21 Verwenden von Skripts

(Programmieren ohne Elemente)

In diesem Kapitel wird erklärt, wie GP-Pro EX zum "Programmieren ohne Elemente" verwendet werden kann und wie man Skripten erstellt.
Bitte lesen Sie zuerst "21.1 Einstellungsmenü" (seite 21-2) und gehen dann zur entsprechenden Seite.

21.1 Einstellungsmenü

Sie können D-Skripte zum Erstellen einfacher Programme verwenden. Mit Hilfe dieser Funktion können Sie Operationen in der GP ausführen oder zwischen der GP und einem nicht unterstütztem Peripherieteilnehmer kommunizieren.

⚠ WARNUNG

Vergewissern Sie sich, keine D-Skripte/Globale D-Skripte zur Steuerung von Systemen zu verwenden, die lebensgefährliche oder ernsthafte Verletzungen hervorrufen können.

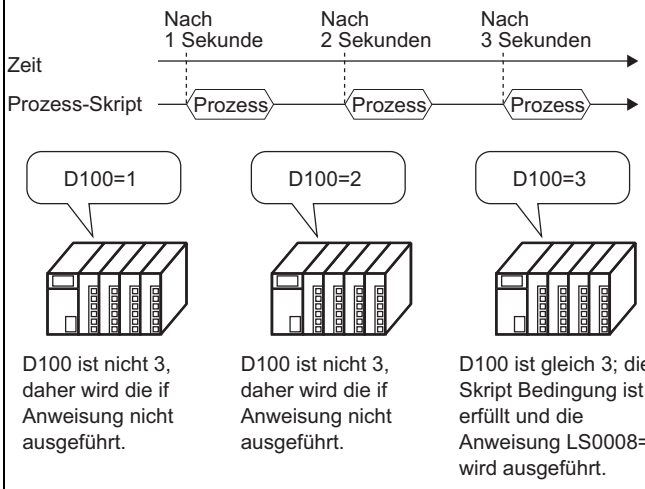
ANMERKUNG

- D-Skripte werden auf dem Basis-Bildschirm eingerichtet. Der Basis-Bildschirm schaut sich die Bedingung an, während sie angezeigt wird und führt das Skript aus.
- Wenn die GP ausführt, wird bei globalen D-Skripten unabhängig vom angezeigten Bildschirm ein Programm ausgeführt, das auf den Trigger beruht.
- Erweiterte Skripte sollten für Kommunikationsprogramme höherer Stufen verwendet werden.
- Außer Skripten können auch Logikprogramme für Steuerungsanwendungen verwendet werden.

☞ "28.1 Einstellungsmenü" (seite 28-2)

Bedingte Operationen

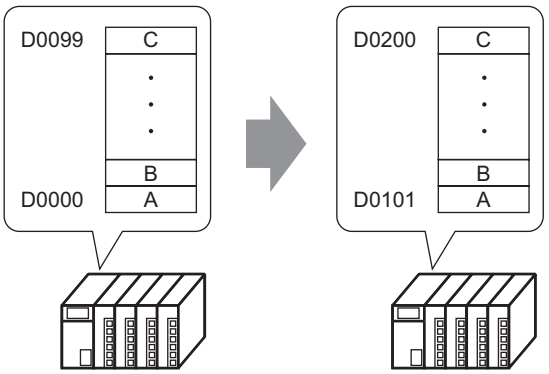
Erstellen Sie ein Skript, das automatisch die Bildschirme wechselt (nach 3 Sekunden auf Bildschirm 7).



☞ Einrichtungsverfahren (seite 21-7)
☞ Einleitung (seite 21-6)

Kopieren von Daten in Blöcke

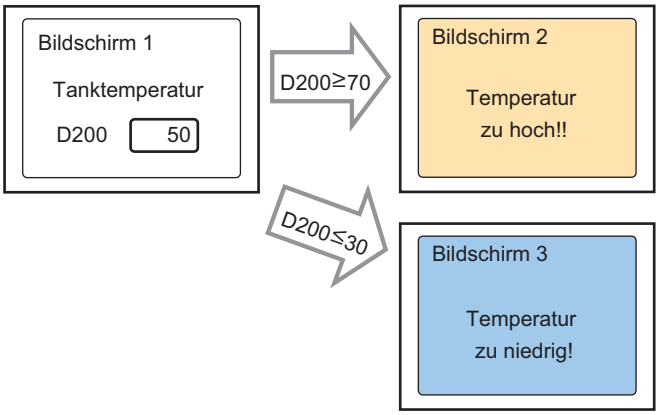
Erstellen Sie ein Skript, welches die ansteigende Flanke (0 bis 1) der Bitadresse M0100 erkennt und die in den verbundenen Teilnehmern gespeicherten Daten in eine andere Adresse kopiert.



☞ Einrichtungsverfahren (seite 21-14)
☞ Einleitung (seite 21-13)

Anzeigen eines Alarms, wenn ein Fehler auftritt

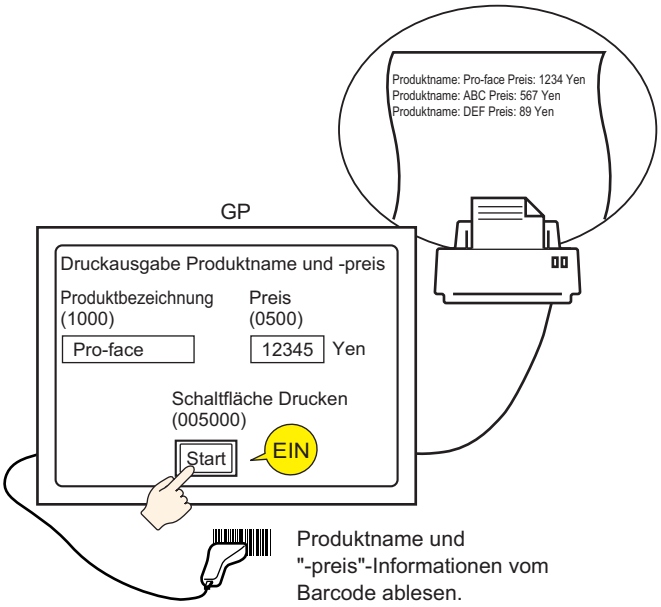
Das Temperatur-Managementsystem erkennt ein Fehlerbit der verbundenen Teilnehmer und zeigt Alarmmeldungen an, wenn die Speicheradresse der Temperaturinformation (D200) auf 70°C oder darüber ansteigt oder auf 30°C oder darunter fällt. Das Skript wird ebenfalls die Anzahl der erkannten Fehler zählen.



☞ Einrichtungsverfahren (seite 21-20)
☞ Einleitung (seite 21-18)

Kommunizieren mit nicht unterstützten peripheren Teilnehmern

Erstellen Sie ein erweitertes Skript, das Daten von einem Barcode ausliest, der mit dem USB-Port verbunden ist und an einen seriellen Drucker ausgibt, der mit COM1 verbunden ist.

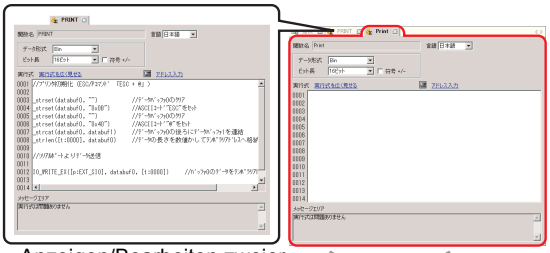


☞ Einrichtungsverfahren (seite 21-35)
☞ Einleitung (seite 21-22)

Referenzieren anderer Skripts

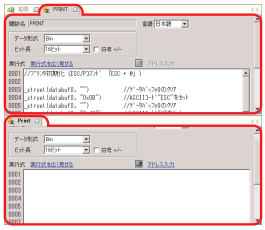
Teilen Sie im Dialogfeld [D-Skript] den Bildschirm horizontal oder vertikal in zwei Bildschirme.

Durch Anklicken der Registerkarten kann zwischen den Bildschirmen gewechselt werden.



Anzeigen/Bearbeiten zweier Bildschirme nebeneinander.

Bildschirm horizontal



Bildschirm vertikal

☞ Durchführungsverfahren (seite 21-41)
☞ Einleitung (seite 21-40)

21.2 Bedingte Operationen

ANMERKUNG

- Weitere Informationen hierzu entnehmen Sie bitte Ihrem Einstellungshandbuch.

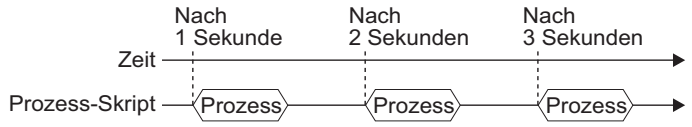
☞ "21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise" (seite 21-56)

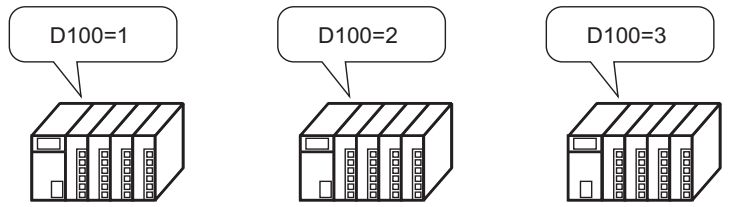
- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

Aktion

Erstellen Sie ein Skript, das automatisch die Bildschirme wechselt (nach 3 Sekunden auf Bildschirm 7).





D100 ist nicht 3, daher wird die if Anweisung nicht ausgeführt.

D100 ist nicht 3, daher wird die if Anweisung nicht ausgeführt.

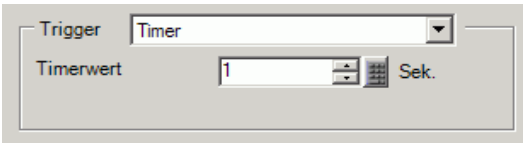
D100 ist gleich 3; die Skript Bedingung ist erfüllt und die Anweisung LS0008=7 wird ausgeführt.

Benutzte Befehle

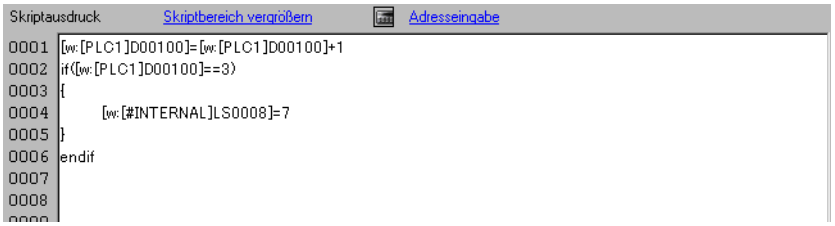
Befehl	Funktionszusammenfassung
Zuweisung (=)	Weisen Sie den Wert auf der rechten Seite dem Wert auf der linken Seite zu. ☞ "21.11.10 Operator" (seite 21-163)
Addition (+)	Fügt den Daten eines Wortteilnehmers eine Konstante hinzu. ☞ "21.11.10 Operator" (seite 21-163)
if ( )	Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr wird, wird der auf die "wenn ( )" Anweisung folgende Vorgang ausgeführt. ☞ "21.11.8 Bedingte Ausdrücke" (seite 21-156)
Gleich (==)	Vergleicht die Werte auf der rechten und linken Seite. Wird "wahr", wenn die linke Seite mit der rechten Seite übereinstimmt. ☞ "21.11.9 Vergleich" (seite 21-161)
LS0008	Änderungen zur Bildschirm-Nr. werden in diesem Wert gespeichert. ☞ "A.1.4.2 Systemdatenbereich" (seite A-10)

Trigger

Wählen Sie den Timer, und legen Sie den [Timerwert] auf 1 Sekunde fest.

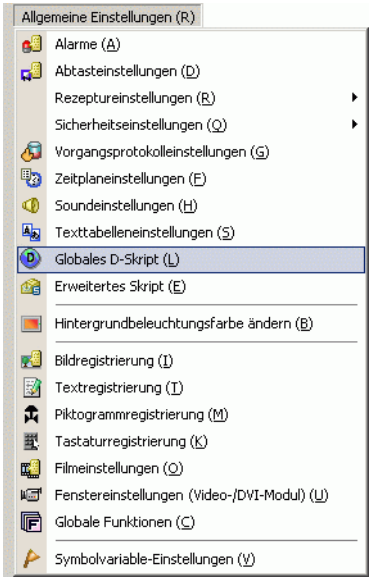


Abgeschlossenes Skript

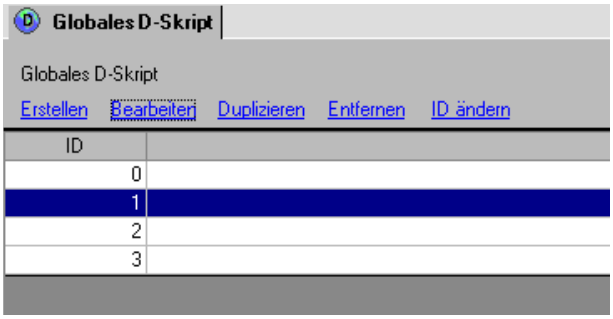


Erstellungsverfahren.

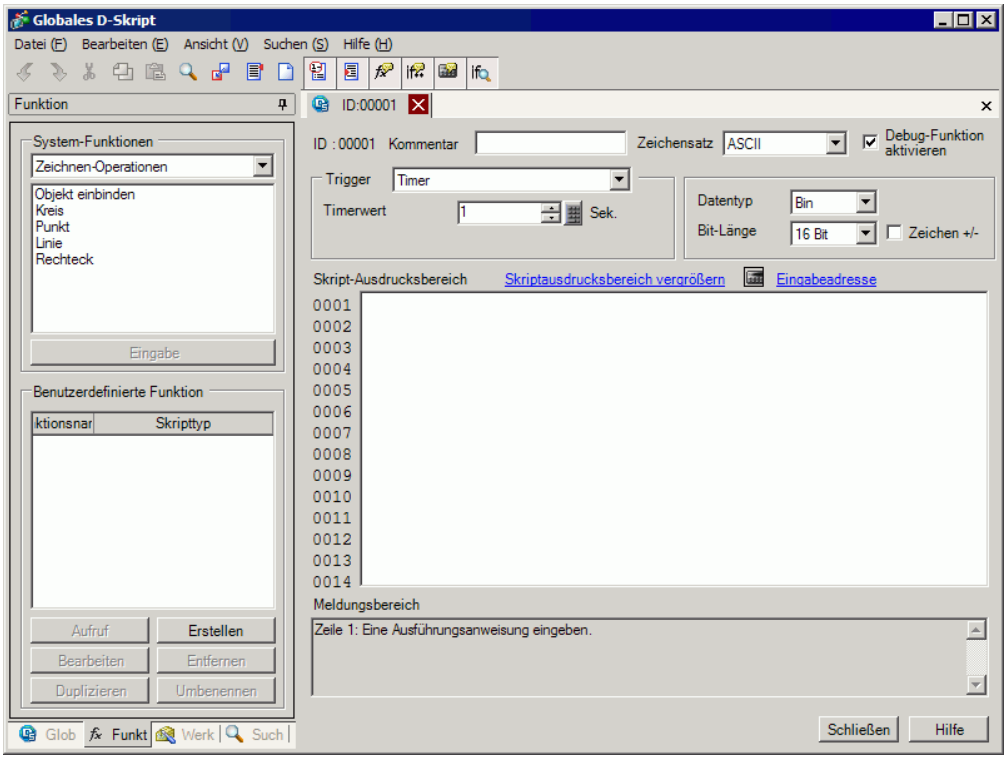
1 Wählen Sie im Menü [Allgemeine Einstellungen (R)] [Globales D-Skript (L)] aus.



2 Klicken Sie auf [Erstellen]. Wenn auf ein vorher registriertes Skript zugegriffen wird, müssen Sie die ID-Nr. eingeben und auf [Bearbeiten] klicken oder die Zeile der ID-Nr. doppelt anklicken.

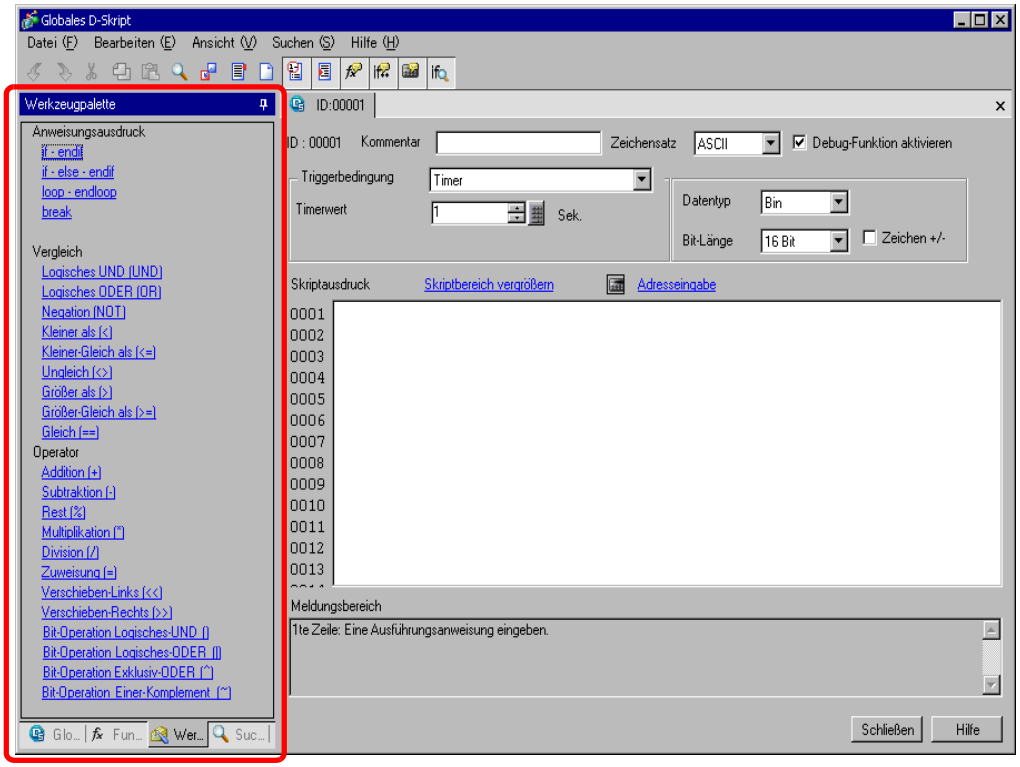


3 Das Dialogfeld [D-Skript] wird angezeigt.

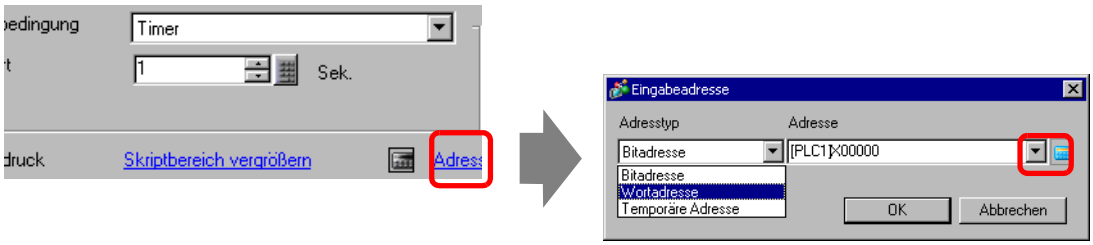


4 Wählen Sie in [Trigger] [Timer] aus und legen die [Timer-Einstellungen] auf 1 Sekunde fest.

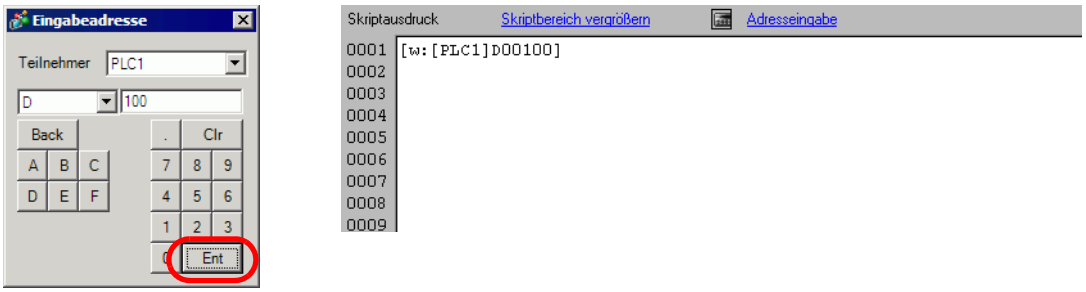
5 Klicken Sie auf die Registerkarte [Werkzeugpalette]. Mit Hilfe der Werkzeugpalette können Befehle einfach zum Verwenden in einem Skript abgelegt werden.



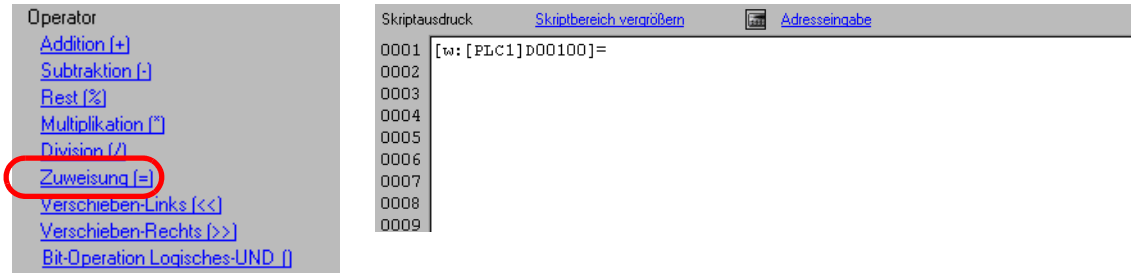
6 Erstellen Sie die erste Zeile des Skripts. Wenn der Standardwert D00100 als 0 festgelegt wird, handelt es sich bei der Operation in der ersten Zeile um eine Anzahl-Operation, die zunimmt und die Anzahl jedes Mal speichert, wenn ein Prozess abgeschlossen ist. Klicken Sie auf , wählen Sie [Wortadresse], und klicken Sie dann auf .



7 Geben Sie D00100 ein klicken auf [EINGABE]. Klicken Sie im Dialogfeld [Adress-Eingabe] auf [OK].

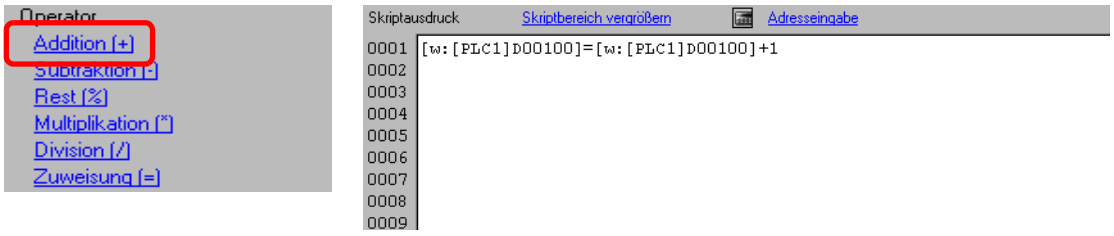


8 Klicken Sie auf [Zuweisung (=)] in der Werkzeugpalette.

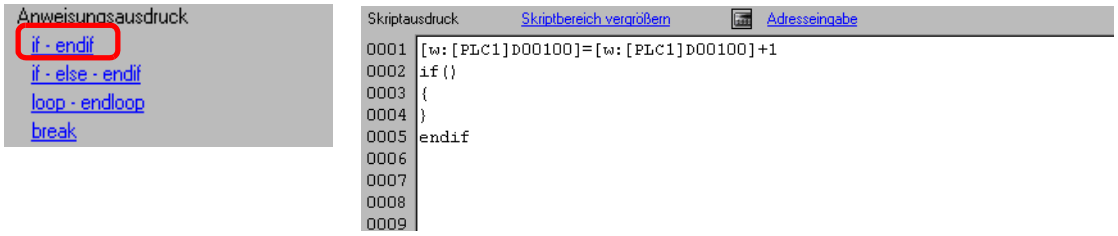


9 Legen Sie D00100 auf dieselbe Art ab wie in den Schritten 6 bis 7.

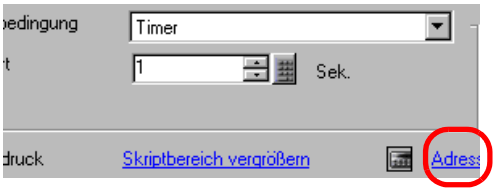
10 Klicken Sie auf [Addition (+)] und geben "1" ein. Die erste Zeile ist damit abgeschlossen.

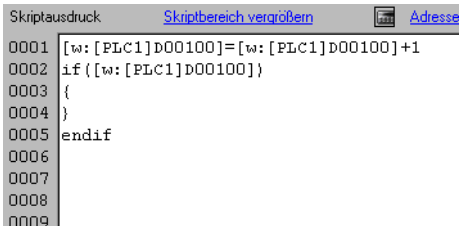


11 Erstellen Sie die zweite Zeile des Skripts. Wenn in der zweiten Zeile eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr wird, wird der auf die "wenn ( )" Anweisung folgende Vorgang ausgeführt.
Klicken Sie auf [if - endif].

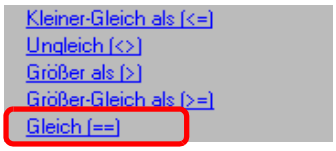


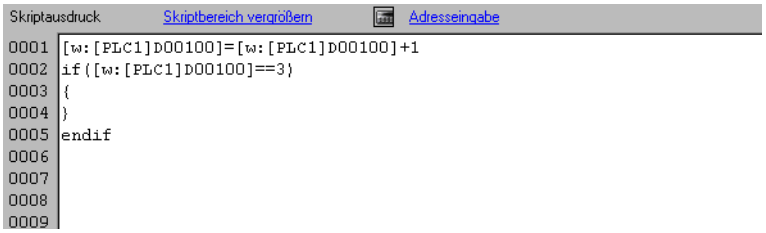
12 Erstellen Sie den Bedingunsausdruck in den Klammern "( )" gefolgt von der Bedingung "if". Der Bedingungsausdruck vergleicht den in D00100 gespeicherten Wert mit "3" und wird "wahr", wenn diese gleich sind. Platzieren Sie den Cursor in die Klammern "( )" und wiederholen die Schritte 6 bis 7, um eine andere D00100 abzulegen.



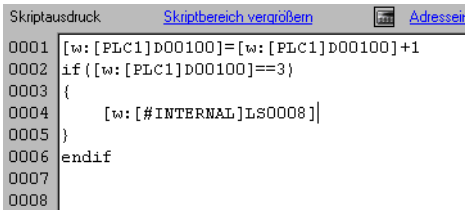


13 Klicken Sie auf [Gleich (==)] und geben "3" ein. Die zweite Zeile ist damit abgeschlossen.

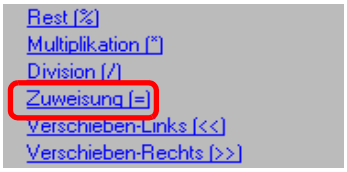


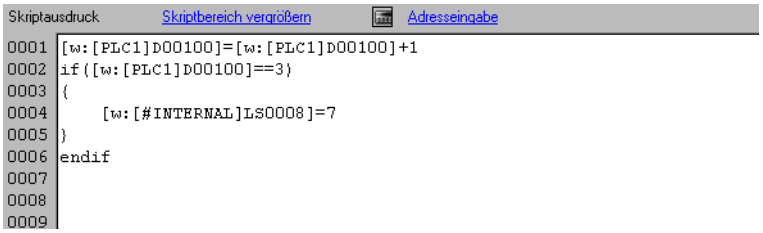


14 Platzieren Sie den Cursor in die Klammern "{ }" und geben einen Zeilenvorschub ein. Wiederholen Sie die Schritte 6 bis 7, um eine andere LS0008 abzulegen.

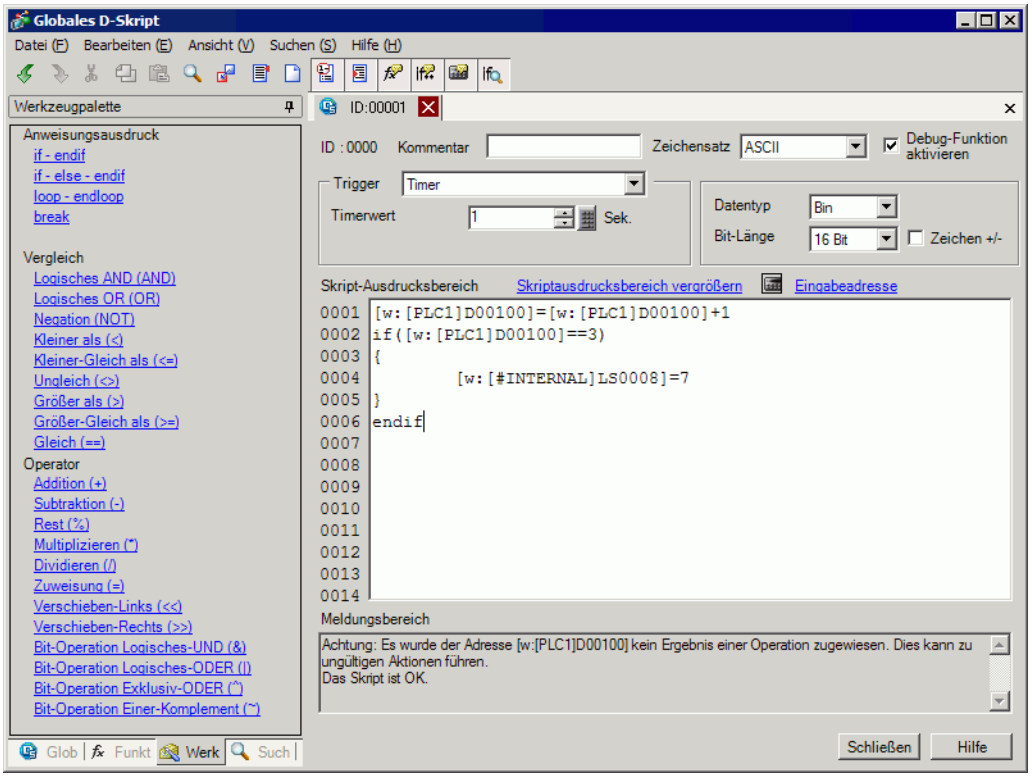


15 Klicken Sie auf [Zuweisung (=)] und geben "7" ein.





16 Das Skript ist nun abgeschlossen.



ANMERKUNG

- Wenn Sie Text auswählen möchten, drücken Sie die [Strg]-Taste + die [Umschalt]-Taste + die [Rechte Pfeiltaste]/[Linke Pfeiltaste], um einen gesamten Bock eines Textes auszuwählen.
- Drücken Sie die [Strg]-Taste + [F4]-Taste, um die aktuell ausgewählte Anzeige zu schließen.
- Drücken Sie die [Esc]-Taste zum Überschreiben des Skripts oder zum Löschen und Beenden.

21.3 Kopieren von Daten in Blöcke

ANMERKUNG

- Weitere Informationen hierzu entnehmen Sie bitte Ihrem Einstellungshandbuch.

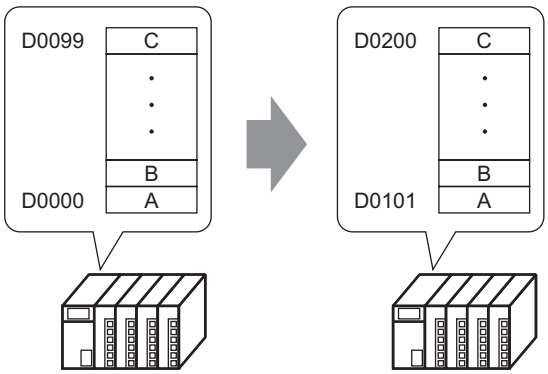
☞ "21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise" (seite 21-56)

- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

Aktion

Erstellen Sie ein Skript, welches die ansteigende Flanke (0 bis 1) der Bitadresse M0100 erkennt und die in den verbundenen Teilnehmern gespeicherten Daten in eine andere Adresse kopiert.

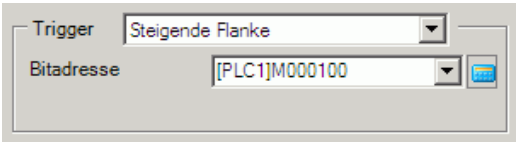


Benutzte Befehle

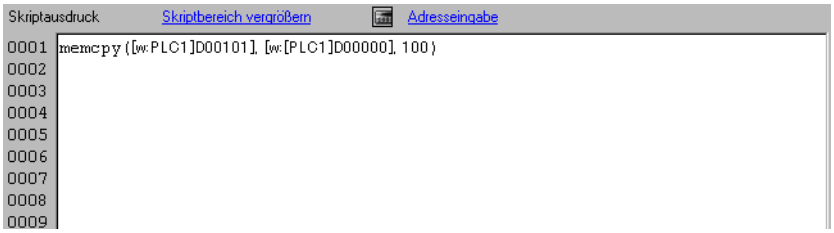
Befehl	Funktionszusammenfassung
Speicher kopieren memcpy ( )	Kopiert einen gespeicherten Wert in einer Operation in einen Teilnehmer. Daten für die Anzahl der Adressen werden in die Kopierziel-Wortadressen kopiert, beginnend mit der ersten Wortadresse der Quelldaten. [Format] memcpy ([In Adresse kopieren], [Aus Adresse kopieren], Worte) ☞ "21.11.3 Speicher-Operationen" (seite 21-84)

Trigger

Wählen Sie in der [Triggerbedingung] [Bit EIN] aus und legen die [Bitadresse] auf M000100 fest.

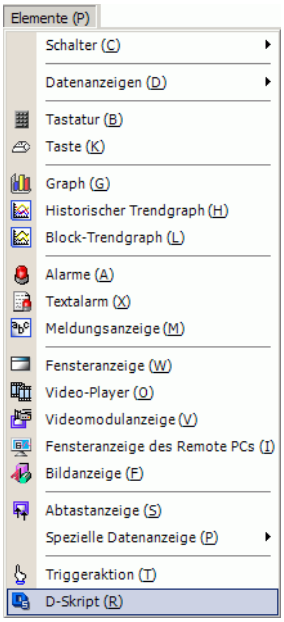


Abgeschlossenes Skript

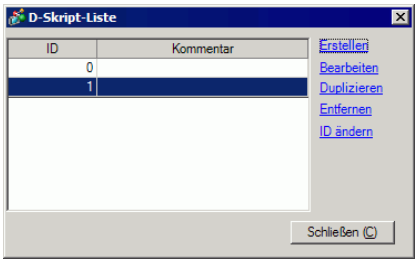


Erstellungsverfahren.

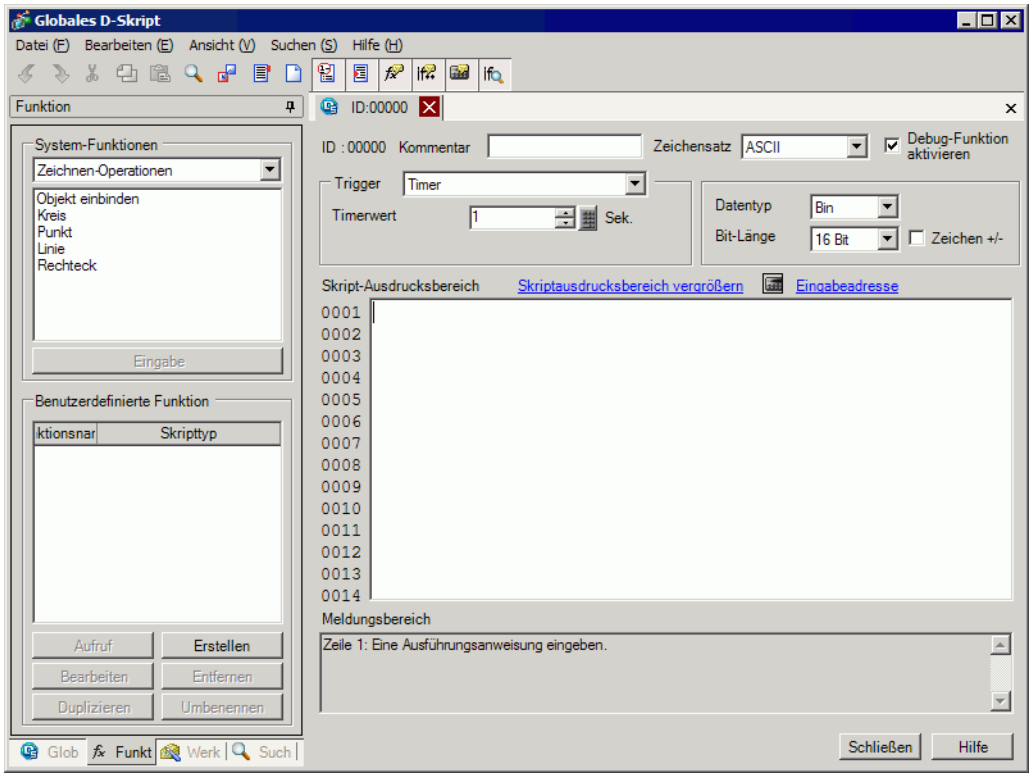
1 Wählen Sie im Menü [Element (P)] [D-Skript (R)] aus oder klicken Sie auf aus der Werkzeugleiste.



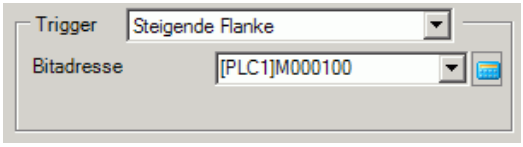
2 Klicken Sie auf [Erstellen]. Die IDs für bestehende Skripte sind in der [D-Skript-Liste] angezeigt.



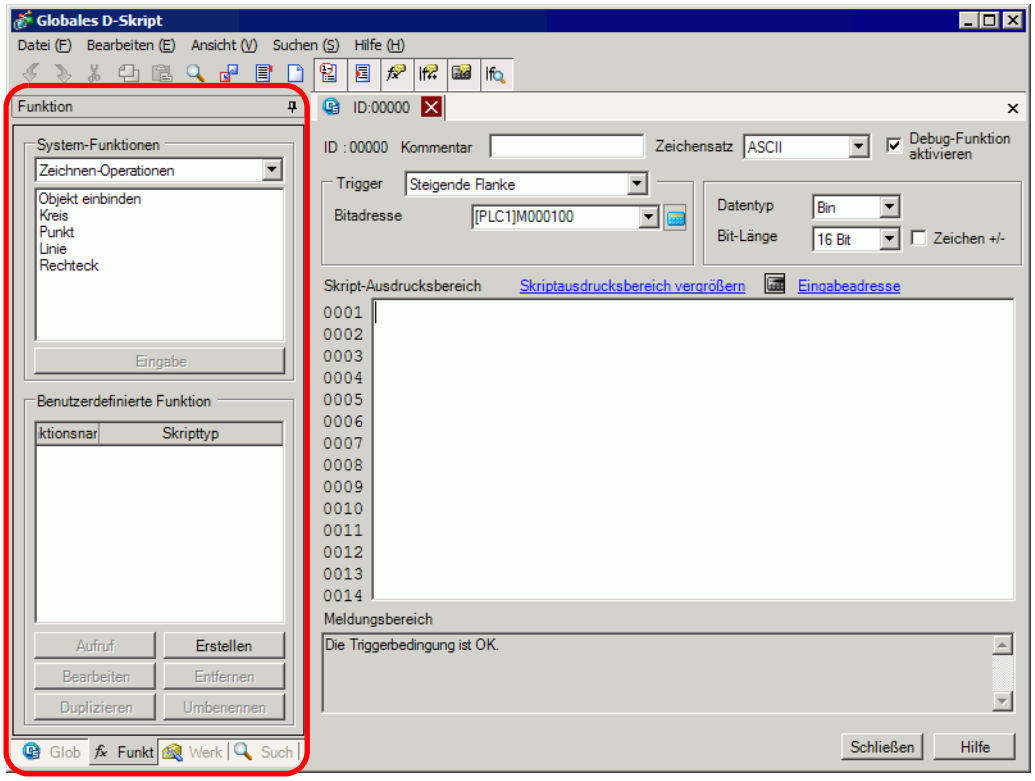
3 Das Dialogfeld [D-Skript] wird angezeigt.



4 Wählen Sie in den Skript-Triggerbedingungen [Trigger] [Bit EIN] aus und bestimmen M000100 als [Bitadresse].

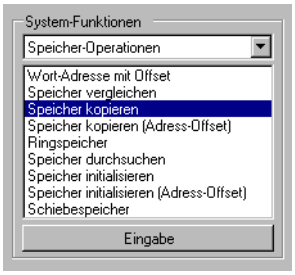


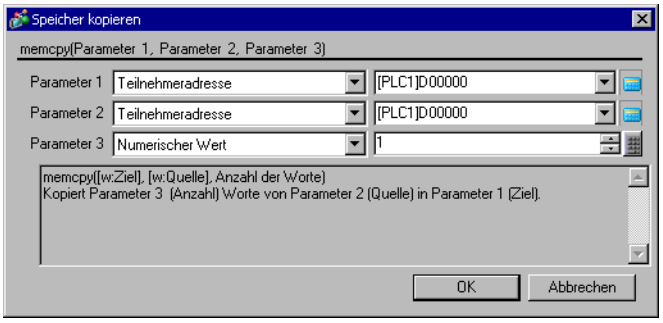
5 Klicken Sie auf die Registerkarte [Funktion]. Mit den System-Funktionen können Sie ganz einfach einen Befehl ablegen, der im Skript benutzt werden kann.



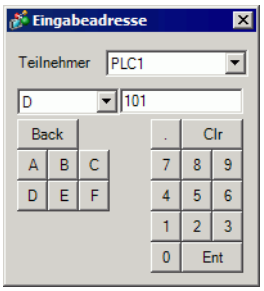
6 Wählen Sie aus [System-Funktionen (Anleitung)] [Speicher-Operationen] aus.

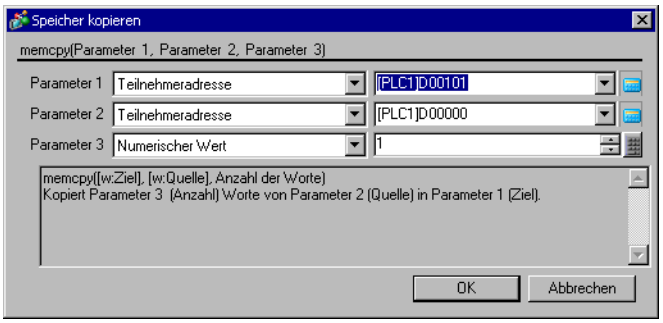
7 Doppelklicken Sie auf [Speicher kopieren], und legen Sie im daruf folgenden Dialogfeld die Parameter für die Zieladresse, Quelladresse und Anzahl der Wörter fest. Klicken Sie auf  .





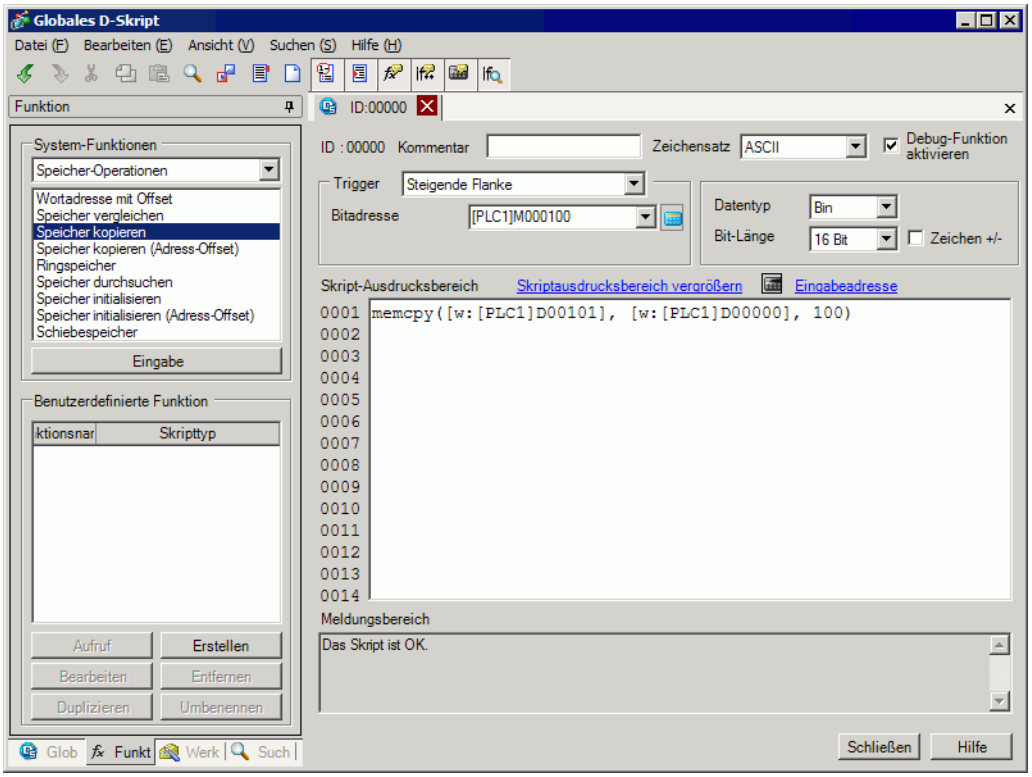
8 Geben Sie für [Parameter 1] D00101 ein und klicken auf [ENT].





9 Geben Sie für [Parameter 2] D00000 ein und klicken auf [OK].

10 Das Skript ist nun abgeschlossen.



21.4 Anzeigen eines Alarms, wenn ein Fehler auftritt

ANMERKUNG

- Weitere Informationen hierzu entnehmen Sie bitte Ihrem Einstellungshandbuch.

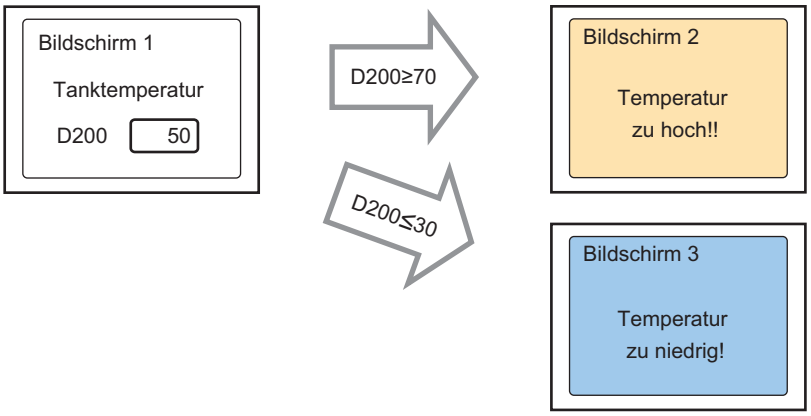
☞ "21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise" (seite 21-56)

- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

Aktion

Das Temperatur-Managementsystem erkennt ein Fehlerbit der verbundenen Teilnehmer und zeigt Alarmmeldungen an, wenn die Speicheradresse der Temperaturinformation (D200) auf 70°C oder darüber ansteigt oder auf 30°C oder darunter fällt. Das Skript wird ebenfalls die Anzahl der erkannten Fehler zählen.



Die Adresse, die jedesmal D200 zählt, fällt auf 70°C oder höher und speichert die Anzahl. LS0300

Die Adresse, die jedesmal D200 zählt, fällt auf 30°C oder niedriger und speichert die Anzahl. LS0301

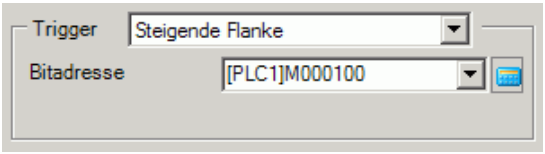
Adresse, die die Alarm-Bildschirm-Nr. speichert: LS0008

Benutzte Befehle

Befehl	Funktionszusammenfassung
if ( )	Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt. ☞ "21.11.8 Bedingte Ausdrücke" (seite 21-156)
Größer-Gleich als (>=)	Ja, wenn N1 größer als oder gleich hoch wie N2 (N1 >= N2) ist. ☞ "21.11.9 Vergleich" (seite 21-161)
Zuweisung (=)	Weisen Sie den Wert auf der rechten Seite dem Wert auf der linken Seite zu. ☞ "21.11.10 Operator" (seite 21-163)
Addition (+)	Fügt den Daten eines Wortteilnehmers eine Konstante hinzu. ☞ "21.11.10 Operator" (seite 21-163)
Kleiner-Gleich als (<=)	Ja, wenn N1 kleiner als oder gleich hoch wie N2 (N1 <= N2) ist. ☞ "21.11.9 Vergleich" (seite 21-161)

Trigger

Wählen Sie in der [Triggerbedingung] [Bit EIN] aus und legen die [Bitadresse] auf M000100 fest.



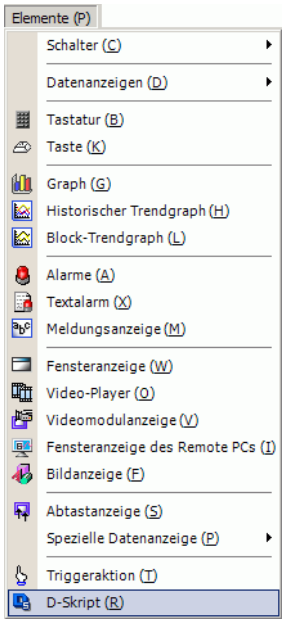
Abgeschlossenes Skript

Skriptausdruck Skriptbereich vergrößern Adresseingabe

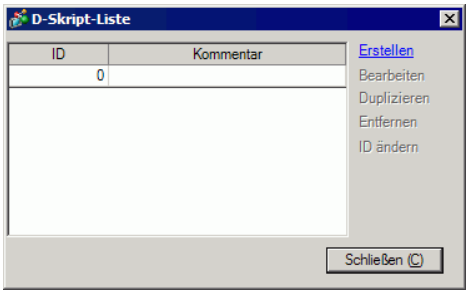
```
0001 if([w:[PLC1]D00200]>=70)                          //When temp is greater than 70 degrees
0002 {
0003    [w:[#INTERNAL]LS0302]=100                      //Greater than 70 degrees alarm screen number 100
0004    [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1  //Increase error count
0005 }
0006 endif
0007
0008 if([w:[PLC1]D00200]>=30)                          //When temp is greater than 30 degrees
0009 {
0010    [w:[#INTERNAL]LS0302]=101                      //Greater than 30 degrees alarm screen number 101
0011    [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1  //Increase error count
0012 }
0013 endif
0014
0015
0016
```

Erstellungsverfahren.

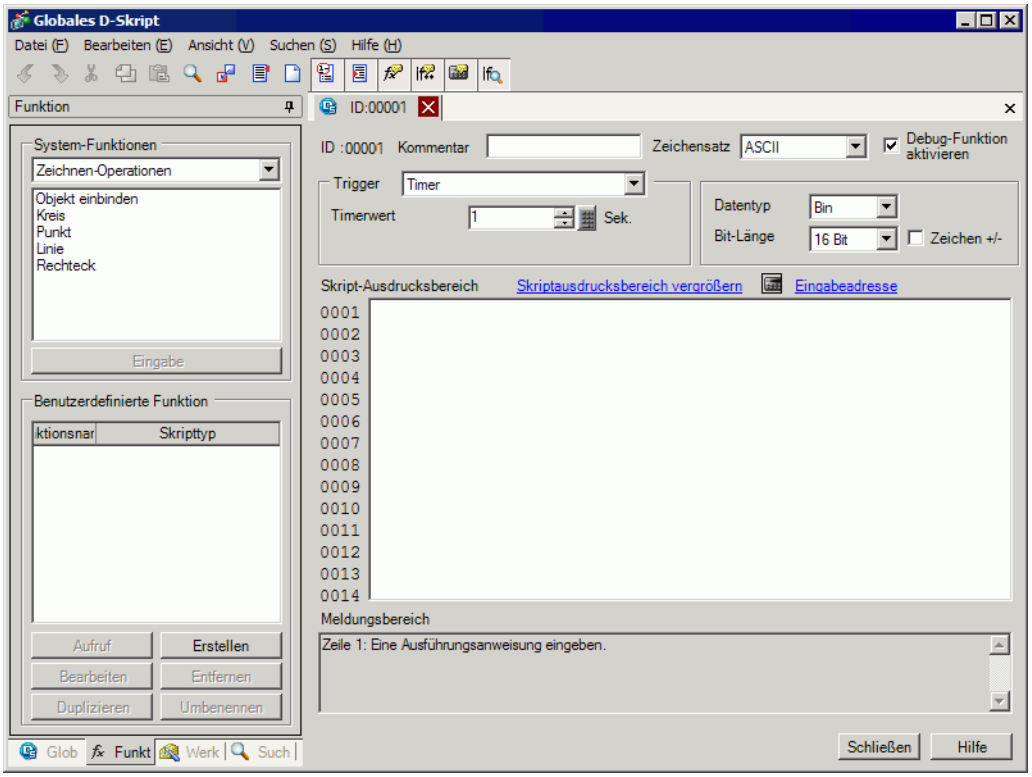
1 Wählen Sie im Menü [Elemente] [D-Skript (R)] aus, oder klicken Sie auf .



2 Klicken Sie auf [Erstellen]. Die IDs für bestehende Skripte sind in der [D-Skript-Liste] angezeigt.



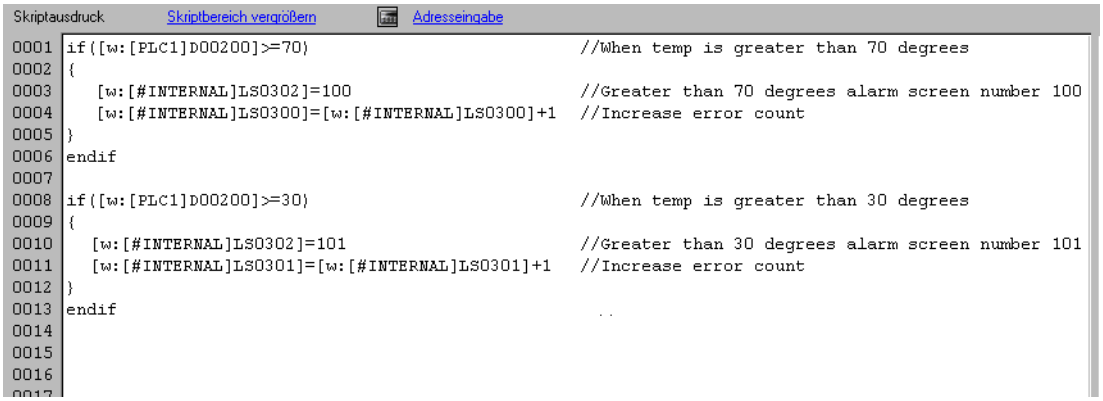
3 Das Dialogfeld [D-Skript] wird angezeigt.



4 Legen Sie Kommentare fest. Geben Sie "Alarmanzeige" ein.

5 Wählen Sie in [Trigger] [Bit EIN] aus und bestimmen M00100 als [Bitadresse].

6 Erstellen Sie ein Programm durch Hinzufügen von Funktionen, Anweisungen und Ausdrücke für den Skriptausdruckbereich, um das Skript fertigzustellen.



ANMERKUNG

- Wenn Sie Text auswählen möchten, drücken Sie die [Strg]-Taste + die [Umschalt]-Taste + die [Rechte Pfeiltaste]/[Linke Pfeiltaste], um einen gesamten Bock eines Textes auszuwählen.
- Drücken Sie die [Strg]-Taste + [F4]-Taste, um die aktuell ausgewählte Anzeige zu schließen.
- Drücken Sie die [Esc]-Taste zum Überschreiben des Skripts oder zum Löschen und Beenden.

21.5 Kommunizieren mit nicht unterstützten peripheren Teilnehmern

ANMERKUNG

- Weitere Informationen hierzu entnehmen Sie bitte Ihrem Einstellungshandbuch.

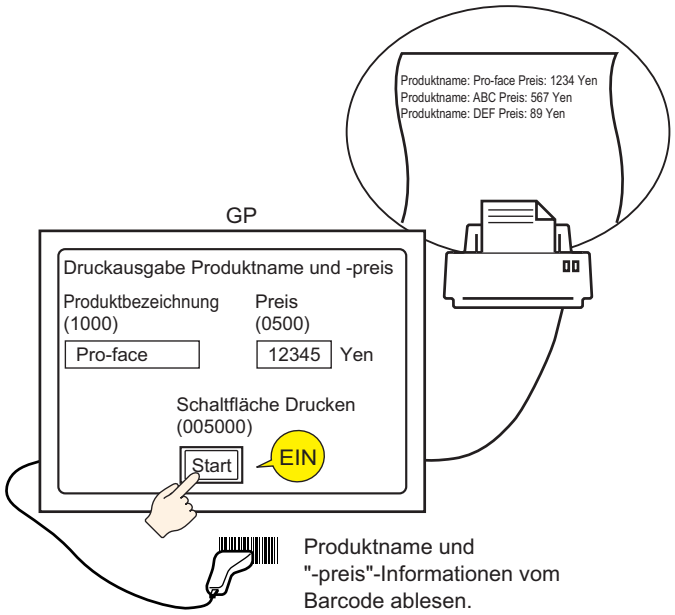
☞ "21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise" (seite 21-56)

- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

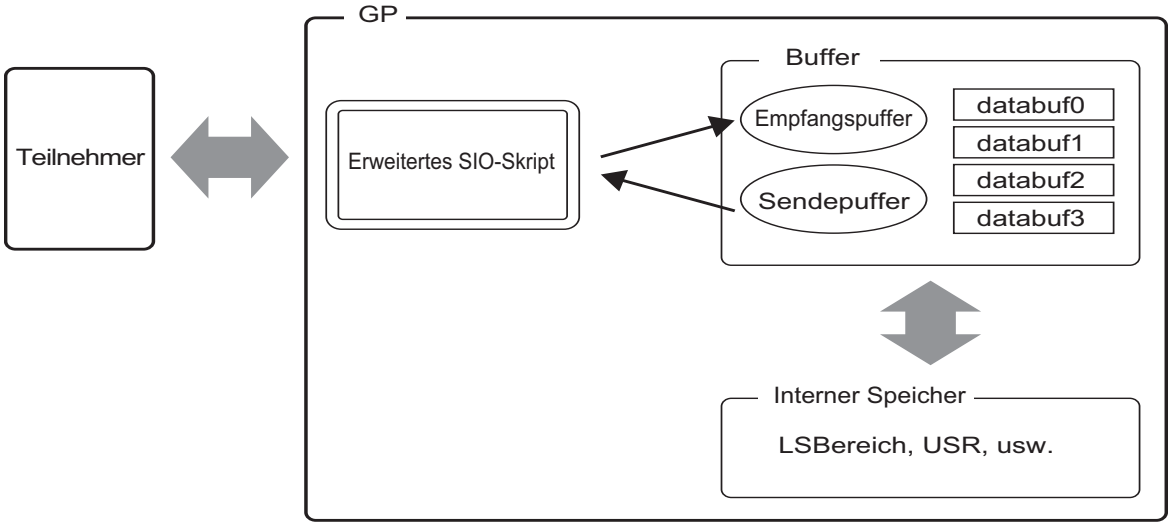
■ Laufzeiteinstellungen

Erstellen Sie ein erweitertes Skript, das Daten von einem Barcode-Leser ausliest, der mit dem USB-Port verbunden ist und an einen seriellen Drucker ausgibt, der mit COM1 verbunden ist.



■ Struktur erweiterter Skripts

Bei erweiterten Skripten handelt es sich um Spezialskripte, die zur Kommunikation zwischen dem internen seriellen Port der GP und den verbundenen Ein-/Ausgabe-Teilnehmern dienen. Wie in der folgenden Abbildung angezeigt, können für das erweiterte Skript-Datenmanagement die Daten in databuf0 bis databuf3 über den Puffer "Senden/Empfangen von Daten" gespeichert werden. Databuf wird nicht durch eine Adresse getrennt; wenn deshalb Daten vom Teilnehmer/von der SPS bearbeitet werden, speichern Sie diese bitte im internen Speicher, bevor diese Daten bearbeitet werden.



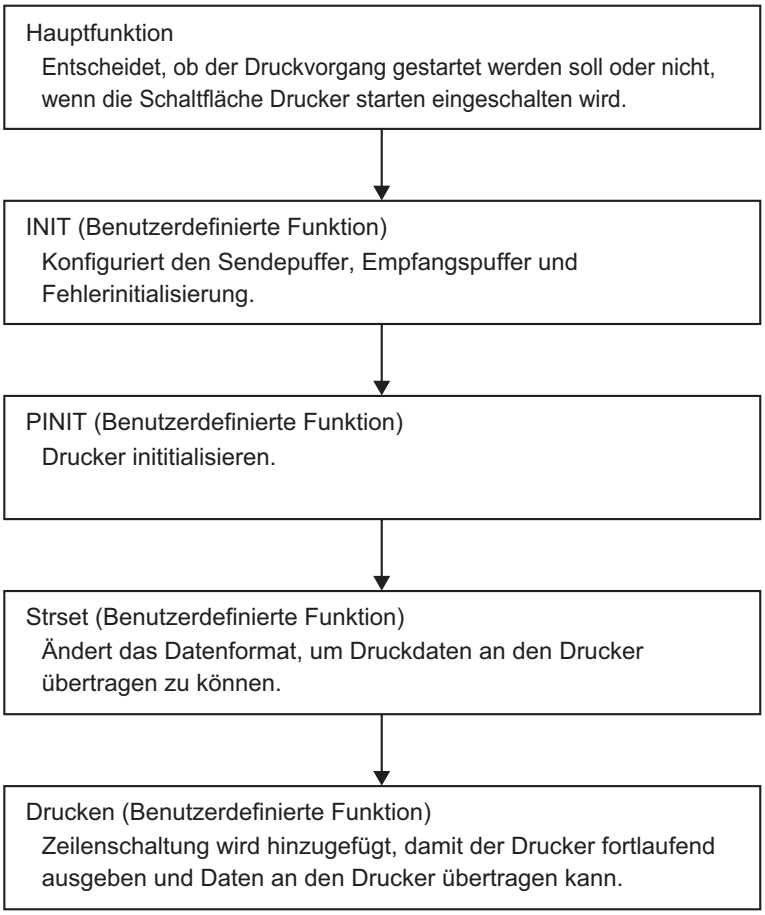
Puffer empfangen/Puffer senden

Bei der Kommunikation mit einem Teilnehmer/einer SPS fungiert der Puffer als Bit-Speicherplatz, der empfangene und gesandte Daten in Echtzeit unterscheidet.

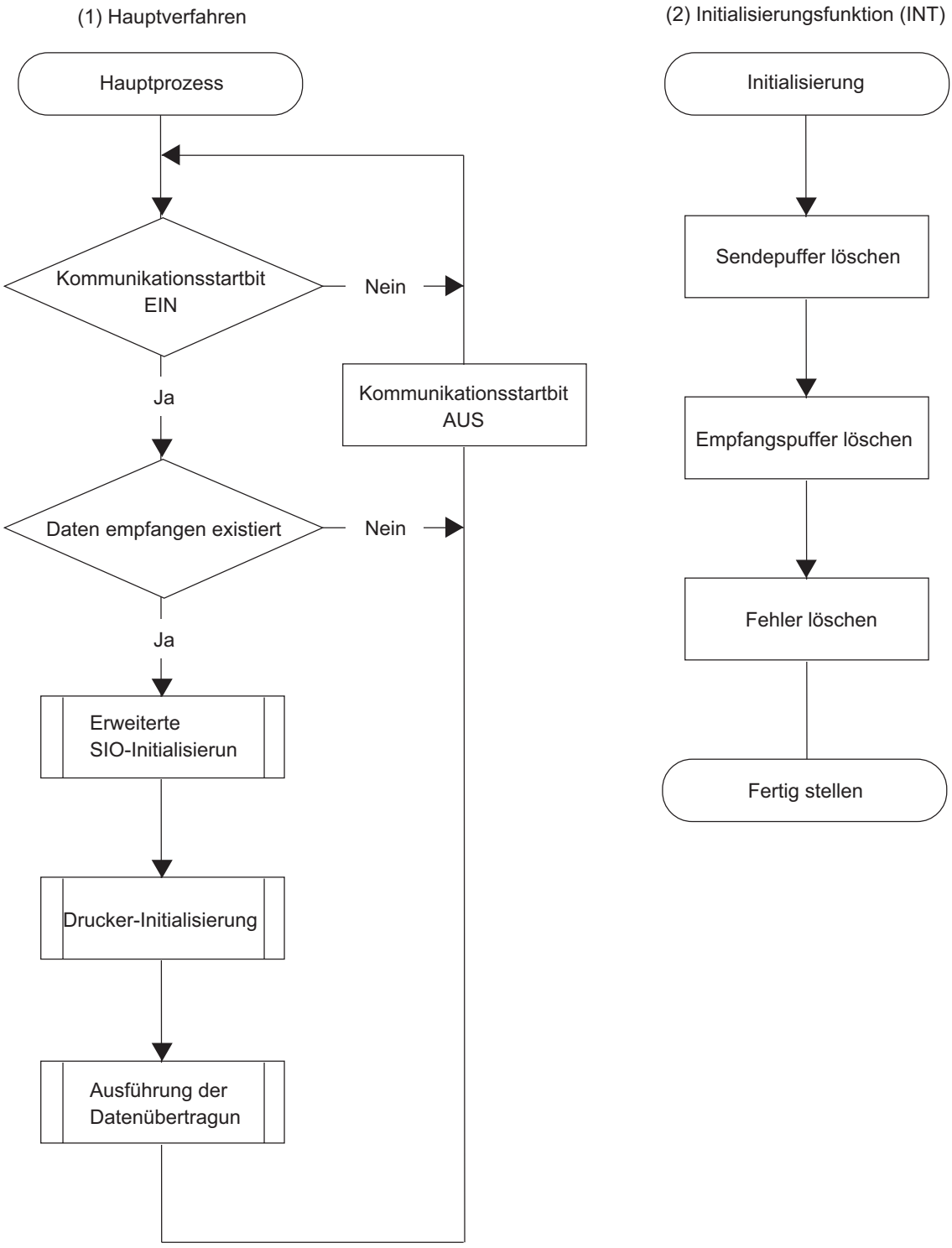
databuf0 bis databuf3

Hierbei handelt es sich um Byte (8-Bit)-Speicherstellen, die als Datenspeicherstellen fungieren. Die Größe eines Puffers beträgt 1 KB.

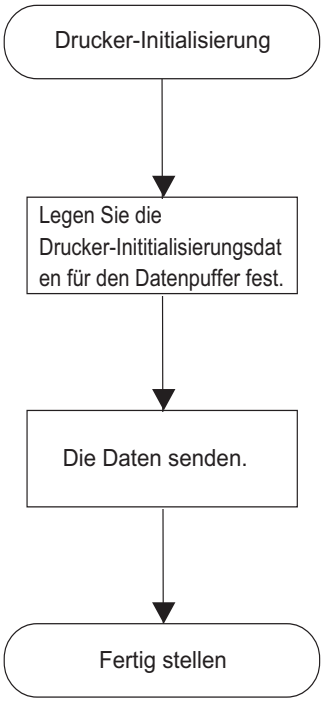
■ Verfahren zum Erstellen von Skripten



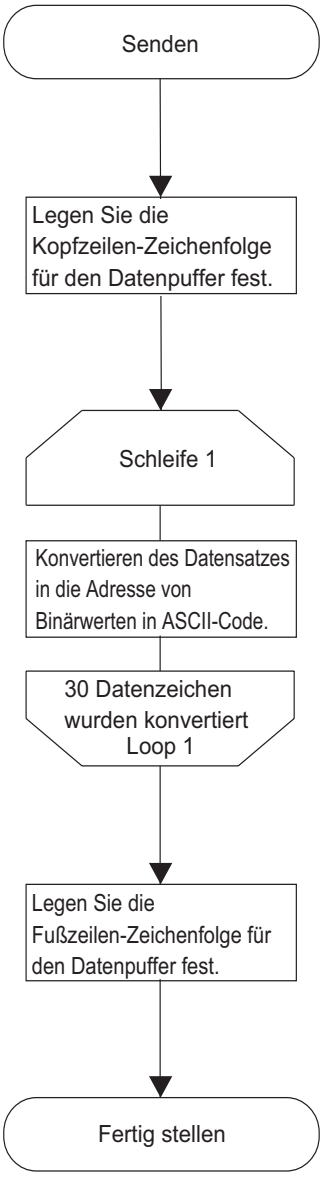
■ Ablaufdiagramm



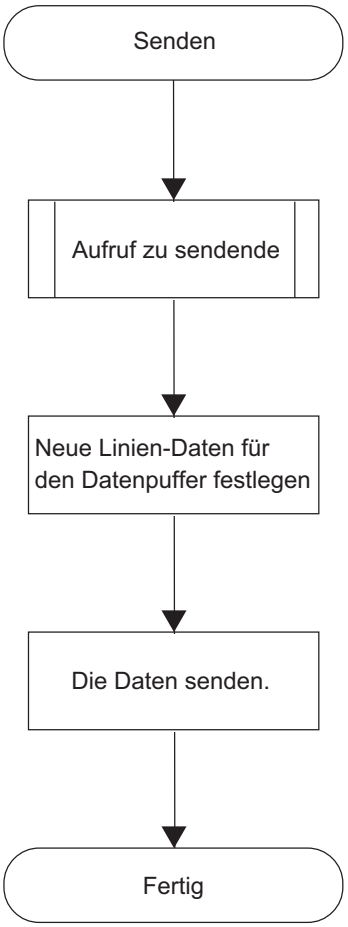
(3) Drucker-Initialisierungsfunktion (PINIT)



(4) Zeichenfolge-Funktion (Strset).



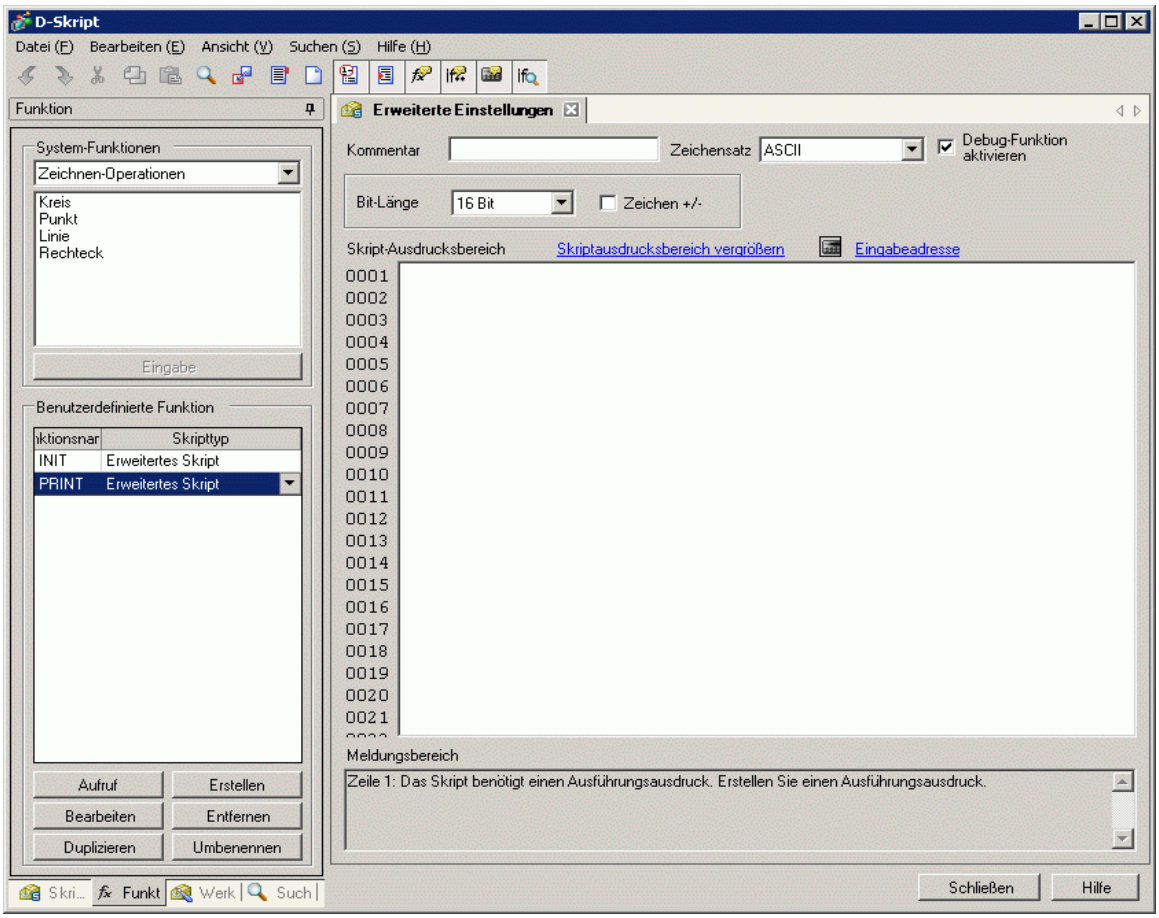
(5) Send-Funktion (Drucken)



■ Übersicht über Skriptoperationen

◆ Hauptfunktionen

Abgeschlossenes Skript



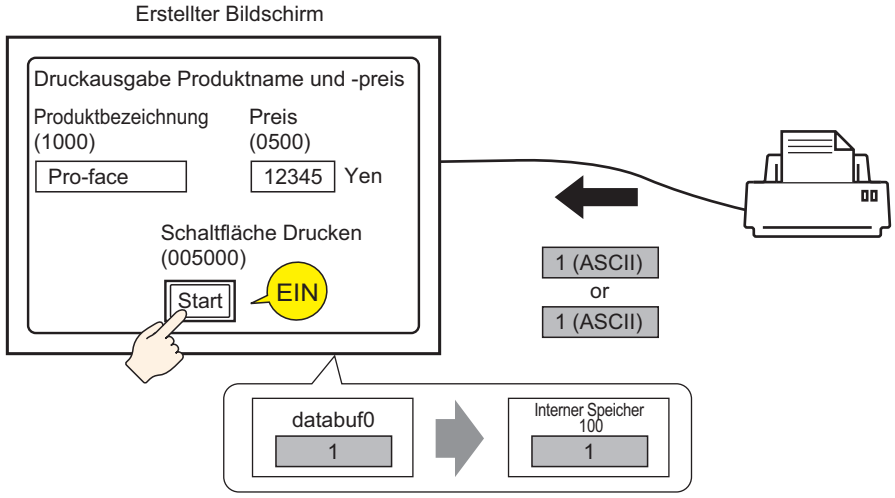
Funktionszusammenfassung

Wenn die Startschaltfläche des Druckers (interner Speicher 005000) eingeschalten wird, wird das Skript entscheiden, ob das Starten des Druckvorgangs vom 1. Byte der Druckerlaubnisdaten aus gestartet werden soll oder nicht.
Die Druckerlaubnisdaten werden die folgende Aktion als Beispiel für die Druckerspezifikationen durchführen.

Druckvorbereitung OK: 0x31 (ASCII -Code "1") an den Teilnehmer/die SPS senden.
Druckvorbereitung ungültig: 0x30 (ASCII -Code "0") an den Teilnehmer/die SPS senden.

Die GP empfängt die Druckerlaubnisdaten in databuf0 und diese Daten werden daraufhin in den leicht zugänglichen internen Speicher 100 mit der nachstehenden Skripthandhabung gespeichert.

Wenn der interne Speicher 100 = 0x31 (ASCII-Code für den Wert "1") beträgt, beginnt der Druckvorgang. Wenn der interne Speicher 0x30 (ASCII-Code für "0") beträgt, kehrt die GP zum Anfang zurück und der Prozeß wird wiederholt, bis die Daten 0x31 empfangen wurden.



◆ INIT (Benutzerdefinierte Funktion)

Abgeschlossenes Skript

Skriptausdruck Skriptbereich vergrößern Adresseingabe

```
0001 [c:EXT_SIO_CTRL00]=1    //Clears send buffer.
0002 [c:EXT_SIO_CTRL01]=1    //Clears receive buffer.
0003 [c:EXT_SIO_CTRL02]=1    //Clears error.
0004
```

Funktionszusammenfassung

Konfiguriert den Sendepuffer, Empfangspuffer und die Fehlerinitialisierung.

◆ PINIT (Benutzerdefinierte Funktion)

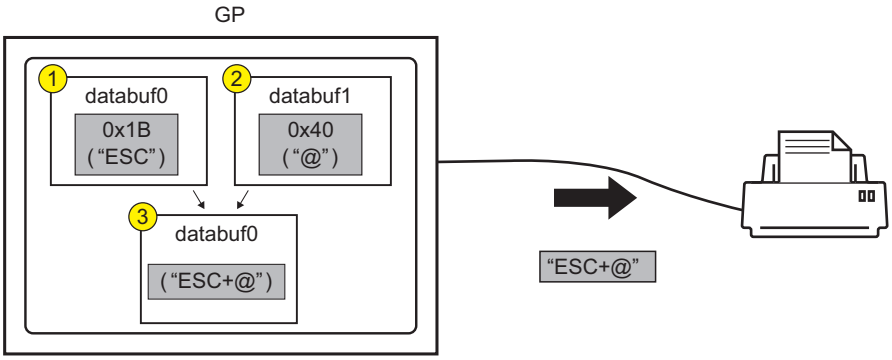
Abgeschlossenes Skript

Skriptausdruck Skriptbereich vergrößern Adresseingabe

```
0001 //Printer initialization ( ESC/P 「ESC + @」 )
0002
0003 _strset(databuf0, "")           //Clears data buffer 0.
0004 _strset(databuf0, 0x1B)         //Sets ASCII code for "ESC".
0005 _strset(databuf1, "")           //Clears data buffer 1.
0006 _strset(databuf1, 0x40)         //Sets ASCII code for "@".
0007 _strcat(databuf0, databuf1)     //Concatenates data buffer 1 to end of data buffer 0.
0008 _strlen([t:0000], databuf0)     //Converts data length to numerical value
0009                                 //and stores it in temporary address.
0010
0011 //Sends data from serial port.
0012
0013 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000])    //Sends amount of buffer 0 data specified by value
0014                                                  //of temporary address.
0015
```

Funktionszusammenfassung

Drucker inititialisieren. Senden Sie den ESC/P-Befehl "ESC+@" an den Drucker.



◆ Strset (Benutzerdefinierte Funktion)

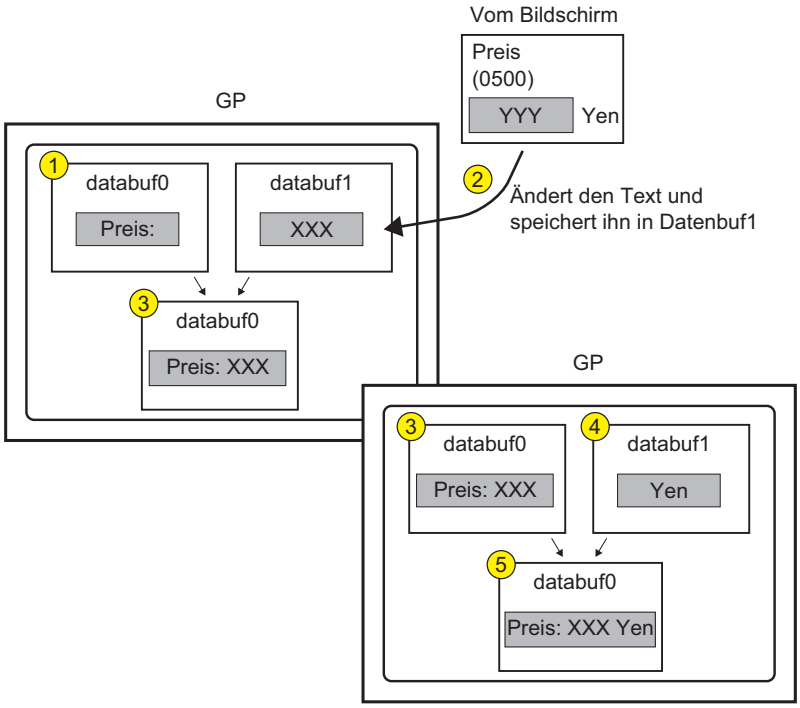
Abgeschlossenes Skript

Skriptausdruck Skriptbereich vergrößern Adresseingabe

```
0001 //Appends "Price:" and "Yen" text strings.
0002 _strset(databuf0, "")                          //Initializes data buffer 0.
0003 _strset(databuf0, "価格 : ")                    //"Price:" Stores text string in data buffer 0.
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])      //Converts numerical value to text string
0005                                                //and stores it in data buffer 1.
0006 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0007 _strset(databuf1, "")                          //Initializes data buffer 1.
0008 _strset(databuf1, "円")                         //"Yen" Stores text string in data buffer 1.
0009 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0010
0011 //Temporary address initialization
0012 [t:0001]=0
0013 [t:0002]=0
0014
0015 //Re-stores text string stored sequentially in word units in internal memory, in byte units (30 characters of data).
0016 loop()
0017 {
0018     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8       //Stores higher-order byte in place of lower-order byte.
0019     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF     //Erases higher-order byte and stores result
0020                                                                        //in next address.
0021     [t:0001]=[t:0001]+1                                                //Address offset + 1
0022     [t:0002]=[t:0002]+2                                                //Address offset + 2
0023     if([t:0001]==15)                                                   //Loop ends when process of storing each 2-byte value
0024     {                                                                  //in 2 words has repeated 15 times.
0025         break
0026     }
0027     endif
0028 }
0029 endloop
0030 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)      //Stores data in internal memory addresses 2000 to 2030 in data buffer
0031                                                //as text string.
0032 //Appends "Product Name:" text string.
0033 _strset(databuf1, "")                          //Initializes data buffer 1.
0034 _strset(databuf1, "品名 : ")                    //"Product Name:" Stores text string in data buffer 1.
0035 _strcat(databuf1, databuf2)                    //Concatenates data buffer 2 to end of data buffer 1.
0036
0037 //Appends price character string to product name.
0038 _strcat(databuf1, databuf0)                    //Concatenates value of data buffer 0 to data buffer 1.
```

Funktionszusammenfassung

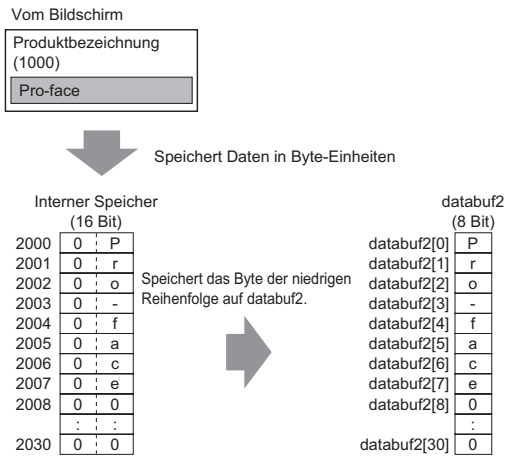
1 Den Text "Preis:" und "Yen" an die Preisdaten hinzufügen, die in der internen Speicheradresse 0500 gespeichert sind.



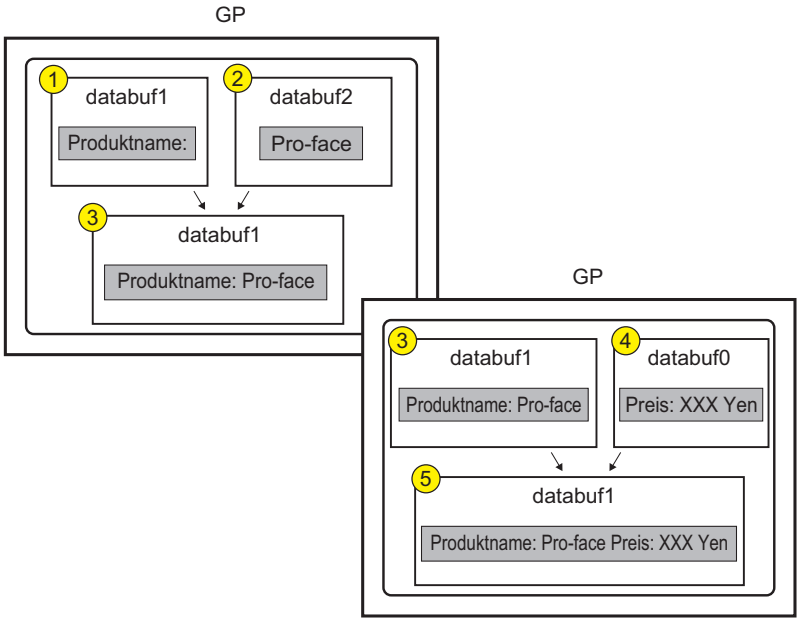
2 Ändern Sie das Datenformat, um Druckdaten an den Drucker übertragen zu können. Teilen Sie die Zeichenfolgedaten (Produktname) in Byte-Einheiten auf, die nacheinander im internen Speicher 1000 gespeichert sind und speichern Sie diese im internen Speicher 2000 bis 2030 als Byte-Zeichenfolgedaten niedriger Reihenfolge. Verwenden Sie die Funktion _ldcopy und speichern Sie die Daten in databuf2 in der Reihenfolge der aufeinanderfolgenden niedrigsten Bytes der Wortadressen.

ANMERKUNG

- Die _ldcopy-Funktion nimmt die als Worte gespeicherten Daten und speichert nur die Bytes niedriger Reihenfolge im Puffer, während die Bytedaten höherer Reihenfolge ignoriert werden.



3 Den Text "Produktname:" und "Preis" an databuf2 hinzufügen.



◆ Drucken (Benutzerdefinierte Funktion)

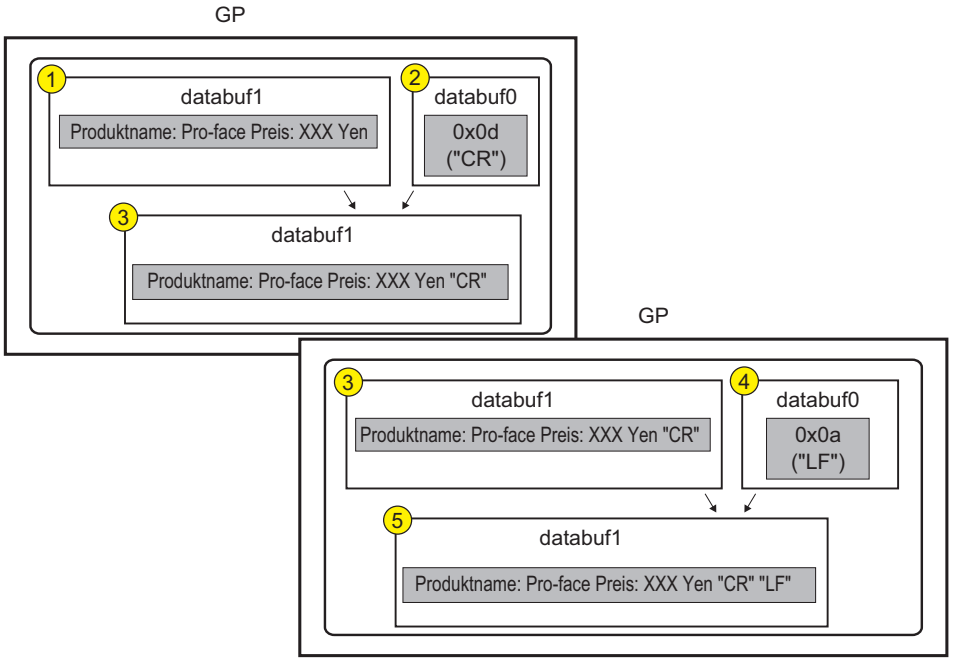
Abgeschlossenes Skript

Skriptausdruck Skriptbereich vergrößern Adresseingabe

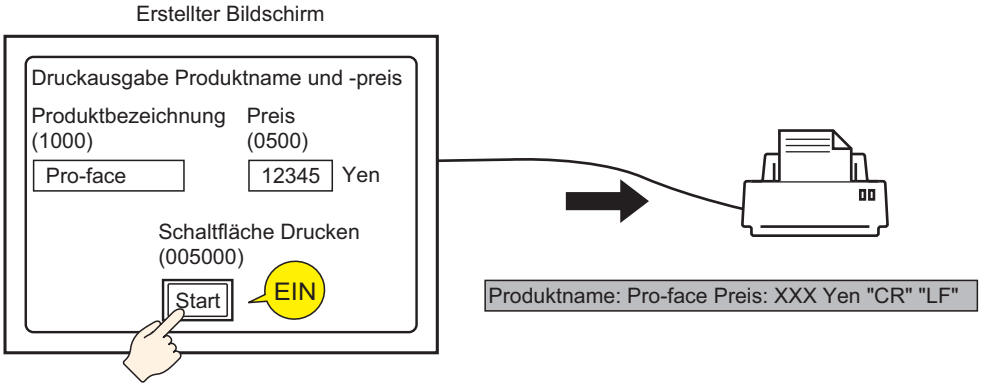
```
0001 Call Strset                          //Calls print data function.
0002 _strset(databuf0, "")                //Clears data buffer 1.
0003
0004 //Printing and delimiter (line feed + carriage return)
0005
0006 _strset(databuf0, 0x0d)              //Prints and returns to head of line.
0007 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0008 _strset(databuf0, "")                //Clears data buffer 1.
0009 _strset(databuf0, 0x0a)              //Sends line feed to move to next line.
0010 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0011
0012 _strlen([t:0000], databuf1)          //Converts data length to numerical value
0013                                      //and stores it in temporary address.
0014 //Sends data from serial port.
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])    //Sends amount of buffer 0 data
0017                                                 //specified by value of temporary address.
0018
```

Funktionszusammenfassung

1 Fügen Sie eine "Zeilenschaltung" hinzu, damit der Drucker fortlaufend ausführen kann.



2 Legen Sie die Druckerdaten für den Drucker fest.

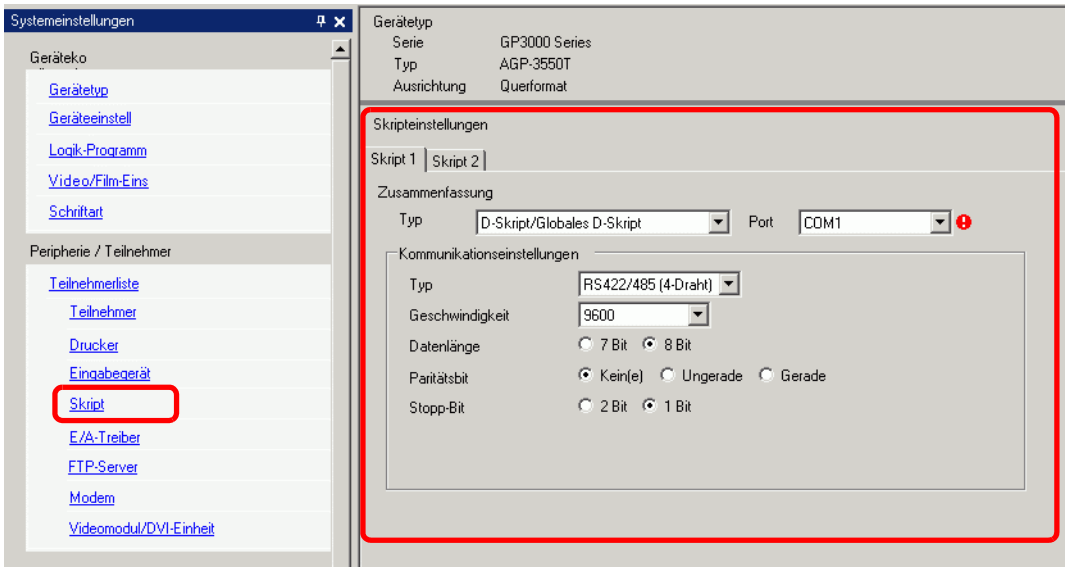


■ Benutzte Befehle

Befehl	Funktionszusammenfassung
if ( )	Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt. ☞ "21.11.8 Bedingte Ausdrücke" (seite 21-156)
Beschriftungsvariablen [r:EXT_SIO_RECV]	Zeigt die zu diesem Zeitpunkt empfangene Datenmenge (Anzahl der Bytes) an. Die empfangene Datengröße ist schreibgeschützt. ☞ "21.11.4 SIO-Port-Operation" (seite 21-104)
Gleich (==)	Ja, wenn N1 gleich hoch wie N2 ist (N1 = N2). ☞ "21.11.9 Vergleich" (seite 21-161)
Setze Datenpuffer (_strset)	Im Datenpuffer ist ein fester String gespeichert. ☞ "21.11.11 String-Operation" (seite 21-167)
Erweiterter Empfang (IO_READ_EX)	Empfängt Daten der in der Empfangsdatengröße festgesetzten Größe (Bytes) vom erweiterten SIO und speichert sie im Datenpuffer. ☞ "21.11.4 SIO-Port-Operation" (seite 21-104)
Kopiere Datenpuffer in interne Adressen (_dlcopy)	Jeder im Offset des Datenpuffers gespeicherte Zeichenfolge-Datenbyte wird gemäß der Anzahl der Zeichenfolgen in den LS-Bereich kopiert. ☞ "21.11.11 String-Operation" (seite 21-167)
Steuerungsvariablen [c:EXT_SIO_CTRL**]**	Diese Steuerelementvariable wird zum Löschen des Sendepuffers, des Empfangspuffers und des Fehlerstatuses verwendet. ☞ "21.11.8 Bedingte Ausdrücke" (seite 21-156)
Strings verketten (_strcat)	Eine Zeichenkette oder ein Zeichencode ist mit dem Textpuffer verkettet. ☞ "21.11.11 String-Operation" (seite 21-167)
Textlänge (_strlen)	Besorgt die Länge des gespeicherten Strings. ☞ "21.11.11 String-Operation" (seite 21-167)
Erweiterte Sendung (IO_WRITE_EX)	Sendet die Daten im Datenpuffer unter Anwendung des erweiterten SIO und gemäß der Anzahl der Sendebytes. ☞ "21.11.4 SIO-Port-Operation" (seite 21-104)
Zuweisung (=)	Weisen Sie den Wert auf der rechten Seite dem Wert auf der linken Seite zu. ☞ "21.11.10 Operator" (seite 21-163)
Addition (+)	Fügt den Daten eines Wortteilnehmers eine Konstante hinzu. ☞ "21.11.10 Operator" (seite 21-163)
Konvertierung Numerischer Wert in Dez-Zeichenfolge (_bin2decasc)	Mit dieser Funktion wird eine Ganzzahl in eine Dezimal-Zeichenfolge konvertiert. ☞ "21.11.11 String-Operation" (seite 21-167)
Kopiere interne Adressen in Datenpuffer (_ldcopy)	Die im LS-Bereich gespeicherten String-Daten werden gemäß der Anzahl der Strings byteweise in den Datenpuffer übertragen. ☞ "21.11.11 String-Operation" (seite 21-167)

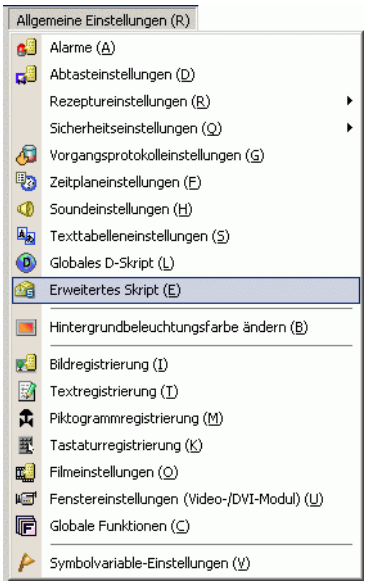
■ Erstellungsverfahren.

1 Bestimmen Sie die Skript-Einstellungen, um Erweiterte Skripts zum Kommunizieren verwenden zu können. Wählen Sie im Menü [Projekt (F)] [Systemeinstellungen (C)] aus. Wählen Sie [Skript]. Legen Sie den [Typ] auf [Erweitertes Skript] fest.



Für die Skripteinstellungen gibt es zwei Register. Stellen Sie den [Port] auf COM1 oder COM2.
Die [Kommunikationseinstellungen] übereinstimmend mit dem erweiterten SIO ein.

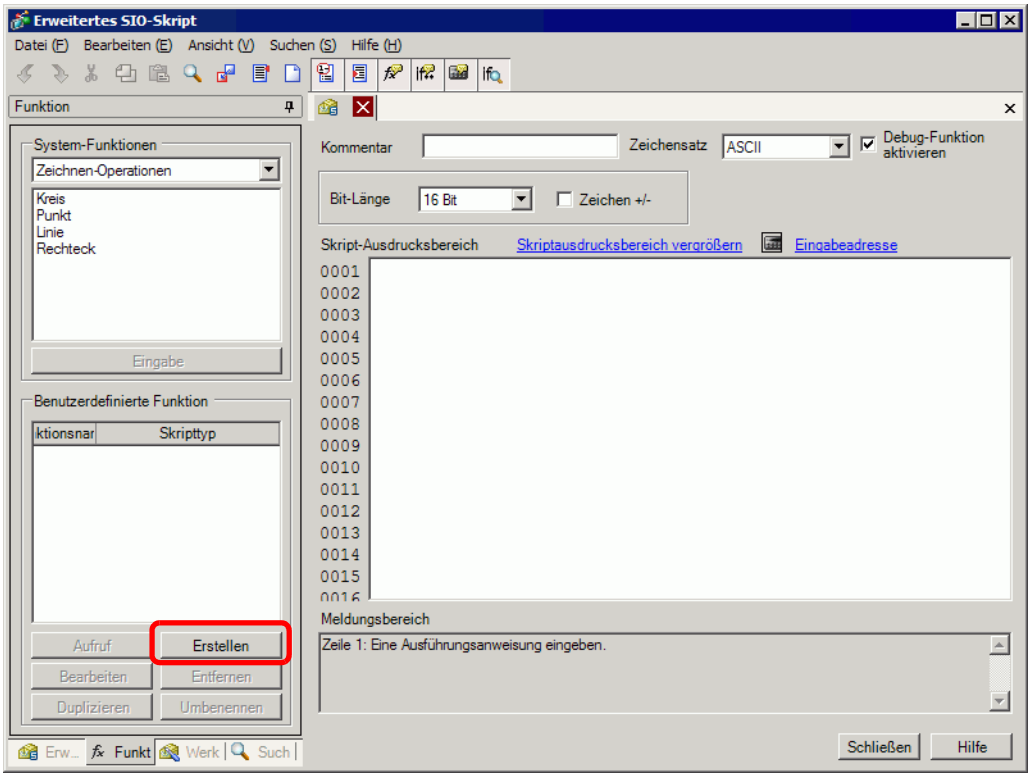
2 Wählen Sie im Menü [Allgemeine Einstellungen (R)] [Erweitertes Skript (E)] aus.



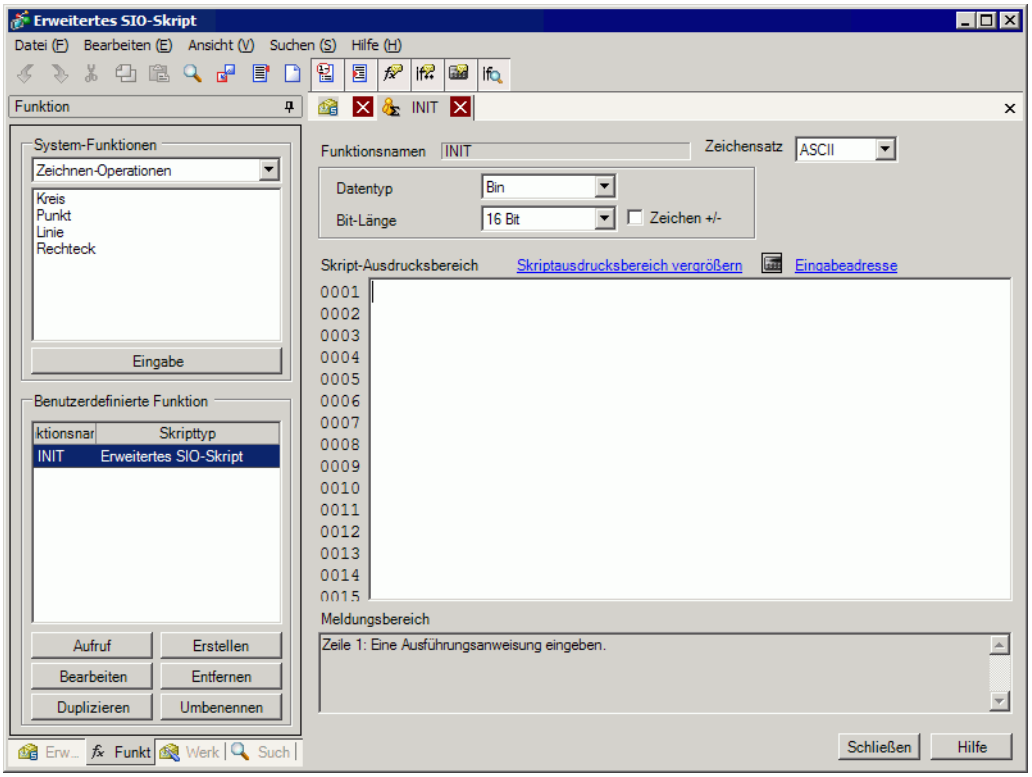
ANMERKUNG

- Wenn die Nachricht "Das erweiterte Skript wird verwendet. Forfahren?" angezeigt wird, klicken Sie auf [Ja].

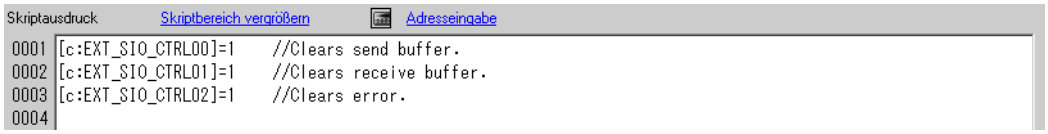
3 Registrieren Sie "INIT" als benutzerdefinierte Funktion. Klicken Sie auf die Registerkarte [Funktion] und auf die Schaltfläche [Erstellen] des benutzerdefinierten Rahmens.



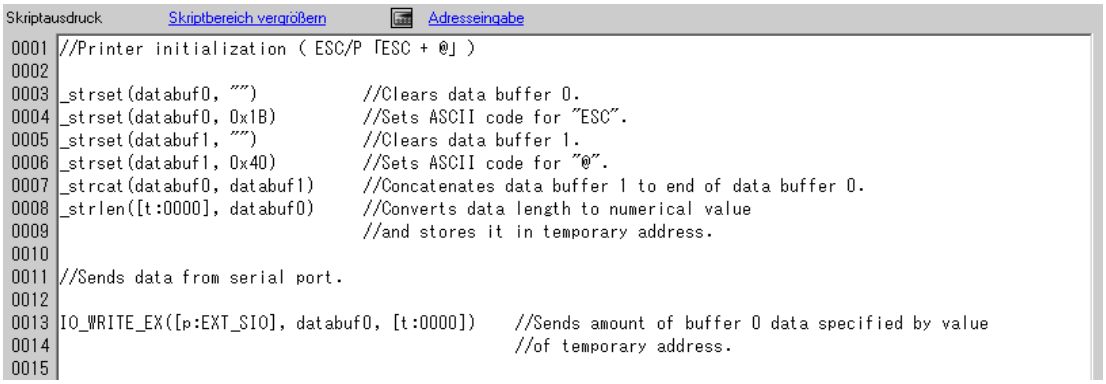
4 Geben Sie [INIT] als Funktionsname ein, klicken auf [OK]. Der folgende Bildschirm wird angezeigt.



5 Erstellen Sie ein Skript im Skriptausdruck mit Befehlen, Anweisungen und konstanter Eingabe.



6 Registrieren Sie ebenfalls "PINIT" als benutzerdefinierte Funktion. Geben Sie [PINIT] als Funktionsnamen ein und erstellen Sie das folgende Skript im Skriptausdruck.



7 Registrieren Sie ebenfalls "Strset" als benutzerdefinierte Funktion. Geben Sie [Strset] als Funktionsnamen ein und erstellen Sie das folgende Skript im Skriptausdruck.

Skriptausdruck | Skriptbereich vergrößern | Adresseingabe

```
0001 //Appends "Price:" and "Yen" text strings.
0002 _strset(databuf0, "")                          //Initializes data buffer 0.
0003 _strset(databuf0, "価格 : ")                    //"Price:" Stores text string in data buffer 0.
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])      //Converts numerical value to text string
0005                                                //and stores it in data buffer 1.
0006 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0007 _strset(databuf1, "")                          //Initializes data buffer 1.
0008 _strset(databuf1, "円")                         //"Yen" Stores text string in data buffer 1.
0009 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0010
0011 //Temporary address initialization
0012 [t:0001]=0
0013 [t:0002]=0
0014
0015 //Re-stores text string stored sequentially in word units in internal memory, in byte units (30 characters of data).
0016 loop()
0017 {
0018     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8       //Stores higher-order byte in place of lower-order byte.
0019     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF     //Erases higher-order byte and stores result
0020                                                                        //in next address.
0021     [t:0001]=[t:0001]+1                                                //Address offset + 1
0022     [t:0002]=[t:0002]+2                                                //Address offset + 2
0023     if([t:0001]==15)                                                   //Loop ends when process of storing each 2-byte value
0024     {                                                                  //in 2 words has repeated 15 times.
0025         break
0026     }
0027     endif
0028 }
0029 endloop
0030 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)      //Stores data in internal memory addresses 2000 to 2030 in data buffer
0031                                                //as text string.
0032 //Appends "Product Name:" text string.
0033 _strset(databuf1, "")                          //Initializes data buffer 1.
0034 _strset(databuf1, "品名 : ")                    //"Product Name:" Stores text string in data buffer 1.
0035 _strcat(databuf1, databuf2)                    //Concatenates data buffer 2 to end of data buffer 1.
0036
0037 //Appends price character string to product name.
0038 _strcat(databuf1, databuf0)                    //Concatenates value of data buffer 0 to data buffer 1.
```

1

2

3

8 Registrieren Sie ebenfalls "Drucken" als benutzerdefinierte Funktion. Geben Sie [Drucken] als Funktionsnamen ein und erstellen Sie das folgende Skript im Skriptausdruck.

Skriptausdruck | Skriptbereich vergrößern | Adresseingabe

```
0001 Call Strset                        //Calls print data function.
0002 _strset(databuf0, "")              //Clears data buffer 1.
0003
0004 //Printing and delimiter (line feed + carriage return)
0005
0006 _strset(databuf0, 0x0d)            //Prints and returns to head of line.
0007 _strcat(databuf1, databuf0)        //Concatenates data buffer 1 to end of data buffer 0.
0008 _strset(databuf0, "")              //Clears data buffer 1.
0009 _strset(databuf0, 0x0a)            //Sends line feed to move to next line.
0010 _strcat(databuf1, databuf0)        //Concatenates data buffer 1 to end of data buffer 0.
0011
0012 _strlen([t:0000], databuf1)        //Converts data length to numerical value
0013                                    //and stores it in temporary address.
0014 //Sends data from serial port.
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])    //Sends amount of buffer 0 data
0017                                                 //specified by value of temporary address.
```

9 Erstellen Sie eine Hauptskript. Erstellen Sie im Skriptausdruck das folgende Skript und die Einstellungen sind daraufhin abgeschlossen.

Skriptausdruck Skriptbereich vergrößern Adresseingabe

```
0001 //Receives 1-byte Print Permit data from printer.
0002 if([r:EXT_SIO_RECV]==1)                                    //When number of received
0003 {                                                          //and stored data items is 1
0004     _strset(databuf0, "")                                  //Initializes data buffer 0.
0005     IO_READ_EX([p:EXT_SIO], databuf0, 1)                   //Reads data into data buffer 0.
0006     _dlcopy([w:[#MEMLINK]0100], databuf0, 0, 1)            //Stores value from data buffer 0
0007 }                                                          //in internal memory.
0008 endif
0009
0010 //Determines whether to start printing from Print Permit data.
0011 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x31)    //When printer start switch is ON
0012 {                                                          //and "ACK" function data is 1 (ASCII)
0013     Call INIT                                              //Calls communication initialization function.
0014     Call PINIT                                             //Calls printer initialization function.
0015     Call PRINT                                             //Sends print data,
0016                                                            //calls printer start function.
0017     clear([b:[#MEMLINK]005000])                            //Printer start switch OFF.
0018 }
0019 endif
0020
0021 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x30)    //When printer start switch is ON
0022 {                                                          //and "ACK" function data is 0 (ASCII)
0023     clear([b:[#MEMLINK]005000])                            //Printer start switch OFF.
0024 }
0025 endif
```

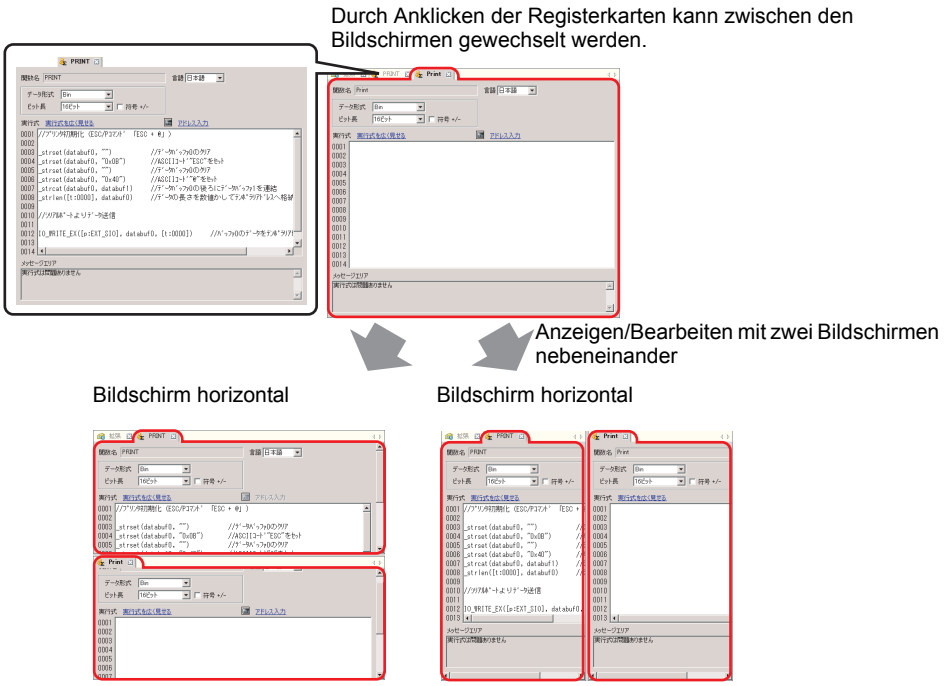
ANMERKUNG

- Wenn die in den Schritten 3 bis 9 erstellten benutzerdefinierten Funktionen in das Hauptskript abgelegt werden, wählen Sie die abzulegende Funktion aus und klicken auf [Aufrufen] in der Registerkarte [Funktion]. Die Funktion wird mittels "Funktionsname aufrufen" abgelegt.
- Wenn Sie Text auswählen möchten, drücken Sie die [Strg]-Taste + die [Umschalt]-Taste + die [Rechte Pfeiltaste]/[Linke Pfeiltaste], um einen gesamten Bock eines Textes auszuwählen.
- Drücken Sie die [Strg]-Taste + [F4]-Taste, um die aktuell ausgewählte Anzeige zu schließen.
- Drücken Sie die [Esc]-Taste zum Überschreiben des Skripts oder zum Löschen und Beenden.

21.6 Referenzieren anderer Skripts

21.6.1 Einleitung

Sie können eine benutzer-definierte Funktion neben einem D-Skript/Globalen D-Skript, Erweiterten Skriptn oder einer anderen benutzerdefinierten Funktion anzeigen.
Sie können die Funktion während des Vergleichs schreiben oder beide gleichzeitig bearbeiten.



ANMERKUNG

- Sie können auch verschiedene Typen benutzerdefinierter Funktion gleichzeitig bearbeiten.

21.6.2 Durchführungsverfahren

ANMERKUNG

- Weitere Informationen hierzu entnehmen Sie bitte Ihrem Einstellungshandbuch.

☞ "21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise" (seite 21-56)

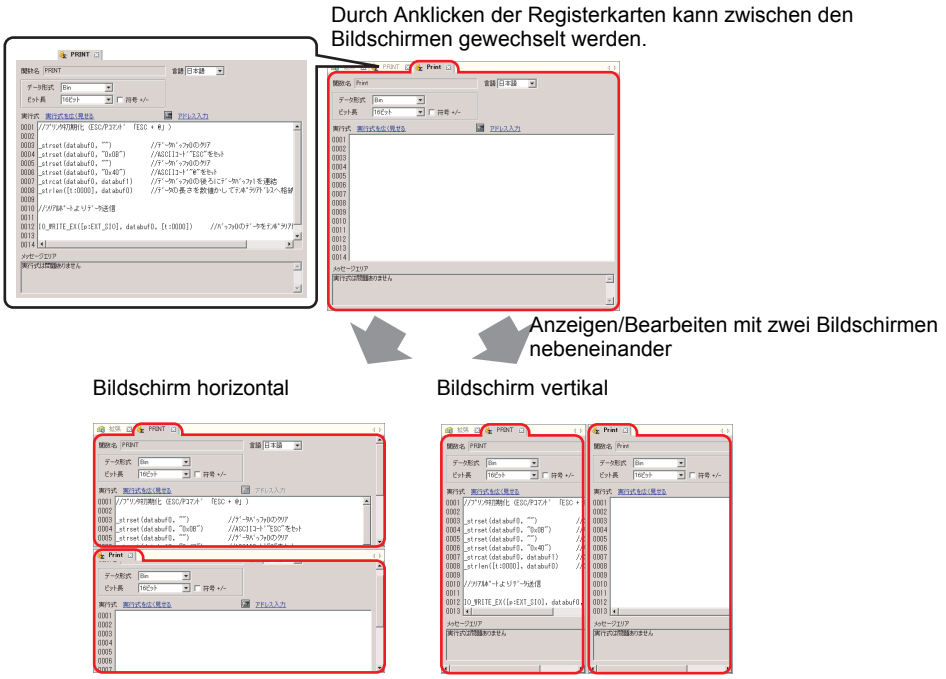
- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

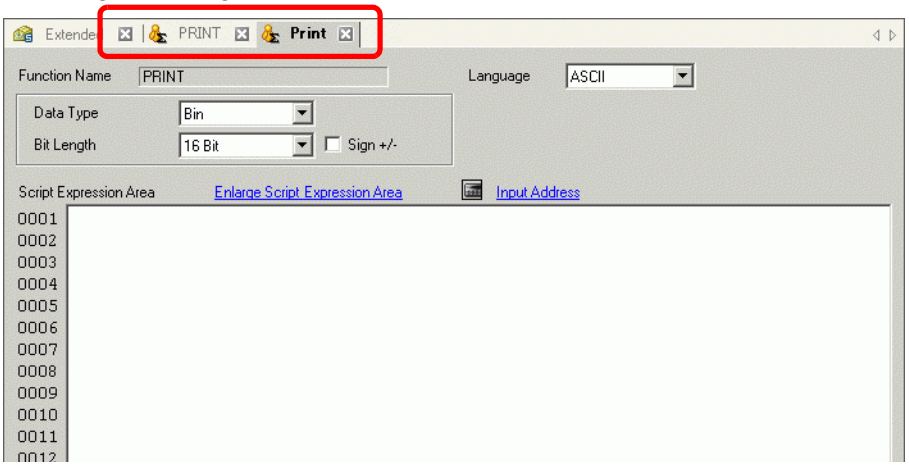
- Weitere Informationen über Befehle, die für Skripte zur Verfügung stehen, finden Sie nachstehend.

☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74)

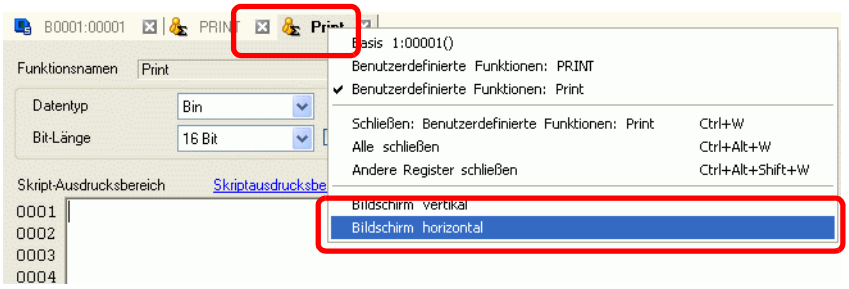
Teilen Sie im Dialogfeld [D-Skript] den Bildschirm horizontal oder vertikal in zwei Bildschirme.
Beispiel: Erstellen Sie "Druck", während die benutzerdefinierte Funktion "DRUCK" aus einem vorherigen Verfahren angezeigt wird.



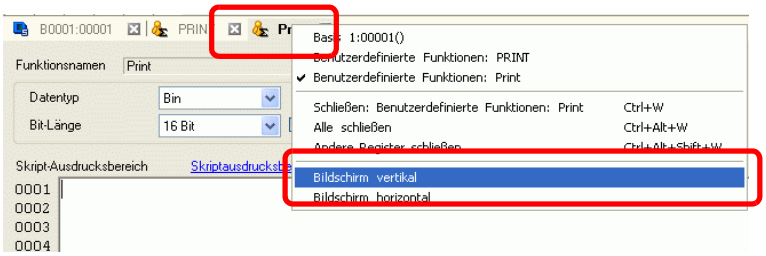
1 Öffnen Sie im Dialogfeld [D-Skript] das Skript und die benutzerdefinierte Funktion, die Sie anzeigen möchten, gleichzeitig.



2 Wenn Sie horizontal anordnen möchten, klicken Sie mit der rechten Maustaste auf die Bildschirmregisterkarte zur Anzeige am Ende, und klicken Sie dann auf [Bildschirm horizontal].

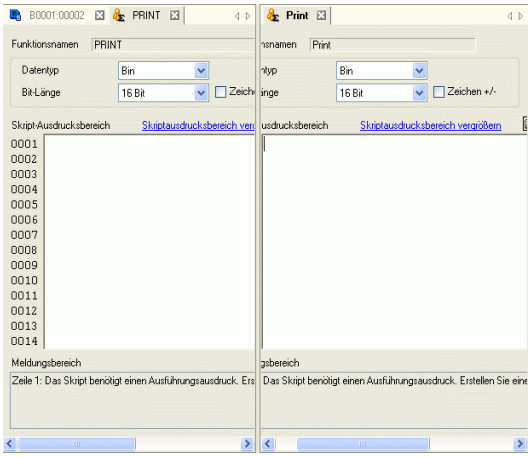


Wenn Sie vertikal anordnen möchten, klicken Sie mit der rechten Maustaste auf die Bildschirmregisterkarte zur Anzeige rechts, und klicken Sie dann auf [Bildschirm vertikal].

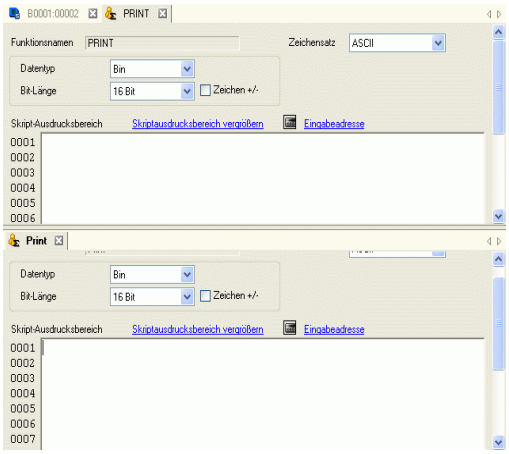


3 Die Bildschirme werden horizontal byw. vertikal angeordnet.

Bildschirm horizontal



Bildschirm vertikal

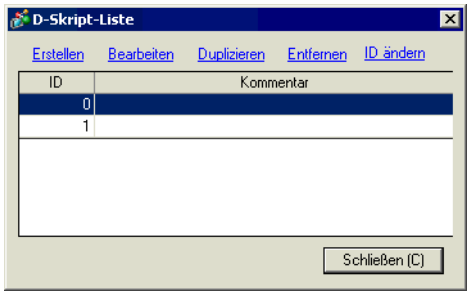


21.7 Erstellen von Skripts

21.7.1 Verfahren zum Erstellen von D-Skripten/Globalen D-Skripten

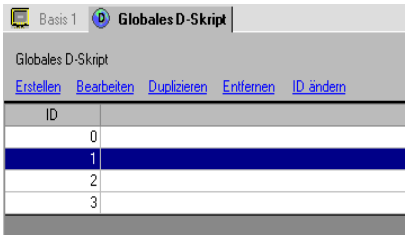
Wählen Sie im Menü [Element (P)] [D-Skript (R)] aus.

Klicken Sie auf [Erstellen]. Wenn auf ein vorher registriertes Skript zugegriffen wird, müssen Sie die ID-Nr. eingeben und auf [Bearbeiten] klicken oder die Zeile der ID-Nr. doppelt anklicken.

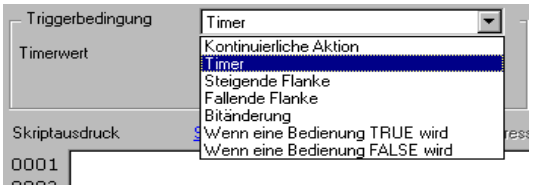


Wählen Sie im Menü [Allgemeine Einstellungen (R)] [Globales D-Skript (L)] aus.

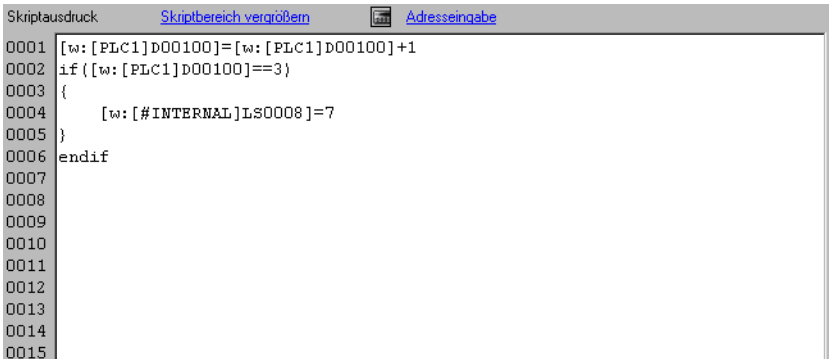
Klicken Sie auf [Erstellen]. Wenn auf ein vorher registriertes Skript zugegriffen wird, müssen Sie die ID-Nr. eingeben und auf [Bearbeiten] klicken oder die Zeile der ID-Nr. doppelt anklicken.



Legen Sie die Triggerbedingung fest, die das Skript zum Ausführen bringt. Weitere Informationen über diese Funktion erfahren Sie unter "21.8 Triggerbedingung - Einrichtung" (seite 21-49) .



Erstellen Sie das Skript (Skriptausdruck). Weitere Informationen zu Befehlen und Funktionen erfahren Sie unter "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74) .

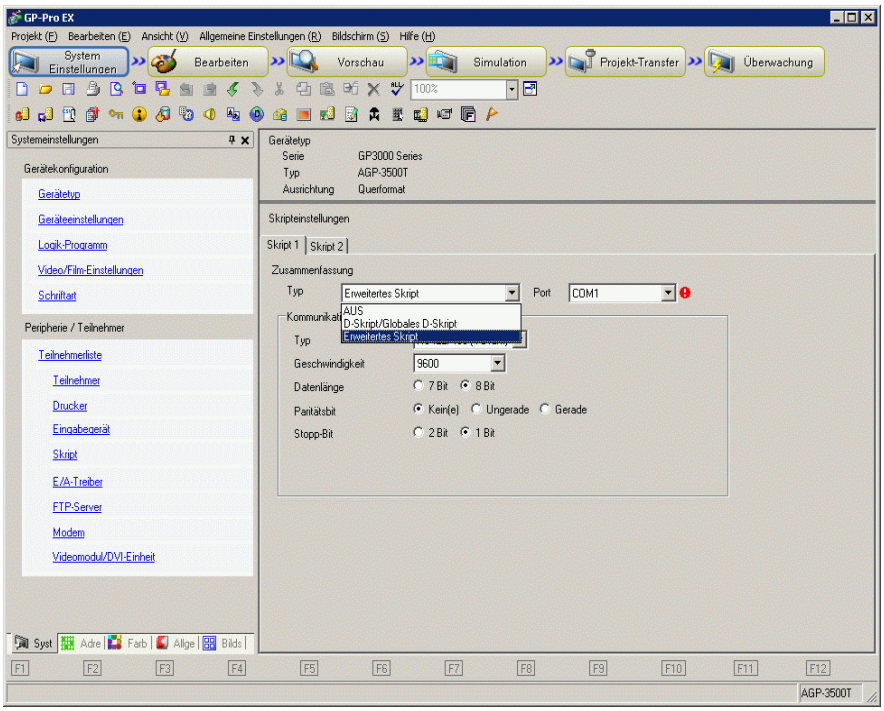


ANMERKUNG

- Die Komponente-Ablage zeigt registrierte D-Skripts in der erstellten Reihenfolge an. Zum Ändern der Reihenfolge in der Komponentenablage, muss die ID-Nummer für die registrierten Elemente geändert und dann vom Menü [Bearbeiten] der Befehl [Skript-Liste ausrichten] gewählt werden. Sie können die ID-Einstellungen durch Doppelklicken auf die Elemente in der Komponenten-Ablage zur Anzeige des Dialogfensters "Bearbeiten" ändern.

21.7.2 Verfahren zum Erstellen von Erweiterten Skripten

Wählen Sie im Menü [Projekt (F)] [Systemeinstellungen (C)] aus. Klicken Sie auf [Skripteinstellungen], um das folgende Dialogfenster anzuzeigen.
Wenn ein erweitertes Skript verwendet wird, wählen Sie [Typ] als [Erweitertes Skript] aus und bestimmen den entsprechenden [Port].

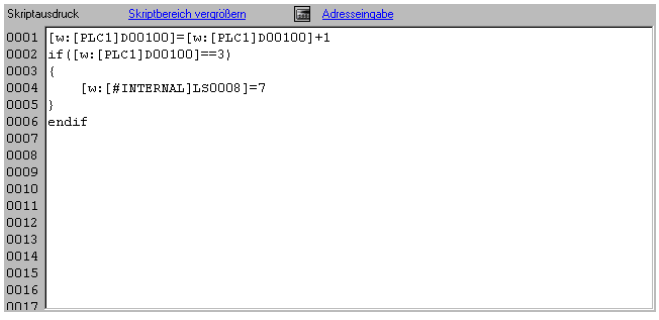


↓

Wählen Sie im Menü [Allgemeine Einstellungen (R)] [Erweitertes Skript (E)] aus.

↓

Erstellen Sie das Skript (Skriptausdruck). Weitere Informationen zu Befehlen und Funktionen erfahren Sie unter "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74) .



21.7.3 Einrichten benutzerdefinierter Funktionen

Registrieren Sie ein bestehendes Skript als benutzerdefinierte Funktion, woraufhin es von anderen Skripten benutzt werden kann. Die registrierte Funktion kann in einem D-Skript/ Globales D-Skript oder Erweitertes Skript verwendet werden.

■ Einrichtungsverfahren

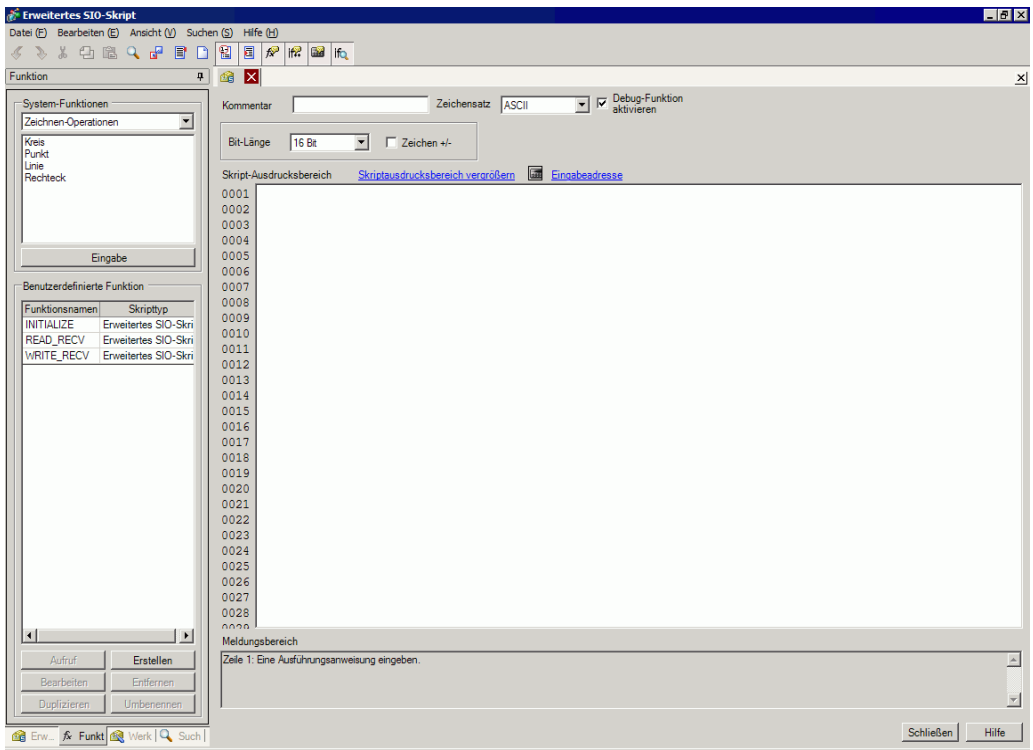
Wenn eine neue benutzerdefinierte Funktion erstellt wird

Klicken Sie auf [Erstellen]. Die registrierte benutzerdefinierte Funktion wird daraufhin angezeigt.

Wenn eine vorher registrierte benutzerdefinierte Funktion bearbeitet wird,

wählen Sie die benutzerdefinierte Funktion aus, die Sie abändern möchten und klicken Sie auf [Bearbeiten]. Die registrierte benutzerdefinierte Funktion wird daraufhin angezeigt.

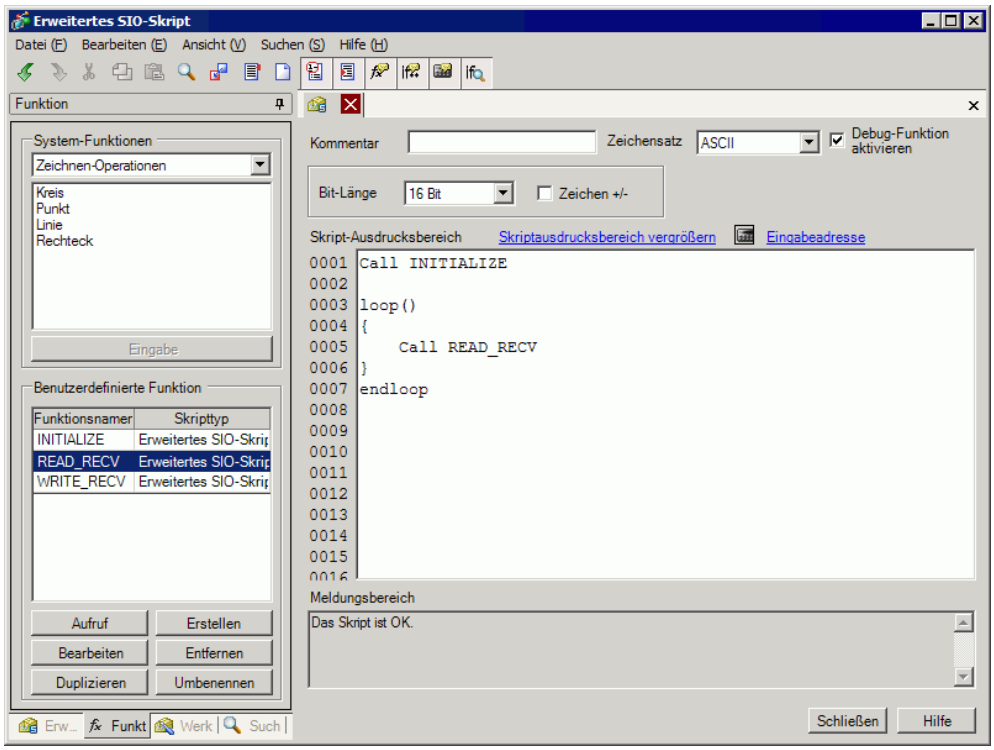
Geben Sie den Funktionsnamen ein und erstellen Sie das Skript im Ausführungsfeld. Klicken Sie auf [OK] und die benutzerdefinierte Funktion ist daraufhin gespeichert.



ANMERKUNG

- Einschränkungen treffen auf Funktionsnamen zu. Weitere Einzelheiten entnehmen Sie bitte "21.10.3 Einschränkungen bei benutzerdefinierten Funktionen" (seite 21-70) .

Wählen Sie die aufzurufende benutzerdefinierte Funktion auf, klicken Sie auf [Aufrufen] und "Funktionsname aufrufen" wird im Ausführungsfeld abgelegt.



21.8 Triggerbedingung - Einrichtung

Ein erstelltes Skript kann jedes der nachstehenden 7 Typen an Triggerbedingungen benutzen.

Einstellung		Beschreibung
Kontinuierliche Aktion		Das Skript wird regelmäßig ausgelöst.
Timer		Das Skript wird ausgelöst, nachdem die bestimmte Zeit abgelaufen ist.
Bit	Bit AN	Wenn die GP erkennt, dass das bestimmte Bit von 0 auf 1 ansteigt, wird das Skript ausgelöst.
	Bit AUS	Wenn die GP erkennt, dass das bestimmte Bit von 1 auf 0 abfällt, wird das Skript ausgelöst.
	Bitänderung	Wenn die GP erkennt, dass das bestimmte Bit von 0 1 ansteigt oder von 1 auf 0 absinkt, wird das Skript ausgelöst.
Bedingte Ausdrücke	Wenn eine Bedingung "wahr" ist	Wenn die GP "Ja" als bestimmten Ausdruck erkennt, wird das Skript ausgelöst.
	Wenn eine Bedingung "falsch" ist	Wenn die GP "Nein" als bestimmten Ausdruck erkennt, wird das Skript ausgelöst.

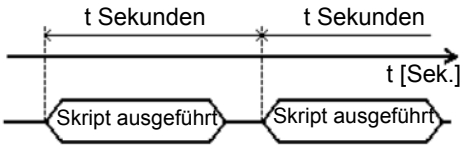
21.8.1 Kontinuierliche Aktion

Führt jede Anzeige-Abtastzeit aus.

21.8.2 Timer

■ Timer

Jedesmal, wenn die bestimmte Zeit abgelaufen ist, wird das Skript einmal ausgeführt. Die Timer-Dauer kann von 1 bis 32767 Sekunden eingestellt werden.



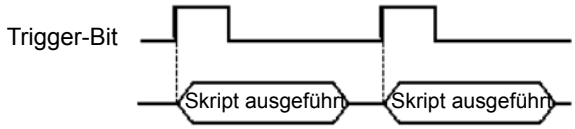
ANMERKUNG

- Wenn Sie die Zeit der Timerfunktion einstellen, wird der Zeitwert die festgesetzte Zeit + Fehler der Anzeigeabtastzeit enthalten. Abhängig von der Zeit, die es dauert, ein Bildschirm-Element zu zeichnen oder Daten auszudrucken, kann die Timer-Funktion jedoch langsamer sein. Weitere Informationen zur Anzeigeabtastzeit erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .
- Wenn D-Skripte benutzt werden, wird das Wechseln eines Bildschirms zum Neustart der Timer-Funktion führen (ab 0).

21.8.3 Bit

■ Bit AN

Wenn die GP erkennt, dass die bestimmte Bitadresse (Trigger-Bit) von 0 auf 1 ansteigt, wird das Skript ausgelöst.

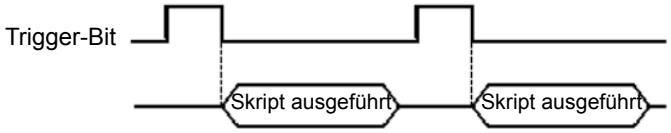


ANMERKUNG

- Für AN/AUS des Trigger-Bits stellen Sie sicher, dass das Intervall länger als die Kommunikations-Zykluszeit oder die Anzeige-Abtastzeit ist, anhängig davon, was länger ist. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .

■ Bit AUS

Wenn die GP erkennt, dass die bestimmte Bitadresse (Trigger-Bit) von 1 auf 0 absinkt, wird das Skript ausgelöst.

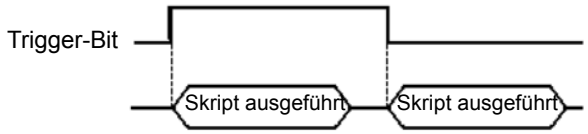


ANMERKUNG

- Für AN/AUS des Trigger-Bits stellen Sie sicher, dass das Intervall länger als die Kommunikations-Zykluszeit oder die Anzeige-Abtastzeit ist, anhängig davon, was länger ist. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .

■ Bitänderung

Wenn die GP erkennt, dass die bestimmte Bitadresse (Trigger-Bit) von 0 auf 1 ansteigt oder von 1 auf 0 absinkt, wird das Skript ausgelöst.



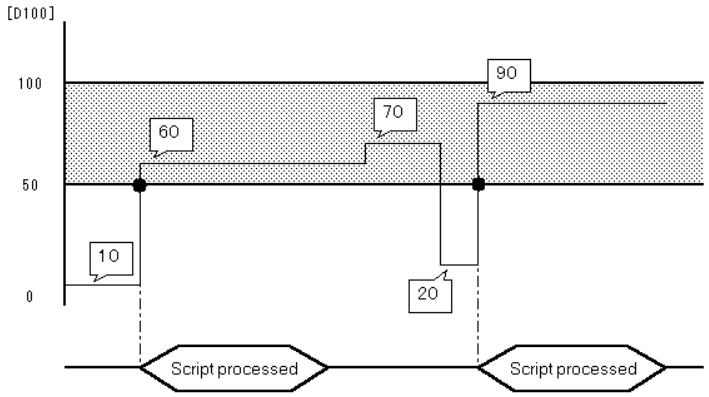
ANMERKUNG

- Für AN/AUS des Trigger-Bits stellen Sie sicher, dass das Intervall länger als die Kommunikations-Zykluszeit oder die Anzeige-Abtastzeit ist, anhängig davon, was länger ist. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .

21.8.4 Bedingte Ausdrücke

■ Wenn eine Bedingung "wahr" ist

Wenn die GP die Triggerbedingung als "wahr" auswertet, wird das Skript einmal ausgeführt.
Wenn die Triggerbedingung auf 100>[D100]>50 eingestellt ist, wird das Skript mit dem folgenden Timing ausgeführt:
[Wahr] −−> [Unwahr] wurde erkannt, das Skript führt aus, und 70 wird D100 zugewiesen.
Das Skript wird nicht ausgeführt, wenn [Unwahr]−−>[Unwahr] erkannt wird.



ANMERKUNG

- Vergewissern Sie sich bei der Triggerbedingung ein Intervall länger als die Kommunikations-Zykluszeit oder Anzeige-Abtastzeit anzulassen, welche auch immer länger ist. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .

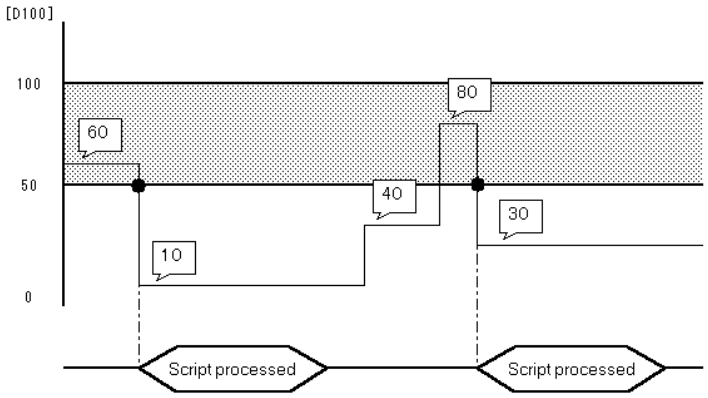
■ Wenn eine Bedingung "falsch" ist

Wenn die GP "Nein" als bestimmten Ausdruck in einem Trigger-Programm erkennt, wird das Skript einmal ausgeführt.

Wenn die Triggerbedingung auf 100>[D100]>50 eingestellt ist, wird das Skript mit dem folgenden Timing ausgeführt:

[Wahr] −−> [Unwahr] wurde erkannt, das Skript führt aus, und 20 wird D100 zugewiesen.

Das Skript wird nicht ausgeführt, wenn [Unwahr]−−>[Unwahr] erkannt wird.



ANMERKUNG

- Vergewissern Sie sich bei der Triggerbedingung ein Intervall länger als die Kommunikations-Zykluszeit oder Anzeige-Abtastzeit anzulassen, welche auch immer länger ist. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Einschränkungen zum Trigger-Bit" (seite 21-53) .

■ Einschränkungen zum Trigger-Bit

- Stellen Sie sicher, dass ein Intervall länger als die Kommunikations-Zykluszeit ist, um die Schreiboperationen in den verbunden Teilnehmer auszuführen. Wenn Schreiboperationen in den verbundenen Teilnehmer häufig ausgeführt werden, indem der Abtastzähler des internen Spezial-Relais der GP verwendet wird, können Kommunikations- oder Systemfehlern auftreten.
- Wenn das Bit, das für die D-Skript-Trigger-Bedingung auf "Touch" eingestellt ist und sich das Bit während einer D-Skript-Ausführung ausschaltet, kann das verwendete Timing beim wiederholten Drücken des Touch-Bereichs die Erkennung des Ansteigens des Bits verhindern. Der D-Skript-Trigger vergleicht vorher ausgelesene Werte mit den aktuellen ausgelesenen Werten, um zu bestimmen, ob der Trigger jetzt "wahr" ist. Während einer einzigen Abtastung bleibt jedoch der Wert, der in der Bit-Adresse gespeichert und während der Trigger-Operation verwendet wird, gleich, selbst wenn sich der Wert während der Ausführung geändert hat. Der neue Wert wird erst ausgelesen, nachdem die nächste Abtastung beginnt.

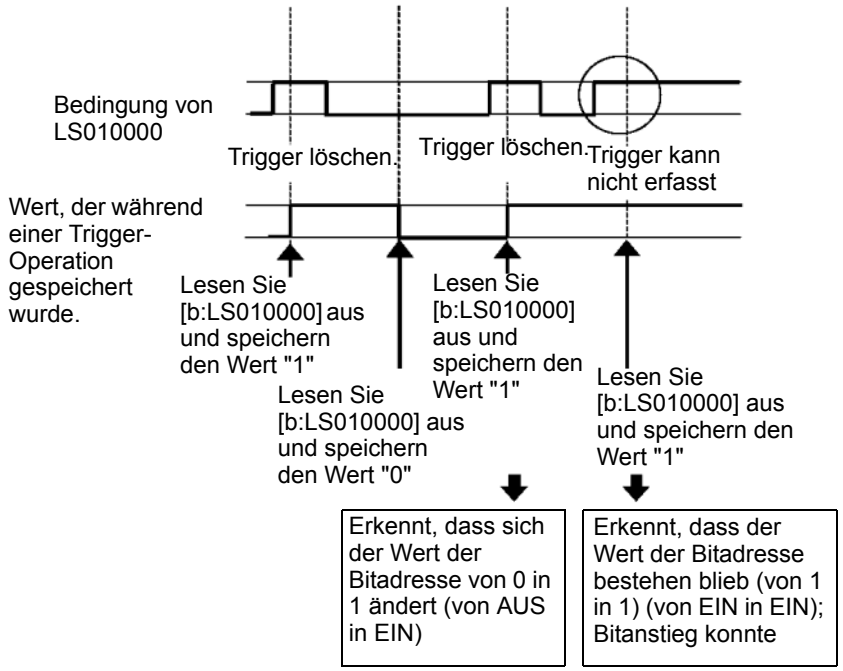
Kommunikations-Zykluszeit	Die Kommunikations-Zykluszeit ist die Zeit, die es dauert, wenn die Geräteeinheit Daten vom Teilnehmer/der SPS anfordert, bis zu dem Zeitpunkt, zu dem die Geräteeinheit die Daten erhält. Sie wird in der internen Adresse LS2037 als Binärdaten gespeichert. Die Einheit beträgt Millisekunden (Ms). Es liegt eine Varianz von +/- 10 ms vor.
Abtastzeit anzeigen:	Die Anzeige-Abtastzeit ist die Zeit, die zur Verarbeitung eines Bildschirms benötigt wird. Sie wird in der internen Adresse LS2036 als Binärdaten gespeichert. Die Einheit beträgt Millisekunden (Ms). Es liegt eine Varianz von +/- 10 ms vor.

z.B.: Wenn Touch benutzt wird, um das Triggerbit (LS010000) einzuschalten, schaltet D-Skript den Wert aus.

Triggerbedingung: Bit AN [#INTERN] LS010000

Skriptausdruck: clear ([b:[#INTERN]LS010000])

◆ Timing-Grafik der D-Skript-Ausführung



Wenn beispielsweise das Timing des D-Skripts nicht verwendet und nur eine Erfassung durchgeführt wird, wird sich die Arithmetik wie folgt gestalten:

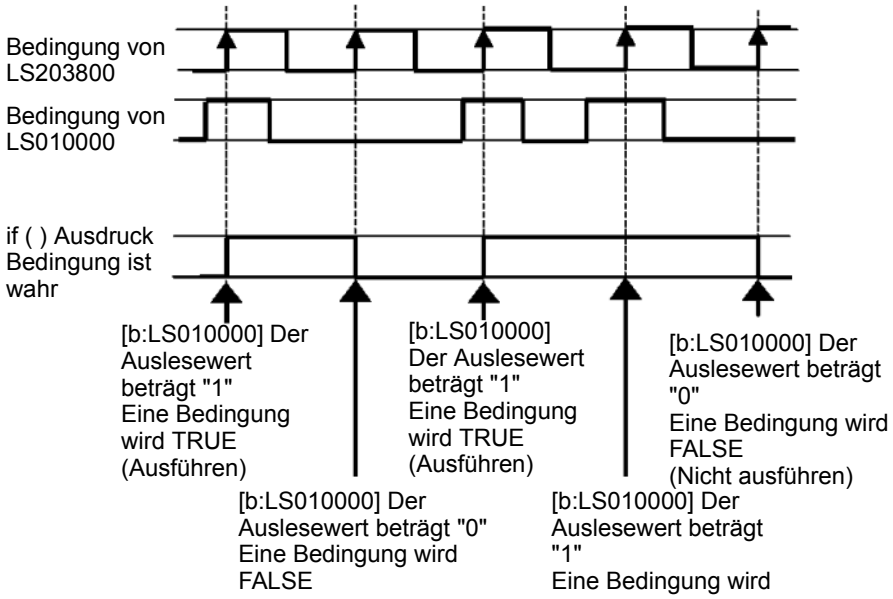
Benutzen einer "wenn" () Anweisung zur Erkennung eines Triggers.
Verwenden Sie eine "Wenn"-Anweisung, wenn die Touch-Operation das Bit setzt. Jedes Mal, wenn die "wenn"-Anweisung ausgeführt wird, liest sie den Wert und nimmt eine Vergleichsüberprüfung vor.

Triggerbedingung: Bit AN ([#INTERN]LS203800 *1)
Skriptausdruck: if ([b:[#INTERN]LS010000]==1)
{
clear ([b:[#INTERN]LS010000])
:
:

*1 Interner Zähler der GP. Der Zähler inkrementiert jedesmal, wenn das Element, das auf dem Anzeigebildschirm bestimmt wurde, ausgeführt wird.

Wenn Sie vorhergehende D-Skripte verwenden, wird das Skript nur ausgeführt, wenn die Bedingung übereinstimmt, selbst wenn Sie aufeinanderfolgende Berührungen eingeben. Wie in der nachstehenden Timing-Tabelle dargestellt, wird in jeder Anzeigeabtastung der Wert gelesen und auf eine Übereinstimmung überprüft und wenn der Wert übereinstimmt, ungeachtet der vorherigen Werte, wird das Skript ausgeführt.

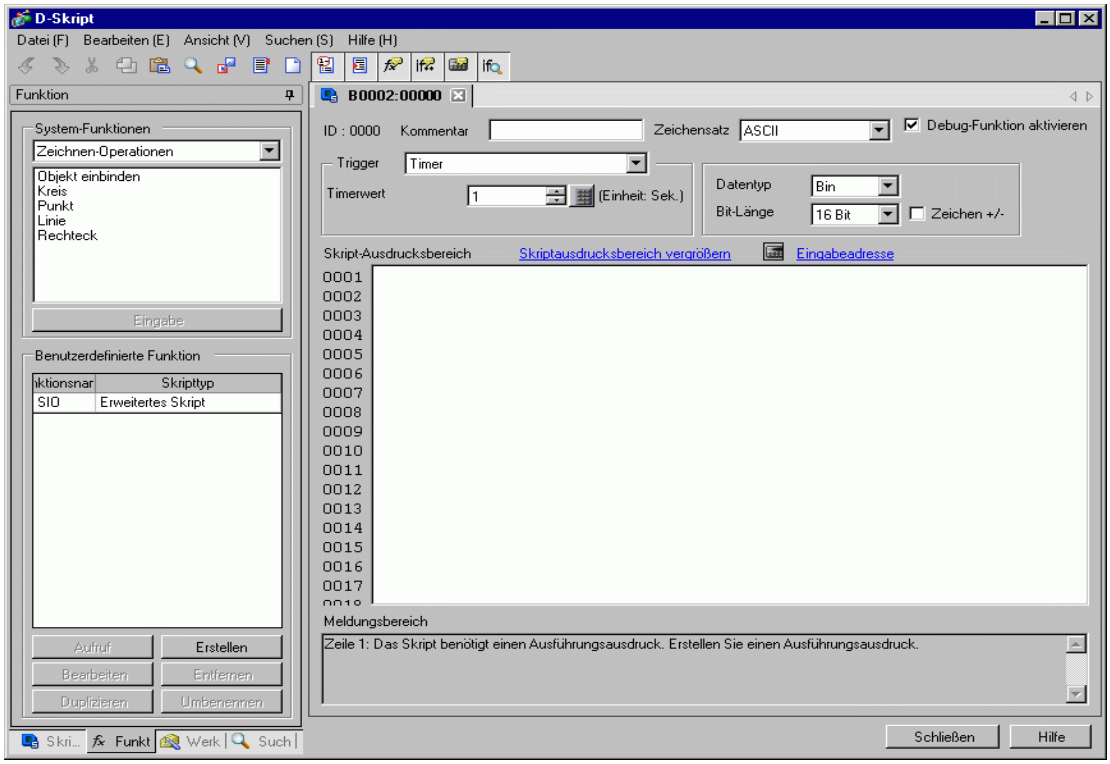
◆ Timing-Grafik der D-Skript-Ausführung



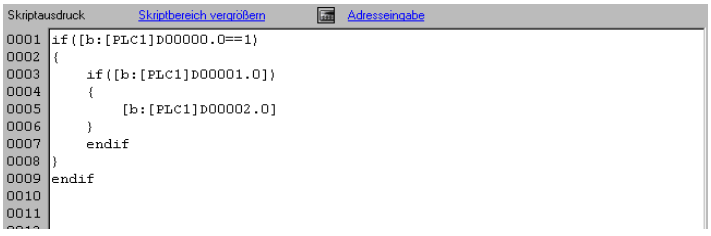
21.9 Einstellungsanleitung

21.9.1 D-Skript/Allgemeine Einstellungen [Globales D-Skript] Einstellungshinweise

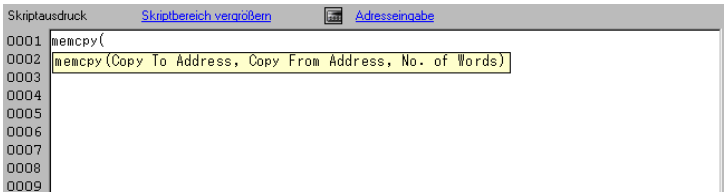
Nachstehend sehen Sie das Dialogfeld des Globalen D-Skripts. Die Einstellungen, die Sie festlegen können, sind dieselben, wie im Dialogfeld. Bei erweiterten Skripten gibt es keine Trigger und ID-Einstellungen; die anderen Einstellungen sind jedoch die gleichen wie unten.



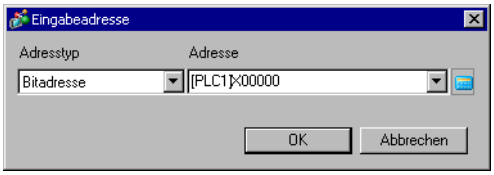
Einstellung	Beschreibung
Exportieren	Kann vom Menü "Datei" ausgewählt werden. Exportieren schreibt ein erstelltes Skript in eine Textdatei (.txt), das dann in andere Skripte importiert werden kann.
Importieren	Kann vom Menü "Datei" ausgewählt werden. Importieren liest ein exportiertes Skript (Textdatei).
Zeilen-Nr.	Zeigt die Zeilennummer rechts vom Programm an.
Automatische Einrücksteuerung	Rückt automatisch Anweisungen wie nachstehend ein:



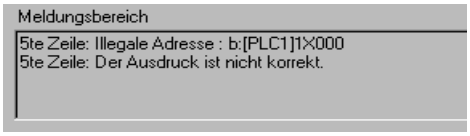
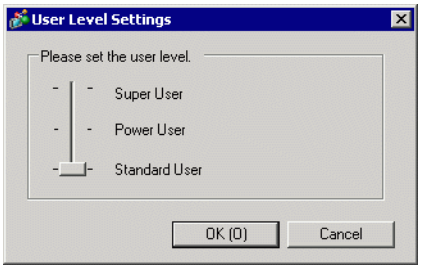
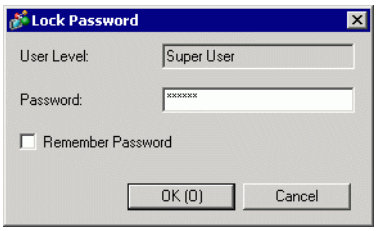
Fortsetzung

Einstellung	Beschreibung
Funktion-Eingabeunterstützung 	Wenn die Funktion und die Anfangsklammer "(" wie unten aufgeführt, eingegeben werden, wird das Format der Funktion angezeigt. 
Automatische Syntaxergänzung 	Wenn "falls" oder "Schleife" mit der Tastatur eingegeben wird, wird der verbleibende Syntax automatisch abgelegt.

Fortsetzung

Einstellung	Beschreibung
Adresseingabe 	Wenn ein Skript erstellt wird, geben Sie eine linke eckige Klammer ein ([) und das Dialogfenster [Eingabeadresse] wird automatisch angezeigt.  Wählen Sie den Adresstyp aus [Bitadresse], [Wortadresse] oder [Temporäre Adresse] aus. • Bitadresse Die Teilnehmer/SPS-Adresse, interne Adresse der GP und Bitvariable kann bestimmt werden. • Wortadresse Die Teilnehmer/SPS-Adresse, interne Adresse der GP und Ganzzahlvariable kann bestimmt werden. • Temporäre Adresse Diese Adresse kann nur für Skripte verwendet werden. Weitere Informationen zu internen Adressen finden Sie nachstehend. ☞ "A.1.2 Kommunizieren mit einem Teilnehmer/einer SPS mittels der direkten Zugriffsmethode" (seite A-4) ☞ "A.1.3 Verwenden der Speicherverknüpfungsmethode mit nicht unterstützten Teilnehmern/SPS" (seite A-6) WICHTIG • Legen Sie in den Skripten bitte KEINE Passwörter, usw. fest, die mit "0" beginnen. Alle numerischen Werte, die mit "0" beginnen, werden als Oct- (Basis-8)-Daten ausgeführt. • Wie man verschiedene Eingabedatenformate beschreibt Zum Beispiel: • DEZ (Basis-10) : Startwert ungleich Null Zum Beispiel: 100 • HEX (Basis-16) : Wert, der mit 0x beginnt. Zum Beispiel: 0x100 • OCT (Basis-8) : Wert, der mit 0x beginnt. Zum Beispiel: 0100 • Beispiel einer Operation mit unterschiedlichem Datenformat mittels Operator UND (Hex. und BCD) Nur Hex. 0x270F & 0xFF00 Ergebnis: 0x2700 BCD und Hex 9999 9999 & 0xFF00 Ergebnis: 0x9900

Fortsetzung

Einstellung	Beschreibung
Automatische Syntaxanalyse 	Überprüft den Syntax während der Skripterstellung. Das Ergebnis wird im unteren Teil des Fensters angezeigt. 
ID	Skripte werden durch eine ID-Nummer verwaltet. Wenn mehrere Skripte mit verschiedenen Triggerbedingungen erstellt werden, legen Sie bitte einen Wert von 0 bis 65535 fest.
Kommentar	Geben Sie einen Kommentar für das Skript ein.
Zeichensatz	Wählen Sie eine Sprache aus der Dropdown-Liste aus: [ASCII], [Japanisch], [Chinesisch (Traditionelle)], [Chinesisch (Vereinfacht)] oder [Koreanisch].
Debug-Funktion aktivieren	Legen Sie fest, ob Sie die Debug-Funktion aktivieren möchten oder nicht. Wenn die _Debug-Funktion im Korpus des Skripts existiert, wird die _Debug-Funktion ausgeführt. Weitere Informationen über diese Funktion erfahren Sie unter " ■ Debug-Funktion" (seite 21-151) .
Passwort verriegeln	Wählen Sie, ob die Passwort-Verriegelungsfunktion entweder aktiviert oder deaktiviert werden sollen. (1) Wählen Sie das Kontrollkästchen zur Anzeige des Bildschirms [Benutzerstufeneinstellungen]. (2) Wählen Sie die Benutzerstufe aus [Standard-Benutzer], [Leistungsbenutzer] und [Super-Benutzer], und klicken Sie dann auf [OK]. (3) Geben Sie [Passwort] ein, und klicken Sie auf [OK]. Wenn Sie das Kontrollkästchen [Passwort speichern] auswählen, wird das Passwort gespeichert und das Dialogfeld [Passwortverriegelung] nicht angezeigt.  
Trigger	Legen Sie die Triggerbedingung fest, die das Skript zum Ausführen bringt. Weitere Informationen über diese Funktion erfahren Sie unter "21.8 Triggerbedingung - Einrichtung" (seite 21-49) . Erweiterte Skripte verfügen über keine Trigger-Bedingungseinstellungen.
Datentyp	Legen Sie das Datenformat für das Skript auf Bin oder BCD fest. Bei erweiterten Skripten ist Bin festgelegt.
Bit-Länge	Legen Sie die Datenlänge für das Skript auf 16 oder 32 Bit fest

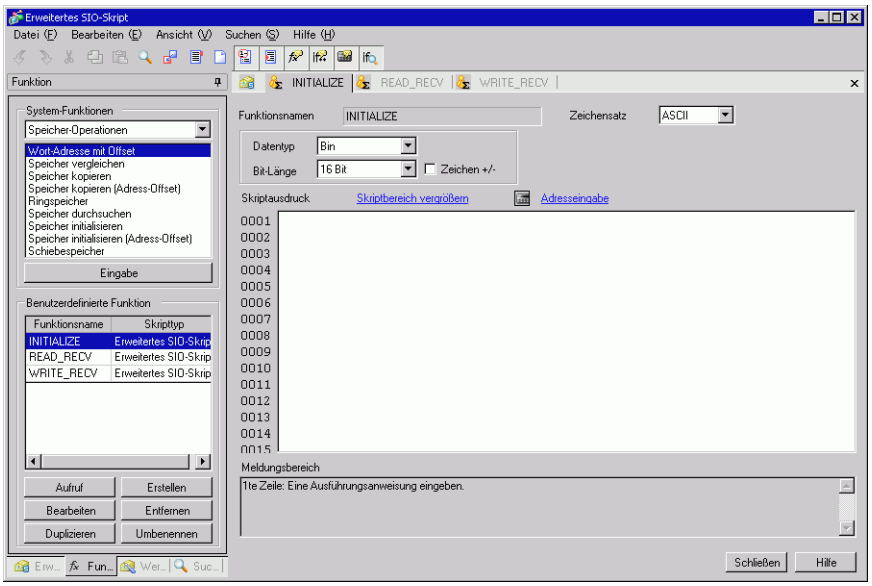
Fortsetzung

Einstellung	Beschreibung
Zeichen +/–	Wählen Sie diese Funktion aus, wenn Sie negative Zahlen einfügen möchten. Kann nur festgelegt werden, wenn der Datentyp "Bin" ist.
Skriptausdruck	Der Inhalt des Skripts.
System-Funktionen (Anleitung)	Wählen Sie in der Werkzeugleiste Befehle und Funktionen aus, die einfacher an ein Skript hinzugefügt werden können. Weitere Informationen zu verfügbaren Befehlen und Funktionen erfahren Sie unter ☞ "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74) System-Funktionen  Wählen Sie eine Kategorie aus dem Menü [System-Funktionen (Anleitung)] aus. Die verwandten Funktionen werden im unteren Bereich angezeigt. Wählen Sie die Funktion aus und klicken auf [Eingabe]. Das entsprechende Einstellungs-Dialogfeld wird angezeigt.
Benutzerdefinierte Funktion	Registrieren Sie ein erstelltes Skript als benutzerdefinierte Funktion, woraufhin es von anderen Skripten benutzt werden kann.  **ANMERKUNG** • Weitere Informationen zu benutzerdefinierten Funktionen erfahren Sie unter "21.9.2 Einstellungshinweise zu benutzerdefinierten Funktionen" (seite 21-62) .

Fortsetzung

Einstellung	Beschreibung
Werkzeugpalette	Wählen Sie als Shortcut Befehle aus der Werkzeugpalette aus, die im Skript verwendet werden können. Sie können außerdem Befehle auswählen, wie beispielsweise das Suchen und Platzieren von Text, der in Skripten verwendet wird. Weitere Informationen zu verfügbaren Befehlen finden Sie unter "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74) . 

21.9.2 Einstellungshinweise zu benutzerdefinierten Funktionen



Einstellung	Beschreibung
Funktionsname	Zeigt die Namen benutzerdefinierter Funktionen an.
Schriftart	Zeigt den Skripttyp an. Sie können anhand des Dropdown-Menüs zwischen [D-Skript] und [Erweitertes Skript] hin und her schalten.
Aufruf	Eine erstellte Funktion aufrufen. Wählen Sie die aufzurufende Funktion aus, klicken Sie auf [Aufrufen] und "Funktionsname aufrufen" wird im Ausführungsfeld angezeigt.
Erstellen	Erstellen einer neuen Funktion. Klicken Sie auf [Erstellen]. Das Dialogfeld mit der benutzerdefinierten Funktion wird angezeigt.
Bearbeiten	Bearbeiten einer bestehenden Funktion. Wählen Sie die zu bearbeitende Funktion aus, klicken auf [Bearbeiten]. Das Dialogfeld [D-Skript-Funktion] wird daraufhin angezeigt.
Entfernen	Löschen einer bestehenden Funktion. Wählen Sie die zu löschende Funktion aus und klicken auf [Löschen].
Duplizieren	Kopieren einer bestehenden Funktion. Wählen Sie die Funktion zum Kopieren aus und klicken auf [Kopieren], um das Dialogfenster anzuzeigen, um den Namen der Kopie der Funktion zu erstellen.
Umbenennen	Ändern Sie den Namen einer erstellten Funktion. Klicken Sie auf [Umbenennen]. Das Dialogfeld mit der Umbenennungsfunktion wird angezeigt.

21.10 Einschränkungen

21.10.1 D-Skript/Globale D-Skript-Einschränkungen

- Als Richtlinie zum D-Skript-Programmieren gilt: drei Adressen benötigen denselben Speicher wie ein Element. Die Höchstzahl der verfügbaren Adressen für ein D-Skript beträgt 255[*1]. Verwenden Sie so wenig wie mögliche Adressen, da die Antwort langsamer ist, je mehr Teilnehmer verwendet werden.
- Ein D-Skript kann keine Berechnungen auf auf Gleitkommawerten (Gleitkommavariablen oder reelle Variablen) ausführen. Es können auch keine Berechnungen an Strukturvariablen durchgeführt werden. Es können jedoch Berechnungen über individuelle Elemente aus den Strukturvariablen erstellt werden.
- Die Größe eines D-Skripts beeinträchtigt die Anzeige-Abtastzeit. Beachten Sie, dass eine große Anzahl von Adressen die Leistung des Programm entscheidend beeinträchtigen kann.
- Bestimmen Sie in den Trigger-Bedingungen nicht [Kontinuierliche Aktion], durch die das Skript in die Teilnehmer/SPS-Adressen schreibt. Dadurch wird ein Fehler auftreten, da die Kommunikationsverarbeitung nicht mit den vielen Schreibanweisungen mithalten kann. Verwenden Sie zum Aktivieren der [Kontinuierlichen Aktion] die interne oder temporäre Adresse der GP.
- Wenn eine Funktion von einer Funktion aufgerufen wurde, beträgt die Höchstzahl der Stufen (Verschachtelungen) 9. Erstellen Sie nicht mehr.
- Es können bis zu 9 Stufen (Verschachtelungen) aufgerufen und erstellt werden.
- Es können bis zu 254 Funktionen erstellt werden.

*1 Gesamtzahl der Teilnehmer, die in Trigger-Ausdrücken und Skript-Programmen verwendet werden.

◆ **Abhängig von den Teilnehmern, die für die Trigger-Bedingungen bestimmt wurden, lauten die D-Skript-Operationen, die durch einen Trigger nach Bildschirmwechsel ausgelöst wurden, wie folgt:**

Triggerbedingung		Jeder verbundene Teilnehmer außer [#MEMLINK]				[#MEMLINK]			
	Aktueller Wert oder Bedingung	Bit "0"	Bit "1"	Eine Bedingung wird FALSE	Eine Bedingung wird TRUE	Bit "0"	Bit "1"	Eine Bedingung wird FALSE	Eine Bedingung wird TRUE
Bit-Vorderkante		X	O	–	–	X	X	–	–
Fallende Bit-Flanke		O	X	–	–	X	X	–	–
Bitänderung		O	O	–	–	X	X	–	–
Timer-Einstellung		X	X	X	X	X	X	X	X
"Wahr" wird erkannt.		–	–	X	O	–	–	X	O
"Unwahr" wird erkannt.		–	–	O	X	–	–	O	X

O: Das Verfahren wird sofort nach Wechseln des Bildschirms oder nach AN Schalten des Stroms ausgeführt.

X: Das Verfahren wird nicht sofort nach Wechseln des Bildschirms oder nach AN Schalten des Stroms ausgeführt.

- Wenn der Trigger-Timer ausgeführt wird, fängt der Timer mit dem Zählen gleich nach dem Bildschirmwechsel an.
- Wenn Globale D-Skripte verwendet werden, werden die oben-genannten Operationen nur ausgeführt, wenn die Netzspannung der GP eingeschalten ist. Wenn die GP-Bildschirme jedoch gewechselt werden, wird die oben-genannte Operation nicht durchgeführt und die eingestellten Triggerbedingungen werden kontinuierlich überwacht.
- Wenn ein globales D-Skript einen Timer enthält, fängt der Timer sofort zum Zählen an, sobald die Stromzufuhr zur GP eingeschalten ist.

ANMERKUNG

- Verwenden Sie nicht die Touch-Bildschirm-Tastatur zum Einstellen des Trigger-Bits oder um das Startbit im Programm auszuführen. Das Timing der Touch-Eingabe kann möglicherweise falsch sein, was dazu führt, dass das Bit falsch eingegeben wird.

◆ **Wenn ein Wert einer Adresse zugewiesen wurde, die Bildschirme während der Ausführung eines D-Skript-Befehls wechselt, wird die Bildschirmwechsel-Operation durchgeführt, nachdem alle D-Skripte durchgeführt wurden.**

Zum Beispiel:

ID 00000
Datentyp Bin Datenlänge 16 Bit Zeichen +/– Keine
Trigger Führendes Bit([b:M0000])

```
[w:[PLC1]D0100]=0                  // (1)
[w:[#INTERNAL]LS0008]=30           // (2) Wechselt zum Basis-Bildschirm Nr. 30
[w:[PLC1]D0101]=1                  // (3)
[w:[PLC1]D0102]=2                  // (4)
```

Wenn das obige D-Skript ausgeführt wird, wird die Verarbeitung des Bildschirmwechsels ausgeführt, wenn (3) und (4) durchgeführt wurde.

◆ **Wenn die in D-Skripten verwendeten Daten mit einer GP-Touch-Operation eingerichtet wurden, stellen Sie bitte sicher, dass die Operation zum Schreiben von Daten abgeschlossen ist, bevor das D-Skript ausgeführt wird.**

◆ **Einschränkungen, die speziell auf Globale D-Skripte zutreffen.**

- Wenn die Spannung der GP eingeschalten wird, werden die auf der vorhergehenden Seite in der Tabelle angezeigten Aktionen durchgeführt. Beim Bildschirmwechsel trifft die obige Tabelle nicht zu und die Triggerbedingungen werden fortlaufend überwacht.
- Die Globalen D-Skript-Operationen werden während Bildschirmwechsel oder anderen GP-Operationen eingestellt.
- Nachdem die GP eingeschalten wurde, werden Globale D-Skript-Aktionen erst durchgeführt, wenn alle Dateneinlesungen für den Anfangsbildschirm abgeschlossen wurden. Globale D-Skript-Aktionen können jedoch vor dem Abschluss des Lesens von Daten durchgeführt werden, nachdem der Anfangsbildschirm gewechselt wurde.
- Die Höchstzahl der Teilnehmer in globalen D-Skripts beträgt 255[*1]. Wenn diese Zahl überschritten wird, werden die D-Skripte nicht funktionieren. Da diese Teilnehmer immer Daten lesen, ungeachtet der Bildschirme, müssen Sie sicherstellen, dass Sie nur die Mindestzahl an Teilnehmereinstellungen in Ihrem D-Skript benutzen. Andernfalls kann die Operationsleistung beeinträchtigt werden.
- Die Höchstzahl an verfügbaren globalen D-Skripten beträgt 32. Die aktuell benutzte Funktion zählt ebenfalls als ein Globales D-Skript). Wenn die Zahl der globalen D-Skripte 32 übersteigt, werden alle neuen Globalen D-Skripte ignoriert.

*1 Gesamtzahl der Teilnehmer, die in Trigger-Ausdrücken und Skript-Programmen verwendet werden.

◆ Einschränkungen zu SIO-Port-Operationen

- Adressen, die in den Funktionen Senden/Empfangen bestimmt wurden, werden nicht zu den D-Skript-Adressen dazu gezählt.
- Das Steuerelement ist eine "Nur Schreiben"-Variable, während Status- und Empfangsdaten "Nur Lese"-Variablen sind. Versuchen Sie nicht, die Steuervariable auszulesen oder Daten in die Statusvariable zu schreiben, da das zu einem Fehlschlagen der Operation führt.
- Erstellen Sie unabhängige D-Skripte (oder Funktionen) für Operationen zum Daten Senden und Empfangen. Weitere Informationen über die Ablaufdiagramme der Datenübertragungen erfahren Sie unter
  ☞" ■ Ablaufdiagramm" (seite 21-25)
- Der gültige Bereich des LS-Teilnehmers, der Daten für die Funktionen zum Senden und Empfangen von Daten speichern kann, ist der Benutzerbereich (LS20 bis LS2031 und LS2096 bis LS8191).
- Wenn im Arbeitsbereich der [Systemeinstellungen] auf der Seite [Skripteinstellungen] der [Typ] nicht auf [D-Skript/Globales D-Skript] festgelegt ist, wird das 13. Bit in der Adresse LS2032 eingechaltet, wenn das [D-Skript/Globale D-Skript] die Bezeichnungseinstellungsfunktionen der [SIO-Port-Operation]'(Senden, Empfangen, Steuern, Lesestatus und Datengröße empfangen) ausführt. Informationen zu speziellen Relais finden Sie unter:
  ☞"A.1.4.3 Spezial-Relais" (seite A-25)
- Wenn die Funktionen Senden/Empfangen benutzt werden, muss die Bitlänge des D-Skripts auf 16 Bit eingestellt werden. Bitte beachten Sie, dass die Operation fehlschlagen wird, wenn die Bitlänge auf 32 Bit eingestellt ist.
- Die Größe des Sendepuffers beträgt 2.048 Bytes, während der Empfangspuffer über 8.192 Bytes verfügt. Das ER-Signal (Ausgabe)-RS-Signal (Ausgabe) wird ausgeschaltet, nachdem mindestens 80% des Empfangspuffers voll mit empfangenen Daten ist.

◆ Einschränkungen bei BCD-Format-Operationen

Wenn ein Wert, der nicht in ein BCD-Format konvertiert werden kann, während einer Operation erkannt wird, führt das Programm nicht weiter aus.
Diese Werte enthalten A bis F im hexadezimalen Format.
Bitte verwenden Sie solche Werte nicht. Wenn das Programm aufgrund von Nicht-BCD-Werten anhält, schaltet sich Bit "7" in der allgemeinen Relais-Information (LS2032) in der GP ein. Das Bit schaltet sich erst aus, wenn die GP ausgeschaltet wird oder Offline ist.
Zum Beispiel:

[w:[PLC1]D0200]=([w:[PLC1]D0300]<<2)+80
Wenn D300 "3" ist, führt das Verschieben von 2 Bits nach links zu "0x000C", die nicht in ein BCD-Format konvertiert werden kann und die Programmausführung unterbricht.

[w:[PLC1]D0200]=[w:[PLC1]D0300]<<Wenn D300 "3" ist, führt das Verschieben von 2 Bits nach links zu "0x000C".
Im Gegensatz zum obigen Beispiel stellt "0x000C" das Ergebnis einer Operation dar, die im Speicher gespeichert wird und die Programmausführung nicht anhält.

◆ Einschränkungen bei Null-Operationen

Wenn Sie durch Null in Divisions- (/) und Modulusoperationen (%) teilen, wird die Ausführung angehalten. Teilen Sie nicht durch Null.
Wenn das Programm aufgrund des oben erwähnten Fehlers anhält, schaltet sich Bit "8" in der allgemeinen Relais-Information (LS2032) in der GP ein. Das Bit schaltet sich erst aus, wenn die GP ausgeschaltet wird oder Offline ist.

◆ Anmerkungen zu Einschaltverzögerungen während Zuweisungsoperation

Das Benutzen von Teilnehmeradressen in einer Zuweisungsoperation kann zu Schreibverzögerungen führen, da die GP die Adressdaten vom verbundenen Teilnehmer lesen muss. Ziehen Sie folgendes in Betracht:
Zum Beispiel:

[w:[PLC1]D0200]=([w:[PLC1]D0300]+1 ...
[w:[PLC1]D0201]=([w:[PLC1]D0200]+1 ...

Vorgang (1) teilt (D0300+1) in D0200 zu. Jedoch wurde in Vorgang (2) das Ergebnis von Vorgang (1) noch nicht in D0200 aufgrund von zeitraubender Kommunikation mit dem Teilnehmer zugewiesen. In solchen Fällen, programmieren Sie den Vorgang so, dass das Ergebnis von Vorgang (1) im LS-Bereich gespeichert wird, bevor er ausgeführt wird (siehe unten).

[w:[#INTERNAL]LS0100]=[w:[PLC1]D0300]+1
[w:[PLC1]D0200] = [w:[#INTERNAL]LS0100]
[w:[PLC1]D0201]=[w:[#INTERNAL]LS0100]+1

◆ Anmerkungen zur Handhabung von negativen Zahlen

Bei Funktionen, für die eine negative Zahl für ein Argument eingegeben wird, das keine negativen Zahlen akzeptiert, [*1] wird die eingegebene Zahl als vorzeichenlos angesehen [*2].

*1 Beispiel: "Anzahl der Bytes" des Arguments _CF_read () kann keine negativen Zahlen akzeptieren, da es so groß ist, wie die zu lesenden Daten.

*2 Beispiel: –1 wird für 16 Bit als 65.535 gehandhabt und 4294967295 für 32 Bit.

21.10.2 Einschränkungen zu erweiterten Skripten

- Für die Teilnehmeradressen können nur der LS-Bereich und USR-Bereich (Erweiterter Benutzerbereich) verwendet werden.
- Die vorübergehenden Adressen des D-Skripts und des Globalen D-Skripts werden unabhängig von den temporären Adressen des erweiterten Skripts verwaltet. Deshalb werden Änderungen an den temporären Adressen des D-Skripts und des Globalen D-Skripts nicht in der temporären Adresse des erweiterten Skripts widergespiegelt.
- Sie können benutzerdefinierte Funktionen aufrufen, die mit einem D-Skript/Globalen D-Skript erstellt wurden; wenn Sie jedoch auf eine Teilnehmeradresse außerhalb der internen Adresse in der Funktion zugreifen möchten, kann diese eventuell nicht normal ausgeführt werden. Außerdem, wenn sie übertragen wurden (während der Erstellung von Daten für die GP), werden benutzerdefinierte Funktionen unabhängig für D-Skripte, Globale D-Skripte und erweiterte Skripte erstellt.
- Wenn eine Funktion von einer Funktion aufgerufen wurde, beträgt die Höchstzahl der Stufen (Verschachtelungen) 9.
- Es können bis zu 254 Funktionen aufgerufen werden. (Die Anzahl an verfügbaren Funktionen mit "Aufrufen" beträgt 254).
- Ein erweitertes Skript hat keinen Einfluß auf die Anzahl der Markierungszählung.
- Funktionen, die nur von einem erweiterten Skript unterstützt werden, wie beispielsweise Zeichenfolge-Operationen, werden nicht funktionieren, wenn Sie mit einem D-Skript oder Globalen D-Skript aufgerufen werden.
- Das verfügbare Datenformat ist Bin. Das BCD-Datenformat ist deaktiviert.
- Die Größe des Sendepuffers beträgt 2.048 Bytes, während der Empfangspuffer über 8.192 Bytes verfügt. Die CTS-Zeile wird ausgeschaltet, nachdem mindestens 80% des Empfangspuffers voll mit empfangenen Daten ist.
- D-Skripte/Globale D-Skripte und Erweiterte Skripte können nicht gleichzeitig ausgewählt werden. Beachten Sie die in der nachstehenden Tabelle aufgeführten Kombinationen:

Erweiterte SIO-Einstellung	D-Skript/Globales D-Skript Erweiterte SIO-Funktion für erweitertes Skript	Erweiterte SIO-Funktion für erweitertes Skript
D-Skript/Globales D-Skript	O: Operation möglich	X: Keine Operation
Erweitertes Skript	X: Keine Operation	O: Operation möglich

- Zeichenkonventionen für die Einstellung der Zeichenfolge

Anzeigen doppelter Anführungszeichen in den Zeichenfolgen, fügen Sie das "\"-Symbol an und drücken es als [\""] aus. Das einfache "\"-Symbol kann nicht dargestellt werden. Zum Anzeigen doppelter Anführungszeichen in den Zeichenfolgen, fügen Sie das "\"-Symbol an und drücken es als [\""] aus. Wenn notwendig, verwenden Sie die Formateinstellung des Zeichencodes (_strset (databuf0, 92).
Zum Beispiel:

"ABC\"DEF"	ABC"DEF
"ABC\DEF"	ABC\DEF
"ABC\\"DEF"	ABC\"DEF
"ABC\\DEF"	ABC\\DEF

- Bei Funktionen, für die eine negative Zahl für ein Argument eingegeben wird, das keine negativen Zahlen akzeptiert, [*1]wird die eingegebene Zahl als vorzeichenlos angesehen [*2].

*1 Beispiel: "Anzahl der Bytes" des Arguments _CF_read () kann keine negativen Zahlen akzeptieren, da es so groß ist, wie die zu lesenden Daten.

*2 Beispiel: –1 wird für 16 Bit als 65.535 gehandhabt und 4294967295 für 32 Bit.

◆ Größe der zugewiesenen Puffer für Erweiterte SIO (databuf0, databuf1, databuf2, und databuf3)

Puffer	Puffer-Name	Größe
Datenpuffer 0	databuf0	1 KB
Datenpuffer 1	databuf1	1 KB
Datenpuffer 2	databuf2	1 KB
Datenpuffer 3	databuf3	1 KB

21.10.3 Einschränkungen bei benutzerdefinierten Funktionen

- Abschnitte der Befehle, die verwendet werden können, sind bei jedem Skript unterschiedlich. Wenn Sie Befehle benutzen möchten, beziehen Sie sich bitte auf "21.11 Programmbefehle/Bedingte Ausdrücke" (seite 21-74) .
- Für den Funktionsnamen können alle beliebigen englischen Buchstaben oder der Unterstrich "_" verwendet werden. (Der Funktionsname muss jedoch mit einem alphanumerischen Zeichen beginnen.)
- Verwenden Sie die folgenden Ausdrücke nicht als Funktionsnamen:

UND	b_call	Bcall	_bin2hexasc	break	Aufruf
_CF_delete	_CF_dir	_CF_read	_CF_read_csv	_CF_rename	_CF_write
_USB_delete	_USB_dir	_USB_read	_USB_read_csv	_USB_rename	_USB_write
Löschen	databuf0	databuf1	databuf2	databuf3	_decasc2bin
_dlcopy	dsp_arc	dsp_circle	dsp_dot	dsp_line	dsp_rectangle
else	endif	fall	_hexasc2bin	if	IO_READ
IO_READ_EX	IO_READ_WAIT	IO_WRITE	IO_WRITE_EX	loop	_memcmp
memcpy	_memcpy_EX	memring	_memsearch	memset	_memset_EX
_memshift	not	ODER	return	rise	rise_expr
set	_strcat	_strlen	_strmid	_strset	timer
toggle	_wait				

21.10.4 Anmerkungen zu Operationsergebnissen

■ Überlaufende Ziffern

Überlaufende Ziffern aufgrund von Operationen werden abgeschnitten.
Wenn Operationen an ungezeichneten 16-Bitdaten durchgeführt werden:

- 65535 + 1 = 0 (Produziert überlaufende Ziffern)
- (65534 ∗ 2) / 2 = 32766 (Produziert überlaufende Ziffern)
- (65534 / 2) ∗ 2 = 65534 (Produziert keine überlaufenden Ziffern)

■ Unterschied der Restarithmetik

Das Ergebnis einer Restarithmetik hängt davon ab, ob die linke und rechte Seite gezeichnet oder ungezeichnet ist.

- –9 % 5 = –4
- 9 % –5 = 4

■ Abgeschnittene Dezimalstellen

Bruchwerte aus der Division werden verkürzt.

- 10 / 3 ∗ 3 = 9
- 10 ∗ 3 / 3 = 10

■ Anmerkungen zum Ausführen von BCD-Daten

Eine BCD-Datenoperation, die überlaufende Ziffern produziert, ergibt kein korrektes Ergebnis.

21.10.5 Fehler

Wenn ein Skript falsch konfiguriert wurde, wird folgende Fehlermeldung angezeigt: Der Fehler wird unten auf dem GP-Bildschirm angezeigt.
Fehlercodes werden in die LS91XX-Adressen geschrieben. Die Zahl, die in den Fehlercodebereich geschrieben wird, ist die Zahl, gefolgt von RAAA in der nachstehenden Tabelle. (Wenn beispielsweise der Fehler RAAA130 auftritt, wird 130 geschrieben.)

Liste der Skript-Fehlercodes

D-Skript (Fehleradresse=LS9120)	Globales D-Skript (Fehleradresse=LS9110)	Erweitertes Skript (Fehleradresse=LS9100)
–	RAAA130	RAAA140
Unbenutzt	Globaler D-Skriptfehler (Die Gesamtzahl der Globalen D-Skripte übersteigt die maximale Anzahl von 32).	Erweiterter D-Skript-Fehler (Die Gesamtzahl der Funktion überschreitet die Höchstzahl von 255).
–	RAAA131	–
Unbenutzt	Globaler D-Skriptfehler (Die Gesamtzahl der Teilnehmer übersteigt die maximale Einstellung von 255[*1]).	Unbenutzt
RAAA120	RAAA132	RAAA141
D-Skript-Fehler (Die bestimmte Funktion existiert nicht oder die Funktion weist einen Fehler auf.)	Globaler D-Skript-Fehler (Die bestimmte Funktion existiert nicht oder die Funktion weist einen Fehler auf.)	Erweiterter D-Skript-Fehler (Die bestimmte Funktion existiert nicht oder die Funktion weist einen Fehler auf.)
RAAA121	RAAA133	RAAA142
D-Skript-Fehler (Diese Funktionen sind in 10 oder mehr Stufen verschachtelt.)	Globaler D-Skript-Fehler (Diese Funktionen sind in 10 oder mehr Stufen verschachtelt.)	Erweiterter D-Skript-Fehler (Diese Funktionen sind in 10 oder mehr Stufen verschachtelt.)
RAAA122	RAAA134	RAAA143
D-Skript-Fehler (Es wurde ein Ausdruck eingegeben, der von dieser Version nicht unterstützt wird).	Globaler D-Skript-Fehler (Es besteht ein Ausdruck, der von dieser Version nicht unterstützt wird).	Erweiterter D-Skript-Fehler (Es besteht ein Ausdruck, der von dieser Version nicht unterstützt wird).
RAAA123	RAAA135	RAAA144
D-Skript-Fehler (Die SIO-Operationsfunktion wird in einer Bedingung verwendet, in der kein Teilnehmer/ keine SPS bestimmt wurde.)	Globaler D-Skript-Fehler (Die SIO-Operationsfunktion wird in einer Bedingung verwendet, in der kein Teilnehmer/keine SPS bestimmt wurde.)	Erweiterterter D-Skript-Fehler (Die SIO-Operationsfunktion wird in einer Bedingung verwendet, in der kein Teilnehmer/keine SPS bestimmt wurde.)

Fortsetzung

RAAA124	RAAA136	RAAA145
Das D-Skript weist einen Fehler auf.	Das Globale D-Skript weist einen Fehler auf.	Das erweiterte D-Skript weist einen Fehler auf.

*1 Gesamtzahl der Teilnehmer, die in Trigger-Ausdrücken und Skript-Programmen verwendet werden.

21.11 Programmbefehle/Bedingte Ausdrücke

■ Funktion

Element	Befehl/Funktion	D-Skript/Globales D-Skript	Erweitertes Skript
Datentyp	Bin, BCD	O	Nur Bin
Bit-Länge	16 Bit, 32 Bit	O	O
Mit Vorzeichen +/-	Aktiviert/deaktiviert	O	O
Trigger	Timer-Einstellung	O	X
	Bit-Vorderkante	O	X
	Fallende Bit-Flanke	O	X
	Toggle-Bit	O	X
	Ausdruck ist wahr	O	X
	Ausdruck ist falsch	O	X
Zeichnen	Bildschirm laden	O	X
	Punkt	O	O
	Linie	O	O
	Kreis	O	O
	Rechteck	O	O
Operator	Addition (+)	O	O
	Subtraktion (–)	O	O
	Modulus (%)	O	O
	Multiplikation (*)	O	O
	Division (/)	O	O
	Zuweisung (=)	O	O
Vergleichsoperation	Logisches AND	O	O
	Logisches OR	O	O
	Negation (nicht)	O	O
	Kleiner als (<)	O	O
	Kleiner-Gleich als (<=)	O	O
	Ungleich (<>)	O	O
	Größer als (>)	O	O
	Größer-Gleich als (>=)	O	O
	Gleich (==)	O	O

Fortsetzung

Element	Befehl/Funktion	D-Skript/Globales D-Skript	Erweitertes Skript
Speicher-Operationen	Speicher kopieren: memcpy ( )	O	O
	Speicher initialisieren: memset ( )	O	O
	Speicher kopieren (Variablen-Spezifikation) _memcpy_EX ( )	O	O
	Speicher initialisieren (Adress-Offset) _memcpy_EX ( )	O	O
	Wort-Adresse mit Offset	O	O
	Schiebespeicher	O	O
	Speicherring	O	O
	Speicher durchsuchen	O	O
	Speicher vergleichen	O	O
Bit-Operation	Links verschieben (<<)	O	O
	Rechts verschieben (>>)	O	O
	Bitweise UND	O	O
	Bitweise ODER	O	O
	Bitweise ODER	O	O
	1er-Komplement	O	O
	Gesetztes Bit: set ( )	O	O
	Bit löschen: clear ( )	O	O
	Bit umschalten: toggle ( )	O	O
Bedingte Ausdrücke	if ( )	O	O
	if ( ) else	O	O
	loop ( ), break	O	O
	loop ( ) Endlosschleife	X	O
Adresse	Bitadresse	O	Interne Adresse
	Wortadresse	O	Interne Adresse
	Temporäre Arbeitsadresse	O	O[*1]
Konstante	Dez, Hex, Oct	O	O

Fortsetzung

Element	Befehl/Funktion	D-Skript/Globales D-Skript	Erweitertes Skript
SIO-Funktion	Empfangen: IO_READ ([p:SIO])	O	O
	Senden: IO _WRITE ([p:SIO])	O	O
	Erweiterter Empfang: _IO_READ_EX ( )	X	O
	Erweiterte Sendung _IO_WRITE_EX ( )	X	O
	Standby-Empfangsfunktion _IO_READ_WAIT ( )	X	O
	Steuerung [c:EXT_SIO_CTRL]	O	O
	Status [s:EXT_SIO_STAT]	O	O
	Empfangene Datengröße [r:EXT_SIO_RCV]	O	O
	Pause: _wait ( )	X	O

Fortsetzung

Element	Befehl/Funktion	D-Skript/Globales D-Skript	Erweitertes Skript
String-Operation	Text	X	O
	Datenpuffer databuf0, databuf1, databuf2, databuf3	X	O
	Zeichenfolge schreiben _strset ( )	X	O
	Kopiere Datenpuffer in int. Adressen _dlcopy ( )	X	O
	Kopieren von interner Adresse in Datenpuffer: _ldcopy ( )	X	O
	Textkonvertierung Hex-nach-Integer _hexasc2bin ( )	X	O
	Stringkonv. Dez-nach-Integer _decasc2bin ( )	X	O
	Konvertierung numerischer Wert nach Hex-String _bin2hexasc ( )	X	O
	Konvertierung numerischer Wert nach Dez-String _bin2decasc ( )	X	O
	String-Länge _strlen ( )	X	O
	String-Verkettung _strcat ( )	X	O
	Teil-Strings kopieren _strmid ( )	X	O
	Status [e:STR_ERR_STAT]	X	O

Fortsetzung

Element	Befehl/Funktion	D-Skript/Globales D-Skript	Erweitertes Skript
Funktion	Aufruf	O	O
	return	X	O
CF-Datei-Verfahren	CSV-Datei lesen	O	O
	Ausgabe-Dateiliste _CF_dir ( )	O	O
	Datei lesen _CF_read ( )	O	O
	CSV-Datei lesen CF_read_csv ( )	O	O
	Datei schreiben _CF_write ( )	O	O
	Datei löschen _CF_delete ( )	O	O
	Dateiname ändern _CF_rename ( )	O	O
USB-Dateioperation	USB-Lesedatei	O	O
	Ausgabe-Dateiliste _USB_dir ( )	O	O
	Datei lesen _USB_read ( )	O	O
	CSV-Datei lesen USB_read_csv ( )	O	O
	Datei schreiben _USB_write ( )	O	O
	Datei löschen _USB_delete ( )	O	O
	Dateiname ändern _USB_rename ( )	O	O
Drucker-Operationen	COM-Port ausgeben: IO_WRITE ([p:PRN])	O	O
Debug	_debug ( )	O	O

*1 Die temporäre Adresse existiert unabhängig von D-Skripten und globalen D-Skripten.

21.11.1 Bit-Operation

Bit-Operation	Funktionszusammenfassung
	Bit setzen ☞" ■ Bit setzen" (seite 21-79) Ändert die angegebene Bitadresse von 0 –> 1.
	Bit löschen ☞" ■ Bit löschen" (seite 21-79) Ändert die angegebene Bitadresse von 1 –> 0.
	Bit invertieren ☞" ■ Bit invertieren" (seite 21-79) Ändert die angegebene Bitadresse von 1 –> 0 oder von 0 –> 1.

■ Bit setzen

Element	Beschreibung
Zusammen-fassung	Ändert die angegebene Bitadresse von 0 –> 1.
Format	set ( )

Beispielausdruck:

set ([b:[#INTERNAL]LS010000])

Im o.g. Beispiel wird das 00ste Bit des LS0100 von 0 –> 1 geändert.

■ Bit löschen

Element	Beschreibung
Zusammen-fassung	Ändert die angegebene Bitadresse von 1 –> 0.
Format	clear ( )

Beispielausdruck:

clear ([b:[#INTERNAL]LS010000])

Im o.g. Beispiel wird das 00ste Bit des LS0100 von 1 –> 0 geändert.

■ Bit invertieren

Element	Beschreibung
Zusammen-fassung	Ändert die angegebene Bitadresse von 1 –> 0 oder von 0 –> 1.
Format	toggle ( )

Beispielausdruck:

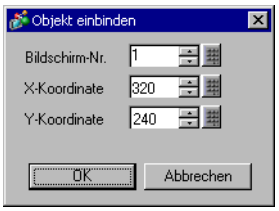
toggle ([b:[#INTERNAL]LS010000])

Im o.g. Beispiel wird das 00te Bit von 1 –> 0 oder von 0 –> 1 geändert.

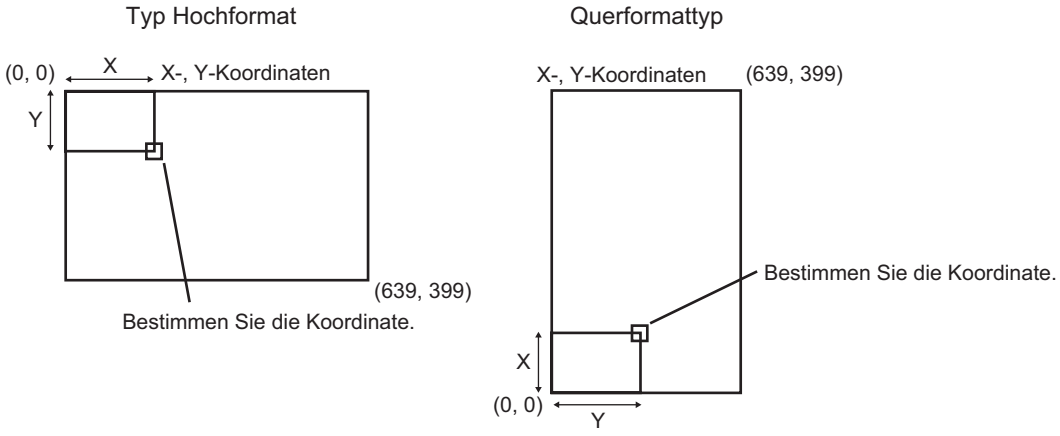
21.11.2 Zeichnen

Zeichnen	Funktionszusammenfassung
	Objekt einbinden ☞" ■ Bildschirm aufrufen" (seite 21-80) Ruft den Bildschirm (Basisbildschirm) mit der bezeichneten Bildschirmnummer auf. Kann nicht in einem erweiterten Skript verwendet werden.
	Kreis ☞" ■ Linie" (seite 21-82) Zeichnet den angegebenen Kreis.
	Punkt ☞" ■ Punkt" (seite 21-82) Zeichnet den angegebenen Punkt.
	Linie ☞" ■ Linie" (seite 21-82) Zeichnet die angegebene Linie.
	Rechteck ☞" ■ Rechteck" (seite 21-83) Zeichnet das angegebene Rechteck.

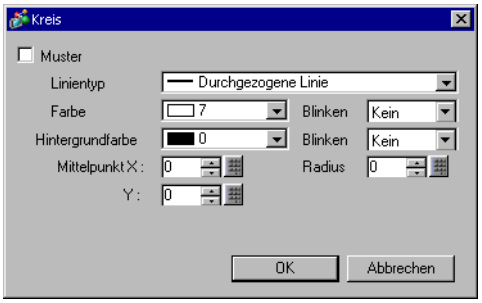
■ Bildschirm aufrufen

Element	Beschreibung
Zusammen-fassung	Mit dieser Funktion wird ein registriertes Bibliothekselement abgerufen. Der bestimmte Bildschirm (Basisbildschirm) wird an den angegebenen X-, Y-Koordinaten aufgerufen. Kann nicht in einem erweiterten Skript verwendet werden.
Format	b_call (Bildschirm-Nr., X-Koordinate, Y-Koordinate)  **ANMERKUNG** • Legen Sie die Mittelkoordinate des aufgerufenen Bildschirms mittels der X-Koordinate und Y-Koordinate fest.

Koordinaten-Position



■ Kreis

Element	Beschreibung
Zusammen-fassung	Zeichnet einen Kreis um den angegebenen Punkt herum. Wenn das Optionsfeld [Muster] ausgewählt wird, wird ein gefüllter Kreis gezeichnet. Wählen Sie aus und geben Sie den Linientyp (oder das Füllmuster bei Wahl eines Musters), die Farbeigenschaften, Mittelkoordinaten und den Radius-Wert ein. Die Mittelkoordinaten und der Radius lassen sich indirekt bestimmen.
Format	dsp_circle (X-Koordinate, Y-Koordinate, Radius, Farbblinken anzeigen + Anzeigenfarbe, Blinken der Hintergrundfarben + Hintergrundfarben, Linientyp)  **ANMERKUNG** • Bei gleichzeitigem Setzen von Schwarz und Blinken erscheint die Hintergrundfarbe transparent.

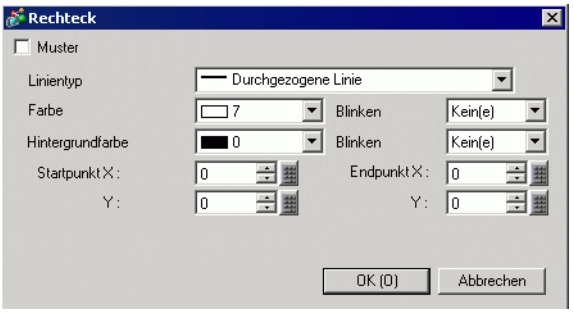
■ Punkt

Element	Beschreibung
Zusammen-fassung	Zeichnet einen Punkt an der angegebenen Position. Bestimmen Sie die X-, Y-Koordinaten und die Anzeigenfarben
Format	dsp_dot (X-Koordinate, Y-Koordinate, Blinken + Anzeigenfarbe)  **ANMERKUNG** • Bei gleichzeitigem Setzen von Schwarz und Blinken erscheint die Hintergrundfarbe transparent.

■ Linie

Element	Beschreibung
Zusammen-fassung	Zeichnet eine Linie an der angegebenen Position. Bestimmen Sie den Typ, die Farbmerkmale sowie die Start- und Endkoordinaten der Linie.
Format	dsp_line (Startpunkt X-Koordinate, Startpunkt Y-Koordinate, Endpunkt X-Koordinate, Endpunkt Y-Koordinate, Farbblinken anzeigen + Anzeigenfarbe, Blinken der Hintergrundfarbe + Hintergrundfarbe, Linientyp und Pfeil)  **ANMERKUNG** • Bei gleichzeitigem Setzen von Schwarz und Blinken erscheint die Hintergrundfarbe transparent.

■ Rechteck

Element	Beschreibung
Zusammen-fassung	Zeichnet ein Rechteck an der angegebenen Position. Wenn das Optionsfeld [Muster] ausgewählt wird, wird ein gefülltes Rechteck gezeichnet. Wählen Sie aus und geben Sie den Linientyp (oder das Füllmuster bei Wahl eines Musters), die Farbeigenschaften sowie die Start- und Endkoordinaten ein.
Format	dsp_rectangle (Startpunkt X-Koordinate, Startpunkt Y-Koordinate, Endpunkt X-Koordinate, Endpunkt Y-Koordinate, Farbblinken anzeigen + Anzeigenfarbe, Blinken der Hintergrundfarbe + Hintergrundfarbe, Muster und Linientyp)  **ANMERKUNG** • Bei gleichzeitigem Setzen von Schwarz und Blinken erscheint die Hintergrundfarbe transparent.

WICHTIG

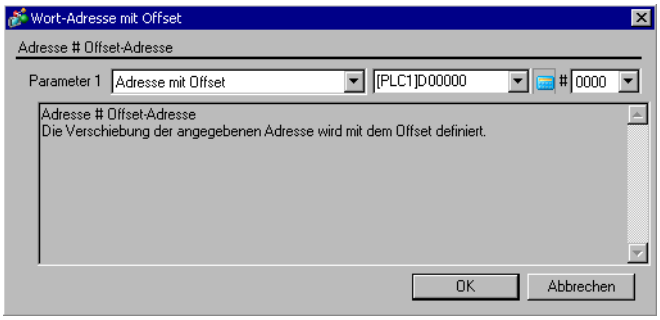
- Bei Verwendung von Farben in den Zeichen-Funktionen werden die Farb-Codes von 0 bis 255 gesetzt. Wenn E1 bis E12 gesetzt und das Skript gespeichert wird, tritt ein Fehler auf.

21.11.3 Speicher-Operationen

Speicher-Operationen	Funktionszusammenfassung
	Wort-Adresse mit Offset ☞" ■ Wortadresse mit Offset" (seite 21-85) Legt ein Adressen-Offset fest.
	Speicher vergleichen ☞" ■ Speicher vergleichen" (seite 21-87) Vergleicht zwei Datenblöcke an den angegebenen Positionen (Offset) und schreibt das Vergleichsergebnis in die Speicheradresse.
	Speicher kopieren ☞" ■ Speicher kopieren" (seite 21-89) Kopiert Teilnehmerspeicher in einem Vorgang.
	Speicher kopieren (Variablen-Spezifikation) ☞" ■ Speicher kopieren (Variable)" (seite 21-93) Kopiert Teilnehmerspeicher in einem Vorgang. Die Quell- (Kopierquell-) Adresse, Ziel- (Kopierziel-) Adresse und die Anzahl der Adressen kann modifiziert werden.
	Speicherring ☞" ■ Speicherring" (seite 21-94) Ring-Verschieben der gespeicherten Daten um die angegebene Anzahl von Wort-Blöcken.
	Speicher durchsuchen ☞" ■ Speicher durchsuchen" (seite 21-97) Führt eine Datensuche in Block-Einheiten aus und gibt (speichert) das Suchergebnis in die angegebene Speicheradresse aus.
	Speicher initialisieren ☞" ■ Speicher initialisieren" (seite 21-100) Initialisiert alle Teilnehmer gleichzeitig.
	Speicher initialisieren (Variablen-Spezifikation) ☞" ■ Speicher initialisieren(Variable)" (seite 21-101) Initialisiert alle Teilnehmer gleichzeitig. Die die obere Adresse, gesetzte Daten sowie die Anzahl der Adressen kann modifiziert werden.
	Schiebespeicher ☞" ■ Schiebespeicher" (seite 21-102) Verschiebt Blockeinheiten nach oben.

■ Wortadresse mit Offset

Element	Beschreibung
Zusammen-fassung	Wortadressen mit Offset können ausgewiesen werden. Nur temporäre Wortadressen können für Offset-Wert-Speicheradressen ausgewiesen werden.
Format	[interne Adresse] # [Wortadresse mit Offset]



Eingangsbereiche der Konstante

Datentyp	Eingabe der Konstante	
	Min.	Max.
Bin16	0	65535
Bin32	0	4294967295
Bin16+/–	–32768	32767
Bin32+/–	–2147483648	2147483647
BCD16	0	9999
BCD32	0	99999999

Beispielausdruck 1:

[w:[PLC1]D0200]=[w:[PLC1]D0100]#[t:0000]

Wenn der Wert von [t:0000] im o.g. Beispiel 2 beträgt, wird der in D0102 gespeicherte Wert zu D0200 versetzt.

Beispielausdruck 2:

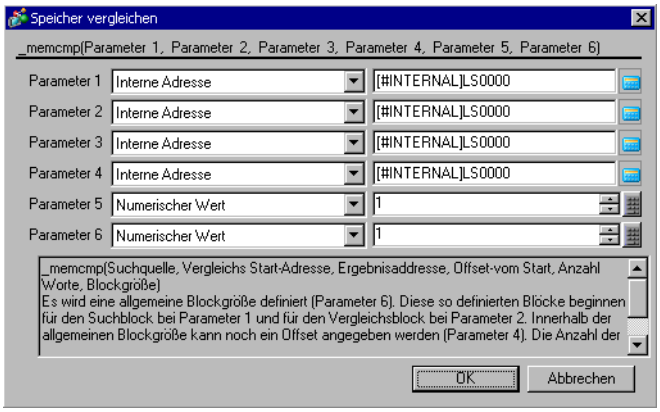
[w:[PLC1]D0100]#[t:0000]=30

Wenn der Wert von [t:0000] im o.g. Beispiel 8 beträgt, wird 30 zu D0108 versetzt.

WICHTIG

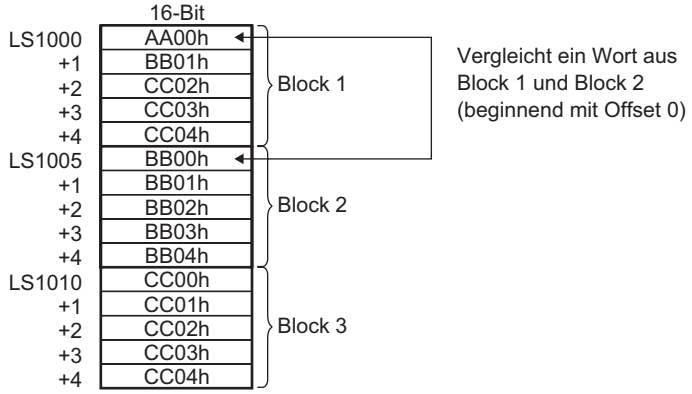
- Im Offset-Adress-Format verwendete Wortadressen zählen nicht als D-Skript-Adressen.
- Daten von einem nicht durch eine Wortadresse mit Offset gekennzeichneten Teilnehmer werden nicht kontinuierlich von dem verbundenen Teilnehmer ausgelesen. Sie werden gelesen, wenn ein D-Skript ausgeführt wird. Wenn ein Fehler während des Auslesens auftritt, wird der Auslesewert als "0" behandelt. Außerdem stellt sich das Bit 12 in der GP-Einheit des internen Sonderrelais LS2032 EIN. Bei ordnungsgemäßen Abschluss des Daten-Auslesens wird Bit-12 AUS geschaltet.
- Falls das Ergebnis des Adress-Offsets 16-Bit überschreitet (max. Wert: 65535), sind Bits bis zu 15 gültig und Bits 16 und höher werden gestrichen.
- Wenn Sie eine Variable als Adresse bestimmen, legen Sie bitte ein Ganzzahl-Feld fest. Stellen Sie sicher, dass ein Ganzzahl-Feld groß genug ist, um alle konsekutiven Adressen unterbringen zu können. Operationen werden ungültig, wenn das Feld nicht groß genug ist, um alle konsekutiven Adressen zu speichern. Operationen sind außerdem ungültig, wenn eine Ganzzahlvariable kein Feld ist.

■ Speicher vergleichen

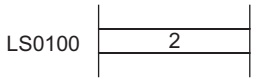
Element	Beschreibung
Zusammen-fassung	Vergleicht zwei Datenblöcke an den angegebenen Positionen (Offset) und schreibt das Vergleichsergebnis in die Speicheradresse. Die folgenden Werte werden als Vergleichsergebnis gespeichert: Bei gleichwertigen Werten: Wenn die Zieldaten größer als die Originaldaten sind: Wenn die Zieldaten kleiner als die Originaldaten sind: Bei Auftreten eines Fehlers wird der Fehlerstatuswert in LS9152 geschrieben.
Format	_memcmp ([Verglichene Blockadresse], [Mit Blockadresse vergleichen], [Speicheradresse des Vergleichsergebnisses], Abstand zum Anfang des Blocks, Anzahl der verglichenen Worte, Anzahl der Worte in 1 Block)  Parameter 1: Interne Adresse Parameter 2: Interne Adresse Parameter 3: Interne Adresse Parameter 4: Numerischer Wert (0 bis 639), Interne Adresse, temporäre Variable Parameter 5: Numerischer Wert (1 bis 640) Parameter 6: Numerischer Wert (1 bis 640) Zu speichernde Daten 0: Übereinstimmung 1: Vergleichen von < Vergleichen bis 2: Vergleichen von > Vergleichen bis

Beispielausdruck 1:

_memcmp ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005],[w:[#INTERNAL]LS0100], 0, 1, 5)
(Vergleicht ein Wort aus Block 1 und Block 2 (beginnend mit Offset 0) und speichert das Vergleichsergebnis in LS0100.)

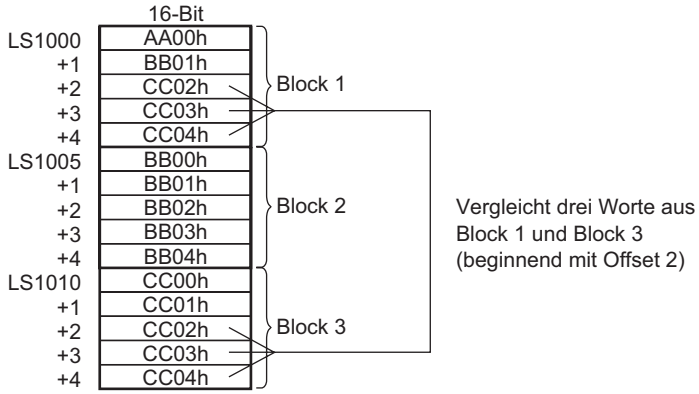


Da der Quellwert kleiner als der Zielwert ist, wird das Vergleichsergebnis "2" in LS0100 gespeichert.

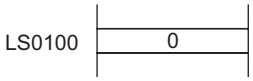


Beispielausdruck 2:

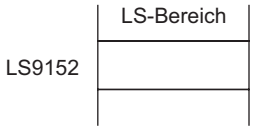
_memcmp ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1010], [w:[#INTERNAL]LS0100], 2, 3, 5)
(Vergleicht ein Wort aus Block 1 und Block 3 (beginnend mit Offset 2) und speichert das Vergleichsergebnis in LS0100.)



Da die Werte der Original- und Zieldaten miteinander übereinstimmen, wird das Vergleichsergebnis "0" in LS0100 gespeichert.



Fehlerzustand

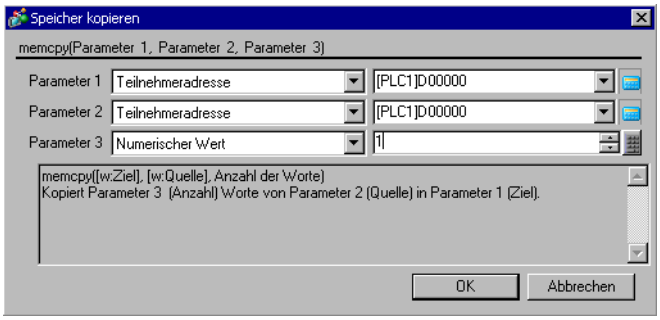


Editor-Funktionsname	LS-Bereich	Fehlerstatus	Ursache
_memcmp ( )	LS9152	0000h	Erfolgreich abgeschlossen.
		0001h	Parameter-Fehler
		0003h	Schreibfehler/Lesefehler

WICHTIG

- Der tatsächliche LS-Teilnehmerbereich, der spezifiziert werden kann, ist begrenzt auf den vorgesehenen Benutzerbereich (LS20 bis LS2031 sowie LS2096 bis LS8191).
- Bei Angabe eines Wertes der die Anzahl der Worte in einem Block überschreitet, funktioniert diese Funktion nicht ordnungsgemäß.
- Wenn die Anzahl der zu vergleichenden Worte größer ist als ein Block, funktioniert diese Funktion nicht ordnungsgemäß.

■ Speicher kopieren

Element	Beschreibung
Zusammen-fassung	Kopiert Teilnehmerspeicher in einem Vorgang. Daten für die Anzahl der Adressen werden in die Kopierziel-Wortadressen kopiert, beginnend mit der ersten Wortadresse der Quelldaten. Die zulässige Anzahl der Adressen liegt zwischen 1 und 640.
Format	memcpy ([In Adresse kopieren], [Aus Adresse kopieren], Worte) 

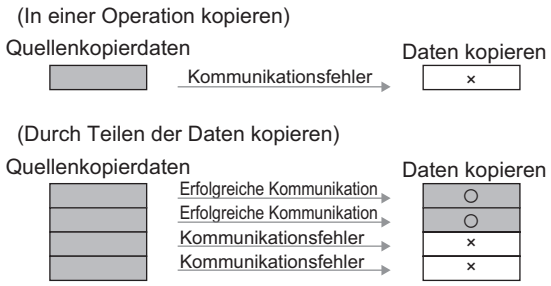
Beispielausdruck:

memcpy ([w:[PLC1]D0200], [w:[PLC1]D0100], 10)

Im o.g. Beispiel werden Daten von D0100 - D0109 bis D0200 - D0209 kopiert.

WICHTIG

- Quellkopierdaten werden bei Bedarf von dem verbundenen Teilnehmer nur einmal gelesen. Bei Auftreten eines Kommunikationsfehlers während des Lesens der Daten geht Bit 12 des internen Sonderrelais LS2032 AN. Bei ordnungsgemäßen Abschluss des Daten-Auslesens wird Bit-12 AUS geschaltet.
- Daten werden in einem Vorgang von den Quellkopierdaten gelesen und in das Ziel geschrieben oder durch die Division der Daten in genauso viele Teile wie die Anzahl der für die Quellkopierdaten verwendeten Adressen geteilt. Bei Auftreten eines Kommunikationsfehlers während des Lesens der Daten unterscheidet sich das Ergebnis des Datenkopierens folgendermaßen und abhängig davon, ob die Daten in einer Operation oder in mehreren Abschnitten verarbeitet wurden. (Ergebnis von Datenschreiboperation O: Schreiben komplett, X: Schreiben nicht möglich)

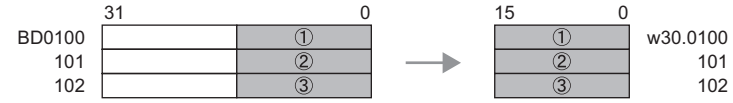


- Mit zunehmender Adressenanzahl ist das Schreiben der Daten in das SPS mit einem höheren Zeitaufwand verbunden. Abhängig von der Adressenanzahl kann der Vorgang von 20 Sekunden bis zu mehreren Minuten in Anspruch nehmen.
- Wenn die zu schreibenden Daten den angegebenen Teilnehmerbereich überschreiten, tritt ein Kommunikationsfehler auf. In diesem Fall muss die Stromzufuhr zur GP-Einheit AUS geschaltet und dann wieder AN geschaltet werden, um die GP wieder rückzusetzen.
- Wenn die Daten mit der Speicherkopierfunktion (memcpy) in den LS-Bereich geschrieben werden, können die Daten nur im Benutzerbereich geschrieben werden. Daten können nicht in den Systemdatenbereich (LS0000 bis LS0019), den Sonderbereich (LS2032 bis LS2047) oder den reservierten Bereich (LS2048 bis LS2095) geschrieben werden. Daten aus diesen Bereichen können jedoch gelesen werden.

Fortsetzung

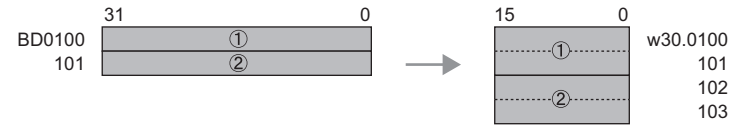
WICHTIG

- Wenn ein D-Skript zum Kopieren von 32-Bit-Teilnehmerdaten auf einen 16-Bit-Teilnehmer kopiert werden und die Bitlänge mit 16 Bits angegeben ist, werden nur die Daten in den unteren 16 Bits kopiert.
  Beispiel: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 3)

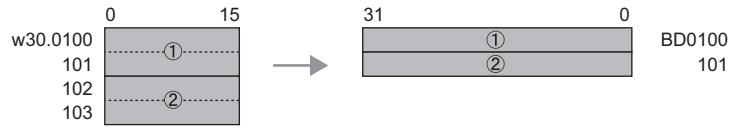


  Wenn man ferner 16-Bit Teilnehmerdaten in einen 32-Bit Teilnehmer kopiert, werden nur die Daten für die unteren 16-Bit kopiert und die oberen 16-Bit mit "0" ausgewiesen.
  Beispiel: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 3)

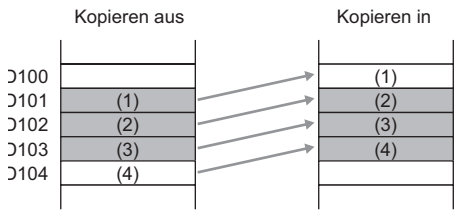
- Wenn 32-Bit Teilnehmerdaten in einen 16-Bit Teilnehmer oder 16-Bit Teilnehmerdaten in einen 32-Bit Teilnehmer kopiert werden, und wenn die im D-Skript angegebene D-Skript-Bitlänge 32 beträgt, wird folgendermaßen kopiert. Wenn es sich bei dem einen Teilnehmer um einen 32-Bit-Teilnehmer und bei dem anderen um einen 16-Bit-Teilnehmer handelt, wird die memcpy-Funktion ( ) 16 Bit als ihren Datenlängen-Parameter verwenden.
  Beispiel: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 4)



  Beispiel: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 4)



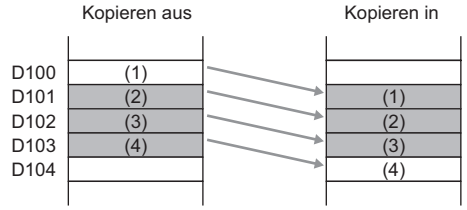
- Bei Überlappen der originalen und Ziel-Datenbereiche werden sämtliche überlappende Daten folgendermaßen überschrieben:
  Beispiel: Beim Kopieren von D101-D104 to D100-D103
  Daten werden in eine kleinere Nummernadresse kopiert.



Fortsetzung

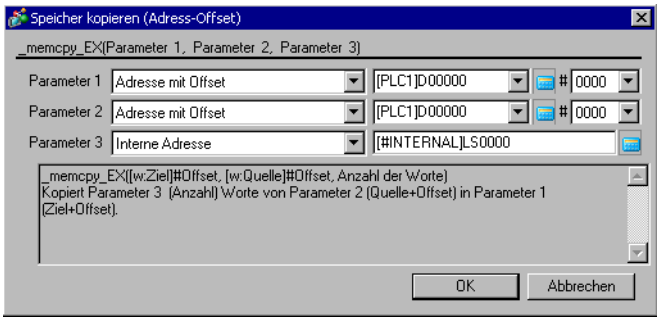
WICHTIG

Beispiel: Beim Kopieren von D100-D103 to D101-D104
Daten werden in eine größere Nummernadresse kopiert.



- Obgleich die Funktion in diesem Beispiel 2 Adressen bezeichnet, zählen jene nicht als D-Skript-Adressen.
- Wenn eine Teilnehmeradresse zur Zuweisung verwendet wird, führt die Kommunikation mit dem Teilnehmer/mit der SPS zu einer geringen Verzögerung bei der Zuweisung des Wertes.

■ Speicher kopieren (Variable)

Element	Beschreibung
Zusammen-fassung	Kopiert Teilnehmerspeicher in einem Vorgang. Die mit Parameter 3 spezifizierten Adressendaten werden von der mit Parameter 2 spezifizierten Kopierquell-Wortadresse in die mit Parameter 1 spezifizierte Kopierziel-Wortadresse kopiert. Die Anzahl der verwendbaren Adressen reicht von 1 bis 640. Mit der Funktion "_memcpy_EX" können die Quell- und Zieladressen sowie die Anzahl der Adressen indirekt festgelegt werden.
Format	_memcpy_EX ([Kopierzieladresse], [Kopierquelladresse], Worte) Parameter 1: Teilnehmeradresse + Temporäre Adresse Parameter 2: Teilnehmeradresse + Temporäre Adresse Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 640.) 

Beispielausdruck:

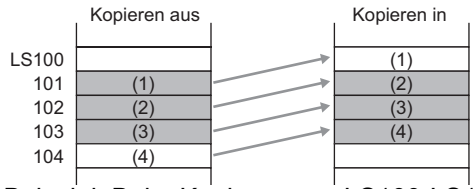
[t:0000]=10, [t:0001]=20

_memcpy_EX ([w:[#INTERNAL]LS0100]#[t:0000], [w:[PLC1]D0100]#[t:0001], 5)

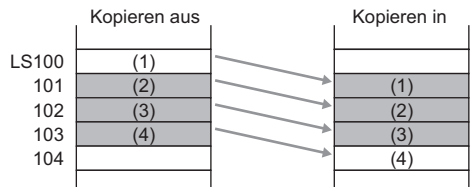
Im o.g. Beispiel werden fünf Worte an Daten von D0120 ausgelesen und in LS0110 bis LS0114 geschrieben.

WICHTIG

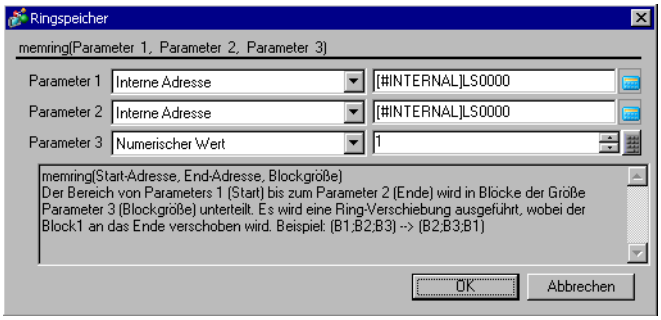
- Bei Überlappen der originalen und Ziel-Datenbereiche werden sämtliche überlappende Daten folgendermaßen überschrieben:
  Beispiel: Beim Kopieren von LS101-LS104 in LS100-LS103
  Daten werden in eine kleinere Nummernadresse kopiert.



Beispiel: Beim Kopieren von LS100-LS103 to LS101-LS104
Daten werden in eine größere Nummernadresse kopiert.

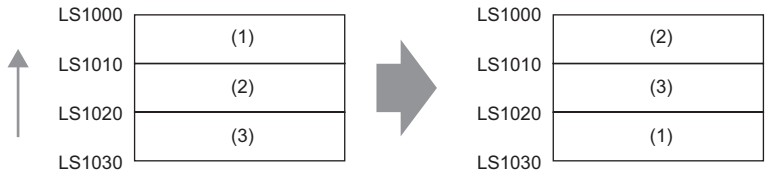


■ Speicherring

Element	Beschreibung
Zusammen-fassung	Ring-Verschieben der gespeicherten Daten in Blöcken. Führt Ring-Verschieben zwischen der Start- und End-Adresse in Blockeinheiten aus (nach angegebener Wortanzahl). Bei Auftreten eines Fehlers, wird der Fehlerzustand in LS9150 geschrieben.
Format	memring ([Startadresse], [End-Adresse], Worte in 1 Block)  Parameter 1: Interne Adresse Parameter 2: Interne Adresse Parameter 3: Numerischer Wert (1 bis 640) Wenn Parameter 1 größer als Parameter 2 ist (P1 < P2), werden die Blockdaten nach oben verschoben. Wenn Parameter 1 größer als Parameter 2 ist (P1 > P2), werden die Blockdaten nach unten verschoben. WICHTIG • Stellen Sie sicher, dass die Start- und End-Adresse auf dieselbe Teilnehmerart gesetzt sind (LS oder USR).

Beispielausdruck 1:

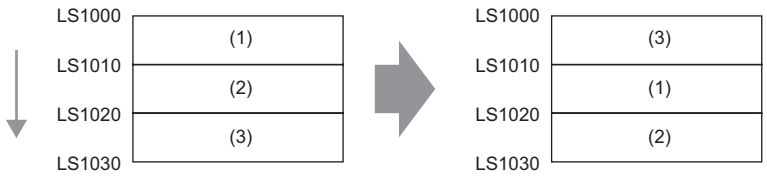
memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1030], 10)
(Wenn Parameter 1 größer als Parameter 2 ist (P1 < P2))



Daten werden in 10-Wort-Blockeinheiten nach oben verschoben.

Beispielausdruck 2:

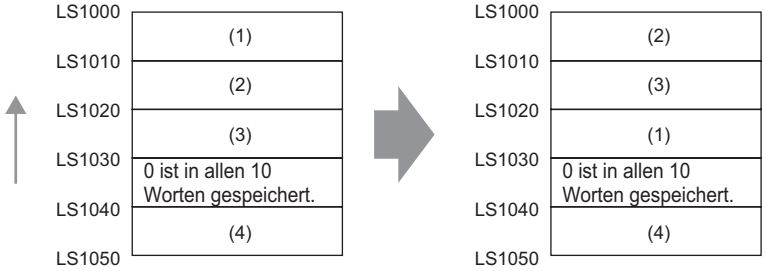
memring ([w:[#INTERNAL]LS1030], [w:[#INTERNAL]LS1000], 10)
(Wenn Parameter 1 größer als Parameter 2 ist (P1 > P2))



Daten werden in 10-Wort-Blockeinheiten nach unten verschoben.

Beispielausdruck 3:

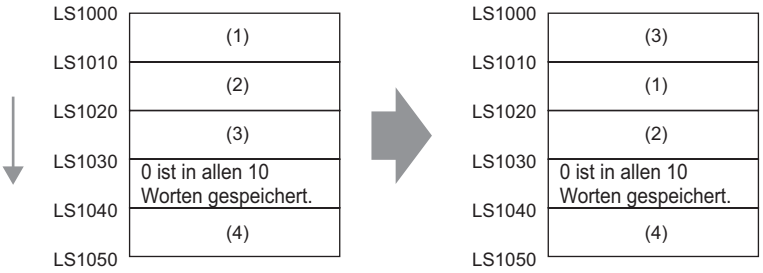
memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1050], 10)
(Wenn der Bereich einen Block enthält, in dem alle Worte "0" sind.)



Daten werden nur in 10-Wort-Blockeinheiten nach oben verschoben, beginnend mit dem Start-Block bis zu Block mit "0" Daten. Falls es Daten nach dem Block mit "0" Daten gibt, werden diese ignoriert.

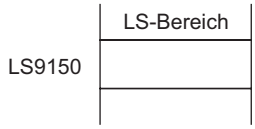
Beispielausdruck 4:

memring ([w:[#INTERNAL]LS1050], [w:[#INTERNAL]LS1000], 10)
(Bei Vorhandensein eines Blocks mit "0" Daten innerhalb des Bereichs.)



Daten werden nur in 10-Wort-Blockeinheiten nach unten verschoben, beginnend mit dem Start-Block bis zu Block mit "0" Daten. Falls es Daten nach dem Block mit "0" Daten gibt, werden diese ignoriert.

Fehlerzustand

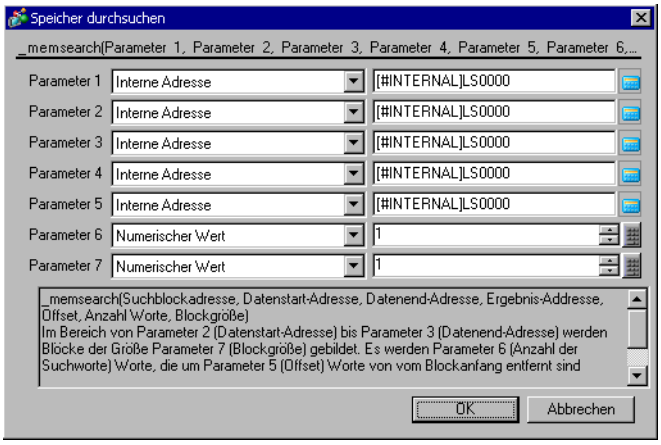


Editor-Funktionsname	LS-Bereich	Fehlerstatus	Ursache
memring ( )	LS9150	0000h	Erfolgreich abgeschlossen.
		0001h	Parameter-Fehler
		0003h	Schreibfehler/Lesefehler

WICHTIG

- Die erforderliche Verarbeitungszeit ist proportional zu dem mit der Start- und Endadresse gekennzeichneten Bereich. Je größer der gekennzeichnete Bereich, desto länger die Bearbeitungszeit. Das Element wird erst nach Abschluss der Bearbeitung aktualisiert.
- Der tatsächliche LS-Teilnehmerbereich, der spezifiziert werden kann, ist begrenzt auf den vorgesehenen Benutzerbereich (LS20 bis LS2031 sowie LS2096 bis LS8191).

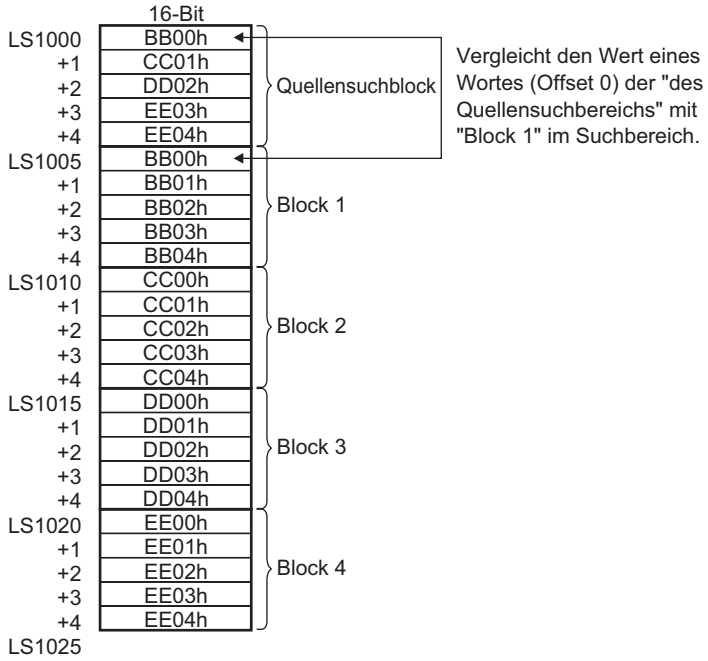
■ Speicher durchsuchen

Element	Beschreibung
Zusammen-fassung	Führt eine Datensuche in Block-Einheiten aus, beginnend mit dem ersten Element im angegebenen Bereich. Vergleicht Datenblöcke, beginnend an den angegebenen (Offset) Blöcken und gibt (speichert) das Suchergebnis an die angegebene Speicheradresse aus. Bei Finden eines übereinstimmenden Blocks wird der Offset-Wert des Blocks (1 oder höher) gespeichert. Wenn kein übereinstimmender Block gefunden wurde, wird "FFFFh" gespeichert. Bei Auftreten eines Fehlers wird der Fehlerstatuswert in LS9151 geschrieben.
Format	_memsearch ([Abgesuchte Blockadresse], [Start-Adresse der Suche], [End-Adresse der Suche], [Suchergebnis-Speicheradresse], Abstand zum Anfang des Blocks, Anzahl der verglichenen Wort, Anzahl der Worte in 1 Block)  Parameter 1: Interne Adresse Parameter 2: Interne Adresse Parameter 3: Interne Adresse Parameter 4: Interne Adresse Parameter 5: Numerischer Wert (0 bis 639), Interne Adresse, temporäre Variable Parameter 6: Numerischer Wert (1 bis 640) Parameter 7: Numerischer Wert (1 bis 640) Zu schreibende Daten Wenn es übereinstimmende Blöcke gibt: Der Offset-Wert des Blocks ("1" oder höher) Wenn es keine übereinstimmenden Blöcke gibt: "FFFFh" WICHTIG • Stellen Sie sicher, dass die Start- und End-Adresse der Suche auf dieselbe Teilnehmerart gesetzt sind (LS oder USR). [Gesuchte Blockadresse] und [Speicheradresse des Suchergebnisses] können jedoch auf die interne Adresse gesetzt werden. • Vergewissern Sie sich bitte, dass [Parameter 2] kleiner ist als [Parameter 3]. Andernfalls tritt ein Fehler auf.

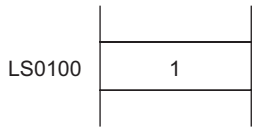
Beispielausdruck 1:

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025], [w:[#INTERNAL]LS0100], 0, 1, 5)

(Suchen von LS1005 bis LS1025 nach einem gleichwertigen Block. Beginnt am Offset 0 des Quellsuchblocks und speichert das Ergebnis in LS0100.)



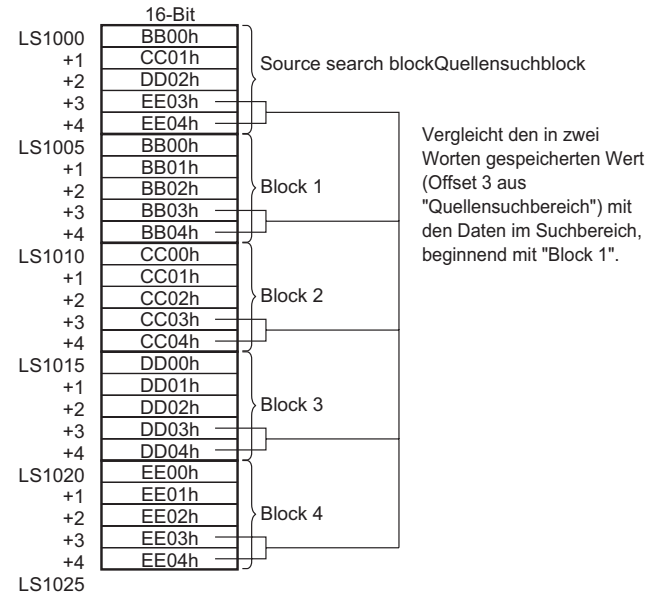
In diesem Fall stimmt der Wert "Block 1" mit dem Wert des "Quellensuchblocks" überein. Dementsprechend wird das Suchergebnis "1" in LS0100 gespeichert.



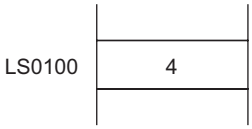
Beispielausdruck 2:

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025], [w:[#INTERNAL]LS0100], 3, 2, 5)

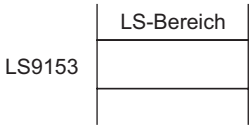
(Suchen von LS1005 bis LS1025 nach einem gleichwertigen Block. Bedient sich zweier Worte, beginnend an einem Offset von 3, und speichert das Ergebnis in LS0100.)



In diesem Fall stimmt der Wert "Block 4" mit dem Wert des "Quellensuchblocks" überein. Dementsprechend wird das Suchergebnis "4" in LS0100 gespeichert.



Fehlerzustand

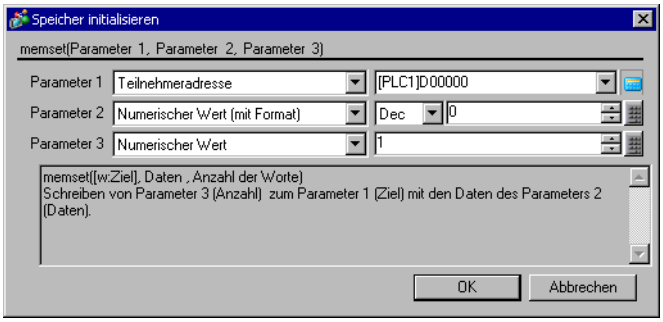


Editor-Funktionsname	LS-Bereich	Fehlerstatus	Ursache
_memsearch ( )	LS9153	0000h	Erfolgreich abgeschlossen.
		0001h	Parameter-Fehler
		0003h	Schreibfehler/Lesefehler

WICHTIG

- Die erforderliche Verarbeitungszeit ist proportional zu dem mit der Start- und Endadresse gekennzeichneten Bereich. Je größer der gekennzeichnete Bereich, desto länger die Bearbeitungszeit. Das Element wird erst nach Abschluss der Bearbeitung aktualisiert.
- Der tatsächliche LS-Teilnehmerbereich, der spezifiziert werden kann, ist begrenzt auf den vorgesehenen Benutzerbereich (LS20 bis LS2031 sowie LS2096 bis LS8191).

■ Speicher initialisieren

Element	Beschreibung
Zusammen-fassung	Initialisiert alle Teilnehmer gleichzeitig. Einstellungsdaten für die Anzahl der Adressen werden von der festgelegten Wortadresse erfasst. Der gültige Bereich für die Anzahl von Adressen ist von 1 bis 640.
Format	memset ([Schreibzieladresse], Daten schreiben, Worte) 

Beispielausdruck:

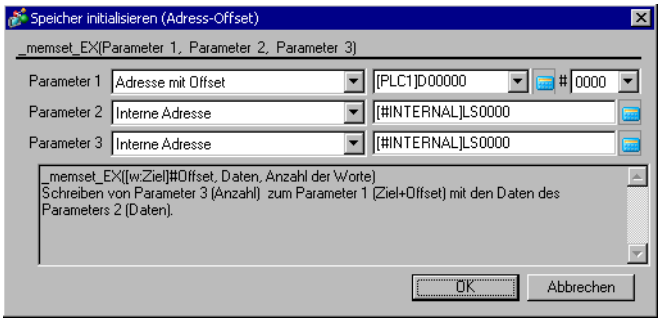
memset ([w:[SPS 1]D0100], 0, 10)

Im o.g. Beispiel wird "0" für die Adressen D0100 bis D0109 gesetzt.

WICHTIG

- Mit zunehmender Adressenanzahl ist das Schreiben der Daten in das SPS mit einem höheren Zeitaufwand verbunden. Abhängig von der Adressenanzahl kann der Vorgang von 20 Sekunden bis zu mehreren Minuten in Anspruch nehmen.
- Wenn die zu schreibenden Daten den angegebenen Teilnehmerbereich überschreiten, tritt ein Kommunikationsfehler auf. In diesem Fall muss die Stromzufuhr zur GP-Einheit AUS geschaltet und dann wieder AN geschaltet werden, um die GP wieder rückzusetzen.
- Obgleich diese Funktion Adressen bezeichnet, werden jene nicht als D-Skript-Adressen gezählt.
- Wenn die Daten mit der Speicherrücksetzfunktion (memset) in den LS-Bereich geschrieben werden, können die Daten nur in den Benutzerbereich geschrieben werden. Daten können nicht in den Systemdatenbereich (LS0000 bis LS0019), den Sonderbereich (LS2032 bis LS2047) oder den reservierten Bereich (LS2048 bis LS2095) geschrieben werden.
- Bei Verwendung von Teilnehmeradressen für das Zuweisungsverfahren werden die Werte aufgrund der Übertragungszeit zwischen GP und SPS nicht sofort zugewiesen.
  Zum Beispiel:
  memset ([w:[PLC1]D0100], 0, 10) //Initialize D100 to D109 to 0
  [w:[PLC1]D200]=[w:[PLC1]D100]//Ersetzt D100 bis D200
  In diesem Fall wird das in D100 geschriebene Operationsergebnis 0 noch nicht D200 zugeordnet.

■ Speicher initialisieren(Variable)

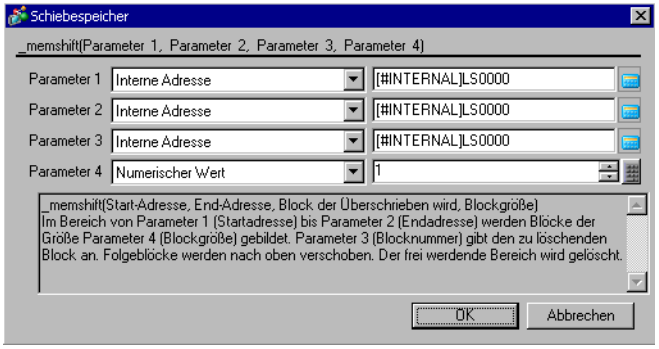
Element	Beschreibung
Zusammen-fassung	Initialisiert alle Teilnehmer gleichzeitig. Die mit Parameter 2 gesetzten Daten werden von der mit Parameter 1 gesetzten Wortadresse in die mit Parameter 3 gesetzten Wortadressen gesetzt. Der gültige Bereich für die Anzahl der Adressen ist von 1 bis 640. Die Schreibzieladresse, die Schreibdaten sowie die Anzahl der Adressen kann indirekt festgelegt werden.
Format	_memset_EX ([Schreibzieladresse], Daten schreiben, Worte)  Parameter 1: Teilnehmeradresse + Temporäre Adresse Parameter 2: Numerischer Wert, interne Adresse, temporäre Adresse (der gültige Bereich für Parameter 2 reicht von 0 bis 65535 für Dez. und von 0 bis FFFF für Hex.) Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 640.)

Beispielausdruck:

[t:0000]=10
[w:[#INTERNAL]LS0050]=0
[w:[#INTERNAL]LS0050]=5
_memset_EX ([w:[#INTERNAL]LS0100]#[t:0000], [w:[#INTERNAL]LS0050], [w:[#INTERNAL]LS0051])

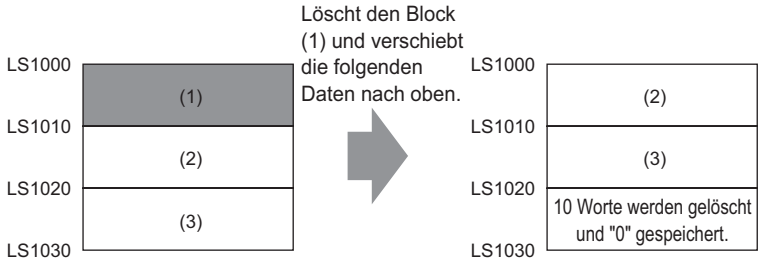
Im o.g. Beispiel wird "0" in die fünf Worte von LS0100 bis LS0114 geschrieben.

■ Schiebespeicher

Element	Beschreibung
Zusammen-fassung	Löscht den angegebenen Block und verschiebt die folgenden Datenblöcke nach oben. Der zu löschende Block wird mit einem Offset gekennzeichnet. Bei Auftreten eines Fehlers wird der Fehlerstatus in LS9151 geschrieben.
Format	_memshift ([Start-Adresse], [End-Adresse], Abstand zwischen Block und Entfernen, Anzahl der Worte in 1 Block)  Parameter 1: Interne Adresse Parameter 2: Interne Adresse Parameter 3: Numerischer Wert (1 bis 65.535), Interne Adresse, temporäre Variable Parameter 4: Numerischer Wert (1 bis 640) WICHTIG • Stellen Sie sicher, dass die Start- und End-Adresse auf dieselbe Teilnehmerart gesetzt sind (LS oder USR). • Vergewissern Sie sich bitte, dass [Parameter 1] kleiner ist als [Parameter 2]. Andernfalls tritt ein Fehler auf.

Beispielausdruck 1:

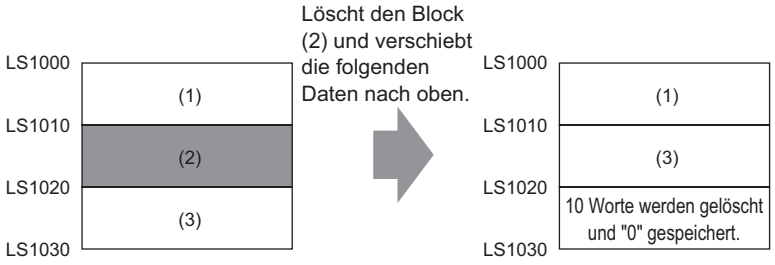
_memshift ([w:[#INTERNAL]LS 1000], [w:[#INTERNAL]LS1030], 1, 10)



Daten werden in 10-Wort-Blockeinheiten (1 Block = 10 Worte) nach oben verschoben, und der letzte Block (10 Worte) wird auf "0" gelöscht.

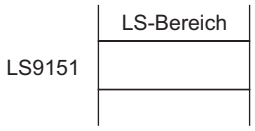
Beispielausdruck 2:

_memshift ([w:[#INTERNAL]LS 1000], [w:[#INTERNAL]LS1030], 2, 10)



Die Daten bewegen sich um Blockeinheiten (1 Block = 10 Worte) nach oben, beginnend an der Offset 2 Position, und der letzte Block (10 Worte) wird auf null gelöscht.

Fehlerzustand



Editor-Funktionsname	LS-Bereich	Fehlerstatus	Ursache
_memshift ( )	LS9151	0000h	Erfolgreich abgeschlossen.
		0001h	Parameter-Fehler
		0003h	Schreibfehler/Lesefehler

WICHTIG

- Die erforderliche Verarbeitungszeit ist proportional zu dem mit der Start- und Endadresse gekennzeichneten Bereich. Je größer der gekennzeichnete Bereich, desto länger die Bearbeitungszeit. Das Element wird erst nach Abschluss der Bearbeitung aktualisiert.
- Bei Angabe eines Wertes als Offset des zu löschenden Blocks, der den für die Start- und Endadresse bestimmten Bereich überschreitet, funktioniert diese Funktion nicht ordnungsgemäß.
- Der tatsächliche LS-Teilnehmerbereich, der spezifiziert werden kann, ist begrenzt auf den vorgesehenen Benutzerbereich (LS20 bis LS2031 sowie LS2096 bis LS8191).

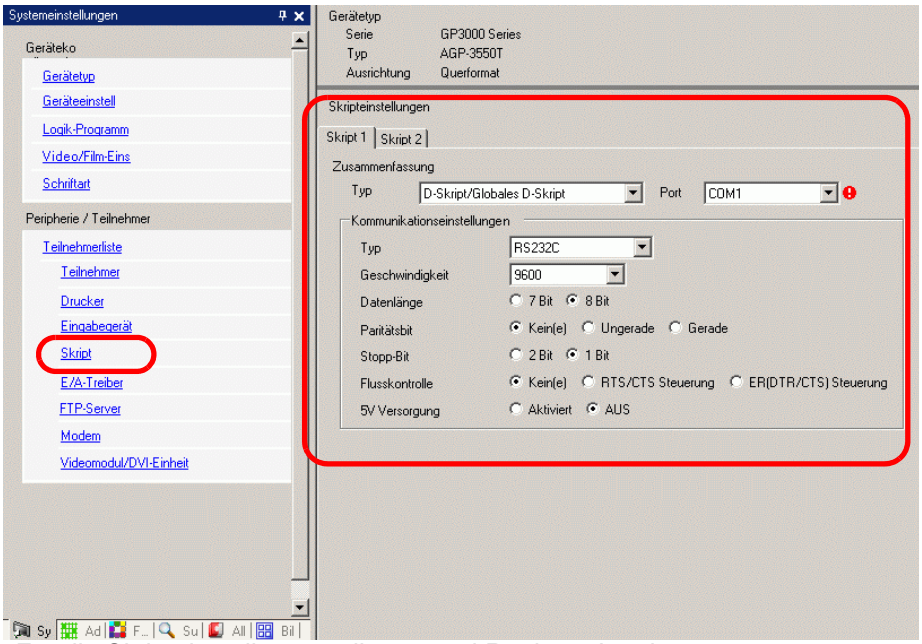
21.11.4 SIO-Port-Operation

SIO-Port-Operation	Funktionszusammenfassung
	Beschriftungsvariablen ☞" ■ Beschriftungsvariablen" (seite 21-106) Von dem Steuerelement, dem Status, der Summe der empfangenen Daten und der Empfangs- und Sendefunktion festgelegt.
	Daten empfangen ☞" ■ Empfangen" (seite 21-108) Liest von dem festgesetzten seriellen Anschluss (COM1 oder COM2) empfangene Daten.
	Senden ☞" ■ Senden" (seite 21-110) Schreibt in den angegebenen seriellen Anschluss (COM1 oder COM2).
	Erweiterter Empfang ☞" ■ Erweiterter Empfang" (seite 21-111) Liest von dem festgesetzten seriellen Anschluss (COM1 oder COM2) empfangene Daten. Kann nur in einem erweiterten Skript verwendet werden.
	Erweiterte Sendung ☞" ■ Erweiterte Sendung" (seite 21-112) Schreibt in den angegebenen seriellen Anschluss (COM1 oder COM2). Kann nur in einem erweiterten Skript verwendet werden.
	Standby-Empfangsfunktion ☞" ■ Standby-Empfangsfunktion" (seite 21-113) Bleibt bis zum Empfang des spezifizierten Texts im Standby-Empfangsmodus. Kann nur in einem erweiterten Skript verwendet werden.
	Standby-Funktion ☞" ■ Standby-Funktion" (seite 21-114) Das System wartet für die angegebene Zeitdauer. Kann nur in einem erweiterten Skript verwendet werden.

WICHTIG

- Steuerungsvariablen, Senden, Empfangen lassen sich leicht in ein D-Skript/ Globales D-Skript mit einbinden.
- Folgende Skripteinstellungen sind für die Kommunikation mit D-Skripten/ Globalen D-Skripten erforderlich. Wenn keine Skripteinstellungen angegeben sind, können sie nicht ausgeführt werden.

  [D-Skript/Globales D-Skript E/A-Verfahren]
  Klicken Sie im Systemeinstellungesfenster auf [Skript].
  Legen Sie den [Typ] auf [D-Script/Globales D-Script] fest.



  Für die Skripteinstellungen gibt es zwei Register. Im oben genannten Beispiel wird [Skript 1] verwendet.
  Stellen Sie den [Port] auf COM1 oder COM2 und die [Kommunikationseinstellungen] übereinstimmend mit dem erweiterten SIO ein.
- Es wird beim Erstellen eines Kommunikationsprogramms mit erweiterter Funktionalität als der SIO-Port-Operation empfohlen, das [Erweiterte Skript] zu verwenden. Beispiele zur Verwendung erweiterter Skripts erhalten Sie unter
  ☞"21.5 Kommunizieren mit nicht unterstützten peripheren Teilnehmern" (seite 21-22)

■ Beschriftungsvariablen

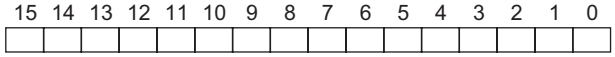
Kontrolle

Element	Beschreibung
Zusammen-fassung	Diese Steuerelementvariable wird zum Löschen des Sendepuffers, des Empfangspuffers und des Fehlerstatuses verwendet. Diese Steuerelementvariable ist nur schreibbar.
Format	Bei Designieren des Bits: [c:EXT_SIO_CTRL**]**: 00 bis 15) Bei Designieren des Wortes: [c:EXT_SIO_CTRL]

◆ Beispielausdruck

Bei Designieren des Bits: [c:EXT_SIO_CTRL00] = 1
Bei Designieren des Wortes: [c:EXT_SIO_CTRL] = 0x0007

◆ EXT_SIO_CTRL



Bit	Inhalt
15	Reserviert
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	1: Empfangs-Timeout löschen
2	1: Fehler löschen
1	1: Empfangspuffer löschen
0	1: Sendepuffer löschen

ANMERKUNG

- Bei Angabe eines Wortes (bei gleichzeitigem gesetzten zwei oder mehreren Bits) wird in folgender Reihenfolge verarbeitet: Fehler bereinigen –> Empfangspuffer bereinigen –> Sendepuffer bereinigen

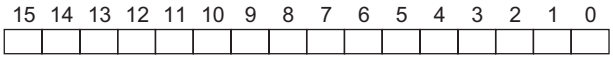
Status

Element	Beschreibung
Zusammen-fassung	Status schließt folgende Informationen mit ein. Diese Statusvariable ist nur schreibbar.
Format	Bei Designieren des Bits: [s:EXT_SIO_STAT**] (** : 00 bis 15) Bei Designieren des Wortes: [s:EXT_SIO_STAT]

◆ Beispielausdruck

Bei Designieren des Bits: wenn ([s:EXT_SIO_STAT 00] == 1)
Bei Designieren des Wortes: wenn (([s:EXT_SIO_STAT] & 0x0001) <> 0)

◆ Inhalt von EXT_SIO_STAT



Bit	Inhalt
15	0: Kein D-Skript/Globales D-Skript 1: D-Skript/Globales D-Skript exisitiert
14	0: Kein erweitertes Skript 1: Erweitertes Skript existiert
13	Reserviert
12	
11	
10	
9	
8	
7	
6	
5	
4	0: Normal 1: Timeout empfangen
3	0: Normal 1: Fehler empfangen
2	0: Kann keine Daten empfangen 1: Daten empfangen existiert
1	0: Normal 1: Sende-Fehler
0	0: Daten bestehen im Sendepuffer 1: Sendepuffer ist leer.

ANMERKUNG

- Die reservierten Bits könnten künftig zugewiesen werden. Stellen Sie deshalb sicher, dass nur die erforderlichen Bits überprüft werden.
- Es gibt zwei Sorten Übertragungsfehler: den Übertragungs-Timeout-Fehler und den Fehler, voller Übertragungspuffer. Bei Auftreten eines dieser beiden Fehler geht das Übertragungsfehler-Bit AN. Die Übertragungs-Timeoutphase beträgt fünf Sekunden.
- Es gibt vier Empfangsfehlertypen: Paritätsfehler, Überlauffehler, Rahmenfehler und Überlauf. Bei Auftreten eines dieser vier Fehler geht das Übertragungsfehler-Bit AN.
- Bei Erkennung eines Übertragungsfehlers bleiben die Sendedaten im Übertragungspuffer. Wenn ein Übertragungsfehler nicht erkannt werden kann, werden die Sendedaten vom Übertragungspuffer gesendet.
- Bei Verwendung einer serieller Schnittstelle COM2, die RS-422 ist, kann das CS- (CTS) Signal nicht erkannt werden. Deshalb kann die Trennung eines Kabels nicht erkannt werden.

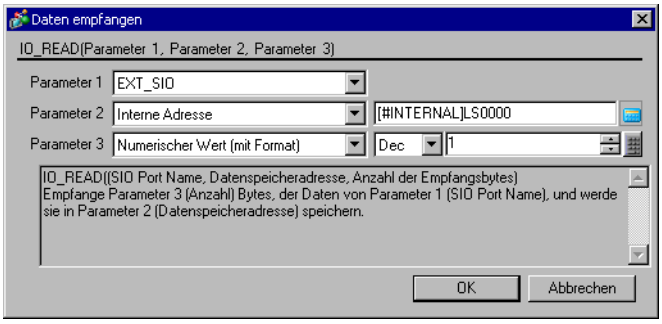
Empfangene Datengröße

Element	Beschreibung
Zusammen-fassung	Zeigt die zu diesem Zeitpunkt empfangene Datenmenge (Anzahl der Bytes) an. Die empfangene Datengröße ist schreibgeschützt.
Format	[r:EXT_SIO_RECV]

WICHTIG

- Beschriftungsname der Anzahl der empfangenen Daten (Anzahl der Bytes)
  In GP-PRO/PB III V. 6.0 und bei früheren Versionen ist der Beschriftungsname für die empfangene Datengröße [r: EXT_SIO_RCV]. Die Beschreibung muss jedoch nicht geändert werden, denn die Funktion bleibt gleich, ungeachtet ob der Ausdruck [r: EXT_SIO_RCV] oder [r: EXT_SIO_RECV] gewählt wird.

■ Empfangen

Element	Beschreibung
Zusammen-fassung	Schreiben Sie die Anweisung beim Auslesen der vom erweiterten SIO empfangenen Daten folgendermaßen.
Format	IO_READ ([p:EXT_SIO], Daten-Speicheradresse, Anzahl der empfangenen Bytes)  Parameter 1: EXT_SIO Parameter 2: Interne Adresse Parameter 3: Numerischer Wert

Beispielausdruck:

IO_READ ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

Im o.g. Beispiel wird die Anzahl der empfangenen Bytes in LS0100 gespeichert. 10 Datenbytes beginnend mit LS0101 werden gespeichert. Folgendes Bild zeigt die gespeicherten Empfangsdaten.

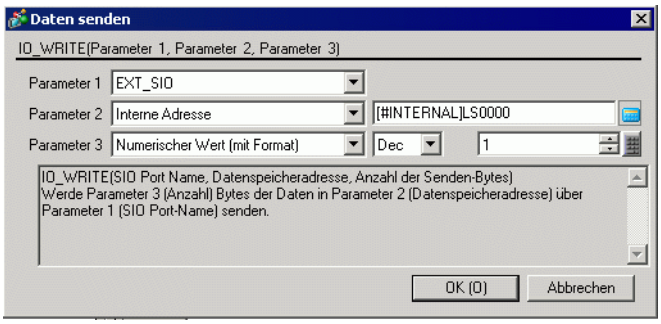
ANMERKUNG

- Die maximale Anzahl der Übertragungsbytes während des Datenempfangs ist 2011. Die Daten werden in Einheiten von 1 Byte in jede Wortadresse geschrieben.

	Empfangene Datengröße	... 10 bytes
LS0100		
LS0101	00	Byte 1
LS0102	00	Byte 2
LS0103	00	Byte 3
LS0104	00	Byte 4
LS0105	00	Byte 5
LS0106	00	Byte 6
LS0107	00	Byte 7
LS0108	00	Byte 8
LS0109	00	Byte 9
LS0110	00	Byte 10

Empfangene Datenspeichermethode

■ Senden

Element	Beschreibung
Zusammen-fassung	Schreiben Sie die Anweisung beim Schreiben von Daten in das erweiterte SIO folgendermaßen.
Format	IO_WRITE ([p:EXT_SIO], Daten-Speicheradresse, Anzahl der empfangenen Bytes)  Parameter 1: EXT_SIO Parameter 2: Interne Adresse Parameter 3: Numerischer Wert

Beispielausdruck:

IO_WRITE ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

Im o.g. Beispiel werden 10 Datenbytes beginnend mit LS0100 gesendet. Das folgende Bild zeigt die gespeicherten gesendeten Daten.

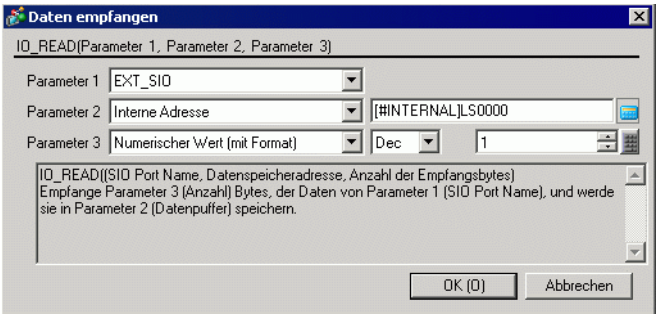
ANMERKUNG

- Die maximale Anzahl der Übertragungsbytes während des Datenempfangs ist 2012.
- Als LS-Teilnehmer für den Sendepuffer werden die Daten in einzelnen Bytes in jede Wortadresse geschrieben.

LS0100	00	Byte 1
LS0101	00	Byte 2
LS0102	00	Byte 3
LS0103	00	Byte 4
LS0104	00	Byte 5
LS0105	00	Byte 6
LS0106	00	Byte 7
LS0107	00	Byte 8
LS0108	00	Byte 9
LS0109	00	Byte 10

Gesendete Datenspeichermethode

■ Erweiterter Empfang

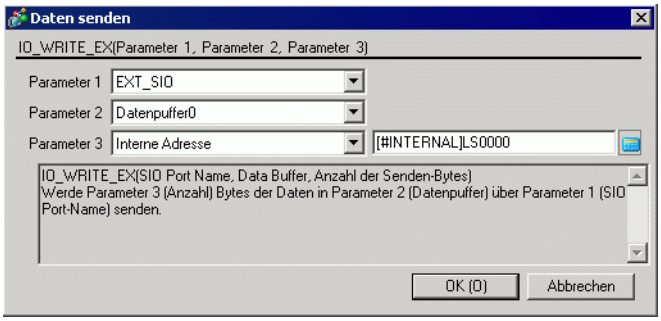
Element	Beschreibung
Zusammen-fassung	Empfängt Daten der in der Empfangsdatengröße festgesetzten Größe (Bytes) vom erweiterten SIO und speichert sie im Datenpuffer. Die mit Parameter 3 spezifizierte Anzahl von Bytes wird von dem erweiterten SIO empfangen und im mit Parameter 2 spezifizierten Datenpuffer gespeichert. Kann nur in einem erweiterten Skript verwendet werden.
Format	IO_READ_EX ([p:EXT_SIO], Datenpuffer, Anzahl der empfangenen Bytes)  Parameter 1: [p:EXT_SIO] Parameter 2: Datenpuffer Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 1024.)

Beispielausdruck:

IO_READ_EX ([p:EXT_SIO], databuf1, 10)

Im o.g. Beispiel werden 10 Datenbytes, die von dem Erweiterten SIO empfangen werden, in "databuf1" empfangen und gespeichert.

■ Erweiterte Sendung

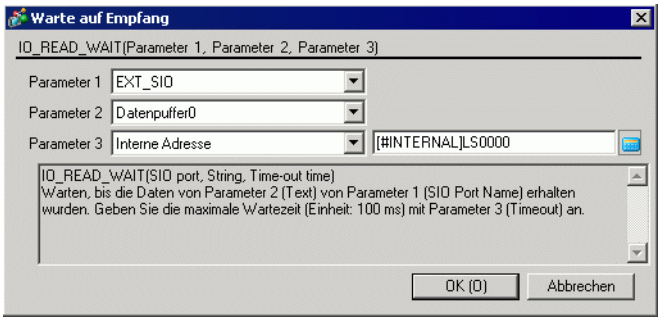
Element	Beschreibung
Zusammen-fassung	Sendet die Daten im Datenpuffer unter Anwendung des erweiterten SIO und gemäß der Anzahl der Sendebytes. Die mit Parameter 3 spezifizierte Anzahl von Bytes wird von dem erweiterten SIO empfangen und im mit Parameter 2 spezifizierten Datenpuffer gespeichert. Kann nur in einem erweiterten Skript verwendet werden.
Format	IO_WRITE_EX ([p:EXT_SIO], Datenpuffer, Anzahl der empfangenen Bytes)  Parameter 1: [p:EXT_SIO] Parameter 2: Datenpuffer Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 1024.)

Beispielausdruck:

IO_WRITE_EX ([p:EXT_SIO], databuf0, 10)

Im o.g. Beispiel werden 10 Datenbytes in "databuf0" vom erweiterten SIO gesendet.

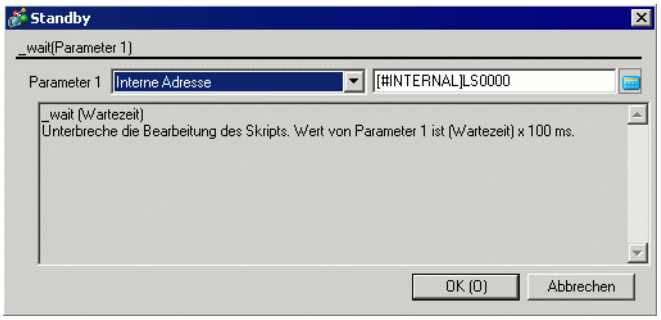
■ Standby-Empfangsfunktion

Element	Beschreibung
Zusammen-fassung	Bleibt bis zum Empfang des spezifizierten Texts im Standby-Empfangsmodus. Nach Ablauf der Timeout-Phase, wird Bit-4 (Timeout-Fehler-Empfang) des Status [s: EXT_SIO_STAT] festgelegt. Die Zeitdauer des Timeout kann in 100 Ms-Inkrementen festgelegt werden. Das System befindet sich bis zum Empfang des mit Parameter 2 spezifizierten Zeichenstrings oder Zeichencodes im Standby-Empfangsmodus. Die Timeout-Operation wird mit Parameter 3 konfiguriert. Kann nur in einem erweiterten Skript verwendet werden.
Format	IO_READ_WAIT([p:EXT_SIO], Text, Timeout)  Parameter 1: [p:EXT_IO] Parameter 2: Numerischer Wert, Text, Datenpuffer Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 600.)

WICHTIG

- Die Empfangsdaten können erst nach Empfang des spezifizierten Texts verwendet werden. (Andernfalls werden die Daten aufgegeben.)
- Bis zu 128 Zeichen (Bytes) können spezifiziert werden. Beachten Sie bitte, dass die Standby-Empfangsoperation bei der Spezifizierung von den Grenzwert übersteigen Strings nicht erfolgreich ausgeführt werden kann.

■ Standby-Funktion

Element	Beschreibung
Zusammen-fassung	Das System wartet für die angegebene Zeitdauer. Die Zeit kann in 100 Ms-Inkrementen konfiguriert werden. Kann nur in einem erweiterten Skript verwendet werden.
Format	_wait(Wartezeit)  Parameter 1: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 1 reicht von 1 bis 600.)

Beispielausdruck:

_wait (10)

Im o.g. Beispiel wartet das System eine Sekunde ab.

21.11.5 CF-Dateiverfahren/USB-Dateiverfahren

CF-Datei-Verfahren	Funktionszusammenfassung
	Beschriftungsvariablen ☞" ■ Beschriftungsvariablen" (seite 21-116) Von der Anzahl der angeführten Dateien, der Anzahl der gelesenen Bytes sowie dem CF-Kartenfehlerzustand/USB-Speicherfehlerzustand festgelegt.
	Datei schreiben ☞" ■ Datei schreiben" (seite 21-127) Es stehen eine der drei nachstehenden Modi zur Verfügung:
	Dateiname ändern ☞" ■ Dateiname ändern" (seite 21-133) Modifiziert den Dateinamen.
	CSV-Datei lesen ☞" ■ CSV-Datei lesen" (seite 21-135) Liest Daten in Zelleneinheiten von einer CSV Datei und schreibt sie in eine Wortadresse.
	Datei lesen ☞" ■ Datei lesen" (seite 21-138) Liest die spezifizierten Daten-Bytes nach dem spezifizierten Offset in die Datei und schreibt sie in die Zieladresse.
	Ausgabe-Dateiliste ☞" ■ Ausgabe-Dateiliste" (seite 21-141) Die in dem festgelegten Ordner existierende Dateiliste wird in die interne Adresse geschrieben.
	Datei löschen ☞" ■ Datei löschen" (seite 21-143) Löscht die Datei.

■ Beschriftungsvariablen

Nachstehend finden Sie die möglichen Statuswerte für den Status der CF-Karte/USB-Speicher:

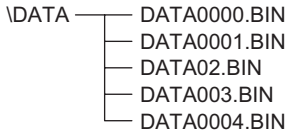
Status-Name	Beschriftungsname	Beschreibung
Aufgelistete Dateien	[s:CF_FILELIST_NUM] [s:USB_FILELIST_NUM]	Speichert die Anzahl der Dateien, die tatsächlich bei Ausführung der Dateilisten-Ausgabefunktion "_CF_dir ( )" oder "_USB_dir ( )" aufgeführt werden.
Die Anzahl der zu lesenden Bytes	[s:CF_READ_NUM] [s:USB_READ_NUM]	Speichert die Anzahl der Bytes, die bei Ausführung der Datei-Lesefunktion "_CF_read ( )" oder "_USB_read ( )" ausgelesen werden können.
CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus	[s:CF_ERR_STAT] [s:USB_ERR_STAT]	Speichert den bei dem Zugriff auf die CF-Karte oder USB-Speicher generierten Fehlerstatus.

Aufgelistete Dateien

Bei Ausführen der Dateilisten-Ausgabefunktion "_CF_dir ( )" oder "_USB_dir" ( ) wird die Anzahl tatsächlich in den LS-Bereich geschriebener Dateilisten in "Gelistete Dateien [s:CF_FILELIST_NUM]/[s:USB_FILELIST_NUM]" gespeichert.

◆ Anwendungsbeispiel

_CF_dir ("\DATA*.*", [w:[#INTERNAL]LS0100], 10, 0)
[w:LS0200] = [s:CF_FILELIST_NUM]



Wenn der Versuch unternommen wird, eine Dateiliste mit 10 Dateien zu erhalten, der angegebene Ordner jedoch nur fünf Dateien enthält, wird "5" in [s:CF_FILELIST_NUM] gespeichert.

WICHTIG
- Wenn keine Dateien geschrieben werden, wird die Gesamtzahl der Dateien im angegebenen Ordner in [s:CF_FILELIST_NUM] geschrieben.

Die Anzahl der zu lesenden Bytes

Bei Ausführen der Dateilesefunktion "_CF_read ( )" oder "_USB_read" ( ) wird die Anzahl tatsächlich geschriebener Bytes in "Ausgelesene Bytes [s:CF_READ_NUM] / [s:USB_READ_NUM] gespeichert.

◆ Anwendungsbeispiel

_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 16, 16)
[w:[#INTERNAL]LS0200] = [s:CF_READ_NUM]

Wenn der Versuch unternommen wird, 16 Bytes zu lesen, aber nur 12 Bytes erfolgreich gelesen werden,
wird "12" in [s:CF_READ_NUM] gespeichert.

CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus

Speichert den bei dem Zugriff auf die CF-Karte oder USB-Speicher generierten Fehlerstatus.

Bit-Position	Fehlername	Beschreibung
15	Reserviert	Reserviert
14		
13		
12		
11		
10		
9		
8		
7		
6	Fehler beim Umbenennen der Datei	• Die CF-Karte/USB-Speicher wird während der Ausführung entfernt. • Angegebene Datei existiert nicht.
5	Fehler beim Löschen der Datei	• Die CF-Karte/USB-Speicher wird während der Ausführung entfernt. • Angegebene Datei existiert nicht. • Löschversuch einer Datei mit einem Nur-Lesen-Attribut.
4	Fehler beim Schreiben der Datei	• Die CF-Karte/USB-Speicher wird während der Ausführung entfernt. • Der freie Speicherplatz der CF-Kartenkapazität/ USB-Speicherkapazität wurde überschritten. • Schreibversuch in eine Datei mit einem Nur-Lesen-Attribut. • Es wurde versucht, eine Datei zu "überschreiben", die nicht existiert.
3	Fehler beim Lesen der Datei	• Die CF-Karte/USB-Speicher wird während der Ausführung entfernt. • Angegebene Datei existiert nicht.
2	Dateilistenfehler	• Die CF-Karte/USB-Speicher wird während der Ausführung entfernt. • Angegebener Ordner existiert nicht.
1	CF-/USB-Speicherkartenfehler	• CF-Karte/USB-Speicher ist ungültig. • Das eingefügte Medium ist keine CF-Karte.
0	CF/USB-Speicher Karte fehlt	• CF-Karte/USB-Speicher ist nicht eingefügt. • Abdeckung ist offen.

- Selbst wenn ein CF-Kartenfehler/USB-Speicherfehler auftritt, wird die Verarbeitung fortgesetzt. Vergewissern Sie sich beim Schreiben des Skripts, dass es auf Fehler hin überprüft wird, wenn die Funktionen der Dateiverfahren einer CF-Karte/eines USB-Speichers verwendet werden.
  Beispiel:

```
_CF_dir ("\DATA\*.*", [w:[#INTERNAL]LS0100], 2, 1) // Gibt eine Dateiliste aus.
if([s:CF_ERR_STAT02] <> 0)          // Überprüft den Fehlerstatus.
{
    set ([b:[#INTERNAL]LS005000]) // Legt die Bitadresse für die Fehleranzeige fest
}
endif
```

◆ CF-Karte/USB-Speicher Speicherbereich des Fehlerdetailstatus

Jedes Bit wird bei Auftreten eines Fehlers gesetzt. Sie können prüfen, welche Faktoren zu einem Fehler führen, indem Sie Detailstatus einstellen. In jeder Funktion wird der Detailstatus in LS9132 bis LS9137 für den erweiterten Systembereich (LS9138 bis LS9143 für den USB-Speicher) gespeichert. Diese Bereiche sind nur zum Einlesen.

LS-Bereich

LS0000	
:	
LS9132	Status der CF-Karten Lis
LS9133	Status der CF-Karten Le
LS9134	Status der CF-Karten Sc
LS9135	Status der CF-Karten Lö
LS9136	Status der CF-Karten Un
LS9137	Status von CSV-Lesen
:	
LS9999	

LS Area

LS0000	
:	
LS9138	USB List Status
LS9139	USB Read Status
LS9140	USB Write Status
LS9141	USB Delete Status
LS9142	USB Rename Status
LS9143	USB CSV Read Status
:	
LS9999	

◆ Fehlerliste für jede Funktion

Editor-Funktionsname		Fehlerstatus	Ursache
_CF_dir ( )	LS9132	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens)
		0012h	Dateinamen- (Pfadname-) fehler
		0018h	Schreibebereichsfehler im LS-Bereich
		0020h	Keine CF-Karte
		0021h	Ungültige CF-Karte
		0100h	Fehler beim Öffnen des Verzeichnisses

Fortsetzung

Editor-Funktionsname		Fehlerstatus	Ursache
_CF_read ( )	LS9133	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0018h	Schreibebereichsfehler im LS-Bereich
		0020h	Keine CF-Karte
		0021h	Ungültige CF-Karte
		0101h	Datei-Positionierungsfehler (Offset-Fehler)
		0102h	Fehler bei der Anzahl der ausgelesenen Bytes
		0110h	Fehler beim Erstellen (Öffnen) der Datei
_CF_write ( )	LS9134	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Keine CF-Karte
		0021h	Ungültige CF-Karte
		0101h	Datei-Positionierungsfehler (Offset-Fehler)
		0104h	Fehler beim Erstellen des Ordners
		0108h	Schreibmodusfehler
		0110h	Fehler beim Erstellen (Öffnen) der Datei
		0111h	Fehler beim Schreiben der Datei (z.B. ungenügend Platz auf CF-Karte)

Fortsetzung

Editor-Funktionsname		Fehlerstatus	Ursache
_CF_delete ( )	LS9135	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Keine CF-Karte
		0021h	Ungültige CF-Karte
		0112h	Fehler beim Löschen der Datei (z.B. angegebene Datei existiert nicht. Angegebene Datei ist nur lesbar.)
_CF_rename ( )	LS9136	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Keine CF-Karte
		0021h	Ungültige CF-Karte
		0114h	Fehler beim Umbenennen der Datei (z.B. angegebene Datei existiert nicht. Dateiname existiert bereits.)
_CF_read_csv ( )	LS9137	0001h	Parameter-Fehler
		0002h	CF-Kartenfehler (Keine CF-Karte/Fehler beim Öffnen von Datei/Fehler beim Lesen von Datei)
		0003h	Schreibfehler

Fortsetzung

Editor-Funktionsname		Fehlerstatus	Ursache
__USB_dir ( )	LS9138	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens)
		0012h	Dateinamen- (Pfadname-) fehler
		0018h	Schreibebereichsfehler im LS-Bereich
		0020h	Kein USB-Speicher
		0021h	Ungültiger USB-Speicher
		0100h	Fehler beim Öffnen des Verzeichnisses
__USB_read ( )	LS9139	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0018h	Schreibebereichsfehler im LS-Bereich
		0020h	Kein USB-Speicher
		0021h	Ungültiger USB-Speicher
		0101h	Datei-Positionierungsfehler (Offset-Fehler)
		0102h	Fehler bei der Anzahl der ausgelesenen Bytes
		0110h	Fehler beim Erstellen (Öffnen) der Datei

Fortsetzung

Editor-Funktionsname		Fehlerstatus	Ursache
__USB_write ( )	LS9140	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Kein USB-Speicher
		0021h	Ungültiger USB-Speicher
		0101h	Datei-Positionierungsfehler (Offset-Fehler)
		0104h	Fehler beim Erstellen des Ordners
		0108h	Schreibmodusfehler
		0110h	Fehler beim Erstellen (Öffnen) der Datei
		0111h	Fehler beim Schreiben der Datei (Beispiel: Ungenügender Speicherplatz auf dem USB-Speicher)

Fortsetzung

Editor-Funktionsname		Fehlerstatus	Ursache
__USB_delete ( )	LS9141	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Kein USB-Speicher
		0021h	Ungültiger USB-Speicher
		0112h	Fehler beim Löschen der Datei (z.B. angegebene Datei existiert nicht. Angegebene Datei ist nur lesbar.)
__USB_rename ( )	LS9142	0010h	Die D-Skript-Daten sind ungültig (Fehler bei Abfrage des mit einem festen String gekennzeichneten Ordnernamens/ Dateinamens)
		0011h	Lesebereichsfehler im LS-Bereich
		0012h	Dateinamen- (Pfadname-) fehler
		0020h	Kein USB-Speicher
		0021h	Ungültiger USB-Speicher
		0114h	Fehler beim Umbenennen der Datei (z.B. angegebene Datei existiert nicht. Dateiname existiert bereits.)
__USB_read_csv ( )	LS9143	0001h	Parameter-Fehler
		0002h	USB-Speicherfehler (Kein USB-Speicher, Fehler beim Öffnen einer Datei, Fehler beim Lesen einer Datei)
		0003h	Schreibfehler

◆ Datenspeicher-Modus

Bei Lesen/Schreiben von/in Teilnehmeradressen während des Ausführens der Funktion Dateilesen/Dateischreiben kann die Speicherreihenfolge der geschriebenen (ausgelesenen) Daten festgelegt werden.
Bei Einstellen des Datenspeichermodus in LS9130 könnte sich die Speicherreihenfolge ändern. Der Modus kann aus vier Optionen gewählt werden: 0, 1, 2 oder 3.

ANMERKUNG

- Verwenden Sie folgende Befehle zum Referenzieren von LS9130.
  _CF_write() CF-Datei-Operation: In die Datei schreiben
  _CF_read() CF-Datei-Operation: Datei lesen
  _CF_read_csv() CF-Dateioperation: CSV-Datei lesen
  _USB_write() USB-Dateioperation: In die Datei schreiben
  _USB_read() USB-Dateioperation: Datei lesen
  _USB_read_csv() USB-Dateioperation: CSV-Datei lesen
  IO_WRITE([p:PRN],...)Drucker-Operation: Senden
- Wenn in Teilnehmeradressen geschrieben oder von Teilnehmeradressen gelesen wird, können die folgenden Funktionen zum Kommunizieren mit der Eigenschaft [Textdatenmodus] im Fenster [Systemeinstellungen] der Seite [Teilnehmer/SPS] verwendet werden, anstelle des Speichermodus LS9130.
  _CF_dir() CF-Dateioperation: Ausgabe Dateiliste
  _USB_dir() USB-Dateioperation: Ausgabe-Dateiliste

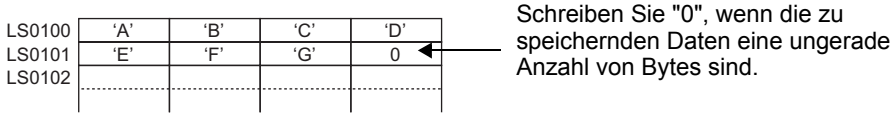
- Modus 0

Bei Verwendung der Datei-Lesefunktion, um die Zeichenfolge "ABCDEFG" in eine interne Adresse zu schreiben

[w:[#INTERNAL]LS9130] = 0
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)

- Wenn die Länge der internen Adresse 16-Bit ist

- Wenn die Länge der internen Adresse 32-Bit ist



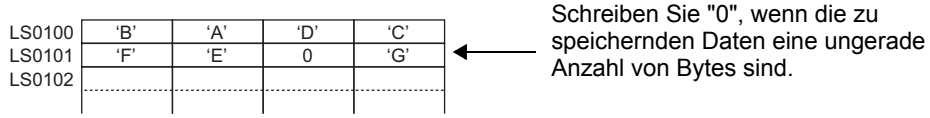
- Modus 1

Beispiel: Bei Verwendung der Datei-Lesefunktion, um die Zeichenfolge "ABCDEFG" in eine interne Adresse zu schreiben

[w:[#INTERNAL]LS9130] = 1
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)

- Wenn die Länge der internen Adresse 16-Bit ist

- Wenn die Länge der internen Adresse 32-Bit ist



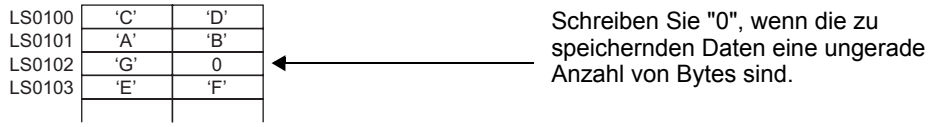
- Modus 2

Beispiel: Bei Verwendung der Datei-Lesefunktion, um die Zeichenfolge "ABCDEFG" in eine interne Adresse zu schreiben

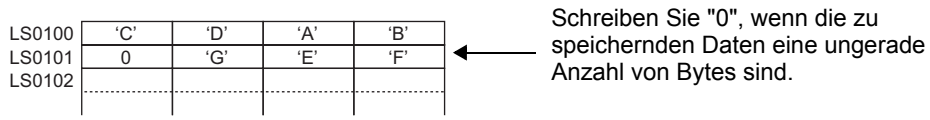
[w:[#INTERNAL]LS9130] = 2

_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)

- Wenn die Länge der internen Adresse 16-Bit ist



- Wenn die Länge der internen Adresse 32-Bit ist



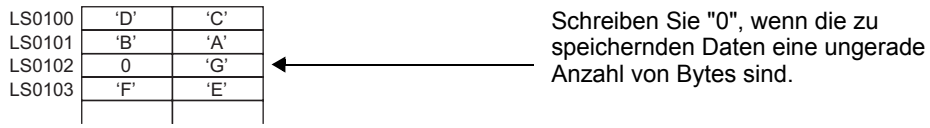
- Modus 3

Beispiel: Bei Verwendung der Datei-Lesefunktion, um die Zeichenfolge "ABCDEFG" in eine interne Adresse zu schreiben

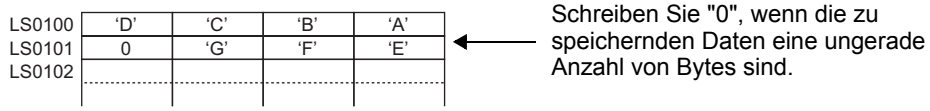
[w:[#INTERNAL]LS9130] = 3

_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)

- Wenn die Länge der internen Adresse 16-Bit ist



- Wenn die Länge der internen Adresse 32-Bit ist



WICHTIG

- In der Systemeinstellung sind Datenspeichermodus und String-Datenmodus nicht gleichzusetzen. Die Beziehung zum String-Datenmodus wird in der folgenden Tabelle veranschaulicht.

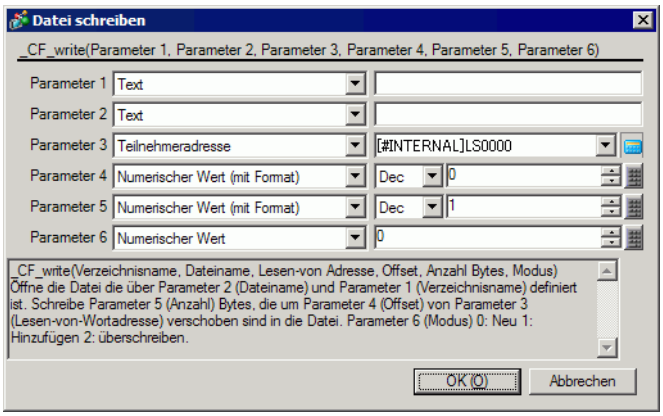
Daten Speicher-Reihenfolge	Byte in Wort LH/HL Speicher-reihenfolge	In Doppelwort LH/HL Speicher-reihenfolge	D-Skript-Daten-speicher-Modus	Text-Doppel-wortstruktur
Speichern ab Start-Daten	HL-Reihenfolge	HL-Reihenfolge	0	1
	LH-Reihenfolge		1	2
	HL-Reihenfolge	LH-Reihenfolge	2	5
	LH-Reihenfolge		3	4
Speichern ab End-Daten	HL-Reihenfolge	HL-Reihenfolge	–	3
	LH-Reihenfolge		–	7
	HL-Reihenfolge	LH-Reihenfolge	–	8
	LH-Reihenfolge		–	6

- Die Frequenz, mit der Daten auf die CF-Karte geschrieben werden können, ist begrenzt. Deshalb sollte dafür gesorgt werden, dass regelmäßig ein Backup sämtlicher CF-Kartendaten auf ein anderes Speichermedium durchgeführt wird. In der Annahme, dass 500KB an DOS Format-Daten überschrieben sind, beträgt der Grenzwert 100.000 Mal so viel.
- Wenn ein Fehler während der CF-Kartenverarbeitung/USB-Speicherverarbeitung auftritt, wird der Fehler in den Fehlerstatus der CF-Karte/des USB-Speichers geschrieben [s:CF_ERR_STAT]/ [s:USB_ERR_STAT]. Weitere Einzelheiten entnehmen Sie bitte " CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus" (seite 21-117) .
- Die folgenden Symbole und Zeichen dürfen weder in Ordner- noch in Dateinamen verwendet werden. Bei Verwendung dieser Symbole und Zeichen in einem Ordner- und Dateinamen tritt ein Fehler auf.

:	,	=	+	/	"	[
]	|	<	>	(Platz)	?	

- Zur Angabe eines Ausgangsordners (Verzeichnis) wird " " (leere Zeichenfolge) als Ordnername festgelegt.

■ Datei schreiben

Element	Beschreibung
Zusammenfassung	Es stehen eine der drei nachstehenden Modi zur Verfügung: "Neu", "Hinzufügen" oder "Überschreiben". Weitere Einzelheiten über die Datenspeicher-Reihenfolge finden Sie unter "Datenspeicher-Modus".
Format	_CF_write/_USB_write (Ordnernamen, Dateinamen, von Adressen lesen, Offset, Anzahl der Bytes, Modus)  Parameter 1 Ordnername: Fester String (maximale Länge: 32 einzelne Bytezeichen) Parameter 2 Dateiname:Fester String, interner Adresse (maximale Länge: 32 einzelne Bytezeichen), interne Adresse + temporäre Adresse Parameter 3 Lesequelladresse: Interne Adresse, interne Adresse + temporäre Adresse Parameter 4 Offset:Numerischer Wert, interne Adresse, temporäre Adresse (maximale Anzahl, die festgelegt werden kann: 65535 für 16-Bit-Länge, 4294967295 für 32-Bit-Länge) Parameter 5 Anzahl der Bytes: Numerischer Wert, interne Adresse, temporäre Adresse (verfügbare Länge: 1280) Parameter 6 Modus: Numerischer Wert, interne Adresse, temporäre Adresse (verfügbare Werte: 0, 1, 2)

Übersicht über das Speicherformat

Mode	Name	Beschreibung
0	Neu	Neue Datei erstellen. Falls schon eine Datei mit demselben Namen existiert, wird sie gelöscht.
1	Hinzufügen	Die Daten zur spezifizierten Datei hinzufügen. Falls die spezifizierte Datei nicht existiert, wird eine neue Datei erstellt.
2	Überschreiben	Teilweises Überschreiben der Datei. Falls das angegebene Offset die Größe der Datei überschreitet, wird der überschüssige Bereich mit den Nullen gefüllt und die Daten werden nach dem Bereich geschrieben. Bei Angabe des Offsets am Ende der Dateidaten ist der Vorgang gleichwertig mit dem Hinzufügen der Daten in die Datei. Falls die Datei nicht existiert, tritt ein Fehler auf. Weitere Informationen über diese Fehler erfahren Sie unter " CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus" (seite 21-117) .

Beispielausdruck:

```
[w:[#INTERNAL]LS0200] = 0//Offset ("0" wenn Modus "New" ist)
[w:[#INTERNAL]LS0202] = 100 // Anzahl der Bytes (100 Bytes)
[w:[#INTERNAL]LS0204] = 0// Modus (Neu)
_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100],
[w:[#INTERNAL]LS0200],
[w:[#INTERNAL]LS0202], [w:[#INTERNAL]:LS0204])
```

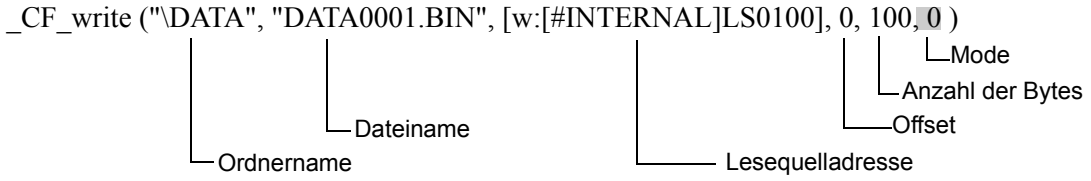
Im vorgehenden Beispiel werden 100 Bytes-Daten von LS0100 gelesen und im Ordner \DATA als DATA0001.BIN gespeichert. Sie können die Byteanzahl und den Bytemodus indirekt durch Definieren des Offset, der Byteanzahl und des Modus mit den internen Adressen bestimmen.

WICHTIG

- Die Offset-Einstellung funktioniert nur im Modus "Überschreiben". Im Modus "Neu" und "Hinzufügen" ist die Offset-Einstellung deaktiviert. Abgesehen vom Modus "Überschreiben" wird der Offset-Wert bei anderen Modi auf "0" festgelegt.
- Wenn der Modus "Neu" angegeben ist und schon eine Datei mit demselben Namen existiert, wird diese überschrieben.
- Wenn der LS-Bereich für "Dateiname" festgelegt wird, zählt "Lesequelladresse" nicht als D-Skript-Adresse.
- Bei Spezifizieren eines Teilnehmers für die "Lesequelladresse" werden bei Ausführung der Funktion die Daten nur einmal aus der SPS gelesen. Wenn während des Datenlesens ein Fehler auftritt, wird dies einen Lesefehler der CF-Karte oder des USB-Speichers zur Folge haben. [s:CF_ERR_STAT] oder [s:<USB_ERR_STAT]. Bei erfolgreichem Lesen der Daten wird der Fehler gelöscht.
- Die Daten werden gemäß der Anzahl der zu lesenden Bytes geteilt und aus der Quelle gelesen, abhängig von der Anzahl der zu lesenden Bytes. Deshalb ist es selbst bei Auftreten eines Kommunikationsfehlers während des Lesens der Daten möglich, dass die Daten teilweise in die angegebene Datei geschrieben wurden.
- Zur Angabe eines ganzen Pfads für einen Dateinamen wird ""∗" (Stern) als Ordnername festgelegt.
  Z.B. _CF_read ("∗", "\DATA\DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 10)

Speicherformat-Beispielausdruck

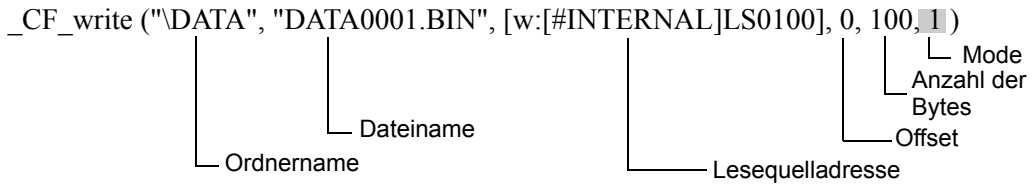
◆ **Bei Angabe des Moduses "Neu"**



Im obigen Beispiel werden 100 Bytedaten von LS0100 gelesen und die Datei DATA0001.BIN wird im Ordner \DATA neu erstellt.

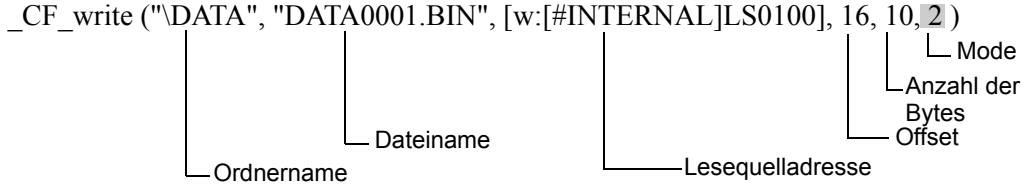
WICHTIG
- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Dateinamen, die länger als dieses Format sind, sind nicht zulässig.

◆ **Bei Angabe des Moduses "Hinzufügen"**



Falls die angegebene Datei (DATA0001.BIN im Beispiel) bei Ausführung der Anweisung schon existiert, werden 100 Datenbytes von LS0100 gelesen und die folgenden Bereiche zur Datei DATA0001.BIN im Ordner \DATA hinzugefügt.

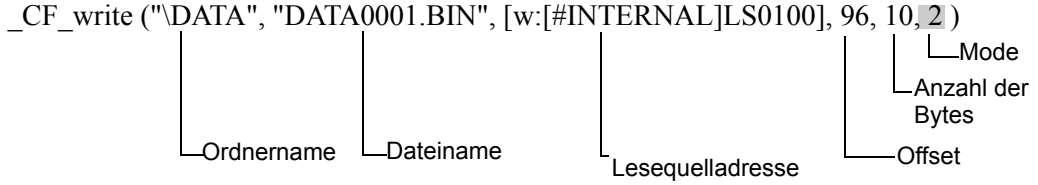
◆ **Bei Angabe des Moduses "Überschreiben" (1)**



Falls die angegebene Datei (DATA0001.BIN im Beispiel) bei Ausführung der o.g. Anweisung schon existiert, werden 10 Datenbytes in LS0100 gespeichert und die folgenden Bereiche gelesen und überschrieben; 10 Datenbytes werden im 17. und die darauffolgenden Bytes im Offset in der Datei DATA0001.BIN im Ordner \DATA gespeichert.

◆ Bei Angabe des Moduses "Überschreiben" (2)

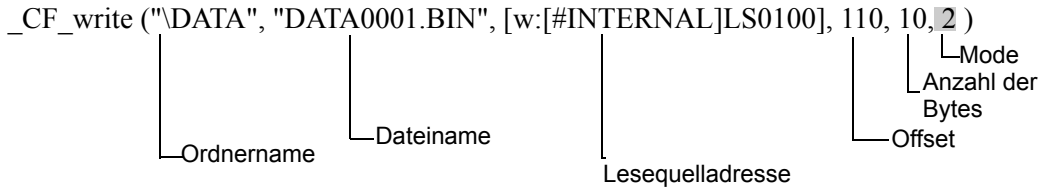
(Die zu überschreibende Datei ist kleiner als die Summe des Offset-Wertes und die Anzahl der hinzugefügten Bytes.)



Die angegebene Datei (DATA0001.BIN im Beispiel) existiert schon und die Dateigröße beträgt 100 Bytes. Wenn für die Überschreibe-Operation das Offset bei 96 Bytes und die Anzahl der Bytes bei 10 Bytes festgelegt wird, werden 10 in LS0100 gespeicherte Datenbytes sowie die folgenden Bereiche gelesen. Dann überschreiben die ersten 4 Bytes ausgelesene Daten die 4 in der 97. gespeicherten Daten sowie die folgenden Bytes in der Datei, und die restlichen 6 Datenbytes werden am Ende der Dateidaten hinzugefügt. Die resultierende Datei enthält 106 Datenbytes.

◆ Bei Angabe des Moduses "Überschreiben" (3)

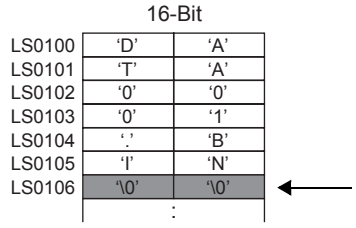
(Die zu überschreibende Datei ist kleiner als der Offset Wert.)



Die angegebene Datei (DATA0001.BIN im Beispiel) existiert schon und die Dateigröße beträgt 100 Bytes. Bei Setzen des Offset auf 110 Bytes und der Anzahl der Bytes auf 10 Bytes für die Überschreibe-Operation wird der Bereich zwischen dem 101sten Byte und dem 110ten Byte mit den gefüllt und die 10 Datenbytes, die aus LS0100 und folgenden Bereichen gelesen werden, in den 111sten und die folgenden Bytes geschrieben. Die resultierende Datei enthält 120 Datenbytes.

WICHTIG

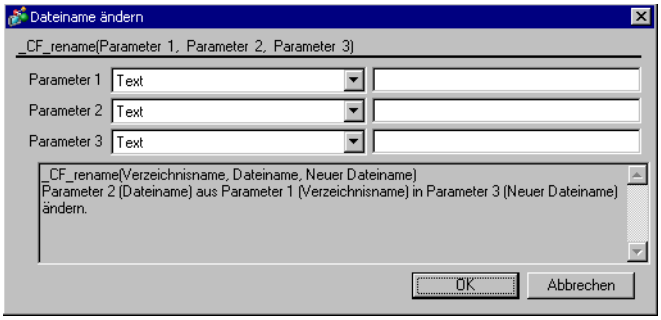
- Die maximal zulässige Anzahl von Zeichen für den ersten Parameter (Ordnername) und den zweiten Parameter (Dateiname) beträgt 32 Einzelbytezeichen.
- Die interne Adresse kann für den zweiten Parameter (Dateiname) bestimmt werden. Durch Bestimmen der internen Adresse kann ein Dateiname indirekt adressiert werden. Zudem können bis zu 32 Einzelbytezeichen zum Festlegen eines Dateinamens verwendet werden.
  Z.B. _CF_write ("\DATA", [w:[#INTERNAL]LS0100], [w:[#INTERNAL]LS0200], 0, 100, 0)
  Das Speichern eines Dateinamens in LS0100 lässt die indirekte Adressierung eines Dateinamens zu. In diesem Beispiel wird ein Dateiname in LS0100 bis LS0106 folgendermaßen gespeichert.



Der Dateinamen muss mit einem NULL Zeichen enden. Der Anzeige-Teilnehmer erkennt die Daten vor dem NULL Zeichen als den Dateinamen.

  Im o.g. Beispiel werden 100 Datenbytes von LS0200 gelesen und eine neue Datei "\DATA\DATA0001.BIN" für die Datenspeicherung erstellt.
- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Lange Dateinamen sind nicht zulässig.

■ Dateiname ändern

Element	Beschreibung
Zusammen-fassung	Modifiziert den Dateinamen. Parameter 1 bestimmt den CF-Kartenordner. Parameter 2 designiert den originalen Dateinamen. Parameter 3 designiert den neuen Namen.
Format	_CF_rename/_USB_rename (Ordnernamen, Dateinamen, geänderte Dateinamen) Der Dateiname kann auch indirekt mit der LS-Adresse festgelegt werden.  Parameter 1 Ordnername: Fester Text Parameter 2 Dateiname: Fester Text, interne Adresse, interne Adresse + temporäre Adresse Parameter 3 Dateiname: Fester Text, interne Adresse, interne Adresse + temporäre Adresse

Beispielausdruck:

_CF_rename ("\DATA","DATA0001.BIN","DATA1234.BIN")

Im o.g. Beispiel wird der Dateiname von "\DATA\DATA0001.BIN" zu "\DATA\DATA1234.BIN" geändert.

WICHTIG

- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Lange Dateinamen sind nicht zulässig.
- Die maximal zulässige Anzahl von Zeichen für den ersten Parameter (Ordnername) und den zweiten Parameter (Dateiname) beträgt 32 Einzelbytezeichen.
- Die interne Adresse kann für den zweiten und dritten Parameter (Dateinamen) bestimmt werden. Durch Bestimmen der internen Adresse kann ein Dateiname indirekt adressiert werden. Zudem können bis zu 32 Einzelbytezeichen zum Festlegen eines Dateinamens verwendet werden.
  Beispiel
  _CF_rename ("\DATA", [w:[#INTERNAL]LS0100], [w:[#INTERNAL]LS0200])
  Das Speichern des Dateinamens in LS0100 und LS0200 aktiviert die indirekte Adressierung des Dateinamens.

- Speichern Sie die Dateinamen folgendermaßen in LS0100 bis LS0106:

16-Bit

LS0100	'D'	'A'
LS0101	'T'	'A'
LS0102	'0'	'0'
LS0103	'0'	'1'
LS0104	'.'	'B'
LS0105	'I'	'N'
LS0106	'\0'	'\0'
	:	

Der Dateinamen muss mit einem NULL Zeichen enden. Das GP erkennt die Daten vor dem NULL-Zeichen als einen Dateinamen.

16-Bit

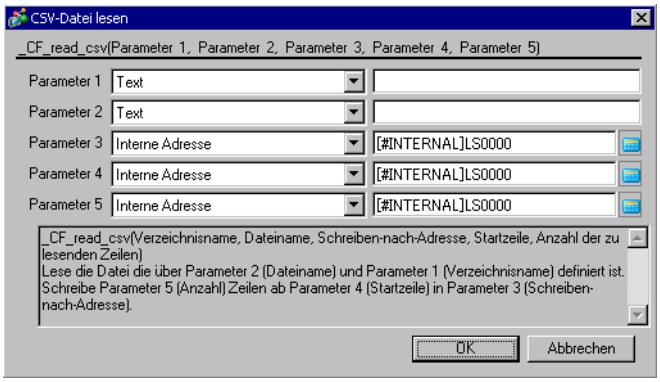
LS0200	'D'	'A'
LS0201	'T'	'A'
LS0202	'1'	'2'
LS0203	'3'	'4'
LS0204	'.'	'B'
LS0205	'I'	'N'
LS0206	'\0'	'\0'
	:	

In der obigen Anweisung wird die Datei "\DATA\DATA0001.BIN" in die Datei "\DATA\DATA1234.BIN" umbenannt.

- Wenn der LS-Bereich für "Dateiname" festgelegt wird, zählt er nicht als D-Skript-Adresse.
- Zur Angabe eines Ausgangsordners (Verzeichnis) wird " " (leere Zeichenfolge) als Ordnername festgelegt.
- Zur Angabe eines ganzen Pfads für einen Dateinamen wird "“∗" (Stern) als Ordnername festgelegt.

■ CSV-Datei lesen

Element	Beschreibung
Zusammen-fassung	Liest Daten in Zelleneinheiten von einer CSV Datei (aus einem durch "," begrenzten Zellenbild erstellt) und schreibt sie in eine Wortadresse.
Format	_CF_read_csv/_USB_read_csv (Ordnernamen, Dateinamen, in Adressen speichern, Startzeile, Anzahl der gelesenen Zeilen)



Parameter 1: Text (bis zu 32 Einzelbytezeichen)
Parameter 2: Text (bis zu 32 Einzelbytezeichen), interne Adresse, interne Adresse + temporäre Adresse
Parameter 3: Interne Adresse, interne mit Offset gekennzeichnete Adresse
Parameter 4: Numerischer Wert (1 bis 65.535), Interne Adresse, temporäre Variable
Parameter 5: Numerischer Wert (1 bis 65.535), Interne Adresse, temporäre Variable

Beispielausdruck:

_CF_read_csv ("\CSV", "SAMPLE.CSV", [w:[#INTERNAL]LS1000], 1, 2)
(Beim Lesen von zwei Datenzeilen, beginnend mit der ersten Dateizeile [\CSV\SAMPLE.CSV] in der CF-Speicherkarte unter Anwendung der Funktion "_CF_read_csv ( )".)

SAMPLE.CSV

001, "DAT01-01", "DAT01-2"
002, "DAT02-01", "DAT02-2"

16-Bit

LS1000	1	
+1	'D'	'A'
+2	'T'	'0'
+3	'1'	'-'
+4	'0'	'1'
+5	00h	00h
+6	'D'	'A'
+7	'T'	'0'
+8	'1'	'-'
+9	'2'	00h
LS1010	2	
+1	'D'	'A'
+2	'T'	'0'
+3	'2'	'-'
+4	'0'	'1'
+5	00h	00h
+6	'D'	'A'
+7	'T'	'0'
+8	'2'	'-'
+9	'2'	00h

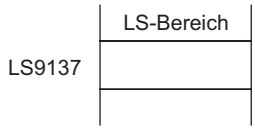
Liest zwei Datenzeilen, beginnend mit der ersten Zeile der CSV Datei. Wenn es sich bei dem ersten Zeichen um einen numerischen Wert handelt ("0" bis "9" oder "–"), werden die Daten als numerischer Wert gespeichert. Wenn es sich bei dem ersten Zeichen um ["] handelt, werden die Daten als Zeichen behandelt und "00h" wird am Ende der Text-Zeichenfolge gespeichert. Beim Speichern von "DAT01-01" beispielsweise beträgt die Datengröße 8 Zeichen, eine gerade Zahl, und es werden insgesamt fünf Worte verwendet: Vier Worte werden für das Speichern der Text-Zeichenfolge und eines für "00h" am Ende verwendet. Beim Speichern von "DAT01-2" beispielsweise beträgt die Datengröße 7 Zeichen, eine ungerade Zahl, und es werden insgesamt 4 Worte für das Speichern des Texts verwendet, wobei "00h" am Ende gespeichert wird.

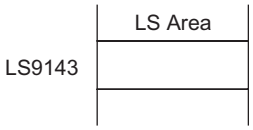
Wenn der Datenspeichermodus 0 beträgt

ANMERKUNG

- Wenn das erste Zeichen einer Zelle ein numerischer Wert ist ("0" to "9" or "–"), wird der Wert in numerische Daten konvertiert und diese Daten dann in den LS-Teilnehmer geschrieben. Der zulässige Bereich liegt zwischen –32.768 und 32.767.
- Wenn das erste Zeichen in der Zelle ["] ist, schreibt es den Bereich mit ["] in den LS-Teilnehmer als Text-Stringdaten. Wenn die Text-Zeichenfolgedaten eine ungerade Anzahl von Bytes sind, wird am Ende "0x00" hinzugefügt. Wenn die Text-Zeichenfolgedaten eine gerade Anzahl von Bytes sind, wird der Adresse nach der letzten Adresse "0x0000" hinzugefügt. In eine Zelle können bis zu 32 einzelne Bytezeichen eingegeben werden.
- Wenn eine CSV-Datei zwei oder mehr Datenzeilen enthält, kann die erwünschte Zeilenanzahl angefangen bei der festgelegten Zeile ausgelesen werden. Bis zu 200 einzelne Bytezeichen können in eine Zeile und bis zu 65535 Zeilen können in eine CSV-Datei eingegeben werden.
- Bei Auftreten eines Fehlers wird der Fehlerzustand in LS9137 geschrieben (LS9143 für den USB-Speicher).
- Bei Schreiben der CSV-Datei-Textdaten in den LS-Teilnehmer, hängt die Datenspeicher-Reihenfolge vom Datenspeicher-Modus ab.

Fehlerzustand



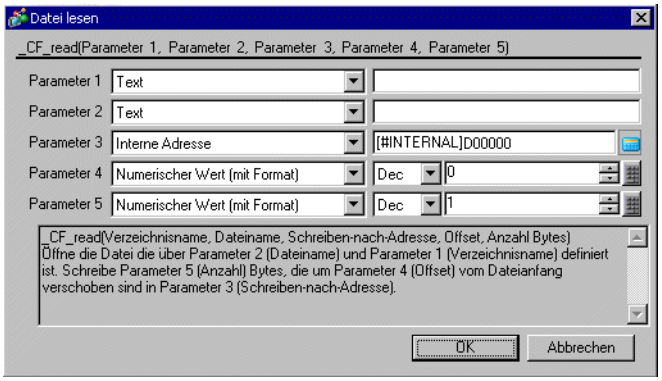


Editor-Funktionsname	LS-Bereich	Fehlerstatus	Ursache
_CF_read_csv ( )/ _USB_read_csv ( )	LS9137/ LS9143	0000h	Erfolgreich abgeschlossen.
		0001h	Parameter-Fehler
		0002h	CF-Karte/USB-Speicherfehler Keine CF-Karte oder USB-Speicher/Datei offen/ Dateilesefehler
		0003h	Schreibfehler/Lesefehler

WICHTIG

- Wenn " ∗ " als Ordnername angegeben wird, kann der vollständige Pfad als Dateiname festgelegt werden.
- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Dateinamen, die länger als dieses Format sind, sind nicht zulässig.
- Der tatsächliche LS-Teilnehmerbereich, um die von einer CSV-Datei importierten Daten zu speichern, ist begrenzt auf den vorgesehenen Benuzterbereich (LS20 bis LS2031 sowie LS2096 bis LS8191.)
- Die für den Datenimport erforderliche Verarbeitungszeit ist proportional zur Datenmenge der auszulesenden CSV-Datei. Das Element wird erst nach Abschluss der Bearbeitung aktualisiert. (Für das Auslesen der Daten von der ersten bis zur hundertsten Zeile einer CSV-Datei, mit 40 Zeichen pro Zeile, sind ungefähr 10 Sekunden erforderlich.)
- Im Gegensatz zur Funktion "_CF_read()/_USB_read()" wird der Status nicht sofort nach Ausführung der Funktion in [s:CF_ERR_STAT] gespeichert. (Manchmal können undefinierte Werte gespeichert werden.)
- Stellen Sie sicher, dass ["] am Anfang und Ende der Textstrings, die mit einem Zahlzeichen beginnen, eingefügt wird.
  Zum Beispiel:
  [123, 2-D4EA] [123, "2-D4EA"]
  X O

■ Datei lesen

Element	Beschreibung
Zusammen-fassung	Liest die spezifizierten Daten-Bytes nach dem spezifizierten Offset in die Datei und schreibt sie in die Zieladresse. Weitere Einzelheiten über die Datenspeicher-Reihenfolge finden Sie unter "Datenspeicher-Modus".
Format	_CF_read/_USB_read (Ordnernamen, Dateinamen, in Adressen speichern, Offset, Anzahl der Bytes)  Parameter 1 Ordnername: Fester String (maximale Länge: 32 einzelne Bytezeichen) Parameter 2 Dateiname: Fester String, interne Adresse, interne Adresse + temporäre Adresse (maximale Länge: 32 einzelne Bytezeichen) Parameter 3 Schreibzieladresse: interne Adresse, interne Adresse + Temporäre Adresse Parameter 4 Offset: Numerischer Wert, interne Adresse, temporäre Adresse (maximale Anzahl, die festgelegt werden kann: 65535 für 16-Bit-Länge, 4294967295 für 32-Bit-Länge) Parameter 5 Anzahl der Bytes: Numerischer Wert, interne Adresse, temporäre Adresse (verfügbare Länge: 1280)

Beispielausdruck:

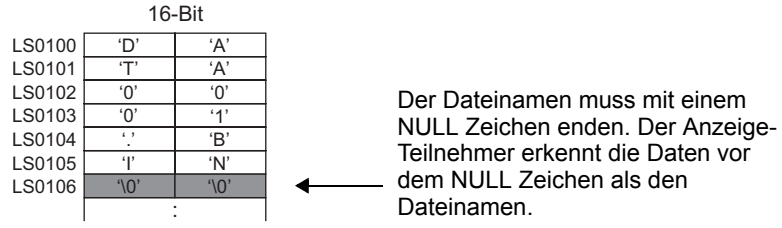
Zum Lesen von 16 Datenbytes in der festgelegten Datei bei einem Offset von 16:

_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 16, 16)

Im o.g. Beispiel werden 16 Datenbytes von dem 17. und darauffolgenden Bytes in die Datei "\DATA\DATA0001.BIN" in LS0100 und in darauffolgende Bereiche geschrieben.

WICHTIG

- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Lange Dateinamen sind nicht zulässig.
- Die maximal zulässige Anzahl von Zeichen für den ersten Parameter (Ordnername) und den zweiten Parameter (Dateiname) beträgt 32 Einzelbytezeichen.
- Die interne Adresse kann für den zweiten Parameter (Dateiname) bestimmt werden. Durch Bestimmen der internen Adresse kann ein Dateiname indirekt adressiert werden. Zudem können bis zu 32 Einzelbytezeichen zum Festlegen eines Dateinamens verwendet werden.
  Beispiel
  Zum Lesen von in einer Datei gespeicherten 10 Datenbytes, wenn die Datei in LS0100 und später angegeben ist und bei einem Offset von 0:
  _CF_read ("\DATA", [w:LS0100], [w:LS0200], 0, 10)
  Das Speichern eines Dateinamens in LS0100 lässt die indirekte Adressierung eines Dateinamens zu. In diesem Beispiel wird ein Dateiname in LS0100 bis LS0106 folgendermaßen gespeichert.



  Im o.g. Beispiel werden 10 Datenbytes am Anfang der Datei "\DATA\DATA0001.BIN" in LS0200 und in darauffolgende Bereiche geschrieben.

- Die Anzahl der Bytes, die erfolgreich gelesen wurden, werden in die ausgelesenen Bytes der CF-Karte/des USB-Speichers geschrieben [s:CF_READ_NUM]/[s:USB_READ_NUM]. Weitere Einzelheiten entnehmen Sie bitte "21.11.5 CF-Dateiverfahren/USB-Dateiverfahren CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus" (seite 21-117) .
- Die in dem "Dateinamen" und der "Schreibzieladresse" festgelegte interne Adresse zählt nicht zu den D-Skript-Adressen.
- Beim Bestimmen eines Teilnehmers für die Schreibzieladresse, ist das Schreiben der Daten in das SPS aufgrund einer höheren Anzahl an Worten (Bytes) zeitaufwendiger. Abhängig von der Anzahl der Worte könnte es ein paar Sekunden dauern.
- Das Überschreiten der von der Datei ausgelesenen Daten des angegebenen Teilnehmerbereichs des SLS verursacht einen Kommunikationsfehler. In diesem Fall muss die Stromzufuhr zum SPS AUS geschaltet und dann wieder AN geschaltet werden, um das SPS rückzusetzen.

Fortsetzung

WICHTIG

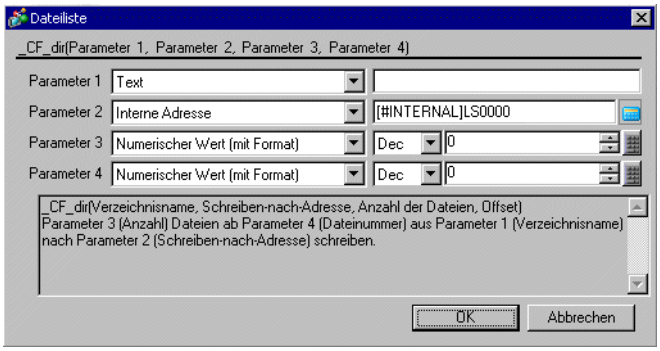
- Bei Spezifizieren eines Teilnehmers als Ziel werden die Werte aufgrund der Übertragungszeit zwischen GP und SPS nicht sofort geschrieben.
  Beispiel
  Im untenstehenden Skript liest Anweisung (1) 10 Datenbytes von der Datei in [w:D0100]. Die Daten wurden jedoch bei Ausführung der Anweisung (2) aufgrund der Übertragungszeit noch nicht in [w:[PLC1]D0100] geschrieben.

```
_CF_read ("\DATA", "DATA0001.BIN", [w:[PLC1]D0100], 0, 10) .....(1)
[w:[PLC1]D0200] = [w:[PLC1]D0100] + 1 ..... (2)
```

  In diesem Fall werden die Daten einmal im LS-Bereich gespeichert und die zweite Anweisung folgendermaßen ausgeführt.

```
_CF_read ("\DATA", "DATA0001.BIN", [w:[PLC1]D0100], 0, 10)
memcpy ([w:[#INTERNAL]LS0100], [w:[PLC1]D0100], 10)
[w:[PLC1]D0200] = [w:[#INTERNAL]LS0100] + 1
```

■ Ausgabe-Dateiliste

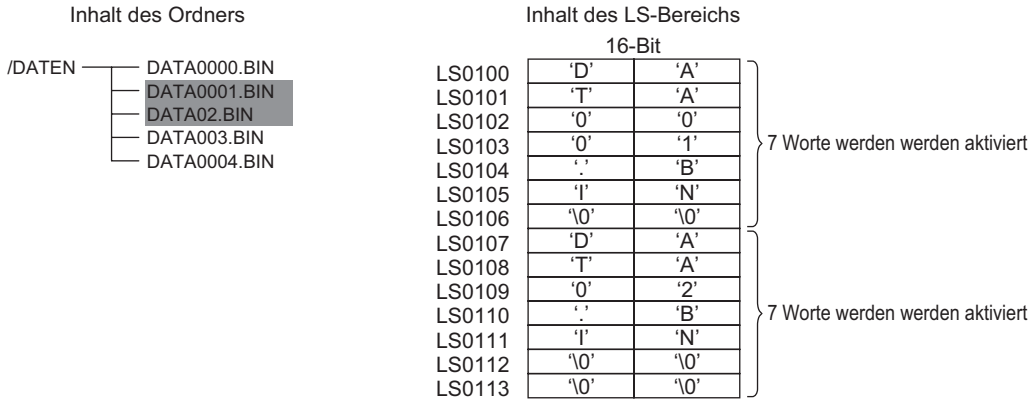
Element	Beschreibung
Zusammen-fassung	Die in dem festgelegten Ordner existierende Dateiliste wird in die interne Adresse geschrieben. Parameter 1 zeigt den CF-Kartenordner an. Parameter 4 bezeichnet das für die Wahl einer Datei/Dateien aus diesem Ordner verwendete Offset. Parameter 3 bezeichnet die Anzahl der aus diesem Ordner gewählten Dateien. Parameter 2 bezeichnet den LS-Bereich, in den die Dateien geschrieben werden. Wenn für das Offset "0" festgelegt wurde, beginnt die Liste mit der ersten (Start-) Datei.
Format	_CF_dir/_USB_dir (Ordnernamen, in Adressen speichern, Anzahl der Dateien, Offset)  Parameter 1 Ordnername: Fester Text (maximale Länge: 32 einzelne Bytezeichen) Parameter 2 Schreibzieladresse: Interne Adresse, interne mit Offset gekennzeichnete Adresse Parameter 3 Anzahl der Dateien: Numerischer Wert, interne Adresse, temporäre Adresse (verfügbare Länge: 32) Parameter 4 Numerischer Wert, interne Adresse, temporäre Adresse

Beispielausdruck:

Zur Ausgabe einer Dateiliste mit zwei Dateien, bei einem Offset 1 (zweite Datei):

_CF_dir ("\DATA*.*", [w:[#INTERNAL]LS0100], 2, 1)

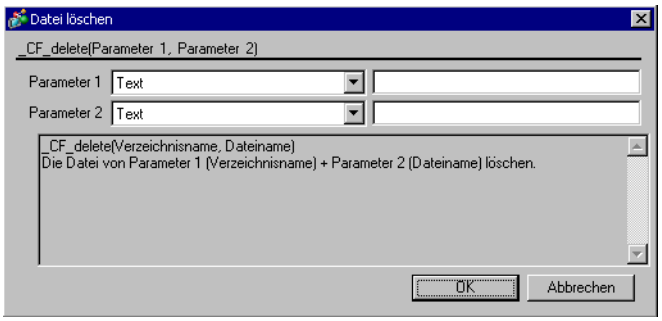
Wenn die o.g. Anweisung ausgeführt wird, während die folgenden Dateien im Ordner DATA existieren, werden Dateinamen "DATA0001.BIN" und "DATA02.BIN" in LS0100 und spätere Bereiche geschrieben.



WICHTIG

- Wenn für das Offset "0" festgelegt wurde, beginnt die Liste mit der ersten (Start-) Datei.
- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Lange Dateinamen sind nicht zulässig.
- Falls der angegebene Ordner nicht genügend festgesetzte Dateien enthält, wird der Rest des LS-Bereichs mit NULL-Zeichen ('\0') belegt.
- Wenn ein Dateiname weniger als 12 Zeichen hat, werden die leeren Positionen mit NULL-Zeichen belegt ('\0').
- Wenn Sie einen Ordnernamen festlegen, z.B. "\DATA*.*", vervollständigen Sie diesen um "*.*". Der Stern *.* bedeutet, dass alle Dateien dargestellt werden.
- Die Anzahl der tatsächlich aufgeführten Dateien wird in die aufgelisteten Dateien der CF-Karte/des USB-Speichers geschrieben [s:CF_FILELIST_NUM]/[s:USB_FILELIST_NUM].
  Details erhalten Sie unter " CF-Karten-Fehlerstatus/USB-Speicher-Fehlerstatus" (seite 21-117)
- Schreibziel-LS-Adressen zählen nicht als D-Skript-Adressen.
- Die Dateinamen werden unsortiert in den LS-Bereich geschrieben. Sie werden in der Erstellungsreihenfolge (die Reihenfolge des FAT-Eintrags) geschrieben.
- Die Liste kann durch das Angeben eines Dateinamenszusatzes erstellt werden. Um Dateien mit einer bestimmten Erweiterung aufzuführen, verwenden Sie bitte ein Format wie beispielsweise "\DATA*.BIN". Das Zeichen "*" kann jedoch nicht innerhalb eines Dateinamens verwendet werden.

■ Datei löschen

Element	Beschreibung
Zusammen-fassung	Löscht die angegebene Datei von der CF-Karte. Parameter 1 zeigt den CF-Kartenordner an. Parameter 2 bestimmt den Dateinamen der zu löschenden Datei.
Format	_CF_delete/_USB_delete (Ordnernamen, Dateinamen) Der Dateiname kann auch indirekt mit der LS-Adresse festgelegt werden.  Parameter 1 Ordnername: Fester Text Parameter 2 Dateiname: Fester Text, interne Adresse, interne Adresse + temporäre Adresse

Beispielausdruck:

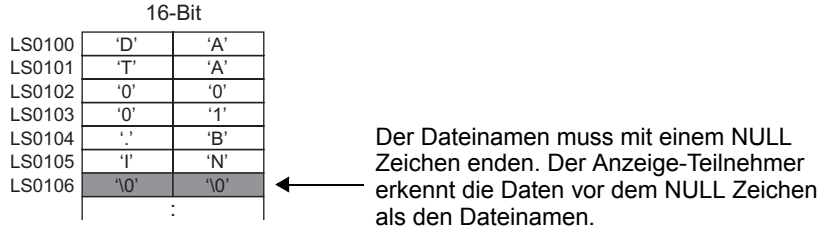
_CF_delete ("\DATA", "DATA0001.BIN")

Im o.g. Beispiel wird die Datei "\DATA\DATA0001.BIN" entfernt.

WICHTIG

- Für den Dateinamen ist nur das "8.3-Format" (maximal 12 Zeichen mit 8 Zeichen für den Dateinamen und 3 Zeichen für den Dateinamenszusatz) zulässig. Lange Dateinamen sind nicht zulässig.
- Die maximal zulässige Anzahl von Zeichen für den ersten Parameter (Ordnername) und den zweiten Parameter (Dateiname) beträgt 32 Einzelbytezeichen.
- Die interne Adresse kann für den zweiten Parameter (Dateiname) bestimmt werden. Durch Bestimmen der internen Adresse kann ein Dateiname indirekt adressiert werden. Zudem können bis zu 32 Einzelbytezeichen zum Festlegen eines Dateinamens verwendet werden.

  In diesem Beispiel wird ein Dateiname in LS0100 bis LS0106 folgendermaßen gespeichert.



  Im o.g. Beispiel wird die Datei "\DATA\DATA0001.BIN" entfernt.
- Zur Angabe eines Ausgangsordners (Verzeichnis) wird " " (leere Zeichenfolge) als Ordnername festgelegt.
- Wenn der LS-Bereich für "Dateiname" festgelegt wird, zählen "Schreibzieladressen" nicht als D-Skript-Adressen.
- Zur Angabe eines ganzen Pfads für einen Dateinamen wird ""∗" (Stern) als Ordnername festgelegt.

21.11.6 Drucker-Operationen

Drucker-Operationen	Funktionszusammenfassung
	Beschriftungsvariablen ☞" ■ Beschriftungsvariablen" (seite 21-145) Von den Steuerelement- und Status-Variablen festgelegt.
	Senden ☞" ■ Senden" (seite 21-147) COM-Port Ausgabe der angegebene Anzahl von Bytes.

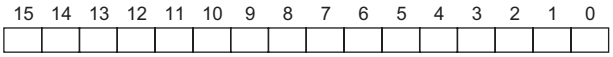
WICHTIG
- COM1 oder USB/PIO (USB-PIO) sind Ports, die als Drucker-Operationsfunktionen verwendet werden können.

■ Beschriftungsvariablen

Kontrolle

Das Steuerelement (PRN_CTRL) ist eine Variable zum Löschen des Sendepuffers und des Fehlerstatus. Diese Steuerelementvariable ist nur schreibbar.

- Steuerelement (PRN_CTRL) Zusammenfassung



Bit	Inhalt
15	Reserviert
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	1: Fehler löschen
1	Reserviert
0	1: Sendepuffer löschen

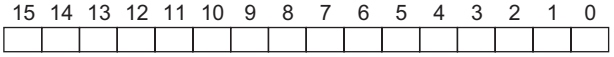
WICHTIG
- Bei Angabe eines Wortes (bei gleichzeitigem gesetzten zwei oder mehreren Bits) wird in folgender Reihenfolge verarbeitet:
  Fehler löschen

  Sendepuffer löschen
- Bitte keine reservierten Bits verwenden. Bestimmen Sie nur die benötigten Bits.

Status

Die Statusvariable (PRN_STAT) dient dazu, das Vorhandensein/Nicht Vorhandensein von Daten im Sendepuffer und den Fehlerstatus zu überprüfen. Diese Statusvariable ist nur schreibbar.

- Inhalt der Statusvariable (PRN_STAT)

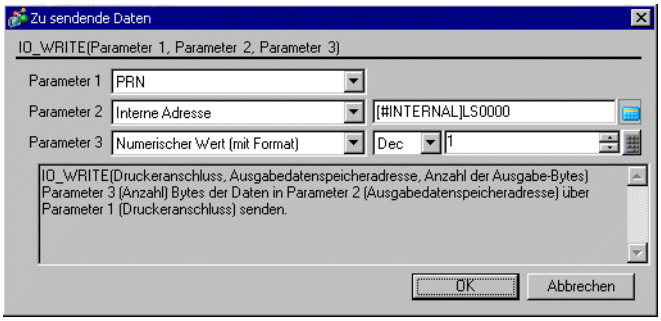


Bit	Inhalt
15	Reserviert
14	Der Status des I/F FEHLER-Signals des Druckers. Fehler drucken (Eingabe): 0: Fehler 1: Normal
13	Der Status des I/F SLCT-Signals des Druckers. Auswählen (Eingabe): 0: Offline 1: Online
12	Der Status des I/F PE-Signals des Druckers. Fehler drucken (Eingabe): 0: Normal 1: Kein Papier (Eingabe):
11	Reserviert
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	0: Normal 1: Sende-Fehler
0	0: Daten bestehen im Sendepuffer 1: Sendepuffer ist leer.

WICHTIG

- Bei Überlaufen des Sendepuffers tritt ein Fehler auf. Bei Auftreten dieses Fehler geht das Übertragungsfehler-Bit AN.
- Der Sendepuffer ist 8192 Bytes.
- Die reservierten Bits könnten künftig zugewiesen werden. Stellen Sie deshalb sicher, dass nur die erforderlichen Bits überprüft werden.

■ Senden

Element	Beschreibung
Zusammen-fassung	COM-Port Ausgabe der angegebene Anzahl von Bytes. Die Daten werden ungeachtet des angegebenen Druckertyps ausgegeben.
Format	IO_WRITE ([p:PRN], Ausgabedaten-Speicheradresse, Anzahl der ausgegebenen Bytes)  Parameter 1: [p:PRN] Parameter 2: Interne Adresse Parameter 3: Ganzzahlwert, Teilnehmeradresse, Temporäre Adresse

WICHTIG

- Der maximale Wert, der für Parameter 3 festgelegt werden kann, ist 1024. Selbst wenn höhere Werte als 1024 spezifiziert werden, gibt der COM-Port nur 1024 Datenbytes aus.

Beispielausdruck 1:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 10)

Im o.g. Beispiel werden 10 Datenbytes in LS1000 gespeichert und darauffolgende Bereiche werden über den COM-Port ausgegeben.

Beispielausdruck 2:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS0800])

Im o.g. Beispiel werden Daten in LS1000 gespeichert und darauffolgende Bereiche werden über den COM-Port ausgegeben. Die Anzahl der Bytes ist gleich hoch wie die der in LS0800 geschriebene.

Beispielausdruck 3:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS 1000], [t:0010])

Im o.g. Beispiel werden Daten in LS1000 gespeichert und darauffolgende Bereiche werden über den COM-Port ausgegeben. Die Anzahl der Bytes ist gleich hoch wie die der in die temporäre Adresse geschriebene [t:0010].

Datenspeicher-Modus

Bei Auslesen von Teilnehmeradressen während des Ausführens der Funktion COM-Port-Operation kann die Speicherreihenfolge der ausgelesenen Daten festgelegt werden.
Bei Einstellen des Datenspeichermodus in LS9130 könnte sich die Speicherreihenfolge ändern.
Der Modus kann aus vier Optionen gewählt werden: 0, 1, 2 oder 3.

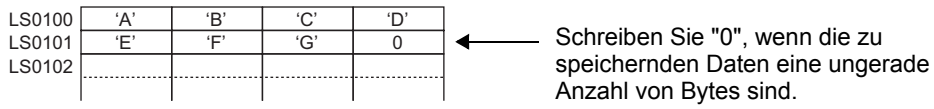
◆ Modus 0

z.B.: bei Verwendung der COM-Port-Operation, um den String "ABCDEFG" von einer internen Adresse zu lesen

[w:[#INTERNAL]LS9130] = 0
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- Wenn die Länge der internen Adresse 16-Bit ist

- Wenn die Länge der internen Adresse 32-Bit ist



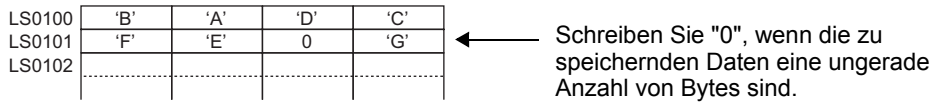
◆ Modus 1

z.B.: bei Verwendung der COM-Port-Operation, um den String "ABCDEFG" von einer internen Adresse zu lesen

[w:[#INTERNAL]LS9130] = 1
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- Wenn die Länge der internen Adresse 16-Bit ist

- Wenn die Länge der internen Adresse 32-Bit ist



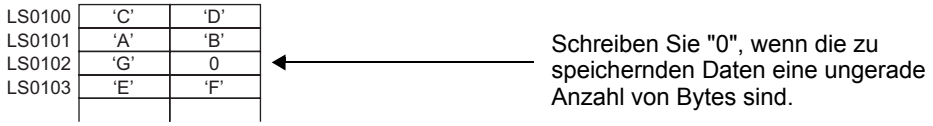
◆ Modus 2

z.B.: bei Verwendung der COM-Port-Operation, um den String "ABCDEFG" von einer internen Adresse zu lesen

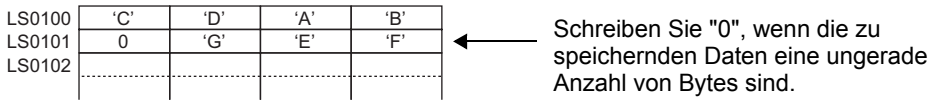
[w:[#INTERNAL]LS9130] = 2

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- Wenn die Länge der internen Adresse 16-Bit ist



- Wenn die Länge der internen Adresse 32-Bit ist



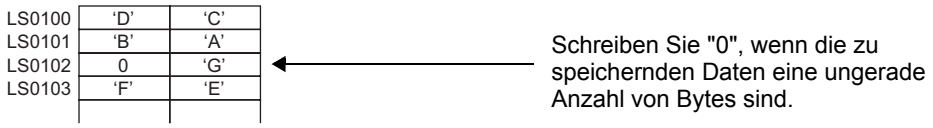
◆ Modus 3

z.B.: bei Verwendung der COM-Port-Operation, um den String "ABCDEFG" von einer internen Adresse zu lesen

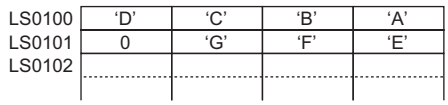
[w:[#INTERNAL]LS9130] = 3

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- Wenn die Länge der internen Adresse 16-Bit ist



- Wenn die Länge der internen Adresse 32-Bit ist



Schreiben Sie "0", wenn die zu speichernden Daten eine ungerade Anzahl von Bytes sind.

WICHTIG

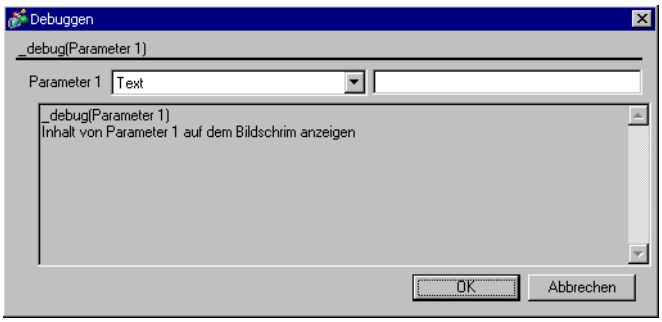
- In der Systemeinstellung sind Datenspeichermodus und String-Datenmodus nicht gleichzusetzen. Die Beziehung zum String-Datenmodus wird in der folgenden Tabelle veranschaulicht.

Daten Speicher-reihenfolge	Byte in Wort LH/HL Speicher-reihenfolge	In Doppelwort LH/HL Speicher-reihenfolge	D-Skript-Daten-speicher-Modus	Text-Doppelworts truktur
Speichern ab Start-Daten	HL-Reihenfolge	HL-Reihenfolge	0	1
	LH-Reihenfolge		1	2
	HL-Reihenfolge	LH-Reihenfolge	2	5
	LH-Reihenfolge		3	4
Speichern ab End-Daten	HL-Reihenfolge	HL-Reihenfolge	–	3
	LH-Reihenfolge		–	7
	HL-Reihenfolge	LH-Reihenfolge	–	8
	LH-Reihenfolge		–	6

21.11.7 Anderes

Weitere	Funktionszusammenfassung
	Debug-Funktion ☞" ■ Debug-Funktion" (seite 21-151) Bildschirmanzeige der angegebenen Adresse oder des Textes zu deren Fehlerbeseitigung.
	Anwendungsauslöser ☞" ■ Anwendungsauslöser" (seite 21-153) Führt den bestimmten Bereich aus und startet die Anwendung.
	WinGP, Beenden ☞" ■ WinGP beenden" (seite 21-155) WinGP beenden.

■ Debug-Funktion

Element	Beschreibung
Zusammen-fassung	Bildschirmanzeige der angegebenen Adresse oder des Textes zu deren Fehlerbeseitigung. Löschen Sie nach der Fehlerbeseitigung das Optionsfeld [Debug-Funktion aktivieren] zum Schließen der Debug-Funktion. Nur der Debug-Bildschirm wird nicht angezeigt.
Format	_debug (Parameter 1)  Parameter 1: Text (bis zu 32 Einzelbytezeichen, 16 Doppelbytezeichen)

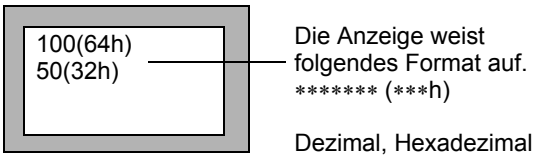
Inhalt des Parameter 1

Parameter 1	Format	Beschreibung
Text	_debug ("ABC")	Zeigt den Text in " " an. Eine Textlänge von bis zu 32 einzelner Bytezeichen ist erlaubt.
Wortadresse oder temporäre Adresse	_debug (w:[PLC1]D1000)	Anzeige des Wertes der angegebenen Wortadresse oder der temporären Adresse.
Zeilenvorschub	_debug (_CRLF)	Bewegt den Cursor an den Anfang der nächsten Zeile.
Wagenrücklauf	_debug (_CR)	Bewegt den Cursor an den Anfang derselben Zeile.

◆ Beispielausdruck 1:

Das folgende Skript zeigt den Wert der Wortadresse an.

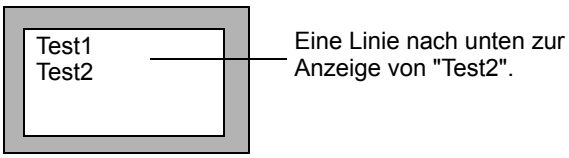
```
[w:[#INTERNAL]LS0100]=100
_debug ([w:[#INTERNAL]LS0100])
_debug (_CRLF)
[w:[#INTERNAL]LS0100]=50
_debug ([w:[#INTERNAL]LS0100])
```



◆ Beispielausdruck 2:

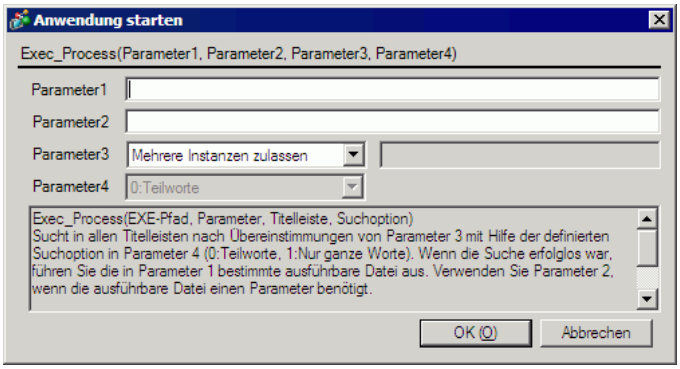
Das folgende Skript zeigt einen Zeilenvorschub sowie Text an.

```
_debug ("Test1")
_debug (_CRLF)
_debug ("Test2")
```



■ Anwendungsauslöser

Diese Funktion funktioniert nur bei Modellen der IPC Series.

Element	Beschreibung
Zusammen-fassung	Führt den bestimmten Bereich aus und startet die Anwendung. Sie können Einstellungen, wie beispielsweise Anlaufparameter und die Überwachung bei Multiplex-Start bestimmen.
Format	Exec-Prozess (Parameter 1, Parameter 2, Parameter 3, Parameter 4)  Parameter 1 EXE-Pfad: Geben Sie den absoluten Pfad der ablaufbereiten Datei (.exe) für die zu startende Anwendung ein. Sie können maximal 255 Zeichen eingeben. Parameter 2 Parameter :Geben Sie das Start-Argument der ablaufbereiten Datei ein. Sie können maximal 255 Zeichen eingeben. Parameter 3 Fenstertitel: Wenn Sie nicht mehrere Instanzen zulassen möchten, wählen Sie bitte "Mehrere Instanzen nicht zugelassen" aus und geben den [Fenstertitel] ein. Sie können maximal 63 Zeichen eingeben. Die Anwendung kann nicht ausführen, wenn ein Fenster festgestellt wird, das gleich ist wie der [Fenstertitel]. Mehrere Instanzen sind zulässig, wenn Sie [Mehrere Instanzen zulassen] ausgewählt haben, oder wenn [Fenstertitel] nicht bestimmt wurde. Parameter 4 Nur ganze Fenstertitel suchen: Nur aktiviert, wenn Sie Parameter 3 - "Mehrere Instanzen nicht zugelassen" ausgewählt haben. Wenn "0: Teilworte" ausgewählt wurde, wird die bestimmte Anwendung nicht ausgeführt, wenn ein Fenster festgestellt wird, das einen ähnlichen Titel aufweist, wie im [Fenstertitel]. Wenn "1: Nur ganze Worte" ausgewählt wurde, wird die bestimmte Anwendung nicht ausgeführt, wenn ein Fenster festgestellt wird, das denselben Titel aufweist, wie im [Fenstertitel].

ANMERKUNG
- Parameter 1 benötigt Test (EXE-Pfad). Es wird ein Fehler auftreten, wenn kein Text eingegeben wird.
- Diese Funktion funktioniert nur bei Modellen der IPC Series.

Parameter 1 (EXE-Pfad) Eingabemethode

Der EXE-Pfad kann auf drei verschiedene Arten eingegeben werden:
Nachstehend wird ein Beispiel zur Ausführung einer Musterdatei (.exe) in C:\Documents und Settingsuser\Local Settings\Temp beschrieben.

1. Spezifikation des vollständigen Pfads
   Zum Beispiel: C:\Documents and Settings\user\Local Settings\Temp\sample.exe
2. Nur EXE-Name:
   Wenn sich die ausführbare Datei in einem Verzeichnis befindet, das als Pfad in den Umgebungseinstellungen auf der IPC Series bestimmt wurde.
   Zum Beispiel: sample.exe
   (Starten, wenn sich die Einstellungen im Pfad =C:\Documents and Settings\user\Local Settings\Temp befinden)
3. Pfad mit Umgebungsvariable bestimmen
   Wenn sich die ausführbare Datei in einem Verzeichnis befindet, das durch die Umgebungsparameter in den Umgebungseinstellungen auf der IPC Series bestimmt wurde.
   Zum Beispiel: %TEMP%\sample.exe
   (Starten, wenn die Umgebungsparameter als TEMP=C:\Documents and Settings\user\Local Settings\Temp festgelegt wurden)

Beispielausdruck 1:

Mehrere Instanzen zulassen (Starten des Notizblocks und Anzeige der Datei Readme.txt)
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","",0)
Exec_Process ("%SystemFolder%\notepad.exe","D:\TEMP\Readme.txt","",1)

Beispielausdruck 2:

Mehrere Instanzen nicht zugelassen:
Teilworte (Starten des Notizblocks und Anzeige der Datei Readme.txt).Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","Readme",0)

Beispielausdruck 3:

Mehrere Instanzen nicht zugelassen: Nur ganze Worte (Starten des Notizblocks und Anzeige der Datei Readme.txt).
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","Readme.txt - Notepad",1)

Beispielausdruck 4:

Mehrere Instanzen nicht zugelassen: Teilworte (Starten des Notizblocks)
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","","Notepad",0)

Beispielausdruck 5:

Kein Parameter (Starten des Notizblocks)
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","","",0)

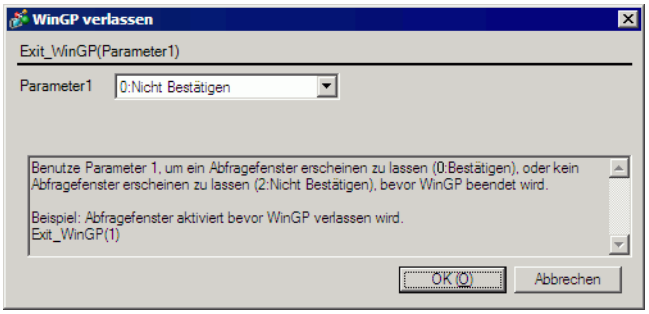
Beispielausdruck 6:

Mehrere Parameter (Starten der Datei sample.exe)

Exec_Process ("C:\WINDOWS\SYSTEM32\sample.exe","/v /a/s","",1)

■ WinGP beenden

Diese Funktion funktioniert nur bei Modellen der IPC Series.

Element	Beschreibung
Zusammen-fassung	WinGP beenden. Sie können eine Quittierungsmeldung beim Beenden anzeigen.
Format	Exit_WinGP(Parameter1)  Parameter 1 Ordnername: Wählen Sie "0: Nicht bestätigen" oder "1: Bestätigen" aus.

ANMERKUNG

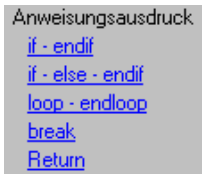
- Parameter 1 benötigt Test (EXE-Pfad). Es wird ein Fehler auftreten, wenn kein Text eingegeben wird.
- Die Funktion wird nicht funktionieren, wenn das Skript "Exit WinGP" auf ein Modell außer der IPC Series übertragen wird.

Beispielausdruck:

Sie können eine Quittierungsmeldung beim Beenden anzeigen.

Exit_WinGP(1)

21.11.8 Bedingte Ausdrücke

Bedingte Ausdrücke	Funktionszusammenfassung
	if - endif ☞ " ■ if - endif" (seite 21-156) Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt.
	if - else - endif ☞ " ■ if - else - endif" (seite 21-156) Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt. Wenn eine Bedingung falsch ist, wird die auf "sonst" folgende Anweisung ausgeführt.
	Schleife - Endlosschleife ☞ " ■ Schleife - Endlosschleife" (seite 21-157) Der Schleifen- (Wiederholungs-) Vorgang wird abhängig von der auf die Schleife folgende eingeklammerte "( )" Nummer wiederholt.
	break ☞ " ■ break" (seite 21-160) Hält Schleifen-Operation während Ausführung der Schleifen ( ) Gleichung an.
	return ☞ " ■ Rücklauf" (seite 21-160) Führt wieder von Anfang an aus. Kann nur in einem erweiterten Skript verwendet werden.

■ if - endif

Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt.

ANMERKUNG

- Das Zuordnungszeichen "=" kann nicht in einem bedingten Ausdruck verwendet werden.

■ if - else - endif

Wenn eine eingeklammerte, auf "wenn" folgende Bedingung "( )" wahr ist, wird die auf den "wenn ( )" Ausdruck folgende Anweisung ausgeführt. Wenn eine Bedingung falsch ist, wird die auf "sonst" folgende Anweisung ausgeführt.

ANMERKUNG

- Das Zuordnungszeichen "=" kann nicht in einem bedingten Ausdruck verwendet werden.

■ Schleife - Endlosschleife

Der Schleifen- (Wiederholungs-) Vorgang wird abhängig von der auf die Schleife folgende eingeklammerte "( )" Nummer wiederholt.

Endlosschleife

Die Schleife ist endlos, wenn keine Anweisung in den Klammern ( ) der Schleife angegeben ist.
In erweiterten Skripten können Endlosschleifen verwendet werden.

Beispielausdruck:

```
loop ( )
{
    [w:[#INTERNAL]LS0100]=[w:[#INTERNAL]LS0100]+1
    if ( [w:[#INTERNAL]LS0100] >10)
    {
        break
    }
    endif
}
Endlosschleife
```

ANMERKUNG

- Das Schleifen ( ) Format ist wie folgt:
  Zum Beispiel:

```
Schleife (Anzahl der Schleifen) // Bestimmt die temporäre Adresse, die die Anzahl der Schleifen speichert.
  {
    Aktionsgleichung
    Abbruch // Zum Beenden der Schleife auf halbem Weg (Optional)
  } Schleifenende// Bestimmt das Ende der Schleife
```

- Nur eine temporäre Wortadresse kann eingegeben werden (in Klammern.) (Beispiel: loop ([t:000]))
- "Schleife ( )" kann für eine ausgelöste Gleichung nicht verwendet werden.
- Die temporäre Wortadresse, die zum Bestimmen der Anzahl der Schleifen verwendet wird, nimmt für jede Schleife ab. Wenn sich der Wert zu 0 ändert, ist die Operation der Schleife abgeschlossen. Beim Modifizieren des für die Anzahl der Schleifen designierten temporären Wortadressenwertes wird die Schleife endlos. Zudem wird die verwendete temporäre Wortadresse als global designiert. Deshalb kann die gleichzeitige Verwendung dieser temporären Wortadresse für andere Zwecke zu einer Endlosschleife führen.
- Die Bildschirmanzeige von Elementen usw. werden erst aktualisiert, wenn eine Schleifenoperationen beendet ist.
- Schleife ( ) kann auch verschachtelt sein. Bei Verschachtelung wird die innerste Schleife ( ) mit dem Befehl "Break" übersprungen.

```
loop ([t:0000])  // Schleife 1
{
    loop ([t:       // Schleife 2
    {
        break       // Unterbrechen der Schleife 2
    }Endlosschleife

    break           // Unterbrechen der Schleife 1
}Endlosschleife
```

- Falls die Schleifen-Operation ohne Anwendung des Unterbrechungsbefehls abgeschlossen wurde, beträgt der temporäre Wert der Wortadresse 0.

ANMERKUNG

- Der für den temporären Wortadressenwert verfügbare Bereich ist unterschiedlich, abhängig vom Datenformat (Bin, BCD), Bit-Länge und verwendetem +/– Code. Falls der +/- Code bestimmt wurde und die temporäre Wortadresse einen negativen Wert besitzt, wird die Bedingung am Anfang der Schleife beurteilt und das Schleifen-Verfahren gestoppt.
- VERWENDEN Sie KEINEN Teilnehmer in der Schleifenformel. Bedienen Sie sich anstatt dessen einer Adresse des internen LS-Benutzerbereichs des Geräts oder einer temporären Wortadresse. Die folgende Beschreibung schreibt die Daten beispielsweise vielfach in das SPS in einem kurzen Zeitraum (100 Mal im folgenden Beispiel.) Das kann einen Systemfehler hervorrufen, da die Kommunikationsverarbeitung (d.h. die für das Schreiben in den PC erforderliche Zeit) nicht in dieser Geschwindigkeit ausgeführt werden kann.

  Zum Beispiel:

```
[t:0000] = 100                                          // 100 loops
loop ([t:0000])
{
  [w:[PLC1]D0200] = [w:[#INTERNAL]LS0100]               // Schreiben in D0200
  [w:[#INTERNAL]LS0100] =                               // Inkrementierung
  [w:[#INTERNAL]LS0100]+1                               LS0100
}Endlosschleife
```

  Bitte folgendermaßen ändern:

```
[t:0000] = 100                                          // 100 loops
loop ([t:0000])
{
  [w:[#INTERNAL]LS0200] =                               // Schreiben in D0200
  [w:[#INTERNAL]LS0100]
  [w:[#INTERNAL]LS0100] =                               // Inkrementierung
  [w:[#INTERNAL]LS0100]+1                               LS0100
}Endlosschleife
[w:[PLC1]D0200]=[w:[#INTERNAL]LS0200]                   //LS0200 Inhalt,
                                                        schreiben in D0200
```

- Die Verwendung von "Schleife" oder "Break" als Funktionsnamen für eine D-Skript-Funktion verursacht einen Fehler.

■ break

Verlässt die Schleifenoperation in der Mitte der Schleifen- ( ) Operation

ANMERKUNG
- Der Befehl "Break" kann nur im Teil { } der Schleife ( ) verwendet werden.
- Skripte werden nicht ordnungsgemäß ausgeführt, wenn ein "Break"-Befehl in Wenn-Ausdrücken {} verwenden.

■ Rücklauf

Wenn die "Benutzerdefinierte Funktion" "Rücklauf" mit einschließt, wird die Ausführung der Funktion bgebrochen, und die Steuerung wird auf den Anrufer der Funktion zurückgesetzt.

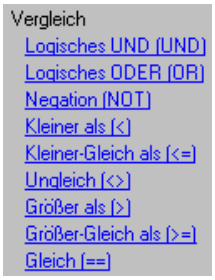
Schließt bei Ausführung (Hauptfunktion) "Rücklauf" ein
Die Ausführung der Hauptfunktion wird momentan abgebrochen und vom Anfang der Hauptfunktion aus neu gestartet.

ANMERKUNG
- Das Zuordnungszeichen "=" kann nicht in einem bedingten Ausdruck verwendet werden.

Beispielausdruck:

```
[w:[#INTERNAL]LS0100]=([w:[#INTERNAL]LS0200]>> 8) & 0xFF
if ([w:[#INTERNAL]LS0100]==0)        // Wenn LS0100 "0" ist, wird die Verarbeitung
                                     nicht mehr ausgeführt
{
   set([b:[#INTERNAL]LS005000])      // Legt die Bitadresse für die Fehleranzeige fest
   return                            // Ende
}
endif
```

21.11.9 Vergleich

Vergleichsoperation	Funktionszusammenfassung
	Logisches UND (and) ☞ " ■ Logisches UND (and)" (seite 21-161) N1 und N2: Ja, wenn N1 sowie N2 AN sind.
	Logisches ODER (or) ☞ " ■ Logisches ODER (or)" (seite 21-161) N1 oder N2: Ja wenn entweder N1 oder N2 AN sind.
	Negation (nicht) ☞ " ■ Negation (nicht)" (seite 21-161) notN1: Wird zu 0 wenn N1 1 ist und zu 1 wenn N1 0 ist.
	Kleiner als (<) ☞ " ■ Kleiner als (<)" (seite 21-162) Ja, wenn N1 größer als N2 ist (N1 < N2).
	Kleiner-Gleich als (=) ☞ " ■ Kleiner-Gleich als (<=)" (seite 21-162) Ja, wenn N1 kleiner als oder gleich hoch wie N2 (N1 <= N2) ist.
	Ungleich (<>) ☞ " ■ Ungleich (<>)" (seite 21-162) Ja, wenn N1 ungleich N2 (N1 <> N2) ist.
	Größer als (>) ☞ " ■ Größer als (>)" (seite 21-162) Ja, wenn N1 größer als N2 ist (N1 > N2).
	Größer-Gleich als (>=) ☞ " ■ Größer-Gleich (>=)" (seite 21-162) Ja, wenn N1 größer als oder gleich hoch wie N2 (N1 >= N2) ist.
	Gleich (==) ☞ " ■ Gleich (==)" (seite 21-162) Ja, wenn N1 gleich hoch wie N2 ist (N1 = N2).

■ Logisches UND (and)

Logische UND-Operationen auf die linke und rechte Seite. Der Wert 0 (Null) gilt als AUS, und andere Werte als AN.
N1 und N2: Ja, wenn N1 sowie N2 AN sind. Andernfalls nein.

■ Logisches ODER (or)

ODER Operationen auf der rechten oder linken Seite. Der Wert 0 (Null) gilt als AUS, und andere Werte als AN.
N1 oder N2: Ja wenn entweder N1 oder N2 AN sind. Andernfalls nein.

■ Negation (nicht)

Invertiert den Wert. Der Wert 0 (null) gilt als 1, und andere Werte als 0.
notN1: Wird zu 0 wenn N1 1 ist und zu 1 wenn N1 0 ist.

■ Kleiner als (<)

Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
Ja, wenn N1 größer als N2 ist (N1 < N2).

■ Kleiner-Gleich als (<=)

Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
Ja, wenn N1 kleiner als oder gleich hoch wie N2 (N1 <= N2) ist.

■ Ungleich (<>)

Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten. Ja, wenn N1 ungleich N2 (N1 <> N2) ist.

■ Größer als (>)

Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
Ja, wenn N1 größer als N2 ist (N1 > N2).

■ Größer-Gleich (>=)

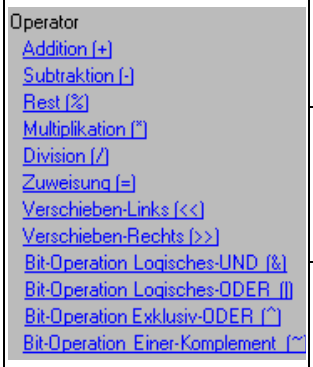
Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
Ja, wenn N1 größer als oder gleich hoch wie N2 (N1 >= N2) ist.

■ Gleich (==)

Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
Ja, wenn N1 gleich hoch wie N2 ist (N1 = N2).

Befehl		Zum Beispiel:
Logisches AND	UND	Wenn ((Operation) und (Operation))
Logisches OR	ODER	Wenn ((Operation) oder (Operation))
Negation	not	Wenn (nicht (Operation))
Kleiner als	<	(Ausdruck 1) < (Ausdruck 2)
Kleiner-Gleich als	<=	(Ausdruck 1) <= (Ausdruck 2)
Ungleich	<>	(Ausdruck 1) <> (Ausdruck 2)
Größer als	>	(Ausdruck 1) > (Ausdruck 2)
Größer-Gleich als (=)	>=	(Ausdruck 1) >= (Ausdruck 2)
Gleich	==	(Ausdruck 1) == (Ausdruck 2)

21.11.10 Operator

Operator	Funktionszusammenfassung
	Addition (+) ☞ " ■ Addition (+)" (seite 21-164) Fügt die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante hinzu.
	Subtraktion (–) ☞ " ■ Subtraktion (–)" (seite 21-164) Subtrahiert die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante.
	Modulus (%) ☞ " ■ Modulus (%)" (seite 21-164) Erkennt den Rest einer Datendivision in zwei Wortadressen oder in einer Wortadresse und einer Konstante.
	Multiplikation (∗) ☞ " ■ Multiplikation (∗)" (seite 21-164) Multipliziert die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante.
	Division (/) ☞ " ■ Division (/)" (seite 21-164) Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten.
	Zuweisung (=) ☞ " ■ Zuweisung (=)" (seite 21-164) Weisen Sie den Wert auf der rechten Seite dem Wert auf der linken Seite zu.
	Verschieben-Links (<<) ☞ " ■ Links verschieben (<<)" (seite 21-165) Verschiebt die Daten auf der linken Seite entsprechend der Nummer auf der rechten Seite nach links.
	Verschieben-Rechts (>>) ☞ " ■ Verschieben-Rechts (>>)" (seite 21-165) Verschiebt die Daten auf der linken Seite entsprechend der Nummer auf der rechten Seite nach rechts.
	Bit-Operator logisches UND (&) ☞ " ■ Bitweise UND (&)" (seite 21-165) Führt Logisches UND der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante.
	Bit-Operator logisches ODER (|) ☞ " ■ Bitweise ODER (|)" (seite 21-165) Führt Logisches ODER der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante.

Fortsetzung

Operator	Funktionszusammenfassung
Operator	Bit-Operator exklusives ODER (^) ☞ " ■ Bitweise Exklusiv ODER (^)" (seite 21-165) Führt Exklusives ODER der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante.
	Bit-Operation Einer-Komplement (~) ☞ " ■ Bitweise 1er Kompliment (~)" (seite 21-165) Invertiert die Bits.

■ Addition (+)

Fügt die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante hinzu. Wenn die Ergebnisse der Datenverarbeitung überfliessen, werden die Zahlen abgetrennt.

■ Subtraktion (–)

Subtrahiert die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante. Wenn die Ergebnisse der Datenverarbeitung überfliessen, werden die Zahlen abgetrennt.

■ Modulus (%)

Erkennt den Rest einer Datendivision in zwei Wortadressen oder in einer Wortadresse und einer Konstante. Das Operationsergebnis könnte vom Zeichen auf der linken und rechten Seite abhängig sein.

■ Multiplikation (*)

Multipliziert die Daten in zwei Wortadressen oder in einer Wortadresse und einer Konstante. Wenn die Ergebnisse der Datenverarbeitung überfliessen, werden die Zahlen abgetrennt.

■ Division (/)

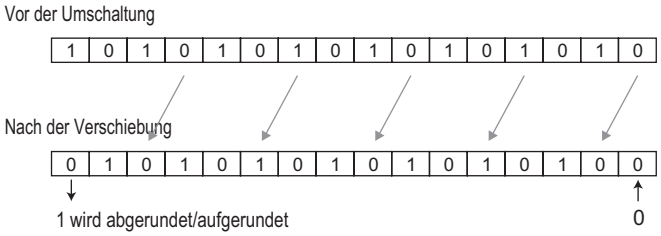
Vergleicht die Daten in zwei Wortadressen oder die Daten in einer Wortadresse und einer Konstanten. Bruchwerte aus der Operation werden verkürzt. Wenn die Ergebnisse der Datenverarbeitung überfliessen, werden die Zahlen abgetrennt.

■ Zuweisung (=)

Weisen Sie den Wert auf der rechten Seite dem Wert auf der linken Seite zu. Es können nur Adressen auf der linken Seite bestimmt werden. Adressen und Konstanten können auf der rechten Seite beschrieben werden. Wenn die Ergebnisse der Datenverarbeitung überfliessen, werden die Zahlen abgetrennt.

■ Links verschieben (<<)

Verschiebt die Daten auf der linken Seite entsprechend der Nummer auf der rechten Seite nach links. Diese Funktion unterstützt nur logische Verschiebungen.
z.B.: Verschieben-Links-Verfahren (verschiebt einen Bit nach links.)



■ Verschieben-Rechts (>>)

Verschiebt die Daten auf der linken Seite entsprechend der Nummer auf der rechten Seite nach rechts. Diese Funktion unterstützt nur logische Verschiebungen.

■ Bitweise UND (&)

Führt Logisches UND der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante. Wird zum Extrahieren eines spezifischen Bits oder zum Maskieren einer bestimmten Bitfolge verwendet.

■ Bitweise ODER (|)

Führt Logisches ODER der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante. Damit wird ein spezifisches Bit EIN geschaltet.

■ Bitweise Exklusiv ODER (^)

Führt Exklusives ODER der Daten zwischen Wort-Teilnehmern aus oder zwischen Wort-Teilnehmerdaten und Konstante.

■ Bitweise 1er Kompliment (~)

Invertiert die Bits.

ANMERKUNG

- Informationen über das durch Verkürzen gebrochener Werte oder von Operationsergebnissen hervorgerufenen Überlauf-Stellen finden Sie unter ☞ "21.10.4 Anmerkungen zu Operationsergebnissen" (seite 21-71)

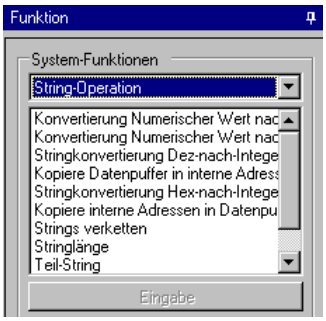
Präzedenz- und Assoziativitätsreihenfolge

In der folgenden Tabelle ist die Rangordnung der Operatoren dargestellt. Wenn zwei oder mehr Operatoren die gleiche Priorität haben, halten Sie sich an die mit der Assoziativität angegebene Richtung.

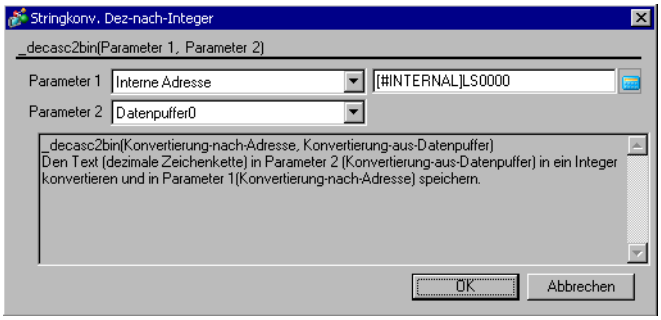
Priorität	Operator	Assoziativität
Hoch	( )	–>
	not ~	<–
	* / %	–>
	+ –	–>
	<< >>	–>
	< <= > >=	–>
	== <>	–>
	& ^ |	–>
	und oder	–>
Langsam	=	<–

21.11.11 String-Operation

Textoperationsfunktionen können nur in einem erweiterten Skript verwendet werden.

String-Operation	Funktionszusammenfassung
	Textkonvertierung Dez-nach-Ganzzahl ☞ " ■ Stringkonvertierung Dez-nach-Integer" (seite 21-168) Mit dieser Funktion wird Dezimal-Text in Ganzzahlen konvertiert.
	Textkonvertierung Hex-nach-Integer ☞ " ■ Stringkonvertierung Hex-nach-Integer" (seite 21-170) Mit dieser Funktion wird Hexadezimal-Text in Ganzzahlen konvertiert.
	Interne Adresse in Datenpuffer kopieren ☞ " ■ Interne Adresse an Datenpuffer" (seite 21-172) Die im Datenpuffer gespeicherten String-Daten werden in den Datenpuffer kopiert.
	Datenpuffer in interne Adresse kopieren ☞ " ■ Datenpuffer in interne Adresse kopieren" (seite 21-174) Die im Datenpuffer gespeicherten String-Daten werden in die interne Adresse kopiert.
	Status ☞ " ■ String-Operation Fehlerstatus" (seite 21-176) Speichert jeden aufgetretenen Fehler.
	Konvertierung Numerischer Wert nach Dez-String ☞ " ■ Konvertierung numerischer Wert nach Dez-String" (seite 21-178) Mit dieser Funktion wird eine Ganzzahl in eine Dezimal-Zeichenfolge konvertiert.
	Konvertierung Numerischer Wert nach Hex-String ☞ " ■ Konvertierung numerischer Wert nach Hex-String" (seite 21-179) Mit dieser Funktion werden Binärdaten in einen Hexadezimal-String konvertiert.
	Teil-String kopieren ☞ " ■ Teil-Text" (seite 21-181) Daten werden von dem angegebenen Offset des String gemäß der Stringlänge wiedergewonnen und in einem anderen Datenpuffer gespeichert.
	Texteinstellungen ☞ " ■ Datenpuffer setzen" (seite 21-182) Im Datenpuffer ist ein fester String gespeichert.
	String-Länge ☞ " ■ Textlänge" (seite 21-183) Besorgt die Länge des gespeicherten Strings.
	String-Verkettung ☞ " ■ String-Verkettung" (seite 21-184) Eine Zeichenkette oder ein Zeichencode ist mit dem Textpuffer verkettet.

■ Stringkonvertierung Dez-nach-Integer

Element	Beschreibung
Zusammen-fassung	Mit dieser Funktion wird ein Dezimal-String in Ganzzahlen konvertiert. Konvertieren Sie den ganzzahligen Hexadezimal-Text in Parameter 2 (Konvertierungsquell-Datenpuffer) in eine Ganzzahl, und speichern Sie diese in Parameter 1 (Konvertierungszieladresse.)
Format	_decasc2bin ([Konvertierungszieladresse], [Konvertierungsquell-Datenpuffer])  Parameter 1: Interne Adresse, Temporäre Adresse Parameter 2: Datenpuffer

Beispielausdruck 1 (Bei einer Datenlänge von 16 Bits)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

"databuf0" hat folgenden Inhalt:

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Die o.g. Daten werden folgendermaßen konvertiert.

	16-Bit
LS0100	1234

Beispielausdruck 2 (Bei einer Datenlänge von 32 Bits)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

"databuf0" hat folgenden Inhalt:

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	NULL

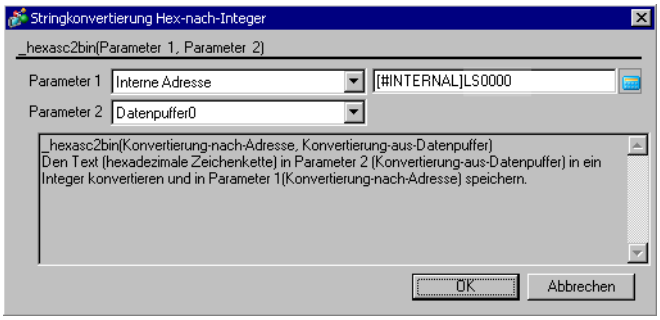
Die o.g. Daten werden folgendermaßen konvertiert.

	32 Bit
LS0100	12345678
LS0102	

WICHTIG

- Wenn die konvertierte Bit-Länge länger als die Bit-Länge des D-Skript-Editoren ist, tritt ein Fehler auf.
  z.B.: bei einer 16-Bit Skript-Länge:
  _strset (databuf0, "123456") //Bei versehentlichem Setzen eines sechstelligen Dezimal-Strings
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  Bei Ausführen der o.g. Ausdrucks wird Fehler Nr. 2 (String-Konvertierungsfehler) des String-Fehlerstatuses [e: Das Bit wird jedoch bei Auftreten eines Fehlers an den Anfang der Hauptfunktion rückgesetzt. Deshalb können nach Ausführung von _decasc2bin andere Funktionen nicht direkt referenziert werden. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)
- Bei der Konvertierung eines Datenstrings, der andere Zeichen außer "0" bis "9" enthält, tritt ein Fehler auf.
  z.B.: bei einer 16-Bit Skript-Länge:
  _strset (databuf0, "12AB") //Bei versehentlichem Setzen eines Nicht-Dezimal-Strings
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  Bei Ausführen der o.g. Ausdrucks wird Fehler Nr. 2 (String-Konvertierungsfehler) des String-Fehlerstatuses [e: Das Bit wird jedoch bei Auftreten eines Fehlers an den Anfang der Hauptfunktion rückgesetzt. Deshalb können nach Ausführung von _decasc2bin andere Funktionen nicht direkt referenziert werden. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)
- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

■ Stringkonvertierung Hex-nach-Integer

Element	Beschreibung
Zusammen-fassung	Mit dieser Funktion wird ein Hexadezimal-String in Binärdaten konvertiert. Konvertieren Sie den ganzzahligen Hexadezimal-Text in Parameter 2 (Konvertierungsquell-Datenpuffer) in eine Ganzzahl, und speichern Sie diese in Parameter 1 (Konvertierungszieladresse.)
Format	_hexasc2bin ([Konvertierungszieladresse], [Konvertierungsquell-Datenpuffer])  Parameter 1: Interne Adresse, Temporäre Adresse Parameter 2: Datenpuffer

Beispielausdruck 1 (Bei einer Datenlänge von 16 Bits)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

"databuf0" hat folgenden Inhalt:

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Die o.g. Daten werden folgendermaßen konvertiert.

	16-Bit
LS0100	1234h

Beispielausdruck 2 (Bei einer Datenlänge von 32 Bits)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

"databuf0" hat folgenden Inhalt:

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	NULL

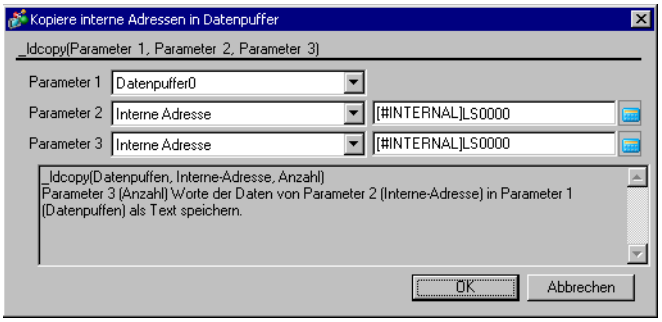
Die o.g. Daten werden folgendermaßen konvertiert.

	32 Bit
LS0100	12345678h
LS0102	

WICHTIG

- Wenn der konvertierte String länger als 16-Bit oder 32-Bit ist, tritt ein Fehler auf.
  z.B.: bei einer 16-Bit Skript-Länge:
  _strset (databuf0, "123456")
  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  Bei Ausführen der o.g. Ausdrucks wird Fehler Nr. 2 (String-Konvertierungsfehler) des String-Fehlerstatuses [e:
- Bei der Konvertierung eines Datenstrings, der andere Zeichen außer "0" bis "9", "A" bis "F" oder "a" bis "f"enthält, tritt ein Fehler auf.
  z.B.: bei einer 16-Bit Skript-Länge:
  _strset (databuf 0, "123G")
  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  Bei Ausführen der o.g. Ausdrucks wird Fehler Nr. 2 (String-Konvertierungsfehler) des String-Fehlerstatuses [e:
- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

■ Interne Adresse an Datenpuffer

Element	Beschreibung
Zusammen-fassung	Die im LS-Bereich gespeicherten String-Daten werden gemäß der Anzahl der Strings byteweise in den Datenpuffer übertragen. Speichern Sie Parameter 3 (Wortanzahl) der Daten aus Parameter 2 (Kopierquelladresse) als Text in Parameter 1 (Kopierziel-Datenpuffer.)
Format	_ldcopy (Kopierziel-Datenpuffer, [Kopierquelladresse], Worte)  Parameter 1: Datenpuffer Parameter 2: Interne Adresse Parameter 3: Ganzzahlwert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 1 bis 1024.)

Beispielausdruck 1:

_ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)

	16-Bit
LS0100	31h
LS0101	32h
LS0102	33h
LS0103	34h

Die Daten in LS0100 bis LS0103 werden sequentiell in die 4 Bytes des Datenpuffers geschrieben, beginnend mit "databuf0". Der LS-Bereich wird in jeden Byte gelesen (die untersten Bits.)

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

WICHTIG

- Der untere 1 Byte des LS-Bereichs wird ausgelesen und die angegebene Datenmenge in den Datenpuffer geschrieben.
- Der maximale Wert, der Parameter 3 zugewiesen werden kann, ist 1024. Beim Festlegen eines Wertes, der die Begrenzung überschreitet, tritt Fehler Nr. 1 (String-Überlauf) des String-Fehlerstatus auf [e: STR_ERR_STAT] wird ausgelöst.
- Selbst wenn sich Daten im oberen Byte der internen Adresse befinden, werden nur die Daten vom unteren Byte gelesen.
- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

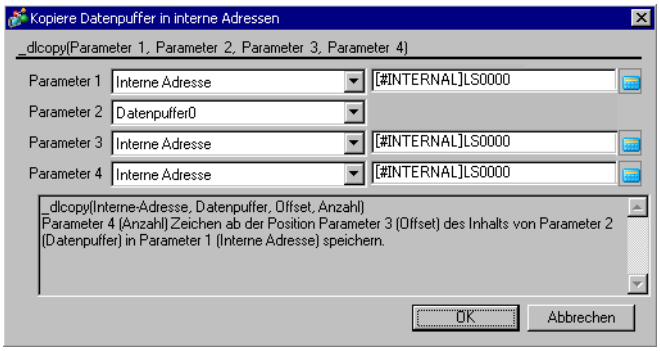
  _ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)

	16-Bit
LS0100	3132h
LS0101	3334h
LS0102	3536h
LS0103	3738h

Wenn Daten, wie oben aufgeführt, gespeichert werden, werden die Daten vom unteren Byte in den Datenpuffer gelesen und geschrieben.

	8 Bit	
databuf0[0]	32h	'2'
databuf0[1]	34h	'4'
databuf0[2]	36h	'6'
databuf0[3]	38h	'8'
databuf0[4]	00h	NULL

■ Datenpuffer in interne Adresse kopieren

Element	Beschreibung
Zusammen-fassung	Jeder im Offset des Datenpuffers gespeicherte Zeichenfolge-Datenbyte wird gemäß der Anzahl der Zeichenfolgen in den LS-Bereich kopiert. Speichert Parameter 4 (Anzahl der zu kopierenden) Datenzeichen von Parameter 3 (Kopierquell-Offset-Wert) des Inhalts von Parameter 2 (Kopierquell-Datenpuffer) in Parameter 1 (Kopierzieladresse).
Format	_dlcopy ([Kopierzieladresse], Kopierquell-Datenpuffer, Kopierquell-Offset-Wert, Anzahl der kopierten Zeichen)  Parameter 1: Interne Adresse Parameter 2: Datenpuffer Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 0 bis 1024.) Parameter 4: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 4 reicht von 1 bis 1024.)

Beispielausdruck 1:

_dlcopy ([w:[#INTERNAL]LS0100], databuf0, 2, 4)

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'

4 von "Offset 2" des "databuf0" wiedergewonnene Datenbytes werden in LS0100 bis LS0103 geschrieben. Die Daten werden in Einheiten von 1 Byte in den LS-Bereich geschrieben.

	16-Bit
LS0100	33h
LS0101	34h
LS0102	35h
LS0103	36h

WICHTIG

- 1 Datenbyte wird aus dem Datenpuffer gelesen und in den LS-Bereich geschrieben. Das heisst, nur die niedrigsten 8 Bits (1 Byte) des LS-Bereichs werden verwendet. Die signifikanten 8 Bits (1 Byte) werden mit "0" gelöscht.
- Wenn der festgelegte Wert [Offset-Quellwert + Anzahl der zu kopierenden Zeichen] die Datenpuffer-Größe überschreitet, Fehler Nr. 3 (String-Überlauffehler) des Stringstatuses [e: STR_ERR_STAT] wird ausgegeben.
- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

■ String-Operation Fehlerstatus

Falls während der Ausführung einer Text-Operation ein Fehler auftritt, wird ein Fehler in Text-Operation-Fehlerstatus geschrieben [e: STR_ERR_STAT]. "0" in [e: STR_ERR_STAT] gibt eine normale Bedingung an, wobei Werte außer "0" hier gespeichert werden [e: STR_ERR_STAT] zeigt Fehlerzustände an. Der letzte Fehler wird in Text-Operation-Fehlerstatus gespeichert [e: STR_ERR_STAT]. Text-Operation-Fehlerstatus kann im D-Skript-Werkzeugpalettenmenü mit [SIO Port-Operation/Steuerungsvariablen] eingestellt werden. Die Text-Operationsfehler sind in folgender Tabelle aufgelistet.

Fehlernummer	Fehlermeldung	Beschreibung
0	Normal	Kein Fehler
1	Text-Überlauf	Für die folgenden Funktionen ist ein String mit mindestens 256 Bytes direkt in das Argument einbezogen: _strset ( ), _strlen ( ), _strcat ( ), _strmid ( ) und IO_READ_WAIT ( ). Oder es wird während der Ausführung der Funktionen _strcat ( ) oder _ldcopy ( ) ein String erstellt, der die Größe des Datenpuffers übersteigt. Zum Beispiel: _strcat (databuf0, databuf1) Die obige Funktion wird ausgeführt, wenn ein String mit 1020 Bytes in databuf0 gespeichert wird und ein String mit 60 Bytes in databuf1. (Ein String mit über 1024 Bytes, der Größe des Datenpuffers, führt zu einem Fehlerstatus.)
2	String-Konvertierungsfehler	Das Zeichencode für die Funktionen _hexasc2bin ( ) oder _decasc2bin ( ) ist ungültig. Zum Beispiel: Ein Zeichencode außer "0" bis "9", "A" bis "F" oder "a" bis "f" ist im zweiten Argument von f _hexasc2bin ( ) vorhanden.
3	String-Abfragefehler	Versuchte Abfrage einer längeren Zeichenkette als jene, die mit der "_strmid ( )" Funktion spezifiziert wurde. Oder es wird ein höherer Offset Wert als der spezifizierte String bestimmt. Zum Beispiel: _strmid (databuf0, "12345678", 2, 8) Versuchte Abfrage einer Kette von 8 Zeichen von Offset 2.

Der String-Kontroll-Fehlerstatus kann mit D-Skripten und globalen D-Skripten nicht verwendet werden. Bei versehentlichem Auslesen wird "0" geladen.
Wird während der Ausführung jeder Funktion unter Fehlerstatus gespeichert.
Zur Überprüfung des Fehlers [e: STR_ERR_STAT] geben Sie folgende Anweisungen. Der Fehler kann mit folgendem Ausdruck bestätigt werden.

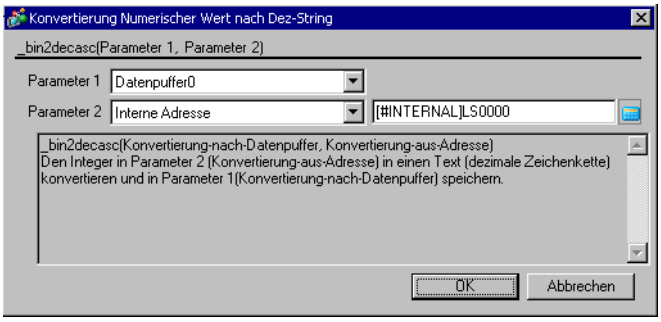
Beispielausdruck:

```
if ([e:STR_ERR_STAT] <> 0)              // Überprüft den Fehlerstatus
{
    set ([b:[#INTERNAL]LS005000])       // Legt Bit an der Fehleranzeigelampe fest
}
endif
```

WICHTIG

- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

■ Konvertierung numerischer Wert nach Dez-String

Element	Beschreibung
Zusammen-fassung	Mit dieser Funktion wird eine Ganzzahl in eine Dezimal-Zeichenfolge konvertiert. Konvertieren Sie die Ganzzahl in Parameter 2 (Konvertierungsquelladresse) in einen ganzzahligen Dezimal-Text, und speichern Sie diesen in Parameter 1 (Konvertierungsziel-Datenpuffer.)
Format	_bin2decasc(Konvertierung-nach-Datenpuffer, Konvertierung-aus-Adresse)  Parameter 1: Datenpuffer Parameter 2: Interne Adresse, Temporäre Adresse

Beispielausdruck 1 (Bei einer Datenlänge von 16 Bits)

_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])

	16-Bit
LS0100	1234

Die o.g. Daten werden folgendermaßen konvertiert. Beachten Sie, dass "NULL (0x00)" hinzugefügt wird.

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Beispielausdruck 2 (Bei einer Datenlänge von 32 Bits)

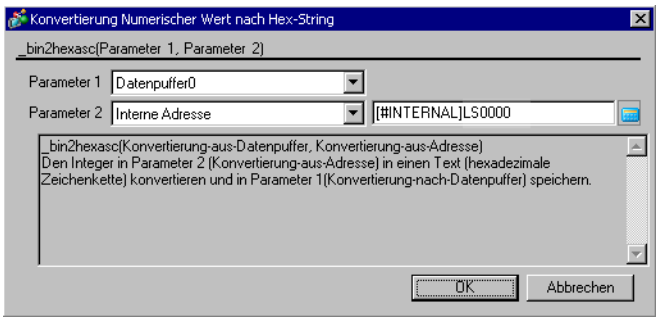
_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])

	32 Bit
LS0100	12345678
LS0102	

Die o.g. Daten werden folgendermaßen konvertiert.

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	NULL

■ Konvertierung numerischer Wert nach Hex-String

Element	Beschreibung
Zusammenfassung	Mit dieser Funktion werden Binärdaten in einen Hexadezimal-String konvertiert. Konvertieren Sie die Ganzzahl in Parameter 2 (Konvertierungsquelladresse) in einen ganzzahligen Hexadezimal-Text, und speichern Sie diesen in Parameter 1 (Konvertierungsziel-Datenpuffer.)
Format	_bin2hexasc (Konvertierungsziel-Datenpuffer, [Konvertierungsquelladresse])  Parameter 1: Datenpuffer Parameter 2: Interne Adresse, Temporäre Adresse

Beispielausdruck 1 (Bei einer Datenlänge von 16 Bits)

_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	16-Bit
LS0100	1234h

Die o.g. Daten werden folgendermaßen konvertiert. Beachten Sie, dass "NULL (0x00)" hinzugefügt wird.

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Beispielausdruck 2 (Bei einer Datenlänge von 32 Bits)

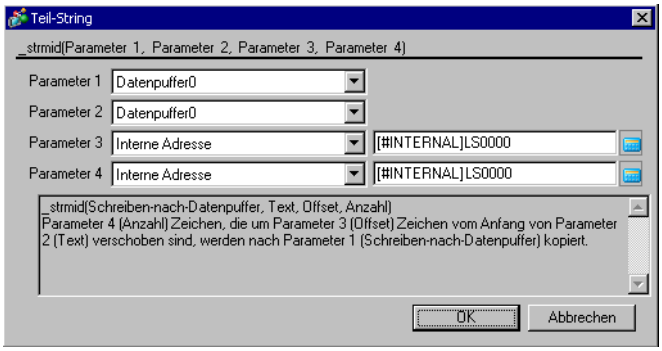
_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	32 Bit
LS0100	12345678h
LS0102	

Die o.g. Daten werden folgendermaßen konvertiert.

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	NULL

■ Teil-Text

Element	Beschreibung
Zusammen-fassung	Daten werden von dem angegebenen Offset des String gemäß der Stringlänge wiedergewonnen und in einem anderen Datenpuffer gespeichert. Parameter 4 (Textlänge) von Parameter 3 (Text-Offset) von Parameter 2 (Text) in Parameter 1 (Schreibziel-Datenpuffer) schreiben.
Format	_strmid (Schreibziel-Datenpuffer, Text, Text Offset, Textlänge)  Parameter 1: Datenpuffer Parameter 2: String, Datenpuffer Parameter 3: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 3 reicht von 0 bis 1.024.) Parameter 4: Numerischer Wert, Interne Adresse, Temporäre Adresse (der gültige Bereich für Parameter 4 reicht von 1 bis 1.024.)

Beispielausdruck:

_strmid (databuf0, "12345678", 2, 4)

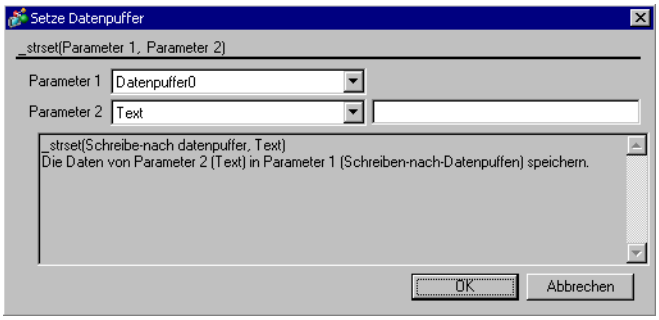
4 von Offset 2 des Strings "12345678" wiedergewonnene Datenbytes werden in "databuf0" gespeichert.

	8 Bit	
databuf0[0]	33h	'3'
databuf0[1]	34h	'4'
databuf0[2]	35h	'5'
databuf0[3]	36h	'6'
databuf0[4]	00h	NULL

WICHTIG

- Bei dem Versuch einen String, der länger ist, als der mit der Funktion "strmid ( )" spezifizierte oder beim Festlegen eines Offset Wertes, der höher ist, als der spezifizierte String, Fehler Nr. 3 (String-Extrahierungsfehler) des String-Fehlerstatuses [e: STR_ERR_STAT] wird ausgegeben.
- Die Ausführung wird bei Auftreten eines Fehlers abgebrochen und wieder an den Anfang der Hauptfunktion rückgesetzt. (Falls der Befehl während einer laufenden Funktion auftritt, wird zu der Zeile rückgesetzt, die jene Funktion aufgerufen hat.)

■ Datenpuffer setzen

Element	Beschreibung
Zusammen-fassung	Im Datenpuffer ist ein fester String gespeichert. Speichert die Daten von Parameter 2 (Text) in Parameter 1 (Schreibziel-Datenpuffer.)
Format	_strset(Schreibe-nach Datenpuffer, String)  Parameter 1: Datenpuffer Parameter 2: Text, Numerischer Wert (Textcode) (der gültige Bereich für Parameter 2 ist 0 und von 1 bis 255.)

Beispielausdruck:

_strset (databuf0, "ABCD")

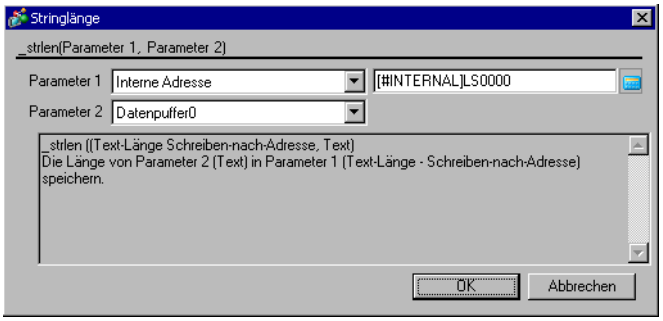
Der String wird wie unten veranschaulicht im Datenpuffer gespeichert:

	8 Bit	
databuf0[0]	41h	'A'
databuf0[1]	42h	'B'
databuf0[2]	43h	'C'
databuf0[3]	44h	'D'
databuf0[4]	00h	NULL

WICHTIG

- Es kann ein String mit bis zu 255 Zeichen spezifiziert werden. Zur Erstellung von über diesen Grenzwert hinausgehender Strings werden diese in einem anderen Puffer gespeichert und unter Anwendung der String-Verkettungsfunktion (_strcat) verkettet.
- Erstellen Sie zum Löschen des Datenpuffers einen leeren String.
  Beispiel: _strset (databuf0,"")
  _strset (databuf0,0)

■ Textlänge

Element	Beschreibung
Zusammen-fassung	Besorgt die Länge des gespeicherten Strings. Speichert die Länge von Parameter 2 (Text) in Parameter 1 (Textlänge der Schreibziel-Adresse.) (Das NULL Zeichen ist nicht mit eingeschlossen.)
Format	_strlen ((Text-Länge Schreiben-nach-Adresse, String)  Parameter 1: Interne Adresse, Temporäre Adresse Parameter 2: String, Datenpuffer

Beispielausdruck 1:

_strlen ([w:[#INTERNAL]LS0100], "ABCD")

Bei Ausführen der o.g. Anweisung wird die String-Länge wie unten dargestellt in LS0100 geschrieben.

LS0100	4

Beispielausdruck 2:

_strlen ([t:0000], databuf0)

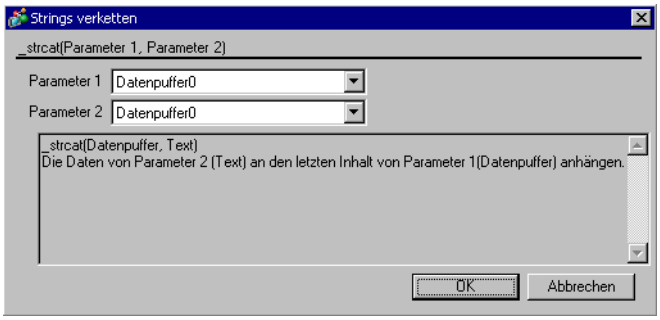
"databuf0" hat folgenden Inhalt:

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Bei Ausführen der o.g. Anweisung wird die String-Länge wie unten dargestellt in [t: 0000] geschrieben.

t0000	4

■ String-Verkettung

Element	Beschreibung
Zusammen-fassung	Eine Zeichenkette oder ein Zeichencode ist mit dem Textpuffer verkettet. Fügt die Daten von Parameter 2 (Text) zu den letzten des Inhalts von Parameter 1 (Datenpuffer kontaktieren.)
Format	_strcat(String-Datenpuffer, String)  Parameter 1: Datenpuffer Parameter 2: Parameter :Text, numerischer Wert (Textcode), Datenpuffer (der gültige Bereich für Parameter 2 ist 0 und von 1 bis 255.)

Beispielausdruck 1:

_strcat (databuf0, "ABCD")

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	NULL

Wenn "ABCD" wie oben verkettet wird, bewirkt dies folgendes Ergebnis. Beachten Sie, dass "NULL (0x00)" hinzugefügt wird.

	8 Bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	41h	'A'
databuf0[5]	42h	'B'
databuf0[6]	43h	'C'
databuf0[7]	44h	'D'
databuf0[8]	00h	NULL

WICHTIG

- Es kann ein String mit bis zu 255 Zeichen spezifiziert werden.
- Bei Festlegen eines leeren Strings für den numerischen Wert 0 bis Parameter 2 ändert sich der Datenpuffer des Parameter 1 'nicht. Beispiel:
  _strcat (databuf0,"")
  _strcat (databuf0,0)

21.11.12 Verfahrensbeispiel

■ Logische Operationsbeispiele

Es folgen logische Operationsbeispiele.

◆ **((100 > 99) und (200 <> 100))**
Ergebnis: EIN

◆ **((100 > 99) und (200 <> 200))**
Ergebnis: AUS

◆ **((100 > 99) oder (200 <> 200))**
Ergebnis: EIN

◆ **((100 < 99) oder (200 <> 200))**
Ergebnis: AUS

◆ **nicht (100 > 99)**
Ergebnis: AUS

◆ **nicht (100 < 99)**
Ergebnis: EIN

◆ **[w:[PLC1]D200] < 10**
Ergebnis: Ja, wenn D200 kleiner als 10 ist.

◆ **nicht [w:[PLC1]D200]**
Ergebnis: Ja, wenn D200 0 ist.

◆ **([w:[PLC1]D200] == 2) oder ([w:[PLC1]D200] == 5)**
Ergebnis: Ja, wenn D200 2 oder 5 ist.

◆ **([w:[PLC1]D200] < 5) und ([w:[PLC1]D300] < 8)**
Ergebnis: Ja, wenn D200 kleiner als 5 und D300 kleiner als 8 ist.

◆ **[w:[PLC1]D200] < 10**
Ergebnis: Ja, wenn D200 kleiner als 10 ist.

◆ **nicht [w:[PLC1]D200]**
Ergebnis: Ja, wenn D200 0 ist.

◆ **([w:[PLC1]D200] == 2) oder ([w:[PLC1]D200] == 5)**
Ergebnis: Ja, wenn D200 2 oder 5 ist.

◆ **([w:[PLC1]D200] < 5) und ([w:[PLC1]D300] < 8)**
Ergebnis: Ja, wenn D200 kleiner als 5 und D300 kleiner als 8 ist.

■ Bit-Operationsbeispiele

Es folgen Bit-Operationsbeispiele.

◆ [w:[PLC1]D200] << 4

Ergebnis: Die Daten in D200 werden um 4 Bit nach links verschoben.

◆ [w:[PLC1]D200] >> 4

Ergebnis: Die Daten in D200 werden um 4 Bit nach rechts verschoben.

◆ 12(0000Ch) wird im BIN-Format in D301 gespeichert.

[w:[PLC1]D200] = [w:[PLC1]D300] >> [w:[PLC1]D301]
Ergebnis: Die Daten in D300 werden um 12 Bit nach rechts verschoben und D200 zugewiesen.

◆ [w:[PLC1]D200] << 4

Ergebnis: Die Daten in D200 werden um 4 Bit nach links verschoben.

◆ [w:[PLC1]D200] >> 4

Ergebnis: Die Daten in D200 werden um 4 Bit nach rechts verschoben.

◆ 12(0000Ch) wird im BIN-Format in D310 gespeichert.

[w:[PLC1]D200] = [w:[PLC1]D300] >> [w:[PLC1]D301]
Ergebnis: Die Daten in D300 werden um 12 Bit nach rechts verschoben und D200 zugewiesen.

◆ Bitweise UND

0 & 0	Ergebnis: 0
0 & 1	Ergebnis: 0
1 & 1	Ergebnis: 1
0x1234 & 0xF0F0	Ergebnis: 0x1030

◆ Bitweise ODER

0 | 0	Ergebnis: 0
0 | 1	Ergebnis: 1
1 | 1	Ergebnis: 1
0x1234 | 0x9999	Ergebnis: 0x9BBD

◆ Bitwise XOR

0 ^ 0	Ergebnis: 0
0 ^ 1	Ergebnis: 1
1 ^ 1	Ergebnis: 0

◆ Komplement von Bit 1 (bei einem Datenformat von Bin16 +)

~ 0	Ergebnis: 0xFFFF
~ 1	Ergebnis: 0xFFFE

■ Bedingte Zweigauslastung - Berechnungsbeispiele

Steuerprogrammfluss unter Verwendung von "if-endif" und "if-else-endif"

◆ if - endif

```
if (Bedingung)
{Prozess 1}
endif
```

Falls die Bedingung wahr ist, wird Prozess 1 ausgeführt. Falls Bedingung nicht war, wird Prozess 1 übersprungen.

Zum Beispiel:

```
if ( [ w:[PLC1]D200 ] < 5 )
{
    [ w:[PLC1]D100 ] = 1
}
endif
```

Falls die Daten in D200 kleiner als 5 sind, wird 1 in D100 zugewiesen.

◆ if - else - endif

```
if (Bedingung)
{Prozess 1}
else
{Prozess 2}
endif
```

Wenn die Bedingung wahr ist, wird Prozess 1 ausgeführt. Wenn unwahr, Prozess 2 wird ausgeführt.

Zum Beispiel:

```
if ( [ w:[PLC1]D200 ] < 5 )
{
    [ w:[PLC1]D100 ] = 1
}
else
{
    [ w:[PLC1]D100 ] = 0
}
endif
```

Wenn der Wert in D200 kleiner als 5 ist, wird 1 in D100 zugewiesen. Andernfalls 0.

■ Berechnungsbeispiele zur Wortadresse mit Offsetanwendung

Offset-Spezifikation: Besondere Berechnungsbeispiele mit [w:D00100]#[t:0000].

◆ Skript-E/A: 16 Bit ohne Vorzeichen, [t:0000]= 65526, die entsprechende Adresse lautet [w:D00090].

100 + 65526 = 64(Hex) + FFF6(Hex) = 1005A(Hex) --> 005A(Hex) = 90

Die unteren 16 Bits sind gültig

◆ Skript-E/A: 16 Bit mit Vorzeichen, [t:0000]= -10, die entsprechende Adresse lautet [w:D00090].

100 + (-10) = 64(Hex) + FFF6(Hex) = 1005A(Hex) --> 005A(Hex) = 90

Die unteren 16 Bits sind gültig

◆ Skript-E/A: 32 Bit ohne Vorzeichen, [t:0000]= 4294901840, die entsprechende Adresse lautet [w:D00180].

100 + 4294901840 = 64(Hex) + FFFF0050(Hex) = FFFF 00B4(Hex) --> 00B4(Hex) = 180

Die unteren 16 Bits sind gültig

◆ Skript-E/A: 32 Bit mit Vorzeichen, [t:0000]= -65456, die entsprechende Adresse lautet [w:D00180].

100 + (-65456) = 64(Hex) + FFFF0050(Hex) = FFFF 00B4(Hex) --> 00B4(Hex) = 180

Die unteren 16 Bits sind gültig

WICHTIG

- Wortadressen mit Offset werden grundsätzlich als 16-Bit Bin-Werte behandelt, unabhängig von der Bit-Länge und den Datenanzeige-Einstellungen des Skripts. Falls das Ergebnis der Operation 16-Bit überschreitet (max. Wert: 65.535), werden Bits 0 bis 15 als gültige Bits behandelt und Bits 16 und höher verworfen.