OB1 - <offline>

"Cycle Execution"

Name: **Familie:**
Autor: **Version:** 00.01
Bausteinversion: 2
Zeitstempel Code: 19.01.2002 11:58:51
Interface: 15.01.2002 22:22:24
Längen (Baustein / Code / Daten): 00512 00356 00020

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	temp	Default	ARRAY[1..20]		
*1.0	temp		BYTE		

Baustein: OB1 Grundschaltungen für die Formelsammlung

Netzwerk: 1 UND-Glied

Eingang E0.0 und Eingang E0.1 ist gleich Ausgang A0.0

```
      U     E      0.0  // Und Eingang E0.0
      U     E      0.1  // Und Eingang E0.1
      =     A      0.0  // ist gleich Ausgang E0.0
```

Netzwerk: 2 UND-NICHT-Glied

Nicht Eingang E0.0 und nicht Eingang E0.1 ist gleich Ausgang A0.0

```
      UN    E      0.0  // UND NICHT Eingang E0.0
      UN    E      0.1  // UND NICHT Eingang E0.1
      =     A      0.0  // ist gleich Ausgang A0.0
```

Netzwerk: 3 NICHT-Glied (in FUP nur mit UND)

Und nicht Eingang E0.0 ist gleich Ausgang A0.0

```
      UN    E      0.0  // UND NICHT Eingang E0.0
      =     A      0.0  // ist gleich Ausgang A0.0
```

Netzwerk: 4 ODER-Glied

Eingang E0.0 oder Eingang E0.1 ist gleich Ausgang A0.0

```
      O     E      0.0  // ODER Eingang E0.0
      O     E      0.1  // ODER Eingang E0.1
      =     A      0.0  // ist gleich Ausgang A0.0
```


Netzwerk: 5 ODER-NICHT-Glied

Nicht Eingang E0.0 oder nicht Eingang E0.1 ist gleich Ausgang A0.0

```
      ON    E      0.0  // ODER NICHT Eingang E0.0
      ON    E      0.1  // ODER NICHT Eingang E0.1
      =     A      0.0  // ist gleich Ausgang A0.0
```

Netzwerk: 6 Setzen eines Ausgangs

Und Eingang E0.0 setzt Ausgang A0.0

```
      U     E      0.0  // UND Eingang E0.0
      S     A      0.0  // Setzen Ausgang A0.0
```

Netzwerk: 7 Rücksetzen eines Ausgangs

Und Eingang E0.0 setzt Ausgang A0.0 zurück

```
      U     E      0.0  // UND Eingang E0.0
      R     A      0.0  // Rücksetzen Ausgang A0.0
```

Netzwerk: 8 NOP-Operation

Tue nichts (Nur in AWL)

```
      NOP   0  // Tue nichts (Nur in AWL)
```

Netzwerk: 9 UND-Verschachtelung

(Eingang E0.0 oder E0.1) und (Eingang E0.1 oder E0.3) und Eingang 0.4 ist
gleich
Ausgang A0.0

```
      U(                 // ** Beginn 1. UND-Verschachtelung
      O     E      0.0   // ODER Eingang E0.0
      O     E      0.1   // ODER Eingang E0.1
      )                  // ** Ende 1. UND-Verschachtelung

      U(                 // ** Beginn 2. UND-Verschachtelung
      O     E      0.2   // ODER Eingang E0.2
      O     E      0.3   // ODER Eingang E0.3
      )                  // ** Ende 2. UND-Verschachtelung

      U     E      0.4   // UND Eingang E0.4
      =     A      0.0   // ist gleich Ausgang A0.0
```

Netzwerk: 10 ODER-Verschachtelung

(Eingang E0.0 und Eingang E0.1) oder (Eingang E0.2 und nicht Eingang E0.3) oder
(Nicht Eingang E0.4 und nicht Eingang E0.5) ist gleich Ausgang A0.0

```
      U     E      0.0  // UND Eingang E0.0
      U     E      0.1  // UND Eingang E0.1

      O                 // ODER

      U     E      0.2  // UND Eingang E0.2
      UN    E      0.3  // UND NICHT Eingang E0.3

      O                 // ODER

      UN    E      0.4  // UND NICHT Eingang E0.4
      UN    E      0.5  // UND NICHT Eingang E0.5
      =     A      0.0  // ist gleich Ausgang A0.0
```

Netzwerk: 11 Flip-Flop ohne Zustands-Auswertung

Und Eingang E0.0 setzt Ausgang A0.0;
Und nicht Eingang E0.1 setzt Ausgang A0.0 zurück;
Rücksetzen hat Vorrang !!!

```
      U     E      0.0  // UND Eingang E0.0
      S     A      0.0  // Setzt A0.0

      UN    E      0.1  // UND NICHT Eingang E0.1
      R     A      0.0  // Setzt A0.0 zurück

      NOP   0            // Zustand des Flip-Flop nicht benutzt
```

Netzwerk: 12 Flip-Flop mit Ausgangsauswertung

Und Eingang E0.0 setzt Ausgang A0.0;
Und nicht Eingang E0.1 setzt Ausgang A0.0 zurück;
Rücksetzen hat Vorrang !!!
Und Ausgang (des Flip-Flop) A0.0 ist gleich Ausgang A0.1

```
      U     E      0.0  // UND Eingang E0.0
      S     A      0.0  // Setzt A0.0

      UN    E      0.1  // UND NICHT Eingang E0.1
      R     A      0.0  // Setzt A0.0 zurück

      U     A      0.0  // UND Ausgang A0.0 (= Zustand des Flip-Flop)
      =     A      0.1  // ist gleich Ausgang A0.1
```


Netzwerk: 13 Timer mit Impulsverhalten - Minimale AWL

Und Eingang E0.0 lädt Zeit (= 5 s) für Timer (Timerwort)
und startet Timer 1 mit Impulsverhalten.

Timer bleibt solange 1 bis E0.0 gleich 0 wird oder Zeit abgelaufen ist

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#5S      // Lade Zeit für Timer (= 5s)
      SI    T      1    // Starte Timer 1 mit Impulsverhalten

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0            // Reset wird nicht benutzt
      NOP   0            // Dual-Zeitwert wird nicht benutzt
      NOP   0            // BCD-Zeitwert wird nich benutzt
      NOP   0            // Zustand des Timers wird nicht benutzt
```

Netzwerk: 14 Timer mit Impulsverhalten - komplette AWL

Und Eingang E0.0 lädt Zeit für Timer (Timerwort, Zeit = 5 min)
und startet Timer 1 mit Impulsverhalten.
UND Eingang E0.1 setzt den Timer 1 zurück.
UND Timer 1 (Zeit läuft) ist gleich Ausgang A0.0

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#5M      // Lade Zeit für Timer (= 5min)
      SI    T      1    // Starte Timer 1 mit Impulsverhalten

      U     E      0.1  // UND Eingang E0.1
      R     T      1    // Setzt Timer 1 zurück

// ***** Von hier ab bei allen Timern gleich ******
      L     T       1   // Lade Zeitwert von Timer 1 in den Akku
      T     MW     10   // Transferiere Akku in Merkerwort 10

      LC    T       1   // Lade Zeitwert von Timer 1 als BCD-Code in Akku
      T     AW     10   // Transferiere Akku in Ausgangswort 10

      U     T       1   // UND Timer 1 (=Zustand des Timers)
      =     A      0.0  // ist gleich Ausgang A0.0
```

Netzwerk: 15 Timer mit verlängertem Impuls - Minimale AWL

Und Eingang E0.0 lädt Zeit (= 2 h) für Timer (Timerwort)
und startet Timer 1 mit verlängertem Impuls.

Timer bleibt solange 1 bis die Zeit abgelaufen ist.

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#2H      // Lade Zeit für Timer (= 2h)
      SV    T      1    // Starte Timer 1 mit verlängertem Impuls

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0            // Reset wird nicht benutzt
      NOP   0            // Dual-Zeitwert wird nicht benutzt
      NOP   0            // BCD-Zeitwert wird nich benutzt
      NOP   0            // Zustand des Timers wird nicht benutzt
```

```
Netzwerk: 16     Timer mit Einschaltverzögerung - Minimale AWL

Und Eingang E0.0 lädt Zeit (= 5 s) für Timer (Timerwort)
und startet Timer 1 mit Einschaltverzögerung

Timer wird erst 1 wenn E0.0 noch gleich 1 ist und die Zeit abgelaufen i
st.
Timer wird 0 wenn E0.0 gleich 0 wird
```

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#5S      // Lade Zeit für Timer (= 5s)
      SE    T      1    // Starte Timer 1 mit Einschaltverzögerung

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0           // Reset wird nicht benutzt
      NOP   0           // Dual-Zeitwert wird nicht benutzt
      NOP   0           // BCD-Zeitwert wird nich benutzt
      NOP   0           // Zustand des Timers wird nicht benutzt
```

```
Netzwerk: 17     Timer mit speichernder Einschaltverzögerung - Minimale
 AWL

Und Eingang E0.0 lädt Zeit (= 5 s) für Timer (Timerwort)
und startet Timer 1 mit speichernder Einschaltverzögerung.
Und nicht Eingang E0.1 setzt den Timer 1 zurück.

Timer wird erst 1 wenn E0.0 noch gleich 1 ist und die Zeit abgelaufen i
st.
Timer bleibt auf 1 bis er zurückgesetzt wird.
```

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#5S      // Lade Zeit für Timer (= 5s)
      SS    T      1    // Starte Timer 1 mit speichernder Einschaltverzö
                        gerung

      UN    E      0.1  // UND NICHT Einang E0.1
      R     T      1    // Timer 1 zurücksetzen

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0           // Dual-Zeitwert wird nicht benutzt
      NOP   0           // BCD-Zeitwert wird nich benutzt
      NOP   0           // Zustand des Timers wird nicht benutzt
```

```
Netzwerk: 18     Timer mit Ausschaltverzögerung - Minimale AWL

Und Eingang E0.0 lädt Zeit (= 5 s) für Timer (Timerwort)
und startet Timer 1 mit Ausschaltverzögerung

Timer wird 1 wenn E0.0 gleich 1 ist. Wenn E0.0 gleich 0 wird, startet d
ie Zeit.
Timer bleibt 1 bis die Zeit abgelaufen ist
```

```
      U     E      0.0  // UND Eingang E0.0
      L     S5T#5S      // Lade Zeit für Timer (= 5s)
      SA    T      1    // Starte Timer 1 mit Ausschaltverzögerung

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0           // Reset wird nicht benutzt
      NOP   0           // Dual-Zeitwert wird nicht benutzt
```



```
      NOP   0           // BCD-Zeitwert wird nich benutzt
      NOP   0           // Zustand des Timers wird nicht benutzt
```

```
Netzwerk: 19     Vorwärtszähler - Minimale AWL

Und Eingang E0.0 zählt Zähler 1 vorwärts.

Der Zähler zählt bis maximal 999 und zählt nicht weiter bis er zurückge
setz
wird.
```

```
      U     E      0.0  // UND Eingang E0.0
      ZV    Z      1    // Zählt Zähler 1 vorwärts

      BLD   101         // Nur für FUP-Umsetzung notwendig

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0           // Zählerstand laden nicht benutzt
      NOP   0           // Zählerstand setzen nicht benutzt
      NOP   0           // Reset nicht benutzt
      NOP   0           // Dual-Zählerwert nicht benutzt
      NOP   0           // BCD-Zählerwert nicht benutzt
      NOP   0           // Zustand des Zählers nicht benutzt
```

```
Netzwerk: 20     Vorwärtszähler - Komplette AWL

Und Eingang E0.0 zählt Zähler 1 vorwärts.
Und Eingang E0.1 setzt Zähler 1 auf bestimmten Zählerstand
Und Eingang E0.2 setzt Zähler 1 zurück
Und Zähler 1 (=Zustand) ist gleich Ausgang A0.0

Der Ausgang A0.0 ist 1 wenn Zähler 1 <> 0 ist
```

```
      U     E      0.0  // UND Eingang E0.0
      ZV    Z      1    // Zählt Zähler 1 vorwärts

      BLD   101         // Nur für FUP-Umsetzung notwendig

      U     E      0.1  // UND Eingang E0.1
      L     C#5         // Lade Akku mit Zählerstand 5
      S     Z      1    // Setze Zählerstand von Zähler 1 auf Akkuinhalt

// ***** Von hier ab bei allen Zählern gleich  *****
      U     E      0.2  // UND Eingang E0.2
      R     Z      1    // Rücksetzen des Zählers Z1

      L     Z      1    // Lade Zählerinhalt von Zähler Z1 in Akku
      T     MW     10   // Transferiere Akku in Merkerwort 10

      LC    Z      1    // Lade Zählerwert von Zähler Z1 als BCD-Code in
                        Akku
      T     AW     10   // Transferiere Akku in Ausgangswort 10

      U     Z      1    // UND Zähler Z1 (=Zustand von Zähler Z1)
      =     A      0.0  // Ist gleich Ausgang A0.0
```

Netzwerk: 21 Rückwärtszähler - Minimale AWL

Und Eingang E0.0 zählt Zähler 1 rückwärts.
Und Eingang E0.1 setzt Zähler auf Zählerstand 5

Der Zähler zählt minimal bis 0

```
      U     E      0.0  // UND Eingang E0.0
      ZR    Z      1    // Zählt Zähler 1 rückwärts

      BLD   101          // Nur für FUP-Umsetzung notwendig

      U     E      0.1  // UND Eingang E0.1
      L     C#5          // Lade Akku mit Zählerstand 5
      S     Z      1    // Setze Zählerstand von Zähler 1 auf Akkuinhalt

// ***** NOP's bei minimaler Anweisungsliste nicht nötig *****
      NOP   0            // Reset nicht benutzt
      NOP   0            // Dual-Zählerwert nicht benutzt
      NOP   0            // BCD-Zählerwert nicht benutzt
      NOP   0            // Zustand des Zählers nicht benutzt
```

Netzwerk: 22 Richtimpuls

Und nicht Merker 80.0 ist gleich Merker 80.1 (=Richtimpuls)
Setzen Merker 80.0

Beim 1. Durchlauf der AWL wird ein Impuls auf Merker 80.1 erzeugt.
Er kann verwendet werden um den Grundzustand zu erreichen

```
      UN    M     80.0  // UND NICHT Merker M80.0
      =     M     80.1  // ist gleich Merker M80.1 (= Richtimpuls)
      S     M     80.0  // Setzen Merker M80.0
```

Netzwerk: 23 Beispiel für Verwendung des Richtimpuls

Merker M80.1 (=Richtimpuls) oder Merker 1.3 (=letzter Zustand, Schritt)
ist gleich Merker 1.1 (=Grundzustand, 1.Schritt)

```
      U(                 // Beginn Und-Verschachtelung
      O     M     80.1   // ODER Merker M80.1 (=Richtimpuls)
      O     M      1.3   // ODER Merker M1.3 (=letzter Zustand)
      )                  // Ende Und-Verschachtelung
      S     M      1.1   // Setzen Merker M1.1 (=Grundzustand)

      U     M      1.2   // UND Merker 1.2 (=nächster Zustand, Schritt)
      R     M      1.1   // Rücksetzen Merker 1.1

      NOP   0            // Zustand des Flip-Flop nicht benutzt
```


Netzwerk: 24 Vergleichsoperationen Ganzzahlen (16 bit)

Vergleicht Merker MW12 mit Merker MW10
Die Merker MW10 und MW12 haben 16 bit Wortlänge

Mögliche Operationen:
==I MW12 gleich MW10
<>I MW12 ungleich MW10

>I MW12 größer MW10
<I MW12 kleiner MW10

>=I MW12 glößer gleich MW10
<=I MW12 kleiner gleich MW10

Wenn die Bedingung stimmt wird der Ausgang A0.0 auf 1 gesetzt

```
      L     MW    10    // Lade Merker MW10 in Akku1
      L     MW    12    // Lade Merker MW12 in Akku2
      ==I               // Vergleiche Akku 2 mit 1
      =     A      0.0  // Ergebnis ist gleich Ausgang A0.0
```

Netzwerk: 25 Vergleichsoperationen Ganzzahl (16 bit) mit Konstante

Vergleicht INT-Konstante mit Merker MW10
Der Merker MW10 hat 16 bit Wortlänge

Die INT-Konstante kann sich im Wert von -32768 bis +32768 bewegen
(INT = Integer)

```
      L     MW    10    // Lade Merker MW10 in Akku1
      L     5           // Lade Konstante 5
      ==I               // Vergleiche Akku 2 mit 1
      =     A      0.0  // Ergebnis ist gleich Ausgang A0.0
```

Netzwerk: 26 Vergleichsoperationen Ganzzahlen (32 bit)

Vergleicht Merker MD12 mit Merker MD10
Die Merker MD10 und MD12 haben 32 bit Wortlänge

Mögliche Operationen:
==I MD12 gleich MD10
<>I MD12 ungleich MD10

>I MD12 größer MD10
<I MD12 kleiner MD10

>=I MD12 glößer gleich MD10
<=I MD12 kleiner gleich MD10

Wenn die Bedingung stimmt wird der Ausgang A0.0 auf 1 gesetzt

```
      L     MD    10    // Lade Merker MD10 in Akku1
      L     MD    12    // Lade Merker MD12 in Akku2
      ==D               // Vergleiche Akku 2 mit 1
      =     A      0.0  // Ergebnis ist gleich Ausgang A0.0
```

Netzwerk: 27 Vergleichsoperationen Ganzzahl (32 bit) mit Konstante

Vergleicht LINT-Konstante mit Merker MD10
Der Merker MD10 hat 32 bit Wortlänge

Die LINT-Konstante kann sich im Bereich von -2147483648 bis +2147483648 bewegen
(LINT = Long Integer)

```
      L     MD    10    // Lade Merker MD10 in Akku1
      L     L#70000     // Lade LINT-Konstante 70000
      ==D               // Vergleiche Akku 2 mit 1
      =     A      0.0  // Ergebnis ist gleich Ausgang A0.0
```

Netzwerk: 28 Vergleichsoperationen Gleitkommazahlen (16 bit)

Vergleicht Merker MD12 mit Merker MD10
Die Merker MD10 und MD12 sind 32bit-Gleitkommazahlen

Mögliche Operationen:
==I MD12 gleich MD10
<>I MD12 ungleich MD10

>I MD12 größer MD10
<I MD12 kleiner MD10

>=I MD12 glößer gleich MD10
<=I MD12 kleiner gleich MD10

Wenn die Bedingung stimmt wird der Ausgang A0.0 auf 1 gesetzt

```
      L     MD    10    // Lade Merker MD10 in Akku1
      L     MD    12    // Lade Merker MD12 in Akku2
      ==R               // Vergleiche Akku 2 mit 1
      =     A      0.0  // Ergebnis ist gleich Ausgang A0.0
```

Netzwerk: 29 Vergleichsoperationen Gleitpunktzahl (32 bit) und Konstante

Vergleicht Konstante mit Merker MD10
Der Merker MD10 und und die Konstante sind 32bit-Gleitkommazahlen

Die Gleitkomma-Konstante kann sich im
negativen Bereich von -1.175495e-38 bis -3.402823e+38 bewegen und im
positiven Bereich von +1.175495e+38 bis +3.402823e+38 bewegen.

```
      L     MD    10       // Lade Merker MD10 in Akku1
      L     1.359000e+002  // Lade Gleitkomma-Konstante
      ==R                  // Vergleiche Akku 2 mit 1
      =     A      0.0     // Ergebnis ist gleich Ausgang A0.0
```