PLC

Programowanie PLC (Język drabinkowy/seria MELSEC iQ-R)

Ten kurs jest adresowany do użytkowników, którzy znają podstawy sterowników programowalnych z serii MELSEC iQ-R i chcą nauczyć się bardziej zaawansowanego programowania.

L(CTS)00693POL

Wprowadzenie

Przeznaczenie kursu

Ten kurs jest przeznaczony dla użytkowników, którzy ukończyli szkolenie w zakresie podstaw programowania (Język drabinkowy) lub mają odpowiednią wiedzę w tym zakresie. W ramach kursu użytkownik uzyska wiedzę o skutecznym programowaniu i debugowaniu programowalnych sterowników z serii MELSEC iQ-R, zaawansowanym użyciu tych urządzeń i programowaniu z użyciem etykiet.

Warunkiem wstępnym przystąpienia do niniejszego kursu jest wcześniejsze ukończenie poniższych kursów lub posiadanie odpowiedniej wiedzy z tego zakresu.

- Seria MELSEC iQ-R - Podstawy
- Podstawy programowania

Wprowadzenie

Struktura kursu

Treść tego kursu posiada następującą strukturę.

Rozdział 1 – Skuteczne programowanie

Metody i ustawienia skutecznego programowania

Rozdział 2 – Zaawansowane programowanie

Zaawansowane użycie pamięci i programowanie z użyciem etykiet

Rozdział 3 – Skuteczne debugowanie

Funkcje oprogramowania projektowego używane do skutecznego debugowania

Test końcowy

Próg zaliczenia: 60% lub więcej

Jak korzystać z tego narzędzia do e-learningu

Przejdź do następnej strony	>	Przejdź do następnej strony.
Wróć do poprzedniej strony	<	Wróć do poprzedniej strony.
Przejdź do żądanej strony	TOC	Wyświetli się „Spis treści”, umożliwiający przejście do żądanej strony.
Zakończ naukę	X	Zakończ naukę.

Wskazówki dotyczące użytkowania

Zalecenia dotyczące bezpieczeństwa

Jeśli uczysz się, korzystając z rzeczywistych produktów, prosimy o dokładne przeczytanie zasad bezpieczeństwa zawartych w odpowiednich instrukcjach obsługi.

Środki ostrożności dla tego kursu

Ekrany wyświetlane dla wersji oprogramowania, którego używasz, mogą się różnić od przedstawionych w tym kursie. W ramach tego kursu użyto następującej wersji oprogramowania:

- GX Works3, wersja 1.044W

Rozdział 1 Wydajne programowanie

W tym rozdziale opisano ustawienia w oprogramowaniu umożliwiające wydajne programowanie oraz podstawy programowania z użyciem etykiet (Labels).

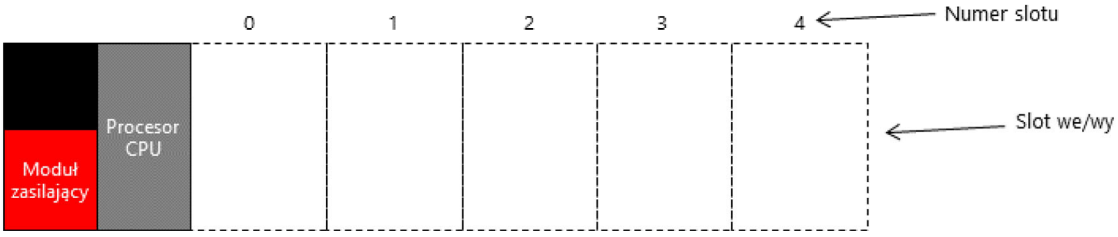
1.1 Łatwiejsze zastosowanie programów w różnych aplikacjach

W tym punkcie opisano rekomendowane przypisywanie adresów we/wy (I/O) w celu łatwiejszego zastosowania programów w różnych aplikacjach.

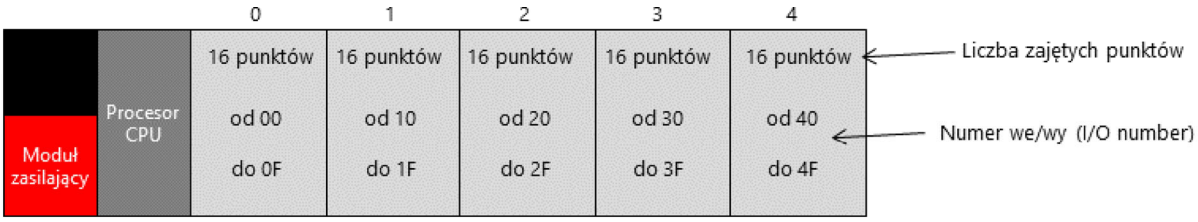
1.1.1 Automatyczne przypisywanie adresów we/wy (I/O numbers)

Adresy we/wy(I/O numbers) są przypisywane kolejno do modułów umieszczonych na płycie bazowej, zaczynając od slotów położonych najbliżej procesora CPU.
Adresy we/wy(I/O numbers) są przypisane w formacie szesnastkowym (od 0 do F).

Dostępna liczba punktów do przypisania różni się w zależności od typu modułu (16, 32, 64 itd.).



W poniższym przykładzie wykorzystano pięć 16-punktowych modułów.



1.1.1 Automatyczne przypisywanie adresów we/wy (I/O numbers)

W przypadku, kiedy w konfiguracji jednocześnie znajdują się moduły zajmujące 16, 32 i 64 punkty , adresy we/wy(I/O numbers) zostaną przypisane w następujący sposób:

		0	1	2	3	4
Moduł zasilający	Procesor CPU	16 punktów od 00 do 0F	32 punkty od 10 do 2F	64 punkty od 30 do 6F	32 punkty od 70 do 8F	16 punktów od 90 do 9F

W przypadku gdy w konfiguracji występuje pusty slot między zamontowanymi modułami, adresy we/wy(I/O numbers) zostaną przypisane również do pustego slotu.

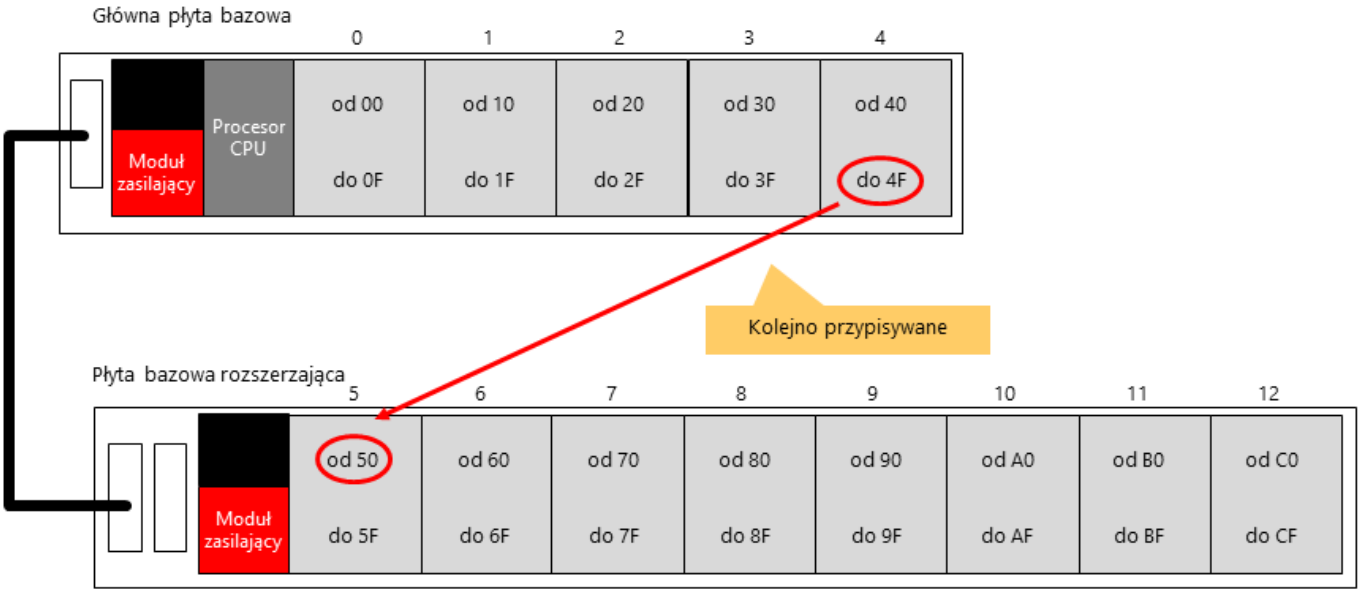
		0	1	2	3	4
Moduł zasilający	Procesor CPU	16 punktów od 00 do 0F	32 punkty od 10 do 2F	64 punkty od 30 do 6F	16 punktów od 70 do 7F	16 punktów od 80 do 8F

Pusty slot

Do pustego slotu, domyślnie zostaje przypisanych 16 punktów. Liczba przypisanych punktów może być zmieniona w zakresie od 0 do 1024 punktów (co 16 punktów).

1.1.1 Automatyczne przypisywanie adresów we/wy (I/O numbers)

Adresy we/wy(I/O numbers) rozszerzającej płyty bazowej są przypisywane automatycznie w kolejności od ostatniego numeru we/wy (I/O number) głównej płyty bazowej.

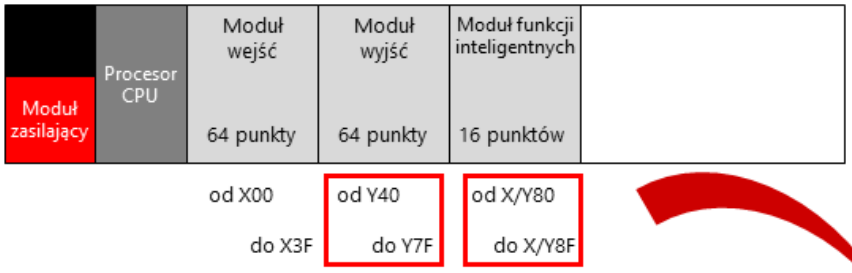


1.1.2 Przypisywanie adresów we/wy (I/O numbers) na sztywno

W przypadku ręcznego przypisywania adresów we/wy (I/O numbers) są one przypisane na sztywno i nie są zmieniane nawet przy zmianie konfiguracji sprzętowej. Oznacza to, że ten sam program może być używany do wykonania tej samej funkcjonalności niezależnie od konfiguracji sprzętowej.

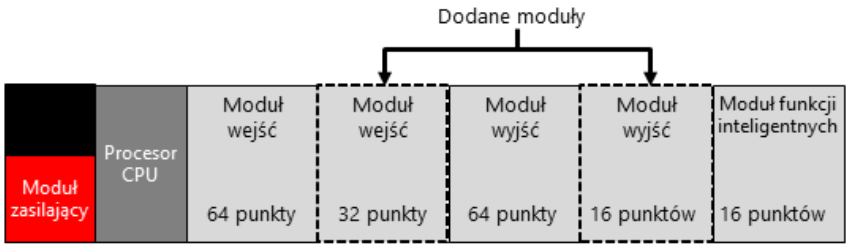
Przypisywanie automatyczne

Przed dodaniem modułów



od X00 do X3F | od Y40 do Y7F | od X/Y80 do X/Y8F

Po dodaniu modułów
(Dodawany jest 32-punktowy moduł wejściowy i 16-punktowy moduł wyjściowy.)



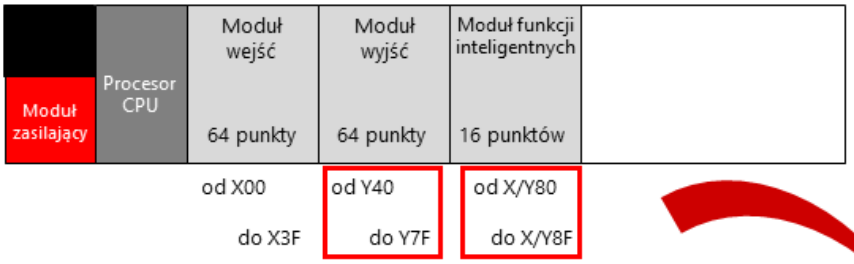
od X00 do X3F | od X40 do X5F | od Y60 do Y9F | od YA0 do YAF | od X/YB0 do X/YBF

Po dodaniu modułów adresy we/wy(I/O numbers) są przypisywane ponownie.

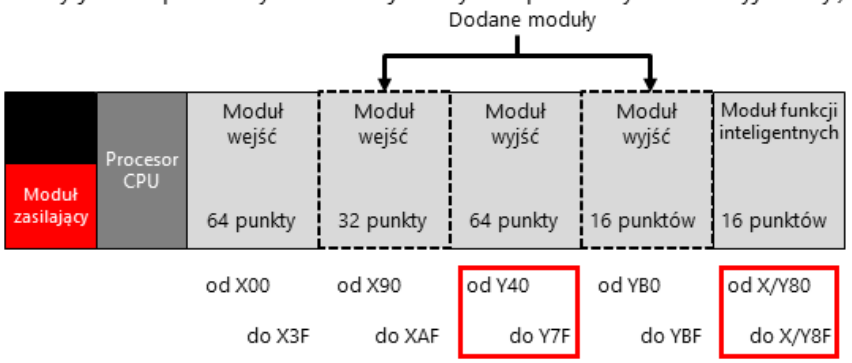
1.1.2 Przypisywanie adresów we/wy (I/O numbers) na sztywno

Ręczne przypisywanie adresów we/wy (I/O numbers) na sztywno

Przed dodaniem modułów



Po dodaniu modułów
(Dodawany jest 32-punktowy moduł wejściowy i 16-punktowy moduł wyjściowy.)

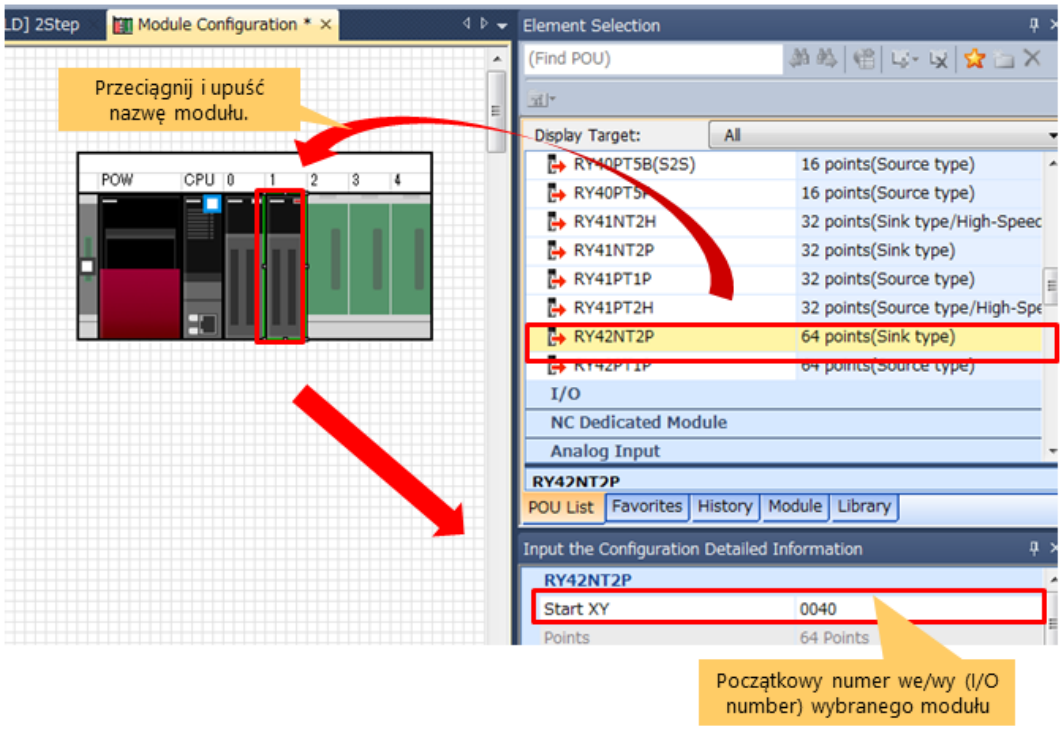


Niezależnie od dodanych modułów przypisane są te same adresy we/wy(I/O numbers).

Ponieważ adresy we/wy(I/O numbers) istniejących modułów pozostają niezmienione, konieczna jest modyfikacja jedynie programów związanych z nowo dodanymi modułami.

1.1.3 Automatyczne przypisywanie adresów we/wy(I/O numbers) przy użyciu okna konfiguracji sprzętowej

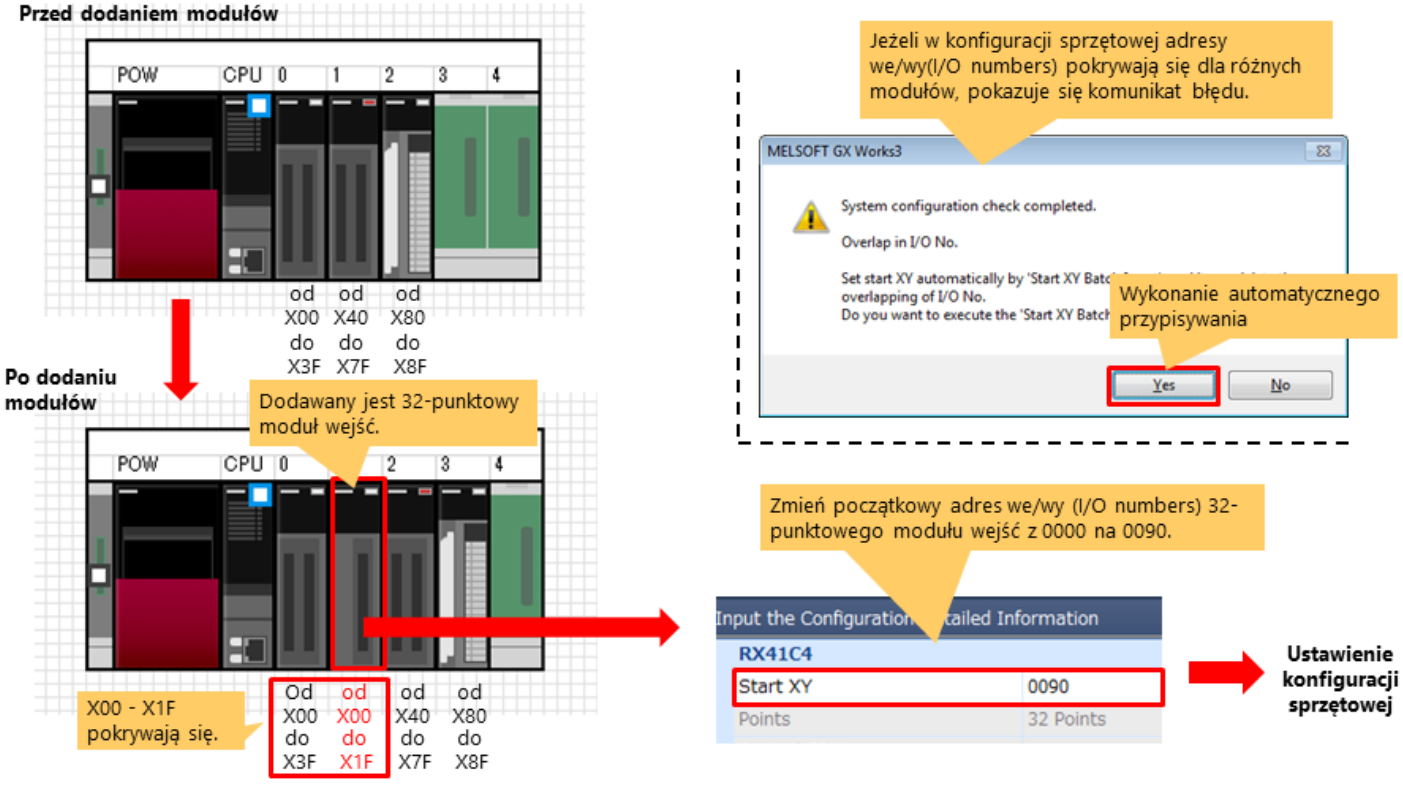
Konfiguracja sprzętowa może zostać wprowadzona z użyciem oprogramowania MELSOFT GX Works3.
Wybierz nazwę modułu, po czym przeciągnij ją i upuść na slot, w którym ma być umieszczony ten moduł.
Adresy we/wy(I/O numbers) są przypisywane kolejno do poszczególnych modułów w kolejności od modułu znajdującego się najbliżej modułu procesora (jak pokazano na diagramie).
Wybierz dodany moduł, aby zobaczyć początkowy numer we/wy (I/O numer) dla tego modułu.



1.1.4 Ręczne przypisywanie adresów we/wy (I/O numbers) z poziomu konfiguracji sprzętowej

Poniższy przykład przedstawia sposób ręcznego przypisywania adresów we/wy (I/O numbers) z poziomu konfiguracji sprzętowej GX Works3.

W przypadku zmiany konfiguracji sprzętowej przez dodanie nowego modułu, adresy we/wy(I/O numbers) dodanego modułu pokryją się z istniejącymi numerami, o ile nastąpi zmiana położenia istniejących modułów. Edytuj adresy we/wy(I/O numbers), aby się nie pokrywały i wprowadź je do konfiguracji sprzętowej.
Jeżeli w konfiguracji sprzętowej adresy we/wy(I/O numbers) pokrywają się dla różnych modułów, wyświetlony zostaje komunikat o wystąpieniu błędu. Można wtedy wykonać automatyczne przypisywanie numerów z poziomu okna zawierającego komunikat o wystąpieniu błędu.



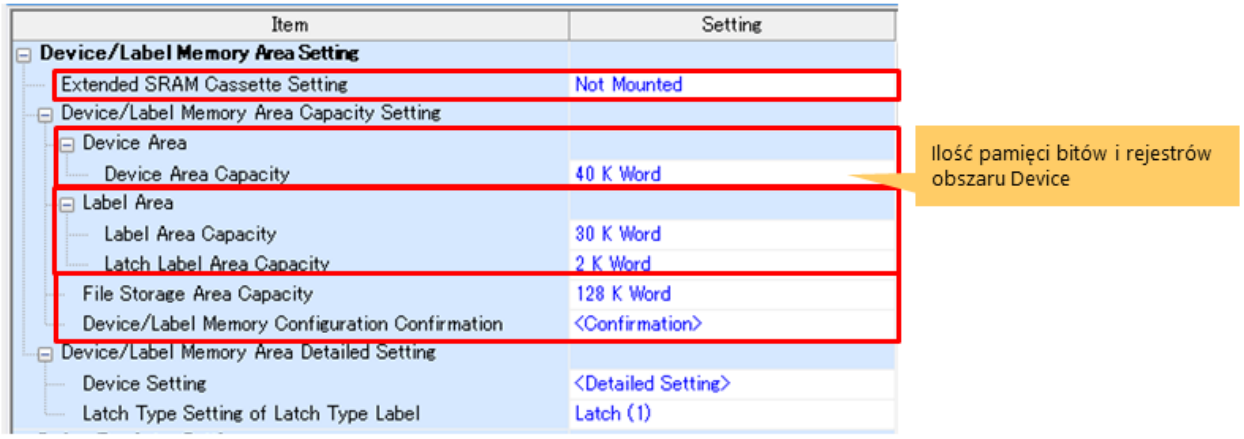
1.2 Ustawianie obszaru pamięci w zależności od wykorzystywanych bitów i rejestrów obszaru Device

1.2.1 Ustawienia pamięci danych

Liczba dostępnych punktów pamięci zależy od typu procesora. Początkowa ilość punktów pamięci jest przypisywana na podstawie rozmiaru pamięci Device Memory w procesorze CPU.

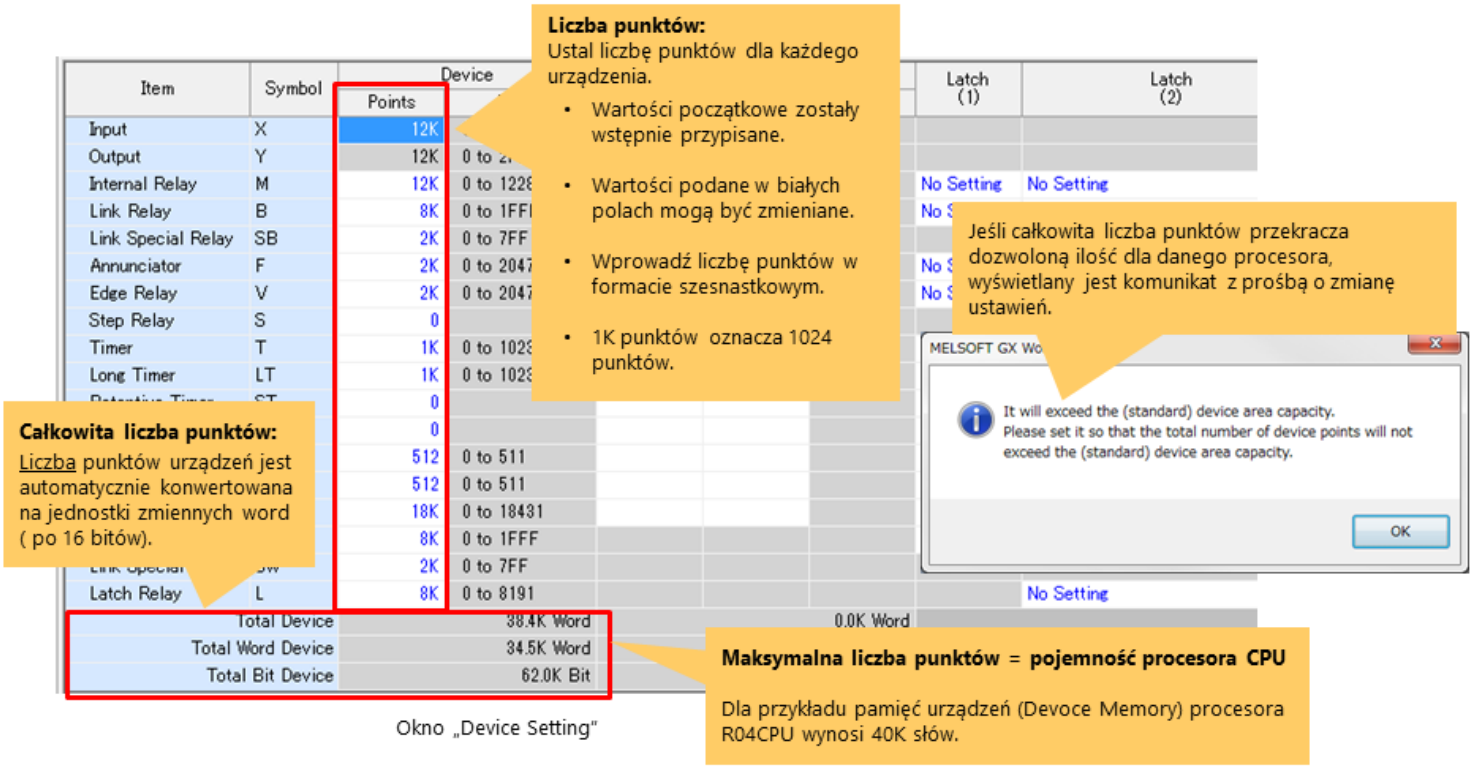
Pojemność procesora R04CPU wynosi 40K rejestrów.
Zmniejszenie niewykorzystanych obszarów pamięci zwiększa liczbę punktów w porównaniu do wartości początkowej.

Poniżej przedstawiono okno „Device/Label Memory Area Setting" zawierające parametry do ustawiania pamięci.
Ilość pamięci urządzeń (Device Memory) wzrasta wraz ze zmniejszaniem się ilości pamięci przeznaczonej dla etykiet (Label Memory) lub pamięci na przechowywania plików (file storage area).
Ponadto, ilość całkowitej pamięci Device/Label może być zwiększona przez użycie kasety SRAM.



1.2.2 Konfiguracja obszaru Device

Liczbę punktów przypisanych do każdego typu urządzenia można zmienić w oknie „Device Setting".
Początkowa wartość dla niektórych urządzeń wynosi 0 punktów. Przypisz pewną liczbę punktów tym urządzeniom, gdy ich używasz.



Item	Symbol	Device Points		Latch (1)	Latch (2)
Input	X	12K			
Output	Y	12K	0 to 2		
Internal Relay	M	12K	0 to 1228	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 2047	No S	
Edge Relay	V	2K	0 to 2047	No S	
Step Relay	S	0			
Timer	T	1K	0 to 1023		
Long Timer	LT	1K	0 to 1023		
	ST	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			

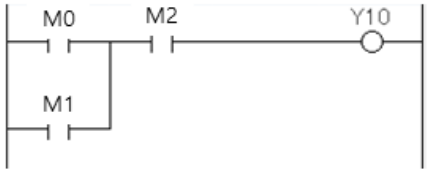
Okno „Device Setting"

1.3 Używanie etykiet (Labels) powiązanych z określonymi funkcjonalnościami aplikacji

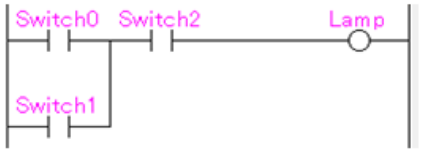
1.3.1 Zalety korzystania z etykiet (Labels)

Nazwy Device używane w programach muszą składać się z litery i cyfry – np. „M0" lub „D5".
Jeśli nazwa etykiety (Label) jest związana z zastosowaniem, tak jak np. „StartSwitch" , to łatwo określić jej rolę w aplikacji.
Nazwy etykiet (Labels) mogą być tworzone dowolnie. odpowiednio do zastosowania. Użytkownik nie musi brać pod uwagę numerów Device dla obszarów, w których używane są etykiety (Labels).

Program używający nazw Device



Program używający etykiet (Labels)



Etykiety (Labels) można zaklasyfikować do jednego z dwóch poniższych typów w zależności od obszaru zastosowania.

- **Etykiety globalne (Global labels)**
  Używane we wszystkich programach w obszarze danego projektu.

- **Etykiety lokalne**
  Mogą być używane wyłącznie w programie, do którego zostały przypisane.

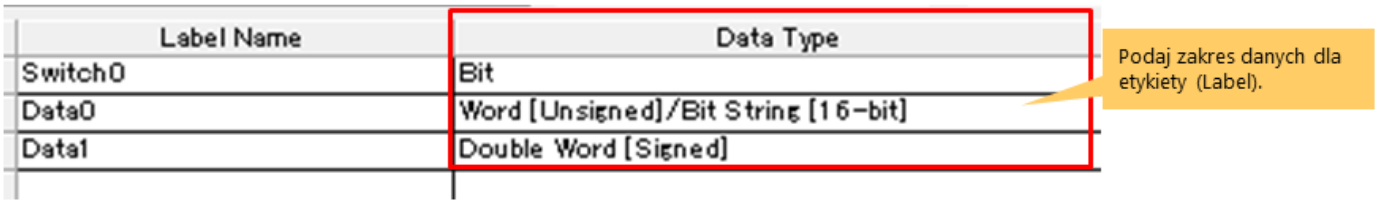
Podczas korzystania z etykiet (Labels) dla rzeczywistych urządzeń (X, Y) ich nazwy muszą być przypisane do etykiet globalnych (Global labels) za pomocą GX Works3.

1.3.2 Typy danych dla etykiet (Labels)

Do każdej etykiety (Label) musi być przypisany określony typ danych.
Typy danych zawierają bit i liczbę całkowitą (patrz poniżej).

Typ danych		Zakres danych
Typ Bit		Status wł./wył. zmiennych bitowych i prawda/fałsz operacji logicznych
Typ Integer	Word (unsigned)	od 0 do 65 535
	Word (signed)	od -32 768 do 32 767
	Double word (unsigned)	od 0 do 4 294 967 295
	Double word (signed)	od -2 147 483 648 do 2 147 483 647

W przypadku zmiennej całkowitej – typu integer wybierz opcję word lub double word zgodne z zakresem danych oraz signed lub unsigned – w zależności od tego, czy wartości będą mogły przyjmować wartości ujemne i dodatnie.
Wprowadź typ danych etykiety (Label) przy jej deklaracji w oprogramowaniu GX Works 3.



Okno ustawień etykiety (Label)

1.3.3 Nazwy etykiet (Labels) reprezentujące typy danych

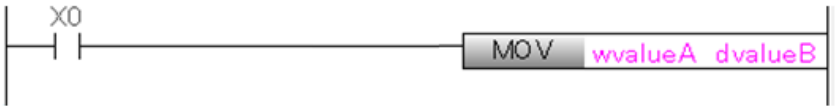
Użycie różnych typów danych na wejściu i wyjściu danej instrukcji może spowodować błąd konwersji lub nieprzewidziany wynik.
Poniżej podano przykładowy program prezentujący taki przypadek.

Wartości zmiennych typu Double Word nie mogą być przepisywane do zmiennych typu Word. Jednak typ danych nie może być określony za pomocą nazwy etykiety.

W celu ułatwienia określenia typu danych etykiet (Labels) do ich nazw można dodać prefiks określający typ danych.
Ten sposób tworzenia nazw etykiet (Labels) jest określany mianem „notacji węgierskiej".

Typ danych		Zakres danych	Prefiks	Rozszerzenie prefiksu
Typ Bit		Status wł./wył. zmiennych bitowych i prawda/fałsz operacji logicznych	b	**b**it
Typ Integer	Word (unsigned)	od 0 do 65 535	u	**u**nsigned word (słowo bez znaku)
	Word (signed)	od -32 768 do 32 767	w	signed **w**ord (słowo ze znakiem)
	Double word (unsigned)	od 0 do 4 294 967 295	ud	**u**nsigned **d**ouble-word (słowo podwójne bez znaku)
	Double word (signed)	od -2 147 483 648 do 2 147 483 647	d	signed **d**ouble-word (słowo podwójne ze znakiem)

Przykładowy program zamieszczony na górze tej strony może być zapisany z użyciem notacji węgierskiej:



Za pomocą notacji węgierskiej można zidentyfikować niezgodności typu danych podczas tworzenia programu.

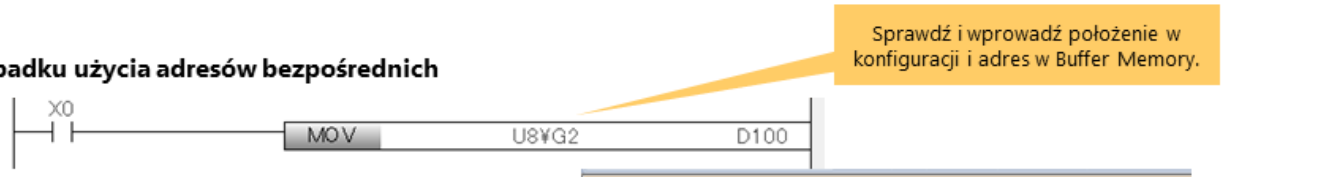
W dalszej części kursu nazwy etykiet (Labels) w przykładach są zapisane w notacji węgierskiej.

1.3.4 Użycie przygotowanych etykiet (Labels)

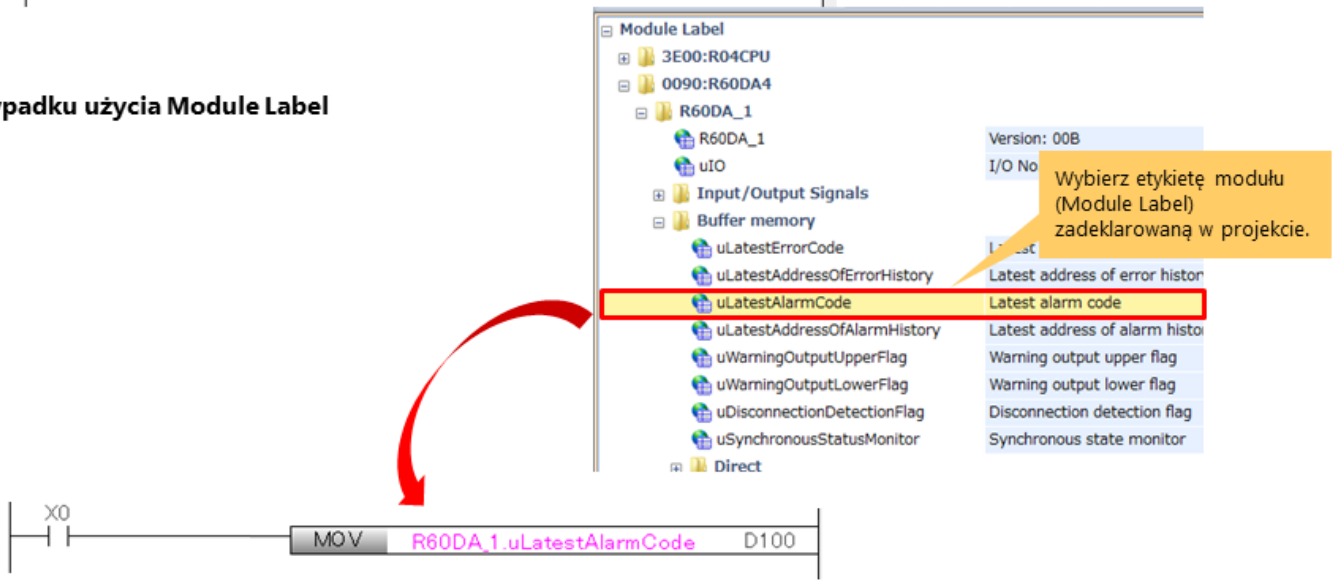
Jeśli konfiguracja sprzętowa została ustawiana w oknie konfiguracji, to etykiety powiązane z sygnałami modułów (module labels) lub wartości związane z umiejscowieniem modułu w konfiguracji są automatycznie rejestrowane.

W przypadku programowania z użyciem bezpośrednich adresów, ich numery i adresy pamięci buforowej muszą być wprowadzone zgodnie z wytycznymi zawartymi w dokumentacji. Czas programowania może być skrócony przez użycie etykiet (module labels), użytkownik może wybrać nazwę etykiety z listy.

W przypadku użycia adresów bezpośrednich



W przypadku użycia Module Label

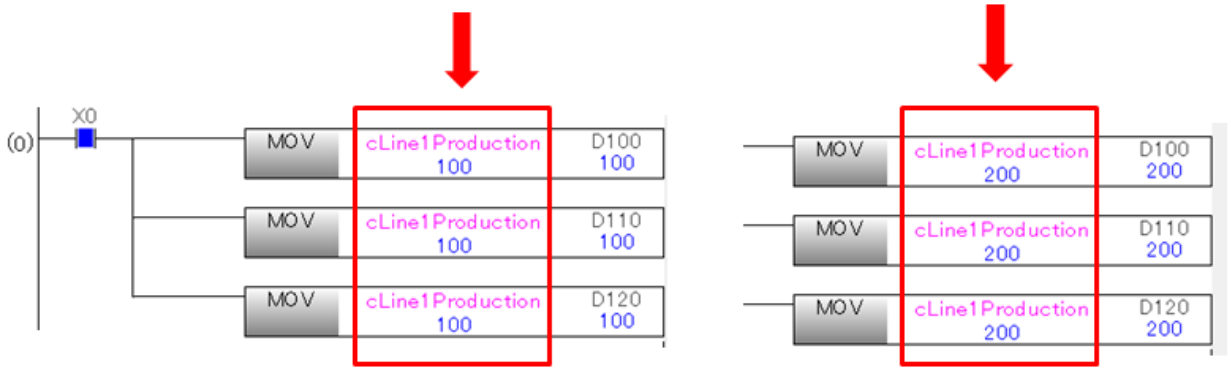


1.3.5 Przypisywanie wartości stałych do etykiet (Labels)

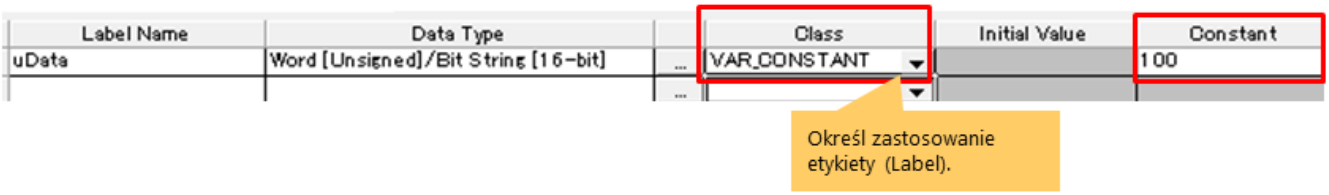
Do etykiet (Labels) można przypisywać wartości stałe.
Kiedy wartości stałe są przypisane do etykiet (Labels), ich wartości mogą być zmieniane bez ingerencji w program.
Jeśli ta sama stała jest używana dla wielu etykiet (Labels), to można ją zmienić za jednym razem dla wszystkich etykiet (Labels).

Przypisz stałą 100 do etykiety (Label) „cLine1Production”.

Przypisz stałą 200.



W celu przypisania stałej do etykiety zmień klasę określającą zastosowanie etykiety w oknie konfiguracji etykiet.
Dla etykiet lokalnych (Local Labels) wybierz „VAR_CONSTANT”.

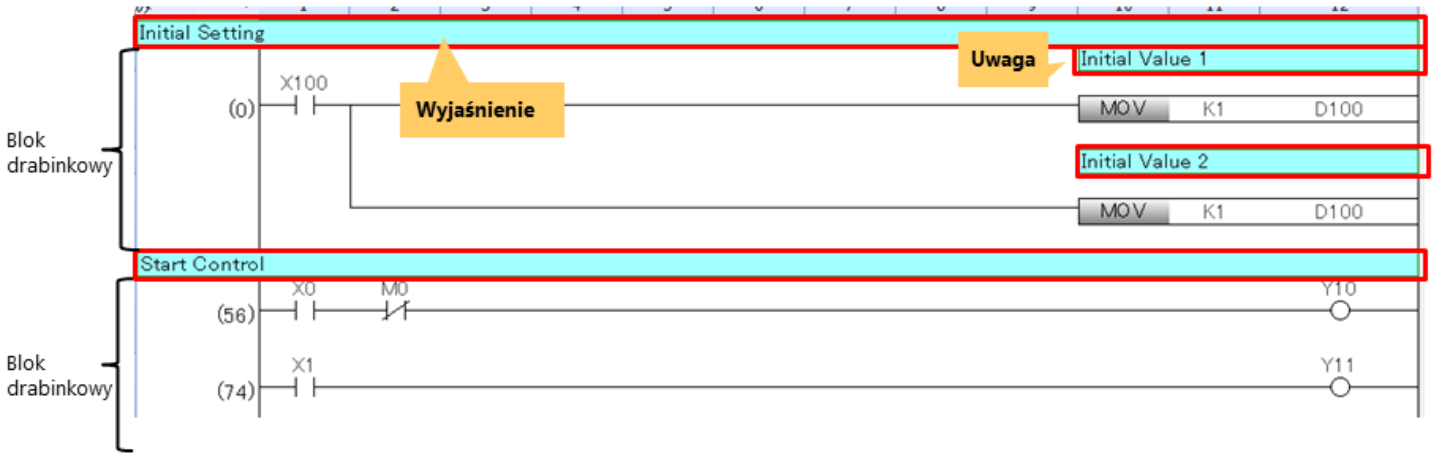


1.4 Poprawa czytelności programu

Do programu można dodawać komentarz (np. szczegóły operacji, nazwy Device).
Komentarze pomagają w wyjaśnieniu działania programu.

Komentarz może odnosić się do pojedynczego elementu lub pewnego zakresu programu.
W powyższym przykładzie programu „device/label comments" został dodany w celu wyjaśnienia zastosowania bitu/rejestru obszaru Device lub etykiety (Label) oraz typów podłączonych urządzeń we/wy (I/O devoce).

Ponadto typ komentarza zawierający „statement" (wyjaśnienie) jest dodawany do bloku drabinkowego w celu pomocy przy objaśnieniu wykonywania programu, natomiast „notes" (uwagi) – w celu opisania cewek i instrukcji.



1.5 Podsumowanie treści tego rozdziału

W tej części kursu udało Ci się przyswoić następujące zagadnienia:

- Przypisywanie adresów we/wy (I/O number)
- Dostosowywanie obszaru pamięci
- Programowanie z użyciem etykiet (Labels)
- Wprowadzanie komentarzy do programów

Istotne punkty

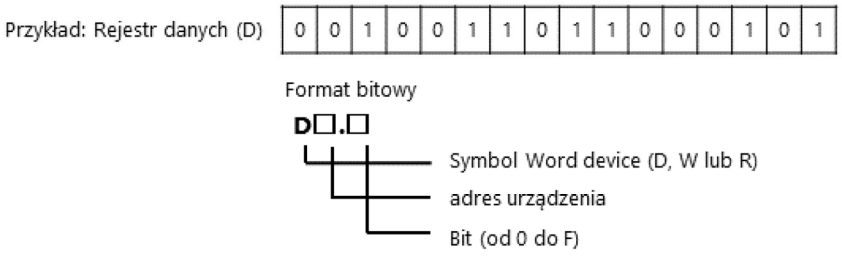
Przypisywanie adresów we/wy (I/O number)	• Adresy we/wy(I/O number) są automatycznie przypisywane do slotów w kolejności od slotu położonego najbliżej modułu procesora. • Przypisywanie stałych adresów we/wy (I/O numbers) modułom umożliwia zastosowania programów w różnych aplikacjach
Konfigurowanie bitów/rejestrów obszaru Device i obszaru pamięci	• Liczba punktów różni się w zależności od procesora CPU • Zmniejszenie niewykorzystywanych obszarów pamięci zwiększa liczbę punktów. • Liczba punktów może zostać zmieniona w odniesieniu do wykorzystywanych bitów i rejestrów obszaru Device
Programowanie z użyciem etykiet (Labels)	Jeśli nazwa etykiety (Label) jest związana z zastosowaniem, to łatwo określić jej rolę w aplikacji
Komentarze	Komentarze pomagają w wyjaśnieniu działania programu.

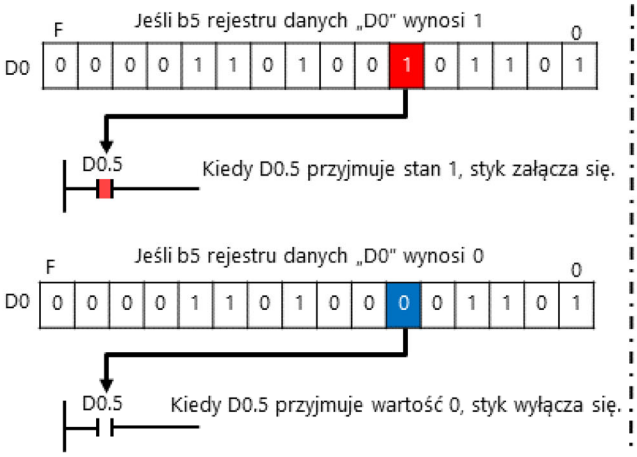
Rozdział 2 Zaawansowane programowanie

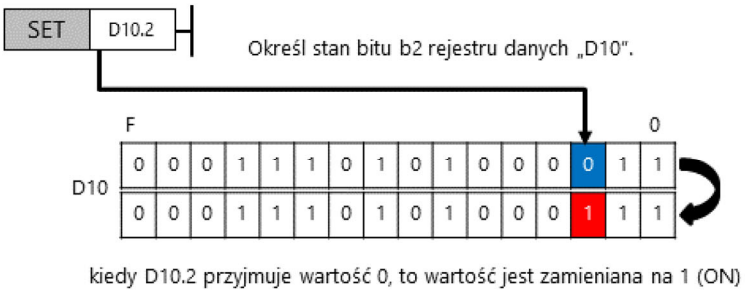
W tym rozdziale opisano zaawansowane użycie bitów/rejestrów obszaru Device i programowanie z użyciem etykiet (Labels).

2.1 Korzystanie z bitów w rejestrach

Rejestry danych są zazwyczaj wykorzystywane w jednostkach słów danych, ale mogą tez być wykorzystane z jednostkach bitowych.
Jednostka bitów jest używana do określania konkretnego bitu w rejestrze danych (D).





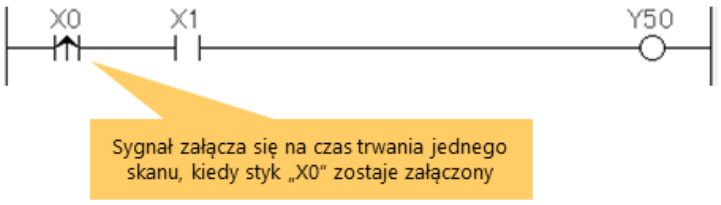


W celu użycia etykiet (Labels) dodaj opis „uData.2" i „uData.5".

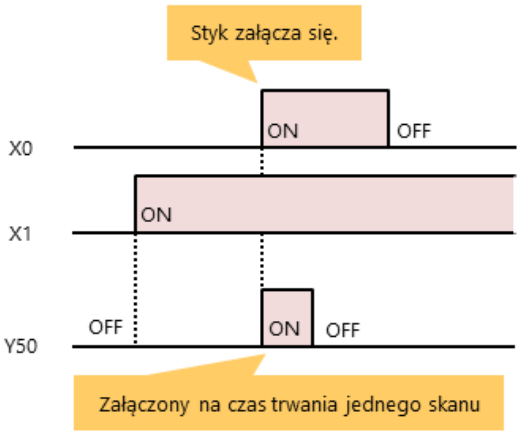
2.2 Załączenie bitu tylko w przypadku zmiany statusu styku

Można wprowadzić sygnał, który będzie załączał się na czas trwania jednego skanu przy wystąpieniu zbocza narastającego lub opadającego
Ta funkcja jest używana do sterowania zboczem warunku wejściowego.

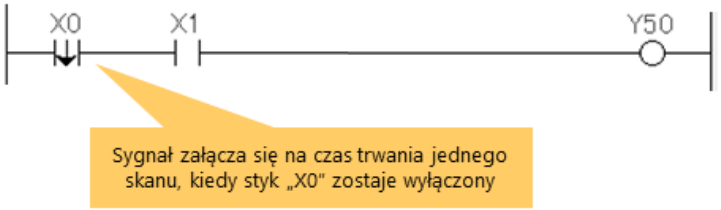
Styk reagujący na zbocze narastające



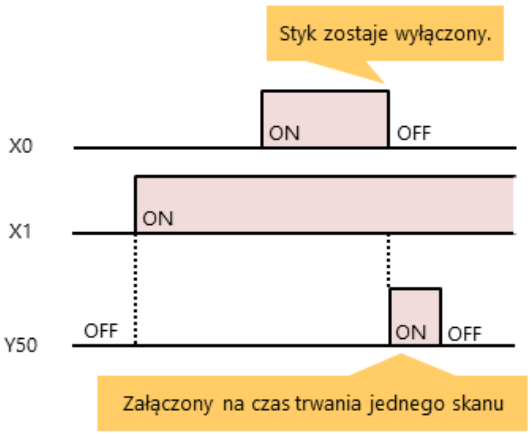
Sygnał załącza się na czas trwania jednego skanu, kiedy styk „X0" zostaje załączony



Styk reagujący na zbocze opadające



Sygnał załącza się na czas trwania jednego skanu, kiedy styk „X0" zostaje wyłączony



2.3 Zachowanie czasu pomiaru timer'a

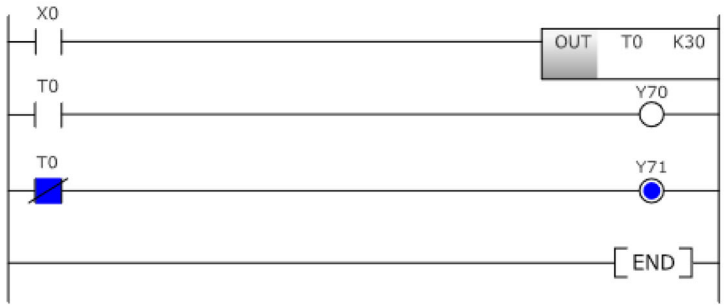
W tym punkcie opisano jeden z typów timer'ów: timer z podtrzymaniem. Podtrzymana zostaje wartość zmierzonego czasu.

2.3.1 Różnica między timer'em zwykłym a timer'em z podtrzymaniem (retentive timer)

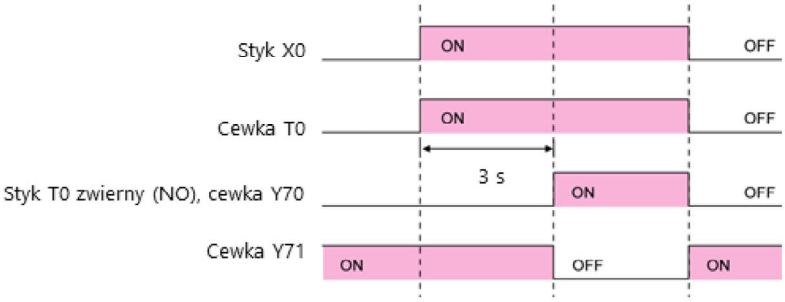
Zanim wyjaśnimy, czym jest timer z podtrzymaniem (retentive timer), przyjrzyjmy się ogólnej zasadzie działania timera.

Timer rozpoczyna pomiar po załączeniu cewki. Po upływie zadanego czasu styk zostaje załączony.
Po wyłączeniu cewki czas pomiaru jest resetowany do wartości „0". Symbolem timera jest litera „T".

Użyj przełącznika z prawej strony, aby zobaczyć, jak działa timer.



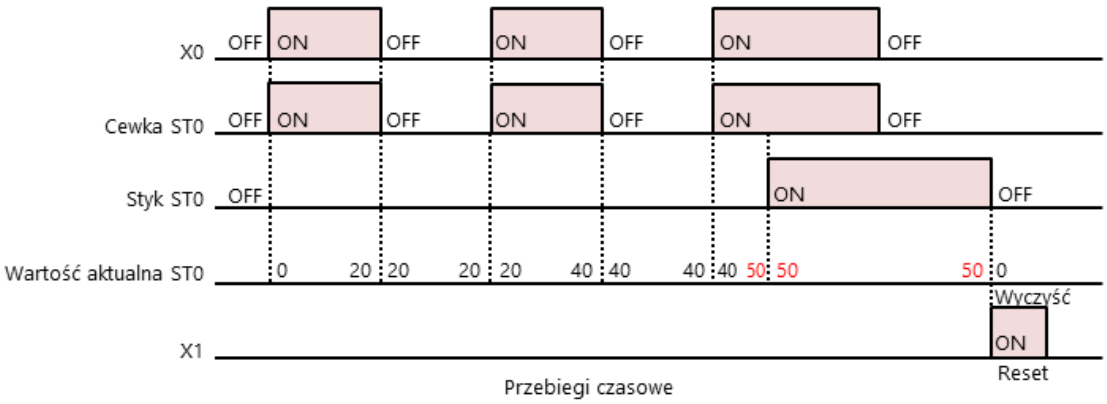
Po trzech sekundach od załączenia X0 załącza się Y70 i wyłącza Y71.



Przebiegi czasowe

2.3.1 Różnica między timer´em zwykłym a timer´em z podtrzymaniem (retentive timer)

Timer z podtrzymaniem (retentive timer) często jest używany do pomiaru całkowitego czasu pracy. Timer z podtrzymaniem (retentive timer) rozpoczyna pomiar po załączeniu cewki. Po upływie zadanego czasu styk zostaje załączony. Po wyłączeniu cewki czas pomiaru nie zostaje zresetowany. Po ponownym załączeniu cewki pomiar zostaje wznowiony od zachowanej wartości. Symbolem timer´a z podtrzymaniem (retentive timer) jest „ST".



Przebiegi czasowe

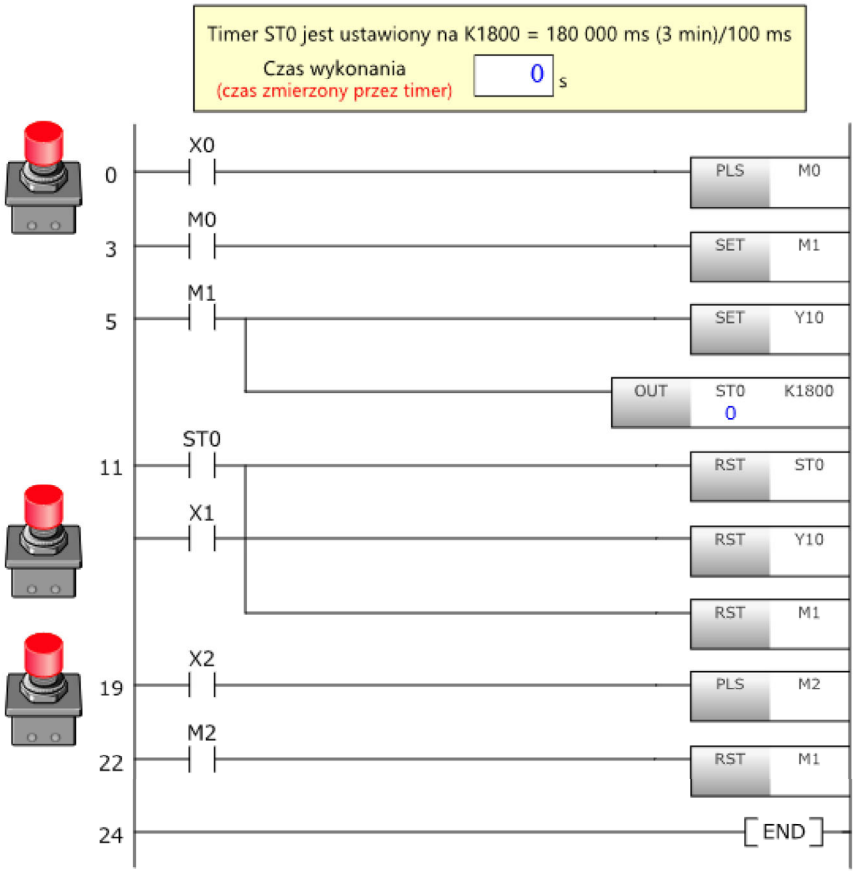
2.3.2 Przykładowy program z użyciem timer'a z podtrzymaniem (retentive timer)

Przyjrzyjmy się działaniu timer'a z podtrzymaniem (retentive timer) na podstawie symulacji pracy urządzenia z użyciem wejść (od X0 do X2).
* Timer z podtrzymaniem (retentive timer) (ST0) jest ustawiony w przyrostach co 100 ms.

od X0 do X2: Przełączniki wejściowe

Y10: Sygnał startu

1. Po załączeniu X0 timer rozpoczyna zliczanie.
2. Po włączeniu X2 timer zatrzymuje zliczanie, a aktualna wartość zostaje zachowana.
3. Po ponownym załączeniu X0 zliczanie jest kontynuowane.
4. Po załączeniu X1 zliczanie kończy się, a wartość bieżąca zostaje zresetowana.



2.3.3 Ustawienia timer'a z podtrzymaniem (retentive timer)

Liczba punktów używanych przez timer z podtrzymaniem (retentive timer) jest ustawiona domyślnie na wartość „0". Przed użyciem timer'a z podtrzymaniem (retentive timer) ustaw liczbę punktów w „Device Setting" procesora CPU za pomocą GX Works3.

W poniższym przykładzie dla timer'a z podtrzymaniem (retentive timer) ustawiono 64 punkty (od ST0 do ST63).

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 Używanie etykiety (Label) dla timer'a

Kiedy etykieta (Label) odnosi się do timer'a, należy ustawić „Timer" jako typ danych

Label Name	Data Type
uTimer1	Timer
uTimer2	Retentive Timer

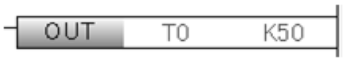
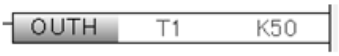
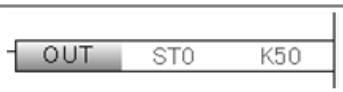
Wprowadzenie ustawień w „Device Setting" jest konieczne do użycia timer'a z podtrzymaniem (retentive timer) (jak opisano w poprzednim punkcie).
Nie jest to jednak konieczne w przypadku etykiet (Labels).

2.4 Zmiana jednostki pomiaru timer'a

Jednostki pomiaru różnią się w zależności od typu timer'a.

- Timer szybki (krótki pomiar)
- Timer wolny (długi pomiar)
- Long Timer umożliwia prowadzenie pomiarów przez długi czas

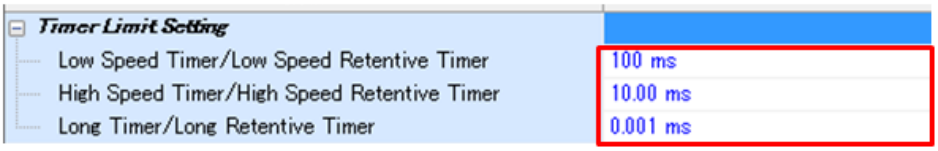
Każdy z powyższych timer'ów ma funkcję podtrzymania (retentive timer)

Typ	Jednostka pomiaru	Przykładowy program	Działanie	Podtrzymanie aktualnej wartości
Timer wolny	100 ms (wartość domyślna)		Timer wolny T0 odlicza 5 sekund.	16 bitów
Timer szybki	10,00 ms (wartość domyślna)		Timer szybki T1 odlicza 0.5 sekundy.	
Timer wolny z podtrzymaniem	100 ms (wartość domyślna)		Timer wolny z podtrzymaniem STO odlicza 5 sekund.	
Timer szybki z podtrzymaniem	10,00 ms (wartość domyślna)		Timer szybki z podtrzymaniem ST1 odlicza 0.5 sekundy.	
Long timer	0,001 ms (wartość domyślna)		Long timer LT0 odlicza 0,005 s.	32 bity
Long timer z podtrzymaniem			Long timer z podtrzymaniem LST0 odlicza 0,005 sekund.	

Początkowymi jednostkami pomiarowymi są: 100, 10 i 0,001 ms odpowiednio dla timer'ów: low-speed, high-speed timer i long.
Zmiana jednostki pomiarowej – patrz następna strona.

2.4 Zmiana jednostki pomiaru timer'a

Jednostkę pomiarową timer'a można zmienić w „Timer Limit Setting" procesora CPU.



2.5 Obsługa wielu urządzeń - rejestr indeksowy (Index register)

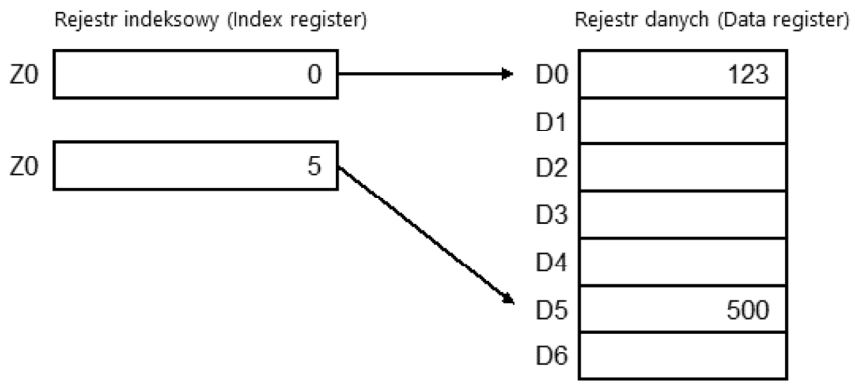
Rejestr indeksowy (Index register) (Z) jest używany w połączeniu z innym bitem/rejestrem do pośredniego określenia adresu docelowego bitu/rejestru. Ponieważ rejestr indeksowy (Index register) może określać całe pakiety rejestrów, jest on używany do uproszczenia programów.

- Rejestr indeksowy (Index register) jest podany za symbolem i adresem bitu/rejestru
- Numer rzeczywistego docelowego urządzenia sterującego = symbol urządzenia (numer urządzenia + rejestr indeksowy (Index register))
- Liczba punktów dla rejestru indeksowego (Index register) jest ustalana domyślnie na 20 (od Z0 do Z19).

2.5.1 Przykład zastosowania rejestru indeksowego (Index register)

Jeśli urządzenie jest oznaczone jako „D0Z0", oznacza to D(0+Z0).

Przykład) Jeśli Z0 = 0, numer rejestru = D0.
Jeśli Z0 = 5, numer rejestru = D5.

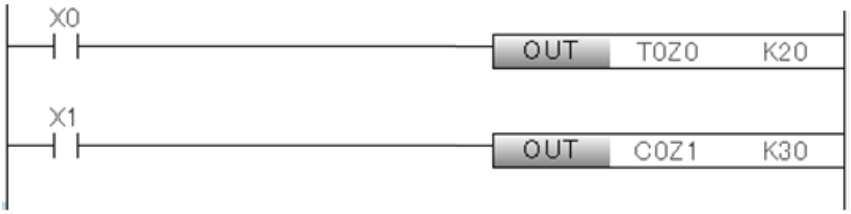


2.5.2 Bity/rejestry obszaru Device, które można modyfikować za pomocą rejestru indeksowego (Index register)

Do bitóe/rejestrów obszaru Device, które można modyfikować za pomocą rejestru indeksowego (Index register) należą:

Bit obszaru Device	X, Y, M, L, S, B, F
Rejestr	T, C, D, R, W
Stała	K, H
Wskaźnik	P

Uwaga) Do styków i cewek używanych w timerach i licznikach można używać tylko rejestru indeksowego (Index register) „Z0" lub „Z1".



Jeśli Z0 = 1, T1 odmierza czas.

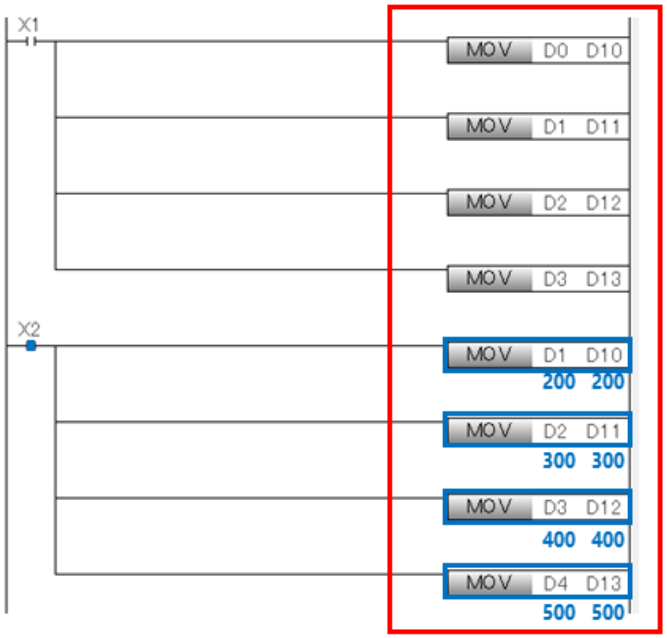
Jeśli Z1 = 5, C5 wykonuje zliczanie.

2.5.3 Uproszczenie programów za pomocą rejestru indeksowego (Index register)

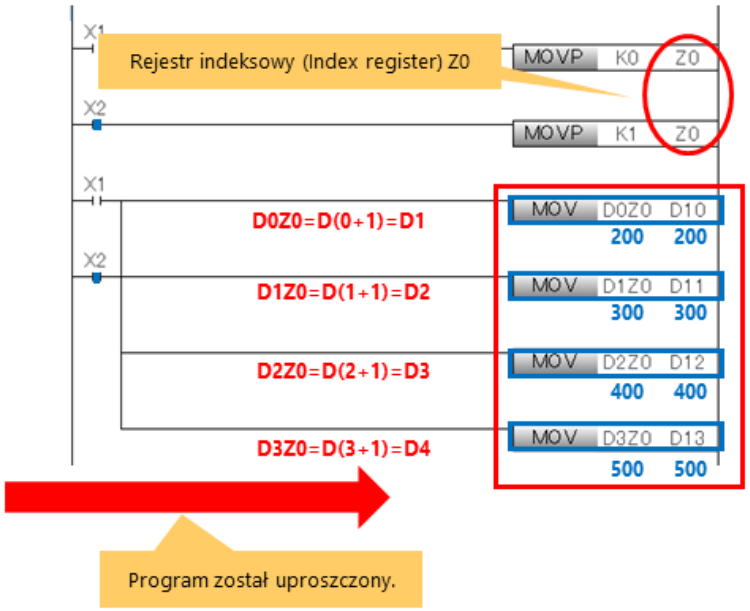
Po załączeniu X1 lub X2 poniższe programy przenoszą wartości zawarte w „od D0 do D4" do „od D10 do D13". Programy (1) i (2) dają ten sam wynik. W programie (1) dane są przenoszone bezpośrednio, natomiast w programie (2) – pośrednio przez rejestr indeksowy (Index register).

Początkowo zapamiętane wartości
D0=100
D1=200
D2=300
D3=400
D4=500

(1) Jeśli rejestr indeksowy (Index register) nie jest używany



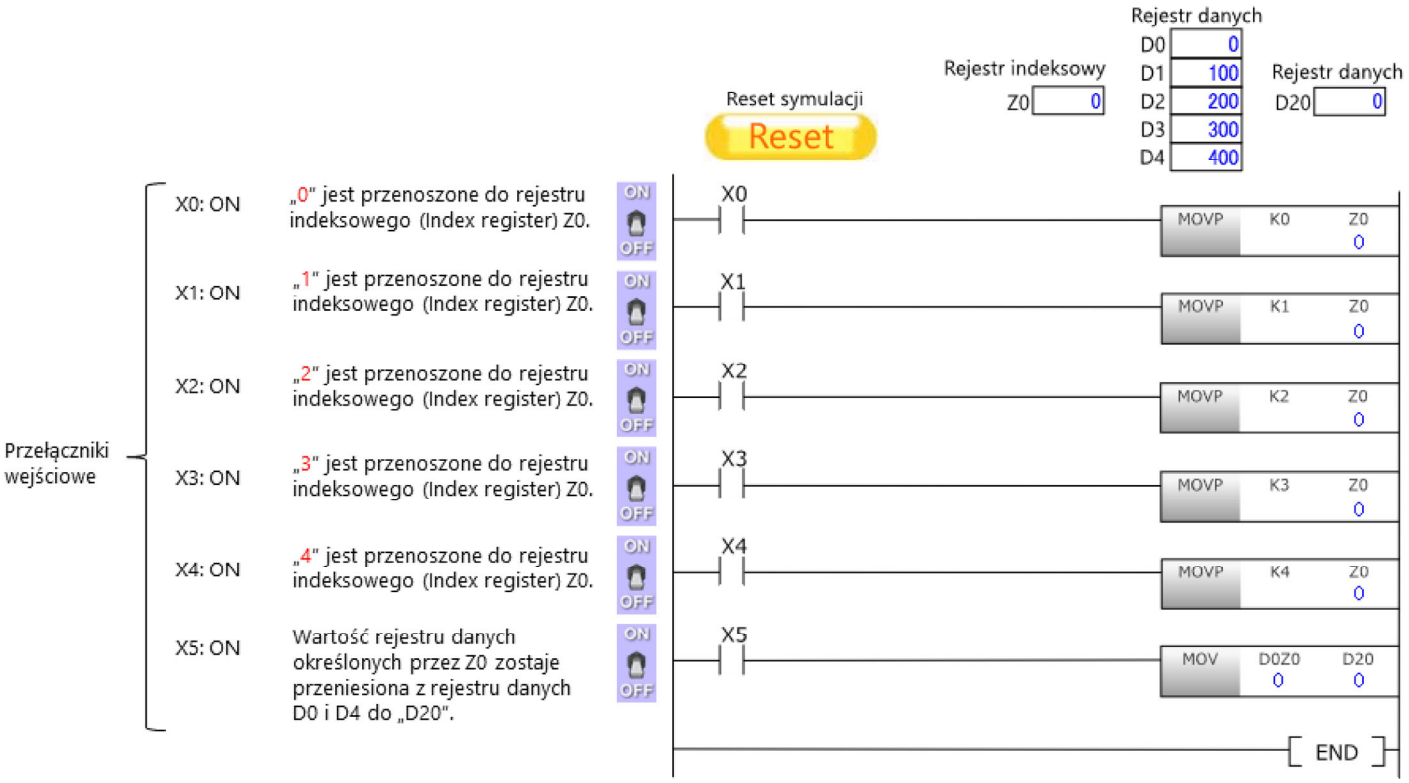
(2) Jeśli rejestr indeksowy (Index register) jest używany



Program został uproszczony.

2.5.4 Przykładowy program z użyciem rejestru indeksowego (Index register)

W celu przeprowadzenia symulacji działania rejestru indeksowego (Index register) Z0, należy załączyć wejścia od X0 do X5.
* Od K0 do K400 zostały zapisane do rejestrów danych D0 do D4.
W celu sprawdzenia zapisanych wartości, należy załączyć wejścia od X0 do X5.



2.6 Obsługa wielu wartości – tablica (array)

Przy użyciu tablicy (array) można obsłużyć wiele wartości za pomocą jednej etykiety (Label).
W poniższym przykładzie zapamiętywane są dane o wielkości produkcji fabryki samochodów zgodnie z ich miejscem docelowym

Miejsce docelowe	Country A	Country B	Country C
Wielkość produkcji	35 jednostek	75 jednostek	65 jednostek

Dane o wielkości produkcji uszeregowane wg miejsca docelowego są przypisane do etykiety (Label).
W przypadku nieużywania tablicy (array) nazwy etykiet (Label) muszą być utworzone dla każdego miejsca docelowego.
Korzystając z tablicy (array), można przyporządkować i zapisać wielkość produkcji dla wielu miejsc docelowych pod jedną nazwą etykiety (Label).

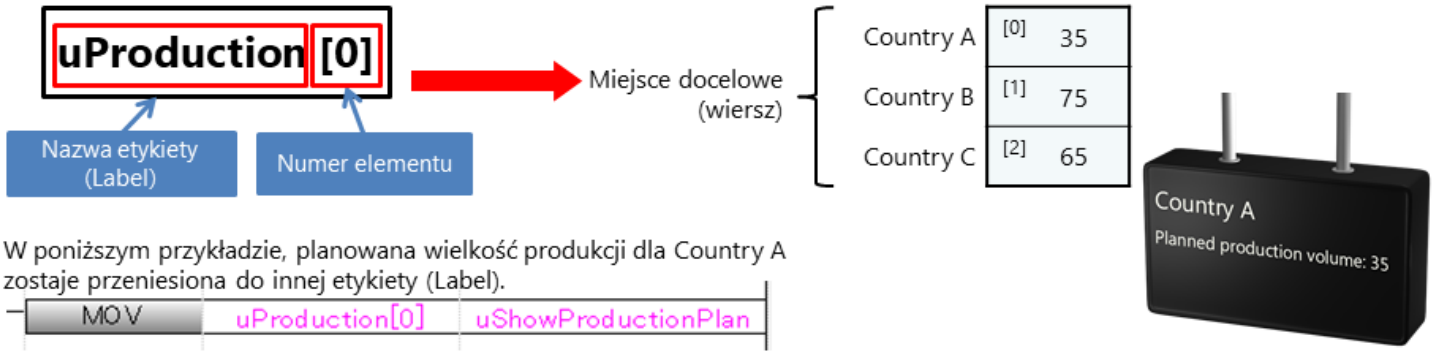
W przypadku nieużywania tablicy (array)

uProductionA
uProductionB
uProductionC

W przypadku użycia tablicy (array)

uProduction

Poszczególne etykiety (Label) w tablicy (array) są określane za pomocą numerów elementów. Numery elementów zaczynają się od [0].



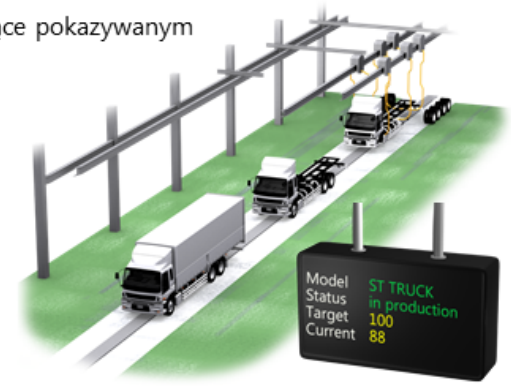
W poniższym przykładzie, planowana wielkość produkcji dla Country A zostaje przeniesiona do innej etykiety (Label).

2.7 Obsługa wielu zmiennych (structure)

Dzięki strukturze można obsłużyć wiele powiązanych ze sobą etykiet (Label) za pomocą jednej nazwy etykiety (Label). W poniższym przykładzie status linii produkcji samochodów jest wyświetlany na Andon (wyświetlaczu).

Poniższa tabela pokazuje nazwy etykiet (Label), wartości i typy danych odpowiadające pokazywanym pozycjom.

Pozycja	Nazwa etykiety (Label)	Wartość	Typ danych dla etykiety (Label)
Model	sModel	'ST TRUCK'	Typ String
Status operacyjny	bStatus	'in production'	Typ Bit
Dzisiejsza zamierzona wielkość produkcji	uPlanQty	100 jednostek	Integer type (word, unsigned)
Aktualna wielkość produkcji	uActualQty	88 jednostek	Integer type (word, unsigned)



Jeśli struktura nie jest używana dla zakładu mającego większą liczbę linii produkcyjnych, to konieczna jest zmiana nazwy etykiety (Label) dla każdej z tych linii.
Poniżej przedstawiono przykłady nazw etykiet (Labels) z dodanymi nazwami linii produkcyjnych.

Pierwsza linia produkcyjna

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

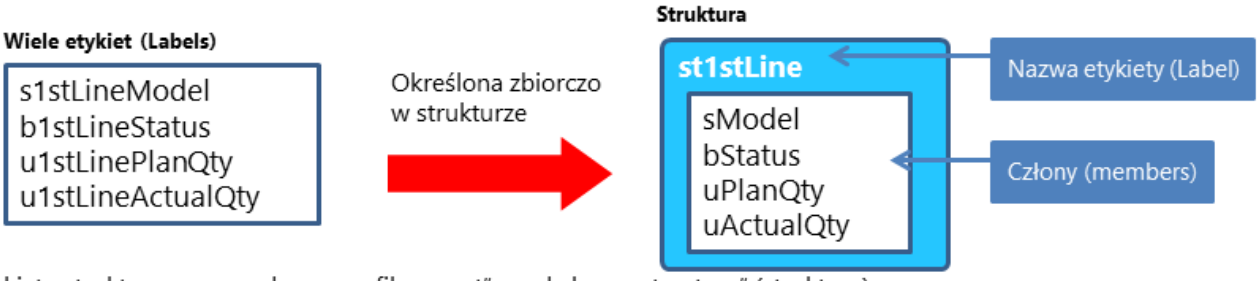
Druga linia produkcyjna

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

Wraz ze wzrostem liczby linii produkcyjnych wzrasta liczba etykiet (Labels) koniecznych do ich obsługi. W konsekwencji, program staje się dłuższy i mniej czytelny.

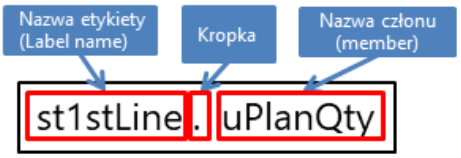
2.7 Obsługa wielu zmiennych (structure)

Struktura umożliwia użycie jednej nazwy etykiety (Label) do reprezentowania wielu etykiet (Labels) związanych z jedną linią produkcyjną.
Dlatego struktury są używane do zbiorczej organizacji, zapamiętywania i obsługiwania warunków i specyfikacji związanych z obiektami fizycznymi i materiałami, takimi jak urządzenia, wyposażenie i obrabiane detale.

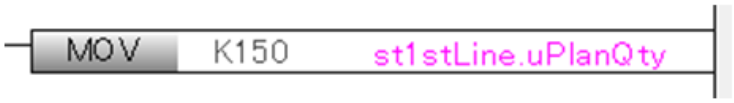


Etykiety struktur są poprzedzane prefiksem „st" - od słowa „structure" (struktura).
Poszczególne etykiety (Labels) określone przez strukturę są nazywane „członami" (members). Każdy człon (member) może mieć różny typ danych.

W celu identyfikacji członu (member) w strukturze dodaj jego nazwę po nazwie etykiety struktury, rozdzielając wpisy kropką (.).



W poniższym przykładzie, stała jest przypisana do etykiety (Label) członu (member) struktury w pierwszej linii produkcyjnej.



2.8 Zachowanie statusu bitów/rejestrów (latch)

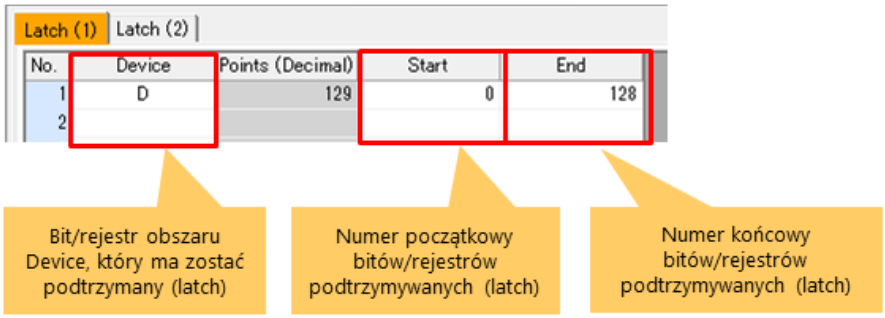
W tym punkcie opisano funkcję podtrzymywania latch, która zachowuje wartości bitów i rejestrów obszaru Device kiedy procesor CPU kończy wykonywanie opracji
Nawet w przypadku awarii zasilania trwającej dłużej niż dopuszczalny chwilowy zanik mocy, PLC może ponownie uruchomić sekwencję z danymi zapisanymi w momencie zatrzymania pracy.

Jeśli funkcja podtrzymywania latch nie jest używana, wartości bitów i rejestrów zostają zresetowane do wartości początkowych (OFF dla bitów i „0" dla rejestrów word). Dzieje się to w następujących przypadkach:

- Wyłączenie zasilania
- Zresetowanie przez naciśnięcie przycisku RUN/STOP/RESET
- Brak zasilania przez czas dłuższy niż dopuszczalny chwilowy zanik mocy

2.8.1 Ustawianie funkcji podtrzymywania (latch)

Ustaw zakres podtrzymywania (latch) w oknie „Device Setting" procesora CPU
Poniżej przedstawiono przykład ustawienia podtrzymywania (latch) rejestrów danych od D0 do D128.



Bit/rejestr obszaru Device, który ma zostać podtrzymany (latch)

Numer początkowy bitów/rejestrów podtrzymywanych (latch)

Numer końcowy bitów/rejestrów podtrzymywanych (latch)

2.8.2 Typy podtrzymywania (latch) i metody usuwania danych

W zależności od metody usuwania danych rozróżnia się dwa typy podtrzymywania: typ (1) i typ (2).

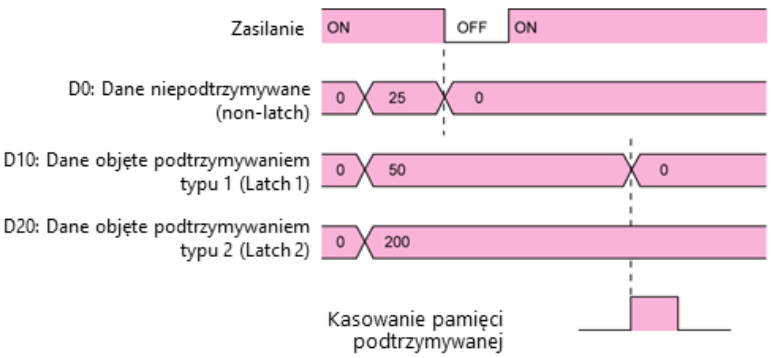
- Latch (Podtrzymywanie typu) (1)

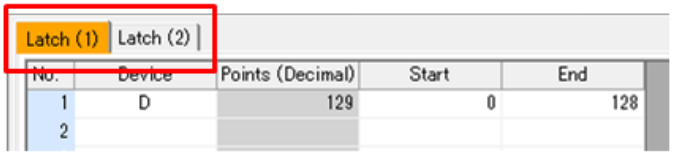
  Podtrzymywane dane można usunąć z użyciem CPU memory operation function * w GX Works3.
  Użyj podtrzymywania (latch) typu 1 w przypadku, gdy podtrzymywane dane mogą być usunięte w docelowej aplikacji.

- Latch (Podtrzymywanie typu) (2)

  Podtrzymywane dane mogą być usunięte za pomocą instrukcji programowej.
  Użyj podtrzymywania typu 2 w przypadku, gdy podtrzymywane dane nie mogą być usuwane w docelowej aplikacji

Poniżej podano przebiegi czasowe kasowania pamięci podtrzymywanej (latch)

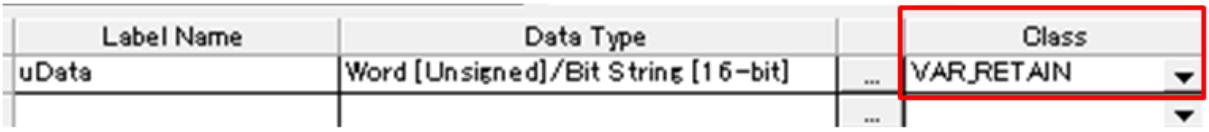




* Wykonaj funkcję przez wybór opcji [Online] → [CPU Memory Operation] w menu GX Works3.

2.8.3 Ustawianie funkcji podtrzymywania (latch) na etykietach (Labels)

W celu ustawienia podtrzymywania na etykiecie (Label) wybierz nazwę klasy zawierającą „RETAIN" w oknie konfiguracji etykiet (Labels)
Dla etykiet lokalnych (Local labels) wybierz „VAR_RETAIN".



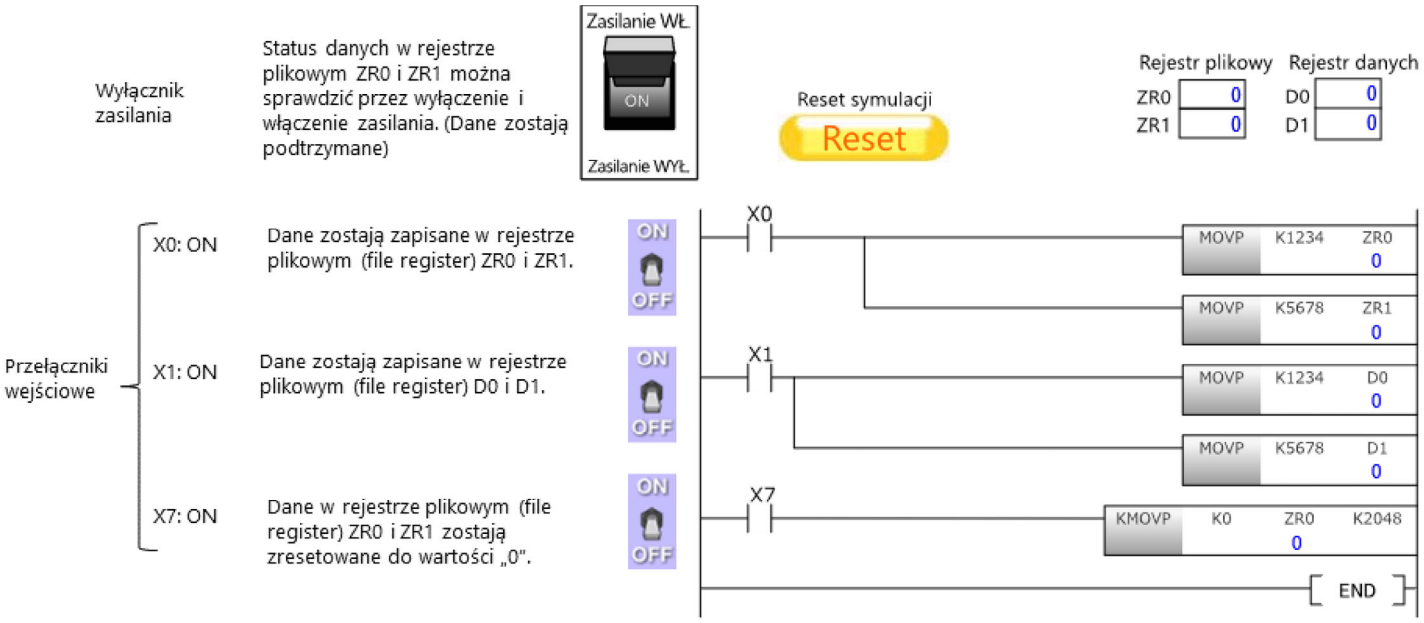
Label Name	Data Type		Class
uData	Word [Unsigned]/Bit String [16-bit]	...	VAR_RETAIN
		...	

2.9 Zachowanie statusu bitów/rejestrów (file register)

- Rejestr plikowy (file register) jest pamięcią podzieloną na słowa danych używaną do rozszerzenia rejestrów danych (D).
- W porównaniu z rejestrem danych rejestr plikowy (file register) może obsługiwać większą ilość danych.
- Rejestr plikowy (file register) jest zapisywany w pamięci danych procesora CPU lub na karcie pamięci.
- Dane zawarte w rejestrze plikowym (file register) są zachowane nawet po wyłączeniu zasilania lub po zresetowaniu procesora CPU
  Rejestr plikowy (file register) jest używany do zapisywania wartości predefiniowanych
- W celu skasowania danych wpisz 0 do rejestrów plikowych (file register) lub wyczyść pamięć za pomocą GX Works3.
- Symbolem rejestru plikowego (file register) jest „ZR".

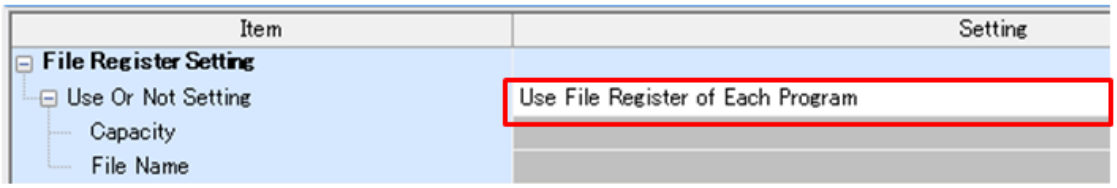
2.9.1 Działanie programu drabinkowego

W celu przeprowadzenia symulacji działania rejestru plikowego (file register), należy wyłączyć i załączyć zasilanie procesora CPU

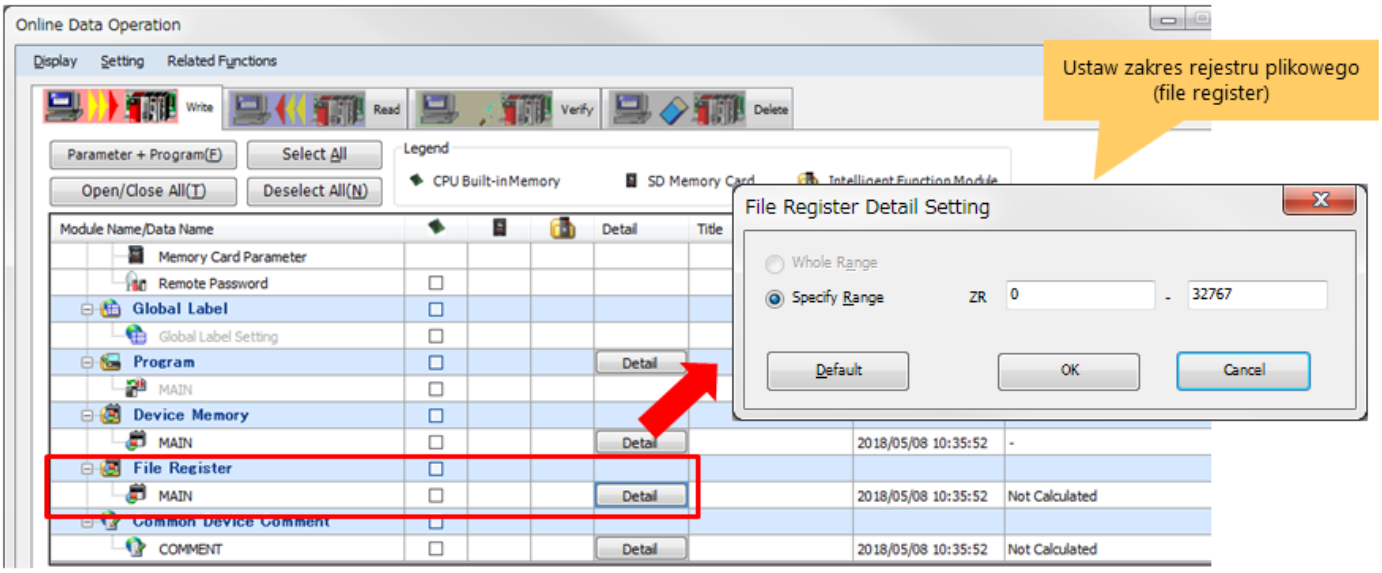


2.9.2 Konfiguracja rejestru plikowego (file register)

W tym punkcie opisano ustawienia, które definiują lokalny rejestr plikowy (file register) jako obszar pamięci dla każdego programu. Wybierz „File Register Setting" w parametrach procesora CPU, a następnie opcję „Use File Register of Each Program"



W celu zapisu danych w sterowniku ustawienie rejestru plikowego (file register) musi być ustawione dla każdego programu.



2.10 Używanie urządzeń z predefiniowanymi funkcjami i operacjami

Bity specjalne i rejestry specjalne używane w module procesora wykonują predefiniowane funkcje i operacje.
Bit specjalny (SM) jest wewnętrznym bitem używanym do informacji bitowych (wł./wył.), natomiast specjalny rejestr (SD) jest wewnętrznym rejestrem używanym do informacji słownych.

2.10.1 Typy specjalnych bitów (special relay) i rejestrów (special register)

Bity specjalne(special relay) i rejestry specjalne (special register)dzielą się ze względu na typ informacji. Najważniejsze typy są wymienione poniżej.
W programach bity i rejestry specjalne (special relays and registers) są używane jako warunki dla procesu sterowania.
Są one również używane do monitoringu działania, który może być wykonany z poziomu programu GX Works3.

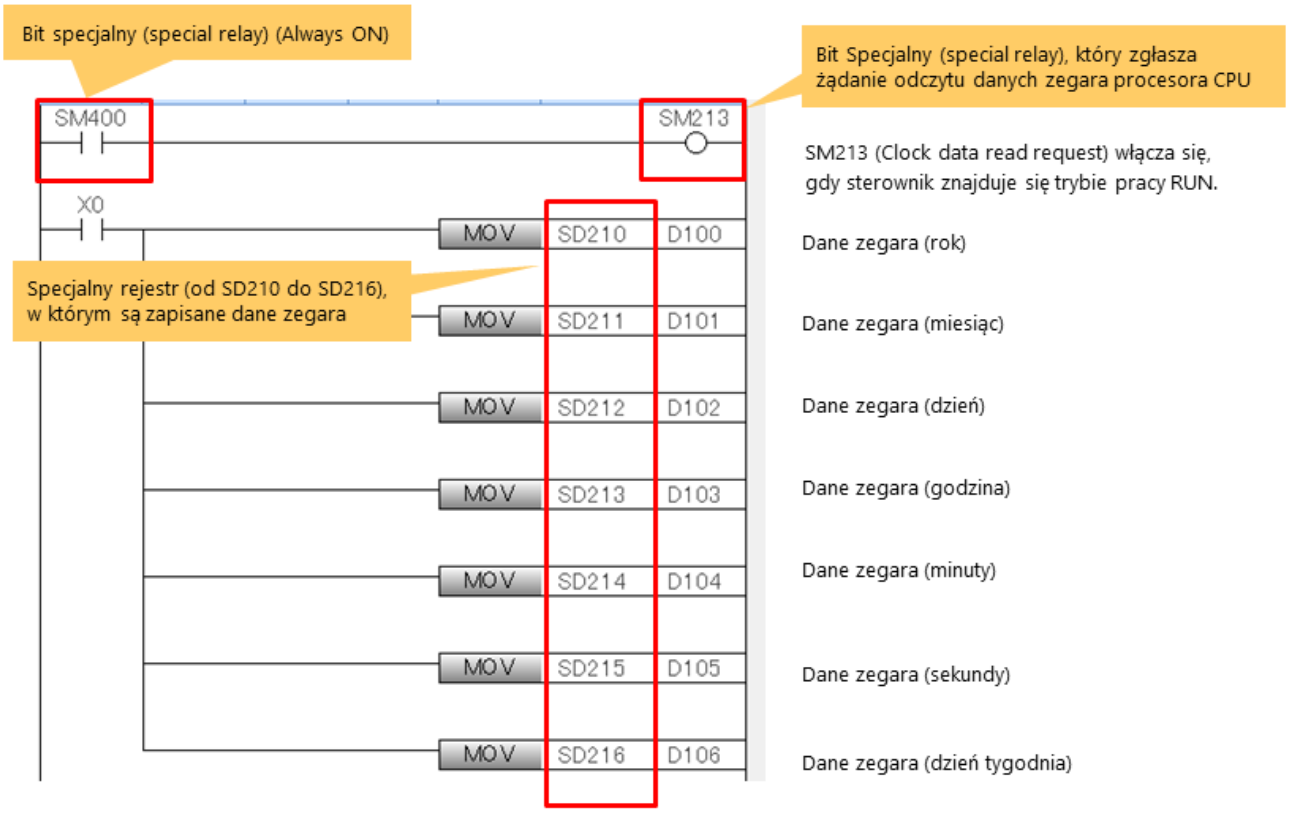
Informacje diagnostyczne
Wyniki diagnostyki procesora CPU
Błędy diagnostyczne i ich kody

Informacje systemowe
Informacje systemowe procesora CPU
Status działania procesora CPU, dane zegara i inne informacje

Zegar systemowy
Sygnały zegara i wskazania liczników używane do zliczania czasu
Różne sygnały zegara (zawsze ON, przełączanie ON/OFF z zadaną częstotliwością, inne sygnały)

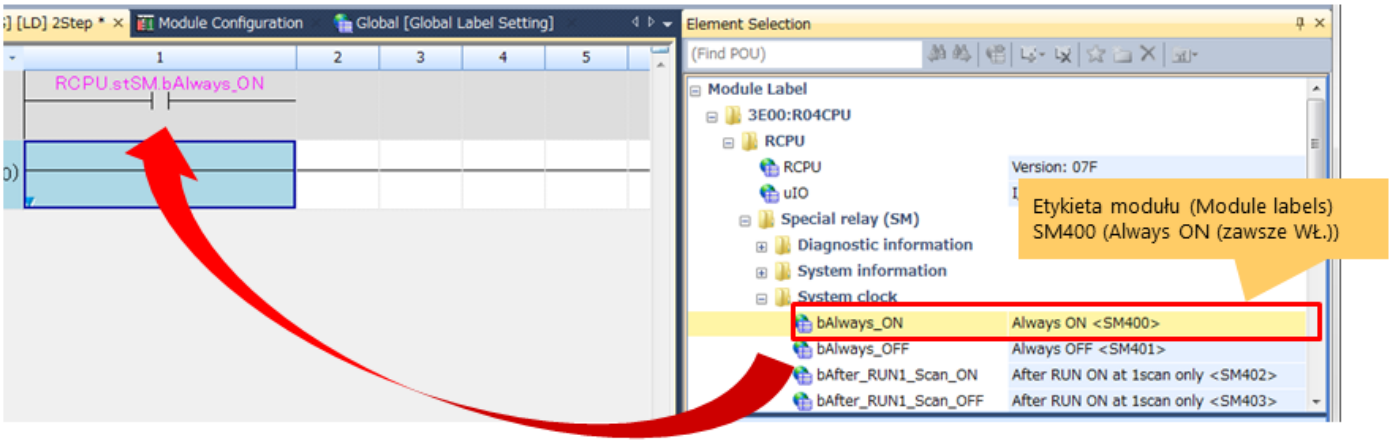
2.10.2 Przykładowy program z użyciem bitu i rejestru specjalnego (special relay and register)

Poniżej podano przykładowy program do odczytu danych zegara procesora CPU wykorzystujący bit i rejestr specjalny (special relay and register).



2.10.3 Używanie etykiet (Labels) dla bitu i rejestru specjalnego (special relay and register)

Bity i rejestry specjalne (special relay and register) są przygotowane w postaci etykiet modułów (Module labels) w procesora CPU. Można ich używać przez wybranie i umieszczenie odpowiednich nazw etykiet (Labels) bez sprawdzania adresów w instrukcji.



2.11 Obliczenia na liczbach rzeczywistych

2.11.1 Zastosowanie liczb rzeczywistych

- „Liczby rzeczywiste" to wartości numeryczne z separatorem dziesiętnym
- W programach sterujących zazwyczaj są używane liczby całkowite. Niemniej jednak liczby rzeczywiste są używane w programach sterowania zaawansowanymi działaniami (takimi jak funkcje trygonometryczne i potęgowe), kiedy programy sterowania muszą obsługiwać wartości numeryczne z separatorem dziesiętnym.
- Dane numeryczne będące liczbami rzeczywistymi obsługiwane przez procesor CPU są zwane „danymi zmiennoprzecinkowymi".

Uwaga:

- Jedna liczba rzeczywista zawsze używa dwóch kolejnych obszarów pamięci word (32-bitowa przestrzeń pamięci) niezależnie od jej wartości*

- W programach sterowania są przygotowywane dedykowane instrukcje (takie jak dodawanie, odejmowanie, mnożenie, dzielenie oraz funkcje specjalne) na liczbach rzeczywistych. Przygotowywane są również instrukcje konwersji liczb całkowitych na rzeczywiste.

* Jeśli konieczne są dokładniejsze obliczenia z większą liczbą cyfr znaczących, użyj czterech obszarów pamięci word.

- Liczby rzeczywiste wykorzystują dwa obszary pamięci word są zwane „liczbami rzeczywistymi z pojedynczą precyzją",
- natomiast wykorzystujące cztery pamięci podzielone na słowa danych – „liczbami rzeczywistymi z podwójną precyzją".

Podczas tego kursu koncentrujemy się na liczbach rzeczywistych z pojedynczą precyzją.

2.11.2 Notacja dla liczb rzeczywistych

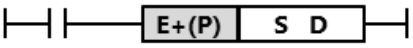
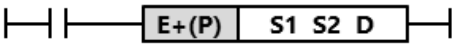
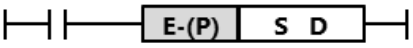
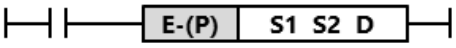
„E" jest używane do przedstawiania liczby rzeczywistej.

Wyrażanie stałych za pomocą liczb rzeczywistych

Stałą zapisuje się, zaczynając od „E".

Wyrażenie normalne	Zapisz wartość numeryczną taką, jaka jest. (Przykład) Zapisz 10,2345 jako „E10.2345".
Wyrażenie potęgowe	Zapisz wartość numeryczną jako „(wartość numeryczna) × 10^{n}". (Przykład) Zapisz 1234,0 jako „E1.234+3".

2.11.3 Instrukcje wykonania działań (dodawania i odejmowania)

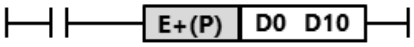
Znak wprowadzający	Przykład drabinki	
E+ (dodawanie liczb rzeczywistych z pojedynczą precyzją)	 Wykonywane jest działanie na liczbach rzeczywistych „D + S = D".	 Wykonywane jest działanie na liczbach rzeczywistych „S1 + S2 = D".
E- (odejmowanie liczb rzeczywistych z pojedynczą precyzją)	 Wykonywane jest działanie na liczbach rzeczywistych „D – S = D".	 Wykonywane jest działanie na liczbach rzeczywistych „S1 – S2 = D".

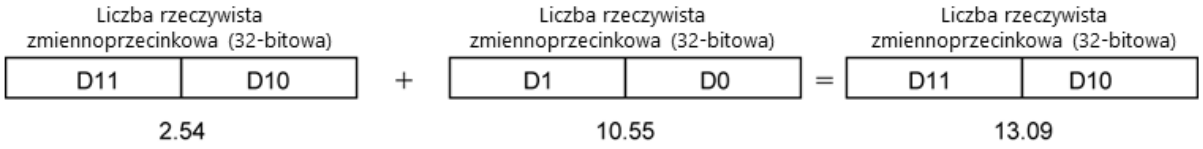
S (źródło): Dane przed wykonaniem działania (stała, numer Device)
D (miejsce docelowe zapisu): Miejsce docelowe zapisu danych po wykonaniu działania (numer Device)
P: Instrukcja do wykonania w przypadku zbocza narastającego (przy przejściu ze stanu OFF na ON)
S1 i S2: Dwie dane, na których wykonywane będzie działanie.

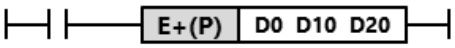
Uwaga:
W tym działaniu nie można używać jednocześnie liczb całkowitych i rzeczywistych.
W przypadku działania na liczbach rzeczywistych o pojedynczej precyzji S, S1 i S2 muszą być liczbami rzeczywistymi o pojedynczej precyzji.
Takie liczby są zapisywane w D.

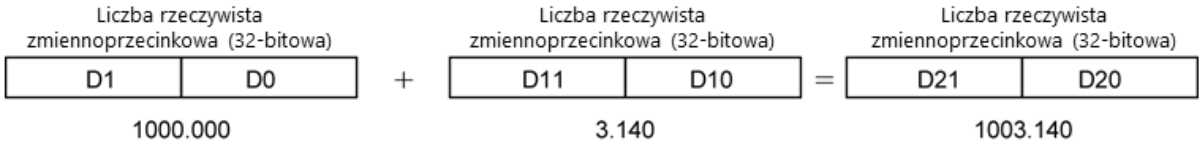
2.11.3 Instrukcje wykonania działań (dodawania i odejmowania)

Przykład zastosowania instrukcji dodawania



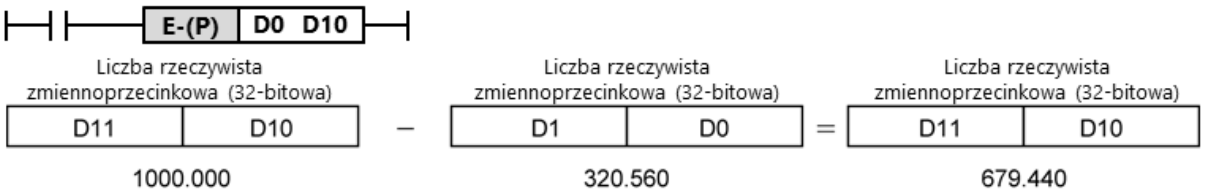


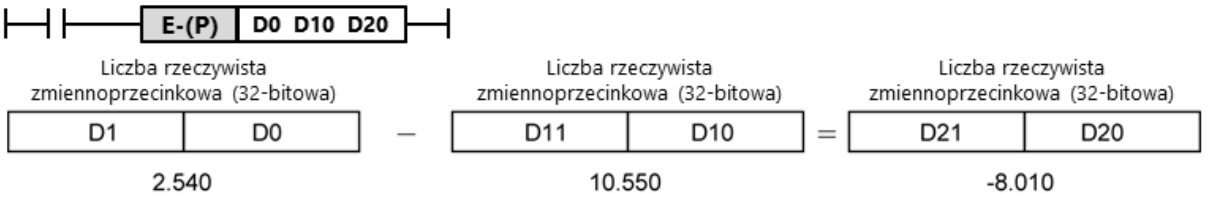




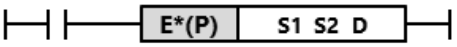
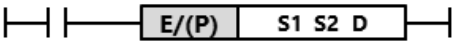
2.11.3 Instrukcje wykonania działań (dodawania i odejmowania)

Przykład zastosowania instrukcji odejmowania





2.11.4 Instrukcje wykonania działań (mnożenia i dzielenia)

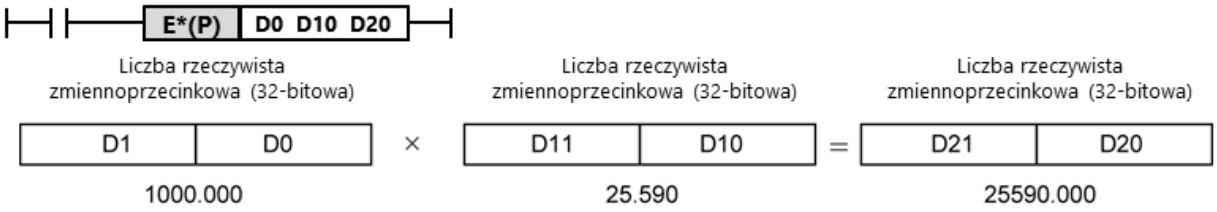
Znak wprowadzający	Przykład drabinki
E* (mnożenie liczb rzeczywistych z pojedynczą precyzją)	 Wykonywane jest działanie na liczbach rzeczywistych „S1 * S2 = D".
E/ (dzielenie liczb rzeczywistych z pojedynczą precyzją)	 Wykonywane jest działanie na liczbach rzeczywistych „S1 / S2 = D".

S1, S2 (źródło): Dwie dane, na których wykonywane będzie działanie.
D (miejsce docelowe): Miejsce przeznaczenia danych po wykonaniu działania (numer urządzenia)
P: Instrukcja do wykonania w przypadku zbocza narastającego (przy przejściu ze stanu OFF na ON)

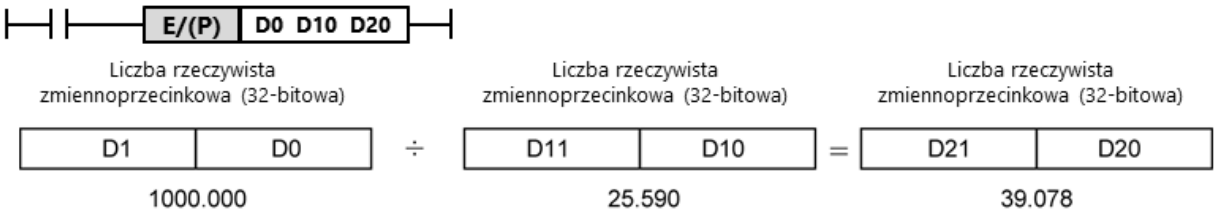
Uwaga:
W przypadku działań na liczbach rzeczywistych o pojedynczej precyzji S1 i S2 muszą być liczbami rzeczywistymi o pojedynczej precyzji.
Takie liczby są zapisywane w D.

2.11.4 Instrukcje wykonania działań (mnożenia i dzielenia)

Przykład zastosowania instrukcji mnożenia



Przykład zastosowania instrukcji dzielenia



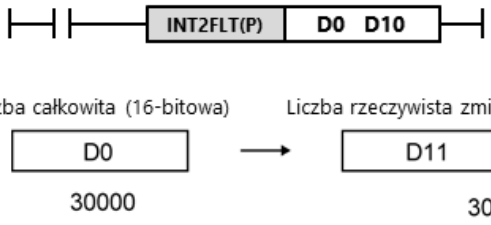
2.11.5 Instrukcje konwersji liczb całkowitych na liczby rzeczywiste

Znak wprowadzający	Przykład drabinki	
INT2FLT (konwersja liczby całkowitej na rzeczywistą o pojedynczej precyzji)	Liczba całkowita (16-bitowa) jest przekształcana na liczbę rzeczywistą (32-bitową).  S (16 bitów) jest przekształcane i zapisywane w D.	Liczba całkowita (32-bitowa) jest przekształcana na liczbę rzeczywistą (32-bitową).  S (32 bity) jest przekształcane i zapisywane w D.
FLT2INT (konwersja liczby rzeczywistej o pojedynczej precyzji na liczbę całkowitą)	Liczba rzeczywista (32-bitowa) jest przekształcana na całkowitą (16-bitową).  S jest przekształcane i zapisywane w D (16 bitów).	Liczba rzeczywista (32-bitowa) jest przekształcana na całkowitą (32-bitową).  S jest przekształcane i zapisywane w D (32 bity).

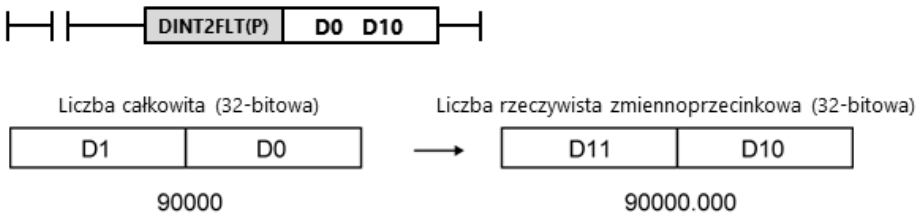
S (źródło): Dane przed wykonaniem działania (stała, numer Device)
D (miejsce docelowe): Miejsce docelowe danych po wykonaniu działania (numer Device)

2.11.5 Instrukcje konwersji liczb całkowitych na liczby rzeczywiste

Przykład zastosowania instrukcji konwersji liczby całkowitej (16-bitowej) na rzeczywistą (32-bitową)

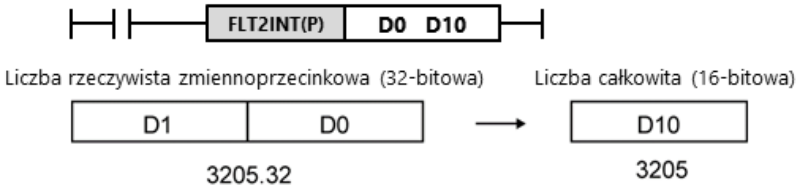


Przykład zastosowania instrukcji konwersji liczby całkowitej (32-bitowej) na rzeczywistą (32-bitową)

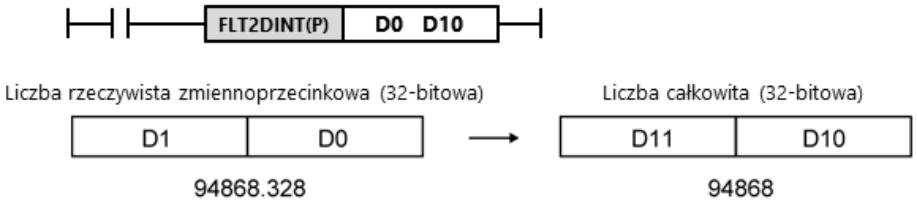


2.11.5 Instrukcje konwersji liczb całkowitych na liczby rzeczywiste

Przykład zastosowania instrukcji konwersji liczby rzeczywistej (32-bitowej) na liczbę całkowitą (16-bitową)



Przykład zastosowania instrukcji konwersji liczby rzeczywistej (32-bitowej) na liczbę całkowitą (32-bitową)



2.11.6 Użycie etykiet (Labels) reprezentujących liczby rzeczywiste

W celu użycia etykiet (Labels) dla liczb rzeczywistych ustaw typ danych na „Single Precision" (pojedyncza precyzja) lub „Double Precision" (podwójna precyzja) w oknie konfiguracji etykiet (Labels).

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

2.12 Podsumowanie treści tego rozdziału

W tej części kursu udało Ci się przyswoić następujące zagadnienia:

- Specyfikacja bitów obszaru pamięci word
- Specyfikacja zbocza narastającego i opadającego dla styków
- Timer z podtrzymaniem (retentive timer)
- Specyfikacja jednostek pomiarowych dla timera
- Rejestr indeksowy (Index register)
- Tablica (array)
- Struktura (Structure)
- Podtrzymywanie (Latch)
- Rejestr plikowy (File register)
- Specjalny przekaźnik i specjalny rejestr (Special relay, special register)
- Obliczenia z użyciem liczb rzeczywistych

Istotne punkty

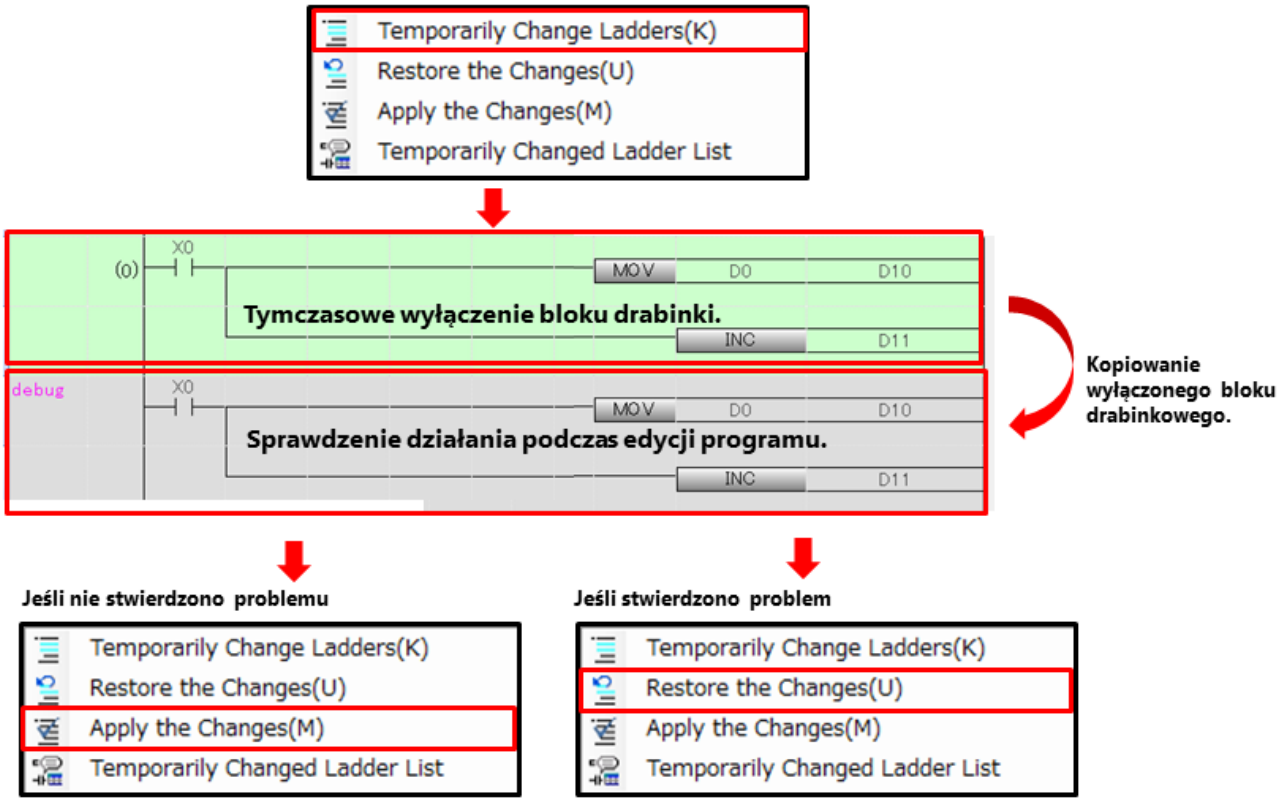
Timer z podtrzymaniem (retentive timer)	Zmierzony czas zostaje zachowany nawet przy wyłączeniu cewki; pomiar zostaje wznowiony po ponownym załączeniu cewki.
Jednostka pomiaru timera	Jednostkę pomiaru timera można zmienić w parametrze.
Rejestr indeksowy (Index register)	Możliwe jest stworzenie grup zmiennych zawierających wiele bitów/rejestrów
Tablica (Array)	Umożliwia obsługę wielu wartości za pomocą jednej nazwy etykiety (Label)
Struktura (Structure)	Umożliwia obsługę wielu powiązanych ze sobą etykiet (Labels) za pomocą jednej nazwy etykiety (Label).
Podtrzymywanie (latch)	• Podtrzymywane (latched) wartości są zachowane w przypadku zatrzymania pracy procesora CPU. • Podtrzymywane wartości można usuwać przez operację czyszczenia pamięci procesora CPU lub za pomocą instrukcji programowej.
Rejestr plikowy (File register)	• W porównaniu z rejestrem danych rejestr plikowy (File register) może obsługiwać większą ilość danych. • Wartości zasobów są zachowane w przypadku zatrzymania pracy procesora CPU • Wartości można usuwać przez operację czyszczenia pamięci procesora CPU lub wpisanie 0.
Bity specjalne i rejestry specjalne (Special relay, special register)	Wewnętrzny status procesora CPU (w tym informacje diagnostyczne i systemowe) został już zapisany w tych obszarach
Liczba rzeczywista	• Wykorzystuje co najmniej dwa obszary pamięci word (32 bity). • Dostępne są dedykowane instrukcje wykonania działania. • W tym działaniu nie można używać jednocześnie liczb całkowitych i rzeczywistych.

Rozdział 3 Skuteczne debugowanie

W tym rozdziale opisano funkcje wydajnego debugowania w GX Works3.

3.1 Tymczasowa zmiana zakresu programu

Podczas edycji dużego zakresu programu do debugowania, bardzo trudno jest cofnąć wszystkie zmiany.
Przez tymczasowe wyłączenie wybranego bloku drabinkowego, użytkownik może wykonać kopię programu do debugowania bez zmiany programu oryginalnego.
Tymczasowa zmiana funkcji drabinki - Temporarily change ladders function
Po zmianie bloku drabinkowego i sprawdzeniu działania za pomocą tej funkcji zmiany zostają zastosowane, jeśli nie ma problemów, lub cofnięte w razie wystąpienia problemu.

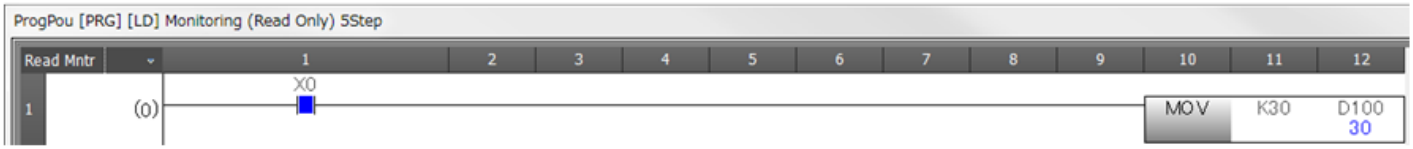


Jeśli nie stwierdzono problemu

Jeśli stwierdzono problem

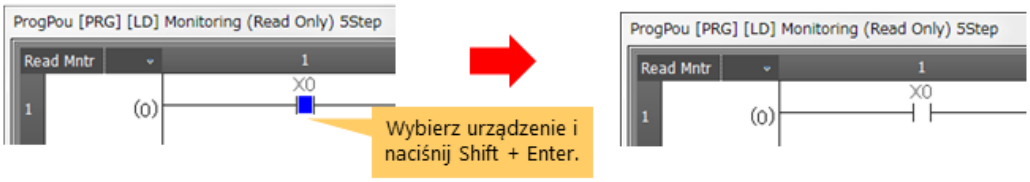
3.2 Sprawdzanie działania przy zmianie wartości

Podczas wykonywania utworzonego programu status ON/OFF bitów i wartości zapisane obszarach pamięci word mogą być wyświetlone w edytorze programu. (Monitor function (Funkcja monitoringu))
Za pomocą funkcji monitoringu użytkownicy mogą łatwo sprawdzić status działania programu.

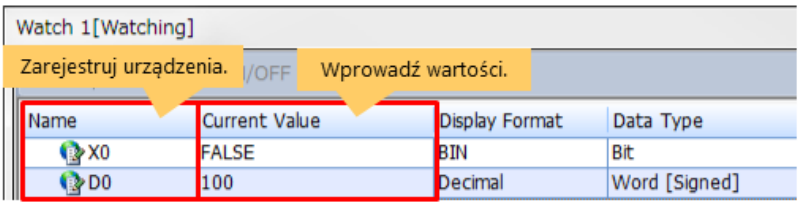


Aktualne wartości obszarów device mogą być wymuszone z poziomu monitoringu (Zmiana aktualnej wartości)
Za pomocą funkcji zmiany aktualnej wartości można wprowadzać zmiany bez konieczności edycji całego programu ani uruchamiania go w rzeczywistym systemie.

Status bitów można zmieniać w edytorze programu.

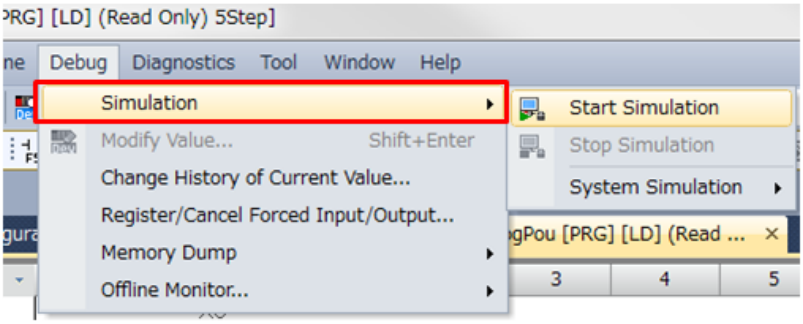


Za pomocą okna „Watch" można monitorować obszarach pamięci word i zmieniać ich aktualne wartości.



3.3 Symulacja działania programu

Podczas pracy nowo utworzonego programu w rzeczywistym systemie może wystąpić nieoczekiwany błąd. Symulację działania programu można przeprowadzić bez użycia rzeczywistego sterownika PLC. (Simulation function (Funkcja symulacji))



Za pomocą funkcji symulacji (Simulation function) można sprawdzać działanie programu w takich warunkach, jakby pracował on na rzeczywistym sterowniku PLC

3.4 Podsumowanie treści tego rozdziału

W tej części kursu udało Ci się przyswoić następujące zagadnienia:

- Wprowadzanie tymczasowych zmian bloku drabinkowego
- Prowadzenie monitoringu programu i zmiany aktualnej wartości
- Symulacja programu

Istotne punkty

Wprowadzanie tymczasowych zmian bloku drabinkowego	Blok drabinkowy można czasowo wyłączyć i użyć kopii programu do debugowania bez konieczności zmiany oryginalnego programu.
Monitoring	• Realizacja programu może zostać zwizualizowana. • Działanie programu można sprawdzać przez wymuszenie zmian aktualnych wartości obszarów Device
Symulacja	Symulację działania programu można przeprowadzić bez użycia rzeczywistego programowalnego sterownika.

3.5 Podsumowanie kursu

To już koniec tego szkolenia e-learningowego.

Poniżej znajduje się podsumowanie kursu.

- **Wydajne programowanie**
  - Przypisywanie stałych adresów we/wy (I/O numbers) modułom w celu łatwego zastosowania programów w różnych systemach
  - Ustawianie pamięci w zależności do wykorzystywanych bitów i rejestrów
  - Używanie etykiet (Labels) w celu łatwiejszego programowania i lepszego zrozumienia działania programu
  - Dodawanie komentarzy w celu poprawy czytelności programu
- **Zaawansowane programowanie**
  - Używanie timera z podtrzymaniem (retentive timer) w celu podtrzymania wartości zmierzonego czasu
  - Używanie rejestru indeksowego (index register), tablic lub struktur w celu zbiorowego przetwarzania wartości
  - Używanie funkcji podtrzymywania (latch) i rejestru plikowego (file register) do zachowania statusu
  - Dostępne są bity i rejestry specjalne (special relay and register) które przechowują status procesora CPU
  - Liczba rzeczywista jest reprezentowana przez dwa lub cztery elementy obszaru pamięci word. W tym działaniu nie można używać jednocześnie liczb całkowitych i rzeczywistych.
- **Wydajne debugowanie**

  Za pomocą GX Works3 użytkownik może:
  - debugować program bez konieczności jego zmiany
  - wizualizować pracę programu
  - symulować działanie programu

Aby przejść do kolejnego etapu, ukończ następujące kursy dotyczące „ustrukturyzowania", w wyniku którego program zostanie podzielony na warstwy i komponenty, tak aby można ich było łatwo użyć ponownie.

- Wydajne programowanie
- Wydajne programowanie (praktyka) (zostanie opublikowane w późniejszym terminie)

Test końcowy

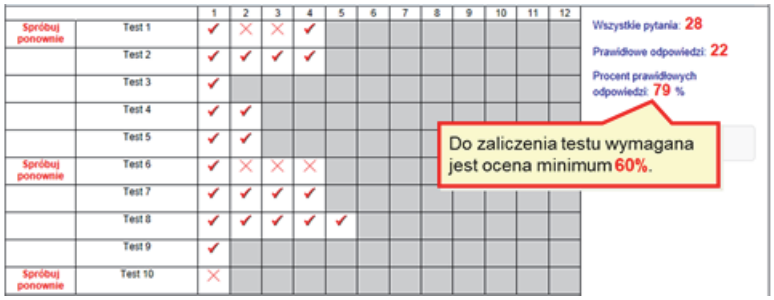
Teraz, po zakończeniu wszystkich etapów kursu **Programowanie PLC (Język drabinkowy/seria MELSEC iQ-R)**, możesz przystąpić do testu końcowego. W razie niejasności w zakresie któregokolwiek z tematów, wykorzystaj tę możliwość do ponownego zapoznania się z tymi zagadnieniami.

Test końcowy składa się z 14 pytań (35 elementów).

Możesz do niego podchodzić dowolną ilość razy.

Punktacja końcowa

Liczba prawidłowych odpowiedzi, liczba pytań, procent prawidłowych odpowiedzi i wynik zaliczony/niezaliczony pojawią się na stronie wyniku.



Test **Test końcowy 1**

Które z poniższych sformułowań dotyczących przypisywania adresów we/wy modułów jest prawdziwe?

Q1

- **Adresy we/wy mogą być przypisane ręcznie do każdego modułu, tak aby program nie musiał być zmieniany przy zmianie konfiguracji sprzętowej**
- **Automatycznie przypisane adresów we/wy nie mogą być zmienione**

Test **Test końcowy 2**

Które z poniższych sformułowań dotyczących przypisywania punktów do urządzeń jest prawdziwe?

Q1

- **Dla każdego urządzenia (nawet jeśli jest nieużywane) musi być przypisany przynajmniej jeden punkt**
- **Punkty można przypisać do urządzeń zgodnie z łączną zastosowaną liczbą punktów**

Test **Test końcowy 3**

Które z poniższych sformułowań dotyczących etykiet jest prawdziwe? (możliwość wielokrotnego wyboru)

Q1

- [ ] **Użycie etykiet pomaga w identyfikacji celu i ułatwia programowanie**
- [ ] **Dostępne są etykiety reprezentujące sygnały modułów i ustawień**
- [ ] **Możliwe jest dodawanie komentarzy do elementów w celu poprawy czytelności programu**
- [ ] **Ponieważ stałe można przypisać do etykiet, ich wartości mogą być zmieniane bez konieczności modyfikacji programu**

Test **Test końcowy 4**

Uzupełnij poniższy tekst opisujący timer retencyjny.

Timer retencyjny uruchamia pomiar, gdy załączony ony zostaje **(Q1)** (cewka przyjmuje stan **(Q2)**).
Timer retencyjny zachowuje zmierzony czas nawet po wejściu sygnału wejściowego w stan **(Q3)** i kontynuuje pomiar od tak zachowanej wartości po ponownym przejściu warunku wejściowego w stan **(Q4)** .
Timer retencyjny wyłącza się po upływie zadanego czasu pomiaru, po czym włącza się **(Q5)** .

Q1 -- Select --

Q2 -- Select --

Q3 -- Select --

Q4 -- Select --

Q5 -- Select --

Q6 -- Select --

Test **Test końcowy 5**

Uzupełnij program sterujący, który wykona poniższy proces przetwarzania.

- Użyj timera retencyjnego (ST0) do pomiaru czasu przebywania sygnału wejściowego X0 lub X1 w stanie ON
- Gdy czas ten wyniesie 30 s, włącz cewkę Y70 i wskaźnik czasu zwłoki
- Po włączeniu X2 wyłącz styk timera retencyjnego (ST0) i skasuj czas pomiaru (aktualną wartość)

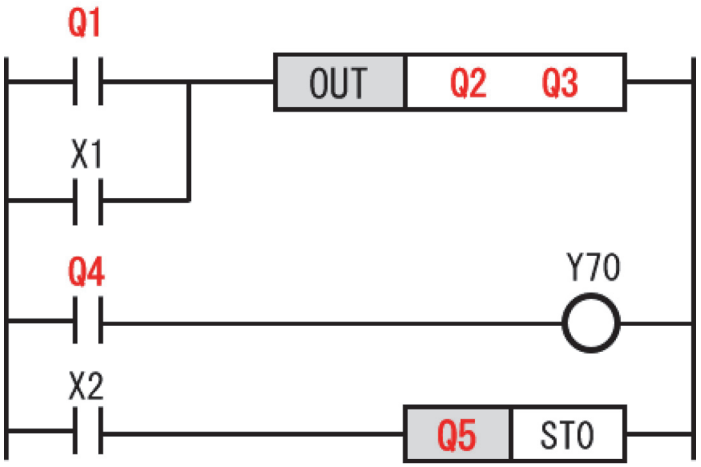
Q1 -- Select --

Q2 -- Select --

Q3 -- Select --

Q4 -- Select --

Q5 -- Select --



[+]

Test **Test końcowy 6**

Wybierz wartość, jaka zostanie zapisana do rejestu danych D20, gdy X0 zostanie włączony po zajściu każdego z następujących warunków w podanym poniżej programie sterującym.

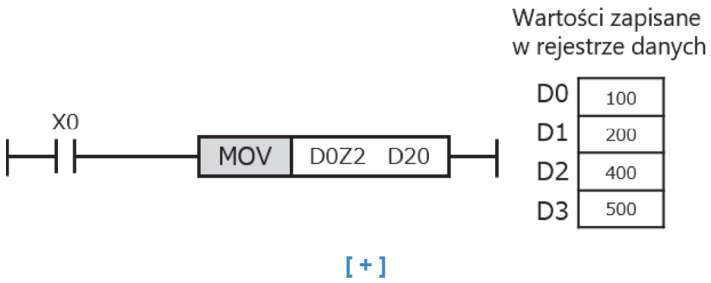
Q1) Jeśli wartość zapisana w Z2 wynosi „0”
Q2) Jeśli wartość zapisana w Z2 wynosi „1”
Q3) Jeśli wartość zapisana w Z2 wynosi „2”

Q1 -- Select --

Q2 -- Select --

Q3 -- Select --

Q4 -- Select --



Wartości zapisane w rejestrze danych

D0	100
D1	200
D2	400
D3	500

[+]

Test **Test końcowy 7**

Które z poniższych sformułowań dotyczących sposobu definiowania elementu w tablicy danych jest prawdziwe?

Q1

- **Podaj numer elementu na końcu nazwy etykiety**
- **Podaj pośrednio numer urządzenia**

Test **Test końcowy 8**

Które z poniższych sformułowań dotyczących struktur (programowych) jest fałszywe?

Q1

- **Struktury są używane do zbiorczej organizacji i zapamiętywania warunków i specyfikacji związanych z obiektami fizycznymi i materiałami**
- **Za pomocą struktur można w przejrzysty sposób opisywać przetwarzanie dużych ilości danych**
- **Elementy zdefiniowane w strukturze muszą posiadać ten sam typ danych**

Test **Test końcowy 9**

Które z poniższych sformułowań dotyczących zatrzasków (latch) są poprawne?

Q1

- **Podtrzymanie wartości urządzeń jest ustawieniem domyślnym**
- **W celu podtrzymania wartości konieczne jest ustawienie parametrów za pomocą oprogramowania**

Test **Test końcowy 10**

Uzupełnij poniższy tekst opisujący rejestr plikowy.

Rejestr plikowy jest rejestrem typu word używanym do rozszerzenia rejestru danych (D) i ma symbol **(Q1)** .
Rejestr plikowy ma pojemność **(Q2)** niż rejestr danych i zapamiętane dane zostają **(Q3)** w razie wyłączenia zasilania systemu lub zresetowania modułu procesora.
W celu użycia rejestru plikowego **(Q4)** ustawienie parametrów za pomocą oprogramowania projektowego.

Q1 -- Select --

Q2 -- Select --

Q3 -- Select --

Q4 -- Select --

Test **Test końcowy 11**

Które z poniższych sformułowań na temat przekaźnika specjalnego i rejestru specjalnego jest prawdziwe?

Q1

- **Wewnętrzny status modułu procesora został już zapisany w specjalnym przekaźniku i rejestrze; urządzenia te są używane jako warunki w programie sterującym**
- **Do specjalnego przekaźnika i rejestru można swobodnie przypisać funkcje specjalne**

Test **Test końcowy 12**

Uzupełnij poniższy tekst opisujący liczby rzeczywiste (z pojedynczą precyzją).

• Jedna liczba rzeczywista używa **(Q1)** urządzeń typu word i jest zapamiętywana w **(Q2)** -bitowej przestrzeni pamięci.
• Wartości numeryczne liczb rzeczywistych są nazywane **(Q3)** . Przykład: wartość 2,035 jest zapisana jako **(Q4)** w programie sterującym.
• Liczby całkowite i rzeczywiste **(Q5)** być wspólnie używane w instrukcjach wykonywania działań na liczbach

Q1 -- Select --

Q2 -- Select --

Q3 -- Select --

Q4 -- Select --

Q5 -- Select --

Test **Test końcowy 13**

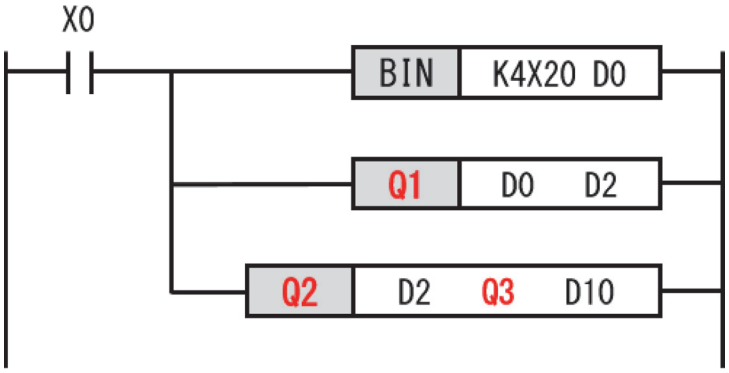
Napisz program sterujący, który wykona następujący proces przetwarzania.

- Po załączeniu styku X0 Odczytaj dane obszaru od X20 do X2F (w kodzie BCD) i zapisz je w D0.
- Wykonaj konwersję wartości w D0 na liczbę rzeczywistą i zapisz wartość po przekształceniu w D2.
- Pomnóż wartość w D2 przez 3,14 i zapisz wynik w D10.

Q1 -- Select --

Q2 -- Select --

Q3 -- Select --



[+]

Test **Test końcowy 14**

Które z poniższych sformułowań dotyczących debugowania programu sterującego jest prawdziwe?

Q1

- **Działanie programu można bezpiecznie symulować za pomocą funkcji oprogramowania projektowego.**
- **W celu debugowania programu należy go wykonać w rzeczywistym systemie.**

Test | **Wynik testu**

Test końcowy został zakończony. Twoje wyniki są przedstawione poniżej.
Aby zakończyć test końcowy, przejdź do następnej strony.

		1	2	3	4	5	6	7	8	9	10
	Test końcowy 1	✓									
	Test końcowy 2	✓									
	Test końcowy 3	✓									
	Test końcowy 4	✓	✓	✓	✓	✓	✓				
	Test końcowy 5	✓	✓	✓	✓	✓					
	Test końcowy 6	✓	✓	✓	✓						
	Test końcowy 7	✓									
	Test końcowy 8	✓									
	Test końcowy 9	✓									
	Test końcowy 10	✓	✓	✓	✓						
	Test końcowy 11	✓									
	Test końcowy 12	✓	✓	✓	✓	✓					
	Test końcowy 13	✓	✓	✓							
	Test końcowy 14	✓									

Wszystkie pytania: **35**

Prawidłowe odpowiedzi: **35**

Procent prawidłowych odpowiedzi: **100** %

Wyczyść

Udało Ci się ukończyć kurs Programowanie PLC (Język drabinkowy/seria MELSEC iQ-R).

Dziękujemy za wzięcie udziału w kursie.

Mamy nadzieję, że poruszone tematy były interesujące, a informacje uzyskane w trakcie tego kursu będą przydatne w przyszłości.

Możesz przeglądać kurs dowolną ilość razy.

Sprawdź

Zamknij