
4	DTX	Output	TX Terminal for download (Connect to RS232C-RD of PC)
5	RX	Input	Common communication terminal RX (Connect to TD of Host)
6	TX	Output	Common communication terminal TX (Connect to RD of Host)
7	485TE	Output	RS485 transmission enable signal
8	VREF	Input	A/D standard voltage Input (AD converting between 0V~VREF Up to 5V)
9	P0:0	I/O	I/O port
10	P0:1	I/O	I/O port
11	P0:2	I/O	I/O port
12	P0:3	I/O	I/O port
13	P1:0	Output	Output port only
14	P1:1	Output	Output port only
15	P1:2	Output	Output port only
16	P1:3	Output	Output port only
17	P1:4	Output	Output port only
18	P1:5	Output	Output port only
19	P1:6	Output	Output port only
20	P1:7	Output	Output port only
21	P2:7	I/O	I/O port
22	P2:6	I/O	I/O port
23	P2:5	I/O	I/O port
24	P2:4	I/O	I/O port
25	P2:3	I/O	I/O port

26	P2:2	I/O	I/O port
27	P2:1	I/O	I/O port
28	P2:0	I/O	I/O port
29	P0:7	I/O	I/O port
30	P0:6	I/O	I/O port (High-speed counter Input port)
31	P0:5	I/O	I/O port
32	P0:4	I/O	I/O port
33	P5:7	Output	Output port only
34	P5:6	Output	Output port only
35	P5:5	Output	Output port only
36	P5:4	Output	Output port only
37	P5:3	Output	Output port only
38	P5:2	Output	Output port only
39	P5:1	Output	Output port only
40	P5:0	Output	Output port only
41	+5V	Power	Power Source Input Terminal (Support 4.5V~5.5V)
42	GND	Power	Ground
43	DA0	Output	DA Output port only 0 (PWM Output state)
44	DA1	Output	DA Output port only 1 (PWM Output state)
45	P7:0	Input	Input port only
46	P7:1	Input	Input port only
47	P7:2	Input	Input port only
48	P7:3	Input	Input port only
49	P7:4	Input	Input port only
50	P7:5	Input	Input port only
51	P7:6	Input	Input port only
52	P7:7	Input	Input port only
53	P4:0	I/O	I/O port
54	P4:1	I/O	I/O port
55	P4:2	I/O	I/O port
56	P4:3	I/O	I/O port
57	P4:4	I/O	I/O port
58	P4:5	I/O	I/O port
59	P4:6	I/O	I/O port
60	P4:7	I/O	I/O port
61	P3:7	Input	Input port only (Analog Ch7 Input port)
62	P3:6	Input	Input port only (Analog Ch6 Input port)
63	P3:5	Input	Input port only (Analog Ch5 Input port)
64	P3:4	Input	Input port only (Analog Ch4 Input port)
65	P3:3	Input	Input port only (Analog Ch3 Input port)
66	P3:2	Input	Input port only (Analog Ch2 Input port)
67	P3:1	Input	Input port only (Analog Ch1 Input port)
68	P3:0	Input	Input port only (Analog Ch0 Input port)
69	P6:7	Output	Output port only
70	P6:6	Output	Output port only
71	P6:5	Output	Output port only
72	P6:4	Output	Output port only
73	P6:3	Output	Output port only
74	P6:2	Output	Output port only
75	P6:1	Output	Output port only

76	P6:0	Output	Output port only
77	P8:3	Input	Input port only
78	P8:2	Input	Input port only
79	P8:1	Input	Input port only
80	P8:0	Input	Input port only

* On installing, make certain location of Pin no. 1.

Ports of TPC38 are digested as follows;

I/O port: 24points (P0, P2, P4)

Input port only: 20points (P3, P7, P8)

Output port only: 24points (P1, P5, P6)

DA Output port: 2points

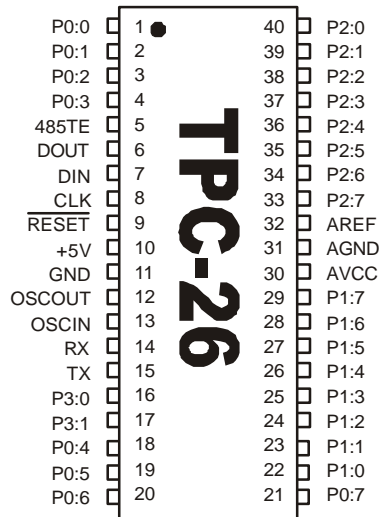
Total: 70points

Caution

The following instruction is prohibited from using at P5~P8 of TPC38.
LCDOUT, SGNOUT, THIN, KEYSKAN

1.6.4. TPC26

TPC26 is semiconductor and Module type PLC different from other TinyPLCs.



Pin no.	Name	Input/ Output*	Function
1	P0:0	I/O	I/O port
2	P0:1	I/O	I/O port (High-speed counter Input port)
3	P0:2	I/O	I/O port
4	P0:3	I/O	I/O port
5	485TE	Output	RS485 transmission enable signal
6	DOUT	Output	For expansion (DATA OUTPUT) signal
7	DIN	Input	For expansion (DATA INPUT) signal
8	CLK	Output	For expansion (CLOCK) signal
9	/RESET	Input	Reset signal (Reset will turn on at Low status)
10	+5V	Power	Power source Input terminal (Support 4.5V~5.5V)
11	GND	Power	Ground terminal
12	OSCOUT	Output	Oscillator connect terminal
13	OSCIN	Input	Oscillator connect terminal (7.3728MHz Crystal connecting)
14	RX	Input	Download and RS232/485 communication port (19200 baud rate)
15	TX	Output	Download and RS232/485 communication port (19200 baud rate)
16	P3:0	I/O	I/O port
17	P3:1	I/O	I/O port
18	P0:4	I/O	I/O port
19	P0:5	I/O	I/O port
20	P0:6	I/O	I/O port
21	P0:7	I/O	I/O port
22	P1:0	I/O	I/O port
23	P1:1	I/O	I/O port
24	P1:2	I/O	I/O port
25	P1:3	I/O	I/O port
26	P1:4	I/O	I/O port

27	P1:5	I/O	I/O port
28	P1:6	I/O	I/O port
29	P1:7	I/O	I/O port
30	AVCC	Power	Analog Power source Input terminal
31	AGND	Power	Analog Ground Input terminal
32	AREF	Input	Analog Input Standard voltage Input terminal Convert 0~3V, if input is 3V. Convert 0~5V, if input is 5V.
33	P2:7	I/O	Input port only (Analog Ch7 Input port)
34	P2:6	I/O	Input port only (Analog Ch6 Input port)
35	P2:5	I/O	Input port only (Analog Ch5 Input port)
36	P2:4	I/O	Input port only (Analog Ch4 Input port)
37	P2:3	I/O	Input port only (Analog Ch3 Input port)
38	P2:2	I/O	Input port only (Analog Ch2 Input port)
39	P2:1	I/O	Input port only (Analog Ch1 Input port)
40	P2:0	I/O	Input port only (Analog Ch0 Input port)

- All pins name of which start with P are I/O pins. The name of all I/O pins starts with P. It can be used as Input or Output pin alternatively in accordance with User's definition.
- Because CLKIN terminal can detect Rising Edge, High-speed pulse even about 20 KHz can be counted.
- OSCIN, OSCOUT must be connected to **7.3728MHz Crystal**.
- RESET terminal should be connected to 5V Power source without any additional circuit.

CAUTION

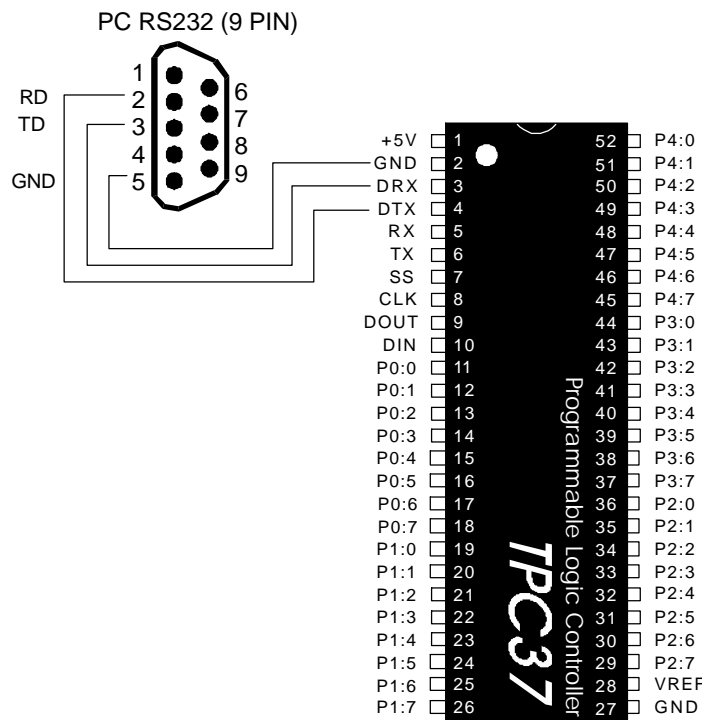
- Reading and erasing TPC26 chip with a general ROM-writer could bring about critical adverse effect or disability. We strongly recommend you do not to perform that.
- TPC26, which is provided in form of chip, is very vulnerable to static electricity.
- It is strongly recommended to install Bypass condenser (around 0.1uF) at the closest part of power supply (+5V, GND). Bypass condenser is an essential component ensuring stable operation of the chip. (Ceramic or monolithic type condenser)
- Data communication of TPC26 is 19200 baud rate. (Other TPC 3X is 9600 baud rate).

1.7. Connection with PC

1.7.1. RS232C cable

DRX and DTX, allocated at pin no. 3 and 4 of TPC33/37/38 each, are communication port for use of download and monitoring. There is no need of extra converting device in connecting with RD and TD terminal in RS232C port of PC.

TinyPLC	RS232C of PC (9 pins)	RS232C of PC (25pins)
2 (GND)	5 (GND)	7 (GND)
3 (DRX)	3 (TD)	2 (TD)
4 (DTX)	2 (RD)	3 (RD)

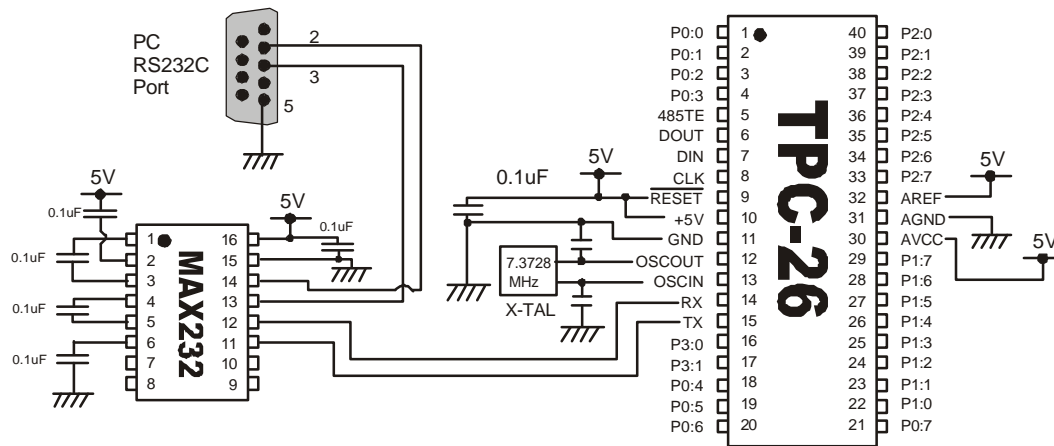


Communication protocol is RS232C, 19200 baud rate, N,8,1. As TPC module has built-in Level convert circuit, download ports (pin no.3 and 4) can be connected to RS232C directly.

*** Caution: Maximum length of download cable can not be more than 2M and the cable must be separated in operation. (To avoid affixing of unnecessary noise)**

1.7.2. Download circuit for TPC26

Communication of TPC26 to PC is different from that of other modules. The picture shown below is simplest circuit for operating TPC26.



Level converting chip, MAX232, is used to communicate with RS232C in PC. (As most of Single-unit PLC and baseboard have a circuit which has same function as shown above circuit, users have only to connect cable for instant use.) Being semiconductor type, TPC26 should be connected with basic circuits such as Oscillator circuit, Power-supply circuit and Reset circuit in order to operate.

Having same composition as above circuit, basic dedicated baseboard of TPC26, BASE-F24R, would be used with ease.

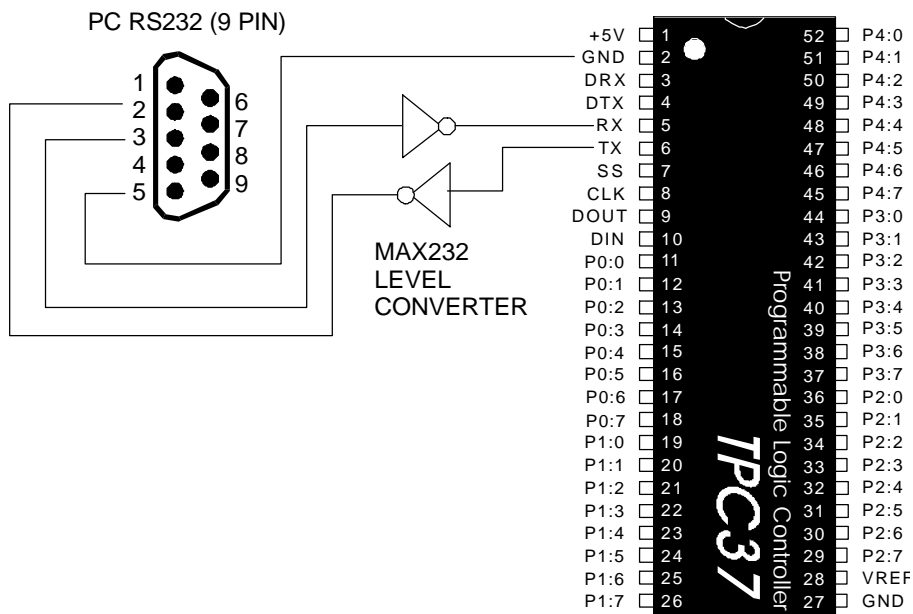
1.7.3. Link to computer

TinyPLC can be linked to computer. Through RS232 (or RS485), user can read and write in data memory of TinyPLC. RX and TX, pin no. 5 and 6, are common communication ports (protocol is fixed at 9600 baud rate, N, 8, 1).

TinyPLC pin	RS232C of PC (9 pins)	RS232C of PC (25 pins)
2 (GND)	5 (GND)	7 (GND)
5 (RX)	3 (TD)	2 (TD)
6 (TX)	2 (RD)	3 (RD)

As common communication port (pin no.5 and 6) have not built-in level convert circuit, user MUST use level converting chip to input 5V level signal.

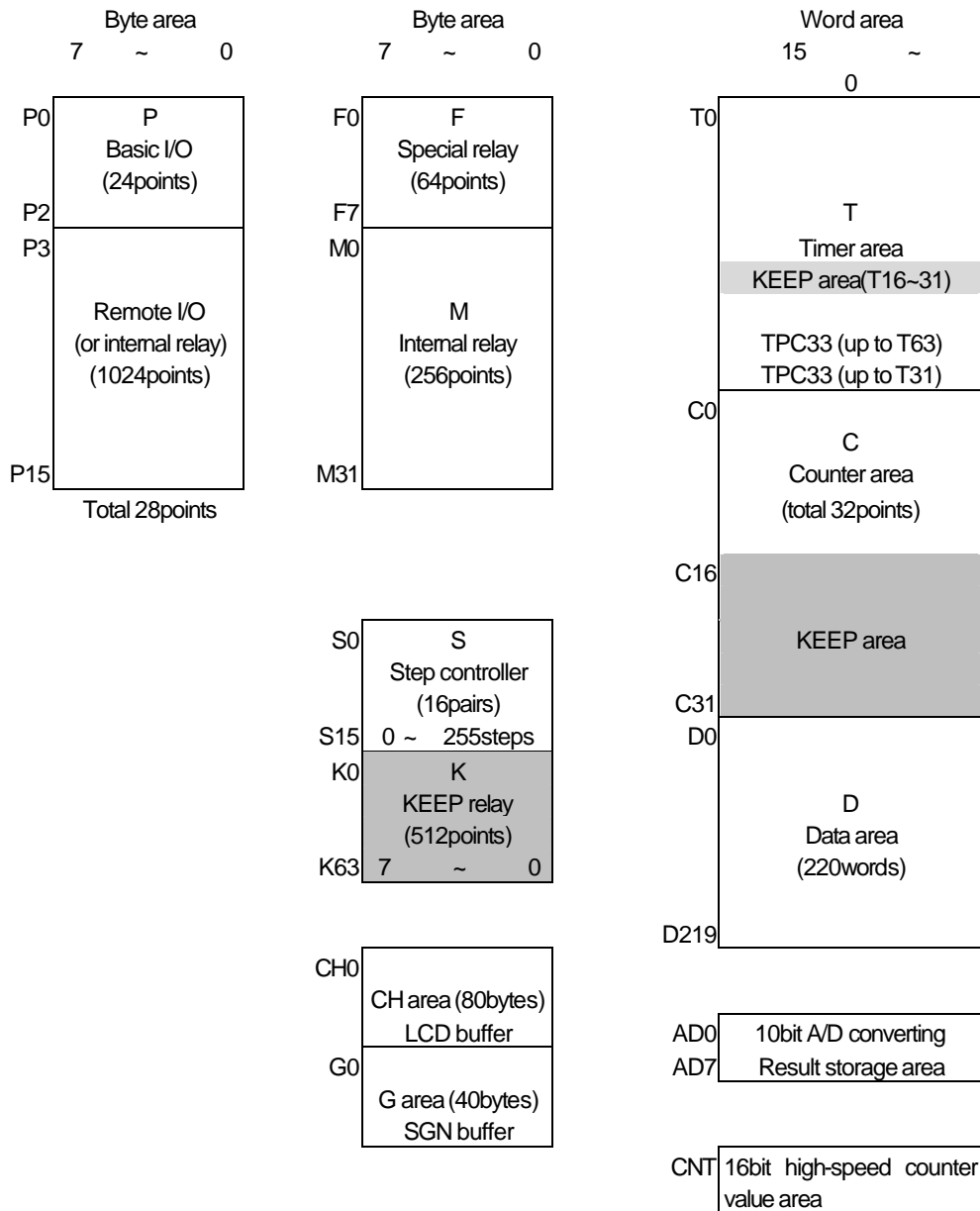
(Baseboard from Comfile Technology includes converting chip)



In case of TPC26, download port can be used for data communication. See chapter 6 Data Communication for detailed information on data communication.

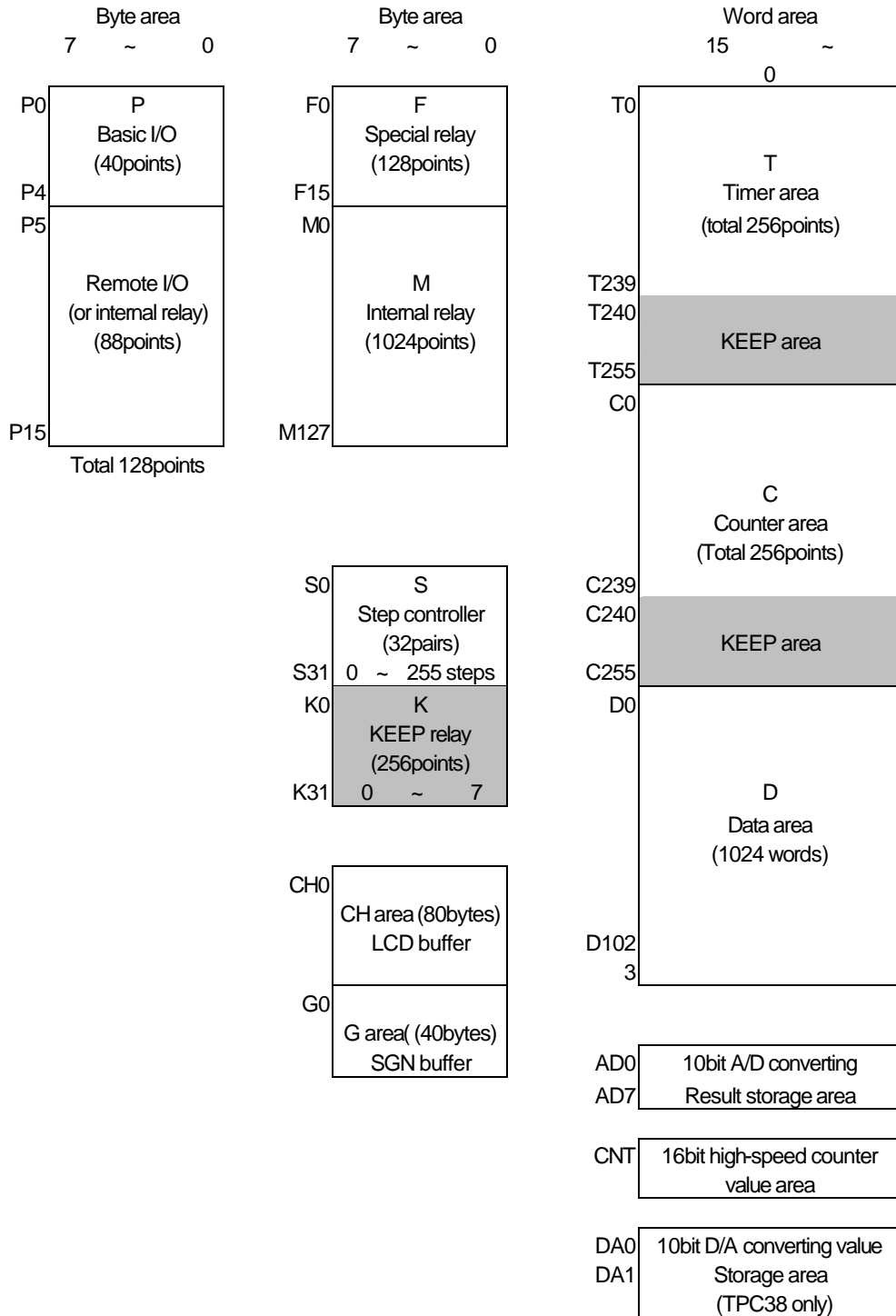
1.8. Data memory map

1.8.1. TPC33

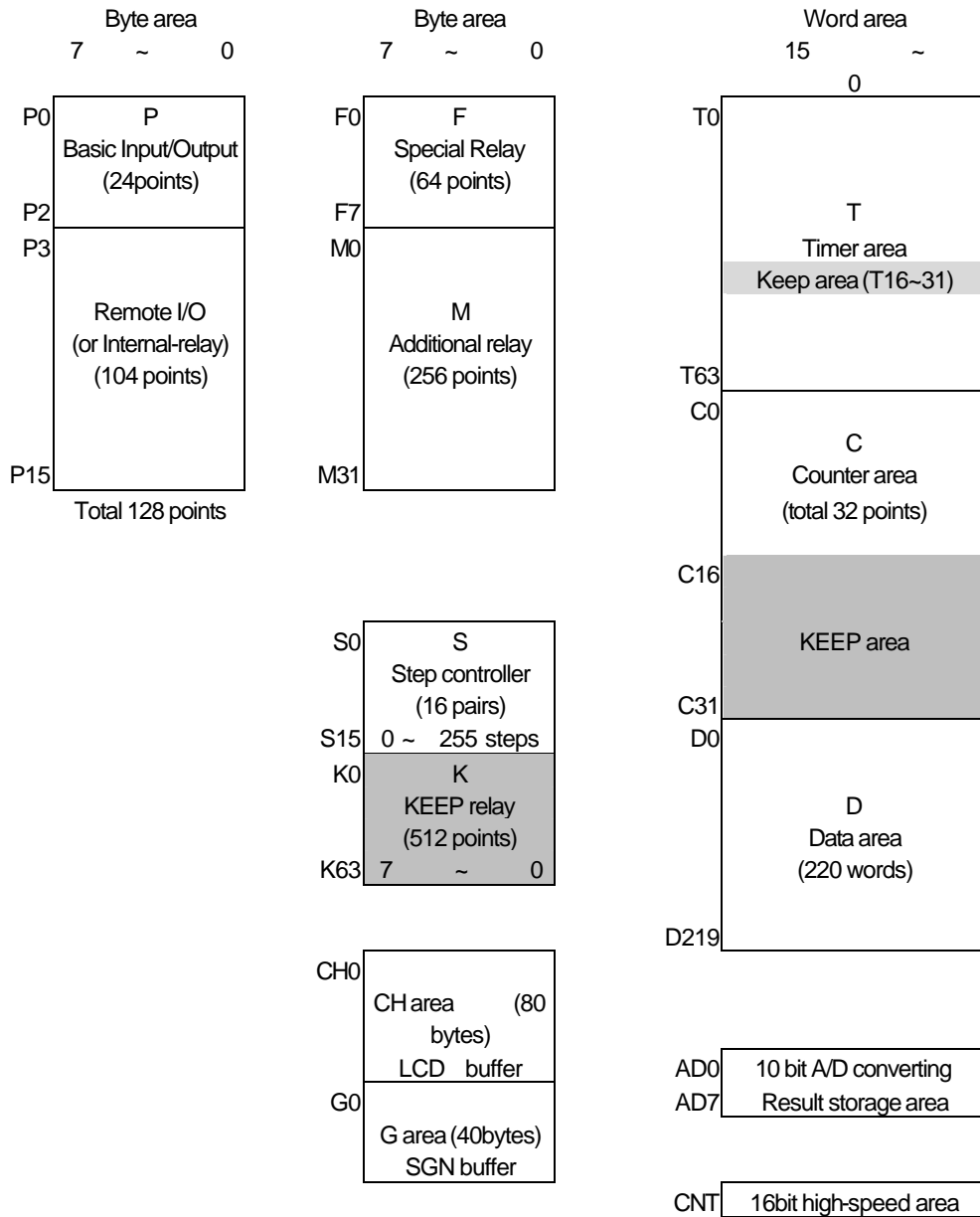


- As being in keep area, some part of Timer/Counter area is able to conserve data even while power is off.
- Timer/Counter area is unitized as 16bit (1 word) unit, the others (P, M, K, CH, G) is unitized as byte unit.
- Word unit area is stored data with following high/low rank byte order. (High rank byte enter low address)
- Except Keep area, all area turn "0" when power-on.
- As data in Keep area is written in EEPROM physically, there is no need of extra battery backup.

1.8.2. TPC37, 38



1.8.3. TPC26

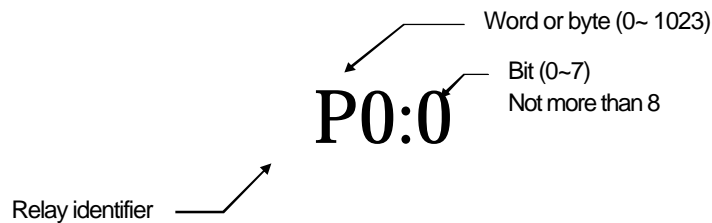


- As being in keep area, some part of Timer/Counter area is able to conserve data even while power is off.
- Timer/Counter area is unitized as 16bit (1 word) unit, the others (P, M, K, CH, G) is unitized as byte unit.
- Word unit area is stored data with following high/low rank byte order. (High rank byte enter low address)
- Except Keep area, all area turn "0" when power-on.
- As data in Keep area is written in EEPROM physically, there is no need of extra battery backup.

1.9. Relay

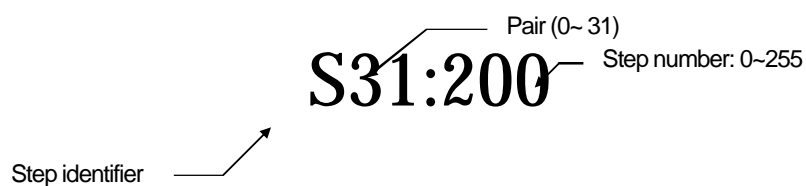
1.9.1. Relay expression

All relay of TinyPLC can be symbolized as follows;



Relay type	Relay identification	Word express range	Bit express range	Function
I/O relay	P	0~15	0~7	Turn ON/OFF outer ports, read pins state.
Special relay	F	0~15	0~7	
Internal relay	M	0~127	0~7	
Keep relay	K	0~31	0~7	Keep state while power off
Timer	T	0~255	Not use	Timer terminal
Counter	C	0~255	Not use	Counter terminal
Data	D	0~1023	Not use	Data area

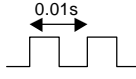
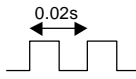
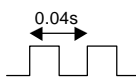
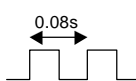
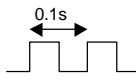
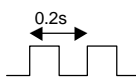
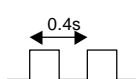
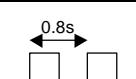
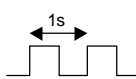
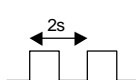
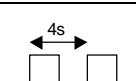
S relay used in step control is expressed as follows;

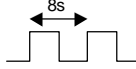
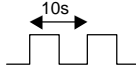
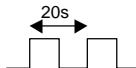
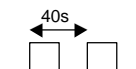
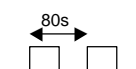


Relay type	Relay identification	Word express range	Bit express range	Function
Step control relay	S	0~31	0~255	Step control (up to 32pair 256step)

1.9.2. Special relay

Special relay is controlling state of CPU, mode selection and signal.

Special relay	Function	Remarks	
F0	F0:0	Constant OFF	
	F0:1	Constant ON	
	F0:2	ON during 1 scan time at initial power input	
	F0:3	OFF during 1 scan time at initial power input	
	F0:4		
	F0:5		
	F0:6		
	F0:7		
F1	Not use		
F2	F2:0	Generating pulse by 0.01sec cycle (5mS on : 5mS off) Repeating ON/OFF by 1 scan time cycle	
	F2:1	Generating pulse by 0.02sec cycle (10mS on : 10mS off)	
	F2:2	Generating pulse by 0.04sec cycle (20mS on : 20mS off)	
	F2:3	Generating pulse by 0.08sec cycle (40mS on : 40mS off)	
F3	F3:0	Generating pulse by 0.1sec cycle (0.05 sec on : 0.05 sec off)	
	F3:1	Generating pulse by 0.2sec cycle (0.1 sec on : 0.1 sec off)	
	F3:2	Generating pulse by 0.4sec cycle (0.2 sec on : 0.2 sec off)	
	F3:3	Generating pulse by 0.8sec cycle (0.4 sec on : 0.4 sec off)	
F4	F4:0	Generating pulse by 1sec cycle (0.5 sec on : 0.5 sec off)	
	F4:1	Generating pulse by 2sec cycle (1sec on : 1sec off)	
	F4:2	Generating pulse by 4sec cycle (2 sec on : 2 sec off)	

	F4:3	Generating pulse by 8sec cycle (4 sec on : 4 sec off)	
F5	F5:0	Generating pulse by 10sec cycle (5 sec on : 5 sec off)	
	F5:1	Generating pulse by 20sec cycle (10 sec on : 10 sec off)	
	F5:2	Generating pulse by 40sec cycle (20 sec on : 20 sec off)	
	F5:3	Generating pulse by 80sec cycle (40 sec on : 40 sec off)	
F6		Selecting Korean font for Korean LCD 0 : SMALL SAEMMOOL 1 : SMALL GOTHIC 2 : BIG MYUNGJO 3 : BIG TAEGOTHIC	For WRITE
F7		Address for RS485 communication Allowed appointing 0~ 255	
F8		Real time sec-clock Operating in 0 ~ 59sec F8:0 ? 1sec ON/1sec OFF	TPC33
F9		Real time minute-clock (operating 0~ 59minutes)	N/A in TPC33
F10		Real time hour-clock (operating 0~255hour, return to 0 after reaching 255)	N/A in TPC33

About REAL TIME CLOCK

RTC that is allocated at F8, F9, F10, is operational in only TPC37, TPC38. RTC can be clear to “0” automatically upon power-on and can conserve data while power is on. RTC is much useful in operation, which need precise timer.

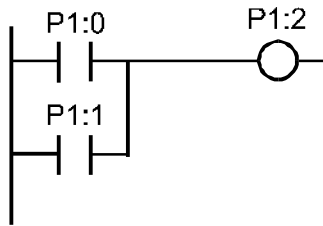
Chapter 2

Basic of PLC

This chapter gives you general idea of PLC and ladder programming. If you have ever used PLC, you do not have to read this chapter.

2.1. What is PLC?

PLC (Programmable Logic Controller), which is one of the most typical FA controllers, has wide range of users and is on the right track as industrial controllers. As being comprehensible diagram programming language, Relay Ladder Diagram is providing ease approach to engineers, who have not majored in electronics/computer science, and is applicable in short time without accumulated experience in industrial field. PLC is a controller that is programmed and operated by Relay Ladder Diagram.



<Example: Relay Ladder Diagram>

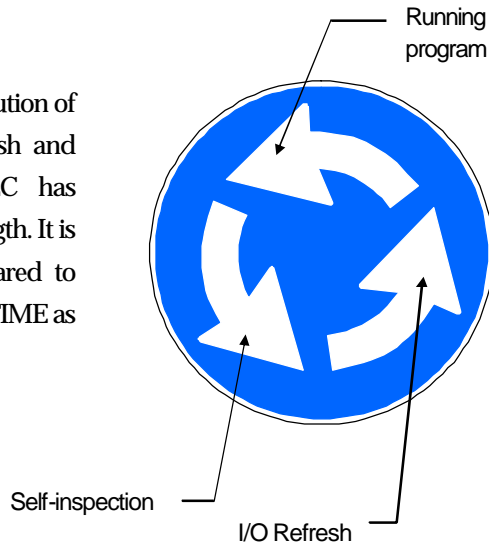
The most outstanding difference of Relay Ladder Diagram from general programming language such as C, Assembly and BASIC is “Multi-tasking”.(Multi-tasking: conducting more than 2 operations simultaneously) C and Assembly are sequential programming language. It means that only one operation can be processed at a certain point of time in processing.

But, it is a problem in PLC programming due that Ladder Diagram supports multi-tasking. Diagram shown below describes a structure that X is on if A inputs and Y is on if B inputs at the same time.



Regardless of complicity of Ladder program, input points/relays on the ladder are always ready to work in accordance with input. A concept of SCANTIME in programming PLC is from this. SCAN TIME means time space between beginning and end of Relay Ladder Program in operation. Relay Ladder Program runs repeatedly at certain constant SCANTIME interval.

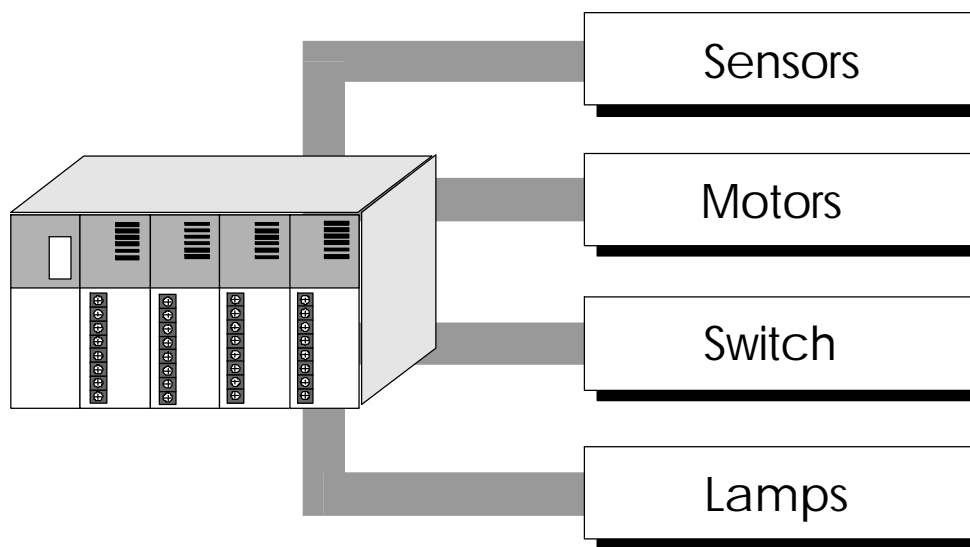
It is performed not only interpretation and execution of Ladder program but also Input/output-refresh and self-inspection during SCANTIME. TinyPLC has fixed-5mS SCANTIME regardless program length. It is much more efficient to manage time compared to variable SCANTIME. (You can change SCAN TIME as per types of machine.)



You don't have to worry too much about SCAN TIME in using Relay Ladder Program, but keep in mind that program is basically placed on SCAN TIME structure. This is the biggest difference from sequential technical language such as C or Assembly.

2.2. What does PLC do?

Just like other processors (One-chip microcontroller, One-board computer, PC), PLC controls and manages and operate various peripheral devices. It is distinguished from other processors in the way that it, as a substitute of Relay switchboard used in equipment and production automation, can effectively control peripheral devices on each production line. For instance, it controls conveyers and enables each proximity sensor and temperature sensor to move in a constant sequence after they contact valves and motors.

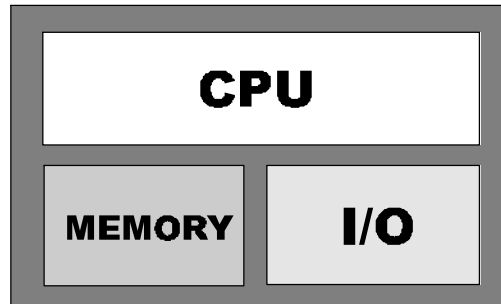


While PLC substituted for Relay switchboard had a various functions beside the most fundamental functions (input/output control, timer, counter function) in the past, it is in the spotlight as a total controller for factory automation at present. PLC has a great influence, thus no one possibly mentions automation without considering PLC. The following table summarizes to what PLC is applied;

Automated warehouse, return line, industrial Robot, machine tools, pumps, thermal controller, compressor, loader in inspection/manufacturing line of semiconductor, industrial generator, production line such as conveyer and reprocessing system etc.

2.3. General structure of PLC

PLC consists of three fundamental structures;



CPU, which play a role in human's brain, reads mnemonic type of data/instruction from the memory then interprets and runs it. I/O part accept external input and proceed output as if human's eyes, ears, hands and legs. Memory part is in charge of memorization and stores all the information incurred during operation and ladder program drawn by user.

There are several types of memory; RAM, ROM, EEPROM, FLASH ROM etc.

- RAM (Random Access Memory) can freely read and write.
- ROM (Read Only Memory) can only read.
- EEPROM can be clear electrically and be changed its contents during running and stays online without electric power. Otherwise, RAM can not conserve its contents without electric power.
- FLASH ROM has a similar feature as EEPROM in the way of running but is different from EEPROM in the way of manufacturing.

Ladder Diagram drawn by user is translated into performable codes and stored in FLASH ROM. All information incurred during operation is stored in RAM. Some information needed to be conserved after power-off is stored separately in EEPROM (KEEP area).

I/O part of TinyPLC is composed of I/O port with the TTL level format (*TTL level recognizes 0V as 0 and 5V as 1). Therefore, you can turn ON/OFF a great amount of load by adding Relay or Photo coupler at I/O ports. COMFILE supplies this I/O extension part as a product named BASEBOARD.

2.4. Type of Relay

Relays in PLC mean not only components connecting outer-part but also internal relays not providing external output. The followings are descriptions for Relays dealt with on TinyPLC. Since Relays are classified with alphabet initial as P, M, K, T, C, you also had better learn them.

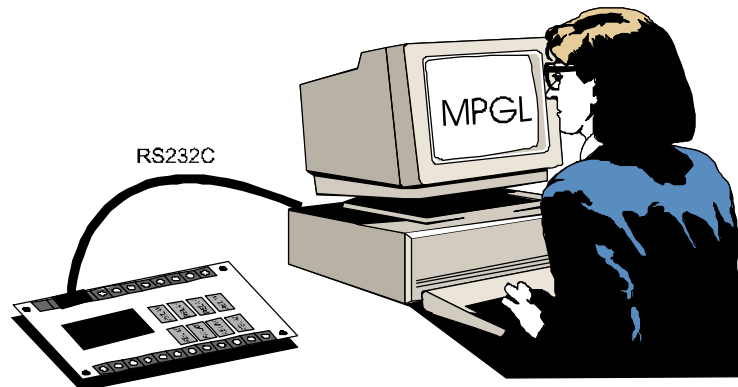
Initial	Name	Functions
P	Input/output Relay	This Relay is directly connected to external Relays or input parts. An input contact point relays the HIGH/LOW state or an external input port. An output contact point is connected to external valve or motor and influence their states when input relay changes. There are totally 128 points of P area on TinyPLC including spares. Some part of P area is used for connection with external I/O. The other part is used as REMOTE I/O etc. If REMOTE I/O is not used, it could be used as an Internal Relay.
M	Internal Relay	This Relay operates only within program. It can not execute input /output directly outside, but play a supporting role in transmitting information etc (M area is only place where can be used for input of DF, DFN.).
K	Keep Relay	This Relay has a same role as M relay, but is different in conserving its data without electric power. Data needed to keep during power failure is to be stored here.
F	Special Relay	This Relay displays state of internal operation, result of operation and various information of time in PLC. For instance, some relay repeat ON/OFF every one second.
T	Timer	This Relay controls time. It has ON timer and OFF timer, and computes time in 10mS or 100mS unit and its contacting point is ON when it reaches a settled number. Please refer to instruction for details.

C	Counter	This Relay counts the number of pulse. It has Up timer and Down timer, counts the edge part of input pulse. Its contact point turns on when it reaches a settled number. Please refer to Chapter 3 instruction for details (Pulses faster than SCAN TIME should be measured by high-speed counter.).
D	Data	This Relay stores data used for operation and deals with data in word (16bit) unit. There are several instructions on TinyPLC, which deal with Data.
S	Step Controller Relay	This Relay is specially used for sequential works. Since PLC runs repeatedly during SCAN TIME, sequential works, which can be solved quite simply by other program languages, can be difficult to work sometimes. In this case, you can solve the problem easily by using Step Controller Relay.
CH	LCD Display	This is for display buffer of LCD. ASCII code data written in CH area will appear on LCD. When the CH area is full with blank (ASCII code 20H), LCD is cleared.
G	SGN Display	This is for buffer of seven-segment display. ASCII code written in G area will appear on SGN (Seven-segment compile Module).
AD	AD Conversion result storage	The result of A/D conversion is recorded in AD area. TinyPLC do A/D converting automatically without user's instruction and the result is recorded in AD area. Users can get A/D results simply by referring to this area.
CNT	High-speed counter result storage	The result of high-speed counter is recorded in CNT area. Without any instruction, users can use the result of high-speed counter simply by referring to CNT area.

*** All S areas except KEEP area is reset to 0 on power-on.**

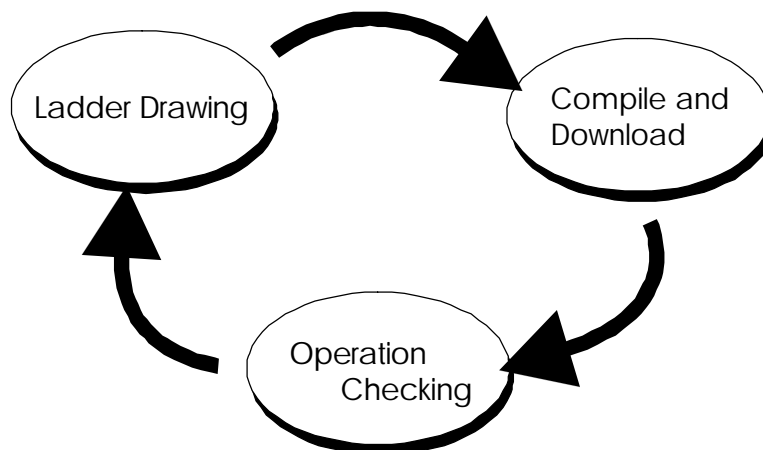
2.5. Ladder INPUT and RUN

In order to use PLC, you need a Ladder Program input device, download cable (RS232C cable) and PLC mail frame.



TinyPLC and BASE-BOARD

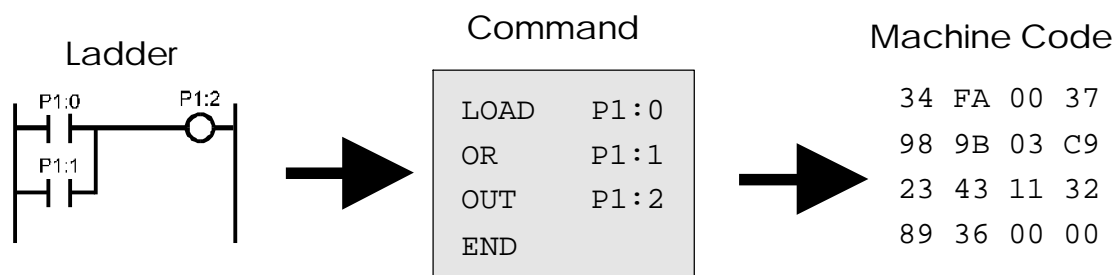
There are many Ladder input devices and some of them use Handy Loader. However, PC is most common device these days. Because Ladder can be described with graphics on PC and that data backup is relatively easy. Ladder diagrams drawn on PC can be downloaded or uploaded through RS232C cable. Downloaded programs are ready to operate in the Stand-Alone state and input programs run whenever power is turned on because they are always saved even after power is off.



Development of program with TinyPLC proceeds repeating three steps, which are **Ladder drawing**, **Compile/download** and **Operation checking**.

Program translation process on TinyPLC

TinyPLC take three steps of translation process in order to download Ladder program. Since TinyPLC is a compile-typed PLC, drawn Ladders are translated into nimonics and then changed into assembler codes by compiling (MPGL2 program operates automatically by this point, users don't have to worry about it). Then, Ladders are downloaded into Tiny PLC module through RS232C cable.

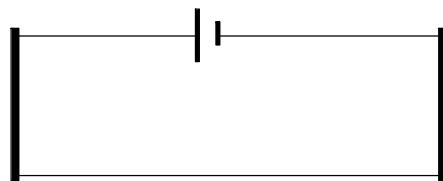


2.6. Basic of Ladder symbols

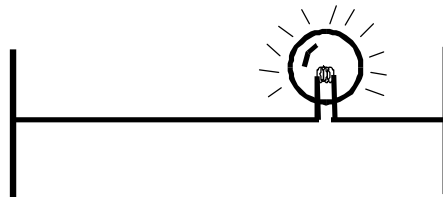
Ladder symbol has very simple structure. Ladder needs two basic bus lines.



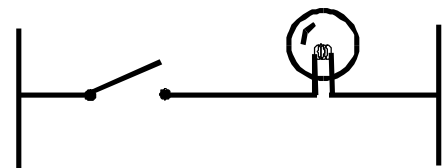
These bus lines are electric lines.



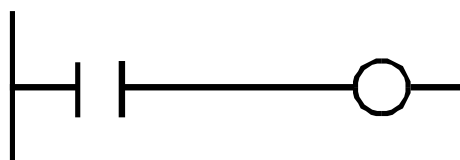
When a lamp is connected to those electric lines, the light is on.



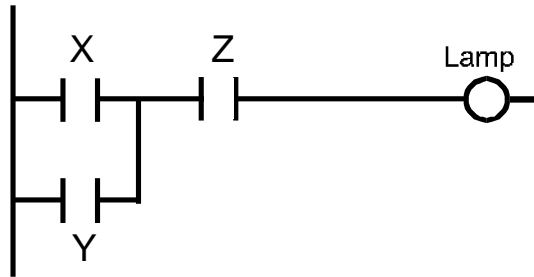
When a switch is connected here, it is described like the following diagram. When the switch is ON, the light is on and when the switch is OFF, the light is off.



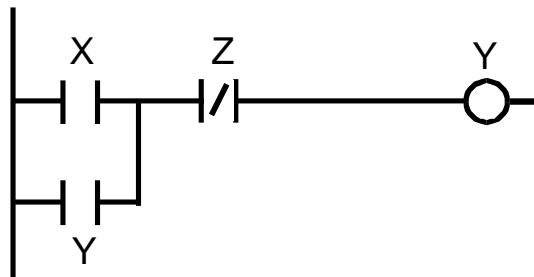
Relay Ladder Diagram describes above circuit with symbols as below.



When several input symbols are organized as below, logic, such as AND and OR, can be realized. Combination of X and Z, as an AND condition, and combination X and Y, as an OR condition, are organized in one circuit. When X and Z are ON, the lamp is on. Or when Y and Z are ON, the lamp is on.



Though Relay Ladder Circuit seems to be simple, you can make various applications. The following is a Self-Sustenance circuit. “Self-sustenance” circuit is a latch circuit, which can remember its state until new signal inputs.



In this circuit, when X is pressed, Y is on. Because output Y is connected to OR condition in inputting, once Y is on, Y sustains ON. When Z is pressed, ON state unlatches. (Z is a contact point switch, which always stays ON, but changes into OFF when input comes in.)

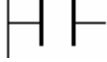
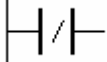
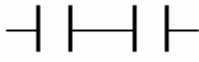

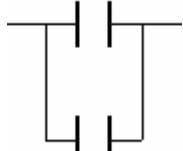
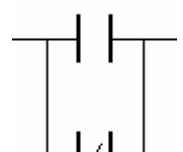
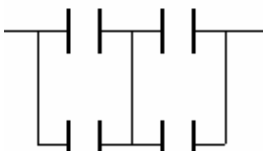
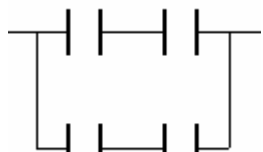
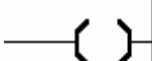


Chapter 3









Basic Instruction

This chapter describes basic instructions of TinyPLC in detail.





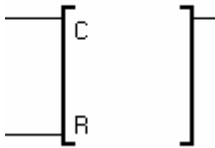
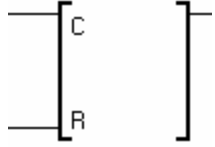
3.1. Introduction Instructions

3.1.1. Basic Instruction

Name	Instruction	Ladder Symbol	Description	Page
Load	LOAD		Operation starts from A contact (Normal Open)	48
Load Not	LOADN		Operation starts from B contact (Normal Close)	48
And	AND		A contact series connection	49
And Not	ANDN		B contact series connection	49
Or	OR		A contact parallel connection	50
Or Not	ORN		B contact parallel connection	50
And Stack	ANDS		AND connection between blocks	56
Or Stack	ORS		OR connection between blocks	57
Output	OUT		Output result of operation	51
Not	NOT		Reverse result of operation	54
Step Sequential Set	STEPSET	STEPSET S0:2 	Output step controller (sequential control)	64

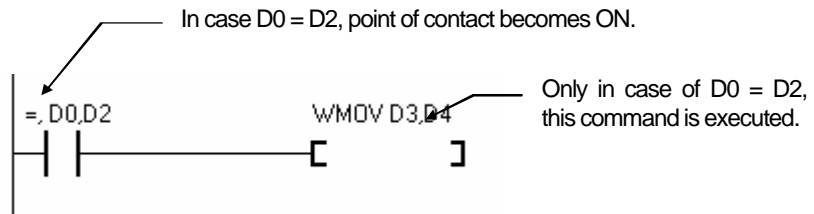
Step Output	STEPOUT	STEPOUT S0:1 	Output step controller (LIFO)	65
Master Control Set	MCS	MCS 0 	Start master control relay	61
Master Control Clear	MCSCLR	MCSCLR 0 	Terminate master control relay	61
Differential	DF		Output "ON" during 1 scan time if input condition rise. (differential input)	60
Differential Not	DFN		Output "OFF" during 1 scan time if input condition pull down. (differential reverse input)	60
Set Output	SETOUT	SETOUT P0:0 	Set output of contact point ON	58
Reset Output	RSTOUT	RSTOUT P0:1 	Reset output of contact point OFF	59
Save Status	SAVES	No ladder	SAVE present operation state	-
Read Status	RDS	No ladder	Read saved state SAVES and RDS are used at ladder branch	-
End	END	END 	End of program	47

3.1.2. TIMER/COUNTER

Name	Instruction	Ladder Symbol	Description	Page
On Timer (10mS)	TON	<p style="text-align: center;">TON T1,100</p> 	<p>0.01sec ON delay timer (maximum 327.67sec)</p> <p>With input, timer starts to operate. Without input, timer becomes reset. In case timer value reaches set point, point of contact becomes On.</p>	58
Off Timer (10mS)	TOFF	<p style="text-align: center;">TOFF T1,100</p> 	<p>0.01 sec OFF delay timer (maximum 327.67 sec)</p> <p>With input, point of contact becomes On. In case input is cut, point of contact is not directly Off. And after set time passed, it becomes Off</p>	59
On Timer (100mS)	TAON	<p style="text-align: center;">TAON T1,100</p> 	<p>0.1 sec ON delay timer (maximum 3276.7 sec)</p> <p>Operation is same with TON command</p>	58
Off Timer (100mS)	TAOFF	<p style="text-align: center;">TAOFF T1,100</p> 	<p>0.1 sec OFF delay timer (maximum 3276.7 sec)</p> <p>Operation is same with TOFF command</p>	59
Up Counter	CTU	<p style="text-align: center;">CTU C1,100</p> 	<p>Up counter (able to count to maximum 65535)</p> <p>With input it augments by 1, and then it reaches to set point, point of contact becomes On. With reset input, counter becomes 0.</p>	60
Down Counter	CTD	<p style="text-align: center;">CTD C1,100</p> 	<p>Down counter (able to count from maximum 65535)</p> <p>With input it decays by1, and then it reaches to 0, point of contact is set to set point.</p>	61

3.1.3. Comparing Command

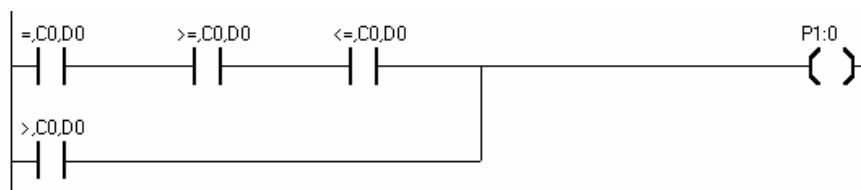
Comparing command can be implemented in Tiny PLC to use command, which start with a sign of inequality, to input symbol as shown below.



Classification	Instruction Format	Description	Page
16 Bit comparing command	=, S1, S2	In case S1 = S2, point of contact becomes ON. It compares 1 word (16 Bit) value.	63
	>, S1, S2	In case S1 > S2, point of contact becomes ON.	63
	<, S1, S2	In case S1 < S2, point of contact becomes ON.	63
	<=, S1, S2	In case S1 <= S2, point of contact becomes ON.	63
	>=, S1, S2	In case S1 >= S2, point of contact becomes ON.	63
	<>, S1, S2	In case S1 ? S2, point of contact becomes ON.	63
32 Bit comparing command	D=, S1, S2	In case S1 = S2, point of contact becomes ON. It compares double word (32 Bit) value.	65
	D>, S1, S2	In case S1 > S2, point of contact becomes ON.	65
	D<, S1, S2	In case S1 < S2, point of contact becomes ON.	65
	D<=, S1, S2	In case S1 <= S2, point of contact becomes ON.	65
	D>=, S1, S2	In case S1 >= S2, point of contact becomes ON.	65
	D<>, S1, S2	In case S1 ? S2, point of contact becomes ON.	65

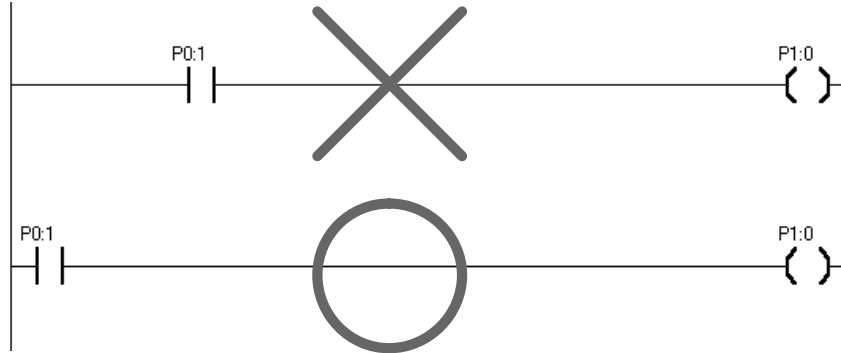
***S1, S2 mean argument1, argument 2.

AND and OR interface is free to be used with Comparing command, same as general point of contact input.

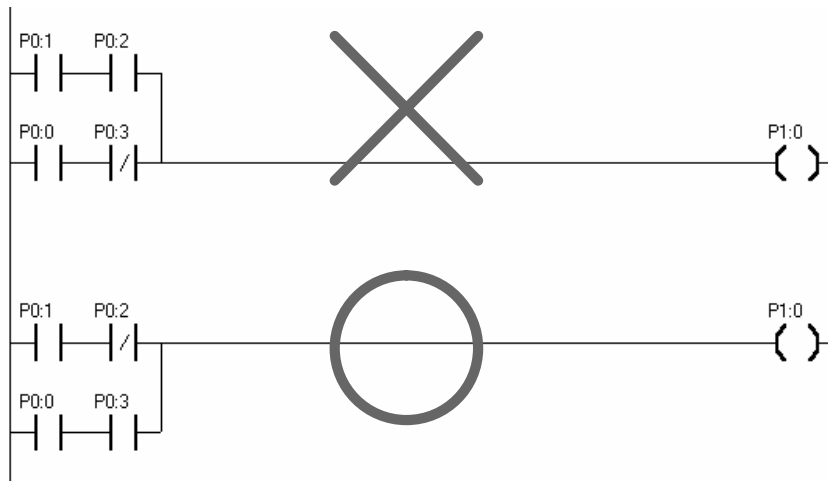


3.1.5. Bad Ladder Input

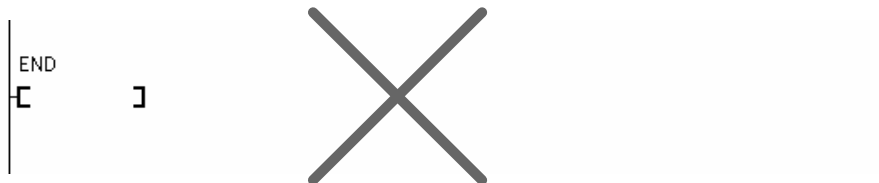
LOAD command should start from the first column (the very side of mother line).



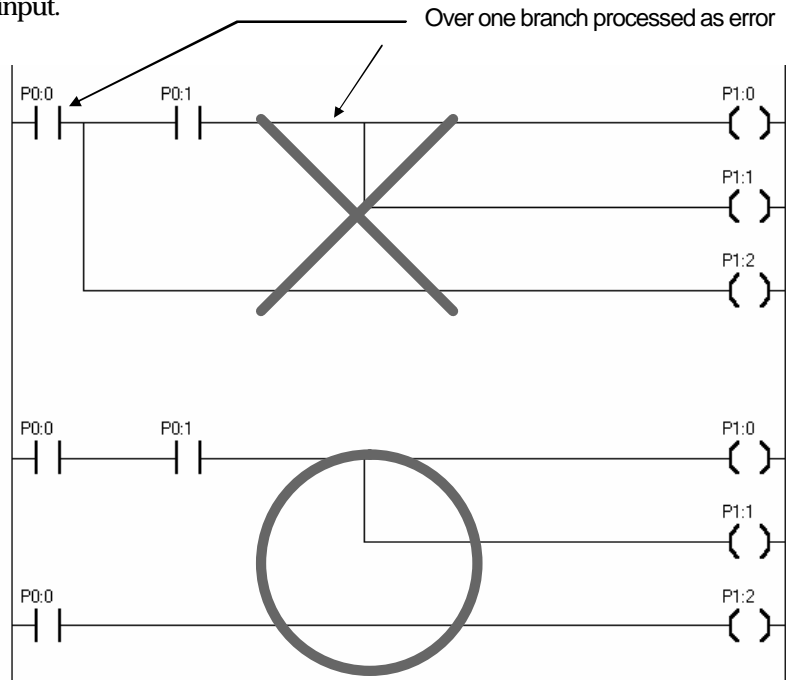
Ladder should spread out as below.



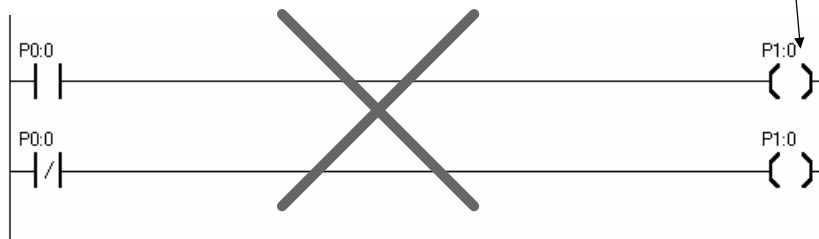
Output symbol could not be located on the first column (the very side of left mother line).



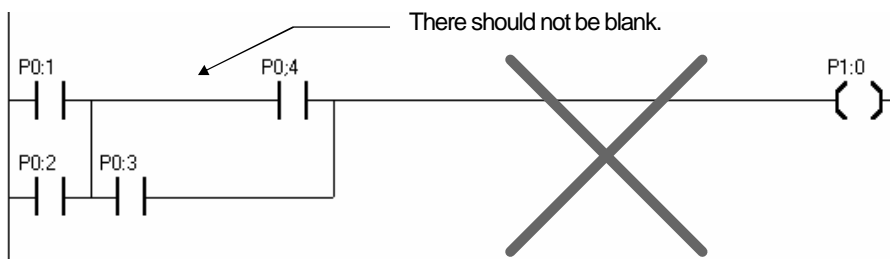
You cannot interface from one line to various output point of contact. In this case, you should correct as below and input.



There should not occur collision of same output. There occurs contradiction the same output have different value



If you insert unnecessary blank (crossway), translation error occurs. (You should make it close without blank.)



3.2. Description of Instructions

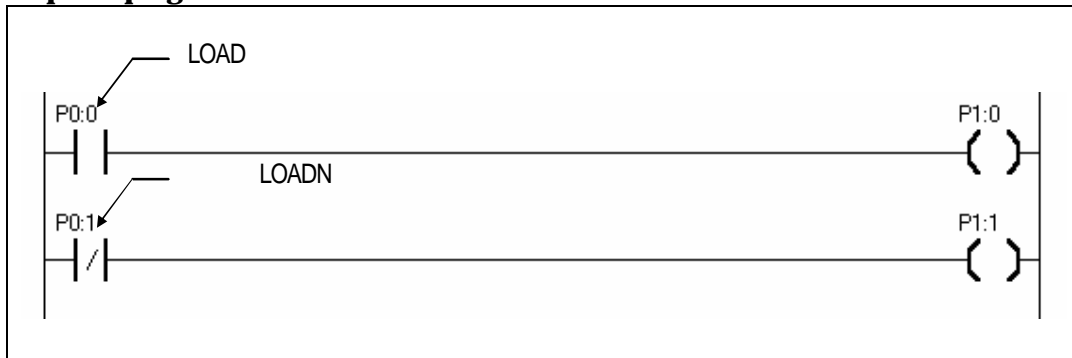
LOAD, LOADN

Start A/B contact

Summary

LOAD is start of A (Normally open) contact; LOADN is start command of B (Normally closed) contact.

Example of program



Operands

Command	Relay					counte	time	Etc.			constant
	P	M	F	K	S	r	r	AD	CH	G	
LOAD	O	O	O	O	O	O	O				
LOADN	O	O	O	O	O	O	O				

Descriptions in detail

If A contact P0:0 become ON, Output P1:0 become ON,.

If B contact become ON, Output P1:1 become ON,.

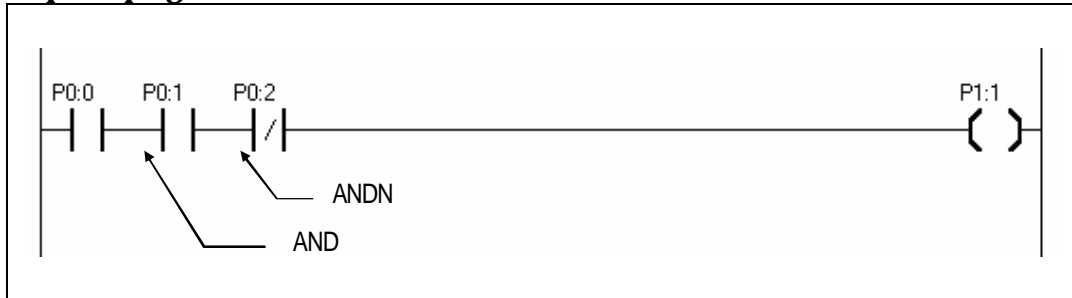
(P0:0 and P0:1for input should be set input on MPGL2 program. Same way, P1:0 and P1:1 for output should be set output. Incase input/output set is wrong, it does not work right.)

AND, ANDN series interface of A/B contact

Summary

AND is series interface of A contact, ANDN is series interface command of B contact.

Example of program



Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
AND	O	O	O	O	O	O	O				
ANDN	O	O	O	O	O	O	O				

Description

If P0:0 and P0:1 become ON, output P1:1 become ON.

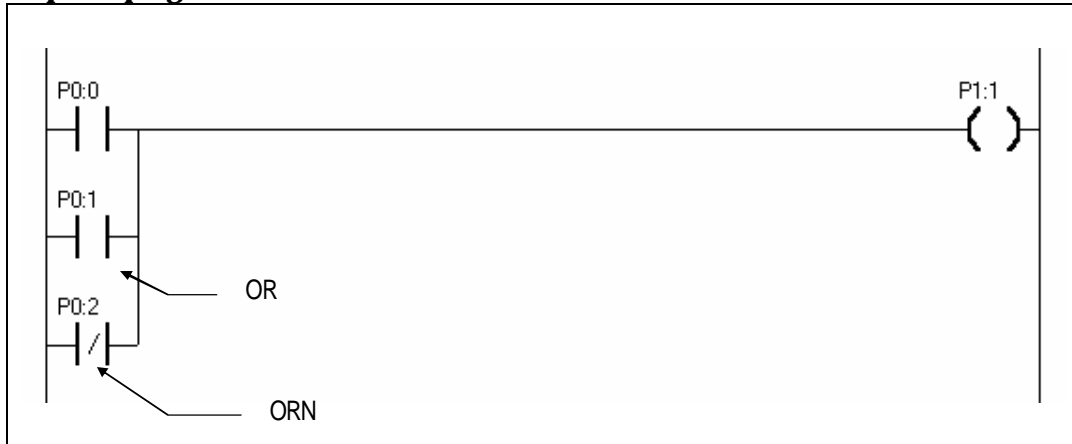
If B contact P0:2 become ON, output P1:1 become OFF.

OR, ORN interface of A/B contact

Summary

OR is parallel interface of A contact, ORN is parallel interface command of B contact.

Example of program



Operands

Command	Relay					Counter C	Timer T	Etc.			Constant
	P	M	F	K	S			AD	CH	G	
OR	O	O	O	O	O	O	O				
ORN	O	O	O	O	O	O	O				

Description

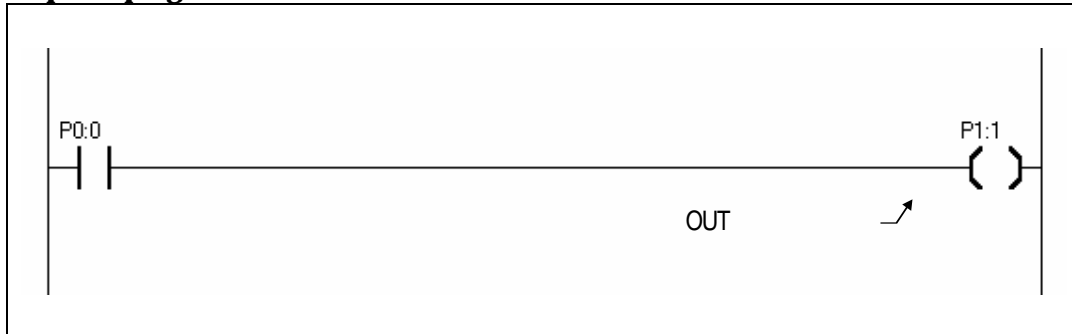
If P0:0 or P0:1 or P0:2 become ON, output P1:1 become ON.

OUT output operation result

Summary

Do output the result of operation to contact point.

Example of program



Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
OUT	O	O		O							

Description

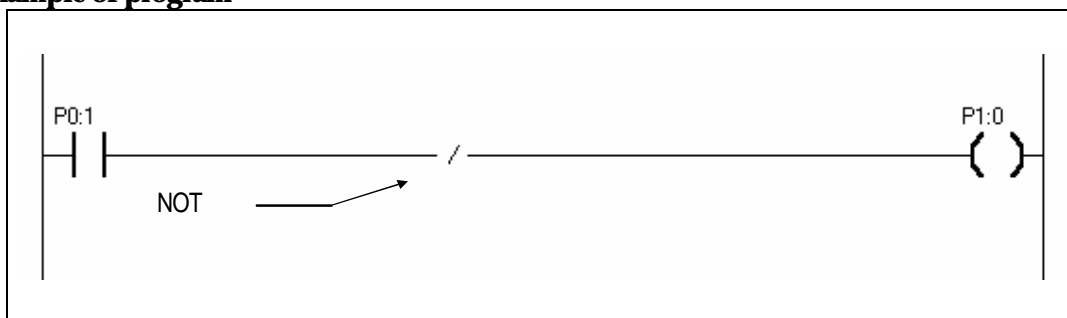
If P0:0 become ON, output P1:1 become ON.

NOT output of operation result

Summary

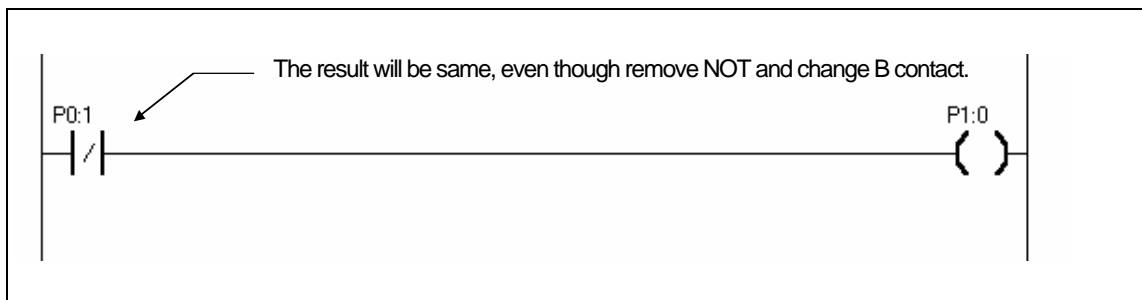
Reverse operation result before NOT.

Example of program



Description

If P0:1 become ON, output P1:1 become OFF. The upper program has the same effect as below.



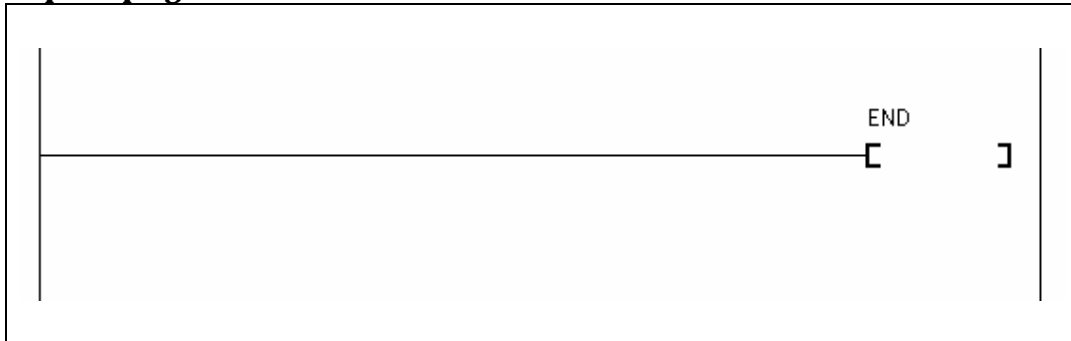
END

end of program

Summary

It shows the end of the entire program. (It should be always located on the end of program.)

Example of program



Description

MPGL2 translate as far as END instruction and deposit. Please be careful, all of the instructions which made after END are ignored.

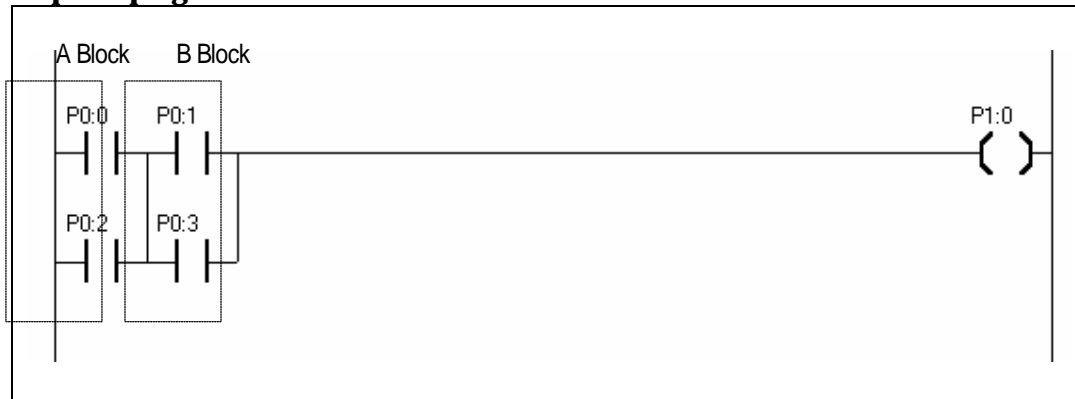
ANDS

AND interface between block

Summary

It is series interface command between block.

Example of program



Description

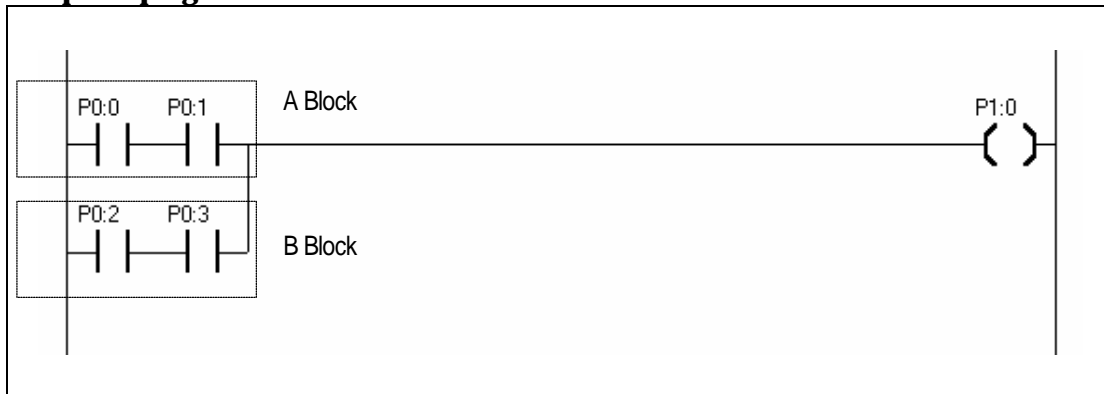
When block A and block B are ON, output P1:0 become ON.

ORS OR interface between blocks

Summary

It is parallel interface command, between block.

Example of program



Description

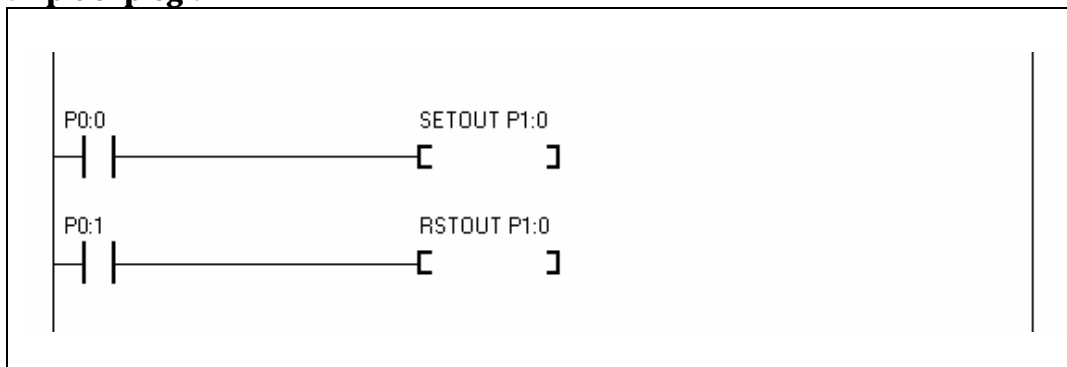
When block A or block B is ON, output P1:0 become ON.

SETOUT maintain output ON

Summary

Maintain output point of contact ON.

Example of program



Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
SETOUT	O	O		O							

Description

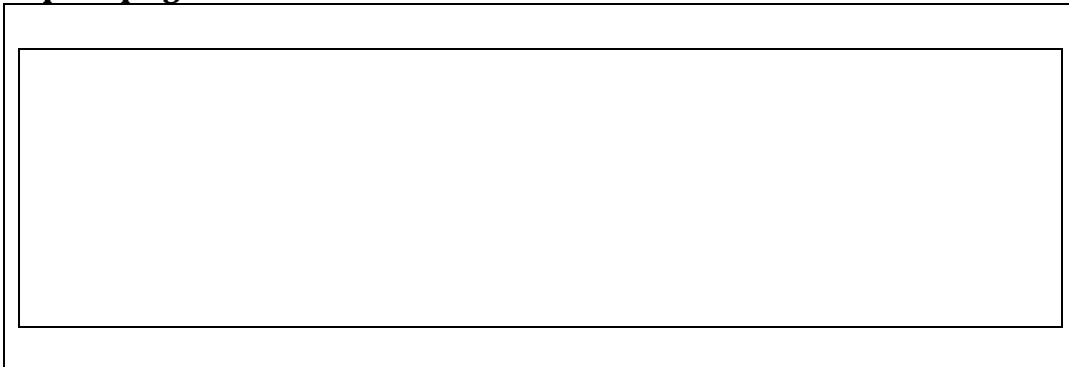
If P0:0 become ON, P1:0 output point of contact maintain ON.

RSTOUT maintain output OFF

Summary

Maintain output point of contact OFF.

Example of program



Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
RSTOUT	O	O		O							

Description

If P0:1 become ON, maintain P1:0 output point of contact OFF.

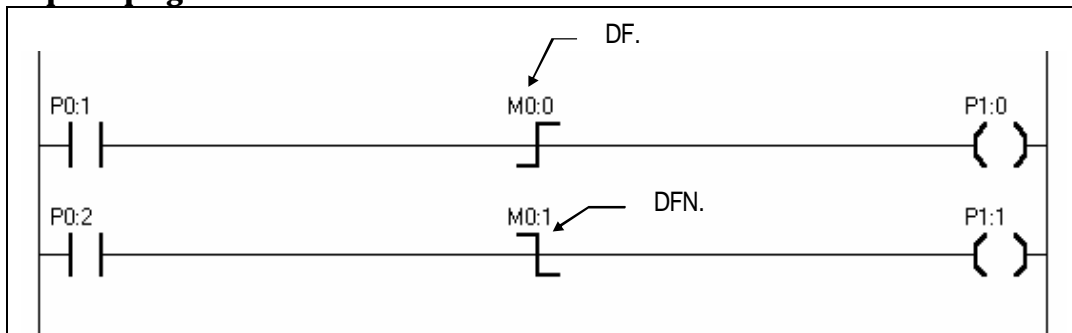
DF, DFN differential input

Summary

DF : If rise edge of input condition (Off-->On) is verified, output point of condition become ON for 1 scan time.

DFN : If pull-down edge of input condition (On-->Off) is verified, output point of condition become ON for 1 scan time.

Example of program



Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
DF		O									
DFN		O									

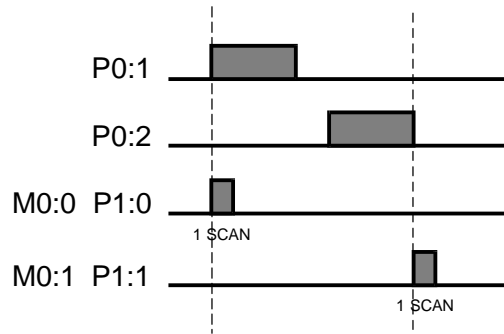
*** Caution: DF and DFN command should be used only at M .**

Description

The moment P0:1 become ON, P1:0 become ON for 1 scan time.

The moment P0:2 become OFF, P1:1 become ON for 1 scan time.

Time Chart

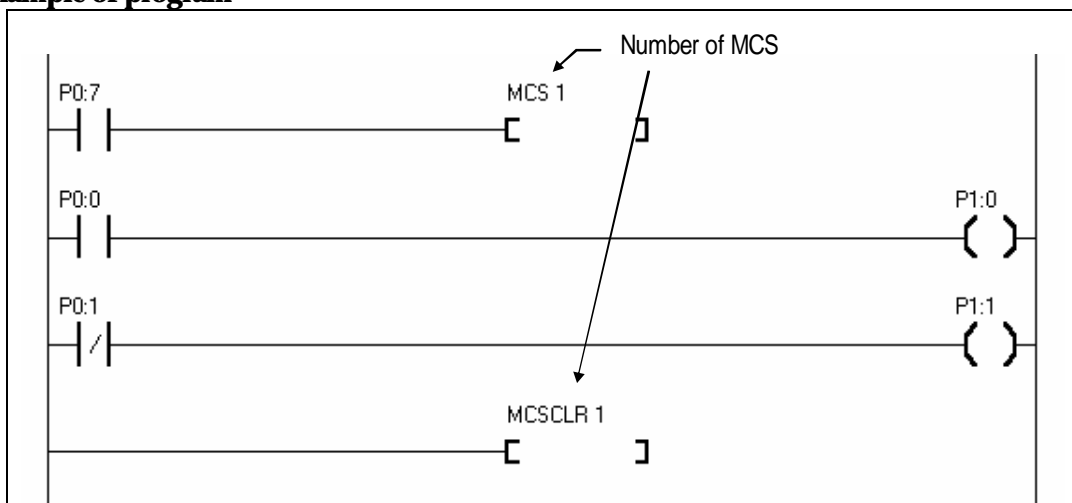


MCS, MCSCLR master control relay

Summary

If input condition of MCS is ON, it executes until MCSCLR which have same number but is OFF, it does not execute. In case it does not execute, all outputs in the range of MCS ~ MCSCLR become OFF.

Example of program

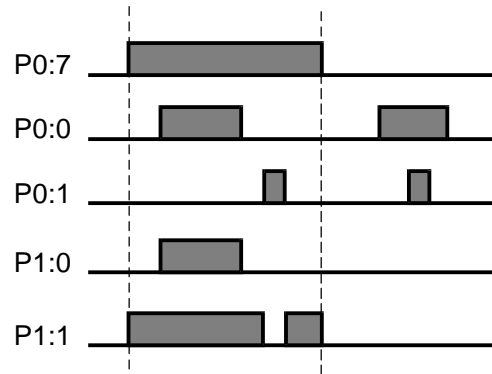


Description

If P0:7 become ON, ladder in the range of MCS 1 ~ MCSCLR 1 is executed. If P0:7 become OFF, ladder in the range of MCS 1 ~ MCSCLR 1 is not executed. Also output P1:0 and P1:1 become OFF.

MCS number is available to use from 0 to 7. 0 have the highest priority and 7 have the lowest priority. Therefore, if you release MCS the highest priority, the rest of MCS are also released.

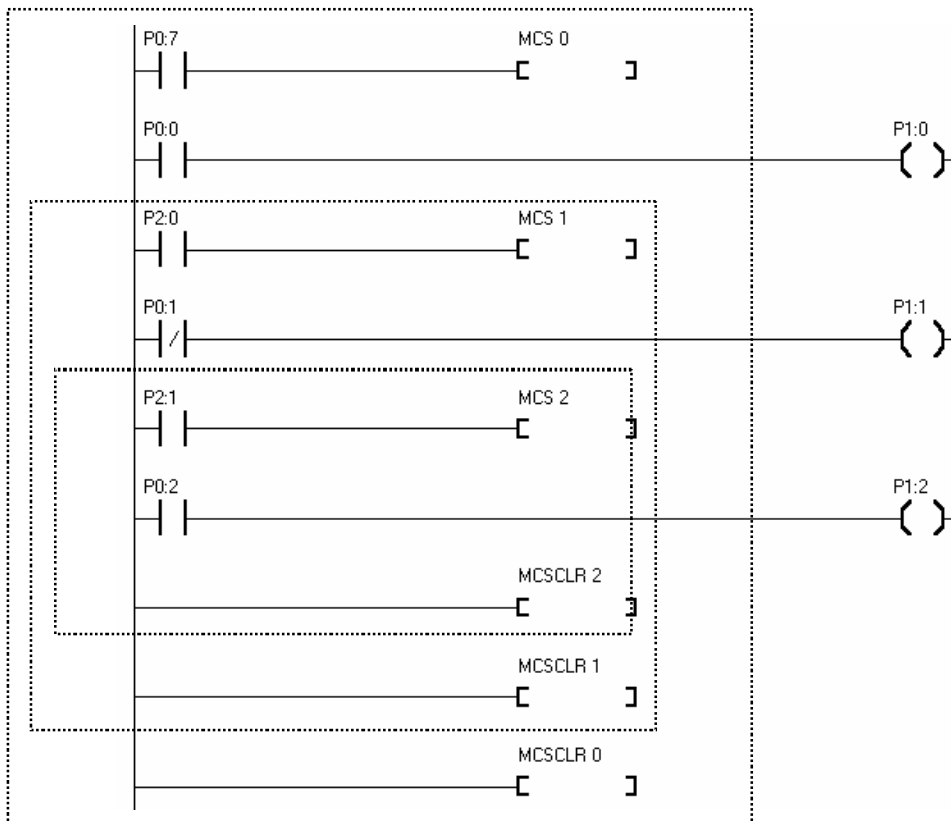
Time Chart



The table below explains instructions influenced in MCS loop.

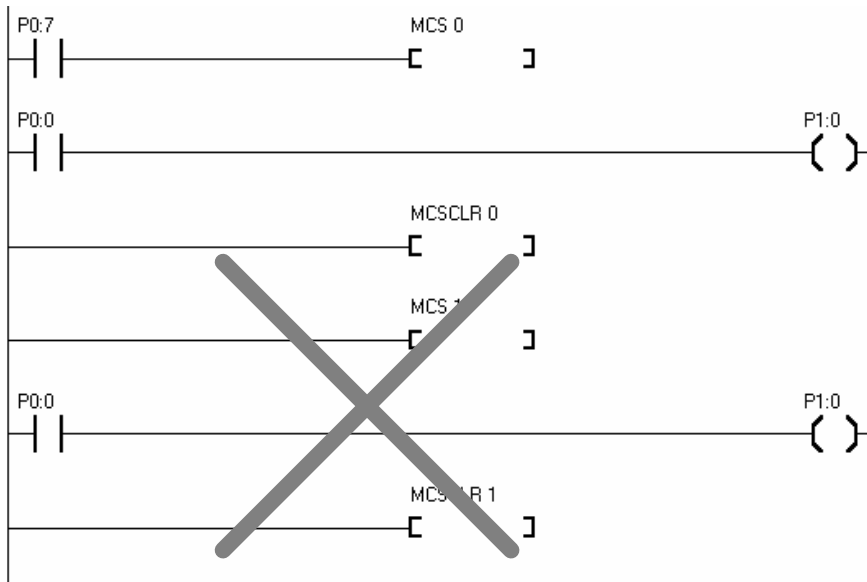
Instruction	MCS is ON condition	MCS is OFF condition
OUT	Normal execution	Unconditional OFF
SETOUT	Normal execution	Maintain constantly the state before MCS become OFF
RSTOUT	Normal execution	Maintain constantly the state before MCS become OFF
Timer instruction (TON, TOFF..)	Normal execution	Reset initial value
Counter instruction (CTU, CTD)	Normal execution	Maintain constantly the state before MCS become OFF
The rest of instructions	Normal execution	Not execute

The nesting of MCS, MCSCLR instruction examples are described as below. (Nesting level is possible until maximum 8 level. Certainly, do layout lower number first and then layout higher number inside.)

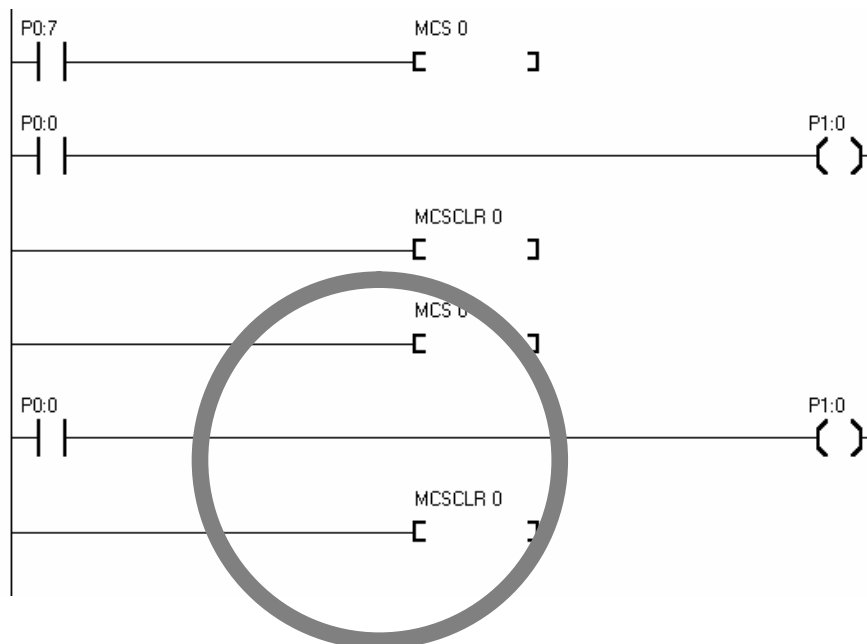


Caution

When you do not nest level, you should keep using MCS0.



Above diagram must be corrected as follows.

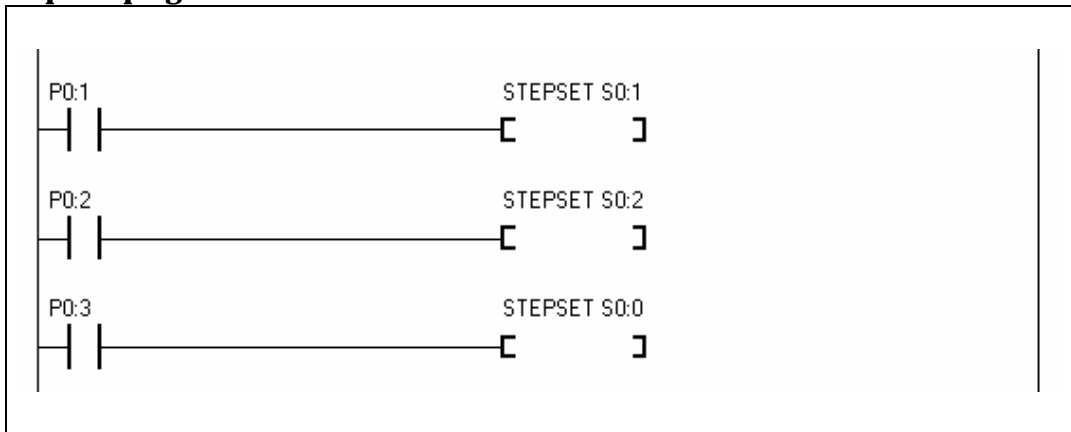


STEPSET step control (sequential control)

Summary

In case the previous number in same group is ON, present number becomes ON and previous number become OFF. (It is called sequential control because it becomes ON in sequential order.) From 0 to 255 steps is available.

Example of program



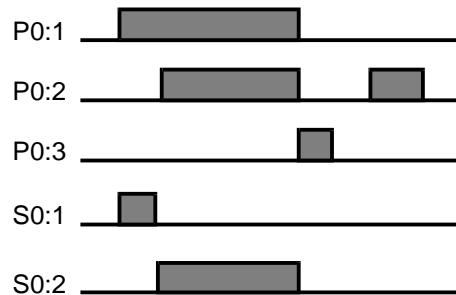
Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
STEPSET					O						

Description

If P0:2 become ON, 2 step of 0 group attempt to be ON. At that time, if 1 step of same group was ON, 1 step become OFF and 2 step become ON. In case P0:3 become ON, unconditionally it put back in 0 step. (0 step is used for reset.)

Time Chart

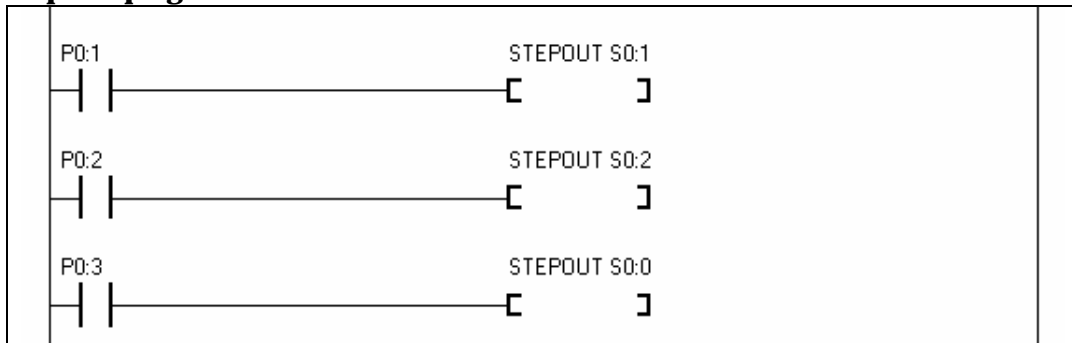


STEPOUT step control (LIFO)

Summary

Even though lots of input is processed in the same group, only the last step become ON and the rest steps become OFF. (The last step has the priority, and then it is called Last In First Out.) From 0 to 255 step is available.

Example of program



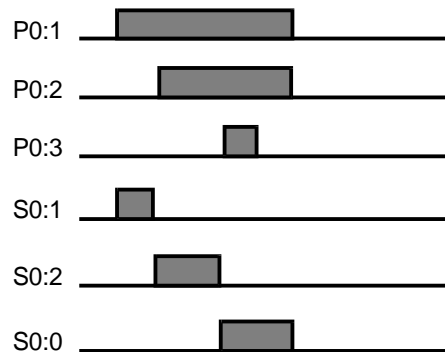
Operands

Command	Relay					Counter	Timer	Etc.			Constant
	P	M	F	K	S	C	T	AD	CH	G	
STEPOUT					O						

Description

If P0:1 become ON, 1step of) group become ON. After that, if P0:3 become ON, 0 step become ON. After that P0:2 become ON, 2 step become ON. Unconditionally, the last step become ON only, and the rest steps become OFF.

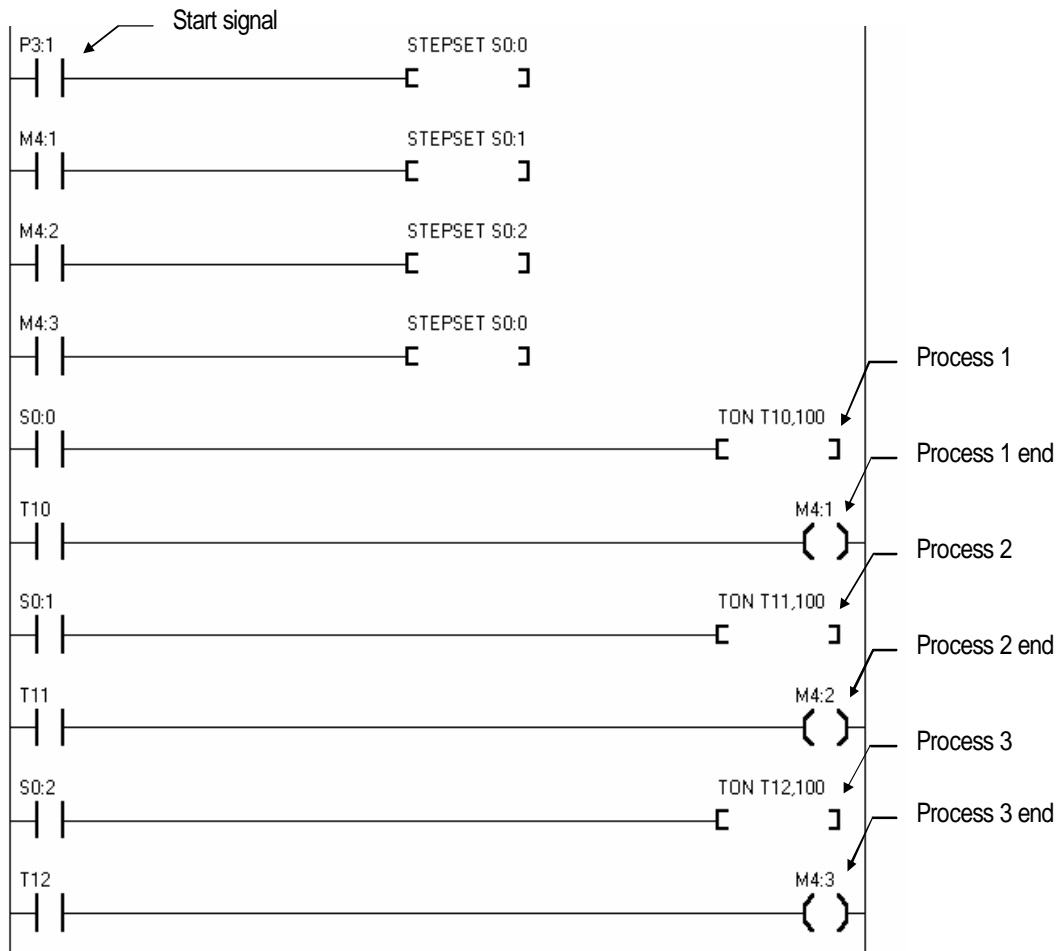
Time Chart



#1) Additional Description about Step Controller

C or assembly easily processes sequential control, but sometimes it cannot be implemented on PLC because of the characteristic of scan time. Then the mode so called, "step controller" is made from this cause. It is useful when it comes to process in regular sequence.

You can execute process in sequence as below, after process 1 is executed and then next step is ON, and after process 2 is executed and then the next step is ON. (The program as shown below use timer instead of process.)



Step relay has self-storage function. (Before other input, maintain present state.)

In one group only one output is ON. (same concept with interlock)

In case of sequential control; It is able to move back only one column.(possible to be ON, incase the previous number is ON)

Incase of LIFO ; Even though lots of input is processed, all is ignored and only the last one become ON.

TON, TAON

ON delay timer

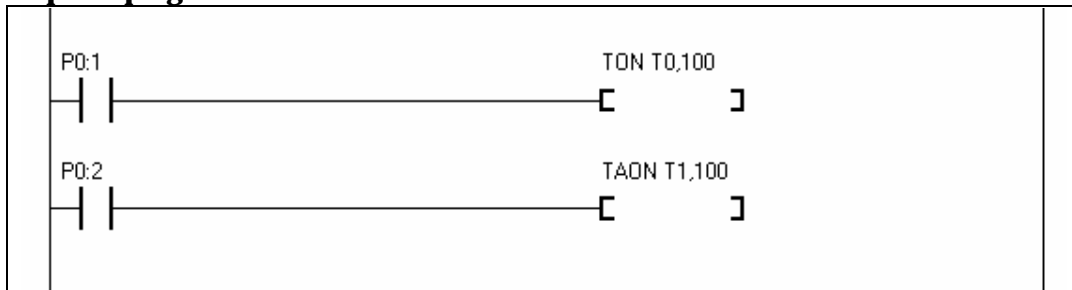
Usage : TON t, n

Summary

If input condition become ON, timer start to move, and become OFF, timer is reset. If timer value reaches set point, output point of contact becomes ON. There is two types of timer, which have different time unit.

Type of Timer	Unit	Maximum value
TON	0.01 sec	327.68 sec
TAON	0.1 sec	3276.8 sec

Example of program



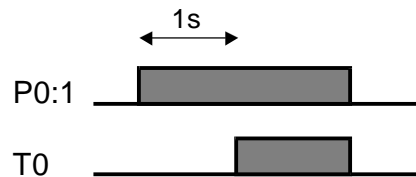
Operands

Operand	Relay					Counter	Timer	Data	Etc.			Constant
	P	M	F	K	S	C	T	D	AD	CH	G	
t							O					
n (set point)								O				O

Description

After P0:1 become ON, if 1 sec go by, T0 point of contact become ON. After P0:2 become ON, if 10 sec go by, T1 point of contact become ON.

Time Chart



TOFF, TAOFF

OFF delay timer

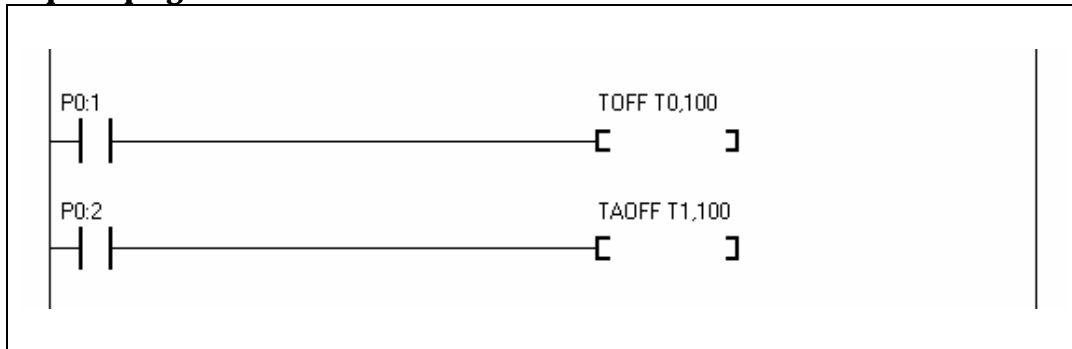
Usage : TOFF t, n

Summary

Right after input condition become ON, output point of contact becomes ON. After that, even if input becomes OFF, point of contact does not changed directly to OFF, but become OFF after set time passed. There are tow types of timer, which have different time unit.

Type of Timer	Units	Maximum value
TOFF	0.01 sec	327.68 sec
TAOFF	0.1 sec	3276.8 sec

Example of program



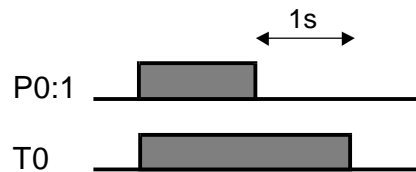
Operands

Operand	Relay					Counter	Timer	Data	Etc.			Constant
	P	M	F	K	S	C	T	D	AD	CH	G	
t (timer point of contact)							O					
n (set point)								O				O

Description

Right after P0:1 become ON, T0 become ON. After P0:1 become OFF, if 1 sec go by, T0 point of contact become OFF.

Time Chart



CTU

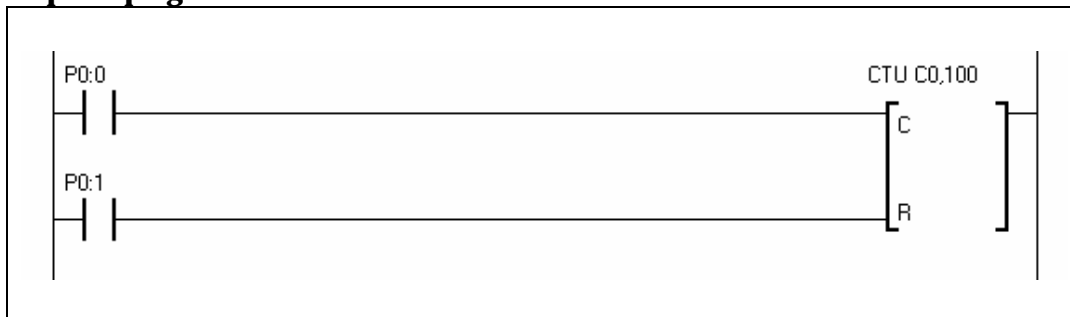
UP counter

Usage : CTU c, n

Summary

Each time of counter input, counter value augment by 1. If counter value is same as set point, output point of contact become ON. If input is also processed after point of contact become ON, counter is augmented continuously. (It is augmented until maximum 65535. Over 65535, counter starts again from 0, and state of point of contact is maintained.) If reset input is processed, counter value become 0.

Example of program



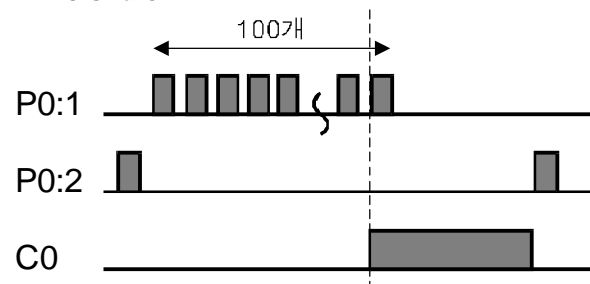
Operands

Operand	Relay					Counter	Timer	Data	Etc.			Constant
	P	M	F	K	S	C	T	D	AD	CH	G	
c (counter point of contact)						O						
n (set point)								O				O

Description

If P0:0 become ON 100 times, C0 point of contact become ON. If P0:1 point of contact become ON, counter is reset and point of contact become OFF.

Time Chart



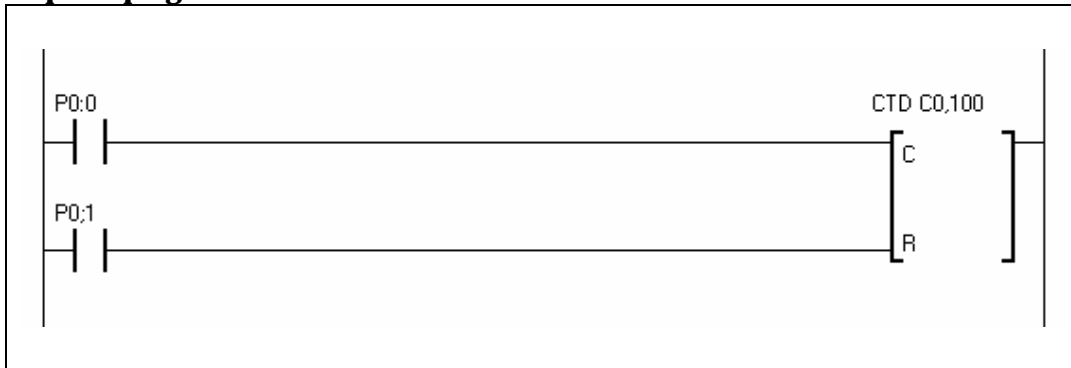
CTD DOWN counter

Usage : CTD c, n

Summary

Each time of counter input, counter value is decayed by 1. If counter value become 0, output point of contact become ON. If reset is input, counter value become set point. (It is set to the point of power-on.)

Example of program



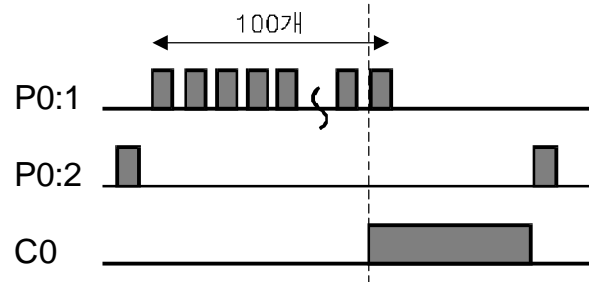
Operands

Operand						Counter	Timer	Data	Etc.			Constant
	P	M	F	K	S	C	T	D	AD	CH	G	
c (counter point of contact)						O						
n (set point)								O				O

Description

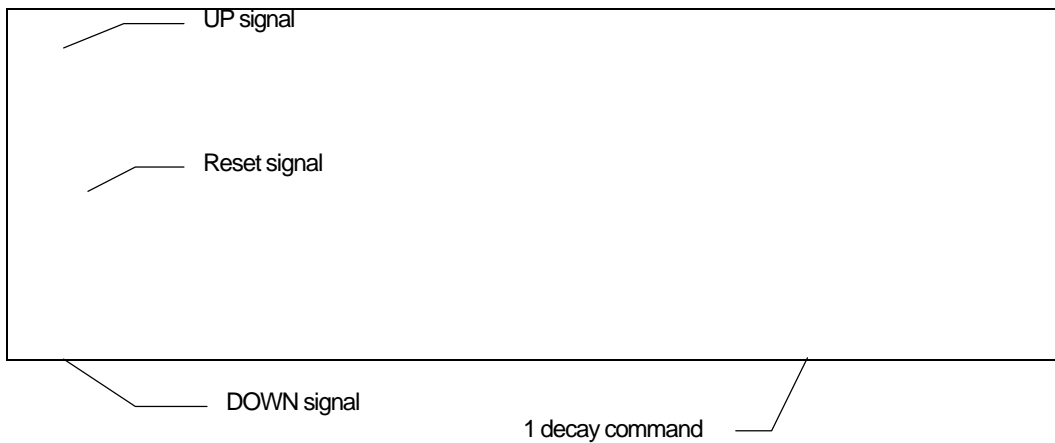
If P0:0 become ON 100 times, C0 point of contact become ON. If P0:1 point of contact become ON, 100 is set to counter and point of contact become OFF.

Time Chart



Implementation of UP/DOWN Counter

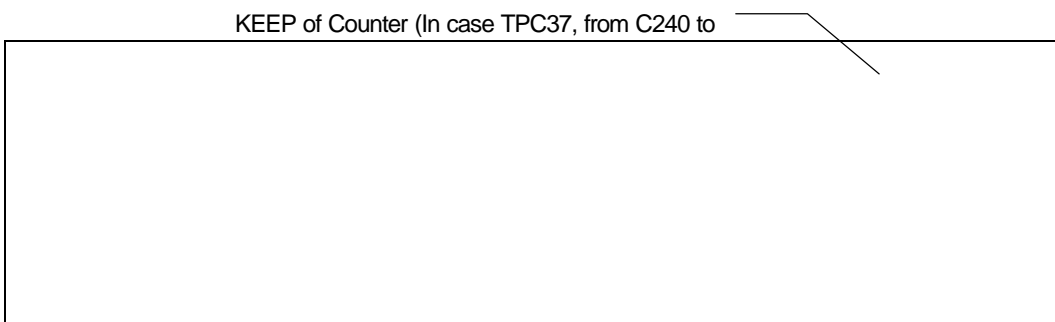
Even though TinyPLC do not have special UP/DOWN counter command, if you input as shown below, UP / DOWN counter is able to be implemented.



After differential P0:2 signal by DF command, execute WDEC, 1 decay command. Then counter CO value is decayed by 1. You can use P0:0 as augment input and, P0:2 as decay input.

KEEP of Timer and Counter

If you see memory map, part of TIMER and COUNT (slash mark) is KEEP, which keep state during the power failure. If you use this, timer (or counter) value under processing is not gone and kept during the power failure, and after power-on it process continuously.



Caution: If you use KEEP of counter, you should use CTU command for counting the consecutive value from power-off, because CTD command is initialized to corresponded value, in case of power-on.

=, <>, >, <, >=, <=

word comparing command

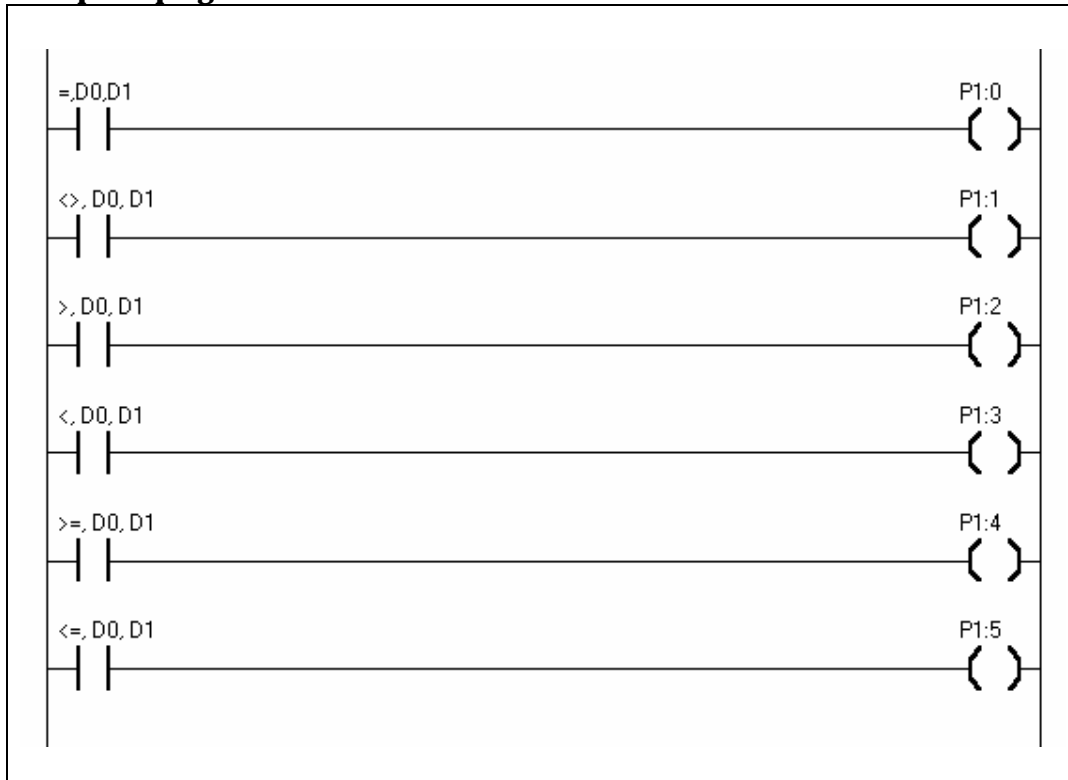
Usage : =, s1, s2

Summary

In case it satisfies condition with comparing two of word value, point of contact becomes ON. There are totally 6 of comparing command.

Comparing command	Execution
=, s1, s2	In case s1 = s2, point of contact becomes on.
<>, s1, s2	In case s1 ≠ s2, point of contact becomes on.
>, s1, s2	In case s1 > s2, point of contact becomes on..
<, s1, s2	In case s1 < s2, point of contact becomes on.
>=, s1, s2	In case s1 >= s2, point of contact becomes on.
<=, s1, s2	In case s1 <= s2, point of contact becomes on.

Example of program



Operands

Operand	Relay					Counter	Timer	Data	Etc.				Constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O

Description

In case $D0 = D1$, P1:0 become ON.

In case $D0 ? D1$, P1:1 become ON.

In case $D0 > D1$, P1:2 become ON. In case $D0 < D1$, P1:3 become ON. In case $D0 \geq D1$, P1:4 become ON.

In case $D0 \leq D1$, P1:5 become ON.

It is possible to use AND, OR interface as shown below.



In case $C2 = 30$ and $C0 \geq 100$, P1:0 become ON. Also in case $D1 < 3$ and $C0 \geq 100$, P1:0 become ON.

D=, D<>, D>, D<, D>=, D<=	double word comparing command
Usage : D=, s1, s2	

Summary

In case it satisfies condition with comparing two of double word value, point of contact becomes ON. There are totally 6 of comparing command.

Comparing command	Execution
D=, s1, s2	In case s1 = s2, point of contact becomes on.
D<>, s1, s2	In case s1 ? s2, point of contact becomes on.
D>, s1, s2	In case s1 > s2, point of contact becomes on..
D<, s1, s2	In case s1 < s2, point of contact becomes on.
D>=, s1, s2	In case s1 >= s2, point of contact becomes on.
D<=, s1, s2	In case s1 <= s2, point of contact becomes on.

Operands

Operand	Relay					Counter	Timer	Data	Etc.				Constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O

Description

All of executions are the same with “word comparing command” which is explained before, except comparing double word (32 Bit value).

Chapter 4

High Level Instruction

This chapter describes high level instructions of TinyPLC in detail.



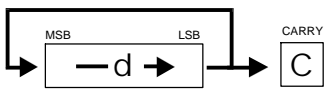

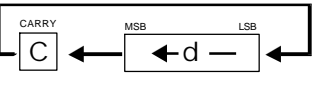
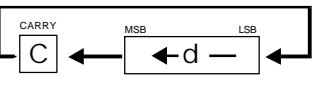
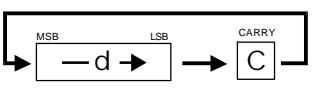
4.1. Introduction of High Level Instructions

“s” means source and “d” means destination.

“n” indicates number(or arbitrary number)

class	Function	Formation of instructions	Description	Pg
Move Instructions	Byte Move	MOVE s, d	Data move by 8bits (s) → (d)	76
	Byte Complement Move	CMOVE s, d	Data complement move by 8bits complement(s) → (d)	77
	Word Move	WMOV s, d	Data move by 16bits (s) → (d)	78
	Double Word Move	DWMOV s,d	Data move by 32bits (s, s+1) → (d, d+1) * For 32bits, high-level word is saved at s, low-level at s+1 (this is similar to other instructions)	79
	Word Complement Move	WCMOV s,d	Data complement move by 16bits complement(s) → (d)	81
	Double Word Complement Move	DWCMOV s,d	Data complement move by 32bits complement (s, s+1) → (d, d+1)	82
	Word Negative Move	WNEG s,d	Data negative move by 16bits Negative 2 (s) → (d)	83
	Double Word Negative Move	DWNEG s,d	Data negative move by 32bits Negative 2 (s, s+1) → (d, d+1)	84
	Word Exchange Move	WXCHG s,d	Data interchange by 16bits (s) ↔ (d)	85
	Double Word Exchange Move	DWXCHG s,d	Data interchange by 32bits (s, s+1) ↔ (d, d+1)	86
	Fill Move	FMOV s,d,n	Data filling instructions (s) → number of n data from (d)	87
Group Move	GMOV s,d,n	Group transmit instructions Number of n data from (s) → number of n data from (d)	88	
convert instructions	Word Binary to BCD code	WBCD s,d	Converting 16bit binary data into 4 Digit BCD code. (16bit Binary) → (4 digit BCD)	89
	Double Word Binary to BCD code	DWBCD s,d	Converting 32bit binary data into 8 Digit BCD code (32bit Binary) → (8 digit BCD)	90

	Word BCD code to Binary	WBIN s,d	Converting 4 Digit BCD code into 16bit binary data (4 digit BCD) → (16 bit Binary)	91
	Double Word BCD code to Binary	DWBIN s,d	Converting 8 Digit BCD code into 32bits binary data (8 digit BCD) → (32 bit Binary)	92
Increment & decrement instructions	Word Increment	WINC d	Increment 1 for 16bits (d) + 1 → (d)	93
	Double Word Increment	DWINC d	Increment 1 for 32 bits (d, d+1) + 1 → (d, d+1)	94
	Word Decrement	WDEC d	Decrement 1 for 16 bits (d) + 1 → (d)	95
	Double Word Decrement	DWDEC d	Decrement 1 for 32 bits (d, d+1) + 1 → (d, d+1)	96
Arithmetic Calculation	Word Addition	WADD s1, s2, d	Addition for 16bits (s1) + (s2) → (d)	97
	Double Word Addition	DWADD s1, s2, d	Addition for 32 bits (s1, s1+1) + (s2, s2+1) → (d, d+1)	98
	Word Subtraction	WSUB s1, s2, d	Subtraction for 16 bits (s1) - (s2) → (d)	99
	Double Word Subtraction	DWSUB s1, s2, d	Subtraction for 32 bits (s1, s1+1) - (s2, s2+1) → (d, d+1)	100
	Word Multiplication	WMUL s1, s2, d	Multiplication by 16bits (result is 32 bits) (s1) * (s2) → (d, d+1)	101
	Double Word Multiplication	DWMUL s1, s2, d	Multiplication by 32bits (result is 64 bits) (s1, s1+1) * (s2, s2+1) → (d, d+1, d+2, d+3)	102
	Word Division	WDIV s1, s2, d	Division for 16 bits (s1) / (s2) →(d) for share, (d+1) for surplus	103
	Double Word Division	DWDIV s1, s2, d	Division for 32 bits (s1, s1+1) / (s2, s2+1) → (d, d+1) for share, (d+2, d+3) for surplus	104
Logic	Word AND	WAND s1, s2, d	AND for 16bits (s1) AND (s2) → (d)	105
	Double Word AND	DWAND s1, s2, d	AND for 32bits (s1, s1+1) AND (s2, s2+1) → (d,	106

Operation			d+1)	
	Word OR	WOR s1, s2, d	OR for 16bits (s1) OR (s2) -> (d)	107
	Double Word OR	DWOR s1, s2, d	OR for 32bits (s1, s1+1) OR (s2, s2+1) -> (d, d+1)	108
	Word XOR	WXOR s1, s2, d	XOR for 16bits (s1) XOR (s2) -> (d)	109
	Double Word XOR	DWXOR s1, s2, d	XOR for 32bits (s1, s1+1) XOR (s2, s2+1) -> (d, d+1)	110
Rotate instructions	Word Rotate Left	WROL d		111
	Double Word Rotate Left	DWROL d		112
	Word Rotate Right	WROR d		113
	Double Word Rotate Right	DWROR d		114
	Word Rotate Left with Carry	WRCL d		115
	Double Word Rotate Left with Carry	DWRCL d		116
	Word Rotate Right with Carry	WRCR d		117

	Double Word Rotate Right with Carry	DWRCR d		118
Shift Instruction	Bit Shift Left	BSHL d,n	<p>Bit shift left instruction Shift (d) left by n bit</p>	119
	Bit Shift Right	BSHR d,n	<p>Bit shift right instruction Shift (d) right by n bit</p>	120
	Word Shift Left	WSHL s1, s2	<p>Word shift left instruction Shift 1 word to left from s1 to s2</p>	121
	Word Shift Right	WSHR s1, s2	<p>Word shift right instruction Shift 1 word to right from s1 to s2</p>	122
Display instructions	Convert for 7 SEGment	SEG s,d	<p>Convert instruction for 7 SEGment Converts 16bit values in s to 4bite data which can be indicated in 7segment</p>	123
	LCD area Clear	LCDCLS	<p>Insert SPACE(ASCII code 20H) in CH area (clears the LCD screen)</p>	125
	LCD data Output	LCDOUT port,mode	<p>Display output to LCD from data in CH area.</p>	126
	SGN data Output	SGNOUT port	<p>Display output to SGN from data in G area.</p>	130
	String Store	STRING d,"string"	<p>Store "string" character at d. "string" -> (d)</p>	133

	Word Bin to ASCII HEX format	HEX s,d	Convert 16bit binary value to hexadecimal ASCII code. (s) -> 4bytes from (d)	134
	Double Word Bin to ASCII HEX format	DHEX s,d	Convert 32bit binary value to hexadecimal ASCII code. (s, s+1) -> 4bytes from (d)	135
	Word Bin to ASCII DEC format	ASC s,d,n	Convert 16bit binary value to decimal ASCII code. (s, s+1) -> n bytes from (d)	136
	Double Word Bin to ASCII DEC format	DASC s,d,n	Convert 32bit binary value to decimal ASCII code. (s, s+1) -> n bytes from (d)	137
Jump	Goto, LABEL	GOTO label LABEL label	Jump instruction Label definition instruction	138
	Subroutine CALL, RET	CALLS label SBRT label RET	Call subroutine Define subroutine Return instruction	139
	Looping	LOOP label, n	Repeat instruction, jump to label till n is 0.	141
Other Instruction	Key Matrix Scan	KEYSCAN	Key Matrix Scan instruction maximum connection to 8 * 8 (64keys) Used for connecting membrane key panels.	142
	Output all off	OUTOFF	Off I/O port set to output	144
	Distribute	DIST s,d,n	Distribute 16bit data by 4bit to d.	145
	Combine	UNIT s,d,n	Combine 16bit data just including lower 4bits into 1word.	146
	Encode	ENCO s,d	Encode lower 4bits of s and store at d.	147
	Decode	DECO s,d	Decode 16bits of s and store at lower 4 bits.	148
	Thermometer Input	THIN port, d	Measure current temperature from Digital thermometer DS1820 and store at d.	149

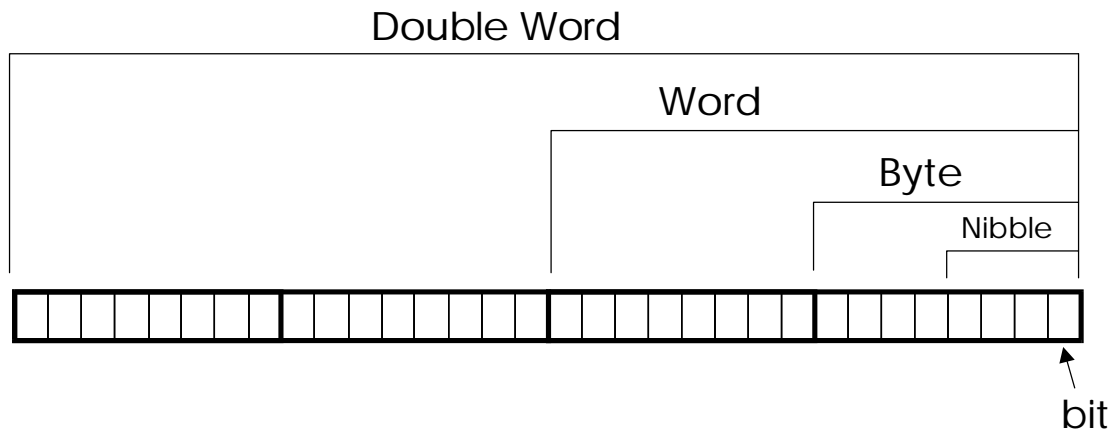
4.2. Number system in TinyPLC

Before explaining High Level Instructions, the concept of number system and code (BCD code, ASCII code) treated in TinyPLC should be explained first. There are binary, decimal and hexadecimal numbers and marked as below.

decimal	binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- ? A binary number of four figures can be expressed by one hexadecimal.
- ? A~F is used to express 10~15 in hexadecimal.
- ? Number expression at ladder program
 - Binary: 00011010B
 - Decimal: 123, 80
 - Hexadecimal: 0ABH, 12345H

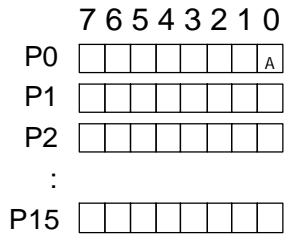
There are units as bit, byte, word and double word to classify the dimension of numbers.



Bit is the basic unit to express 1 and 0. 1 nibble is composed of 4bits. 1 byte is composed of 8bits and 1 word of 16bits.

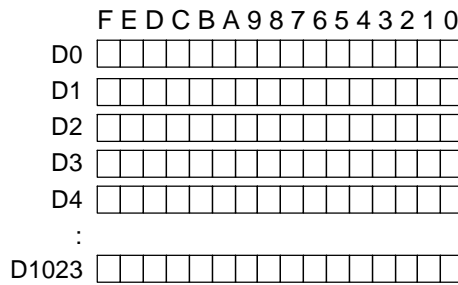
There are areas parted by byte unit and word unit in TinyPLC.

Area by byte	Area by word
P area (in/output)	T area (timer)
K area (KEEP)	C area (counter)
M area (sub relay)	D area (data)
CH area (LCD display data)	AD area (AD result)
G area (SGN data)	CNT area (high speed counter value)



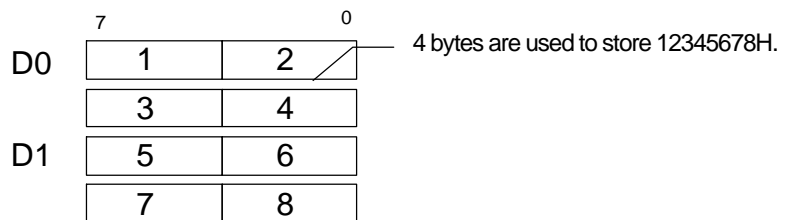
P area is parted by byte unit. P0:0 shows 0th bit (A spot) of P0 byte. This shows that the P0:8 is the wrong expression because there is no spot after 8th bit.

D area in which data is stored is parted by word unit.



The instructions of arithmetic operation and logic operation in TinyPLC are basically in word unit. There is "double word" instruction, which 2 words are put together. In using double word, the assigned word number is used with the next word number together.

For example, if you run DWMOV 12345678H, D0 (means store 12345678H in D0), D0 and D1 are used together. Data is stored in byte order. The following figure shows the state of D0 how 12345678H is stored.



ASCII CODE

ASCII Code system is used for displaying and printing. 1byte indicates 1number. The following is the table of ASCII code. (ASCII code of number 8 is 38H)

	Lower rank 4bit																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
High rank 4bit	2	!	“	#	\$	%	&	'	()	*	+	,	-	.	/	
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	7	p	q	r	s	t	u	v	w	x	y	z	{		}		

BCD CODE

BCD (binary-coded decimal) code system represents decimal (0~9) numbers to binary 4BI (represents a decimal number of one figure by 4bit unit).1byte can express 2 digit decimal number and 2 byte 4digit. The following chart shows BCD code.

decimal	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

It is similar to the binary number, but the next example shows the difference between binary number and BCD code.

Decimal (original number)	Transfer by binary	Transfer by BCD code
100	01100100 (64H)	000100000000 (100H)

4.2. Description of high level instructions

MOVE

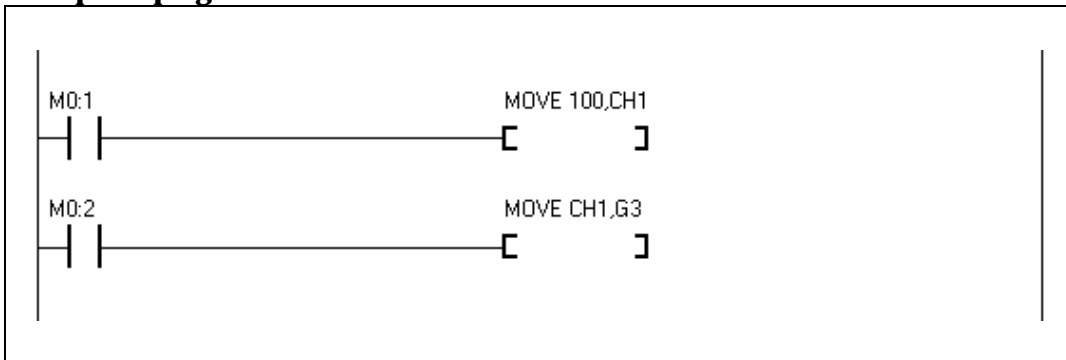
8BIT Move

Usage : MOVE s, d

Summary

It is data transfer instruction by 8bit. Transfers the value (or 8bit constants) in s to d.

Example of program



Operands

operand	Relay					counter time data			etc				Constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

if the M:01 becomes ON, 100 is transferred to CH1. If M0:2 becomes ON, CH1 is transferred into G3. (Byte can indicate 0~255)

Precautions in using C, T, D area:

As MOVE instruction transfers data by byte unit and C, T, D area by word unit, we need some precautions in usage. If the instruction is "MOVE 100, D0", 100 will be stored at the higher rank area of D0 and all the values of D0 will be wrong. In this case, we must use WMOV instruction (word unit instruction) which will be explained later.

CMOVE

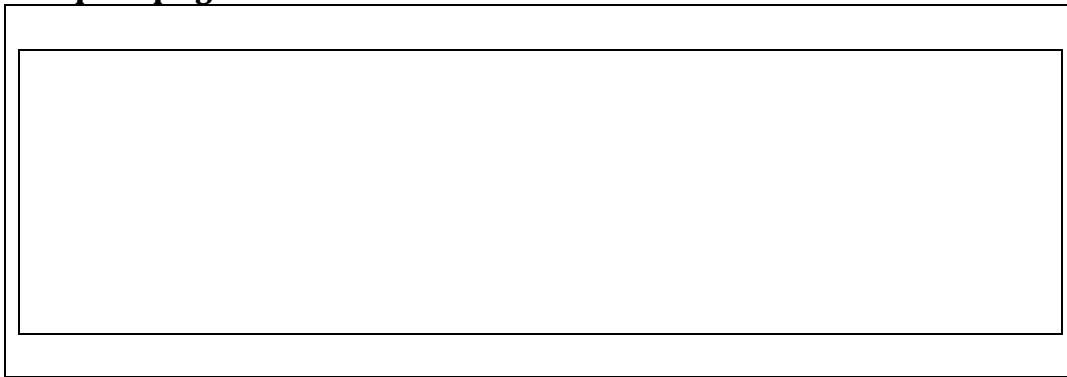
8bit Complement Move

Usage : CMOVE s, d

Summary

It is data complement transfer instruction by 8bit. After completing the value (or 8bit constants) is s, it is transferred to d.

Example of program

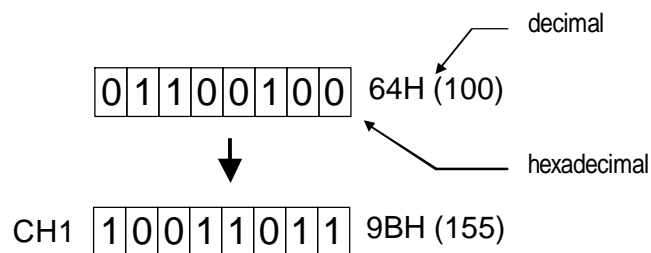


Operands

operand	Relay					counte	time	data	Etc.				constant
	P	M	F	K	S	r	r	D	AD	CNT	CH	G	
s (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, complement 100(155) is transferred to CH1. If M0:2 becomes ON, complement CH1 is transferred to G3.



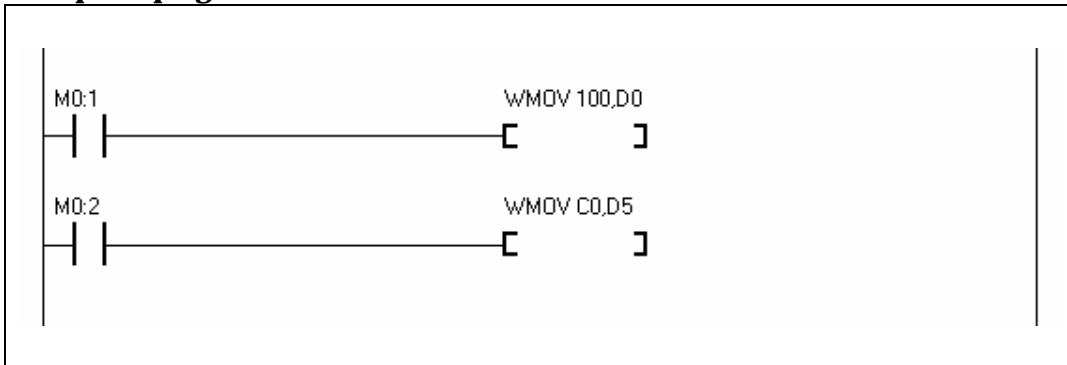
WMOV 16bit Move

Usage : WMOV s, d

Summary

It is data transfer instruction by 16bit. The value (or 16bit constant) is transferred to d.

Example of program



Operands

operand	Relay					counter	timer	data	Etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s (source)	○	○		○		○	○	○	○	○	○	○	○
d (destination)	○	○		○		○	○	○	○	○	○	○	

Description

If M0:1 becomes ON, 100 is transferred to D. If M0:2 becomes ON, C0 is transferred to D5. In using constants, we can't use over 65535. (Word can express 0~65535)

Precautions in using byte area:

As WMOV instruction transfers data by word unit and P,M,K by byte, we need some precautions in usage. When we run WMOV, it executes by 2 bytes. For example, if you run "WMOV 1234H, CH0", 12H is stored at CH0 and 34H at CH1.

DWMOV

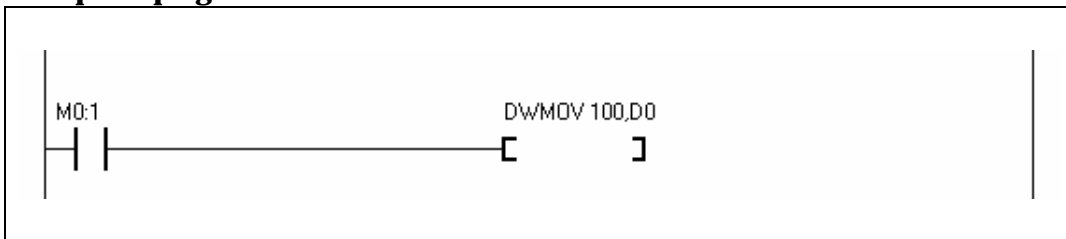
32Bit Move

Usage : DWMOV s, d

Summary

It is data transfer instruction by 32 bit. The value (or 32bit constants) in s is transferred to d.

Example of program



Operands

operand	Relay					counter	timer	data	Etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s (source)	○	○		○		○	○	○	○	○	○	○	○
d (destination)	○	○		○		○	○	○	○	○	○	○	

Description

If M0 becomes ON, 0 is transferred to D0 and 100 to D1. Figure beside shows that 100 is stored at D1 and 0 at D0.(higher area in order)

you cannot use over 2,147,418,112 (7FFF0000H in hexadecimal) in using constants)

	HIGH byte	LOW byte
D0	00H	00H
D1	00H	64H
D2		
D3		

About storage of byte, word and double word data...

There are three kinds of data transfer instruction.

Classification	Transfer instruction	Complement transfer instruction
Byte unit	MOVE	CMOVE
Word unit	WMOV	WCMOV
Double word unit	DWMOV	DWCMOV

Following shows the result of the instruction of transferring hexadecimal 12H to D0.

<p>MOVE 100,D0</p> <p style="margin-left: 40px;">HIGH byte LOW byte</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px;">D0</td><td style="padding: 2px;">64H</td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D1</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D2</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D3</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> </table>	D0	64H		D1			D2			D3			<p>WMOV 100,D0</p> <p style="margin-left: 40px;">HIGH byte LOW byte</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px;">D0</td><td style="padding: 2px;">00H</td><td style="padding: 2px;">64H</td></tr> <tr><td style="padding: 2px;">D1</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D2</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D3</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> </table>	D0	00H	64H	D1			D2			D3			<p>DWMOV 100,D0</p> <p style="margin-left: 40px;">HIGH byte LOW byte</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px;">D0</td><td style="padding: 2px;">00H</td><td style="padding: 2px;">00H</td></tr> <tr><td style="padding: 2px;">D1</td><td style="padding: 2px;">00H</td><td style="padding: 2px;">64H</td></tr> <tr><td style="padding: 2px;">D2</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr><td style="padding: 2px;">D3</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> </table>	D0	00H	00H	D1	00H	64H	D2			D3		
D0	64H																																					
D1																																						
D2																																						
D3																																						
D0	00H	64H																																				
D1																																						
D2																																						
D3																																						
D0	00H	00H																																				
D1	00H	64H																																				
D2																																						
D3																																						

Usable constant range by each unit is as follows.

Classification	Range(decimal)	Range(hexadecimal)
Byte unit	0~255	0~0FFH
Word unit	0~65535	0~0FFFFH
Double word unit	0~2147418112	0~7FFF0000H

Reference

Word instructions starts with W (example: WMOV)

Double word instructions start with DW (example: DWMOV)

WCMOV

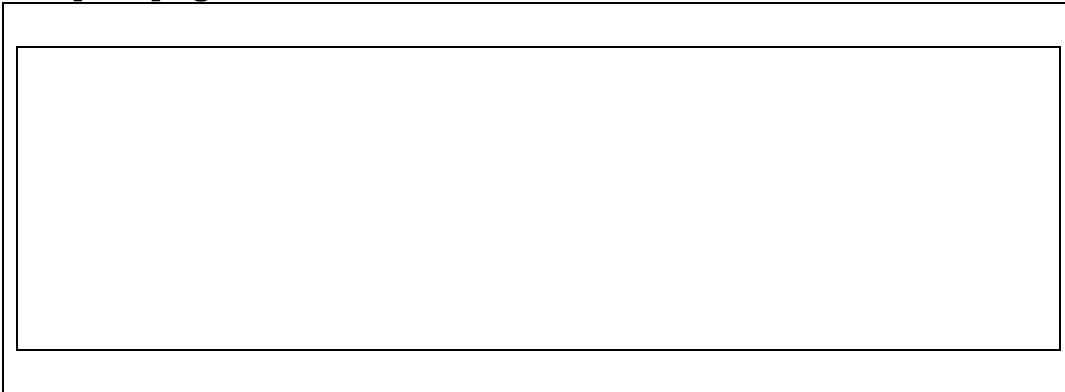
16bit complement move

Usage : WCMOV s, d

Summary

It is complement transfer instruction by 16 bit.. Transfers the complement value in s (or 16bit constant) to d

Example of program



Operands

operand	Relay					counter time data			Etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s (source)	○	○		○		○	○	○	○	○	○	○	○
d (destination)	○	○		○		○	○	○	○	○	○	○	

Description

If M0:1 becomes ON, it transfers 65435(complement of 100) to D0. If M0:2 becomes ON, it transfers complement C0 to D5. In using constants, over 65535 cannot be used. (word can express 0~65535)

DWCMOV

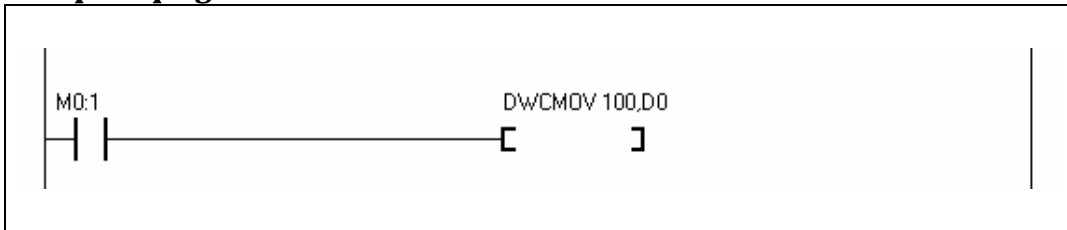
32bit complement move

Usage : DWCMOV s, d

Summary

It is 32bit complement transfer instruction. It transfers the complement value in s(or 32bit constant) to d.

Example of program



Operands

operand	relay					counte	time	data	Etc.				constant
	P	M	F	K	S	r	r	D	AD	CNT	CH	G	
s (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it transfers 0 to D0 and 100 to D1. The figure beside shows the result.

	HIGH byte	LOW byte
D0	0FFH	0FFH
D1	0FFH	09BH
D2		
D3		

WNEG

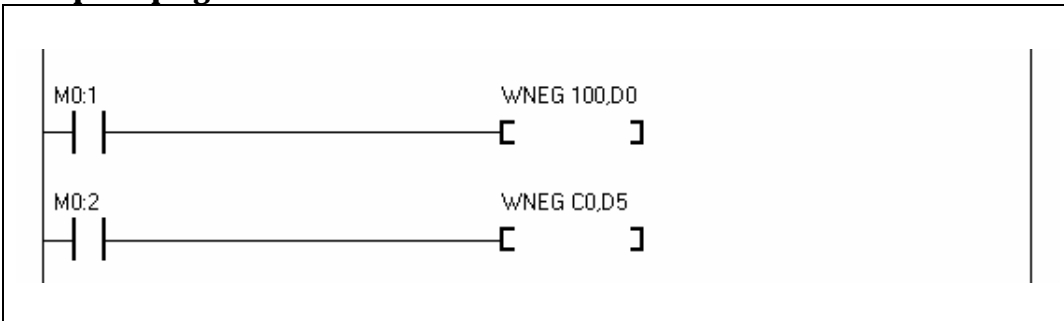
16bit two's complement move

Usage : WNEG s, d

Summary

It is transfer instruction of complement, two's. After converting s(or 16bit constant) by complement, two's, transfers to d.

Example of program



Operands

operand	relay					counter	timer	data	Etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	○	○		○		○	○	○	○	○	○	○	○
d (destination)	○	○		○		○	○	○	○	○	○	○	

Description

If M0:1 becomes ON, it transfers -100 (100 by complement, two's) to D0. If M0:2 becomes ON, it transfers C0 after converting by complement, two's to D5.

What is complement, two's?

In computers complement, two's means negative number. Complement, two's complements ordinary numbers and add 1. (Complement, two's of 1 is 0FFFFH.) As $1 + (-1) = 0$, $1 + 0FFFFH$ is 0. Because of this principal we can express all the negative numbers by using complement, two's. In TinyPLC, there is no actual negative instruction but you can use complement, two's to express it.

DWNEG

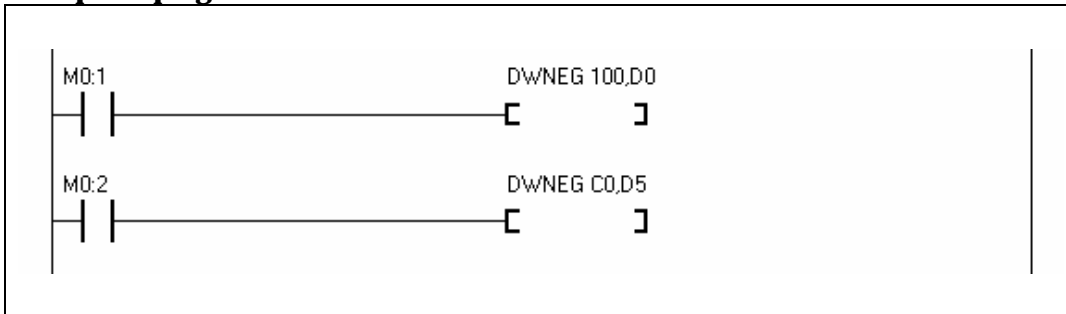
32bit two's complement move

Usage : DWNEG s, d

Summary

It is 32bit transfer instruction by complement two. It transfers values(or 32bit constants) in s by converting complement two to d.

Example of program



Operands

operand	relay					counte	time	data	etc.				constant
	P	M	F	K	S	r	r	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it transfers -100 (100 by complement, two's) to D0, D1. If M0:2 becomes ON, it transfers C0,C1 after converting by complement, two to D5, D6.

WXCHG

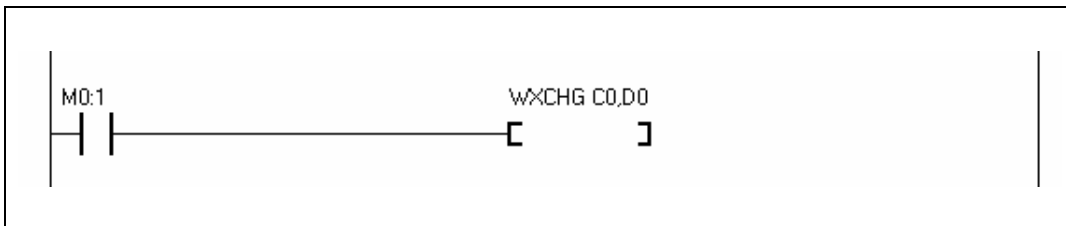
16bit data exchange

Usage : WXCHG s, d

Summary

It is data exchange instruction by 16bit.. It exchanges the value in s with the value in d.

Example of program

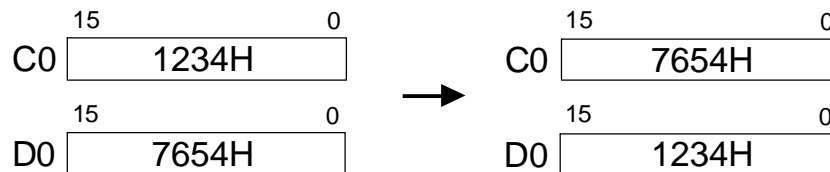


Operands

operand	relay					counter	timer	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	○	○		○		○	○	○	○	○	○	○	
d (destination)	○	○		○		○	○	○	○	○	○	○	

Description

If M0:1 becomes ON, it exchanges the value in D0 with in C0.



DWXCHG

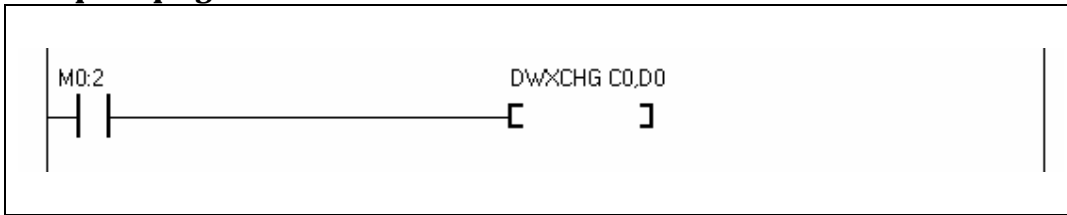
32bit data exchange

Usage : DWXCHG s, d

Summary

It is data exchange instruction by 32bit.. It exchanges the value in s with the value in d.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s (source)	O	O		O		O	O	O	O	O	O	O	
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it exchanges the values in D0, D1 with in C0, C1.

FMOV

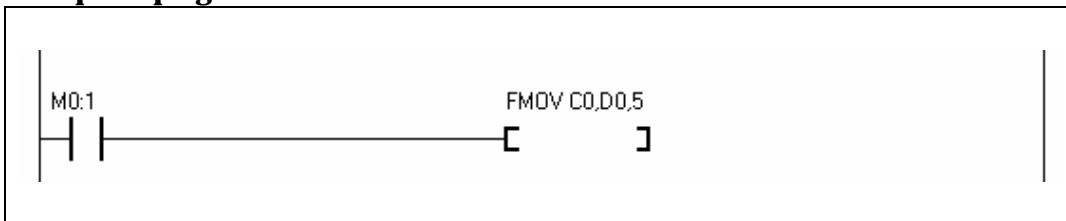
Filling up data

Usage : FMOV s, d, n

Summary

It fills up the value (or 16bit constants) in s from d by assigned number n.

Example of program

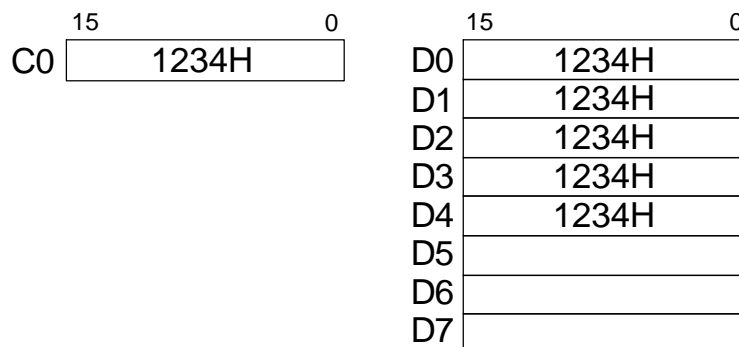


Operands

operand	relay					counter	timer	data	etc.				constant
	P	M	F	K	S				C	T	D	AD	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	
N(number)													O

Description

If M0:1 becomes ON, it stores value (16bit value) in C0 to D0, D1, D2, D3, D4.



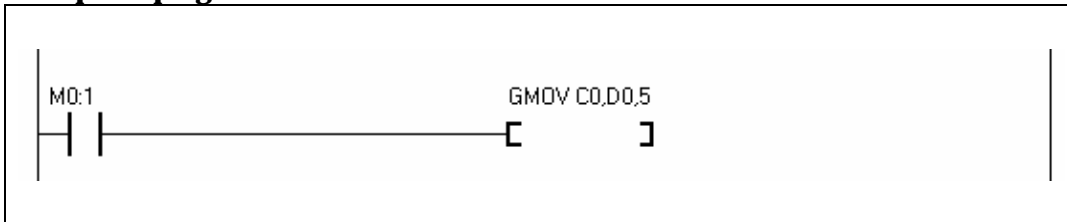
GMOV Group transfer

Usage : GMOV s, d, n

Summary

It is the transfer instruction by group which can transfer various data. It transfers the assigned number (n) of value from s to d.

Example of program

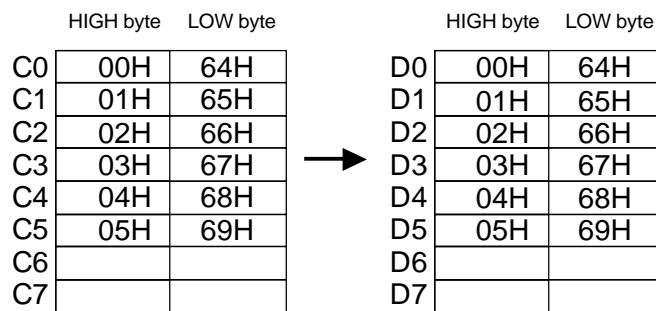


Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	
d (destination)	O	O		O		O	O	O	O	O	O	O	
n (number)													O

Description

If M0:1 becomes ON, it transfers C0-->D0, C1--> D1, C2-->D2, C3--> D3, C4--> D4.



WBCD

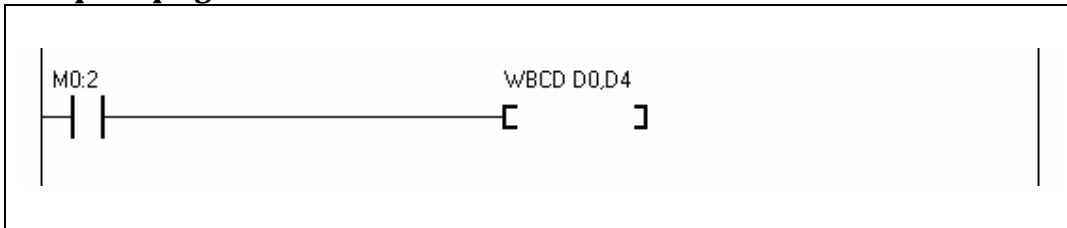
16bit BIN to BCD convert

Usage : WBCD s, d

Summary

It converts 16bit binary data to 4 digit BCD data.

Example of program

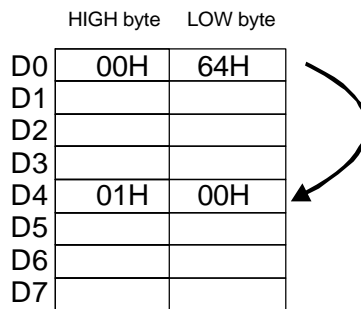


Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:2 becomes ON, it stores the value in D0 to D4 after converting to 4digit BCD code.



DWBCD

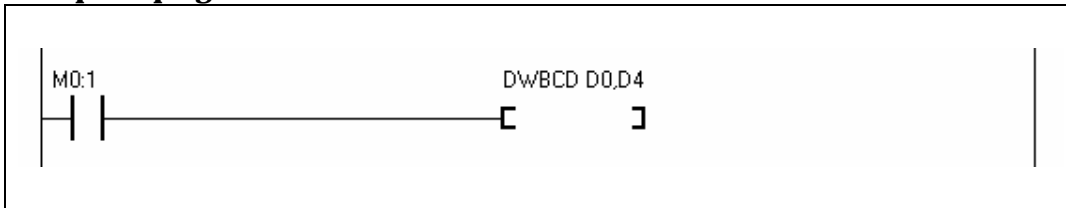
32bit BIN to BCD convert

Usage : DWBCD s, d

Summary

It converts 32bit binary data to 8 digit BCD data.

Example of program



Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it stores the value in D0,D1 to D4 after converting to 8digit BCD code. (0BC614EH is 12345678 in decimal number.)

	HIGH byte	LOW byte
D0	00H	BCH
D1	61H	4EH
D2		
D3		
D4	12H	34H
D5	56H	78H
D6		
D7		

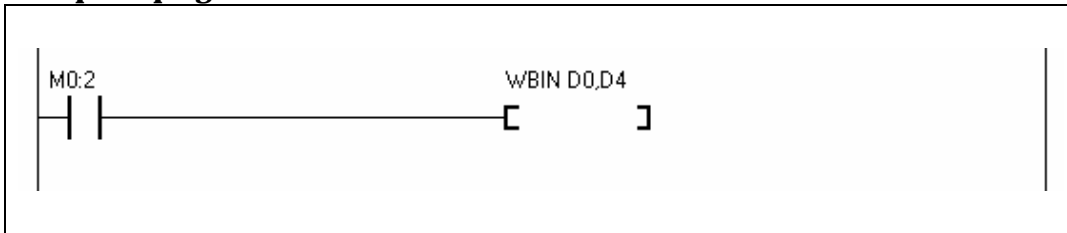
WBIN 16bit BCD to BIN convert

Usage : WBIN s, d

Summary

It converts 4 digit BCD data to 16bit binary data.

Example of program

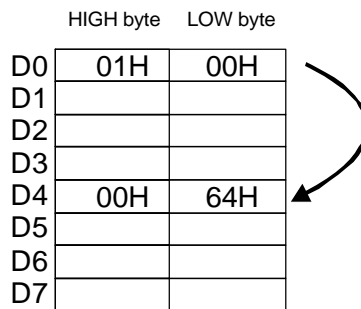


Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:2 becomes ON, it stores the value from D0 to D4 after converting from 4digit BCD code to binary value.



DWBIN

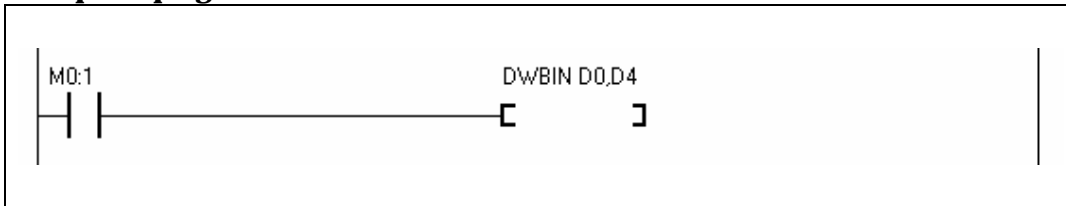
32bit BCD to BIN convert

Usage : DWBIN s, d

Summary

It converts 8 digit BCD data to 32bit binary data.

Example of program



Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S (source)	O	O		O		O	O	O	O	O	O	O	O
d (destination)	O	O		O		O	O	O	O	O	O	O	

Description

If M0:2 becomes ON, it stores the value from D0 to D4 after converting 8digit BCD code to binary value.

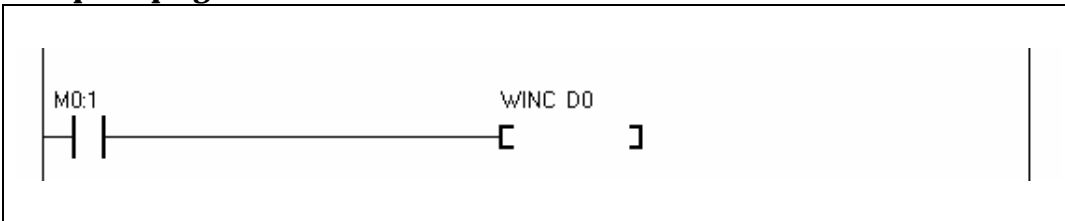
	HIGH byte	LOW byte
D0	00H	12H
D1	34H	56H
D2		
D3		
D4	00H	01H
D5	E2H	40H
D6		
D7		

WINC	Increase 1 by 16bit
Usage : WINC d	$d = d + 1$

Summary

It is increase 1 instruction by 16bit.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d						O	O	O	O	O			

Description

If M0:1 becomes ON, 1 increase at D0.

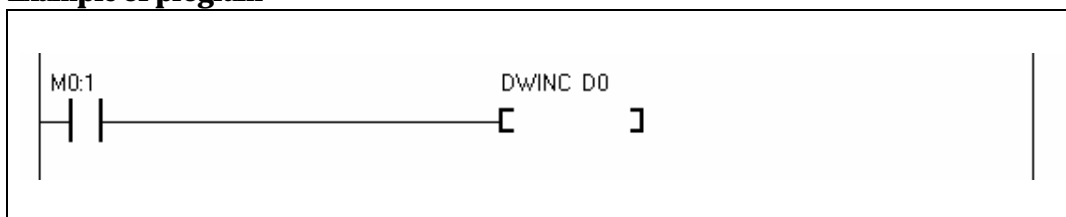
(D0 = D0 + 1)

DWINC	Increase 1 by 32bit
Usage : DWINC d	$d = d + 1$

Summary

It is increase 1 instruction by 32bit.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d						O	O	O	O	O			

Description

If M0:1 becomes ON, 1 increases at D0, D1.

(D0 ,D1 = D0,D1 + 1)

WDEC

1 decrease by 16bit

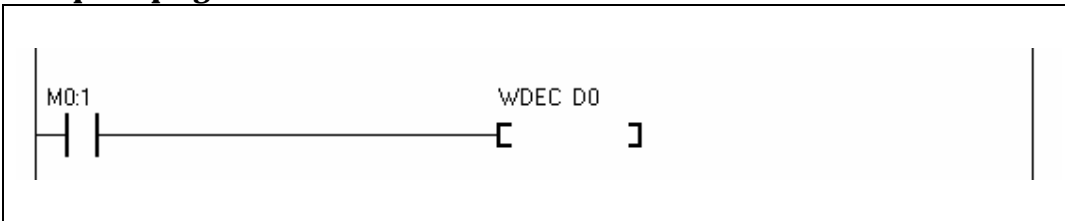
Usage : WDEC d

$d = d - 1$

Summary

It is decrease 1 instruction by 16bit.

Example of program



Operands

operand	relay					counte r	timer	data	etc.				constan t
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d						O	O	O	O	O			

Description

If M0:1 becomes ON, 1 decrease at D0.

$(D0 = D0 - 1)$

DWDEC	1 decrease by 32bit
Usage : DWDEC d	$d = d - 1$

Summary

It is decrease 1 instruction by 32bit.

Example of program

Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d						O	O	O	O	O			

Description

If M0:1 becomes ON, 1 decreases at D0, D1.

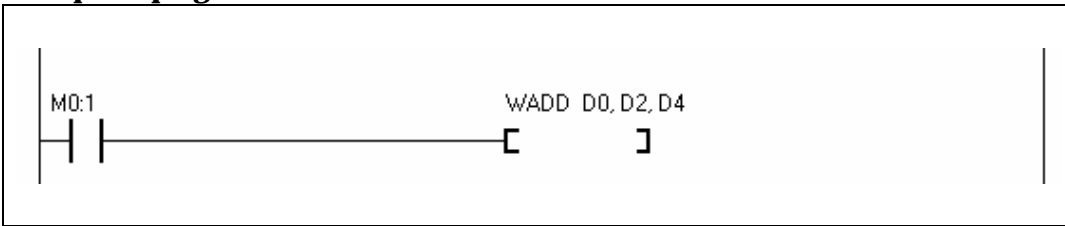
(D0, D1 = D0, D1 - 1)

WADD	Addition by 16bit
Usage : WADD s1,s2, d	$d = s1 + s2$

Summary

It is addition instruction by 16bit. It stores the result of the addition of s1 and s2 at d.

Example of program

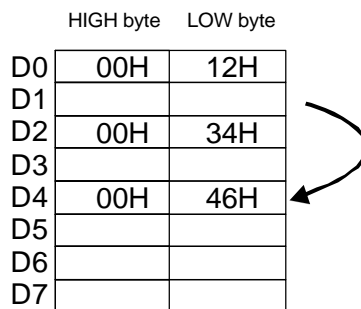


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of the addition of D0 and D2 at D4.

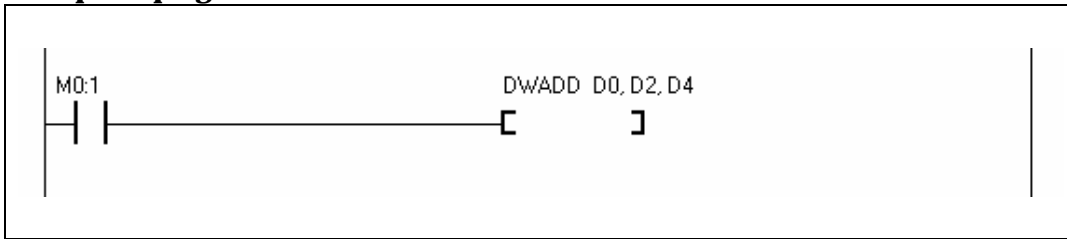


<h1 style="margin: 0;">DWADD</h1>	Addition by 32bit
Usage : DWADD s1,s2, d	$d = s1 + s2$

Summary

It is addition instruction by 32bit. It stores the result of the addition of s1 and s2 at d.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of the addition of D0, D1 and D2, D3 at D4 and D5.

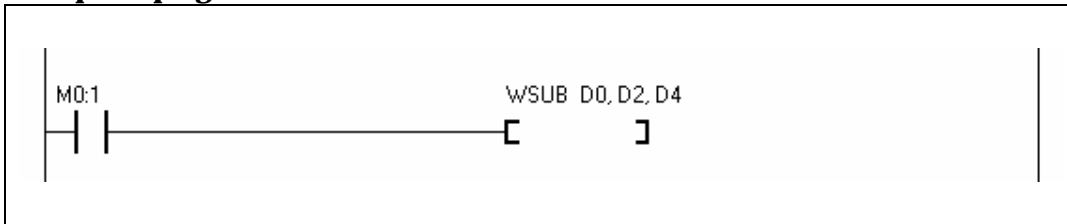
	HIGH byte	LOW byte	
D0	00H	01H	
D1	03H	12H	
D2	00H	02H	
D3	04H	34H	
D4	00H	03H	
D5	07H	46H	
D6			
D7			

WSUB	Subtraction by 16bit
Usage : WSUB s1,s2, d	$d = s1 - s2$

Summary

It is subtraction instruction by 16bit. It stores the result of subtraction from s1 to s2 at d.

Example of program



Operands

operand	relay					counte	timer	data	etc.				constan t
	P	M	F	K	S	r	C	T	D	AD	CNT	CH	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of subtraction from D0 to D2 at D4.

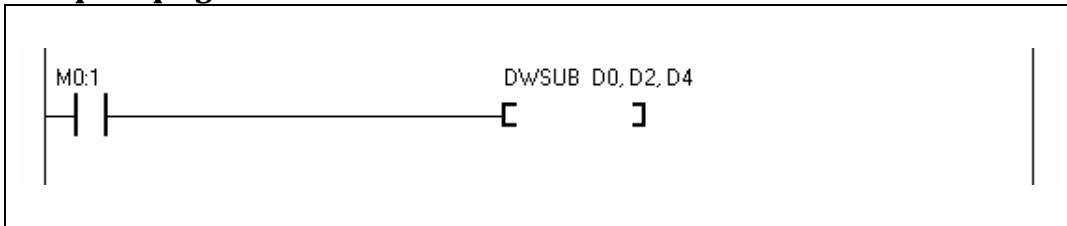
	HIGH byte	LOW byte	
D0	34H	56H	
D1			
D2	12H	34H	
D3			
D4	22H	22H	}
D5			
D6			
D7			

DWSUB	32bit subtract
Usage : DWSUB s1,s2, d	d = s1 - s2

Summary

It is subtraction instruction by 32bit. It stores the result of subtraction from s1 to s2 at d.

Example of program



Operands

operand	relay					counter	time r	data D	etc.				constant
	P	M	F	K	S				AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of subtraction from D0, D1 to D2, D3 at D4, D5.

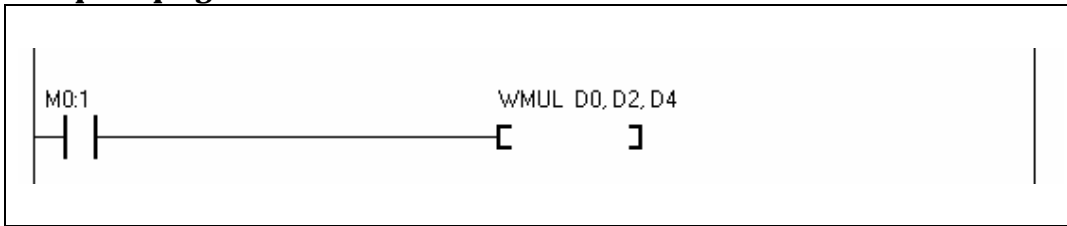
	HIGH byte	LOW byte	
D0	34H	56H	
D1	78H	9AH	
D2	12H	34H	
D3	56H	78H	
D4	22H	22H	
D5	22H	22H	
D6			
D7			

WMUL	16bit multiplication
Usage : WSUB s1,s2, d	$d = s1 * s2$

Summary

It is 16bit multiplication instruction. It stores the multiplication result of s1 s2 at d by 32bit.

Example of program



Operands

operand	relay					counter	time r	data D	etc.				constan t
	P	M	F	K	S				AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the multiplied value of D0 and D2 at D4 by 32bit. (you need precaution as the result is by double word unit).

	HIGH byte	LOW byte	
D0	01H	23H	
D1			
D2	04H	56H	
D3			
D4	00H	04H	↷
D5	EDH	C2H	
D6			
D7			

DWMUL

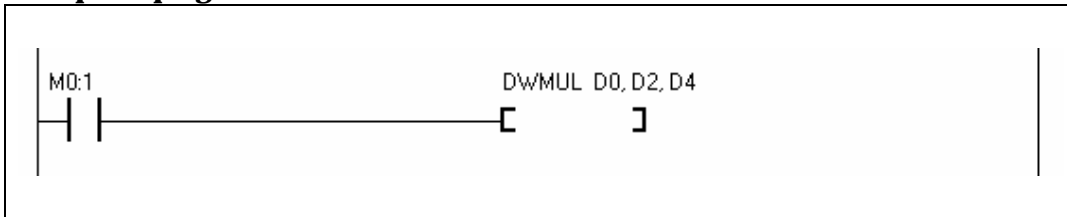
32bit multiplication

Usage : DWSUB s1,s2, d d = s1 * s2

Summary

It is 32bit multiplication instruction. It stores the multiplication result of s1 s2 at d by 64bit.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the multiplied value of D0, D1 and D2, D3 at D4, D5, D6, and D7 by 64bit.

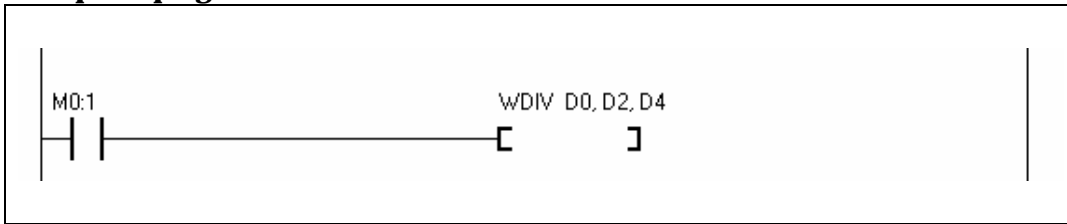
	HIGH byte	LOW byte
D0	00H	12H
D1	34H	56H
D2	00H	33H
D3	33H	33H
D4	00H	00H
D5	00H	00H
D6	7FH	70H
D7	FDH	6CH

WDIV	16bit division
Usage : WDIV s1,s2, d	$d = s1 / s2$

Summary

It is division instruction by 16bit. It stores the quotient of s1 divided by s2 at d and remainder at d+1.

Example of program



Operands

operand	relay					counte	timer	data	etc.				constan t	
	P	M	F	K	S	r	C	T	D	AD	CNT	CH		G
s1						O	O	O	O	O				O
s2						O	O	O	O	O				O
d						O	O	O						

Description

If M0:1 becomes ON, it stores the quotient of D0 divided by D1 at D4 and remainder at D5.

	HIGH byte	LOW byte
D0	12H	34H
D1		
D2	00H	03H
D3		
D4	06H	11H
D5	00H	01H
D6		
D7		

DWDIV

32bit division

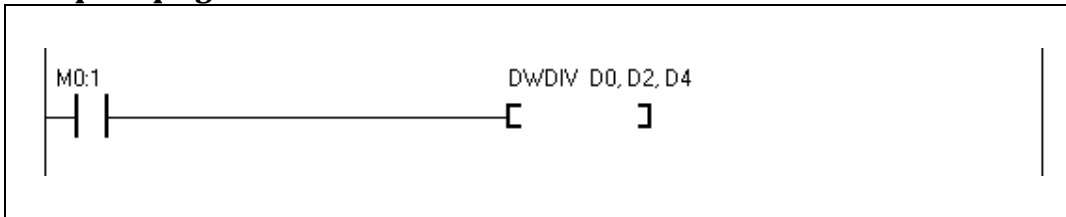
Usage : DWDIV s1,s2, d

$d = s1 / s2$

Summary

It is division instruction by 32bit. It stores the quotient of s1 divided by s2 at d and remainder at d+1.

Example of program



Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
S1						O	O	O	O	O			O
S2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the quotient of D0, and D1 divided by D2, D3 at D4, D5 and remainder at D6, D7.

	HIGH byte	LOW byte
D0	12H	34H
D1	56H	78H
D2	00H	00H
D3	00H	07H
D4	02H	99H
D5	C3H	35H
D6	00H	00H
D7	00H	05H

WAND	16BIT AND Operation
Usage : WAND s1,s2, d	d = s1 and s2

Summary

It is 16bit AND instruction. It stores the result of AND operation of s1 and s2 at d.

Example of program

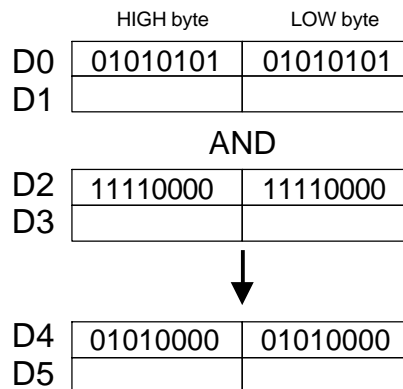


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of D0 and D2 by AND operation at D4.



DWAND

32bit AND operation

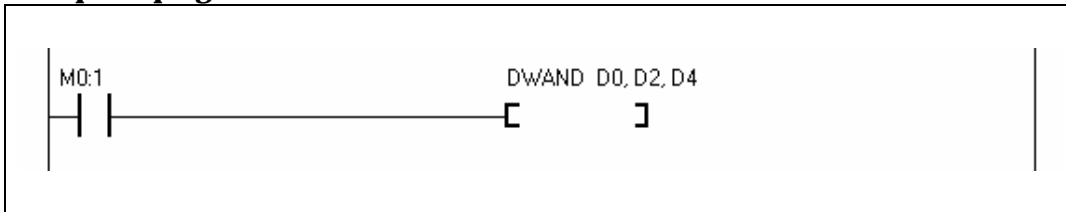
Usage : DWAND s1,s2, d

d = s1 and s2

Summary

It is 32bit AND instruction. It stores the result of AND operation of s1 and s2 at d.

Example of program

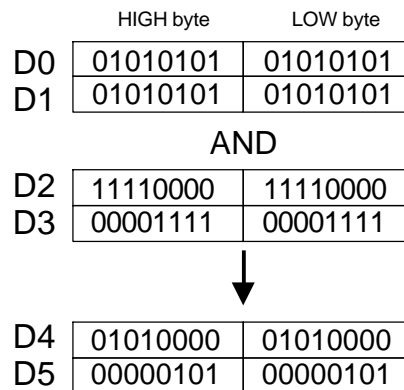


Operands

operand	relay					counter	time r	data D	etc.				constant
	P	M	F	K	S				AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of D0, D1 and D2, D3 by AND operation at D4, D5.



WOR

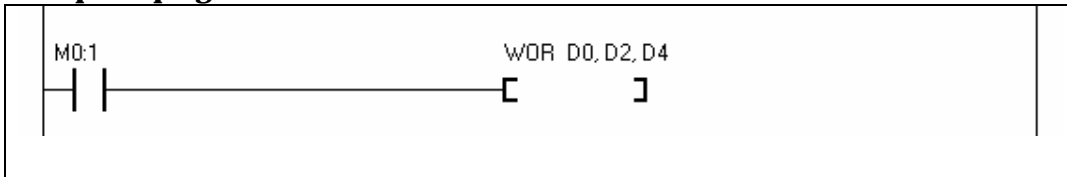
16BIT OR Operation

Usage : WOR s1,s2, d d = s1 or s2

Summary

It is 16bit OR instruction. It stores the result of OR operation of s1 and s2 at d.

Example of program

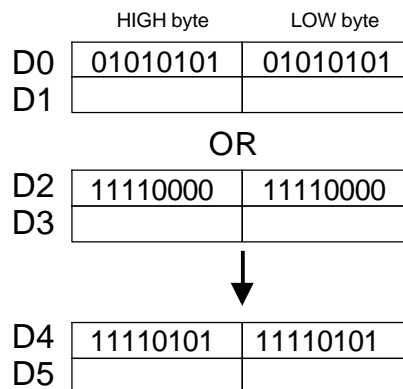


Operands

operand	RELAY					counter	time	data	etc.				constant
	P	M	F	K	S				C	T	D	AD	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of D0 and D2 by OR operation at D4.

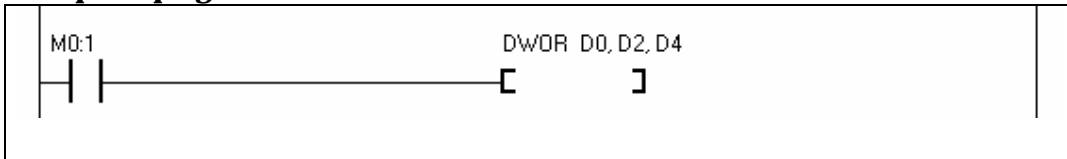


DWOR	32bit OR Operation
Usage : DWOR s1,s2, d	d = s1 or s2

Summary

It is 32bit OR instruction. It stores the result of OR operation of s1 and s2 at d.

Example of program

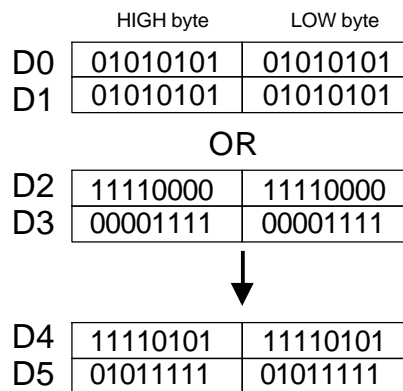


Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of D0,D1 and D2,D3 by OR operation at D4,D5.



WXOR

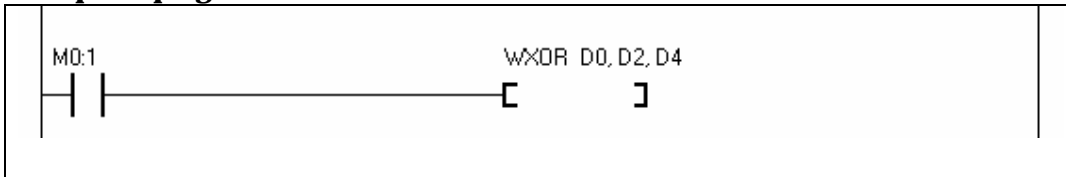
16bit XOR operation

Usage : WXOR s1,s2, d d = s1 xor s2

Summary

It is 16bit XOR instruction. It stores the result of XOR operation of s1 and s2 at d.

Example of program

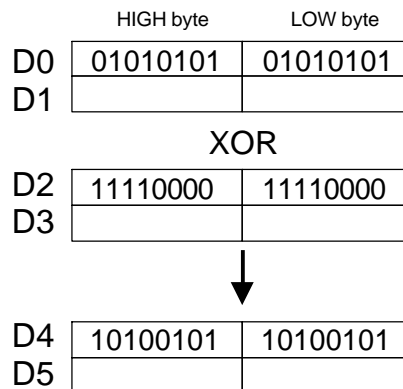


Operands

operand	RELAY					counter	time	data	etc.				constant
	P	M	F	K	S				C	T	D	AD	
S1						O	O	O	O	O			O
s2						O	O	O	O	O			O
d						O	O	O					

Description

If M0:1 becomes ON, it stores the result of D0 and D2 by XOR operation at D4.



DWXOR

32BIT XOR Operation

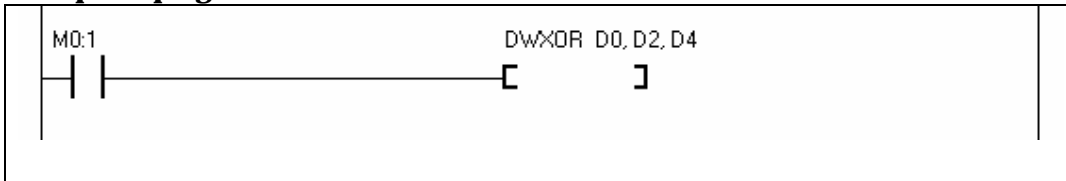
Usage : DWXOR s1,s2, d

d = s1 xor s2

Summary

It is 32bit XOR instruction. It stores the result of XOR operation of s1 and s2 at d.

Example of program



Operands

operand	RELAY					counter	time r	data	etc.				constant	
	P	M	F	K	S				C	T	D	AD		CNT
s1						O	O	O	O	O				O
s2						O	O	O	O	O				O
d						O	O	O						

Description

If M0:1 becomes ON, it stores the result of D0, D1 and D2, D3 by XOR operation at D4, D5.

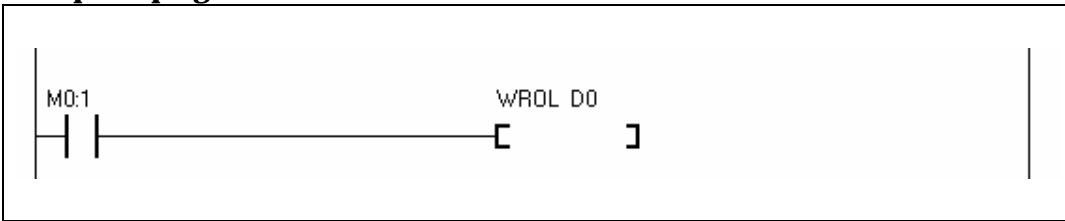
	HIGH byte	LOW byte
D0	01010101	01010101
D1	01010101	01010101
XOR		
D2	11110000	11110000
D3	00001111	00001111
↓		
D4	10100101	10100101
D5	01011010	01011010

WROL	16BIT ROTATE LEFT
Usage : WROL d	$d = d \ll 1$

Summary

It is rotate left instruction by 16bit. It rotates d to left by one bit.

Example of program

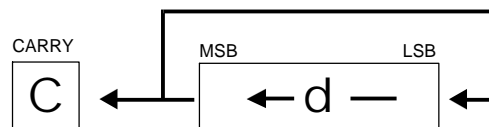


Operands

operand	relay					counte r	timer	data	etc.				constan t
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0 to left by one bit. The value in bit(MSB) of highest level moves to CARRY bit.



DWROL

32 BIT ROTATE LEFT

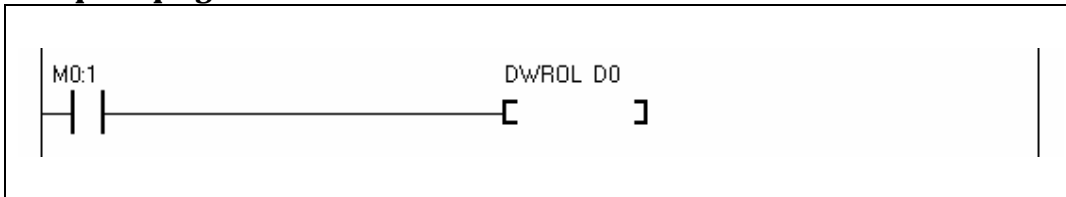
Usage : DWROL d

$d = d \ll 1$

Summary

It is rotate left instruction by 32bit. It rotates d to left by one bit.

Example of program



Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0, D1 to left by one bit. The value in bit (MSB) of the highest level moves to CARRY bit.

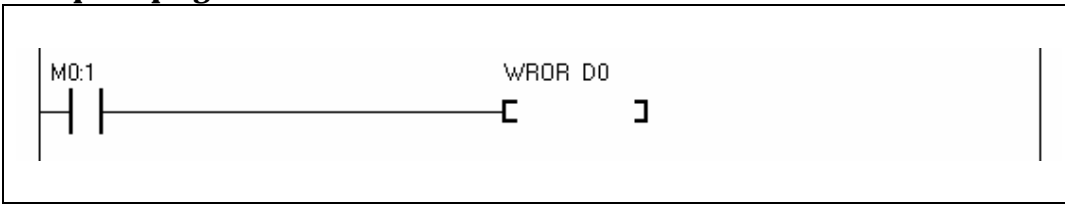


WROR	16BIT ROTATE right
Usage : WROR d	$d = d \gg 1$

Summary

It is rotate right instruction by 16bit. It rotates d to right by one bit.

Example of program



Operands

operand	relay					counte r			timer data				etc.				constan t
	P	M	F	K	S	C	T	D	AD	CNT	CH	G					
d	○	○		○		○	○	○	○	○	○	○	○	○	○	○	

Description

If M0:1 becomes ON, it rotates D0 to right by one bit. The value in bit (LSB) of the lowest level moves to CARRY bit.



DWROR

32 BIT ROTATE right

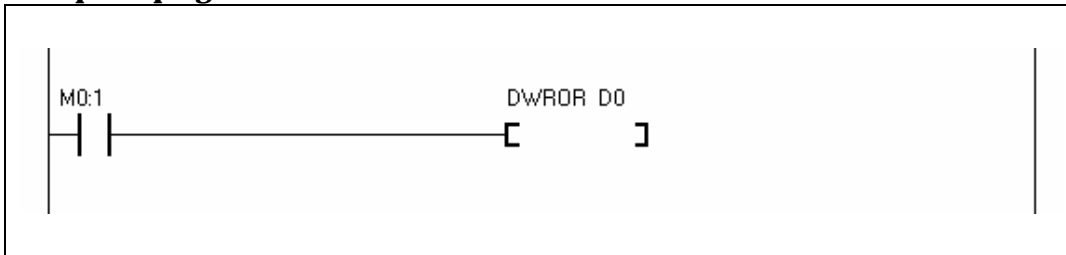
Usage : DWROR d

$d = d \gg 1$

Summary

It is rotate right instruction by 32bit. It rotates d to right by one bit.

Example of program



Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0, D1 to right by one bit. The value in bit (LSB) of the lowest level moves to CARRY bit.



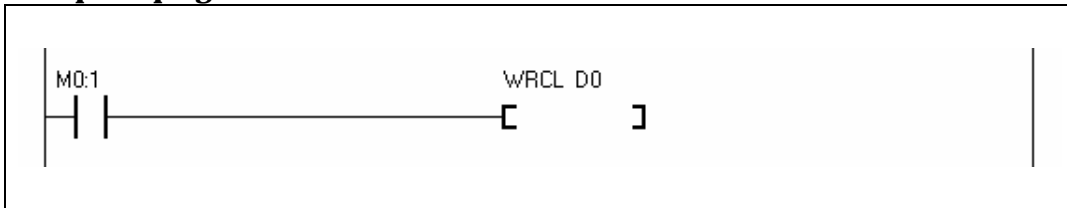
WRCL

16bit rotate left(including Carry)
Usage : WRCL d d = d << 1 with C

Summary

It is rotate left instruction by 16bit. It rotates d including carry to left by one bit.

Example of program

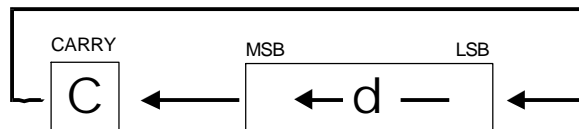


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0 with carry flag to right by one bit. The value in bit (MSB) of the highest level moves to CARRY bit.



DWRCL

32 bit rotate left(including Carry)

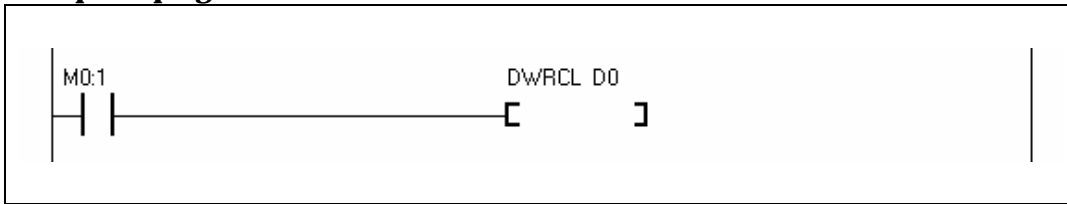
Usage : DWRCL d

$d = d \ll 1$ with C

Summary

It is rotate left instruction by 32bit. It rotates d including carry to left by one bit.

Example of program



Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0,D1 with carry flag to left by one bit. The value in bit(MSB) of the highest level moves to CARRY bit.



WRCR

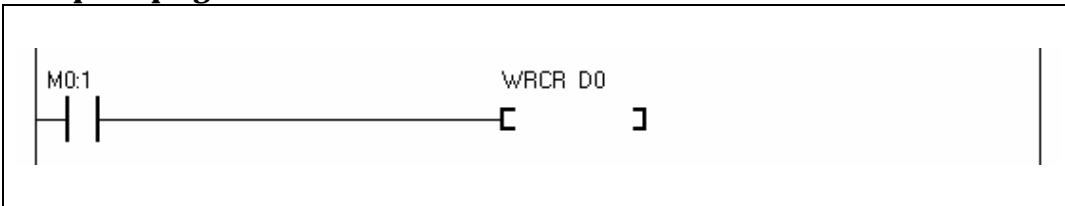
16 bit rotate right(including Carry)

Usage : WRCR d d = d >> 1 with C

Summary

It is rotate right instruction by 16bit. It rotates d including carry to right by one bit.

Example of program

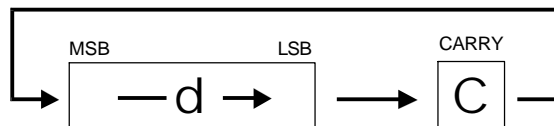


Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0 with carry to right by one bit. The value in bit(LSB) of the lowest level moves to CARRY bit.



DWRCR

32bit rotate right(including Carry)

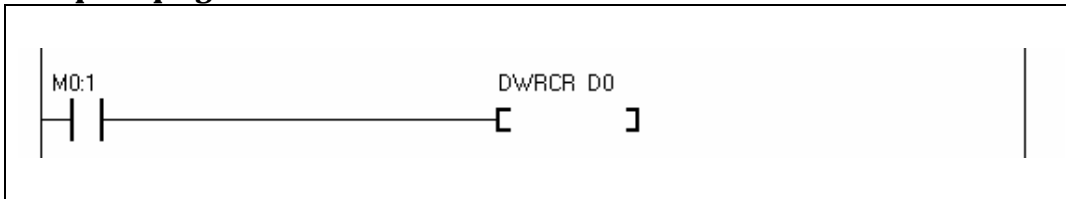
Usage : DWRCR d

d = d >> 1 with C

Summary

It is rotate right instruction by 16bit. It rotates d including CARRY to right by one bit.

Example of program



Operands

operand	RELAY					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0,D1 with CARRY to right by one bit. The value in bit(LSB) of the lowest level moves to CARRY bit.



BSHL

Shift left by bit unit

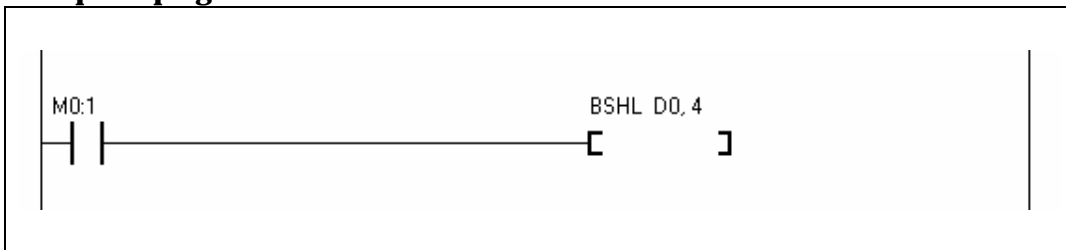
Usage : BSHL d,n

$d = d \ll n$

Summary

It is shift left instruction by bit unit. It shifts d to left by n bit. The newly inserted bit is set to 0.

Example of program

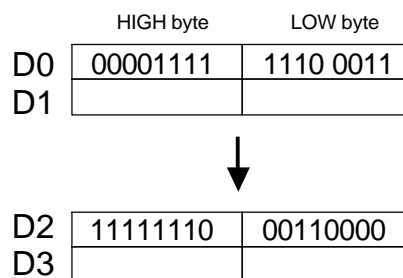


Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	
n												O	

Description

If M0:1 becomes ON, it rotates D0 to left by 4 bit. The bits of highest level give way (disappear), and the bits of lowest level are set to 0.



BSHR

Shift right by bit unit

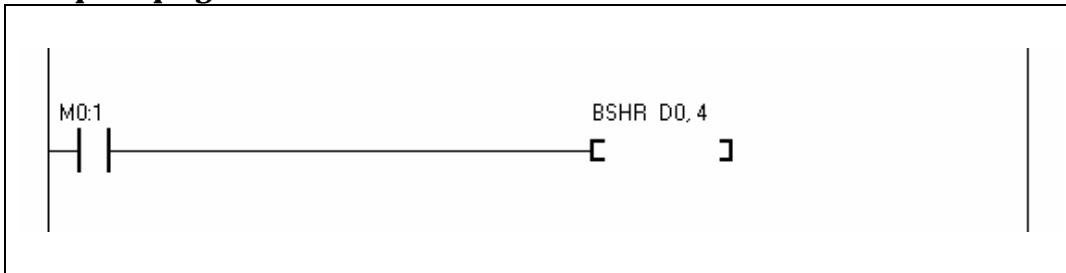
Usage : BSHR d,n

$d = d \gg n$

Summary

It is shift right instruction by bit unit. It shifts d to right by n bit. The newly inserted bit is set to 0.

Example of program

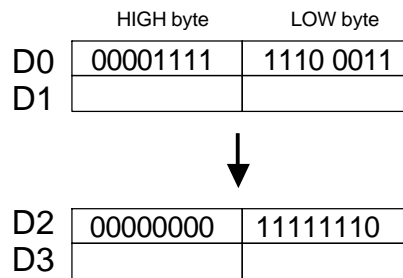


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates D0 to right by 4 bit. The bits of the highest level give way (disappear), and the bits of lowest level are set to 0.



WSHL

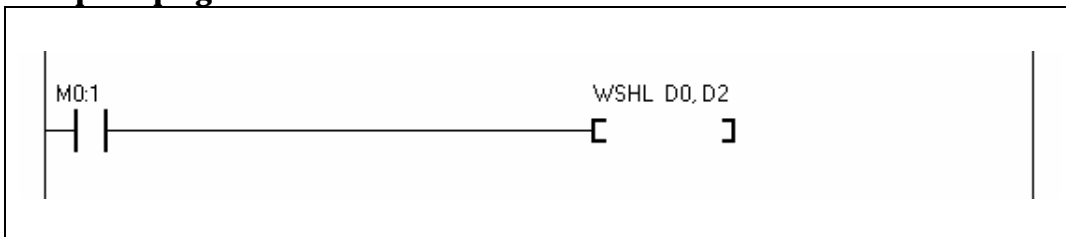
Shift left by word unit

Usage : BSHL s1,s2

Summary

It is shift left instruction by word unit. It shifts from s1 to s2 left by 1 word. The newly inserted word is set to 0.

Example of program

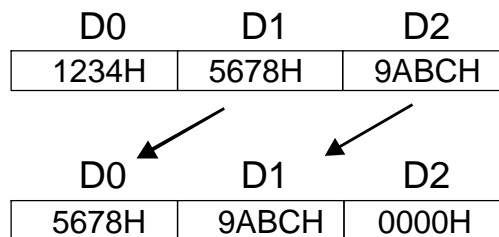


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S				C	T	D	AD	
s1,s2	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates from D0 to D0 left by 1 word. D0 disappears and D2 is set to 0.



WSHR

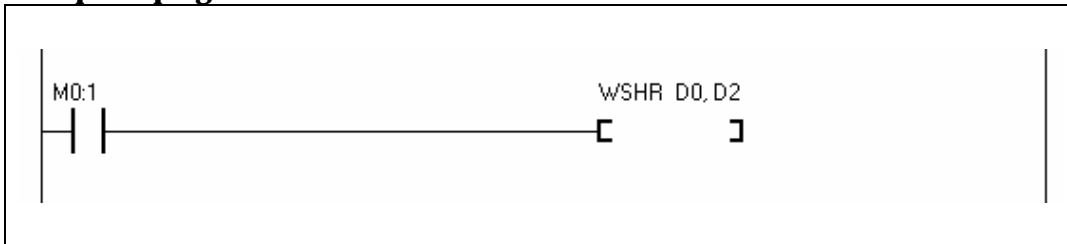
Shift right by word unit

Usage : BSHR s1,s2

Summary

It is shift right instruction by word unit. It shifts from s1 to s2 right by 1 word. The newly inserted word is set to 0.

Example of program

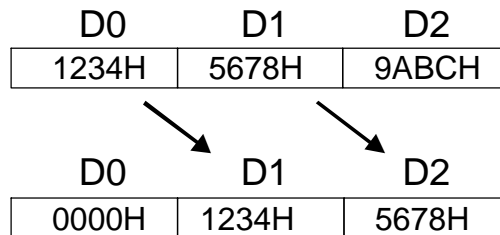


Operands

operand	relay					counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s1,s2	O	O		O		O	O	O	O	O	O	O	

Description

If M0:1 becomes ON, it rotates from D0 to D0 right by 1 word. D0 disappears and D2 is set to 0.



SEG

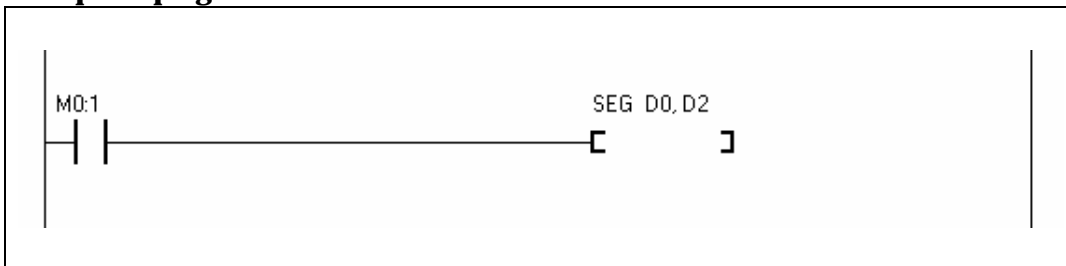
Convert data for 7 segment

Usage : SEG s,d

Summary

It converts 16bit data in s to 4 digit data for 7 segment.

Example of program

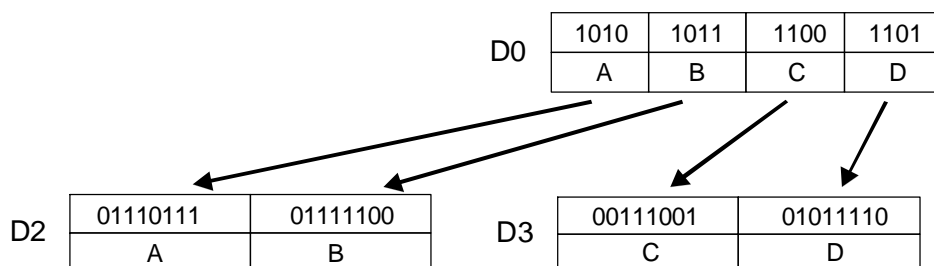


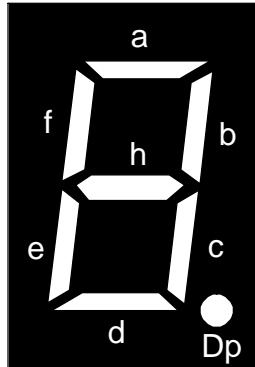
Operands

operand	relay					counte	timer	data	etc.				constan	
	P	M	F	K	S	r	C	T	D	AD	CNT	CH		G
s	O	O		O			O	O	O	O	O	O	O	O
d	O	O		O			O	O	O	O	O	O	O	

Description

It is data convert instruction for 7 segment. If M0:1 becomes ON, it stores the 16bit data in D0, after converting to 4digit data for 7 segment, by 2 words(4byte).





hexadecimal	Dp	g	f	e	d	c	b	a	By 7 segment
0	0	0	1	1	1	1	1	1	0
1	0	0	0	0	0	1	1	0	1
2	0	1	0	1	1	0	1	1	2
3	0	1	0	0	1	1	1	1	3
4	0	1	1	0	0	1	1	0	4
5	0	1	1	0	1	1	0	1	5
6	0	1	1	1	1	1	0	1	6
7	0	0	1	0	0	1	1	1	7
8	0	1	1	1	1	1	1	1	8
9	0	1	1	0	1	1	1	1	9
A	0	1	1	1	0	1	1	1	A
B	0	1	1	1	1	1	0	0	B
C	0	0	1	1	1	0	0	1	C
D	0	1	0	1	1	1	1	0	D
E	0	1	1	1	1	0	0	1	E
F	0	1	1	1	0	0	0	1	F

There are two kinds of method to drive segments in TinyPLC. First method is to connect segment directly to I/O port using SEG instruction, and the second is to use special SGN (Serial segment module which is marketed in Comfile)

It is efficient to use SEG instruction if there are only 1 or 2 segments to drive, but it also has the weak point of using many ports. If we use SGN module we can save the port because only 1 I/O port is used, and can control many segments by 1 line as 1 line can connect 8 SGN modules (7 segments of 40). (For more details of SGN, refer to SGNOUT instruction description)

LCDCLS

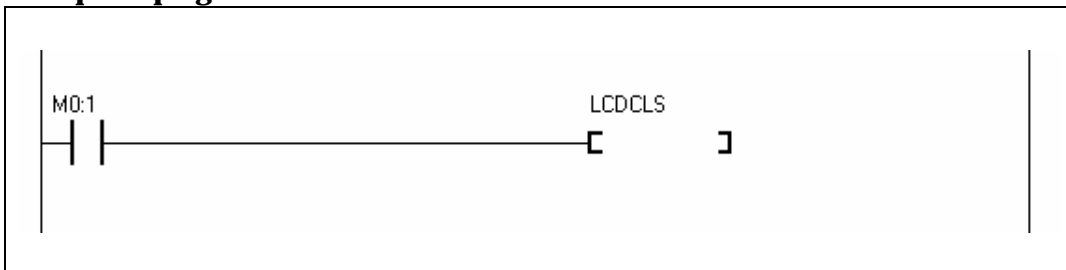
Clear LCD Screen

Usage : LCDCLS

Summary

Fill CH of LCD display area with 20H (SPACE in ASCII code).

Example of program



Description

Fill CH of LCD display area with 20H (SPACE in ASCII code).

There is no effect in LCD without LCDOUT instruction. It is the instruction of clearing only CH area.

LCDOUT

LCD transmission start

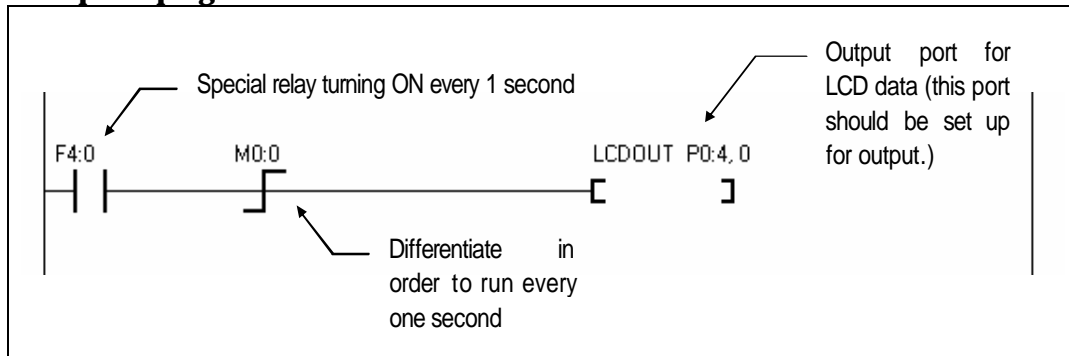
Usage : LCDOUT port, mode

Summary

Transmit data on CH area to serial module.

- Port describes port locations connected to a LCD module (e.g.: P0:4)
- Modes are types of connected LCD.
 - 0: 16 by 2 English LCD module (ELCD162)
 - 1: 16 by 4 English LCD module (ELCD164)
 - 2: 20 by 4 English LCD module (ELCD204)
 - 3: Korean LCD (HLCD112 or HLCD114)

Example of program



Description

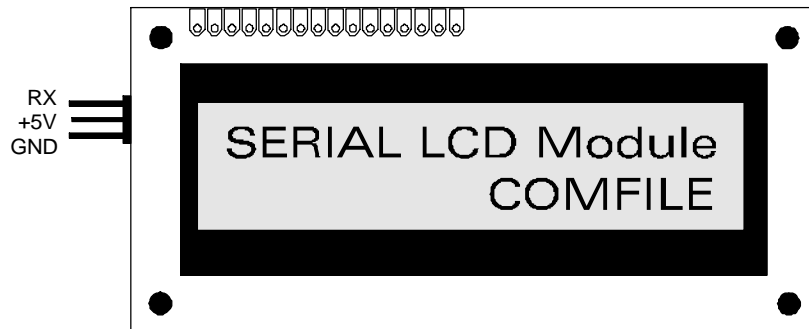
F4:0 is a special relay repeating ON/OFF every one second. And F4:0 differentiates and makes LCDOUT runs every one second. LCDOUT instruction transmits data on CH area to LCD. Once LCDOUT instruction is given, all data on CH area transmits to LCD. (It takes about 200mS~500mS)

Be careful not to transmit during other transmissions. Therefore, you have to use a method transmitting data every one second like above example or program to show when you press down a key.

Only serial LCD products made by Compile Co. can be used for LCD modules. Serial module should be set up in 19200 mode (It should be set up with jumpers.).

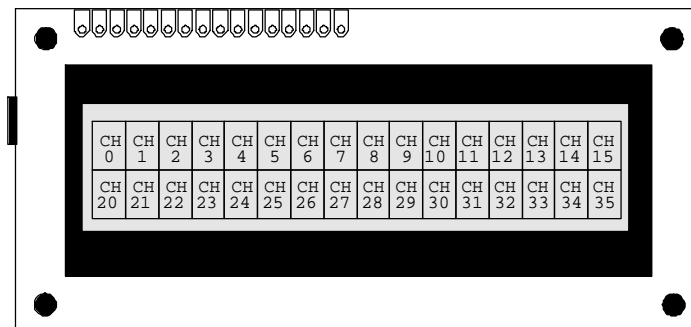
About serial English LCD modules...

A serial LCD modules manufactured by Compile Co. are different from other existing LCD modules in the way that it connects with three electric lines. (The existing way is a parallel connection with 14 electric lines.)



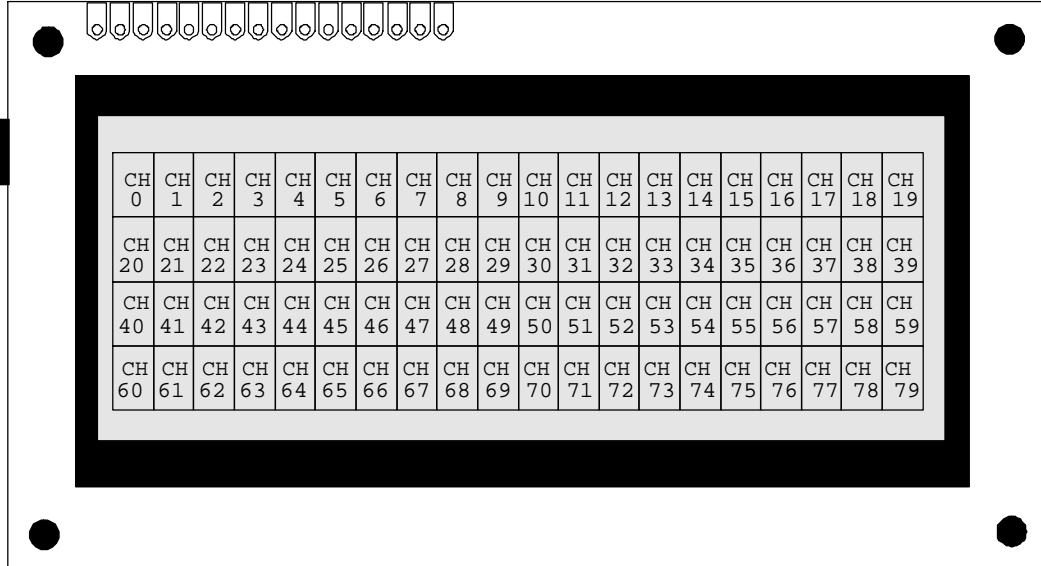
Two of three electric lines are power electric line (+5V, GND) and the rest transmits data. Data use RS232C of 5V level and 19200 baud rate and you can indicate letters on an designated position by transmitting ASCII cord. LCDOUT instruction of TinyPLC transmits all the data (ASCII cord) on CH area to RS232.

A position on LCD is closely related to H location. When you record on CH0, it appears first line on the left side. The following shows the relationship between 16 by 2 LCD and CH area.



The first line is assigned from CH0 and the second line is assigned from CH20. In using 16 by2, LCD doesn't show if you save data on CH40~79 area.

The following is the case of 20 by 4 lines. (20 by 4 LCD can show all the 80 bytes on CH area.)



The table below shows serial modules able to be connected to PLC.

Model	Product	Screen size (mm)
ELCD162	16 by 2 English LCD	64.5 * 13.8
ELCD162-BL	16 by 2 English LCD back right	64.5 * 13.8
ELCD164	16 by 4 English LCD	61.8 * 25.2
ELCD164-BL	16 by 4 English LCD back right	61.8 * 25.2
ELCD204	20 by 4 English LCD	76 * 25.2
ELCD204-BL	20 by 4 English LCD back right	76 * 25.2
ELCD162-BIG-BL	Double size 16 by 2 English LCD back right	99 * 36
HLCD112-BL	Korean LCD 11 by 2 line back right	60.5 * 18.5
HLCD114-BL	Korean LCD 11 by 4 line back right	70.7 * 38.8

Additional information

If you change output port, you can run several LCD modules simultaneously. In running several LCD simultaneously, you should make sure if output times (200~500mS) don't conflict with each other. (This is applied same to **SGNOUT instruction**.)

SGNOUT

SGN transmission start

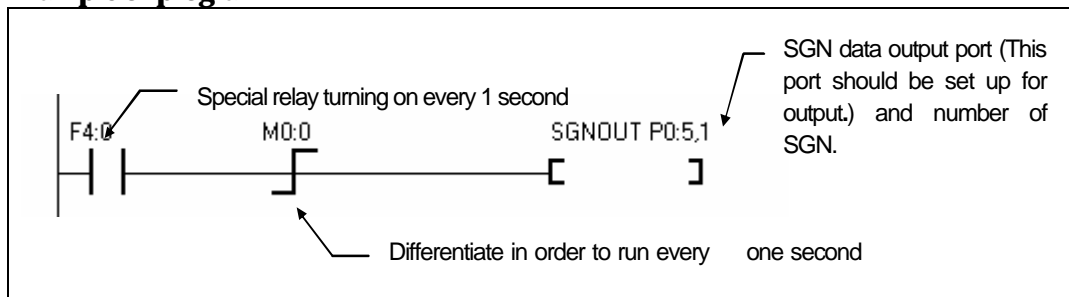
Usage : SGNOUT port, n

Summary

Data on G area is transmitted by serial seven segment module (SGN). Port describe the port location connected to a LCD module (Example P0:5), n is number of SGN module.

*SGN is the registered trademark of Compile Technology. (Seven seGment Network)

Example of program



Description

F4:0 is a special relay repeating ON/OFF every one second. And F4:0 differentiates and makes SNGOUT instruction run every one. SGNOUT instruction transmits data on G area to SGN. Once SGNOUT instruction is given, all the data on G area transmits to SGN. (It takes about 50mS~100mS)

Be careful not to transmit during other transmissions. Therefore, you have to use a method transmitting data every one second like above example or program to show when you press down a key.

Since maximum five seven segments are equipped at one SGN and maximum eight SGN modules can be connected to each other, on I/O port can control 40 seven segment at maximum.

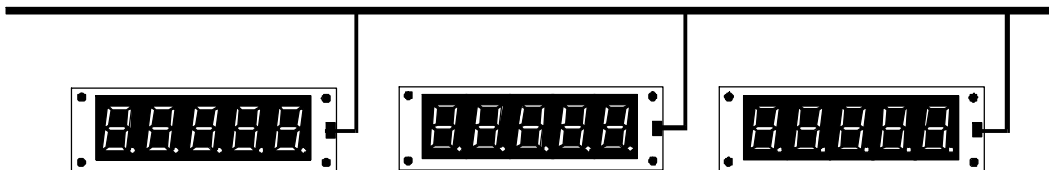
About SGN modules...

SGN module is a seven segment display module manufactured by Compile Technology. (SGN is an abbreviation of Seven segment Network) The dynamic display is generally used in operate seven segment, but it is not convenient in the way it needs many electric lines and good operating timing. SGN module equipped on PCB with the chip operating maximum five units of seven segment and it shows on the designated point when RS232C of 5V level and 9600 baud rate transmit data.

TinyPLC got SGN instruction which runs this kind of SGN modules and, it can be installed simply by connecting one of I/O ports to SGN.



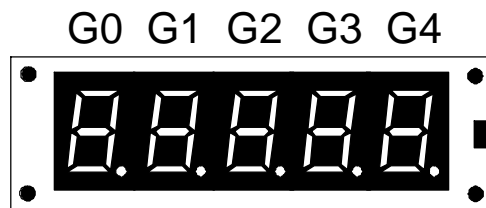
Separate ports are available for SGN module connections and one I/O port can control maximum eight SGN modules.



On the back side of SGN module is there DIP switches which set each address. G area is allocated according to DIP switch setting.

DIP switch	G area allocation
0 0 0 0	G0 ~ G4
0 0 0 1	G5 ~ G9
0 0 1 0	G10 ~ G14
0 0 1 1	G15 ~ G19
0 1 0 0	G20 ~ G24
0 1 0 1	G25 ~ G29
0 1 1 0	G30 ~ G34
0 1 1 1	G35 ~ G39

When one SGN module is used, location 0 on G area is allocated to first seven segment from the left.



Therefore, if 31H is on G0 (1 in ASCII cord) , 32H is on G1, 33H is on G2, 34H is on G3, 35H is on G4 and SGNOUT instruction is given, the result comes out as follows;



In order to output data on SGN, you should save the designated data (in ASCII cord) from G area by using ASC, HEX or STRING and then transmit it to SGNOUT.

The below SGN modules are available at present. (August, 1999)

Model	Product	Total size(mm)
SGN-S3	SGN SMALL 3 DIGIT module	80 * 19.4
SGN-S4	SGN SMALL 4 DIGIT module	80 * 19.4
SGN-S5	SGN SMALL 5 DIGIT module	80 * 19.4
SGN-M3	SGN MIDDLE 3 DIGIT module	163 * 42
SGN-M4	SGN MIDDLE 4 DIGIT module	163 * 42
SGN-M5	SGN MIDDLE 5 DIGIT module	163 * 42

You can also take a look at detailed specification of SGN modules and LCD modules (life-size photos) on our homepage [http:// www.comfile.co.kr](http://www.comfile.co.kr).

STRING

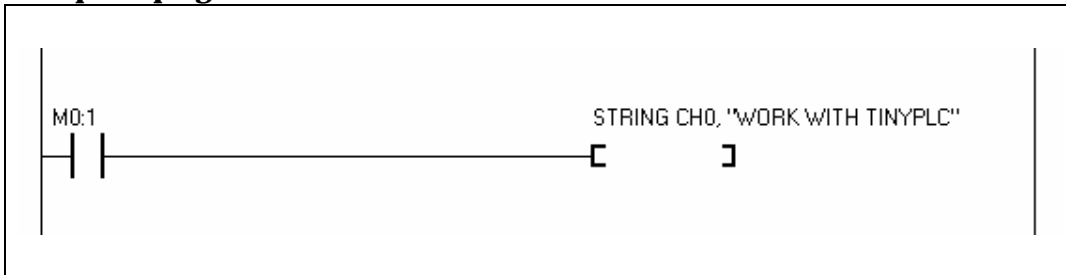
characters line saving

Usage : STRING d,"string"

Summary

Save character line "string" to d according to the number of line.

Example of program



Operands

Offer land	Relay					Counte	Timer	Data	Others				Coeff.	
	P	M	F	K	S	r	C	T	D	AD	CNT	CH		G
d									O			O	O	

Description

This is a character line saving instruction which displays letters on LCD or SGN. When M0:1 is ON, save "WORK WITH TINYPLC" line in ASCII cord to CH0.

CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10	CH11	CH12	CH13	CH14	CH15	CH16	CH17
W	O	R	K		W	I	T	H		T	I	N	Y	P	L	C	
57	4F	52	4B	20	57	49	54	48	20	54	49	4E	59	50	4C	43	

HEX

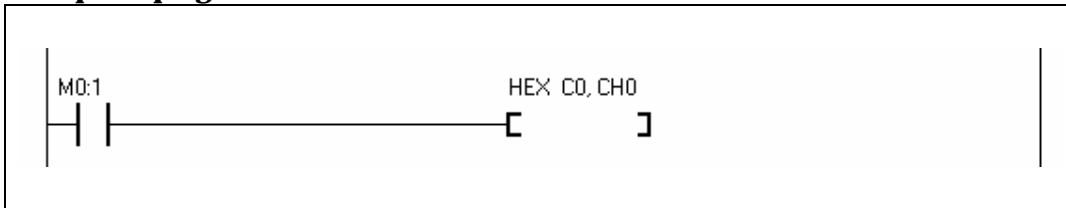
Convert 16bit -> hexadecimal ASCII

Usage : HEX s,d

Summary

It converts 16bit binary value to hexadecimal ASCII code.

Example of program

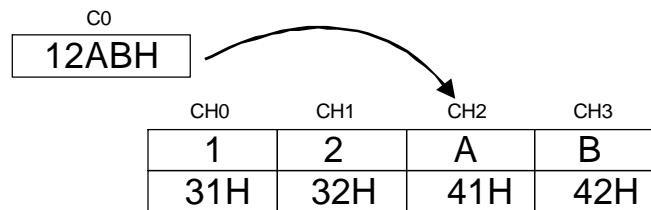


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s	O	O	O	O		O	O	O	O	O	O	O	O
d								O			O	O	

Description

It is the instruction of converting to ASCII code by which letters can be displayed at LCD or SGN. If M0:1 becomes ON, it stores C0 value, after converting to hexadecimal ASCII code, from CH0.



DHEX

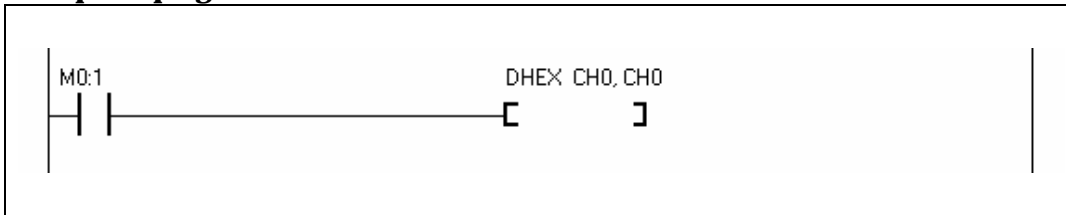
Convert 32bit -> hexadecimal ASCII

Usage : DHEX s,d

Summary

It converts 32bit binary value to hexadecimal ASCII code.

Example of program

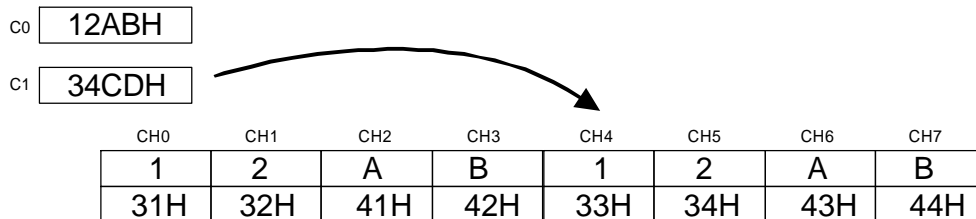


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s	O	O	O	O		O	O	O	O	O	O	O	O
d								O			O	O	

Description

It is the instruction of converting to ASCII code by which letters can be displayed at LCD or SGN. If M0:1 becomes ON, it stores C0, C1 value (double word), after converting to hexadecimal ASCII code, from CH0.



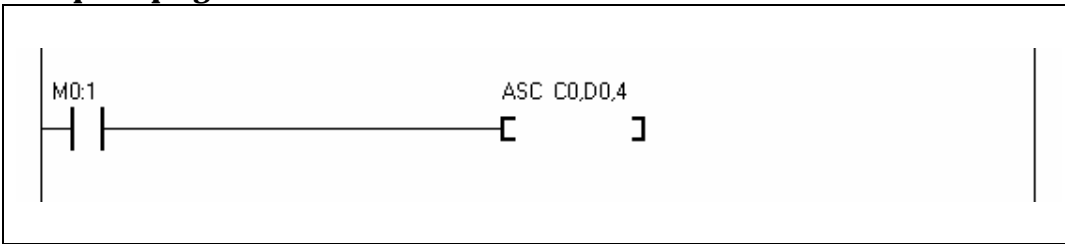
ASC Convert 16 bit-> decimal ASCII

Usage : ASC s,d,n

Summary

It converts 16bit binary value to decimal ASCII code.

Example of program

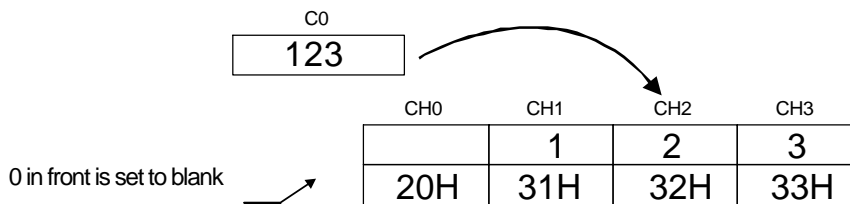


Operands

operand	relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s	O	O	O	O		O	O	O	O	O	O	O	O
d								O			O	O	
n (number converted)													O

Description

It is the instruction of converting to ASCII code by which letters can be displayed at LCD or SGN. If M0:1 becomes ON, it stores C0 value, after converting to decimal ASCII code, from CH0 by 4 byte. (0 in front is set to blank.)



DASC

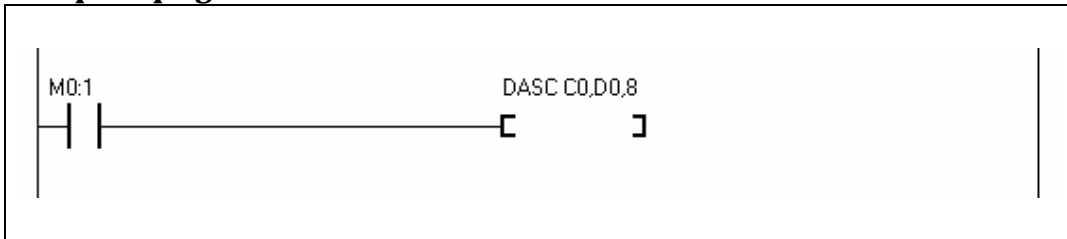
Convert 32 bit -> decimal ASCII

Usage : DASC s,d,n

Summary

It converts 32bit binary value to decimal ASCII code.

Example of program

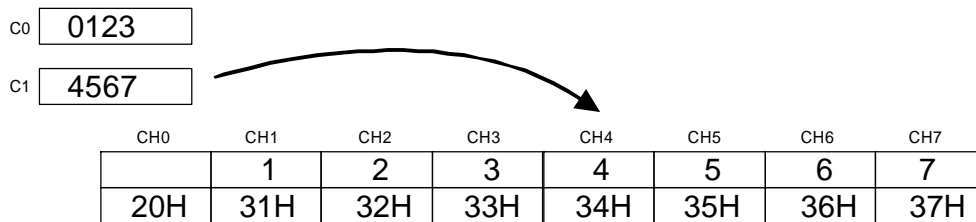


Operands

operand	relay					counter timer data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s	○	○	○	○		○	○	○	○	○	○	○	○
d								○			○	○	
n (number converted)													○

Description

It is the instruction of converting to ASCII code by which letters can be displayed at LCD or SGN. If M0:1 becomes ON, it stores C0, C1 value (double word), after converting to decimal ASCII code, from CH0 by 8 byte. (0 in front is set to blank.)



GOTO, LABEL

Jump and Label

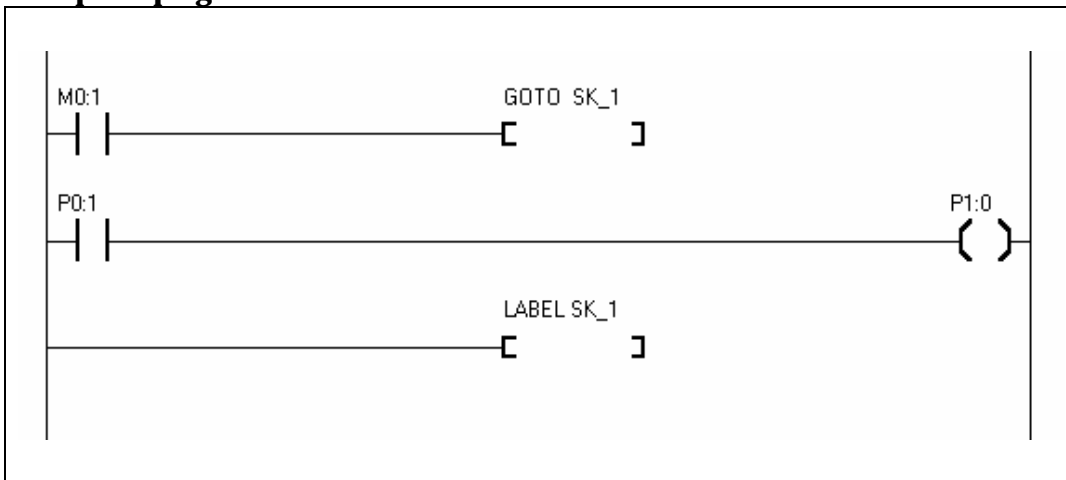
Usage : GOTO label

Usage : LABEL label

Summary

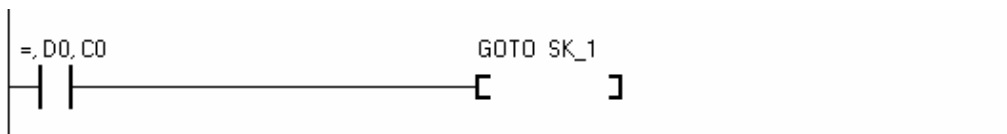
It is the transfer instruction. It jumps to the label which GOTO indicated. LABEL is the instruction which declares the label.

Example of program



Operands

If M0:1 becomes ON, it jumps to SK_1. Using as follows, you can use it as conditional branch instruction.
(If D0 = C0, it jumps to SK_1.)

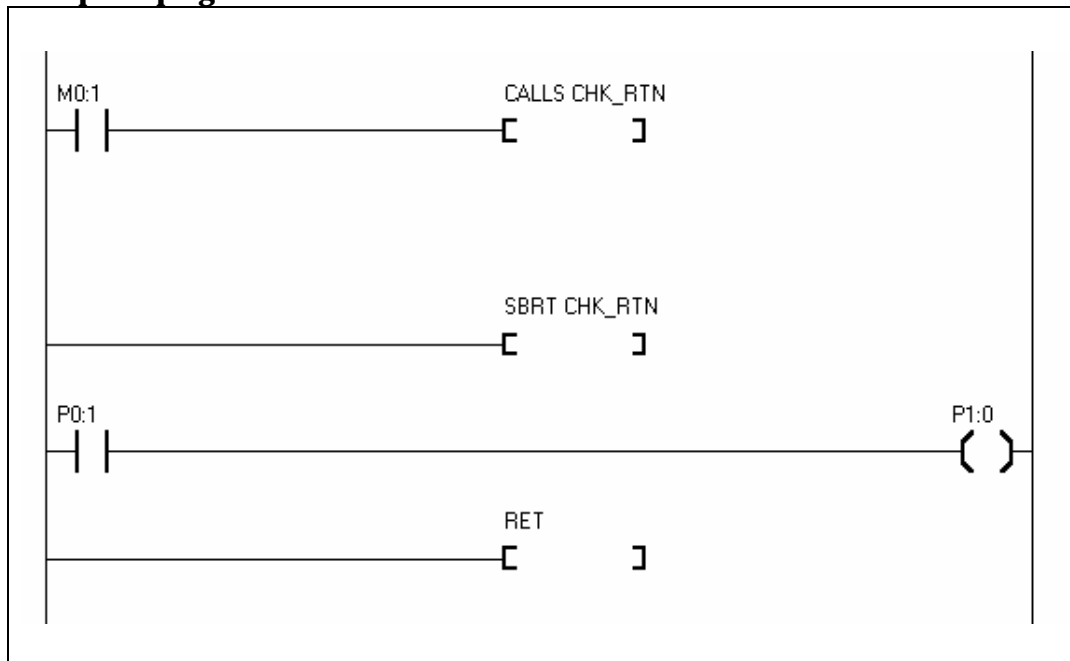


CALLS, SBRT	subroutine call, subroutine start
RET	subroutine return
Usage : CALLS label	
Usage : SBRT label	

Summary

It is instruction related with subroutine. CALLS calls subroutine, SBRT defines subroutine. RET should be placed end of subroutine.

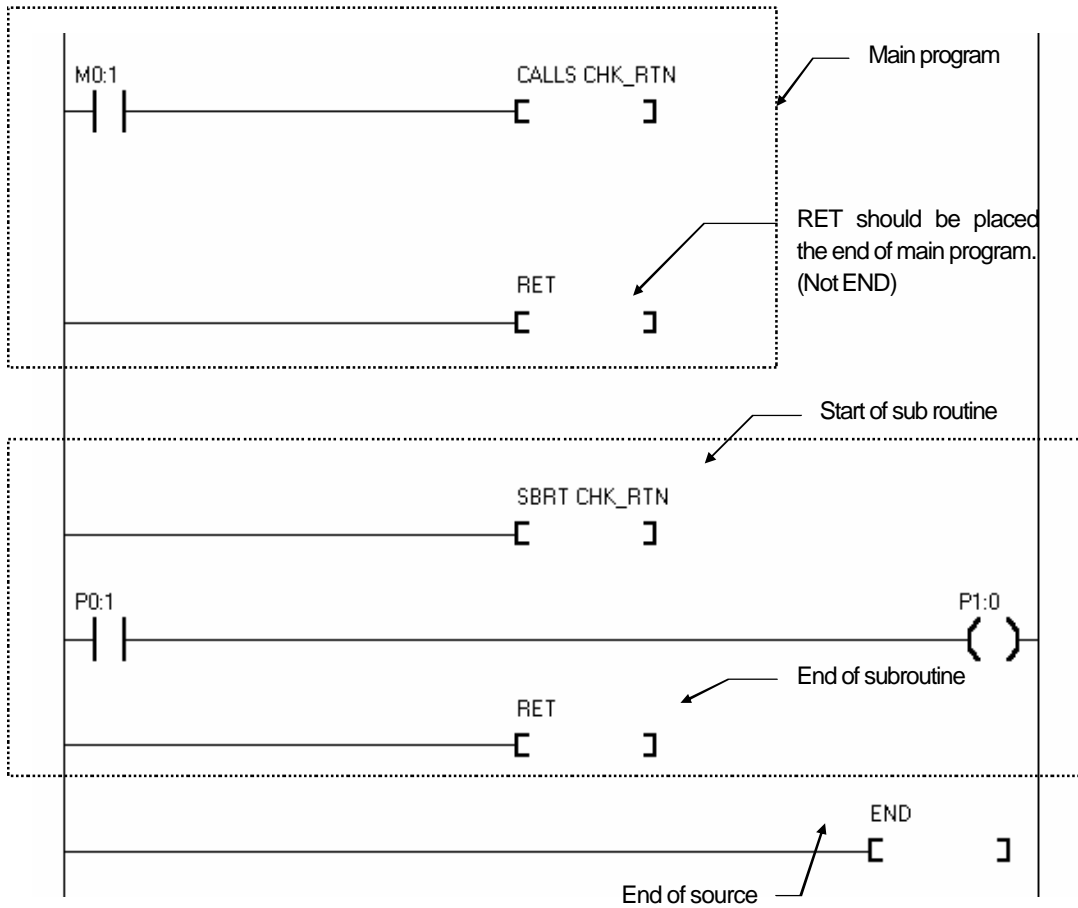
Example of program



Description

If M0:1 become ON, it calls CHK_RTN. Nesting execution of CALL is possible to maximum 16level. (Nesting execution means calling another subroutine inside of subroutine.)

Please follow procedure as below, in case of making main routine and subroutine.



Please notice that END instruction in Tiny PLC simply means the end of source program. You should write RET instruction at the end of main routine.

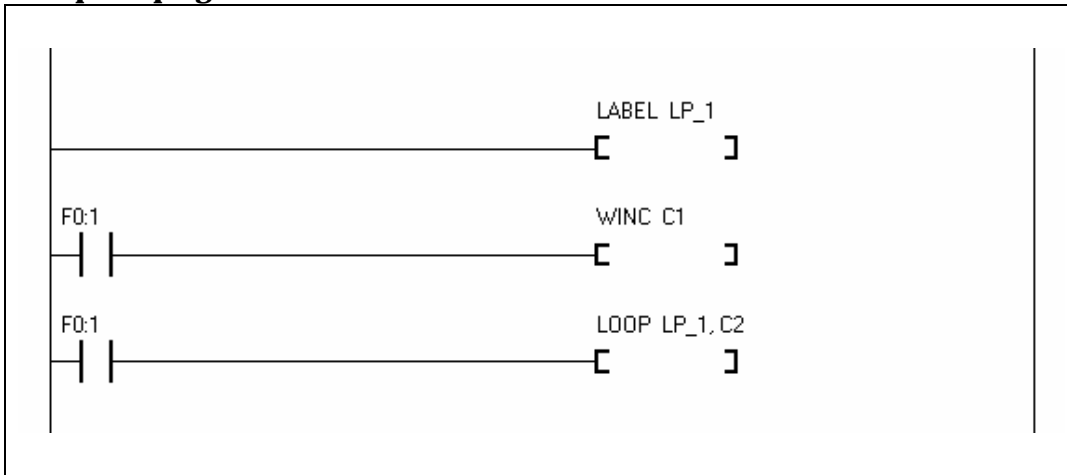
LOOP iterative execution process

Usage : LOOP label, d

Summary

It jumps to label until d value become 0.

Example of program



Operands

operand	Relay					counter	time	data	etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d						O	O	O					O

Description

LOOP instruction executed unconditionally because F0:1 is always ON point of contact. It jumps to LP_1 until C2 value becomes 0.

KEYSCAN

Keyscan input

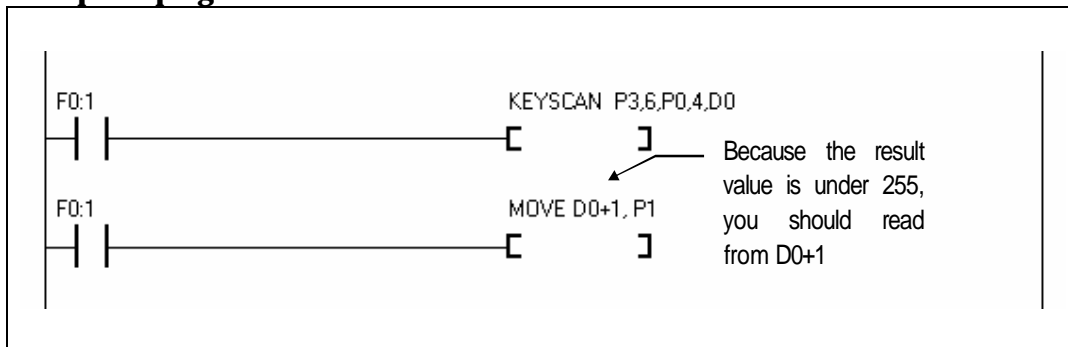
Usage : KEYSCAN a,b,c,d,e

Summary

This instruction reads key matrix in the way of scanning.

- a : byte number of input port (e.g. : P3)
- b : number of input port
- c : byte number of output port (e.g. : P0)
- d : number of output port
- e : location on which scan cord is saved (saved in word file)

Example of program



Operands

Offer land	Relay					Counter	Timer	data	Others				Coeff.
	P	M	F	K	S				C	T	D	AD	
d						O	O	O	O	O	O	O	

Description

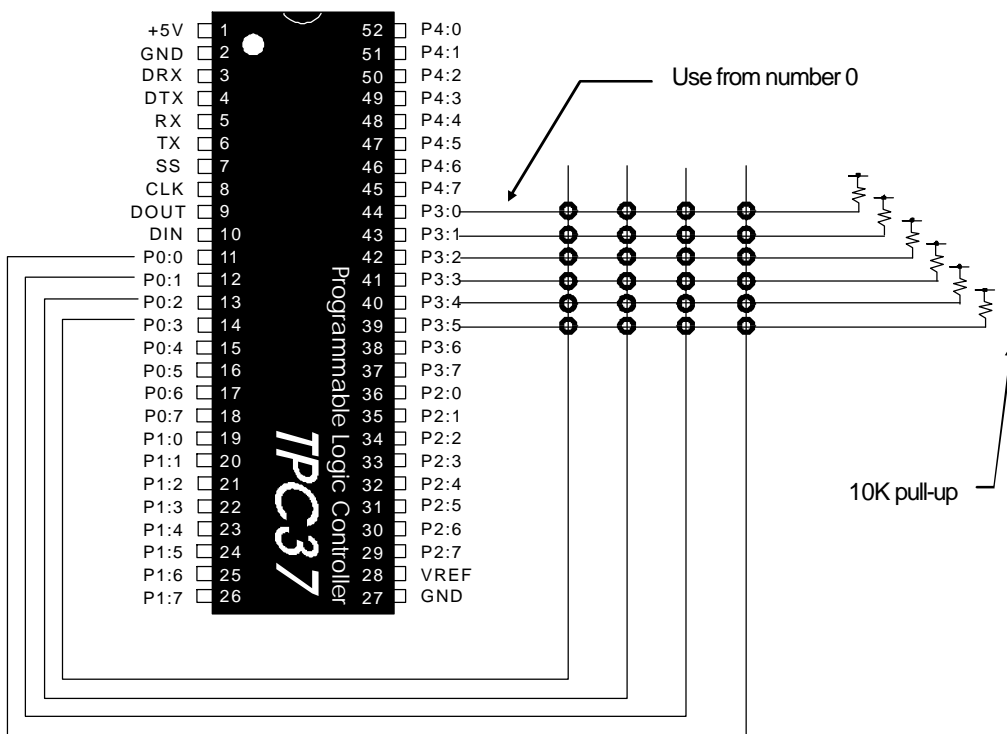
Since F0:1 is always ON, KEYSCAN instruction is absolutely operated. Save the presently pressed key on D0 with 6 * 4 key matrixes connected to P3 and P0 port. The following MOVE instruction appears the value of KEYSCAN cord that is saved on D0 on port 1. If port1 is all connected to LED, pressed scan key cord can be seen easily.

This instruction is made to read membrane keyboard recorded in matrix way. (The membrane keyboard is generally used for exclusive machines on automation field.) There are a few hardware precautions in order to run this instruction.

1. Locate from port 0. In other words, when you use P0, connect from P0:0.
2. Connect input port with pull-up resistance 10K Ohm.

KEYSCAN instruction can read 8* 8 key matrix at maximum and the result is saved in word size, but actual value does not exceed 255. Without key input, the result will be 0. If only about 6 * 4 is connected, the rest ports of relevant byte is only available only for input port. (If they are used for output ports, irrelevant data will come out.)

The following diagram shows 6 * 4 key matrix equipped on TinyPLC module.



OUTOFF

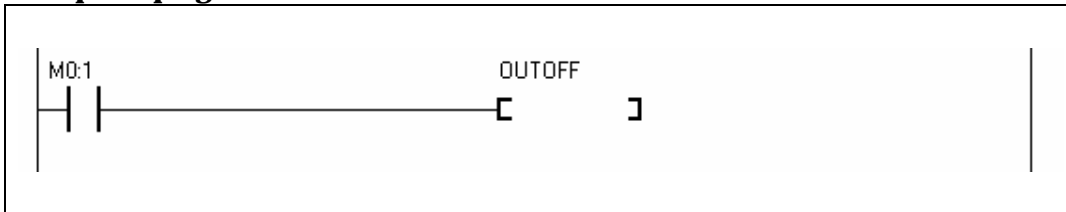
alloutput OFF

Usage : OUTOFF

Summary

It makes output from all I/O port, which is set to output, OFF.

Example of program



Description

If M0:1 become ON, all output become OFF

DIST

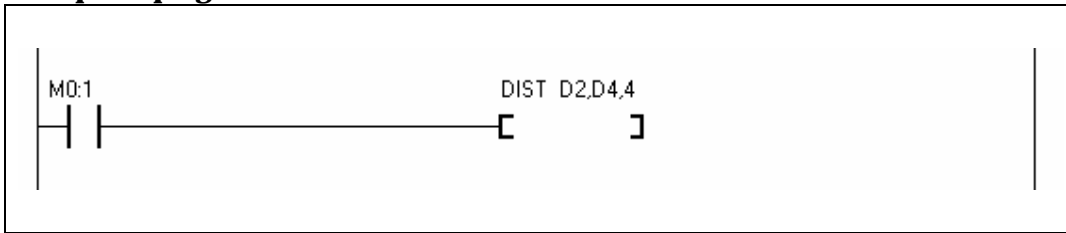
disperse 16 Bit

Usage : DIST s,d,n

Summary

It divides 16 Bit value into 4 Bit value and deposits d area.

Example of program

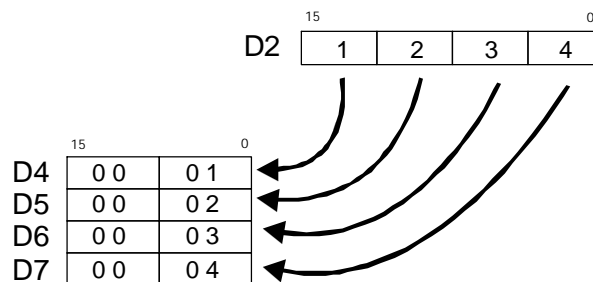


Operands

operand	Relay					counter C	time T	data D	etc.				constant
	P	M	F	K	S				AD	CNT	CH	G	
s	O	O	O	O		O	O	O	O	O	O	O	O
d						O	O	O					
n						O	O	O					O

Description

If M0:1 become ON, it divides data in D2 into 4 Bit (nibble unit) and deposits from D4 to D7. (It can designate the numbers which can be divided by n. n should be smaller than 5.)



UNIT

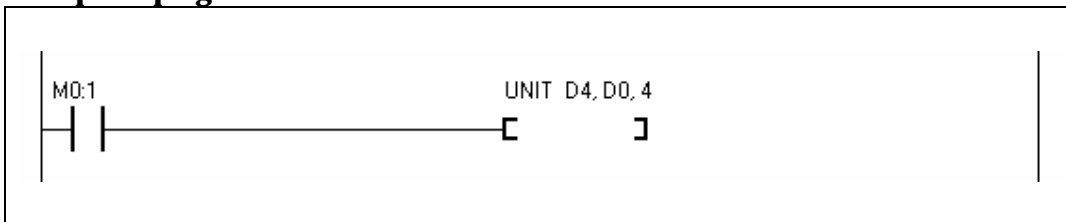
16 Bit combination

Usage : UNIT s,d,n

Summary

It combines data area (maximum 4 words) where only the lower 4bit exists, and makes one word. (It combines the value, which dispersed by DIST instruction.)

Example of program

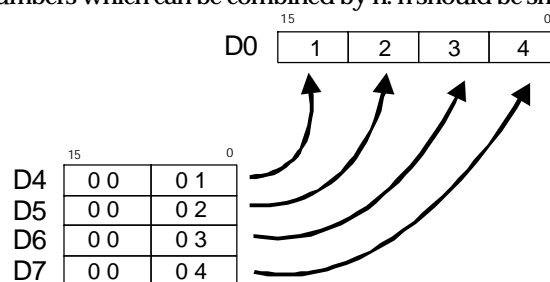


Operands

operand	Relay					counter	time r	data D	etc.				constant
	P	M	F	K	S				AD	CNT	CH	G	
s						O	O	O					
d						O	O	O					
n						O	O	O					O

Description

If M0:1 become ON, it combines only the lower nibble from D4 to D7 and makes one word and deposit to D0. (It can designate the numbers which can be combined by n. n should be smaller than 5.)



ENCO

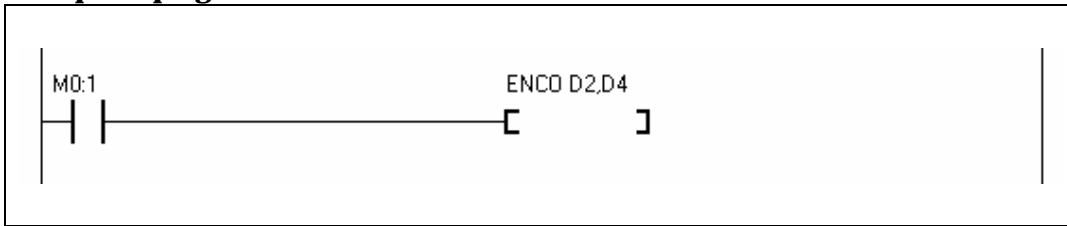
16bit encoder

Usage : ENCO s,d

Summary

It encodes the lower 4 bit of s and deposit to d.

Example of program

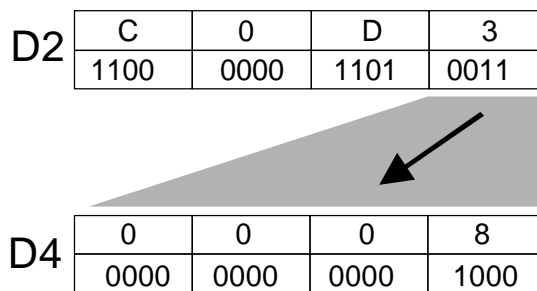


Operands

operand	Relay					counter	time r	data D	etc.				constant
	P	M	F	K	S				AD	CNT	CH	G	
s	O	O		O		O	O	O	O	O	O	O	O
d						O	O	O					

Description

If M0:1 become ON, it encode the lower 4 bit of D2 and deposit to D4. (Please use bit shift instruction BSHR together to encode not the lower bit but another bit.)



DECO

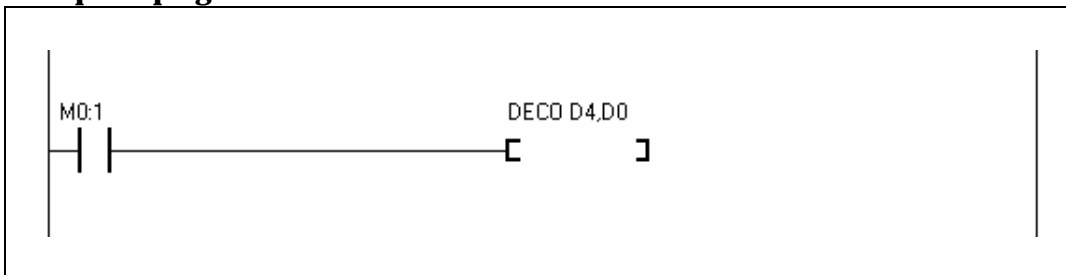
16 bit decoder

Usage : DECO s,d

Summary

It decodes the lower 4 bit of s and deposit to d.

Example of program

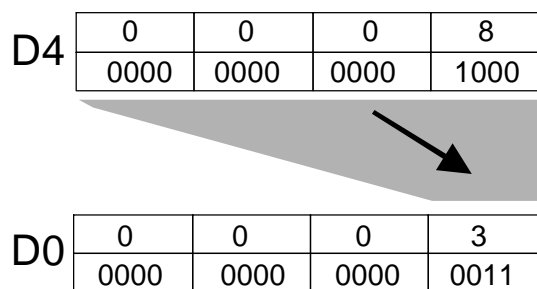


Operands

operand	Relay					Counter time data			etc.				constant
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
s	○	○		○		○	○	○	○	○	○	○	○
d						○	○	○					

Description

If M0:1 become ON, it decode the value of D4 and deposit D0. (Please use bit shift instruction BSHL, BSHR together to move decoded data to another bit column.)



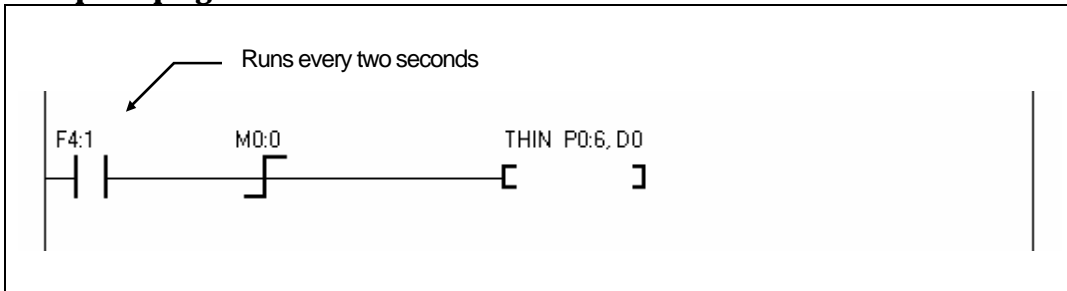
THIN
DS1820 thermal sensor input

Usage : THIN port,d

Summary

It reads a present temperature from the digital thermal sensor DS1820 connected to an indicated port and saves it to d.

Example of program



Operands

Offer land	Relay					counter	Timer	Data	Others				Coeff.
	P	M	F	K	S	C	T	D	AD	CNT	CH	G	
d								O					

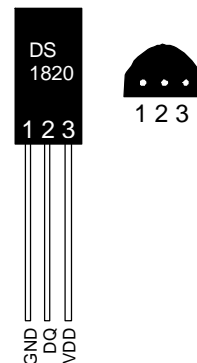
Description

F4:1 repeats On/Off every two second. If this signal is differentiated with DF, 1 scan pulse occurs every two seconds. As the result, THIN runs every two seconds. THIN reads a temperature from DS1820 connected to P0:6 port and save it to D0.

- DS1820 has a thermal sensor internally.
- THIN should run every one or two seconds. (Thermal conversion in DS1820 takes about one second.)
- Measurable temperature ranges from -55 to +125 . (Notch mark in at every 0.5)
- Accuracy : ±1 within 0 ~70
- The port connected to DS1820 should be set up for output. (Two way port should be connected to DS1820. **On TPC37, P1 and P3 cannot be used.**)

About DS1820...

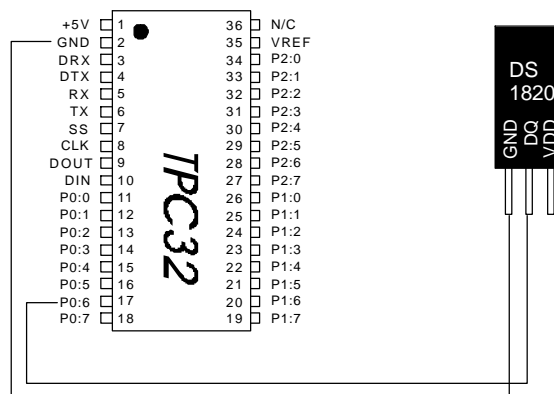
DS1820 is a one wire digital temperature sensor chip manufactured by Dallas Co. It can read a present temperature of a measured point simply by using main equipment and one line (two lines including ground). Since it transmits an actual temperature in binary type, receivers do not have to calculate or transform and use it as it is.



Temperature	Digital input (HEX)	Digital output (DEC) Negative integers are 2's complement
+125 °C	00FAH	250
+25 °C	0032H	50
+0.5 °C	0001H	1
0 °C	0000H	0
-0.5 °C	FFFFH	-1
-25 °C	FFC3H	-50
-55 °C	FF92H	-110

If it is divided by 2, then the quotient is a real value.

DS1820 is shaped as TR and can be installed immediately on designated point (just like PT100 thermistor) since you only need two lines. (It can be extended two meter at maximum.) Actually there are three lines, but VDD line is not used. (Power is supplied through DQ line.) If you appoint ports differently, you can use several DS1820 simultaneously. The below diagram show how you connect DS1820 to TinyPLC (When P0:6 is used)



HCNTCLR

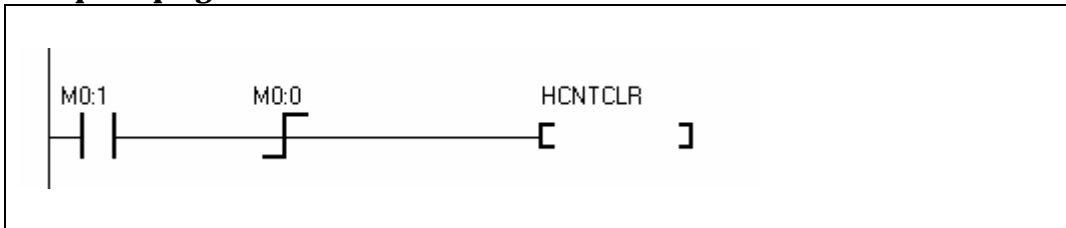
Clear High speed counter

Usage : HCNTCLR

Summary

It clears high speed counter area (CNT)

Example of program



Description

If M0:1 become ON, high-speed counter become RESET.