```
||
|| ; --add comment here--
||
```

dodawanie komentarzy do programu

.....

```
||
||       Xnew
||-------] [-------
||
```

```
||
||       Xnew
||-------]/[-------
||
```

sygnał wejściowy, prosty lub zanegowany,

sygnalizuje stan sygnału na wejściu PLC (input pin), wyjściu (cewki, output pin) lub wewnętrznej komórki pamięci (InternalRelay)

.....

```
||
||----[OSF¯\_]-----
||
```

na wyjściu pojawia się napięcie w chwili pojawienia się opadającego zbocza sygnału

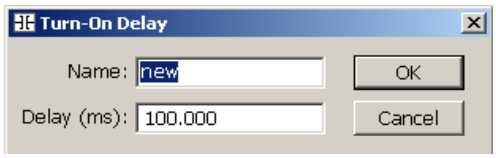
.....

```
||
||----[OSR_/¯]-----
||
```

na wyjściu pojawia się napięcie w chwili pojawienia się narastającego zbocza sygnału

.....

```
||
||       Tnew
||-[TON 100.0 ms]--
||
```



jeśli na wejściu pojawi się napięcie to po nastawionym czasie pojawi się na wyjściu (opóźnione załączenie)

.....

```
||
||       Tnew
||-[TOF 100.0 ms]--
||
```

jeśli z wejścia zniknie napięcie to po nastawionym czasie zniknie z wyjścia (opóźnione wyłączenie)

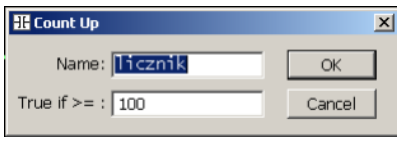
.....

```
||
||       Tnew
||-[RTO 100.0 ms]--
||
```

jeśli na wejściu przez czas nastawiony lub większy jest sygnał to przedostaje się na wyjście

.....

```
||
||      Cnew
||----[CTU >=0]----
||
```

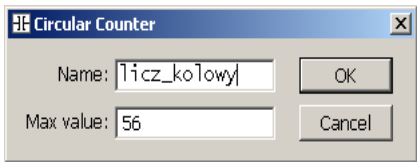


licznik zliczający w górę, przepuszcza sygnał jeśli zliczy nastawioną liczbę impulsów

```
||
||      Cnew
||----[CTD >=0]----
||
```

licznik zliczający w dół, przepuszcza sygnał jeśli pozostanie nie mniej niż nastawiona liczba impulsów

```
                ||
  Cnew          ||
{CTC 0:0}----||
                ||
```

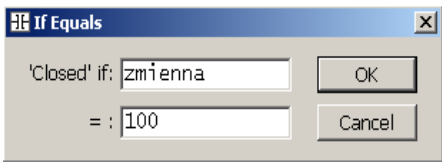


liczy do wartości nastawionej a później zeruje się i zlicza od początku

.....

```
||
||    [var ==]
||----[ 1    ]-----
||
```

sygnał przechodzi jeśli zmienna równa wartości nastawionej




```
||
||    [var /=]
||----[ 1    ]-----
||
```

sygnał przechodzi jeśli zmienna różna od wartości nastawionej

```
||
||     [var >]
||-----[ 1   ]-----
||
```

sygnał przechodzi jeśli zmienna większa od wartości

nastawionej

```
||     [var >=]
||-----[ 1     ]------
```

sygnał przechodzi jeśli zmienna większa lub równa wartości nastawionej

```
||      [var <]
||------[ 1   ]------
```

sygnał przechodzi jeśli zmienna mniejsza od wartości nastawionej

```
||
||    [var <=]
||----[ 1    ]-----
```

sygnał przechodzi jeśli zmienna mniejsza lub równa wartości nastawionej

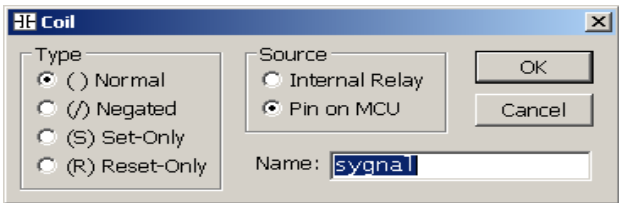
.....

```
-------{MASTER RLY}---||
                       ||
```

jeśli nie dociera do niego sygnał wszystkie następujące po nim szczeble drabiny są pomijane aż do kolejnej instrukcji tego typu

.....

```
     Ynew          ||
-----( )-------||
                   ||
```



sygnał wyjściowy, może być prosty lub zanegowany, może dotyczyć sygnału wyjściowego (pin on MCU) lub stanu wewnętrznej komórki pamięci (InternalRelay), może być typu *set only* (sygnał z wejścia może go ustawić w stan wysoki i już taki pozostanie) lub *reset only* (sygnał wejściowy go kasuje i taki już pozostaje)

.....

```
   Tnew          ||
---{RES}------||
                 ||
```

służy do resetowania liczników i czasomierzy (timerów) (oczywiście jeśli na wejściu pojawi się sygnał)

.....

```
 {dest   :=     }   ||
-{ src      MOV}--||
                    ||
```

służy do przemieszczania danych ze źródła (src) do przeznaczenia (dest)

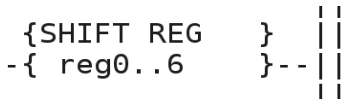
.....

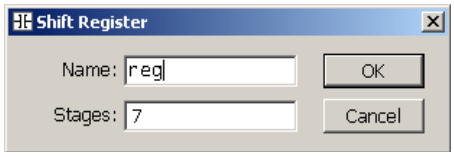
```
 {ADD  dest  :=}           ||
-{ src + 1     }----------||
                           ||
 {MUL  dest  :=}           ||
-{ src * 1     }----------||
                           ||
 {DIV  dest  :=}           ||
-{ src / 1     }----------||
                           ||
```

dodawanie

mnożenie

dzielenie

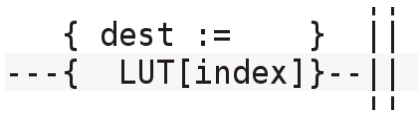


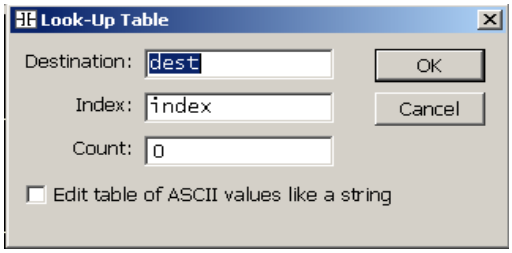


deklarujemy rejestr (uporządkowany ciąg zmiennych) w tym przypadku składający się z 7 – dmiu zmiennych reg0-reg6

każde narastające zbocze sygnału pojawiające się na wejściu powoduje następująca operację reg6=reg5, reg5=reg4, …. reg1 = reg0, reg0 pozostaje bez zmian

…….



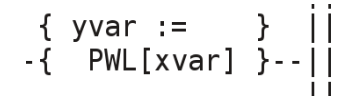


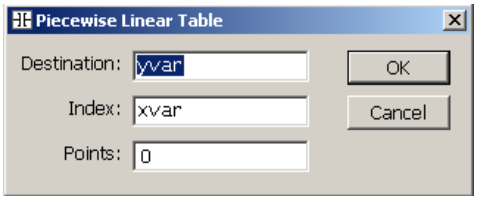
ciąg zmiennych,
jeśli na wejściu pojawi się sygnał to zmienna dest przyjmuje wartość zmiennej o numerze index

przykład:

tablica to zestaw znaków”poniedzialek”
jeśli index=3 i na wejściu instrukcji pojawi się sygnał to dest = n (bo „n” to trzeci znak słowa poniedziałek)

…….





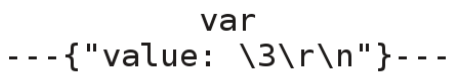
służy do aproksymowania funkcji lub krzywych łamaną
np. jeśli:

f(0) = 2
f(5) = 10
f(10) = 50
f(100) = 100

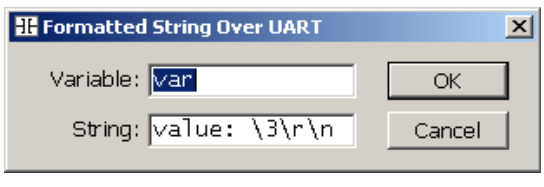
to znaczy , że punkty

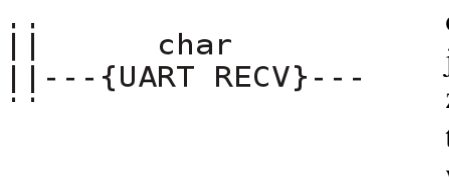
(x0, y0) = (0, 2)
(x1, y1) = (5, 10)
(x2, y2) = (10, 50)
(x3, y3) = (100, 100)

…….



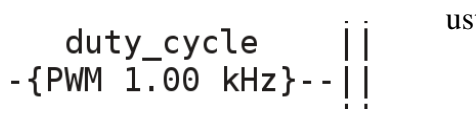
wysyłanie ciągu znaków przez RS232



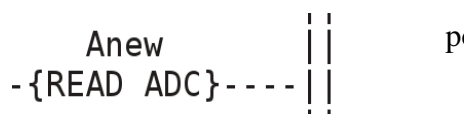


odbiór znaku z portu RS232
jeśli na wejściu pojawi się sygnał to czyta z RS232 jeśli brak znaku to stan wyjścia jest zerowy, jeśli odczyta znak ASCII, to jego wartość jest przechowywana w zmiennej „var" a na wyjściu pojawia się sygnał, odbywa się to dla każdego cyklu PLC.

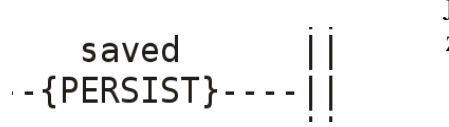
wysyłanie znaku przez port RS232
w czasie wysyłania na wyjściu pojawia się sygnał zajętości



ustawienie sygnału PWM (zakres 0-100)



pomiar wartości napięcia (np. sygnału z czujnika)



jeśli sygnał na wejściu to zapisuje do pamięci nieulotnej zmienną „saved”