PLC

Aplikasi Pemrograman (Diagram Ladder/MELSEC Seri iQ-R)

Kursus ini dirancang bagi pengguna yang telah memahami dasar-dasar pengontrol terprogram MELSEC Seri iQ-R dan ingin mempelajari langkah pemrograman selanjutnya.

L(CTS)00687IND

Pendahuluan

Tujuan kursus

Kursus ini dirancang bagi pengguna yang telah menyelesaikan Dasar-dasar Pemrograman (Diagram Ladder) atau menguasai pengetahuan yang setara. Kursus ini memberikan pengetahuan tentang pemrograman dan debugging yang efisien untuk pengontrol terprogram MELSEC Seri iQ-R, penggunaan tingkat lanjutan untuk perangkat, dan pemrograman menggunakan label.

Sebagai prasyarat untuk kursus ini, Anda harus sudah menyelesaikan kursus berikut atau menguasai pengetahuan yang setara.

- Dasar-dasar MELSEC Seri iQ-R
- Dasar-dasar Pemrograman

Pendahuluan

Struktur kursus

Berikut adalah daftar isi kursus.

Bab 1 - Pemrograman yang efisien

Metode dan pengaturan untuk pemrograman yang efisien

Bab 2 – Pemrograman tingat lanjut

Penggunaan tingkat lanjut dari perangkat dan pemrogaman menggunakan label.

Bab 3 - Debugging yang efisien

Penggunaan Fungsi-fungsi software yang digunakan untuk debugging yang efisien

Tes Akhir

Nilai kelulusan: 60% atau lebih tinggi diharuskan

Pendahuluan **Cara menggunakan alat e-Learning ini**

Buka halaman berikutnya	>	Buka halaman berikutnya.
Kembali ke halaman sebelumnya	<	Kembali ke halaman sebelumnya.
Beralih ke halaman yang diinginkan	TOC	"Daftar Isi" akan ditampilkan, memungkinkan Anda untuk menavigasi ke halaman yang diinginkan.
Keluar dari kursus	X	Keluar dari kursus.

Perhatian untuk penggunaan

Petunjuk keselamatan

Apabila Anda belajar menggunakan produk aktual, bacalah dengan saksama tindakan pencegahan dalam panduan yang terkait.

Petunjuk keselamatan dalam kursus ini

Layar yang ditampilkan pada versi perangkat lunak yang Anda gunakan mungkin berbeda dengan yang ada di dalam kurus ini.
Kursus ini menggunakan versi perangkat lunak berikut:

- GX Works3 Versi 1.044W

Bab 1 Pemrograman yang efisien

Bab ini menjelaskan pengaturan software untuk pemrograman yang efisien dan dasar-dasar pemrograman menggunakan label.

1.1 Pemanfaatan program dalam sistem yang berbeda secara lebih mudah

1.2 Menyetel area memori sesuai dengan status penggunaan device

1.3 Menggunakan nama label yang berkaitan dengan aplikasi

1.4 Meningkatkan keterbacaan program

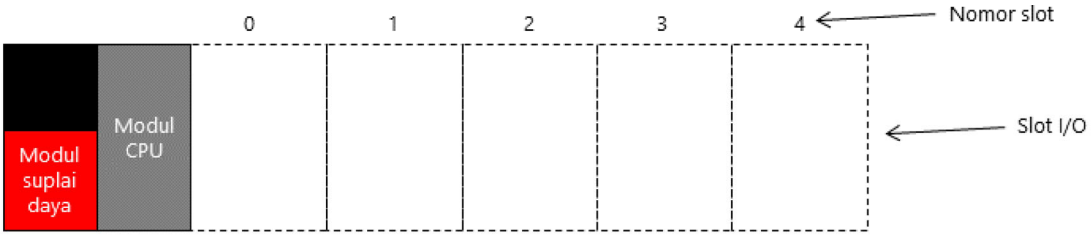
1.1 Pemanfaatan program dalam sistem yang berbeda secara lebih mudah

Bagian ini menjelaskan pemilihan nomor I/O yang efisien untuk pemanfaatan program dalam sistem yang berbeda secara lebih mudah.

1.1.1 Pemilihan nomor I/O otomatis

Nomor I/O yang dipilih secara sekuensial untuk modul yang dipasang pada base unit, dimulai dari slot yang terdekat dengan modul CPU.
Nomor I/O ditetapkan dalam unit 16 poin (0 hingga F).

Nomor poin yang tersedia untuk ditetapkan (diisi) bervariasi bergantung pada tipe modul (16, 32, 64, dan seterusnya).



Dalam contoh berikut, terpasang lima modul 16 poin.

1.1.1 Pemilihan nomor I/O otomatis

Ketika modul dengan 16, 32, dan 64 poin yang terisi digunakan pada saat yang sama, nomor I/O ditetapkan sebagai berikut:

		0	1	2	3	4
Modul suplai daya	Modul CPU	16 poin 00 hingga 0F	32 poin 10 hingga 2F	64 poin 30 hingga 6F	32 poin 70 hingga 8F	16 poin 90 hingga 9F

Jika terdapat slot kosong di antara modul yang terpasang, nomor I/O juga ditetapkan ke slot yang kosong.

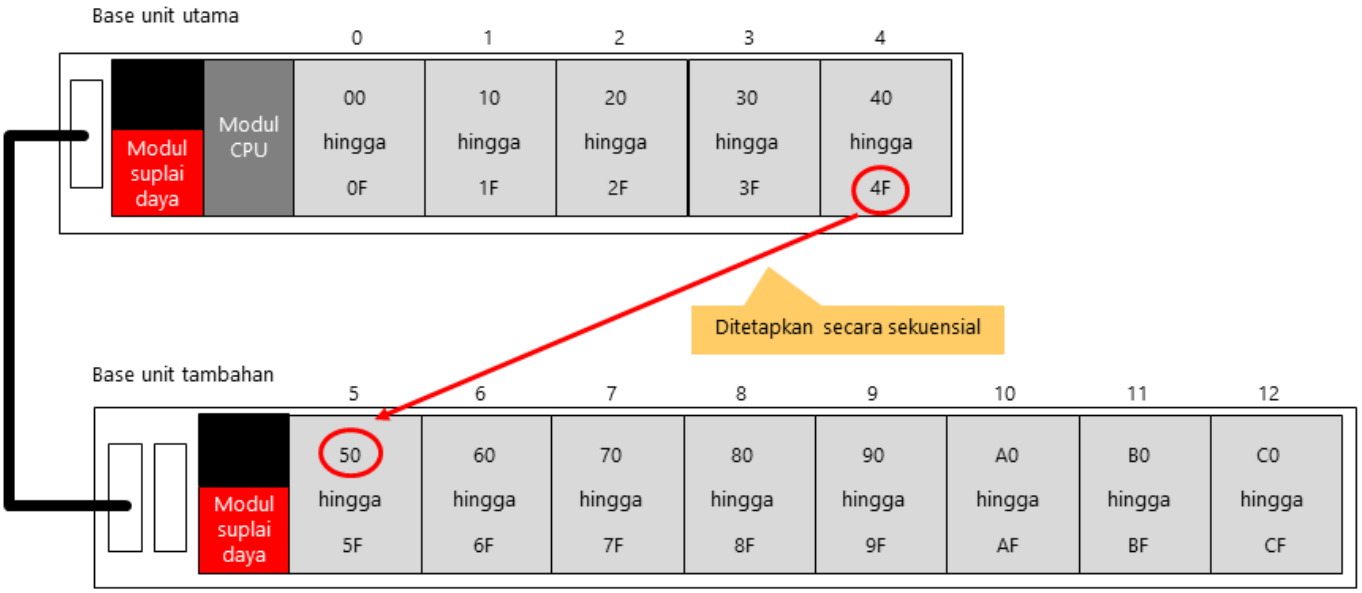
		0	1	2	3	4
Modul suplai daya	Modul CPU	16 poin 00 hingga 0F	32 poin 10 hingga 2F	64 poin 30 hingga 6F	16 poin 70 hingga 7F	16 poin 80 hingga 8F

Slot kosong

Secara default, 16 poin ditetapkan ke slot kosong. Jumlah poin yang ditetapkan dapat diubah dengan pengaturan parameter dalam rentang antara 0 sampai 1024 poin (dalam satuan 16 poin).

1.1.1 Pemilihan nomor I/O otomatis

Nomor I/O base unit tambahan ditetapkan secara otomatis, dengan mengikuti nomor I/O terakhir base unit utama.

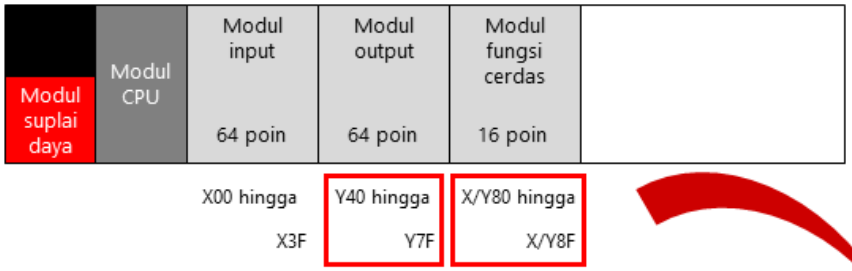


1.1.2 Pemilihan nomor I/O tetap

Jika nomor I/O ditetapkan secara manual, nomor I/O yang dipilih bersifat tetap dan tidak bisa diubah meskipun konfigurasi modul diubah. Hal ini berarti bahwa program kontrol yang sama dapat digunakan untuk kontrol yang sama tanpa memperhatikan konfigurasi modul.

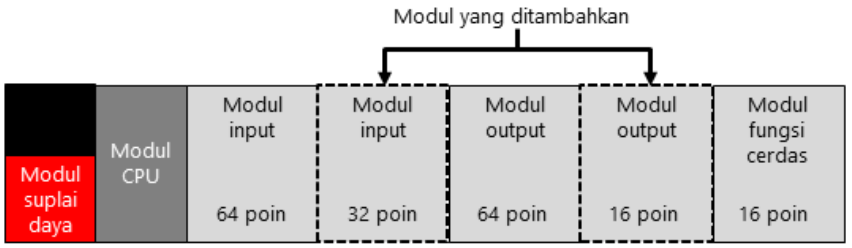
Penetapan otomatis

Sebelum modul ditambahkan



X00 hingga X3F	Y40 hingga Y7F	X/Y80 hingga X/Y8F

Setelah modul ditambahkan
(Modul input 32 poin dan modul output 16 poin ditambahkan.)



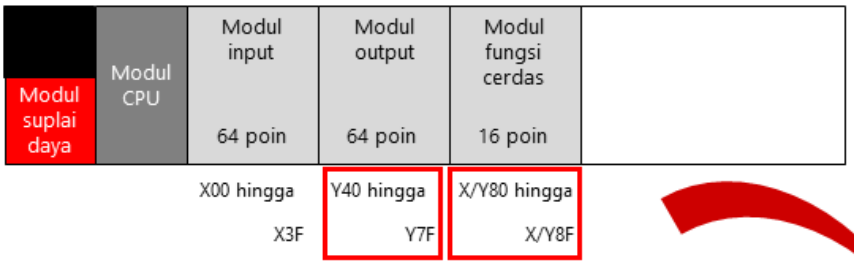
X00 hingga X3F	X40 hingga X5F	Y60 hingga Y9F	YA0 hingga YAF	X/YB0 hingga X/YBF

Nomor I/O ditetapkan kembali setelah penambahan modul

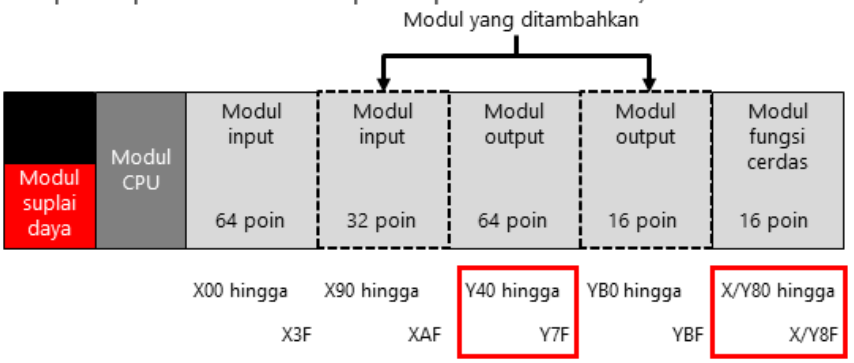
1.1.2 Pemilihan nomor I/O tetap

Penetapan manual untuk nomor I/O yang tetap

Sebelum modul ditambahkan



Setelah modul ditambahkan
(Modul input 32 poin dan modul output 16 poin ditambahkan.)

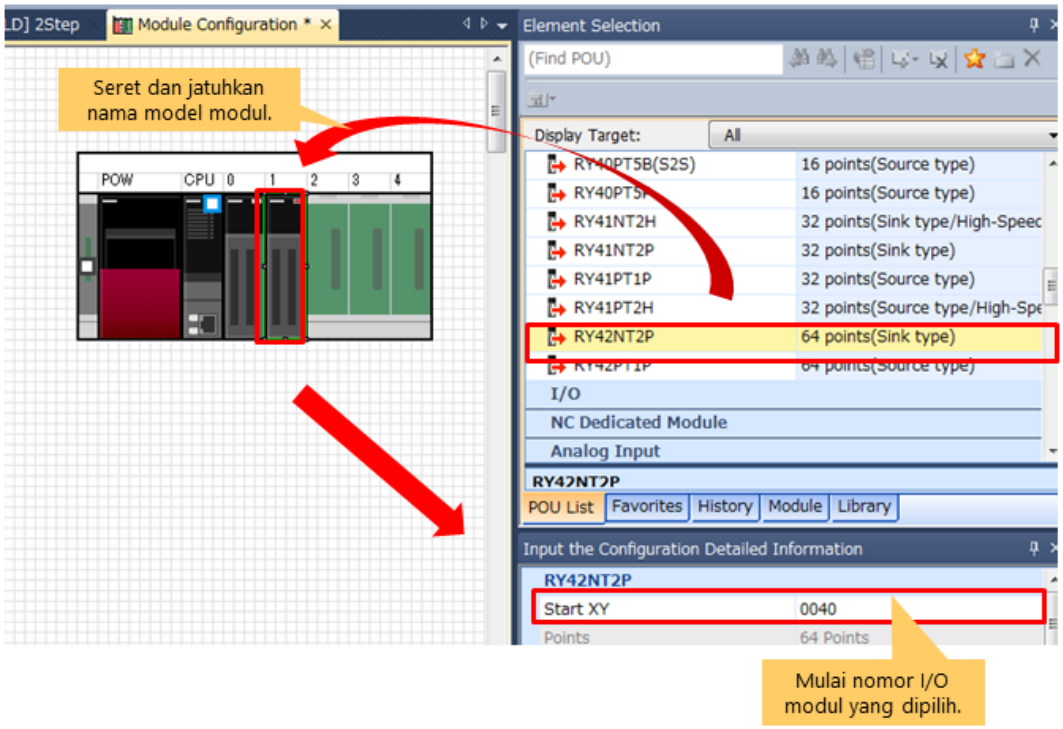


Nomor I/O yang sama ditetapkan terlepas dari penambahan modul.

Karena nomor I/O modul yang sudah ada tetap tidak berubah, hanya program yang berkaitan dengan modul yang ditambahkan yang perlu ditambahkan atau dimodifikasi.

1.1.3 Pemilihan nomor I/O otomatis menggunakan diagram konfigurasi modul

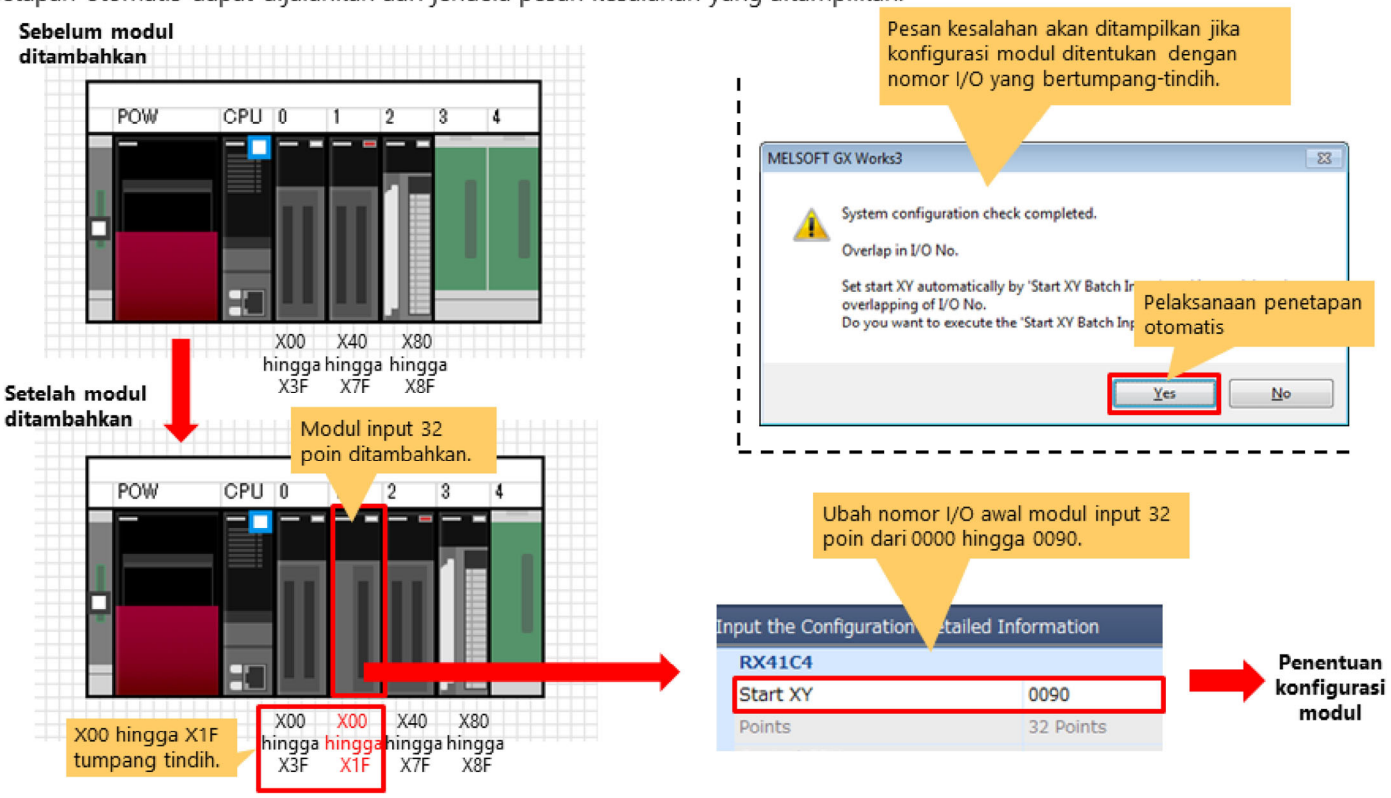
Konfigurasi modul dapat diatur menggunakan diagram konfigurasi modul pada software, MELSOFT GX Works3.
Pilih nama model modul, kemudian seret dan jatuhkan di atas slot untuk meletakkan modul pada slot tersebut.
Nomor I/O ditetapkan secara sekuensial pada modul, dimulai dari yang paling dekat dengan modul CPU seperti yang ditampilkan dalam diagram.
Pilih modul yang sudah ditempatkan untuk melihat nomor I/O awal modul tersebut.



1.1.4 Pemilihan nomor I/O manual menggunakan diagram konfigurasi modul

Contoh berikut menjelaskan cara menetapkan nomor I/O secara manual menggunakan diagram konfigurasi modul GX Works3.

Jika konfigurasi modul diubah dengan menambahkan modul baru ke diagram konfigurasi modul, nomor I/O modul yang ditambahkan akan bertumpang-tindih dengan nomor yang sudah ada jika penempatan modul yang sudah ada diubah. Edit nomor I/O agar tidak bertumpang-tindih dan tentukan konfigurasi modul.
Pesan kesalahan akan ditampilkan jika konfigurasi modul ditentukan dengan nomor I/O yang bertumpang-tindih. Pada saat ini, penetapan otomatis dapat dijalankan dari jendela pesan kesalahan yang ditampilkan.



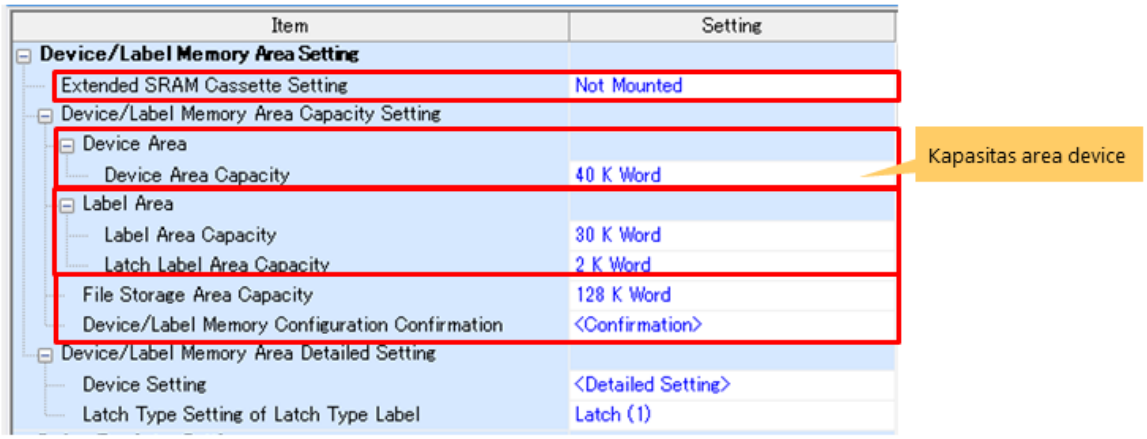
1.2 Menyetel area memori sesuai dengan status penggunaan device

1.2.1 Pengaturan area memori device/label

Jumlah poin device yang digunakan untuk modul CPU bervariasi berdasarkan tipe modul CPU. Jumlah poin awal device ditetapkan berdasarkan kapasitas area device pada modul CPU.

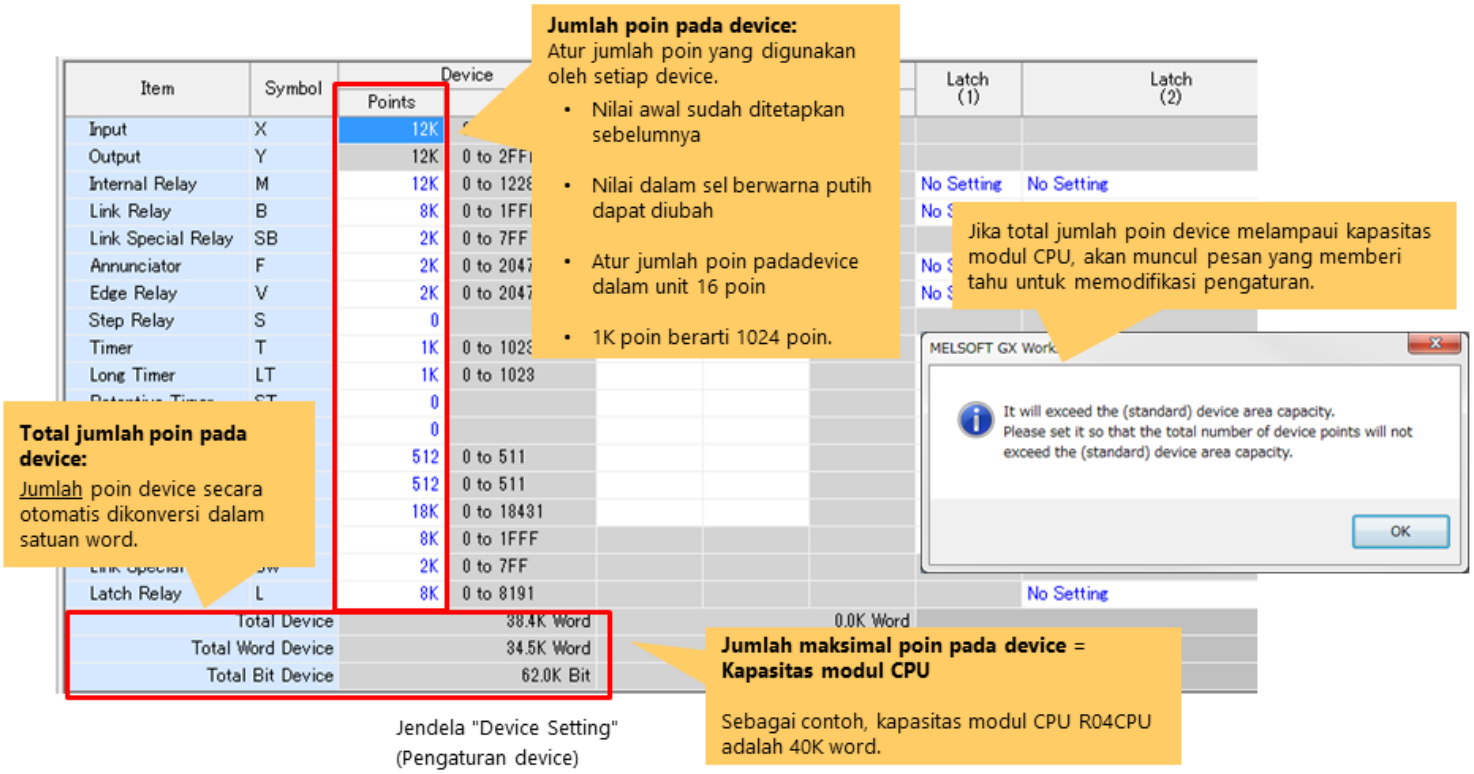
Kapasitas area device R04CPU adalah 40K word.
Mengurangi area yang tidak digunakan akan meningkatkan jumlah poin device dari nilai awal.

Gambar berikut menunjukkan jendela "Device/Label Memory Area Setting" (Pengaturan Area Memori device/Label) sebagai parameter untuk menyetel area memori.
Kapasitas area device meningkat ketika kapasitas area label atau area penyimpanan file dikurangi.
Selain itu, kapasitas seluruh area memori device/label dapat diperluas menggunakan memori SRAM.



1.2.2 Pengaturan device

Jumlah poin device yang dipilih untuk setiap device dapat diubah di jendela "Device Setting" (Pengaturan Device). Nilai awal beberapa device adalah 0 poin. Tetapkan jumlah poin saat menggunakan device tersebut.



Item	Symbol	Device Points		Latch (1)	Latch (2)
Input	X	12K			
Output	Y	12K	0 to 2FF		
Internal Relay	M	12K	0 to 122	No Setting	No Setting
Link Relay	B	8K	0 to 1FF	No S	
Link Special Relay	SB	2K	0 to 7FF		
Annunciator	F	2K	0 to 204	No S	
Edge Relay	V	2K	0 to 204	No S	
Step Relay	S	0			
Timer	T	1K	0 to 102		
Long Timer	LT	1K	0 to 1023		
		0			
		0			
		512	0 to 511		
		512	0 to 511		
		18K	0 to 18431		
		8K	0 to 1FFF		
		2K	0 to 7FF		
Latch Relay	L	8K	0 to 8191		No Setting
Total Device		38.4K Word		0.0K Word	
Total Word Device		34.5K Word			
Total Bit Device		62.0K Bit			

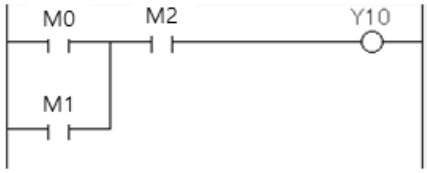
Jendela "Device Setting" (Pengaturan device)

1.3 Menggunakan nama label yang berkaitan dengan aplikasi

1.3.1 Keuntungan menggunakan label

Nama device yang digunakan dalam program kontrol harus terdiri dari satu huruf dan satu angka, seperti "M0" dan "D5". Ketika nama label berkaitan dengan aplikasi, seperti "StartSwitch", target pemrosesan menjadi jelas.
Nama label dapat diatur dengan bebas menurut aplikasi. Pengguna tidak perlu mempertimbangkan nomor device untuk area tempat label digunakan.

Program yang menggunakan nama device



Program yang menggunakan label

Label diklasifikasikan menjadi dua jenis berikut menurut lingkup penggunaan.

- **Label global**
  Label global dapat digunakan untuk semua program dalam suatu proyek.

- **Label lokal**
  Label lokal hanya dapat digunakan dalam program yang terdaftar.

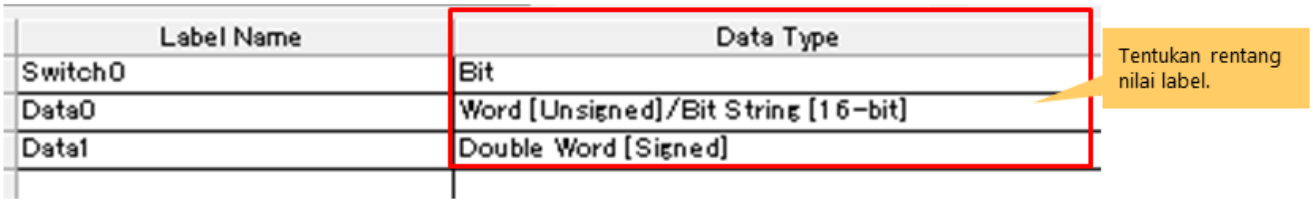
Ketika menggunakan label untuk device yang sebenarnya (X, Y), nama device harus ditetapkan menjadi label global menggunakan GX Works3.

1.3.2 Tipe data label

Tipe data harus ditentukan untuk setiap label guna menentukan rentang nilai yang akan ditangani.
Tipe data mencakup bit dan bilangan bulat, seperti yang ditunjukkan di bawah ini.

Tipe data		Rentang data
Tipe bit		Keadaan on/off device bit dan keadaan benar/salah hasil pelaksanaan
Tipe bilangan bulat	word (bilangan positif)	0 hingga 65.535
	word (bilangan negatif)	-32.768 hingga 32.767
	dobel word (bilangan positif)	0 hingga 4.294.967.295
	dobel word (bilangan negatif)	-2.147.483.648 hingga 2.147.483.647

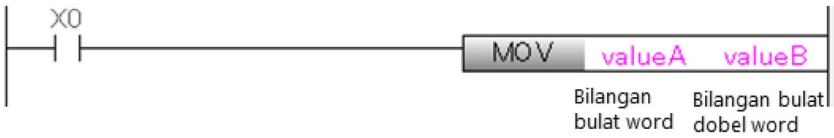
Saat menggunakan tipe bilangan bulat, pilih tipe word atau dobel word menurut rentang data, dan pilih tipe dengan atau tanpa minus menurut kebutuhan untuk menangani nilai negatif.
Tentukan tipe data label saat mengatur nama label menggunakan GX Works3.



Jendela pengaturan label

1.3.3 Nama label yang merepresentasikan tipe data

Menggunakan tipe data yang berbeda pada sumber dan destinasi transfer dapat menyebabkan kesalahan konversi atau hasil yang tidak diharapkan.
Di bawah ini adalah contoh program dari kasus semacam itu.

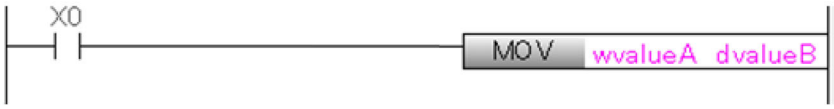


Nilai dobel word tidak dapat ditransfer ke label tipe word. Namun, tipe data tidak dapat diidentifikasi dengan nama label.

Oleh karena itu, prefiks yang merepresentasikan tipe data dapat ditambahkan ke nama label sehingga tipe data dapat diidentifikasi secara visual.
Penamaan label jenis ini dikenal sebagai notasi Hungaria.

Tipe data		Rentang data	Prefiks	Ekspansi prefiks
Tipe bit		Keadaan on/off device bit dan keadaan benar/salah hasil pelaksanaan	b	**b**it
Tipe bilangan bulat	word (bilangan positif)	0 hingga 65.535	u	**u**nsigned word (bilangan negatif)
	word (bilangan negatif)	-32.768 hingga 32.767	w	signed **w**ord (bilangan negatif)
	dobel word (bilangan positif)	0 hingga 4.294.967.295	ud	**u**nsigned **d**ouble-word (dobel word bilangan positif)
	dobel word (bilangan negatif)	-2.147.483.648 hingga 2.147.483.647	d	signed **d**ouble-word (dobel word bilangan negatif)

Contoh program di bagian atas halaman ini dapat ditulis sebagai berikut menggunakan notasi Hungaria:



Dengan menggunakan notasi Hungaria, inkonsistensi tipe data dapat diidentifikasi dalam proses penulisan program.

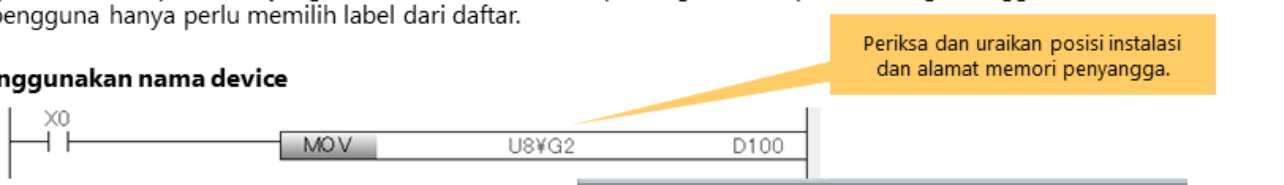
Selanjutnya dalam kursus ini, nama label dalam contoh akan ditulis dalam notasi Hungaria.

1.3.4 Menggunakan label yang disiapkan

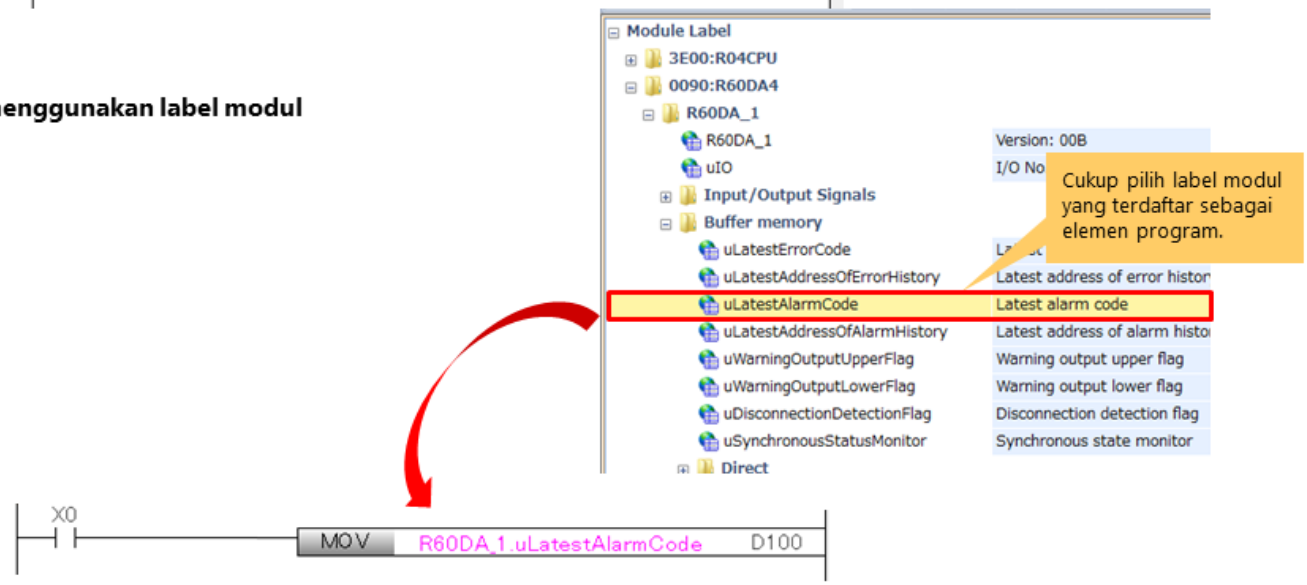
Saat konfigurasi modul diatur dalam diagram konfigurasi modul, label (label modul) yang merepresentasikan sinyal modul atau nilai pengaturan yang sesuai dengan posisi instalasi modul akan terdaftar secara otomatis.

Untuk pemrograman menggunakan nama device, nomor device dan alamat memori penyangga yang sesuai dengan sinyal harus diperiksa dalam panduan, yang memakan waktu. Waktu pemrograman dapat dikurangi menggunakan label modul karena pengguna hanya perlu memilih label dari daftar.

Jika menggunakan nama device

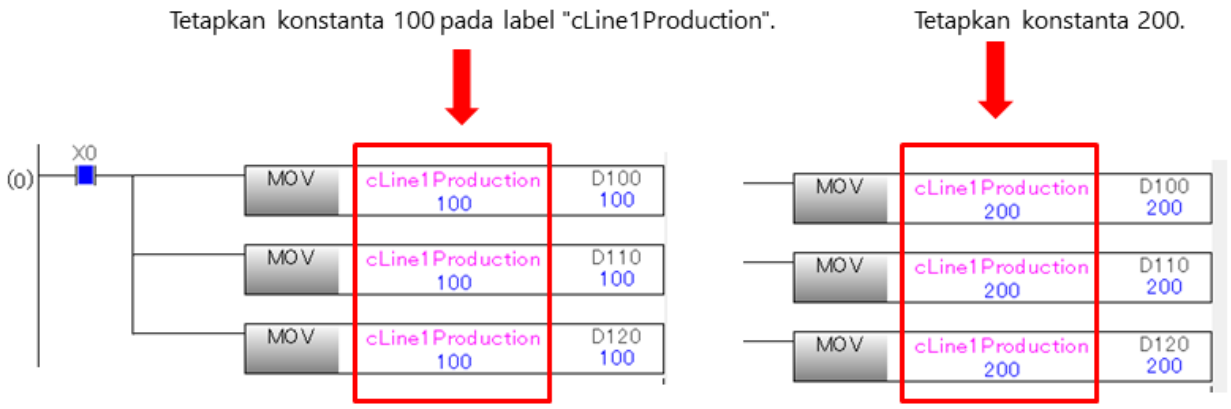


Jika menggunakan label modul

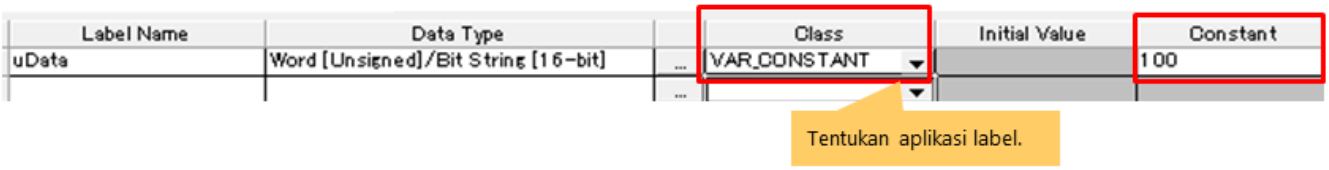


1.3.5 Menetapkan konstanta pada label

Konstanta dapat ditetapkan pada label.
Jika konstanta ditetapkan pada label, nilainya dapat diubah tanpa memodifikasi program.
Konstanta yang sama yang digunakan untuk beberapa label dapat diubah secara kolektif.



Untuk menetapkan konstanta pada label, ubah kelas yang menentukan aplikasi label pada jendela untuk mengatur label.
Untuk label lokal, pilih "VAR_CONSTANT".

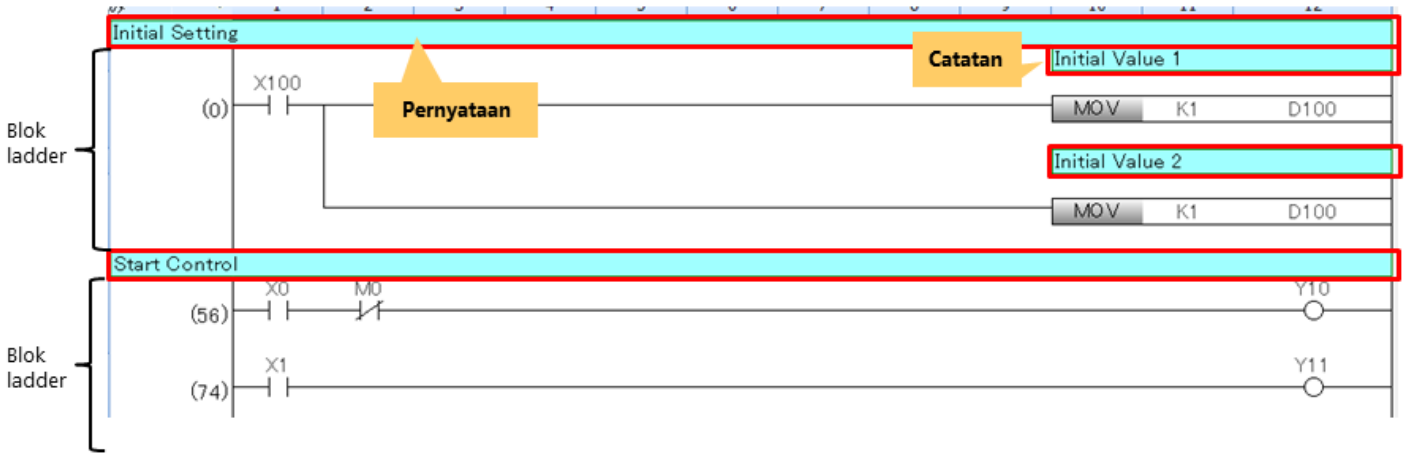


1.4 Meningkatkan keterbacaan program

Komentar, seperti detail pemrosesan dan nama device, dapat ditambahkan ke program.
Komentar membantu menjelaskan cara program berjalan.

Tipe komentar dapat dipilih menurut elemen atau rentang program.
Dalam contoh program di atas, "komentar device/label" ditambahkan untuk menjelaskan aplikasi device atau label dan tipe device I/O yang terkoneksi.

Selain itu, tipe komentar mencakup "pernyataan" yang ditambahkan pada blok ladder untuk membantu menjelaskan alur pemrosesan dan "catatan" yang membantu menjelaskan detail coil dan instruksi aplikasi.



1.5 Ringkasan bab ini

Dalam bab ini, Anda telah mempelajari:

- Pemilihan nomor I/O
- Penyetelan area memori
- Pemrograman dengan label
- Komentar dalam program

Hal-hal penting

Pemilihan nomor I/O	• Nomor I/O ditetapkan secara otomatis pada slot, dimulai dari yang paling dekat dengan modul CPU • Program dapat digunakan dalam sistem yang berbeda jika nomor I/O ditetapkan pada modul secara manual
Pengaturan device dan pengaturan area memori	• Jumlah poin device bervariasi bergantung pada modul CPU • Jumlah poin device dapat ditingkatkan dengan mengurangi area memori yang tidak digunakan • Jumlah poin device untuk setiap device dapat diubah menurut status penggunaan
Pemrograman dengan label	Target pemrosesan menjadi jelas saat label digunakan
Komentar	Detail dan alur pemrosesan menjadi lebih mudah dipahami jika komentar ditambahkan

Bab 2 Pemrograman lanjutan

Bab ini menjelaskan penggunaan lanjutan pemrograman device dan label.

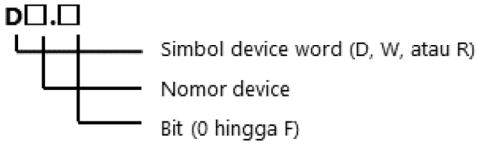
2.1 Menggunakan device word dalam satuan bit

Word device, seperti register data, biasanya digunakan dalam satuan word, tetapi dapat juga digunakan dalam satuan bit. Satuan bit digunakan untuk menentukan bit tertentu dalam register data (D).

Contoh) Register data (D)

0	0	1	0	0	1	1	0	1	1	0	0	0	1	0	1

Format spesifikasi bit



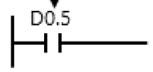
Jika b5 register data "D0" adalah 1

	F														0
D0	0	0	0	0	1	1	0	1	0	0	**1**	0	1	1	0	1

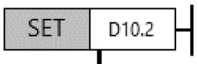
Karena D0.5 adalah 1, kontak menyala.

Jika b5 register data "D0" adalah 0

	F														0
D0	0	0	0	0	1	1	0	1	0	0	**0**	0	1	1	0	1



Karena D0.5 adalah 0, kontak mati



Tentukan b2 register data "D10".

	F														0
D10	0	0	0	1	1	1	0	1	0	1	0	0	0	**0**	1	1
	0	0	0	1	1	1	0	1	0	1	0	0	0	**1**	1	1

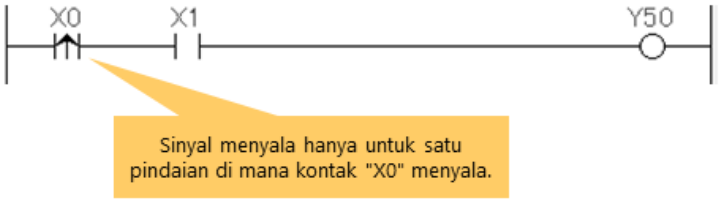
Karena D10.2 adalah 0, nilai diinversi ke 1 (on)

Untuk menggunakan label, buat deskripsi sebagai "uData.2" dan "uData.5".

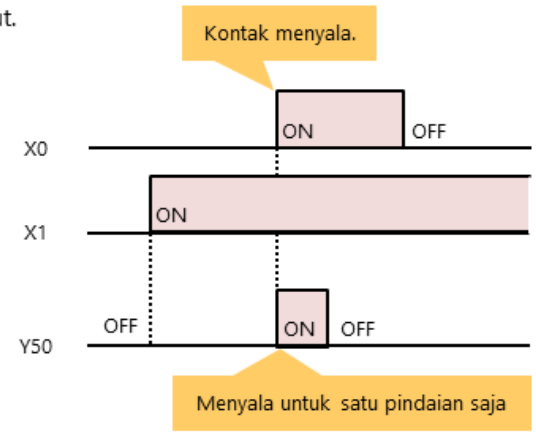
2.2 Menyalakan device hanya ketika status kontak berubah

Sinyal yang menyala hanya untuk satu pindaian pada transisi rendah ke tinggi (rising edge) atau pada transisi tinggi ke rendah (falling edge) kontak dapat ditentukan.
Fungsi ini berguna untuk mengontrol naik dan turun berdasarkan kondisi input.

Spesifikasi transisi rendah ke tinggi (rising edge) kontak

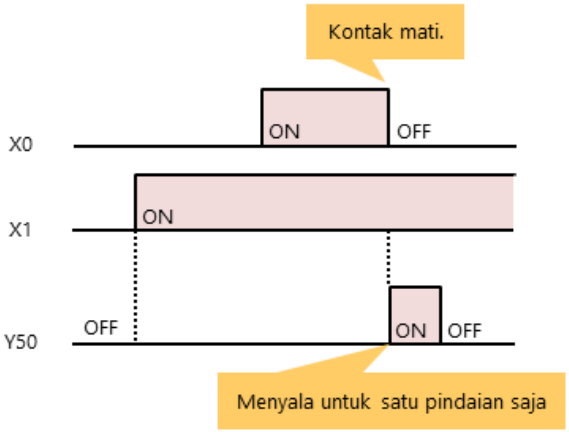


Sinyal menyala hanya untuk satu pindaian di mana kontak "X0" menyala.



Spesifikasi transisi tinggi ke rendah (falling edge) kontak

Sinyal menyala hanya untuk satu pindaian di mana kontak "X0" mati.



2.3 Mengunci waktu pengukuran timer

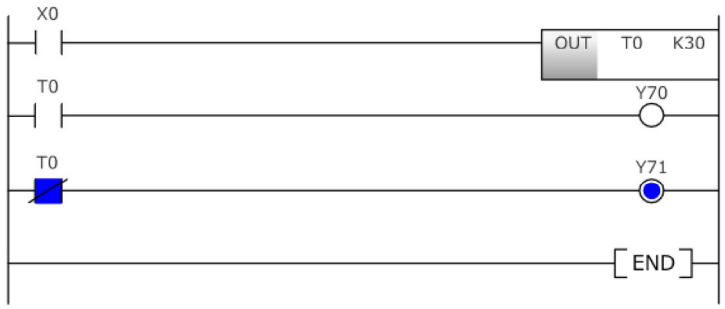
Bagian ini menjelaskan salah satu device timer, yaitu timerretentif, yang dapat menahan waktu pengukuran.

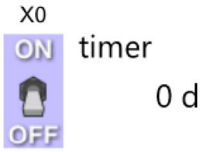
2.3.1 Perbedaan antara timer dan timer retentif

Sebelum menjelaskan timer retentif, mari kita lihat cara timer beroperasi.

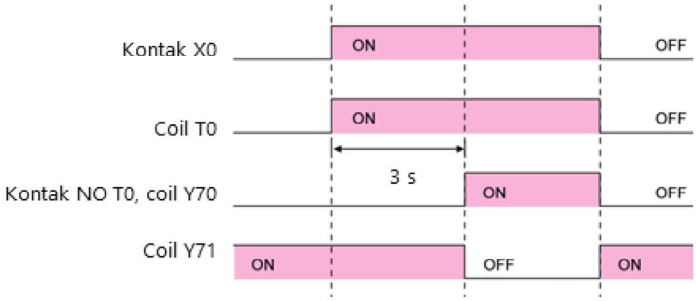
Timer memulai pengukuran ketika coil menyala. Jika waktu yang ditentukan telah berlalu, waktu habis dan kontak menyala. Ketika coil mati, waktu pengukuran direset ke "0". Simbol device timer adalah "T".

Gunakan sakelar input di sebelah kanan untuk melihat cara timer beroperasi.





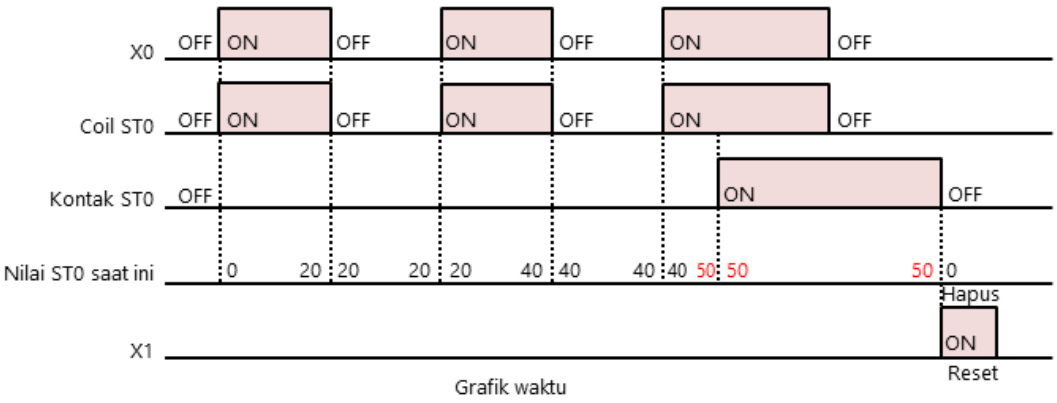
Ketika tiga detik telah berlalu setelah X0 menyala, Y70 menyala dan Y71 mati.



Grafik waktu

2.3.1 Perbedaan antara timer dan timer retentif

Timer retentif bermanfaat untuk mengukur waktu operasi total. Timer retentif memulai pengukuran saat coil menyala. Jika waktu yang ditentukan telah berlalu, waktu habis dan kontak menyala. Ketika coil mati, waktu pengukuran tidak direset. Ketika coil menyala kembali, pengukuran dimulai kembali dari nilai yang ditahan. Simbol device timer retentif adalah "ST".



Grafik waktu

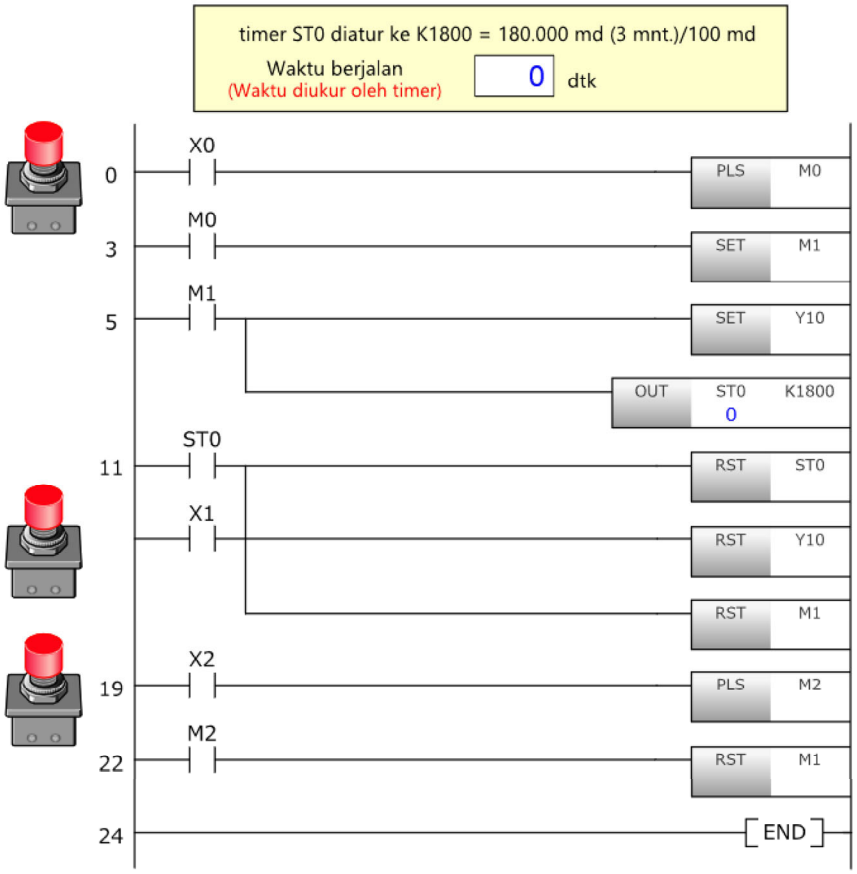
2.3.2 Contoh program timer retentif

Mari kita lihat cara timer retentif beroperasi dengan menyimulasikan mesin yang berjalan menggunakan sakelar input (X0 hingga X2).
*Timer retentif (ST0) diatur dengan ipenambahan sebesar 100 md.

1. Ketika X0 menyala, pengoperasian dimulai.
2. Ketika X2 menyala, pengoperasian dijeda dan nilai saat ini ditahan.
3. Ketika X0 menyala lagi, pengoperasian dimulai lagi.
4. Ketika X1 menyala, pengoperasian berakhir dan nilai saat ini direset.

X0 hingga X2: Sakelar input

Y10: Sinyal mulai



2.3.3 Pengaturan untuk timer retentif

Secara default, jumlah poin yang digunakan oleh timer retentif adalah "0".
Sebelum menggunakan timer retentif, atur jumlah poin dalam "Device Setting" (Pengaturan Device) parameter CPU menggunakan GX Works3.

Dalam contoh berikut, 64 poin (ST0 hingga ST63) diatur untuk timer retentif.

Item	Symbol	Device		Local Device			Latch (1)	Latch (2)
		Points	Range	Start	End	Points		
Input	X	12K	0 to 2FFF					
Output	Y	12K	0 to 2FFF					
Internal Relay	M	12K	0 to 12287				No Setting	No Setting
Link Relay	B	16K	0 to 3FFF				No Setting	No Setting
Link Special Relay	SB	16K	0 to 3FFF					
Annunciator	F	2K	0 to 2047				No Setting	No Setting
Edge Relay	V	2K	0 to 2047				No Setting	No Setting
Step Relay	S	0						
Timer	T	1K	0 to 1023				No Setting	No Setting
Long Timer	LT	1K	0 to 1023				No Setting	No Setting
Retentive Timer	ST	64	0 to 63				No Setting	No Setting
Long Retentive Time	LST	0					No Setting	No Setting
Counter	C	512	0 to 511				No Setting	No Setting
Long Counter	LC	512	0 to 511				No Setting	No Setting
Data Register	D	18K	0 to 18431				No Setting	No Setting
Link Register	W	8K	0 to 1FFF				No Setting	No Setting
Link Special Registe	*SW*	2K	0 to 7FF					
Latch Relay	L	8K	0 to 8191					No Setting
Total Device			39.9K Word			0.0K Word		
Total Word Device			34.6K Word			0.0K Word		
Total Bit Device			84.2K Bit			0.0K Bit		

2.3.4 Menggunakan label untuk menentukan timer

Atur "Timer" (Timer) ke tipe data ketika label menggunakan device timer.

Label Name	Data Type
uTimer1	Timer
uTimer2	Retentive Timer

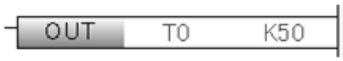
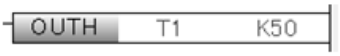
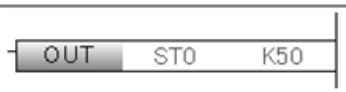
Pengaturan device diperlukan untuk menggunakan timer retentif sebagaimana dijelaskan pada bagian sebelumnya. Namun, hal itu tidak diperlukan untuk label.

2.4 Mengubah satuan pengukuran timer

Satuan pengukuran dan waktu bervariasi bergantung pada tipe timer.

- Timer kecepatan tinggi (satuan pengukuran pendek)
- Timer kecepatan rendah (satuan pengukuran panjang)
- Timer panjang mampu melakukan pengukuran waktu panjang

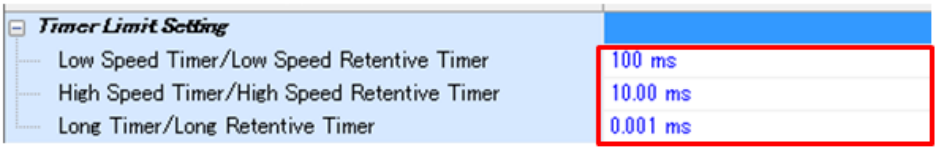
Timer di atas masing-masing memiliki fungsi timer retentif.

Tipe	Satuan pengukuran	Contoh program	Operasi	Retensi nilai saat ini
Timer kecepatan rendah	100 md (default)		Timer kecepatan rendah T0 mengukur 5 detik.	16 bit
Timer kecepatan tinggi	10,00 md (default)		Timer kecepatan tinggi T1 mengukur 0,5 detik.	
Timer retentif kecepatan rendah	100 md (default)		Timer retentif kecepatan rendah ST0 mengukur 5 detik.	
Timer retentif kecepatan tinggi	10,00 md (default)		Timer kecepatan tinggi ST1 mengukur 0,5 detik.	
Timer panjang	0,001 md (default)		Timer panjang LT0 mengukur 0,005 detik.	32 bit
Timer retentif panjang			Timer retentif panjang LST0 mengukur 0,005 detik.	

Satuan pengukuran awal adalah 100 md untuk timer kecepatan rendah, 10 md untuk timer kecepatan tinggi, dan 0,001 md untuk timer panjang.
Lihat halaman berikutnya untuk mengetahui cara mengubah satuan pengukuran.

2.4 Mengubah satuan pengukuran timer

Satuan pengukuran timer dapat diubah dalam "Timer Limit Setting" (Pengaturan Batas Timer) pada parameter CPU.



2.5 Menangani beberapa device (register indeks)

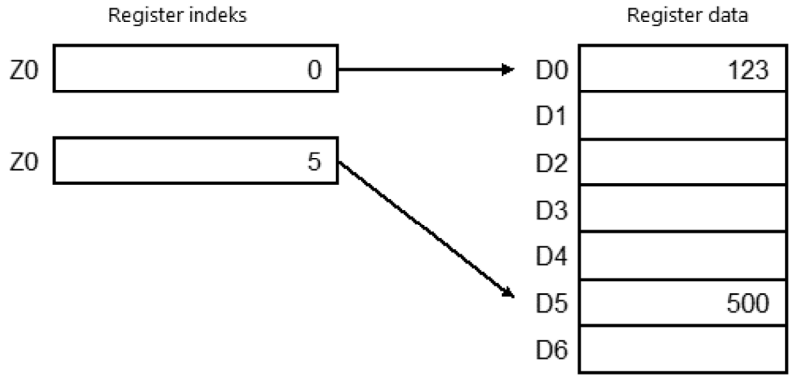
Register indeks (Z) digunakan dalam kombinasi bersama dengan device lain untuk secara tidak langsung menentukan (memodifikasi) nomor device dari device target kontrol. Register indeks berguna untuk menyederhanakan program karena register indeks dapat mendeskripsikan beberapa device dalam suatu batch.

- Register indeks dideskripsikan setelah simbol device dan nomor device.
- Nomor device target kontrol aktual = Simbol device (nomor device + register indeks)
- Jumlah poin device untuk register indeks adalah 20 poin (Z0 hingga Z19) secara default

2.5.1 Contoh aplikasi register indeks

Jika device dideskripsikan sebagai "D0Z0", hal ini berarti D(0+Z0).

Contoh) Jika Z0 adalah 0, nomor device adalah D0.
Jika Z0 adalah 5, nomor device adalah D5.

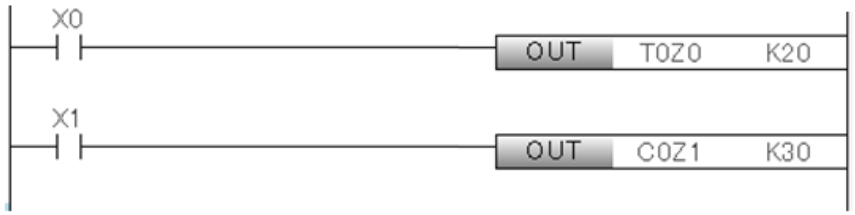


2.5.2 Device yang dapat dimodifikasi oleh register indeks

Device yang dapat dimodifikasi oleh register indeks mencakup berikut ini:

Device bit	X, Y, M, L, S, B, F
Device word	T, C, D, R, W
Konstanta	K, H
Penunjuk	P

Catatan) Untuk kontak dan coil yang digunakan dalam timer dan counter, hanya register indeks "Z0" atau "Z1" yang dapat digunakan.



Jika Z0 adalah 1, T1 mengukur waktu.

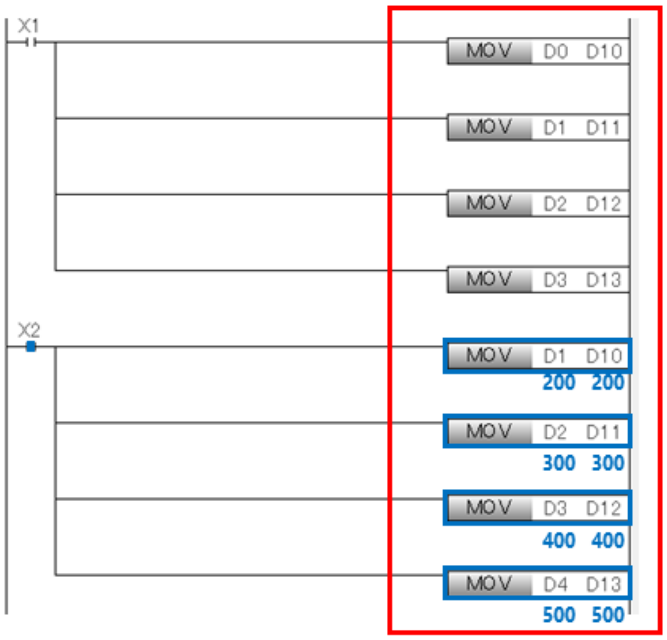
Jika Z1 adalah 5, C5 melakukan penghitungan.

2.5.3 Penyederhanaan program menggunakan register indeks

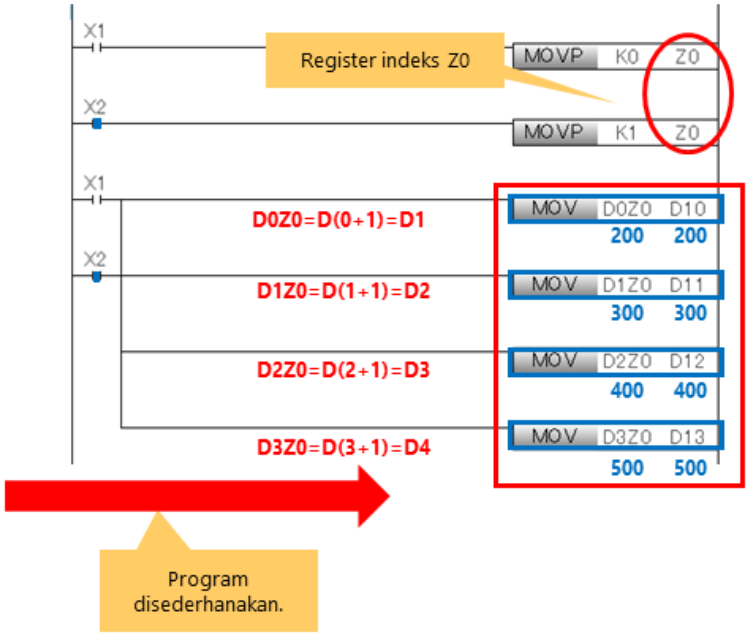
Program-program yang ditunjukkan di bawah ini mentransfer nilai dalam "D0 to D4" ke "D10 to D13" ketika X1 atau X2 menyala. Program (1) dan (2) akan memiliki hasil yang sama. Dalam program (1), data ditransfer secara langsung. Dalam program (2), data ditransfer secara tidak langsung melalui register indeks.

Nilai tersimpan awal
D0=100
D1=200
D2=300
D3=400
D4=500

(1) Ketika register indeks tidak digunakan

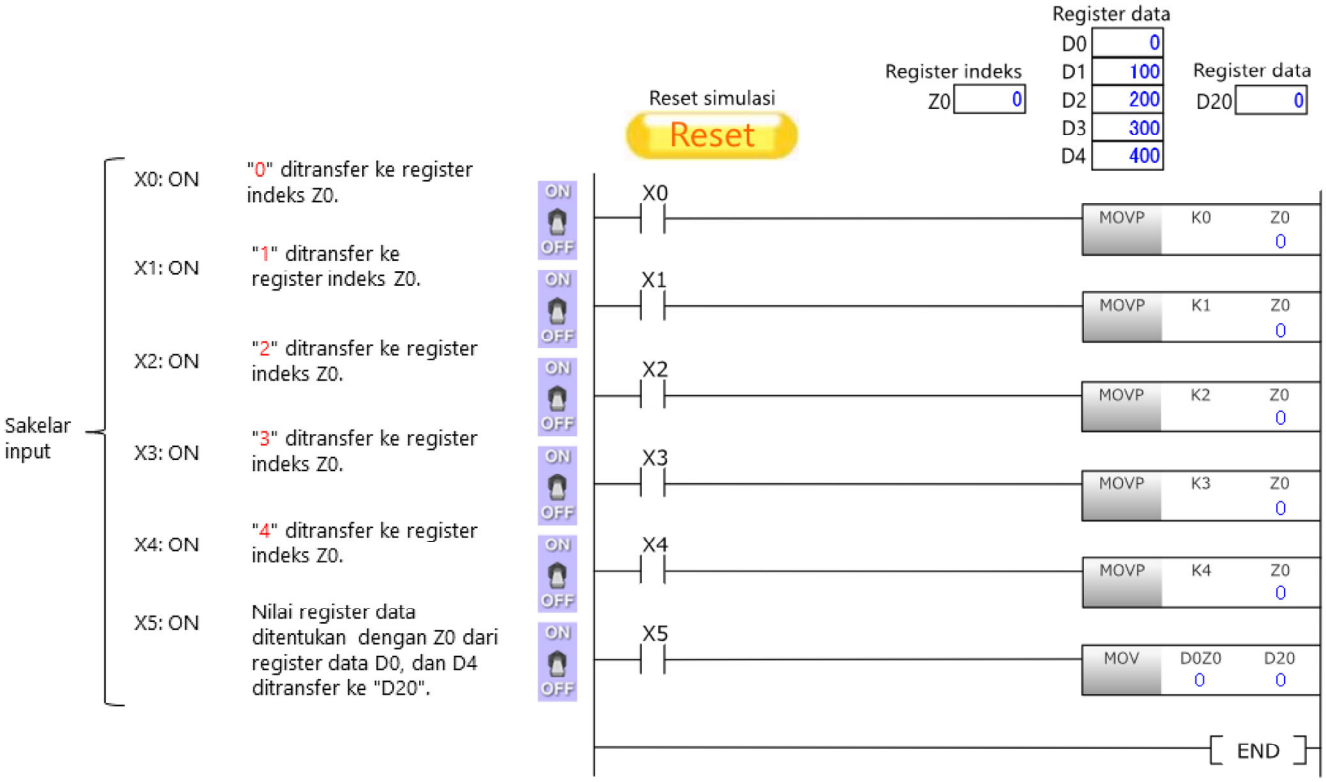


(2) Ketika register indeks digunakan



2.5.4 Contoh program register indeks

Cara register indeks Z0 beroperasi dapat disimulasikan dengan menyalakan sakelar input X0 hingga X5.
*K0 hingga K400 sudah tersimpan dalam register data D0 hingga D4.
Nyalakan sakelar input X0 hingga X5 untuk memeriksa nilai yang disimpan ke setiap area device.



2.6 Menangani beberapa nilai (array)

Dengan menggunakan array, beberapa nilai dapat ditangani dengan satu nama label.
Dalam contoh berikut, data volume produksi dalam suatu pabrik manufaktur mobil disimpan berdasarkan destinasi.

Destinasi	Country A	Country B	Country C
Volume produksi	35 unit	75 unit	65 unit

Data volume produksi berdasarkan destinasi ditetapkan ke suatu label.
Ketika tidak menggunakan array, nama label harus dibuat untuk setiap destinasi.
Dengan menggunakan array, volume produksi untuk beberapa destinasi dapat ditetapkan ke dan disimpan dalam satu nama label.

Ketika tidak menggunakan array

uProductionA
uProductionB
uProductionC

Ketika menggunakan array

uProduction

Label individual dalam array ditentukan menggunakan nomor elemen. Nomor elemen dimulai dari [0].

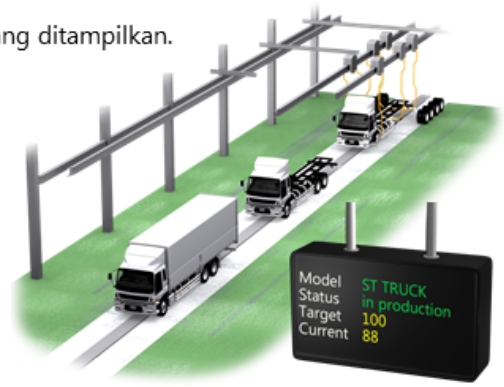
Dalam contoh program berikut, volume produksi yang direncanakan untuk Country A ditransfer ke label lain.

2.7 Menangani beberapa nilai (struktur)

Dengan menggunakan struktur, beberapa label yang berkaitan dapat ditangani dengan satu nama label. Dalam contoh berikut, status lini produksi mobil ditampilkan pada Andon (papan tampilan).

Tabel berikut mendaftar nama label, nilai, dan tipe data yang sesuai dengan item yang ditampilkan.

Item	Nama label	Nilai	Tipe data label
Model	sModel	'ST TRUCK'	Tipe string
Status operasi	bStatus	'in production'	Tipe bit
Target volume produksi hari ini	uPlanQty	'100' unit	Tipe bilangan bulat (word, bilangan positif)
Volume produksi saat ini	uActualQty	'88' unit	Tipe bilangan bulat (word, bilangan negatif)



Jika struktur tidak digunakan untuk pabrik yang memiliki beberapa lini produksi, nama label harus diubah untuk setiap jalur. Berikut ini adalah contoh nama label dengan penambahan nama lini produksi.

Lini produksi pertama

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

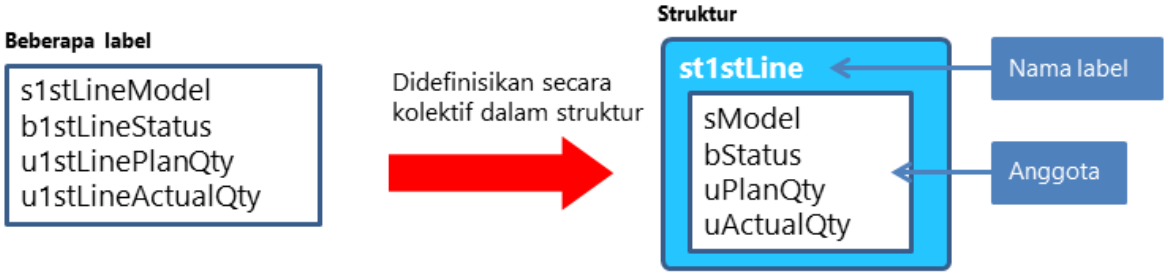
Lini produksi kedua

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Seiring pertambahan jumlah lini produksi, jumlah label yang harus ditangani akan bertambah. Sebagai akibatnya, program menjadi lebih panjang dan lebih sulit untuk dibaca.

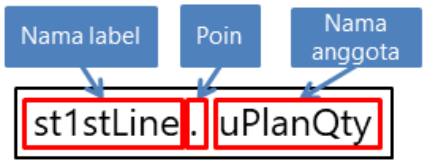
2.7 Menangani beberapa nilai (struktur)

Struktur memungkinkan satu nama label untuk merepresentasikan beberapa label yang berkaitan dengan satu lini produksi. Oleh karena itu, struktur digunakan untuk secara kolektif menata, menyimpan, dan menangani kondisi dan spesifikasi yang berkaitan dengan objek dan hal fisik seperti device, peralatan, dan potongan pekerjaan.

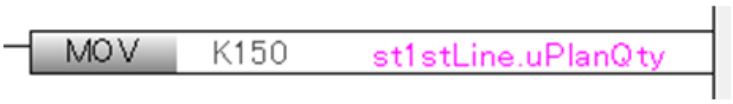


Label struktur diberi prefiks "st" yang merepresentasikan **st**ructure (struktur).
Label individual yang didefinisikan oleh struktur disebut anggota. Tipe data setiap anggota dapat berbeda.

Untuk menentukan setiap anggota dalam struktur, tambahkan nama anggota setelah label struktur dengan poin (.) sebagai pembatas di antaranya.



Dalam contoh program berikut, konstanta ditetapkan pada anggota label tipe struktur untuk lini produksi pertama.



2.8 Menahan status device (pengunci)

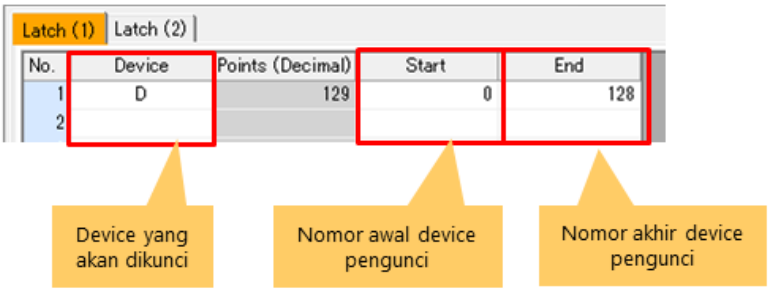
Bagian ini menjelaskan fungsi pengunci, yang menahan nilai device saat modul CPU berhenti beroperasi.
Sebagai contoh, bahkan saat terjadi kegagalan daya yang melampaui waktu kegagalan daya sementara yang diizinkan, pengontrol terprogram dapat memulai ulang kontrol urutan menggunakan data yang ditahan pada pemberhentian operasi.

Jika fungsi pengunci tidak digunakan, nilai device dibersihkan dan direset ke nilai awal (off untuk device bit dan "0" untuk device word) dalam kejadian berikut:

- Daya mati
- Reset dengan sakelar RUN/STOP/RESET (JALANKAN/HENTIKAN/RESET)
- Kegagalan daya melampaui waktu kegagalan daya sementara yang diizinkan

2.8.1 Mengatur pengunci pada device

Atur jangkauan pengunci dalam jendela "Device Setting" (Pengaturan Device) pada parameter CPU.
Berikut ini adalah contoh pengaturan pengunci register data, D0 hingga D128.



Device yang akan dikunci

Nomor awal device pengunci

Nomor akhir device pengunci

2.8.2 Tipe pengunci dan metode penghapusan

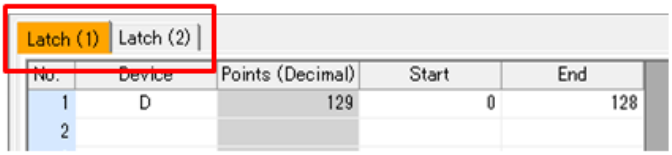
Terdapat dua tipe pengunci (pengunci (1) dan pengunci (2)) menurut metode penghapusan.

- Latch (1) (Pengunci (1))

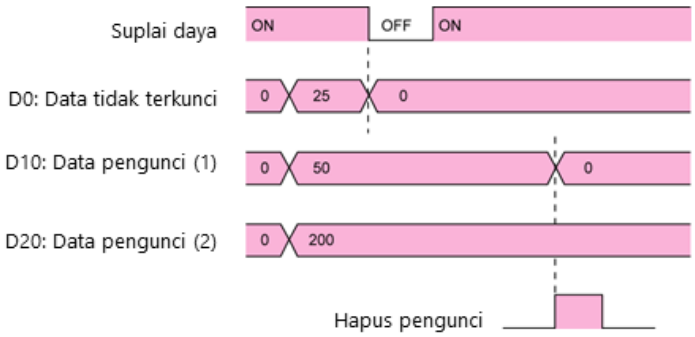
  Data yang dikunci dapat dihapus menggunakan fungsi* operasi memori CPU GX Works3.
  Gunakan pengunci 1 jika data yang dikunci perlu dihapus di situs instalasi.

- Latch (2) (Pengunci (2))

  Data yang dikunci dapat dihapus menggunakan instruksi dalam program.
  Gunakan pengunci 2 jika data yang dikunci tidak dihapus di situs instalasi.



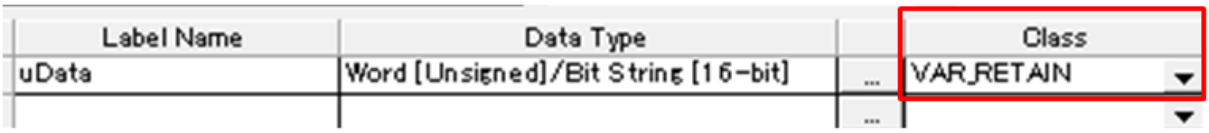
Berikut ini adalah grafik waktu penghapusan pengunci.



*Jalankan fungsi dengan memilih [Online] → [CPU Memory Operation] (Operasi Memori CPU) dari menu GX Works3.

2.8.3 Mengatur pengunci pada label

Untuk mengatur pengunci pada label, pilih nama kelas yang mengandung "RETAIN" pada jendela untuk mengatur label. Untuk label lokal, pilih "VAR_RETAIN".



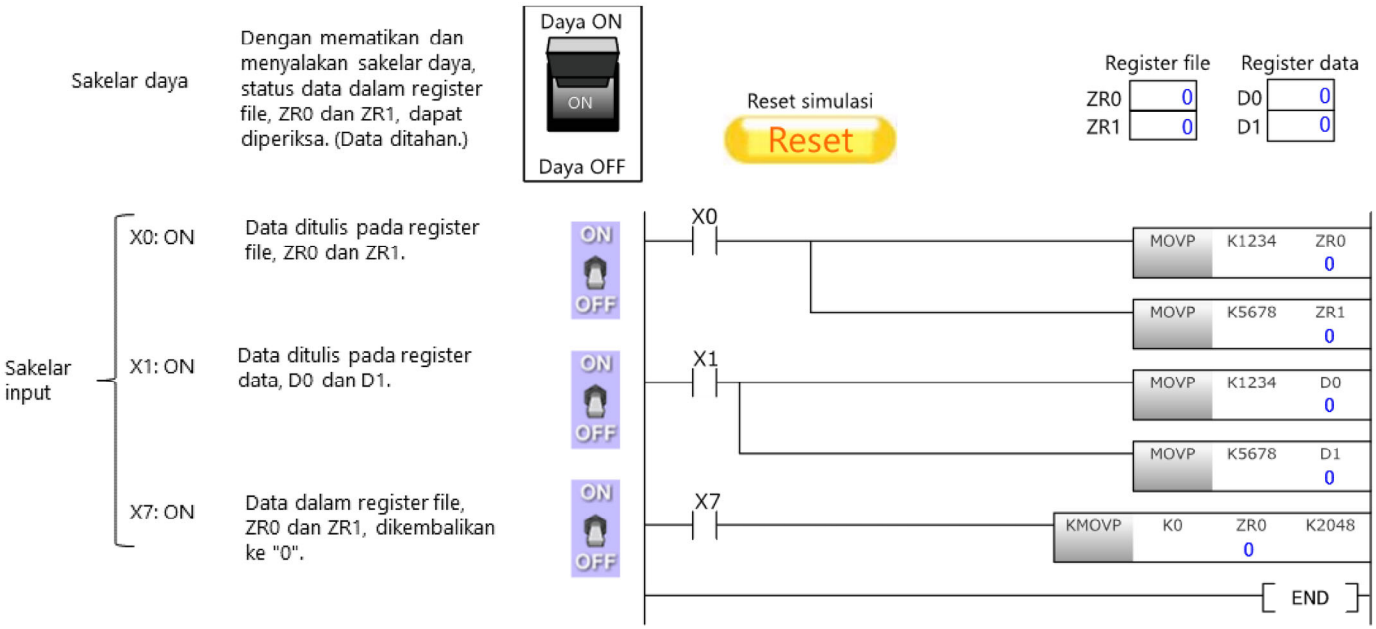
Label Name	Data Type		Class
uData	Word [Unsigned]/Bit String [16-bit]	...	VAR_RETAIN
		...	

2.9 Menahan status device (register file)

- Register file adalah device word yang digunakan untuk memperluas register data (D)
- Dibandingkan dengan register data, register file dapat menangani jumlah data yang lebih besar
- Register file disimpan dalam memori data modul CPU atau dalam kartu memori
- Data yang disimpan dalam register file akan ditahan bahkan ketika sistem dimatikan dayanya atau modul CPU direset. Register file bermanfaat untuk menyimpan nilai yang telah ditentukan sebelumnya, seperti nilai standar
- Untuk menghapus data, tulis 0 untuk rentang register file atau hapus memori dari GX Works3
- Simbol device adalah "ZR"

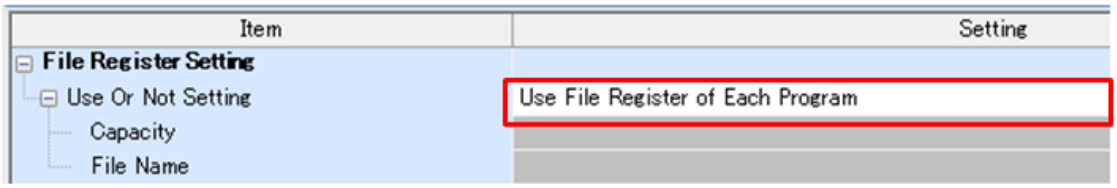
2.9.1 Operasi program ladder

Cara register file beroperasi dapat disimulasikan dengan menyalakan sakelar input dan sakelar daya.

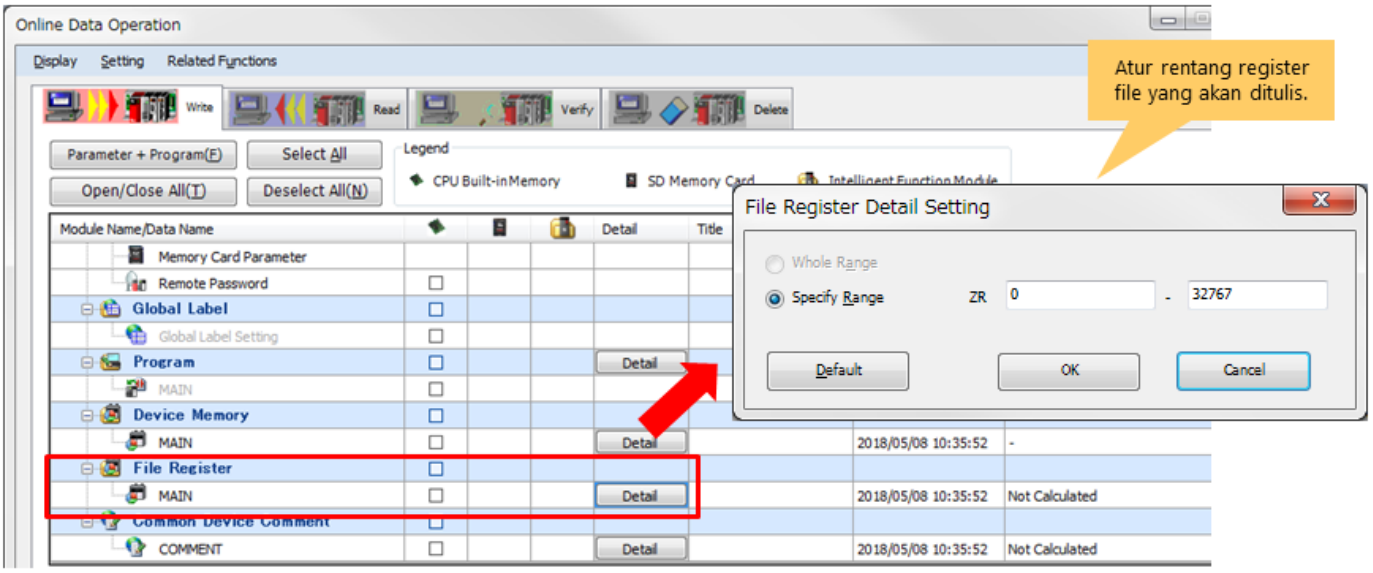


2.9.2 Pengaturan untuk register file

Bagian ini menjelaskan pengaturan yang menentukan register file lokal sebagai destinasi penyimpanan setiap program. Pilih "File Register Setting" (Pengaturan Register File) pada parameter CPU, dan pilih "Use File Register of Each Program" (Gunakan Register File Setiap Program).



Untuk menulis data untuk pengontrol terprogram, pengaturan register file harus ditulis untuk setiap program.



2.10 Menggunakan device dengan fungsi dan operasi yang telah ditentukan

Relai khusus dan register khusus yang digunakan dalam modul CPU memiliki fungsi dan operasi yang telah ditentukan sebelumnya. Relai khusus (SM) adalah relai internal yang digunakan untuk informasi bit (on/off), dan register khusus (SD) adalah register internal yang digunakan untuk informasi word.

2.10.1 Tipe relai khusus dan register khusus

Relai khusus dan register khusus dikategorikan berdasarkan tipe informasinya. Tipe-tipe utama disebutkan di bawah ini. Dalam program pengguna, relai khusus dan register khusus digunakan sebagai kondisi penentuan untuk kontrol. Relai khusus dan register khusus juga digunakan untuk pemantauan operasi, yang dapat dilakukan pada monitor device GX Works3.

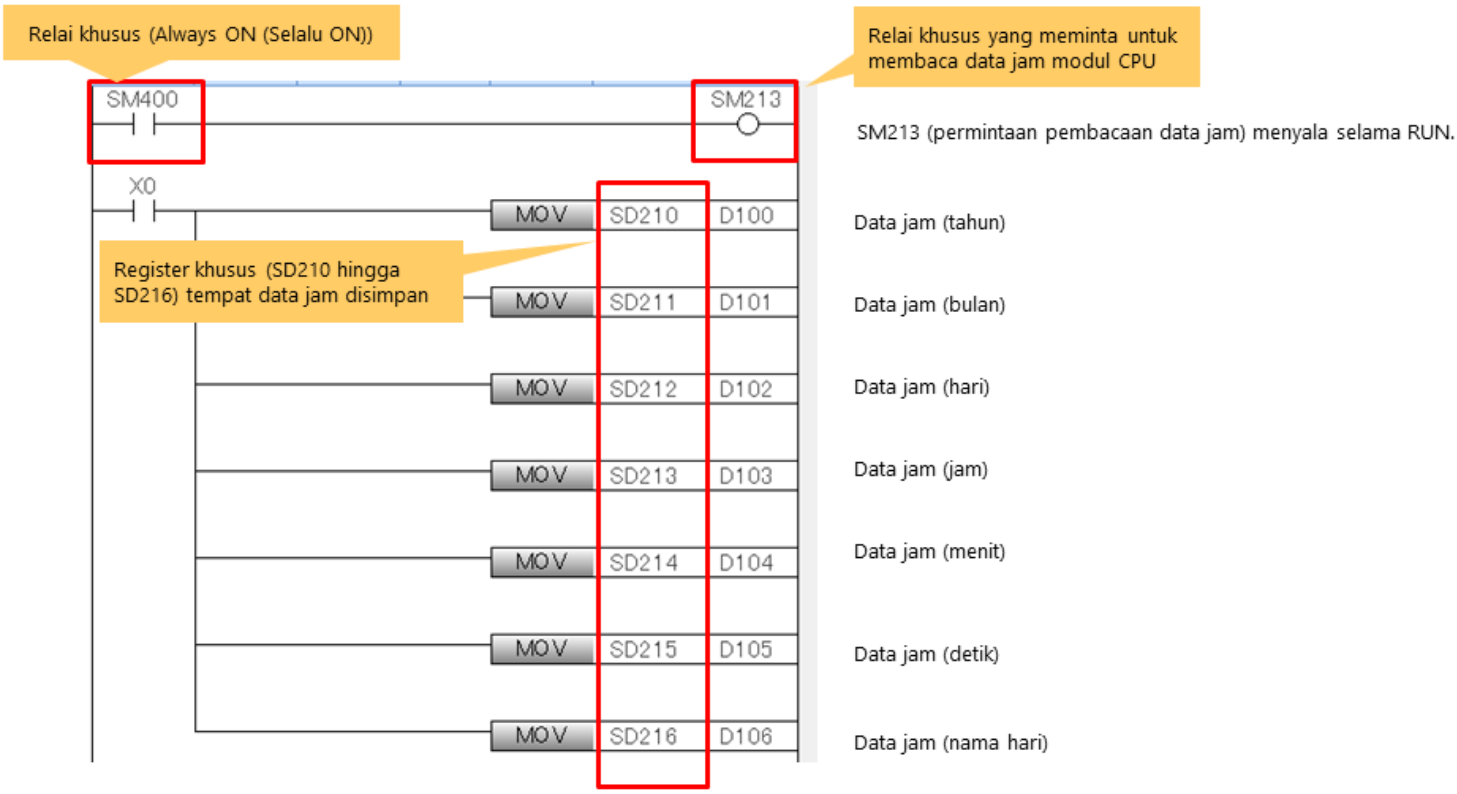
Informasi diagnostik
Hasil diagnostik modul CPU
Kesalahan diagnostik dan kode kesalahan

Informasi sistem
Informasi sistem modul CPU
Status pengoperasian modul CPU, data jam, dan informasi lain

Jam sistem
Sinyal jam dan nilai hitungan yang digunakan sebagai elemen pewaktuan dasar
Berbagai sinyal jam (selalu ON, ON/OFF pada interval tertentu, dan sinyal lain)

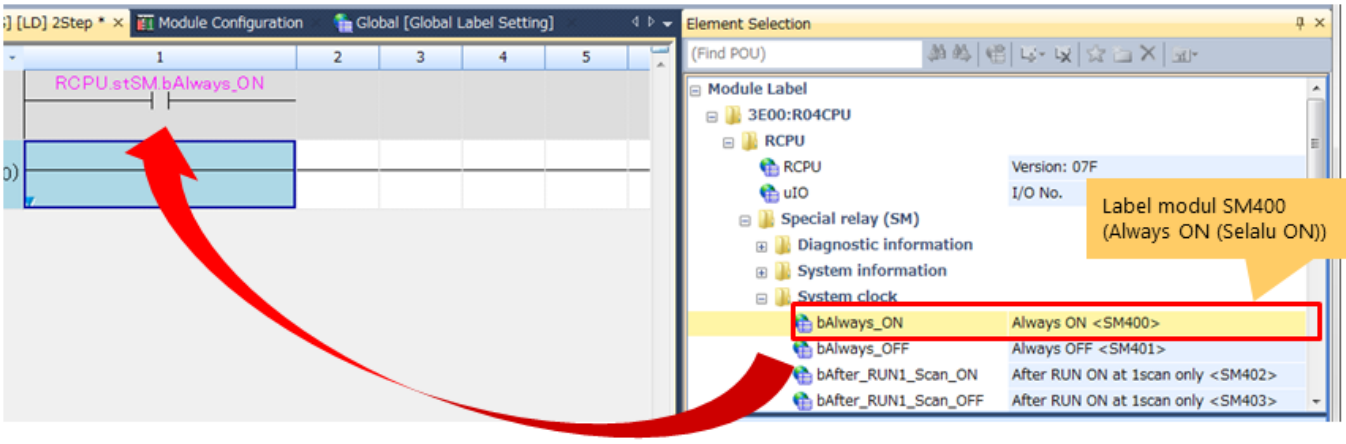
2.10.2 Contoh program relai khusus dan register khusus

Berikut ini adalah contoh program untuk membaca data jam modul CPU menggunakan relai khusus dan register khusus.



2.10.3 Menggunakan program untuk relai khusus dan register khusus

Relai khusus dan register khusus disiapkan sebagai label modul dalam modul CPU. Relai khusus dan register khusus dapat digunakan cukup dengan memilih dan menempatkan nama label yang relevan, tanpa memeriksa nomor device dalam panduan.



2.11 Menghitung dengan bilangan asli

2.11.1 Aplikasi bilangan asli

- "Bilangan asli" adalah nilai numerik dengan poin desimal.
- Bilangan bulat biasanya digunakan dalam program kontrol. Namun, bilangan asli digunakan dalam program untuk kontrol operasi lanjutan seperti fungsi trigonometri dan operasi eksponensial karena nilai numerik dengan poin desimal perlu ditangani dalam program semacam itu
- Data numerik bilangan asli yang ditangani dalam modul CPU disebut sebagai "data floating poin"

Catatan:

- Satu bilangan asli selalu menggunakan dua device word yang berturut-turut (ruang memori 32 bit), terlepas dari angkanya *

- Dalam program kontrol, petunjuk pengoperasian khusus (seperti penjumlahan, pengurangan, perkalian, pembagian, dan fungsi khusus) yang menangani bilangan asli disiapkan. Instruksi konversi antara bilangan bulat dan bilangan asli juga disiapkan

*Jika diperlukan kalkulasi yang lebih akurat dengan jumlah digit signifikan yang lebih besar, gunakan empat device word.

- Bilangan asli yang menggunakan dua device word disebut bilangan asli presisi tunggal
- Bilangan asli yang menggunakan empat device word disebut bilangan asli presisi ganda

Kursus ini fokus pada bilangan asli presisi tunggal.

2.11.2 Notasi untuk bilangan asli

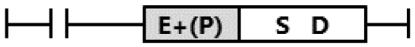
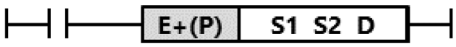
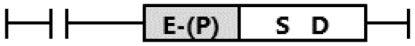
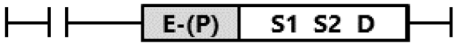
"E" digunakan untuk merepresentasikan bilangan asli.

Mengekspresikan konstanta dengan bilangan asli

Untuk menulis konstanta, mulailah dengan "E".

Ekspresi normal	Tulis nilai numerik apa adanya. (Contoh) Tulis 10,2345 sebagai "E10.2345".
Ekspresi eksponensial	Tulis nilai numerik sebagai "(nilai numerik) × 10^{n}". (Contoh) Tulis 1234,0 sebagai "E1.234+3".

2.11.3 Petunjuk pengoperasian (penjumlahan dan pengurangan)

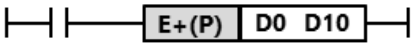
Simbol instruksi	Contoh ladder	
E+ (Menjumlahkan bilangan asli presisi tunggal)	 Operasi bilangan asli "D + S = D" dijalankan.	 Operasi bilangan asli "S1 + S2 = D" dijalankan.
E- (Mengurangkan bilangan asli presisi tunggal)	 Operasi bilangan asli "D - S = D" dijalankan.	 Operasi bilangan asli "S1 - S2 = D" dijalankan.

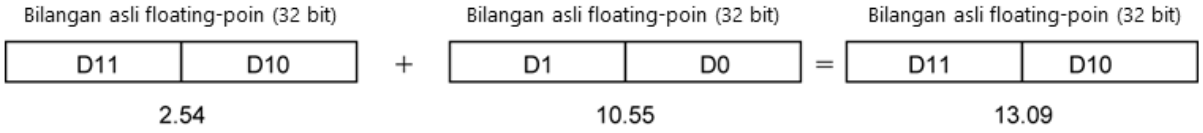
S (sumber): Data sebelum operasi (konstanta, nomor device)
D (destinasi): Destinasi data setelah operasi (nomor device)
P: Instruksi yang akan dijalankan pada transisi rendah ke tinggi (rising edge) (dari off ke on)
S1 dan S2: Dua item data yang akan dioperasikan.

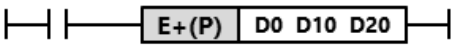
Catatan:
Bilangan bulat dan bilangan asli tidak dapat dicampur dalam satu operasi.
Untuk operasi bilangan asli presisi tunggal, S, S1, dan S2 dalam ekspresi operasional harus merupakan bilangan asli presisi tunggal.
Bilangan asli presisi tunggal disimpan dalam D.

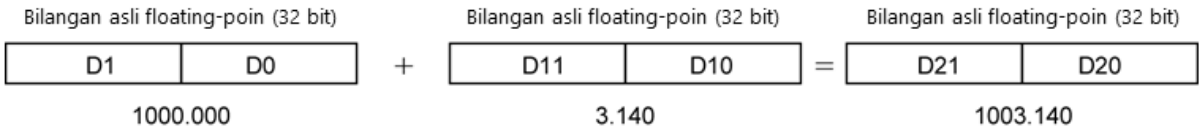
2.11.3 Petunjuk pengoperasian (penjumlahan dan pengurangan)

Contoh aplikasi instruksi penjumlahan



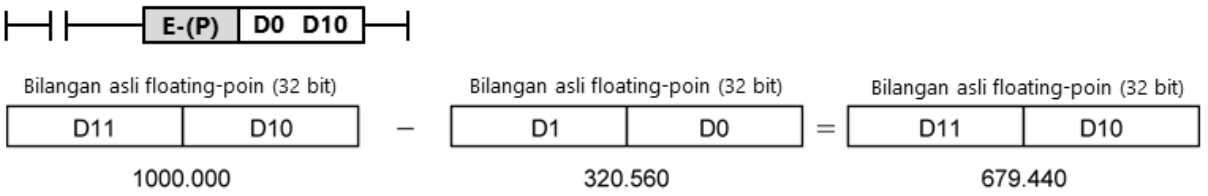


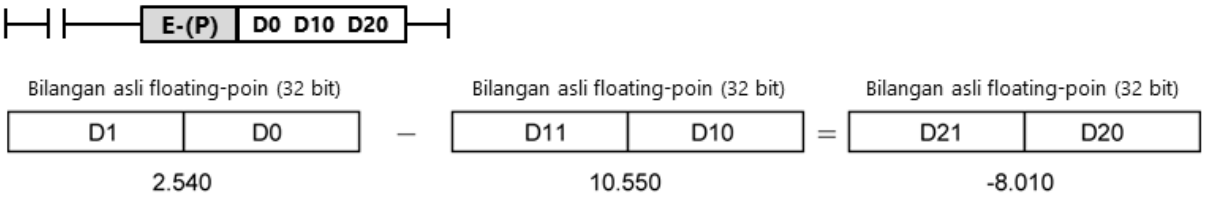




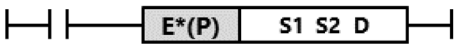
2.11.3 Petunjuk pengoperasian (penjumlahan dan pengurangan)

Contoh aplikasi instruksi pengurangan





2.11.4 Petunjuk pengoperasian (perkalian dan pembagian)

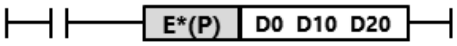
Simbol instruksi	Contoh ladder
E* (Mengalikan bilangan asli presisi tunggal)	 Operasi bilangan asli "S1 * S2 = D" dijalankan.
E/ (Membagi bilangan asli presisi tunggal)	 Operasi bilangan asli "S1 / S2 = D" dijalankan.

S1, S2 (sumber) Dua item data yang akan dioperasikan.
D (destinasi): Destinasi data setelah operasi (nomor device)
P: Instruksi yang akan dijalankan pada transisi rendah ke tinggi (rising edge) (dari off ke on)

Catatan:
Untuk operasi bilangan asli presisi tunggal, S1, dan S2 dalam ekspresi operasi harus merupakan bilangan asli presisi tunggal. Bilangan asli presisi tunggal disimpan dalam D.

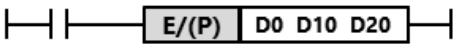
2.11.4 Petunjuk pengoperasian (perkalian dan pembagian)

Contoh aplikasi instruksi perkalian



Bilangan asli floating-poin (32 bit)			Bilangan asli floating-poin (32 bit)			Bilangan asli floating-poin (32 bit)	
D1	D0	×	D11	D10	=	D21	D20
1000.000			25.590			25590.000	

Contoh aplikasi instruksi pembagian



Bilangan asli floating-poin (32 bit)			Bilangan asli floating-poin (32 bit)			Bilangan asli floating-poin (32 bit)	
D1	D0	÷	D11	D10	=	D21	D20
1000.000			25.590			39.078	

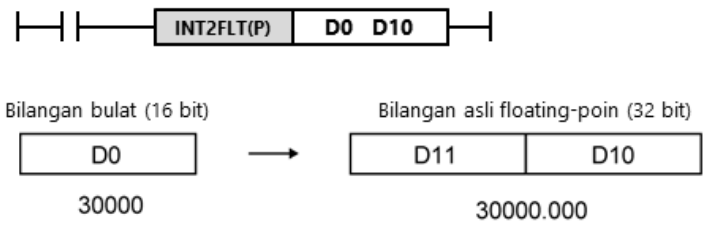
2.11.5 Instruksi konversi antara bilangan bulat dan bilangan asli

Simbol instruksi	Contoh ladder	
INT2FLT (Mengonversi bilangan bulat ke bilangan asli presisi tunggal)	Bilangan bulat (16 bit) dikonversi ke bilangan asli (32 bit).  S (16 bit) dikonversi dan disimpan di D.	Bilangan bulat (32 bit) dikonversi ke bilangan asli (32 bit).  S (32 bit) dikonversi dan disimpan di D.
FLT2INT (Mengonversi bilangan asli presisi tunggal ke bilangan bulat)	Bilangan asli (32 bit) dikonversi ke bilangan bulat (16 bit).  S dikonversi dan disimpan di D (16 bit).	Bilangan asli (32 bit) dikonversi ke bilangan bulat (32 bit).  S dikonversi dan disimpan di D (32 bit).

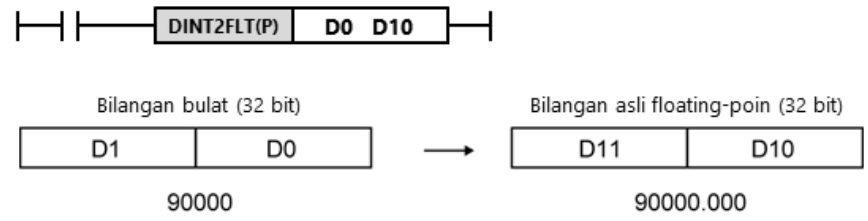
S (sumber): Data sebelum operasi (konstanta, nomor device)
D (destinasi): Destinasi data setelah operasi (nomor device)

2.11.5 Instruksi konversi antara bilangan bulat dan bilangan asli

Contoh aplikasi instruksi konversi bilangan bulat (16 bit) ke bilangan asli (32 bit)

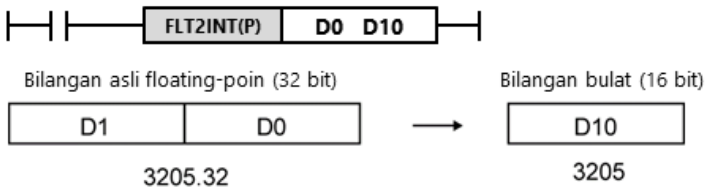


Contoh aplikasi instruksi konversi bilangan bulat (32 bit) ke bilangan asli (32 bit)

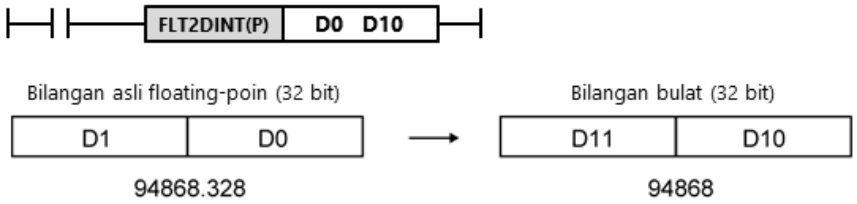


2.11.5 Instruksi konversi antara bilangan bulat dan bilangan asli

Contoh aplikasi instruksi konversi bilangan asli (32 bit) ke bilangan bulat (16 bit)



Contoh aplikasi instruksi konversi bilangan asli (32 bit) ke bilangan bulat (32 bit)



2.11.6 Menggunakan label yang merepresentasikan bilangan asli

Untuk menggunakan label untuk bilangan asli, atur tipe data menjadi "Single Precision" (Presisi Tunggal) atau "Double Precisition" (Presisi Ganda) pada jendela untuk mengatur label.

Label Name	Data Type
eData	FLOAT [Single Precision]
leData	FLOAT [Double Precision]

2.12 Ringkasan bab ini

Dalam bab ini, Anda telah mempelajari:

- Spesifikasi bit device word
- Spesifikasi transisi rendah ke tinggi (rising edge) atau transisi tinggi ke rendang (falling edge) untuk kontak
- Timer retentif
- Spesifikasi satuan pengukuran timer
- Register indeks
- Array
- Struktur
- Pengunci
- Register file
- Relai khusus, register khusus
- Kalkulasi dengan bilangan asli

Hal-hal penting

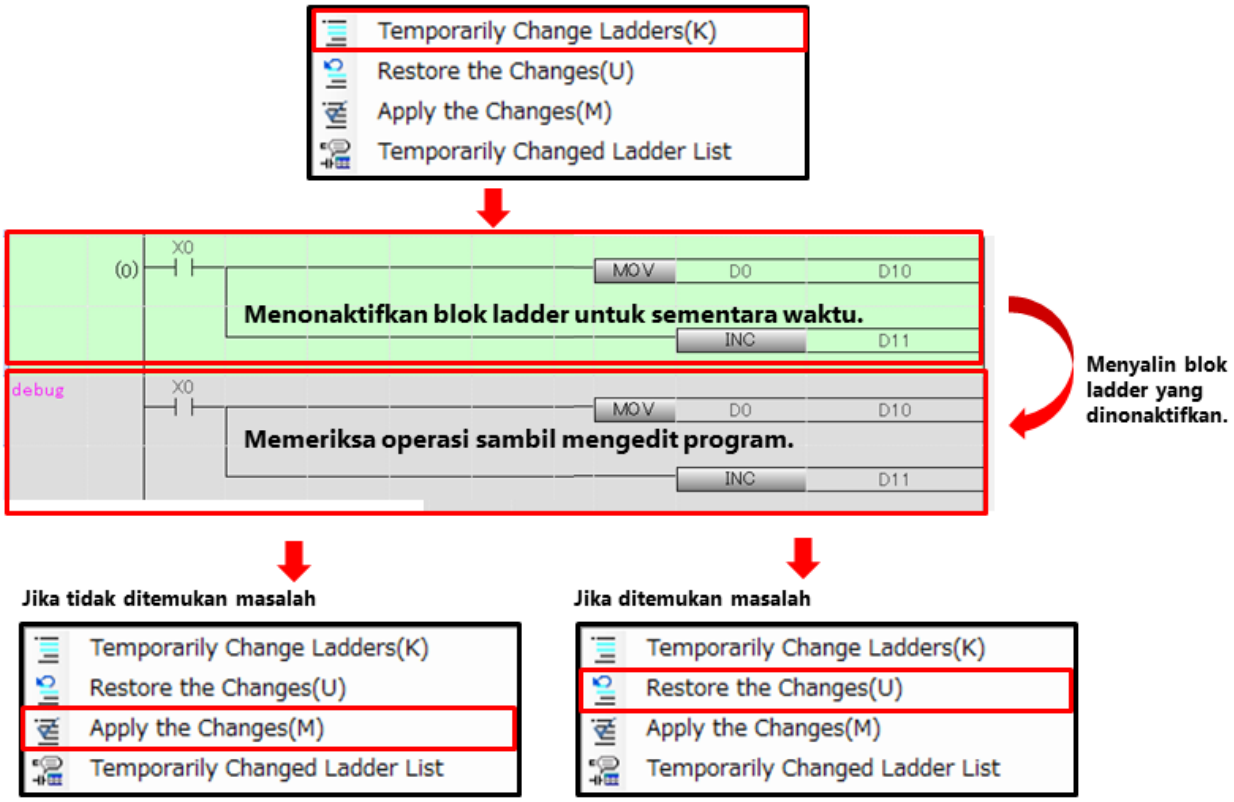
Timer retentif	Waktu yang diukur akan ditahan bahkan jika coil mati, dan pengukuran dilanjutkan ketika coil menyala kembali.
Satuan pengukuran timer	Satuan pengukuran timer dapat diubah dalam parameter.
Register indeks	Beberapa device dapat dideskripsikan dalam suatu batch.
Array	Beberapa nilai dapat ditangani dengan satu nama label.
Struktur	Beberapa label yang berkaitan dapat ditangani dengan satu nama label.
Dikunci	• Nilai device yang dikunci ditahan ketika modul CPU menghentikan operasi • Nilai device yang dikunci bisa dihapus dengan operasi memori CPU atau menggunakan instruksi program
Register file	• Dibandingkan dengan register data, register file dapat menangani jumlah data yang lebih besar • Nilai device ditahan ketika modul CPU berhenti beroperasi • Nilai device dapat dihapus dengan operasi memori CPU atau dengan menulis 0 pada device
Relai khusus, register khusus	Status internal modul CPU, seperti informasi diagnostik dan sistim informasi, telah disimpan dalam device ini.
Bilangan asli	• Menggunakan setidaknya dua word device (32 bit) • Terdapat petunjuk pengoperasian khusus • Bilangan bulat dan bilangan asli tidak dapat dicampur dalam satu operasi

Bab 3 Debugging yang efisien

Bab ini menjelaskan fungsi GX Works3 untuk debugging yang efisien.

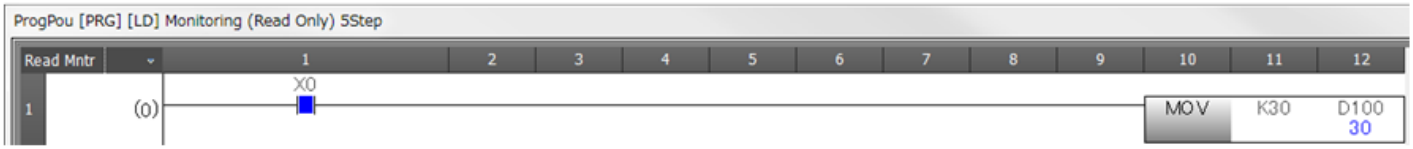
3.1 Mengubah rentang program untuk sementara waktu

Saat mengedit berbagai program untuk debugging, sangat sulit untuk membatalkan semua perubahan.
Dengan menonaktifkan blok ladder yang diinginkan untuk sementara waktu, pengguna dapat menggunakan salinan program untuk debugging tanpa mengganti yang asli.
(Fungsi temporarily change ladders (mengubah ladder untuk sementara waktu))
Setelah mengganti blok ladder dan memeriksa fungsi operasinya, perubahan akan diterapkan jika tidak terdapat masalah atau perubahan akan dipulihkan jika terdapat masalah.



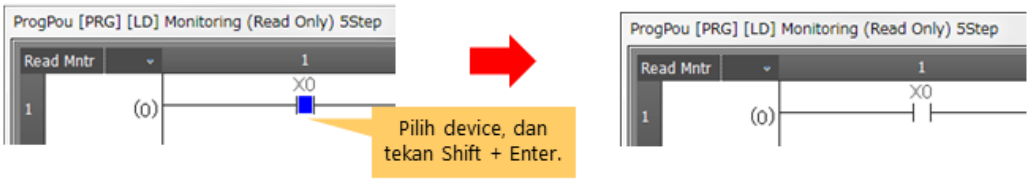
3.2 Memeriksa operasi sambil mengubah nilai dari device

Saat menjalankan program yang dibuat, status on/off device bit dan nilai yang disimpan dalam device word dapat ditampilkan di editor program. (Fungsi Monitor (Pemantauan))
Dengan fungsi monitor, pengguna dapat dengan mudah memeriksa status pengoperasian program.

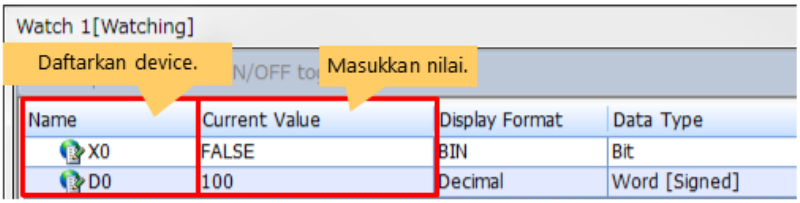


Nilai saat ini dari device dapat diubah secara paksa selama pemantauan. (Pengubah nilai saat ini)
Dengan fungsi pengubah nilai saat ini, perubahan dapat dilakukan tanpa mengedit seluruh program atau menjalankannya pada sistem yang sebenarnya.

Status device bit dapat diubah pada editor program.

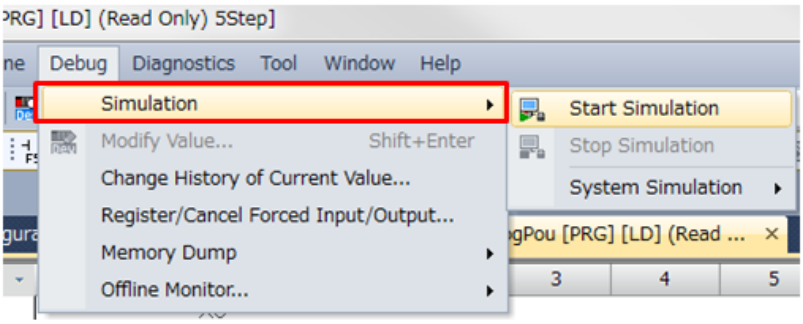


Dengan menggunakan jendela "Watch" (Lihat), device word yang akan dipantau dapat diregister dan nilai aktual dapat diubah.



3.3 Menyimulasikan operasi program

Jika program yang baru dibuat dijalankan pada sistem aktual, kesalahan yang tidak diharapkan dapat terjadi.
Operasi program dapat disimulasikan tanpa menggunakan pengontrol terprogram yang sebenarnya. (Fungsi Simulasi (simulasi))



Dengan fungsi simulasi, operasi program dapat diperiksa seolah-olah program berjalan pada pengontrol terprogram yang sebenarnya.

3.4 Ringkasan bab ini

Dalam bab ini, Anda telah mempelajari:

- Perubahan sementara blok ladder
- Memonitor program dan mengubah nilai aktual
- Simulasi program

Hal-hal penting

Perubahan sementara blok ladder	Blok ladder dapat dinonaktifkan untuk sementara waktu, dan program yang disalin dapat digunakan untuk debugging tanpa mengubah program asli.
Monitor	• Cara program berjalan dapat divisualisasikan. • Operasi program dapat diperiksa sambil mengubah nilai actual device.
Simulasi	Operasi program dapat disimulasikan tanpa menggunakan pengontrol terprogram yang sebenarnya.

3.5 Ringkasan kursus ini

Kursus e-learning ini telah selesai.

Berikut ringkasan kursus ini.

- **Pemrograman yang efisien**
  - Menetapkan nomor I/O pada modul agar mudah dimanfaatkan pada program dengan sistem yang berbeda
  - Menyetel area memori sesuai dengan status penggunaan device
  - Menggunakan label agar pemrograman menjadi lebih mudah dan operasi lebih mudah dipahami
  - Menambahkan komentar untuk meningkatkan keterbacaan program
- **Pemrograman lanjutan**
  - Menggunakan timer retentif untuk menahan waktu yang diukur
  - Menggunakan register indeks, array, atau struktur untuk menangani nilai secara kolektif
  - Menggunakan fungsi latch dan register file untuk menahan status device
  - Tersedia relai khusus dan register khusus yang menyimpan status internal modul CPU
  - Bilangan asli direpresentasikan oleh dua atau empat device word. Bilangan bulat dan bilangan asli tidak dapat dicampur dalam satu operasi
- **Debugging yang efisien**

  Pengguna dapat melakukan hal-hal berikut menggunakan GX Works3:
  - Debug program tanpa mengubah program asli
  - Memvisualisasikan cara program berjalan
  - Menyimulasikan operasi program

Untuk melanjutkan ke langkah berikutnya, ambil kursus tentang "penstrukturan" berikut, yang memisahkan program menjadi lapisan dan komponen sehingga dapat digunakan kembali dengan mudah.

- Pemrograman Efisien
- Pemrograman Efisien (Praktik) (akan dirilis)

Tes

Tes Akhir

Setelah menyelesaikan semua pelajaran dari kursus **Aplikasi Pemrograman (Diagram Ladder/MELSEC Seri iQ-R)**, kini Anda siap mengikuti tes akhir. Jika Anda masih kurang memahami salah satu topik yang dibahas, gunakan kesempatan ini untuk mengulas topik tersebut.

Total terdapat 14 pertanyaan (35 item) dalam Tes Akhir ini.

Anda dapat mengikuti tes akhir sebanyak mungkin.

Hasil penilaian

Jumlah jawaban yang benar, jumlah pertanyaan, persentase jawaban yang benar, dan hasil lulus/gagal akan ditampilkan pada halaman nilai.

Tes **Tes Akhir 1**

Manakah di antara kalimat-kalimat tentang pemilihan nomor I/O modul berikut yang benar?

T1

- **Nomor I/O dapat dipilih secara manual untuk setiap modul sehingga program tidak perlu dimodifikasi ketika mengubah konfigurasi modul**
- **Nomor I/O yang ditetapkan secara otomatis tidak dapat diubah**

Tes **Tes Akhir 2**

Manakah di antara kalimat-kalimat tentang pengaturan titik perangkat berikut yang benar?

T1

- **Setidaknya satu poin harus ditetapkan untuk setiap perangkat meski perangkat tersebut tidak digunakan**
- **Poin dapat ditetapkan sesuai dengan jumlah poin yang digunakan**

Tes **Tes Akhir 3**

Manakah di antara kalimat-kalimat tentang label berikut yang benar? (Jawaban ganda)

T1

- [ ] **Penggunaan label dapat membantu mengidentifikasi target yang diproses dan membuat pemrograman menjadi lebih mudah**
- [ ] **Label yang merepresentasikan sinyal dari modul dan nilai pengaturan disediakan**
- [ ] **Komentar dapat ditambahkan ke elemen untuk meningkatkan pembacaan program**
- [ ] **Karena konstanta dapat ditetapkan pada label, nilainya dapat diubah tanpa memodifikasi program.**

Tes **Tes Akhir 4**

Lengkapi teks yang menjelaskan pengatur timer retentifberikut ini.

Pengatur timer retentif memulai pengukuran ketika **(T1)** menyala (coil menjadi **(T2)**).
Retentive timer menahan nilai waktu yang diukur bahkan ketika kondisi input menjadi **(T3)**, dan melanjutkan pengukuran dari nilai yang ditahan saat kondisi input menjadi **(T4)** kembali.
Timer retentif mencapai nilai waktunya ketika waktu yang diukur mencapai nilai yang ditentukan, saat **(T5)** menyala.

T1 -- Select --

T2 -- Select --

T3 -- Select --

T4 -- Select --

T5 -- Select --

T6 -- Select --

Tes **Tes Akhir 5**

Lengkapi program kontrol yang menjalankan prosesberikut.

- Gunakan timer retentif (ST0) untuk mengukur waktu on sinyal input X0 atau X1
- Ketika waktu on X0 atau X1 mencapai 30 detik, nyalakan coil Y70 dan indikator waktu tercapai
- Ketika X2 menyala, matikan kontak timer retentif (ST0) dan hapus waktu yang diukur (nilai saat ini)

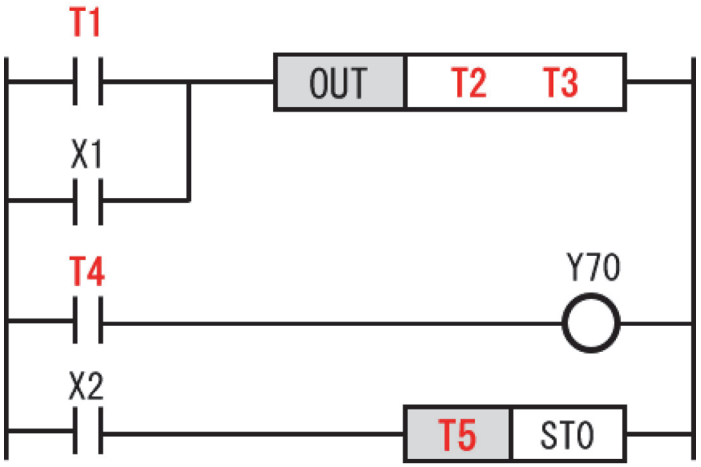
T1 -- Select --

T2 -- Select --

T3 -- Select --

T4 -- Select --

T5 -- Select --



[+]

Tes **Tes Akhir 6**

Pilih nilai yang tersimpan dalam register data D20 ketika X0 menyala pada masing-masing kondisi berikut dalam program kontrol di bawah ini.

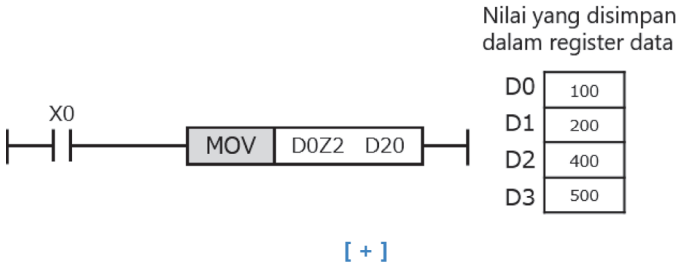
T1) Ketika nilai yang disimpan dalam Z2 adalah "0"
T2) Ketika nilai yang disimpan dalam Z2 adalah "1"
T3) Ketika nilai yang disimpan dalam Z2 adalah "2"

T1 -- Select --

T2 -- Select --

T3 -- Select --

T4 -- Select --



Nilai yang disimpan dalam register data

D0	100
D1	200
D2	400
D3	500

[+]

Tes **Tes Akhir 7**

Manakah di antara kalimat-kalimat tentang cara menentukan elemen array berikut yang benar?

T1

- **Tambahkan nomor elemen pada akhir nama label**
- **Tentukan nomor perangkat secara tidak langsung**

Tes **Tes Akhir 8**

Manakah di antara kalimat-kalimat tentang struktur berikut yang salah?

T1

- **Struktur digunakan secara kolektif tertata dan kondisinya tersimpan dan spesifikasi yang berkaitan dengan objek atau hal fisik**
- **Dengan menggunakan struktur, pemrosesan data dalam jumlah besar dapat dideskripsikan secara ringkas**
- **Member yang didefinisikan dalam struktur harus mempunyai tipe data yang sama**

Tes **Tes Akhir 9**

Manakah di antara kalimat-kalimat tentang fungsi latch berikut yang benar?

T1

- **Device pada dasarnya memiliki fungsi untuk menahan nilai**
- **Diperlukan pengaturan parameter untuk menahan nilai menggunakansoftware**

Tes **Tes Akhir 10**

Lengkapi teks yang menjelaskan register file berikut ini.

Register file adalah device word yang digunakan untuk memperluas register data (D), dan simbolnya adalah **(T1)**.
Register file memiliki kapasitas yang **(T2)** daripada register data, dan data yang disimpan **(T3)** bahkan ketika sistem dimatikan dayanya atau modul CPU direset.
Untuk menggunakan register file, pengaturan parameter **(T4)** menggunakan software.

T1 -- Select --

T2 -- Select --

T3 -- Select --

T4 -- Select --

Tes **Tes Akhir 11**

Manakah di antara kalimat-kalimat tentang relai khusus dan register khusus berikut yang benar?

T1

- **Status internal modul CPU telah tersimpan dalam relai khusus dan register khusus, dan perangkat-perangkat tersebut digunakan sebagai kondisi penentuan dalam program kontrol**
- **Fungsi khusus dapat ditetapkan dengan bebas pada relai khusus dan register khusus**

Tes **Tes Akhir 12**

Lengkapi teks yang menjelaskan bilangan asli (bilangan berkoma) berikut ini.

• Satu bilangan asli menggunakan **(T1)** device word dan disimpan dalam ruang memori **(T2)**-bit.
• Data nilai numerik untuk bilangan asli disebut **(T3)**. Sebagai contoh, 2,035 dideskripsikan sebagai **(T4)** dalam program kontrol.
• Bilangan integer dan bilangan asli **(T5)** dicampur dalam petunjuk pengoperasian yang menangani bilangan asli.

T1 -- Select --

T2 -- Select --

T3 -- Select --

T4 -- Select --

T5 -- Select --

Tes **Tes Akhir 13**

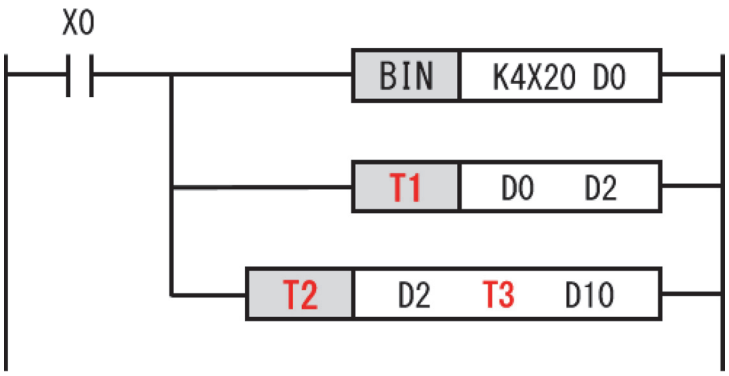
Lengkapi program kontrol yang menjalankan pemrosesan berikut ini.

- Baca data dari X20 hingga X2F (Data BCD) ketika X0 menyala, dan simpan dalam D0.
- Konversikan nilai dalam D0 menjadi bilangan asli, dan simpan nilai yang dikonversi dalam D2.
- Kalikan nilai dalam D2 dengan 3,14,dan simpan hasilnya dalam D10.

T1 -- Select --

T2 -- Select --

T3 -- Select --



[+]

Tes **Tes Akhir 14**

Manakah di antara kalimat-kalimat tentang debugging program kontrol berikut yang benar?

T1

- **Operasi program dapat disimulasikan dengan aman menggunakan fungsi pada software.**
- **Untuk debug program, program harus dijalankan di sistem aktual.**

Tes **Skor Tes**

Anda telah menyelesaikan Tes Akhir.
Hasil Anda adalah sebagai berikut.

		1	2	3	4	5	6	7	8	9	10
	Tes Akhir 1	✓									
	Tes Akhir 2	✓									
	Tes Akhir 3	✓									
	Tes Akhir 4	✓	✓	✓	✓	✓	✓				
	Tes Akhir 5	✓	✓	✓	✓	✓					
	Tes Akhir 6	✓	✓	✓	✓						
	Tes Akhir 7	✓									
	Tes Akhir 8	✓									
	Tes Akhir 9	✓									
	Tes Akhir 10	✓	✓	✓	✓						
	Tes Akhir 11	✓									
	Tes Akhir 12	✓	✓	✓	✓	✓					
	Tes Akhir 13	✓	✓	✓							
	Tes Akhir 14	✓									

Jumlah total pertanyaan : **35**

Jawaban yang benar : **35**

Persentase: **100** %

Hapus

Anda telah menyelesaikan kursus Aplikasi Pemrograman (Diagram Ladder/MELSEC Seri iQ-R).

Terima kasih telah mengikuti kursus ini.

Kami harap Anda menikmati pelajaran, dan kami harap informasi yang diperoleh dalam kursus ini dapat bermanfaat di masa mendatang.

Anda dapat mengulas kursus ini sesering yang Anda inginkan.

Tinjau

Tutup