Logische Bedingungen

Dieses Kapitel beschreibt den Zweck und die Benutzung logischer Bedingungen, die in den folgenden Statements benutzt werden können: `FIND`, `READ`, `HISTOGRAM`, `ACCEPT`/`REJECT`, `IF`, `DECIDE FOR`, `REPEAT`

Folgende Themen werden behandelt:

- Einleitung
- Relationaler Ausdruck
- Erweiterter Relationaler Ausdruck
- Auswertung einer logischen Variablen
- Felder in logischen Bedingungen
- Logische Operatoren in komplexen logischen Ausdrücken
- BREAK-Option - Aktuellen Wert mit Wert des vorangegangenen Schleifendurchlaufs vergleichen
- IS-Option - Prüfen ob Inhalt von Alphanumerischem oder Unicode-Feld konvertiert werden kann
- MASK-Option - Ausgewählte Stellen eines Feldes auf bestimmten Inhalt prüfen
- MASK-Option im Vergleich zur IS Option
- MODIFIED-Option - Prüfen ob Feldinhalt verändert worden ist
- SCAN-Option - Nach einem bestimmten Wert in einem Feld suchen
- SPECIFIED-Option - Prüfen ob ein Wert für einen optionalen Parameter übergeben wird

---

Einleitung

Die Grundform einer logischen Bedingung ist ein relationaler (vergleichender) Ausdruck. Mit den logischen Operatoren `AND` und `OR` können mehrere relationale Ausdrücke (`AND`, `OR`) zu komplexen logischen Bedingungen verknüpft werden.

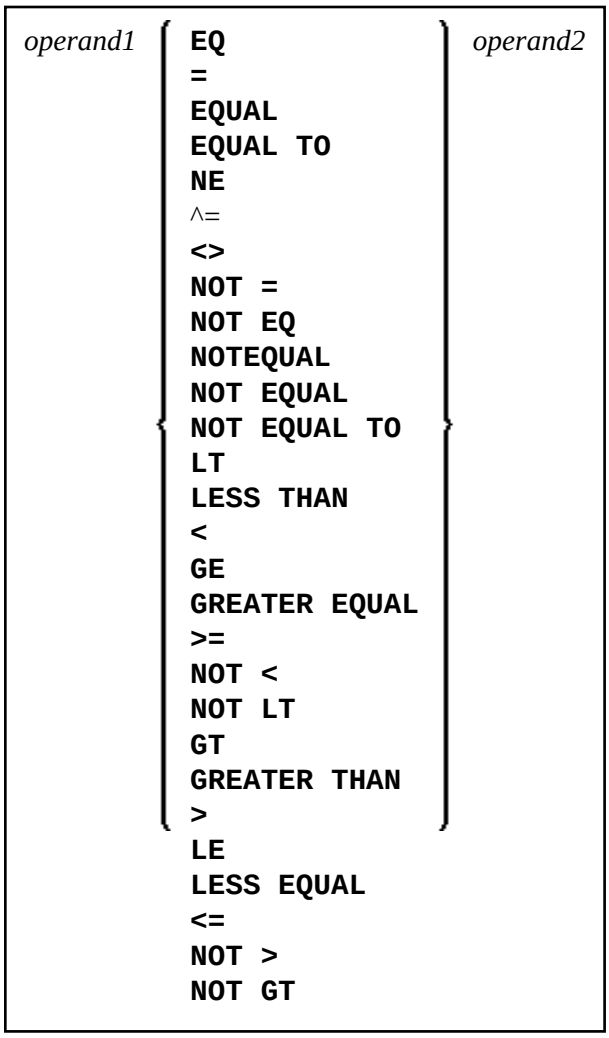
Arithmetische Ausdrücke können ebenfalls in logischen Bedingungen verwendet werden.

Logische Bedingungen können in den folgenden Statements angegeben werden:

Statement	Bedingungen
FIND	Zusätzlich zu dem primären Selektionskriterium, das in der WITH-Klausel angegeben wird, kann in einer WHERE-Klausel als zusätzliches Selektionskriterium eine logische Bedingung angegeben werden. Die in der WHERE-Klausel angegebene Bedingung wird erst ausgewertet, nachdem ein Datensatz aufgrund des WITH-Kriteriums ausgewählt und gelesen worden ist. In der WITH-Klausel werden "Basic Search Criteria" (Suchkriterien) angegeben (vgl. FIND-Statement), aber keine logische Bedingung.
READ	In einer WHERE-Klausel kann eine logische Bedingung angegeben werden, die darüber entscheidet, ob ein gerade gelesener Datensatz weiterverarbeitet wird oder nicht. Diese Bedingung wird erst ausgewertet, *nachdem* ein Datensatz gelesen wurde.
HISTOGRAM	In einer WHERE-Klausel kann eine logische Bedingung angegeben werden, die darüber entscheidet, ob ein gerade gelesener Datensatz weiterverarbeitet wird oder nicht. Diese Bedingung wird erst ausgewertet, *nachdem* ein Datensatz gelesen wurde.
ACCEPT/REJECT	Zusätzlich zu den Selektionskriterien, aufgrund derer ein Datensatz mit einem FIND-, READ- oder HISTOGRAM-Statement gelesen wurde, kann in der IF-Klausel eines ACCEPT- oder REJECT-Statements eine weitere logische Bedingung angegeben werden, die über die weitere Verarbeitung eines Datensatzes entscheidet. Diese Bedingung wird erst ausgewertet, nachdem ein Datensatz gelesen wurde und die Verarbeitung des Datensatzes begonnen hat.
IF	Die Ausführung des Statements kann von der Erfüllung einer logischen Bedingung abhängig gemacht werden.
DECIDE FOR	Die Ausführung des Statements kann von der Erfüllung einer logischen Bedingung abhängig gemacht werden.
REPEAT	In der UNTIL- oder WHILE-Klausel eines REPEAT-Statements kann eine logische Bedingung angegeben werden, die darüber entscheidet, wann eine Verarbeitungsschleife beendet werden soll.

Relationaler Ausdruck

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N	E	A	U	N	P	I	F	B	D	T	L		G	O	ja	ja
operand2	C	S	A		N	E	A	U	N	P	I	F	B	D	T	L		G	O	ja	nein

Die obige Operandentabelle ist in der *Statements*-Dokumentation unter *Syntax-Symbole und Operandentabellen* erklärt.

In der Spalte "Mögliche Struktur" der Tabelle steht "E" für arithmetischer Ausdruck, d.h. innerhalb eines relationalen Ausdrucks kann ein beliebiger arithmetischer Ausdruck als Operand verwendet werden. Weitere Informationen siehe *arithmetic-expression* in der Beschreibung des `COMPUTE`-Statements.

Erklärung der Vergleichsoperatoren:

Vergleichsoperator	Erklärung
EQ = EQUAL EQUAL TO	gleich
NE ^= <> NOT = NOT EQ NOTEQUAL NOT EQUAL NOT EQUAL TO	ungleich
LT LESS THAN <	kleiner als
GE GREATER EQUAL >=	größer als oder gleich
NOT < NOT LT	nicht größer als
GT GREATER THAN >	größer als
LE LESS EQUAL <=	kleiner als oder gleich less
NOT > NOT GT	nicht größer als

Beispiele für relationale Ausdrücke:

```
IF NAME = 'SMITH'
IF LEAVE-DUE GT 40
IF NAME = #NAME
```

Weitere Informationen über den Vergleich von Arrays in einem relationalen Ausdruck siehe *Verarbeitung von Arrays*.

Anmerkung:
Wird ein Gleitkomma-Operand verwendet, so erfolgt der Vergleich in Gleitkommaform. Da Gleitkomma-Zahlen per se nur eine begrenzte Genauigkeit haben, lassen sich Rundungs- bzw. Abschneidefehler bei der Konvertierung von Zahlen in/aus Gleitkommaform nicht ausschließen.

Arithmetische Ausdrücke in logischen Bedingungen

Das folgende Beispiel zeigt, wie arithmetische Ausdrücke in logischen Bedingungen eingesetzt werden können:

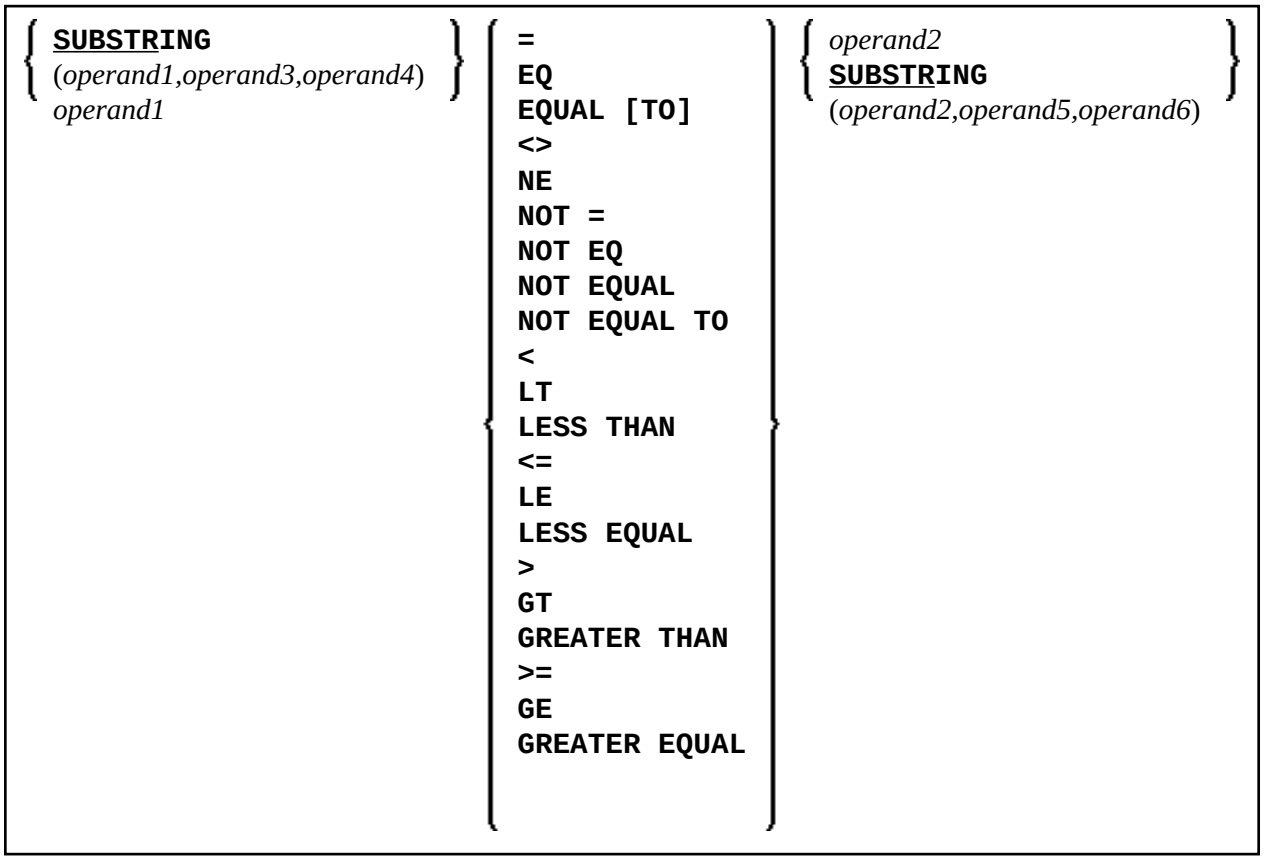
```
IF #A + 3 GT #B - 5 AND #C * 3 LE #A + #B
```

Handles in logischen Bedingungen

Wenn die Operanden in einem relationalen Ausdruck Handles sind, dürfen nur EQUAL- und NOT EQUAL-Operatoren verwendet werden.

SUBSTRING-Option in relationalem Ausdruck

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N		A	U					B						ja	ja
operand2	C	S	A		N		A	U					B						ja	nein
operand3	C	S							N	P	I		B						ja	nein
operand4	C	S							N	P	I								ja	nein
operand5	C	S							N	P	I								ja	nein
operand6	C	S							N	P	I								ja	nein

Mit der SUBSTRING-Option können Sie einen *Teil* eines alphanumerischen, binären oder Unicode-Feldes vergleichen. Nach dem Feldnamen (*operand1*) geben Sie zunächst die erste Stelle (*operand3*) und dann die Länge (*operand4*) des zu vergleichenden Feldteils an.

Sie können auch einen Feldwert mit einem Teil eines anderen Feldwertes vergleichen. Nach dem Feldnamen (*operand2*) geben Sie zunächst die erste Stelle (*operand5*) und dann die Länge (*operand6*) des Feldteils an, mit dem *operand1* verglichen werden soll.

Sie können auch beide Formen miteinander kombinieren, d.h. SUBSTRING gleichzeitig für *operand1* und für *operand2* angeben.

Beispiele:

Dieser Ausdruck vergleicht die 5. bis einschließlich 12. Stelle des Wertes in Feld #A mit dem Wert von Feld #B:

```
SUBSTRING(#A,5,8) = #B
```

wobei 5 die erste Stelle und 8 die Länge ist.

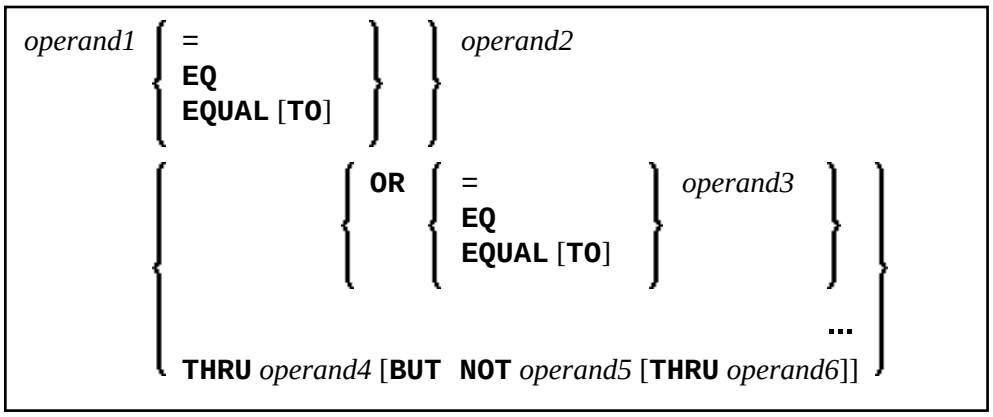
Dieser Ausdruck vergleicht den Wert von Feld #A mit der 3. bis einschließlich 6. Stelle des Wertes in Feld #B:

```
#A = SUBSTRING(#B,3,4)
```

Anmerkung:
Wenn Sie *operand3*/*operand5* weglassen, wird ab Anfang des Feldes verglichen. Wenn Sie *operand4*/*operand6* weglassen, wird ab der angegebenen Stelle (*operand3*/*operand5*) bis zum Ende des Feldes verglichen.

Erweiterter Relationaler Ausdruck

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein
operand2	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein
operand3	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein
operand4	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein
operand5	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein
operand6	C	S	A		N*	E	A	U	N	P	I	F	B	D	T			G	O	ja	nein

* Mathematische Funktionen und Systemvariablen sind erlaubt. Gruppenwechsel-Funktionen sind nicht erlaubt.

Operand3 kann auch unter Verwendung einer MASK- oder SCAN-Option angegeben werden; d.h. er kann angegeben werden als:

MASK (*mask-definition*) [*operand*]

MASK *operand*

SCAN *operand*

Einzelheiten zu diesen Optionen finden Sie unter *MASK-Option* und *SCAN-Option*.

Beispiele:

```
IF #A = 2 OR = 4 OR = 7
IF #A = 5 THRU 11 BUT NOT 7 THRU 8
```

Auswertung einer logischen Variablen

Syntax:

operand1

Diese Option kann in Verbindung mit einer logischen Variablen (Format L) eingesetzt werden. Eine logische Variable kann die Werte TRUE (wahr) oder FALSE (falsch) haben. Mit *operand1* geben Sie den Namen der Variablen an.

Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A												L				nein	nein

Beispiel für eine logische Variable:

```
** Example 'LOGICX05': Logical variable in logical condition
************************************************************************
DEFINE DATA LOCAL
1 #SWITCH (L)  INIT <true>
1 #INDEX  (I1)
END-DEFINE
*
FOR #INDEX 1 5
  WRITE NOTITLE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX =' #INDEX
  WRITE NOTITLE #SWITCH (EM=OFF/ON) 7X 'INDEX =' #INDEX
  IF #SWITCH
    MOVE FALSE TO #SWITCH
  ELSE
    MOVE TRUE TO #SWITCH
  END-IF
  /*
  SKIP 1
END-FOR
END
```

Ausgabe des Programms LOGICX05:

```
TRUE     INDEX =    1
ON       INDEX =    1

FALSE    INDEX =    2
OFF      INDEX =    2

TRUE     INDEX =    3
ON       INDEX =    3

FALSE    INDEX =    4
OFF      INDEX =    4

TRUE     INDEX =    5
ON       INDEX =    5
```

Felder in logischen Bedingungen

Bei der Konstruktion logischer Bedingungen dürfen sowohl Datenbankfelder als auch Benutzervariablen verwendet werden. Datenbankfelder, die Teil einer Periodengruppe oder multiple Felder sind, dürfen ebenfalls verwendet werden. Wenn ein Bereich von Werten für multiple Felder oder ein Bereich von Ausprägungen für Periodengruppen angegeben wird, dann ist die Bedingung erfüllt, wenn der Suchwert in einem Wert bzw. einer Ausprägung innerhalb des angegebenen Bereichs gefunden wird.

Jeder verwendete Wert muss mit dem ihm in einem relationalen Ausdruck gegenüberstehenden Feld kompatibel sein. Dezimalstellen können nur bei Werten für numerische Felder angegeben werden, wobei die Anzahl der Dezimalstellen von Wert und Feld kompatibel sein muss.

Haben zwei Operanden unterschiedliches Format, wird das Format des zweiten Operanden in das des ersten umgesetzt.

Anmerkung:
Eine numerische Konstante ohne Dezimalzeichen-Notation wird im Wertebereich `-2147483648` bis `+2147483647` im Format I gespeichert, siehe *Numerische Konstante*. Deshalb wird beim Vergleich mit einer solchen ganzzahligen Konstante der *operand2* in einen ganzzahligen Wert umgesetzt. Das hat zur Folge, dass die Nachkommastellen beim *operand2* abgeschnitten und nicht berücksichtigt werden.

Beispiel:

```
IF 0 = 0.5      /* is true because 0.5 (operand2) is converted to 0 (format I of operand1)
IF 0.0 = 0.5  /* is false
IF 0.5 = 0      /* is false
IF 0.5 = 0.0  /* is false
```

Die folgende Tabelle zeigt, welche Operandenformate zusammen in einer logischen Bedingung verwendet werden können:

operand1	***operand2***												
	A	**U**	**B*n* (*n*=<4)**	**B*n* (*n*>=5)**	**D**	**T**	**I**	**F**	**L**	**N**	**P**	**GH**	**OH**
A	Y	Y	Y	Y									
U	Y	Y	[2]	[2]									
B*n* (*n*=<4)	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y		
B*n* (*n*>=5)	Y	Y	Y	Y									
D			Y		Y	Y	Y	Y		Y	Y		
T			Y		Y	Y	Y	Y		Y	Y		
I			Y		Y	Y	Y	Y		Y	Y		
F			Y		Y	Y	Y	Y		Y	Y		
L													
N			Y		Y	Y	Y	Y		Y	Y		
P			Y		Y	Y	Y	Y		Y	Y		
GH [1]												Y	
OH [1]													Y

Legende:

Y = ja

[1] GH = GUI Handle, OH = Object Handle.

[2] Es wird davon ausgegangen, dass der Binärwert Unicode-Codepunkte enthält, und der Vergleich wird wie für einen Vergleich zweier Unicode-Werte durchgeführt. Die Länge des binären Feldes muss geradzahlig sein.

Wird ein Array mit einem Skalarwert in Relation gesetzt, so wird jedes Element des Arrays mit dem Skalarwert verglichen; die Bedingung ist erfüllt, wenn mindestens ein Array-Element die Bedingung erfüllt (ODER-Verknüpfung).

Wird ein Array mit einem Array in Relation gesetzt, so wird jedes Element des einen Arrays mit dem entsprechenden Element des anderen Arrays verglichen; die Bedingung ist nur dann erfüllt, wenn alle Element-Vergleiche die Bedingung erfüllen (UND-Verknüpfung).

Siehe auch *Verarbeitung von Arrays*.

Anmerkung:
Phonetische Deskriptoren (Adabas) dürfen in einer logischen Bedingung nicht verwendet werden

Beispiele für logische Bedingungen:

```
FIND EMPLOYEES-VIEW WITH CITY = 'BOSTON' WHERE SEX = 'M'
READ EMPLOYEES-VIEW BY NAME WHERE SEX = 'M'
ACCEPT IF LEAVE-DUE GT 45
IF #A GT #B THEN COMPUTE #C = #A + #B
REPEAT UNTIL #X = 500
```

Logische Operatoren in komplexen logischen Ausdrücken

Mittels der Boole'schen Operatoren AND, OR und NOT ist es möglich, logische Bedingungen miteinander zu verknüpfen. Mit Hilfe von Klammern können logische Bedingungen logisch zusammengefasst werden.

Die Operatoren werden in der folgenden Reihenfolge ausgewertet:

Priorität	Logische Verknüpfung	Bedeutung
1	( )	Klammer-Rechnung
2	NOT	Negation
3	AND	UND-Verknüpfung
4	OR	ODER-Verknüpfung

Die folgenden logischen Bedingungen können miteinander verknüpft werden, um einen komplexen logischen Ausdruck zu bilden:

- Relationale Ausdrücke
- Erweiterte relationale Ausdrücke
- MASK-Option

- SCAN-Option
- BREAK-Option

Die Syntax für eine logische Bedingung (*logical-expression*) ist wie folgt:

```
[NOT] { logical-condition-criterion } [ { OR  } logical-expression ] ...
      { (logical-expression)       }   { AND }
```

Beispiele für *logical-expressions*:

```
FIND STAFF-VIEW WITH CITY = 'TOKYO'
     WHERE BIRTH GT 19610101 AND SEX = 'F'
IF NOT (#CITY = 'A' THRU 'E')
```

Informationen über den Vergleich von Arrays in einem logischen Ausdruck finden Sie in *Verarbeitung von Arrays*.

Anmerkung:
Wenn mehrere *logical-condition-criteria* mit AND verknüpft werden, wird die Auswertung beendet, sobald das erste dieser Kriterien gefunden wird, das nicht erfüllt ist.

BREAK-Option - Aktuellen Wert mit Wert des vorangegangenen Schleifendurchlaufs vergleichen

Mit der BREAK-Option kann der aktuelle Wert eines Feldes (oder eines Teils eines Feldes) mit dem Wert verglichen werden, den das Feld im vorangegangenen Durchlauf durch die Verarbeitungsschleife hatte.

Syntax:

```
BREAK [OF] operand1 [/n/]
```

Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1		S					A	U	N	P	I	F	B	D	T	L				ja	nein

Syntax-Elementbeschreibung:

operand1	Mit *operand1* geben Sie das Feld an, das überprüft werden soll. Eine bestimmte Ausprägung eines Arrays kann auch als ein Kontrollfeld benutzt werden.
/*n*/	Soll nur ein Teil des Feldes überprüft werden, so geben Sie eine mit Schrägstrichen eingegrenzte Zahl *n* an, die angibt, wieviele Stellen (von links nach rechts gezählt) des Feldes auf einen Wertwechsel überprüft werden sollen. Die Notation */n/* kann nur bei Operanden des Formats A, B, N oder P benutzt werden. Eine BREAK-Bedingung ist erfüllt, wenn der Wert des Kontrollfeldes (bzw. der angegebenen Stellen des Feldes) sich ändert. Eine BREAK-Bedingung ist nicht erfüllt, wenn eine AT END OF DATA-Bedingung auftritt. Beispiel: In diesem Beispiel wird überprüft, ob der Wert der ersten Stelle des Feldes FIRST-NAME sich geändert hat. `BREAK FIRST-NAME /1/` Der Einsatz von Natural-Systemfunktionen (die bei einem AT BREAK-Statement zur Verfügung stehen) ist bei der BREAK-Option nicht erlaubt.

Beispiel für BREAK-Option:

```
** Example 'LOGICX03': BREAK option in logical condition
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 BIRTH
*
1 #BIRTH (A8)
END-DEFINE
*
LIMIT 10
READ EMPLOY-VIEW BY BIRTH
  MOVE EDITED BIRTH (EM=YYYYMMDD) TO #BIRTH
  /*
  IF BREAK OF #BIRTH /6/
    NEWPAGE IF LESS THAN 5 LINES LEFT
    WRITE / '-' (50) /
  END-IF
  /*
  DISPLAY NOTITLE BIRTH (EM=YYYY-MM-DD) NAME FIRST-NAME
END-READ
END
```

Ausgabe des Programms LOGICX03:

```
  DATE            NAME              FIRST-NAME
   OF
 BIRTH
---------- -------------------- --------------------

1940-01-01 GARRET               WILLIAM
1940-01-09 TAILOR               ROBERT
1940-01-09 PIETSCH              VENUS
1940-01-31 LYTTLETON            BETTY
```

```
------------------------------------------------

1940-02-02 WINTRICH             MARIA
1940-02-13 KUNEY                MARY
1940-02-14 KOLENCE              MARSHA
1940-02-24 DILWORTH             TOM

------------------------------------------------

1940-03-03 DEKKER               SYLVIA
1940-03-06 STEFFERUD            BILL
```

IS-Option - Prüfen ob Inhalt von Alphanumerischem oder Unicode-Feld konvertiert werden kann

Syntax:

operand1 IS (*format*)

Mit dieser Option können Sie prüfen, ob der Inhalt eines alphanumerischen oder Unicode-Feldes (*operand1*) in ein bestimmtes anderes Format übertragbar ist.

Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N		A	U											ja	nein

Dieses *Format* kann sein:

N*ll.ll*	Numerisch mit Länge *ll.ll*.
F*ll*	Gleitkomma (floating point) mit Länge *ll*.
D	Datum. Folgende Datumsformate sind möglich: *dd-mm-yy*, *dd-mm-yyyy*, *ddmmyyyy* (*dd* = Tag, *mm* = Monat, *yy* oder *yyyy* = Jahr). Die Abfolge der Tages-, Monats- und Jahreskomponenten sowie die Trennzeichen zwischen den Komponenten werden durch den Profilparameter DTFORM (der in der *Parameter-Referenz*-Dokumentation beschrieben ist) bestimmt.
T	Zeit (Time); entsprechend dem Standard-Zeitanzeigeformat.
P*ll.ll*	Gepackt numerisch mit Länge *ll.ll*.
I*ll*	Ganzzahl (integer) mit Länge *ll*.

Bei der Prüfung werden für den Wert in *operand1* vor- oder nachgestellte Leerzeichen ignoriert.

Die Prüfung mit der IS-Option ist sinnvoll, wenn beispielsweise vor Ausführung der mathematischen Funktion VAL (Erhalt des numerischen Wertes eines alphanumerischen Feldes) das Format des Wertes überprüft wird, um zu vermeiden, dass ein falsches Format einen Laufzeitfehler verursacht.

Anmerkung:
Mit der IS-Option kann nicht geprüft werden, ob der Wert eines alphanumerischen Feldes in dem angegebenen Format ist, sondern ob er in das Format *übertragbar* ist. Um zu prüfen, ob ein Wert in einem bestimmten Format ist, können Sie die *MASK-Option* verwenden.

Beispiel für IS-Option:

```
** Example 'LOGICX04': IS option as format/length check
************************************************************************
DEFINE DATA LOCAL
1 #FIELDA (A10)          /* INPUT FIELD TO BE CHECKED
1 #FIELDB (N5)           /* RECEIVING FIELD OF VAL FUNCTION
1 #DATE   (A10)          /* INPUT FIELD FOR DATE
END-DEFINE
*
INPUT #DATE #FIELDA
IF #DATE IS(D)
  IF #FIELDA IS (N5)
    COMPUTE #FIELDB = VAL(#FIELDA)
    WRITE NOTITLE 'VAL FUNCTION OK' //  '=' #FIELDA '=' #FIELDB
  ELSE
    REINPUT 'FIELD DOES NOT FIT INTO N5 FORMAT'
            MARK *#FIELDA
  END-IF
ELSE
  REINPUT 'INPUT IS NOT IN DATE FORMAT (YY-MM-DD) '
          MARK *#DATE
END-IF
*
END
```

Ausgabe des Programms LOGICX04:

```
#DATE 150487    #FIELDA
```

```
INPUT IS NOT IN DATE FORMAT (YY-MM-DD)
```

Weitere Informationen siehe *MASK-Option im Vergleich zur IS-Option*.

MASK-Option - Ausgewählte Stellen eines Feldes auf bestimmten Inhalt prüfen

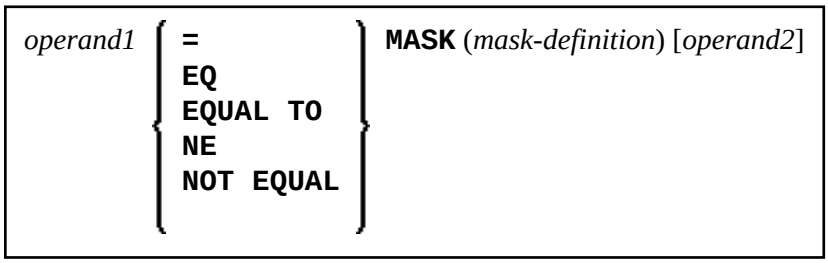
Mit der MASK-Option können Sie bestimmte ausgewählte Stellen eines Feldes nach einem bestimmten Wert absuchen.

Folgende Themen werden behandelt:

- Konstante Maske
- Variable Maske
- Zeichen in einer Maske
- Maskenlänge
- Datumsprüfungen
- Prüfung unter Verwendung von Konstanten oder Variablen
- Bereichsprüfungen
- Auf gepackte oder ungepackte numerische Daten abprüfen

Konstante Maske

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N		A	U	N	P									ja	nein
operand2	C	S					A	U	N	P			B						ja	nein

Operand2 kann nur angegeben werden, wenn die *mask-definition* mindestens ein X enthält. *Operand1* und *operand2* müssen formatkompatibel sein:

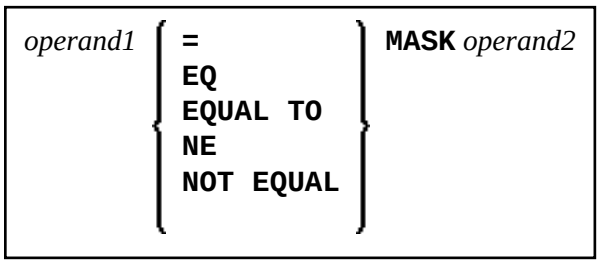
- wenn *operand1* Format A hat, muss *operand2* Format A, B, N oder U haben
- wenn *operand1* Format U hat, muss *operand2* Format A, B, N oder U haben
- wenn *operand1* Format N oder P hat, muss *operand2* Format N oder P haben.

Wird ein X in der *mask-definition* angegeben, werden die betreffenden inhaltlichen Stellen von *operand1* und *operand2* zum Wertevergleich ausgewählt.

Variable Maske

Anstatt einer konstanten *mask-definition* (siehe oben) können Sie auch eine variable MASK-Definition angeben:

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate											Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N		A	U	N	P								yes	no
operand2		S					A	U										yes	no

Der Inhalt von *operand2* wird dann als MASK-Definition genommen. Nachgestellte Leerzeichen in *operand2* werden ignoriert.

- Wenn *operand1* Format A, N oder P hat, muss *operand2* Format A haben.
- Wenn *operand1* Format U hat, muss *operand2* Format A haben.

Zeichen in einer Maske

In einer *mask-definition* können Sie folgende Zeichen verwenden (die Masken-Definition ist bei einer konstanten Maske in der *mask-definition* und bei einer variablen Maske in *operand2* enthalten):

Zeichen	Bedeutung
. oder ? oder _	Ein Punkt (.) oder Fragezeichen (?) oder ein Unterstrich (_) markiert eine einzelne Stelle, die nicht überprüft werden soll.
* oder %	Eine beliebige Anzahl von Stellen, die nicht überprüft werden sollen.
/	(Schrägstrich) Prüft, ob der Wert mit einem (oder mehreren) bestimmten Zeichen endet. Beispiel: Die folgende Bedingung ist wahr, wenn entweder an der letzten Stelle des Feldes ein E steht oder nach dem E nur noch Leerzeichen stehen: `IF #FIELD = MASK (*'E'/)`
A	Eine Stelle, die nach Groß- oder Kleinbuchstaben abgesucht werden soll.

Zeichen	Bedeutung
'c'	Eine oder mehrere Stellen, die nach den in Apostrophen (') stehenden Zeichen abgesucht werden sollen (doppelte Apostrophe bedeuten, dass innerhalb der Zeichenkette nach einem Apostroph gesucht wird). Wenn *operand1* Unicode-Format hat, muss *'c'* Unicode-Zeichen enthalten.
C	Eine Stelle, die nach alphanumerischem Inhalt (Groß- oder Kleinbuchstabe, Zahl) oder Leerzeichen abgesucht werden soll.
DD	Zwei Stellen, die nach einem gültigen Tagesdatum abgesucht werden sollen (01 - 31; abhängig von den Werten für MM und YY/YYYY, falls angegeben; siehe auch *Datumsprüfungen*).
H	Eine Stelle, die nach einem Hexadezimalzeichen (A - F, 0 - 9) abgesucht werden soll.
JJJ	Die Stellen sollen nach einem gültigen Julianischen Tag abgesucht werden, d.h. die Tageszahl im Jahr: 001-366, abhängig vom Wert von YY/YYYY, wenn angegeben; siehe auch *Datumsprüfungen*.
L	Eine Stelle, die nach Kleinbuchstaben (a - z) abgesucht werden soll.
MM	Zwei Stellen, die nach einer gültigen Monatsangabe (01 – 12) abgesucht werden sollen; siehe auch *Datumsprüfungen*.
N	Eine Stelle, die nach einer Ziffer abgesucht werden soll.
n...	Eine oder mehrere Stellen, die nach einem numerischen Wert im Bereich von 0 bis *n* abgesucht werden sollen.
*n*1-*n*2 oder *n*1:*n*2	Stellen, die nach einem numerischen Wert im Bereich von *n*1-*n*2 abgesucht werden sollen. *n*1 und *n*2 müssen gleich lang sein.
P	Eine Stelle, die nach einem druckbaren Zeichen (U, L, N oder S – Buchstabe, Zahl oder Sonderzeichen) abgesucht werden soll.
S	Eine Stelle, die nach Sonderzeichen abgesucht werden soll. Siehe auch *Support of Different Character Sets with NATCONV.INI* in der *Operations*-Dokumentation.
U	Eine Stelle, die nach Großbuchstaben (A – Z) abgesucht werden soll.
X	Eine Stelle, die mit der entsprechenden Stelle des auf die *mask-definition* folgenden Wertes (*operand2*) verglichen werden soll. In einer variablen Maske ist X nicht erlaubt, da sinnlos.
YY	Zwei Stellen, die nach einer gültigen Jahreszahl (00 - 99) abgesucht werden sollen; siehe auch *Datumsprüfungen*.
YYYY	Vier Stellen, die nach einer gültigen Jahreszahl (0000-2699) abgesucht werden sollen.

Zeichen	Bedeutung
Z	Eine Stelle, die nach einem Zeichen abgesucht werden soll, dessen linkes Halbbyte hexadezimal 3 oder 7 und dessen rechtes Halbbyte hexadezimal 0 – 9 ist. Damit können Sie korrekt nach Ziffern in negativen Zahlen suchen. Mit N (womit Sie eine Stelle nach einer Ziffer absuchen können) erhalten Sie bei Suche von Ziffern in negativen Zahlen falsche Ergebnisse, da das Vorzeichen in der letzten Stelle der Zahl gespeichert ist, wodurch diese Stelle hexadezimal als Nicht-Ziffer dargestellt wird. Geben Sie innerhalb einer Maske für jede Reihe numerischer Stellen, die geprüft werden sollen, nur jeweils ein Z an.

Maskenlänge

Welche Stellen abgesucht werden sollen, ergibt sich aus der Definition der Maske.

Beispiel:

```
DEFINE DATA LOCAL
1 #CODE (A15)
END-DEFINE
...
IF #CODE = MASK (NN'ABC'....NN)
...
```

Die ersten beiden Stellen von #CODE werden nach einem numerischen Wert abgesucht, die 3. bis 5. Stelle nach dem Wert ABC, die 10. und 11. Stelle nach einem numerischen Wert; die 6. bis 9. und 12. bis 15. Stelle werden nicht überprüft.

Datumsprüfungen

Pro Maske darf nur ein Datum geprüft werden. Wenn in der Maske derselbe Datumsbestandteil (JJJ, DD, MM, YY or YYYY) mehr als einmal angegeben wird, dann wird nur der Wert der letzten Ausprägung auf Konsistenz mit anderen Datumsbestandteilen geprüft.

Wird bei der Prüfung eines Tagesdatums (DD) keine Monatsangabe (MM) in der Maske gemacht, wird der aktuelle Monat angenommen.

Wird bei der Prüfung eines Tagesdatums (DD) oder Julianischen Tagesdatums (JJJ) keine Jahresangabe (YY bzw. YYYY) in der Maske gemacht, wird das aktuelle Jahr angenommen.

Bei der Prüfung einer zweistelligen Jahreszahl (YY) wird das aktuelle Jahrhundert angenommen, sofern kein Sliding Window oder Fixed Window gesetzt ist. Weitere Einzelheiten über Sliding oder Fixed Windows, entnehmen Sie dem Profilparameter YSLW in der *Parameter Reference*-Dokumentation.

Beispiel 1:

```
MOVE 1131 TO #DATE (N4)
IF #DATE = MASK (MMDD)
```

In diesem Beispiel werden Monat und Tag auf ihre Gültigkeit überprüft. Der Monatswert `11` ist gültig, wohingegen der Tageswert `31` ungültig ist, da der 11. Monat nur 30 Tage hat.

Beispiel 2:

```
IF #DATE(A8) = MASK (MM'/'DD'/'YY)
```

In diesem Beispiel wird überprüft, ob das Feld `#DATE` ein gültiges Datum im Format `MM/DD/YY` (Monat/Tag/Jahr) enthält.

Beispiel 3:

```
IF #DATE (A8) = MASK (1950-2020MMDD)
```

In diesem Beispiel wird der Inhalt des Feldes `#DATE` auf eine vierstellige Zahl im Bereich 1950 bis 2020 geprüft, auf die ein gültiger Monat und Tag im aktuellen Jahr folgen:

Anmerkung:
Obwohl es so aussieht, ermöglicht die oben angegebene Maske nicht das Abprüfen auf ein gültiges Datum in den Jahren 1950 - 2020, weil der numerische Wertebereich 1950-2020 unabhängig von der Gültigkeitsprüfung für Monat und Tag abgeprüft wird. Die Prüfung liefert die beabsichtigten Ergebnisse mit Ausnahme des 29. Februars, denn an diesem Tag ist das Ergebnis davon abhängig, ob das aktuelle Jahr ein Schaltjahr ist oder nicht. Um zusätzlich zur Datumsgültigkeitsprüfung auf einen bestimmten Bereich von Jahren zu prüfen, bauen Sie eine Gültigkeitsprüfung für das Datum und eine andere für den Bereich in Ihrem Programm ein.

```
IF #DATE (A8) = MASK (YYYYMMDD) AND #DATE = MASK (1950-2020)
```

Beispiel 4:

```
IF #DATE (A4) = MASK (19-20YY)
```

In diesem Beispiel wird überprüft, ob das Feld `#DATE` eine zweistellige Zahl im Bereich von `19` bis `20`, gefolgt von einem gültigen zweistelligen Jahr (`00` bis `99`) enthält. Das Jahrhundert wird wie oben beschrieben von Natural angegeben.

Anmerkung:
Obwohl es so aussieht, ermöglicht die oben angegebene Maske nicht das Abprüfen auf ein gültiges Jahr im Bereich von `1900` bis `2099`, weil der numerische Wertebereich `19 - 20` unabhängig von der Gültigkeitsprüfung für das Jahr abgeprüft wird. Um auf Bereiche von Jahren abzuprüfen, bauen Sie eine Gültigkeitsprüfung für das Datum und eine andere für den Bereich in Ihrem Programm ein:

```
IF #DATE (A10) = MASK (YYYY'-'MM'-'DD) AND #DATE = MASK (19-20)
```

Prüfung unter Verwendung von Konstanten oder Variablen

Ist der für die Maskenprüfung verwendete Wert eine Konstante oder der Inhalt einer Variablen, dann muss dieser Wert (*operand2*) unmittelbar nach der *mask-definition* angegeben werden.

operand2 muss mindestens so lang sein wie die Maske.

In der Maske geben Sie für jede zu überprüfende Stelle ein `X` und für jede nicht zu überprüfende Stelle einen Punkt (.) (oder `?` oder _) an.

Beispiel:

```
DEFINE DATA LOCAL
1 #NAME (A15)
END-DEFINE
...
IF #NAME = MASK (..XX) 'ABCD'
...
```

Hier wird geprüft, ob die 3. bis 4. Stelle des Feldes #NAME den Wert CD enthält. Die ersten beiden Stellen werden nicht überprüft.

Wieviele Stellen geprüft werden, hängt von der Länge der definierten Maske ab. Die Maske wird immer linksbündig auf das zu überprüfende Feld bzw. die zu prüfende Konstante ausgerichtet. *Operand1* muss dasselbe Format haben wie *operand2*.

Hat das zu überprüfende Feld (*operand1*) Format A, muss ein konstanter Wert (*operand2*) in Apostrophen stehen. Ist das Feld numerisch, muss der Wert eine Zahl oder der Inhalt eines numerischen Datenbankfeldes bzw. einer numerischen Benutzervariablen sein.

In jedem Fall werden Zeichen/Stellen, die nicht an einer in der Maskendefinition mit X markierten Stelle stehen, ignoriert.

Die MASK-Bedingung ist erfüllt, wenn alle in der Maske angegebenen Stellen dem geforderten Wert entsprechen.

Beispiel:

```
** Example 'LOGICX01': MASK option in logical condition
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 CITY
END-DEFINE
*
HISTOGRAM EMPLOY-VIEW CITY
  IF CITY =
MASK (....XX) '....NN'


    DISPLAY NOTITLE CITY *NUMBER
  END-IF
END-HISTOGRAM
*
END
```

In diesem Beispielprogramm werden nur Datensätze akzeptiert, bei denen das Feld CITY einen Wert enthält, der an der 5. und 6. Stelle jeweils ein N hat.

Bereichsprüfungen

Bei Bereichsprüfungen wird die Anzahl der verifizierten Stellen durch die Genauigkeit des in der Maske angegebenen Wertes definiert. Zum Beispiel würde die Maske (...193...) die Stellen 4 bis 6 nach einer dreistelligen Zahl im Bereich von 000 bis 193 überprüfen.

Weitere Beispiele für Masken-Definitionen:

- In diesem Beispiel wird überprüft, ob alle Stellen des Feldes #NAME einen Buchstaben enthalten:

  ```
  IF #NAME (A10) = MASK (AAAAAAAAAA)
  ```

- In diesem Beispiel wird überprüft, ob die 4. bis 6. Stelle von #NUMBER eine Zahl enthält:

  ```
  IF #NUMBER (A6) = MASK (...NNN)
  ```

- In diesem Beispiel wird überprüft, ob die 4. - 6. Stelle von #VALUE den Wert 123 enthält:

  ```
  IF #VALUE(A10) = MASK (...'123')
  ```

- In diesem Beispiel wird überprüft, ob das Nummernschild-Feld #LICENSE ein KFZ-Kennzeichen enthält, das mit NY– beginnt, gefolgt vom Wert der letzten fünf Stellen des Feldes #VALUE:

  ```
  DEFINE DATA LOCAL
  1 #VALUE(A8)
  1 #LICENSE(A8)
  END-DEFINE
  INPUT 'ENTER KNOWN POSITIONS OF LICENSE PLATE:' #VALUE
  IF #LICENSE = MASK ('NY-'XXXXX) #VALUE
  ```

- Die folgende Bedingung wird von jedem Wert erfüllt, der NAT und AL enthält, ganz gleich wieviele andere Zeichen zwischen NAT und AL stehen (dies würde z.B. auf die Werte NATURAL und NATIONALITAET genauso zutreffen wie auf den Wert NATAL):

  ```
  MASK('NAT'*'AL')
  ```

Auf gepackte oder ungepackte numerische Daten abprüfen

In Altanwendungen sind gepackte oder ungepackte numerische Variablen häufig mit alphanumerischen oder binären Feldern neu definiert. Solche Neudefinitionen sind nicht empfehlenswert, weil die Verwendung einer gepackten oder ungepackten Variablen in einer Zuweisung oder Berechnung zu Fehlern oder unvorhersagbaren Ergebnissen führen kann.

Um den Inhalt einer solchen neu definierten Variablen auf Gültigkeit hin abzuprüfen, bevor die Variable verwendet wird, benutzen Sie die Option N (siehe *Zeichen in einer Maske*) so oft wie die Anzahl der Stellen minus 1 mal, gefolgt von einer einzelnen Option Z.

Beispiele :

```
IF #P1 (P1) = MASK (Z)
IF #N4 (N4) = MASK (NNNZ)
IF #P5 (P5) = MASK (NNNNZ)
```

Weitere Informationen zum Abprüfen von Feldinhalten siehe *MASK-Option im Vergleich zur IS-Option*.

MASK-Option im Vergleich zur IS Option

Dieser Abschnitt beschreibt den Unterschied zwischen der MASK-Option und der IS-Option und enthält ein Beispielprogramm, das den Unterschied zwischen den beiden Optionen veranschaulicht.

Mit der IS-Option können Sie prüfen, ob der Inhalt eines alphanumerischen oder Unicode-Felds in ein bestimmtes anderes Format umgesetzt werden kann. Sie können mit dieser Option jedoch nicht abprüfen, ob der Wert eines alphanumerischen Feldes im angegeben Format vorliegt.

Mit der MASK-Option können Sie den Inhalt einer neu definierten gepackten oder ungepackten numerischen Variablen auf Gültigkeit abprüfen.

Beispiel zur Erläuterung des Unterschieds:

```
** Example 'LOGICX09': MASK versus IS option in logical condition
************************************************************************
DEFINE DATA LOCAL
1 #A2   (A2)
1 REDEFINE #A2
  2 #N2 (N2)
1 REDEFINE #A2
  2 #P3 (P3)
1 #CONV-N2 (N2)
1 #CONV-P3 (P3)
END-DEFINE
*
#A2 := '12'
WRITE NOTITLE   'Assignment #A2 := "12" results in:'
PERFORM SUBTEST
#A2 := '-1'
WRITE NOTITLE / 'Assignment #A2 := "-1" results in:'
PERFORM SUBTEST
#N2 :=  12
WRITE NOTITLE / 'Assignment #N2 :=  12  results in:'
PERFORM SUBTEST
#N2 :=  -1
WRITE NOTITLE / 'Assignment #N2 :=  -1  results in:'
PERFORM SUBTEST
#P3 :=  12
WRITE NOTITLE / 'Assignment #P3 :=  12  results in:'
PERFORM SUBTEST
#P3 :=  -1
WRITE NOTITLE / 'Assignment #P3 :=  -1  results in:'
PERFORM SUBTEST
*
DEFINE SUBROUTINE SUBTEST
IF #A2 IS (N2) THEN
  #CONV-N2 := VAL(#A2)
  WRITE NOTITLE 12T '#A2 can be converted to' #CONV-N2 '(N2)'
END-IF
IF #A2 IS (P3) THEN
  #CONV-P3 := VAL(#A2)
  WRITE NOTITLE 12T '#A2 can be converted to' #CONV-P3 '(P3)'
END-IF
IF #N2 = MASK(NZ) THEN
  WRITE NOTITLE 12T '#N2 contains the valid unpacked number' #N2
END-IF
IF #P3 = MASK(NNZ) THEN
  WRITE NOTITLE 12T '#P3 contains the valid packed number' #P3
END-IF
END-SUBROUTINE
*
END
```

Ausgabe des Programms LOGICX09:

```
Assignment #A2 := ’12’ results in:
           #A2 can be converted to  12 (N2)
           #A2 can be converted to   12 (P3)
           #N2 contains the valid unpacked number  12

Assignment #A2 := ’-1’ results in:
           #A2 can be converted to  -1 (N2)
           #A2 can be converted to   -1 (P3)

Assignment #N2 :=  12  results in:
           #A2 can be converted to  12 (N2)
           #A2 can be converted to   12 (P3)
           #N2 contains the valid unpacked number  12

Assignment #N2 :=  -1  results in:
           #N2 contains the valid unpacked number  -1

Assignment #P3 :=  12  results in:
           #P3 contains the valid packed number   12

Assignment #P3 :=  -1  results in:
           #P3 contains the valid packed number   -1
```

MODIFIED-Option - Prüfen ob Feldinhalt verändert worden ist

Syntax:

operand1 [**NOT**] **MODIFIED**

Mit dieser Option wird überprüft, ob der Inhalt eines Feldes während der Ausführung eines INPUT- oder PROCESS PAGE-Statements verändert worden ist. Als Voraussetzung muss dem Feld eine Kontrollvariable mit dem Parameter CV zugewiesen worden sein.

Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1		S	A													C			nein	nein

Von einem INPUT- oder PROCESS PAGE-Statement referenzierte Kontrollvariablen erhalten immer den Status "not modified" (nicht verändert), wenn die Map zur Ausgabe an das Terminal übertragen wird.

Wird der Inhalt eines Feldes, das eine Kontrollvariable (*operand1*) referenziert, verändert, erhält die Kontrollvariable den Status "modified" (verändert). Referenzieren mehrere multiple Felder dieselbe Kontrollvariable, so erhält die Variable den Status MODIFIED, wenn mindestens eines dieser Felder verändert wurde.

Ist die Kontrollvariable *operand1* ein Array, so erhält die Variable den Status "modified", wenn mindestens eins der Elemente des Arrays verändert wurde (ODER-Verknüpfung).

Beispiel für MODIFIED-Option:

```
** Example 'LOGICX06': MODIFIED option in logical condition
************************************************************************
DEFINE DATA LOCAL
1 #ATTR (C)
1 #A    (A1)
1 #B    (A1)
END-DEFINE
*
MOVE (AD=I) TO #ATTR
*
INPUT (CV=#ATTR) #A #B
IF #ATTR NOT MODIFIED
  WRITE NOTITLE 'FIELD #A OR #B HAS NOT BEEN MODIFIED'
END-IF
*
IF #ATTR MODIFIED
  WRITE NOTITLE 'FIELD #A OR #B HAS BEEN MODIFIED'
END-IF
*
END
```

Ausgabe des Programms LOGICX06:

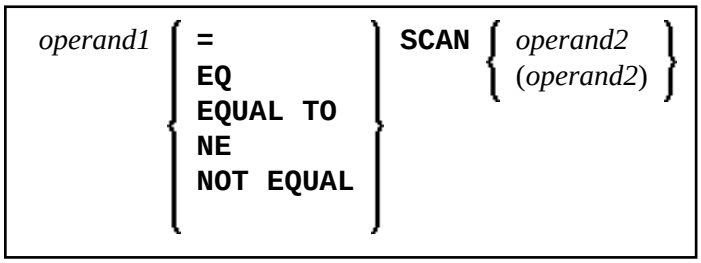
```
#A   #B
```

Nach Eingabe eines Wertes und Drücken der Freigabetaste, wird die folgende Ausgabe angezeigt:

```
FIELD #A OR #B HAS BEEN MODIFIED
```

SCAN-Option - Nach einem bestimmten Wert in einem Feld suchen

Syntax:



Operanden-Definitionstabelle:

Operand	Mögliche Struktur						Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N		A	U	N	P									ja	nein
operand2	C	S					A	U					B*						ja	nein

* *Operand2* darf nur binär sein, wenn *operand1* alphanumerisch oder Unicode ist. Wenn *operand1* Format U hat und *operand2* Format B, dann muss die Länge von *operand2* geradzahlig sein.

Mit der SCAN-Option können Sie nach einem bestimmten Wert in einem Feld suchen.

Der zu suchende Wert (*operand2*) kann entweder als alphanumerische oder Unicode-Konstante (eine in Apostrophen stehende Zeichenkette) oder als Inhalt einer alphanumerischen oder Unicode-Variablen (Datenbankfeld oder Benutzervariable) angegeben werden.

Vorsicht:
Dem Wert nachgestellte Leerzeichen in *operand1* werden automatisch eliminiert. Deshalb kann die SCAN-Option nicht zum Suchen nach Leerzeichen verwendet werden. *operand1* und *operand2* dürfen vorangestellte oder eingebettete Leerzeichen enthalten. Falls *operand2* nur aus Leerzeichen besteht, dann gilt, unabhängig vom Wert in *operand1*, die Suche immer als erfolgreich; vergleiche EXAMINE FULL-Statement, wenn vorangestellte Leerzeichen bei der Suche nicht ignoriert werden sollen.

Das Feld, das abgesucht werden soll (*operand1*), kann das Format A, N, P oder U haben. Die SCAN-Operation kann mit den Operatoren EQ (gleich) oder NE (ungleich) angegeben werden.

Die Länge der gesuchten Zeichenkette sollte kürzer als die Länge des abgesuchten Feldes sein. Ist die Länge des Wertes gleich der des Feldes, sollte statt der SCAN-Option ein relationaler Ausdruck mit dem Operator EQUAL TO verwendet werden.

Beispiel für SCAN-Option:

```
** Example 'LOGICX02': SCAN option in logical condition
************************************************************************
DEFINE DATA
LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
*
1 #VALUE   (A4)
1 #COMMENT (A10)
END-DEFINE
*
INPUT 'ENTER SCAN VALUE:' #VALUE
LIMIT 15
*
HISTOGRAM EMPLOY-VIEW FOR NAME
  RESET #COMMENT
  IF NAME = SCAN #VALUE
    MOVE 'MATCH' TO #COMMENT
  END-IF
  DISPLAY NOTITLE NAME *NUMBER #COMMENT
END-HISTOGRAM
*
END
```

Ausgabe des Programms LOGICX02:

```
ENTER SCAN VALUE:
```

Eine Suche nach LL führt zu drei Übereinstimmungen bei 15 Namen:

```
        NAME           NMBR      #COMMENT
-------------------- --------- ----------

ABELLAN                      1 MATCH
ACHIESON                     1
ADAM                         1
ADKINSON                     8
AECKERLE                     1
AFANASSIEV                   2
AHL                          1
AKROYD                       1
ALEMAN                       1
ALESTIA                      1
ALEXANDER                    5
ALLEGRE                      1 MATCH
ALLSOP                       1 MATCH
ALTINOK                      1
ALVAREZ                      1
```

SPECIFIED-Option - Prüfen ob ein Wert für einen optionalen Parameter übergeben wird

Syntax:

parameter-name [**NOT**] **SPECIFIED**

Mit dieser Option wird überprüft, ob ein optionaler Parameter in einem aufgerufenen Objekt (Subprogramm, externe Subroutine, Dialog oder ActiveX Control) vom aufrufenden Objekt einen Wert aufgenommen hat oder nicht.

Ein optionaler Parameter ist ein Feld, das mit dem Schlüsselwort OPTIONAL im DEFINE DATA PARAMETER-Statement des aufgerufenen Objekts definiert worden ist. Wenn ein Feld als OPTIONAL definiert wird, kann ein Wert von einem aufrufenden Objekt an dieses Feld übergeben werden.

Im aufrufenden Statement wird die Notation *n*X benutzt, um Parameter anzugeben, für die keine Werte übergeben werden.

Wenn Sie einen optionalen Parameter verarbeiten, der keinen Wert empfangen hat, so führt dies zu einem Laufzeit-Fehler. Um solch einen Fehler zu vermeiden, benutzen Sie die SPECIFIED-Option im aufgerufenen Objekt, um zu überprüfen, ob ein optionaler Parameter einen Wert aufgenommen hat oder nicht, und ihn erst dann zu verarbeiten, wenn dies der Fall ist.

parameter-name ist der Name des im DEFINE DATA PARAMETER-Statement des aufgerufenen Objekts angegebenen Parameters.

Bei einem nicht als OPTIONAL definierten Feld ist die SPECIFIED-Bedingung immer TRUE.

Beispiel für die SPECIFIED-Option:

Aufrufendes Programm:

```
** Example 'LOGICX07': SPECIFIED option in logical condition
************************************************************************
DEFINE DATA LOCAL
1 #PARM1 (A3)
1 #PARM3 (N2)
END-DEFINE
*
#PARM1 := 'ABC'
#PARM3 := 20
*
CALLNAT 'LOGICX08' #PARM1 1X #PARM3
*
END
```

Aufgerufenes Subprogramm:

```
** Example 'LOGICX08': SPECIFIED option in logical condition
************************************************************************
DEFINE DATA PARAMETER
1 #PARM1 (A3)
1 #PARM2 (N2) OPTIONAL
1 #PARM3 (N2) OPTIONAL
END-DEFINE
*
WRITE '=' #PARM1
*
IF #PARM2 SPECIFIED
  WRITE '#PARM2 is specified'
  WRITE '=' #PARM2
ELSE
  WRITE '#PARM2 is not specified'
* WRITE '=' #PARM2               /* would cause runtime error NAT1322
END-IF
*
IF #PARM3 NOT SPECIFIED
  WRITE '#PARM3 is not specified'
ELSE
  WRITE '#PARM3 is specified'
  WRITE '=' #PARM3
END-IF
END
```

Ausgabe des Programms LOGICX07:

```
Page      1                                              04-12-15  11:25:41

#PARM1: ABC
#PARM2 is not specified
#PARM3 is specified
#PARM3:  20
```