Podręczny Podręcznik Programowania PLC

4xP

STL

V2023-2

Do pobrania z

GrzegorzCzekala.pl

Lista najczęściej popełnianych błędów:

1. Wielokrotne użycie tej samej cewki (!!!),
2. Nie załączony / zasilony sterownik (lub w błędzie SF),
3. Nowa funkcja nie wywołana w OB1,
4. Dalej nie działa? Przejdź do aplikacji „**4xP App**".

Przydatne skróty klawiszowe: (część tylko TIA lub STEP7)

Ctrl+R – nowy network **Ctrl+L –** prześlij na CPU

Ctrl + Tab - przełączanie kart **Ctrl + I -** Zdalna stacyjka CPU

Ctrl + J - Wstaw symbol w STL **Ctrl + D -** Modul Information

Ctrl + Q - Wyświetl symbole / adresy **Ctrl + 1 / 2 / 3 -** zmiana LAD / STL / FBD

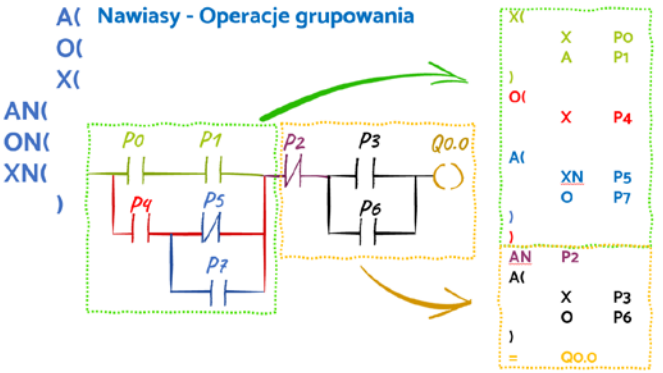
Domyślny Clock Memory (po aktywacji w HW Config)

M0.7	M0.6	**M0.5**	M0.4	M0.3	M0.2	M0.1	M0.0
0.5Hz	0.625Hz	**1Hz**	1.25Hz	2Hz	2.5Hz	5Hz	10Hz

Podstawowe instrukcje

=	Przypisanie stanu (cewka)
A	Iloczyn logiczny
O	Suma logiczna
X	XOR (nieparzystość)
...N	Negacja pojedynczego bitu (np. **AN** "ON", **XN** I1.1)

Grupowanie



...(Rozkaz otwierający grupowanie np.: A(, AN(, X(.

) Rozkaz zamykający grupowanie

Przerzutniki

R	Warunkowe kasowanie bitu
S	Warunkowe setowanie bitu

Operacje na RLO i wywołania funkcji

BEC Warunkowe zakończenie wykonywanego bloku

BE / BEU Bezwarunkowe zakończenie wykonywanego bloku

Operacje na RLO (bezwarunkowe)

NOT Negacja stanu RLO

SET Ustawienie stanu RLO

CLR Kasowanie stanu RLO

SAVE Zapisanie RLO w bicie BR

Wywołania funkcji

CC Warunkowe wywołanie funkcji

UC Bezwarunkowe wywołanie funkcji np. UC FC 1

CALL Bezwarunkowe wywołanie funkcji z parametrami

```
// Warunkowe
wywołanie Fc1
X „ON”
CC Fc1
```

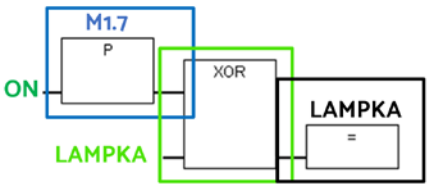
Zamieszczone grafiki są mojego autorsta - Grzegorza Czekały. Wykorzystuję je na swoich szkoleniach i w podręcznikach. W tym miejscu poprzez **Fundację CALM edu** udostepniam je Tobie za darmo ponieważ wierzę, że wartosciowe treści edukacyjne należy promować i rozpowszechniać, tak aby jak najwięcej osób mogło się dzieki nim rozwijać dla naszego wspólnego dobra. Też tak uważasz? Chcesz podzielić się swoimi materiałami? Chcesz się przyłączyć do Fundacji? Daj nam znać.

Chcesz poznać nowoczesne podejście do edukacji? Chcesz coś zmienić? Odwiedź stronę **Fundacji CALM edu** i odkryj nowy wymiar nauki. **„Edukacja 4.0” i „Szkoła 4.0”**.

Detekcja zbocza umożliwia zamianę sygnału trwającego wiele cyklu CPU na pojedyncze impulsy. **Zbocze narastające (FP)** to zmiana stanu niskiego na wysoki (zwykle wciśniecie przycisku).**zbocze opadające (FN)** to zmiana z stanu wysokiego na niski (puszczenie przycisku, zanik sygnału).

```
// Opadające
X „ON”
FN M6.1
S Q0.0
```

Przykład detekcji zbocza




```
X ON
FP M1.7
X Lampka
= Lampka
```

Naurodydaktyka - czyli jak się skutecznie uczuć

Chcesz nauczyć się jak się uczyć, aby się nauczyć? Poszukaj informacji na stronie fundacji na temat **neurodydaktyki** – dziedziny nauki badajacej m.in. w jaki sposób nasz mózg przyswaja nowe informacje i w jaki sposób tworzyć skuteczne materiały edukacyjne (takie jak 4xP ;) Chcesz poznać szkołę przyszłości? Znajdź „Szkoła 4.0” i „Edukacja 4.0”

Arytmetyka, komparatory i konwersja

TAK – zamienia miejscami zawartość ACCU 1 i ACCU2

PUSH – kopiuje z ACCU1 do ACCU2 (w dwóch będzie to samo)

POP – kopiuje z ACCU2 do ACCU1 (w dwóch będzie to samo)

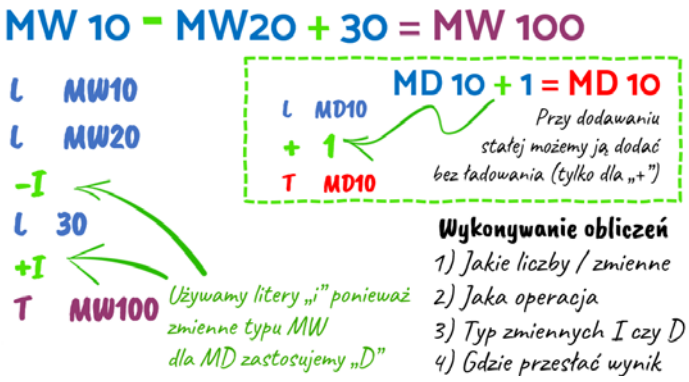
L ... – załadowanie do ACCU1 **T ...** –transfer (kopiowanie z ACCU1)

ITD – Integer to Double, **DTR** – Double to REAL,

RND, RND+, RND-, TRUNC – zaokrąglanie i ucinanie REAL

Znalazłeś błędy, literówki, masz pomysły jak coś zrobić lepiej? **Koniecznie** daj nam znać w wiadomości lub komentarzu.

Podoba się? Oceń post i udostępnij go swoim znajomym.



Arytmetyka

„+" „+I" „+D" „+R" (ADD)

„-I" „-D" „-R" (SUB)

„*I" „*D" „*R" (MUL)

„/I" „/D" „/R" (DIV)

„MOD" reszta z dziel

INC / DEC +1 / -1

Komparatory

==I / D / R (EQ)

<>I / D / R (NE)

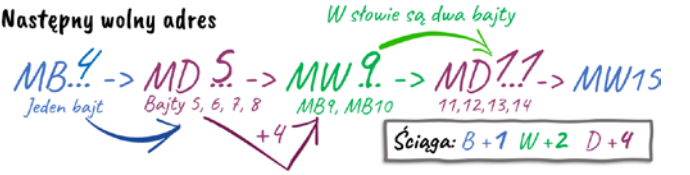
>I / D / R (GT)

<I / D / R (LE)

>=I / D / D (GE)

<=I / D / R (LE)

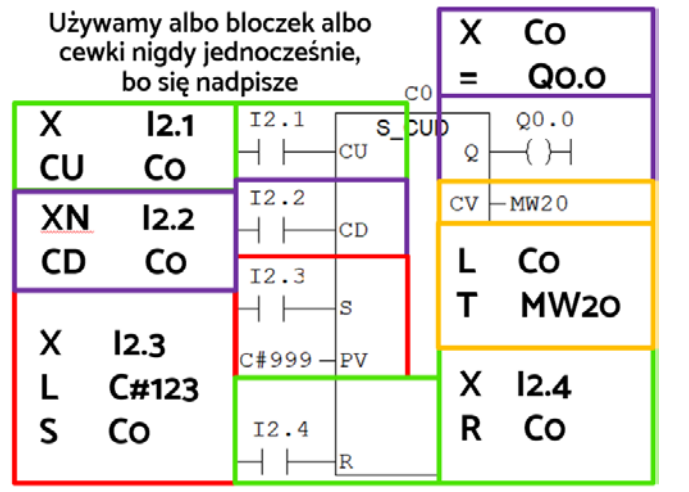
```
// Spr. większość
L   MD10
L   45
>D
= Lampka
```



Rotacje i przesunięcia

RLD / RRD – rotacja 32 bitów w L (lewo) lub R (prawo)
SLW / SLD – przesunięcie w lewo 16 / 32 bitów
SRW / SRD – przesunięcie w prawo 16 / 32 bitów

Liczniki w STL



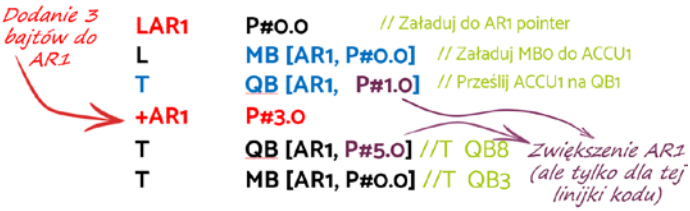
Wskaźniki 32 bitowe

LAR1 / LAR2 – kopiuje ACCU1 do AR1 / AR2

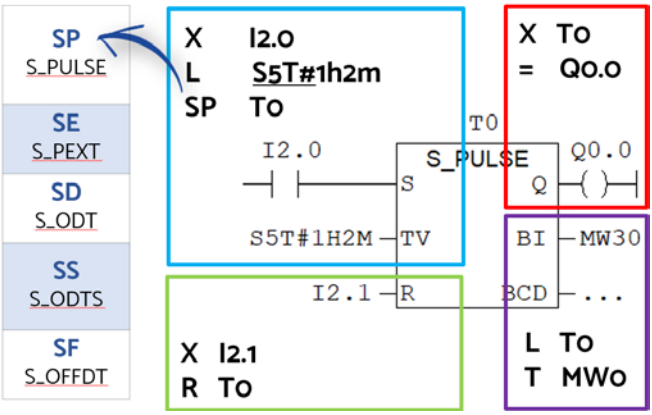
TAR1 / TAR2 - kopiuje AR1 do ACCU1

+AR1 / +AR2 - dodaje wartość ACCU1 do AR1 / AR2

Odwołanie do zmiennej o długości BYTE poprzez wskaźniki 32 bitowy



Timery w STL



Wszystkie countery oraz timery z powodu ograniczonego tutaj miejsca zostały zamieszczone na osobnej wersji 4xP „Countery i Timery w TIA Portal / Step7".

Bloki Danych w STL (dostęp bez optymalizacji)

OPN DB... / DI .... – otwarcie DB dla rejestrów DB1 / DB2

Odwołanie do zawartości DB o rozmiarach: (? – odpowiedni numer)

bit – DB?.DBX?.? np.: X DB10.DBX5.3 **BYTE**–DB?.DBB np.: L DB11.DBB8

WORD-DB?.DBW np.: L DB11.DBW2 **DWORD**–DB?.DBD (też dla REAL)

Przykady zastosowania:

```
OPN DB1 // otwarcie DB1
L  DBW2 // załaduj drugie słowo
+ 1        // dodaj 1
T  DBW4 // prześlij do word 4 DB2
```

Przekopiowanie zawartości

```
OPN  DB1  // otwórz DB1
OPN  DI2  // otwórz też DB2
L  DBW2  //pobież słowo 2 z DB1
T  DIW0 //prześlij na słowo 0 z
```

Struktura pętli LOOP

```
L 8   // załadowanie liczby pętli do wykonania
etyk: T #LicznikPętli // zmienna temp przechowująca liczbę
                      // pętli do wykonania
// Zawartość
// Zawartość
// Zawartość
L #LicznikPętli // Załadowanie liczby pętli do ACCU1
LOOP etyk
```

Rozkaz LOOP zmniejsza wartość ACCU1 o 1 i sprawdza czy jest teraz większa od zero, jeśli tak skacze do podanej etykiety jeśli nie program idzie dalej

Operacje logiczne na słowach

AW – AND na WORD, **OW** – OR na WORD, **XOW** – XOR na WORD

INVI – inwersja WORD, **INVD** – inwersja DWORD

AD – AND na DWORD, **OD** – OR na DWORD, **XOD** - XOR DWORD

SKOKI podstawowe

JU Skok bezwarunkowy **NOP 0** Pusty rozkaz **BE** - Zakończ blok

JC Skok RLO=1 **JCN** Skok RLO=0

Gdy wcisniety ON to ładuje 123 do Mw10, a gdy nieaktywny przeskakuje do et1.

```
X   „ON"
JCN  et1
L    123
T    MW10
et1: NOP 0
```

P1 przesyła wartość „0", a P2 stałą analogową na AQ. [nie jest to optymalne rozwiązanie ale działa i używa wiele skoków, dobre jako powtórka]

```
X   „P1"
JC  et_1
X   „P2"
JC  et_2
BE
et1: L  0
     JU end // !!!!
et2: L  27648
end: T  PQW800
```