Kapitel 3

# Definition und Initialisierung von Variablen

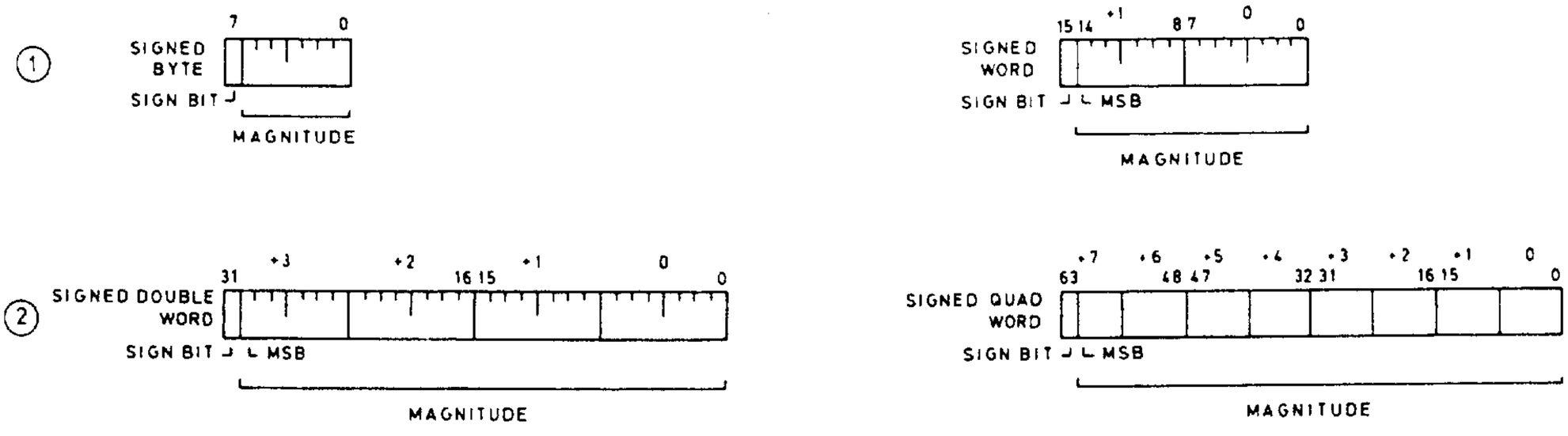
(für MASM-, ASM86- und ASM286-Programmierer)

## 3.1 Datentypen

Obwohl Bytes und Words die fundamentalen Datentypen von Operanden sind, unterstützen die Mikroprozessoren 8086/80186/80286 zusätzliche Interpretationen dieser Bytes und Words.

Ein Befehl, der auf einen Operanden Bezug nimmt, kann die folgenden zusätzlichen Datentypen erkennen.

### 3.1.1 INTEGER (vorzeichenbehaftete Binärzahl)



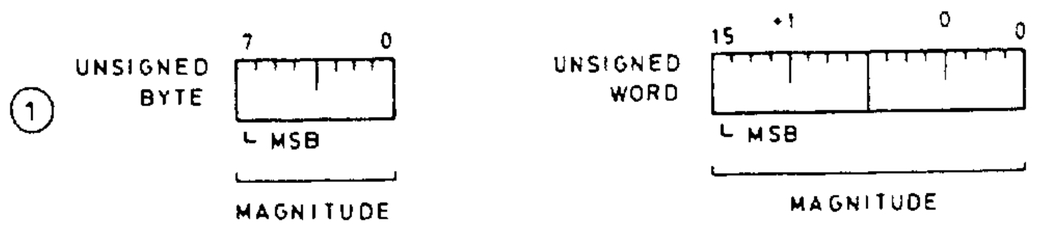

Eine vorzeichenbehaftete (Signed) Binärzahl kann entweder 8 oder 16 Bit lang sein ①. Dabei wird das MSB (Most Significant Bit) als das Vorzeichen der Zahl interpretiert: 0 = positiv und 1 = negativ.

Negative Zahlen werden in der Zweierkomplement-Notation dargestellt. Da das MSB für das Vorzeichen verwendet wird, liegt eine 8-Bit-Integerzahl im Bereich von -128 bis +127 und eine 16-Bit-Integerzahl im Bereich von -32768 bis +32767.

Liegt im System eine numerische Prozessorerweiterung in Form des 80X87 vor, werden noch zusätzlich 32-Bit bzw. 64-Bit vorzeichenbehaftete Integerzahlen unterstützt ②.

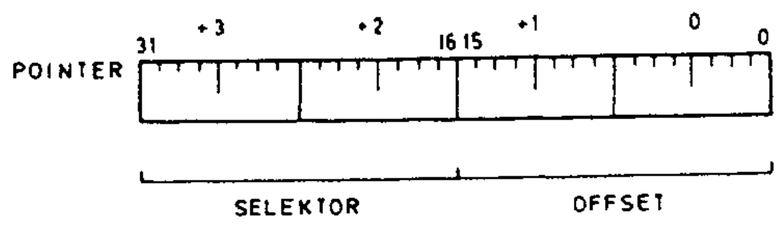
Eine 32-Bit-Integerzahl liegt im Bereich von $-2x10^9$ bis $+2x10^9$ und eine 64-Bit-Integerzahl liegt im Bereich von $-9x10^{18}$ bis $+9x10^{18}$.

## 3.1.2 ORDINAL (vorzeichenlose Binärzahl)



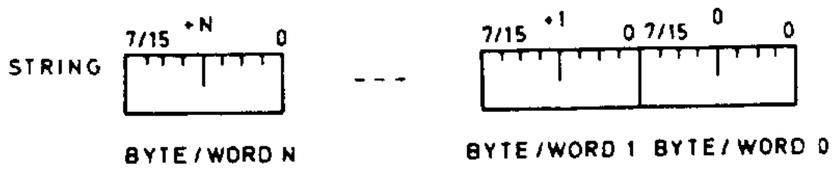

Eine vorzeichenlose (unsigned) Binärzahl kann entweder 8 oder 16 Bit lang sein ①. Dabei bestimmen alle Bits die Größe (Magnitude) einer Zahl. Der Wertebereich einer 8-Bit vorzeichenlosen Binärzahl liegt zwischen 0 bis 255, der einer 16-Bit vorzeichenlosen Binärzahl zwischen 0 bis 65535.

## 3.1.3 POINTER



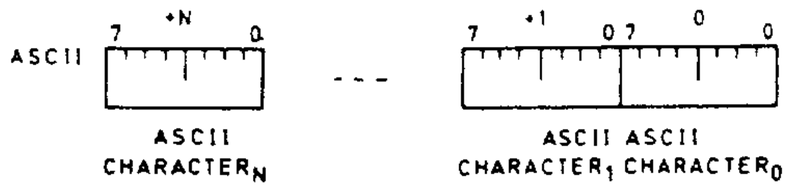

Ein Pointer ist eine 32-Bit-Adreßeinheit, die aus einer Segment-Selektorkomponente und einer Offsetkomponente zusammengesetzt ist. Jede Komponente ist ein 16-Bit-Word.

## 3.1.4 STRING



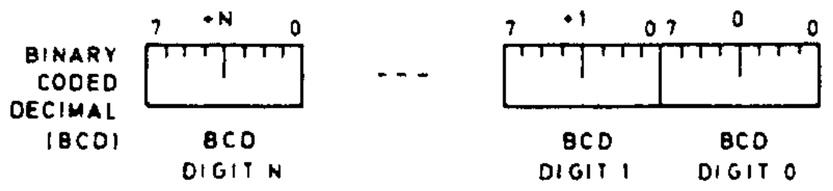

Ein String ist eine zusammenhängende Sequenz von Bytes oder Words. Jeder String kann zwischen 1 bis 64 KBytes lang sein. Im Allgemeinen enthält ein String ASCII-Zeichen. Die 8086/80186/80286-CPUs kennen Befehle, um Strings zu transferieren, zu prüfen oder zu verändern.

## 3.1.5 ASCII (alphanumerische und Steuerzeichen)




Alphanumerische Zeichen und Steuerzeichen werden durch ein Byte dargestellt. Dabei wird der ASCII-Standard für eine Zeichendarstellung benützt.

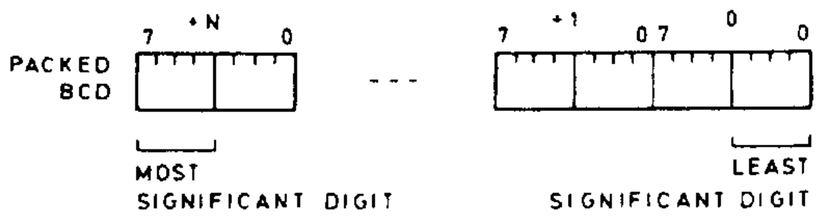
## 3.1.6 BCD (Binär codierte Dezimalzahlen)




Dezimale Zahlen (BCD) werden als vorzeichenlose Byteeinheiten gespeichert. Dabei wird ein Digit in einem Byte hinterlegt. Man spricht von ungepackten Dezimalzahlen. Die Größe einer solchen Zahl wird bestimmt durch das untere Halbbyte (Low Nibble). Die hexadezimalen Werte 0 bis 9 sind gültig und werden als dezimale Zahlen interpretiert.

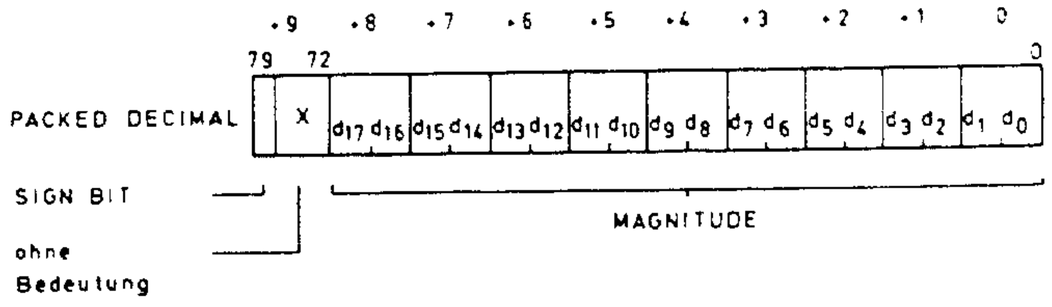
Für Multiplikationen und Divisionen muß das obere Halbbyte Null sein. Für Additionen und Subtraktionen darf das obere Halbbyte beliebige Werte enthalten.

## 3.1.7 PACKED BCD (gepackte binär codierte Dezimalzahlen)



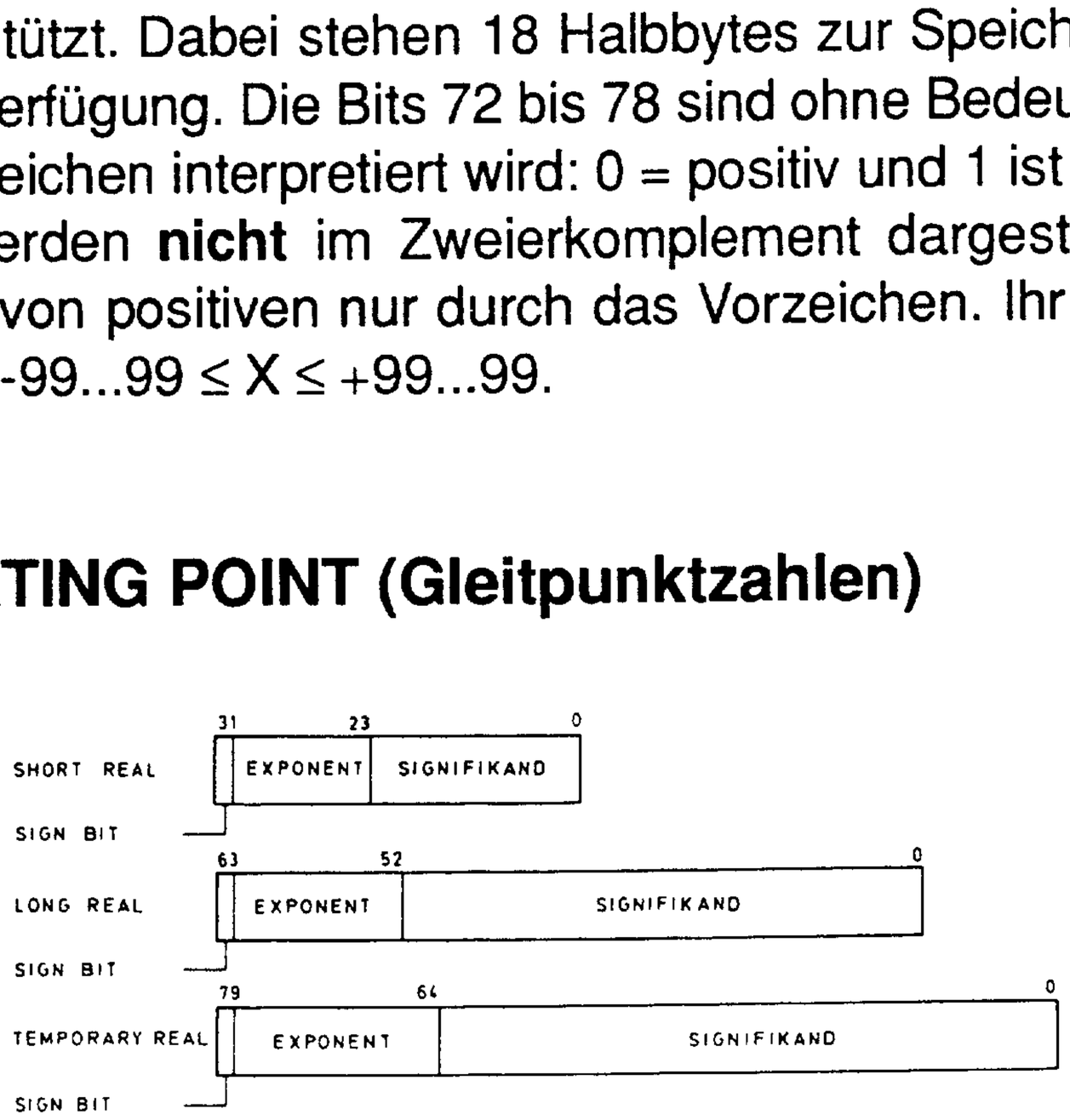

Gepackte Dezimalzahlen (Packed BCD) werden als vorzeichenlose Byteeinheiten gespeichert. Dabei wird in jedem Halbbyte (Nibble) ein dezimales Digit gespeichert. Das Digit im oberen Halbbyte ist das Most Significant Digit, das im unteren Halbbyte ist das Least Significant Digit. Die Werte 0 bis 9 sind für jedes Halbbyte gültig, so daß der Bereich einer gepackten Dezimalzahl zwischen 00 bis 99 liegt.

## 3.1.8 18-DIGIT PACKED BCD




Liegt im System eine numerische Prozessorerweiterung in Form des 80X87 vor, wird zusätzlich eine 80-Bit vorzeichenbehaftete gepackte Dezimalzahl unterstützt. Dabei stehen 18 Halbbytes zur Speicherung dezimaler Digits zur Verfügung. Die Bits 72 bis 78 sind ohne Bedeutung, während Bit 79 als Vorzeichen interpretiert wird: 0 = positiv und 1 ist negativ. Negative Zahlen werden **nicht** im Zweierkomplement dargestellt. Sie unterscheiden sich von positiven nur durch das Vorzeichen. Ihr Zahlenbereich liegt zwischen -99...99 ≤ X ≤ +99...99.

## 3.1.9 FLOATING POINT (Gleitpunktzahlen)




Die angegebenen Gleitpunktformate werden nur vom 80X87-Arithmetikprozessor unterstützt.

- Die Short Real-Zahl ist 32 Bit lang und wird wie eine 32-Bit vorzeichenbehaftete Integerzahl in einem Double Word gespeichert.
  Ihr Zahlenbereich liegt bei $1.18x10^{-38} \leq |x| \leq 3.4x10^{38}$.
- Die Long Real-Zahl ist 64 Bits lang und wird wie eine 64-Bit vorzeichenbehaftete Integerzahl in einem Quad Word gespeichert.
  Ihr Zahlenbereich liegt bei $2.23x10^{-308} \leq |x| \leq 1.80x10^{308}$.
- Die Temporary Real-Zahl ist 80 Bits lang und wird in einem Feld von 10 Bytes gespeichert.
  Ihr Zahlenbereich liegt bei $3.30x10^{-4932} \leq |x| \leq 1.21x10^{4932}$.

## 3.2 Datenzuweisung

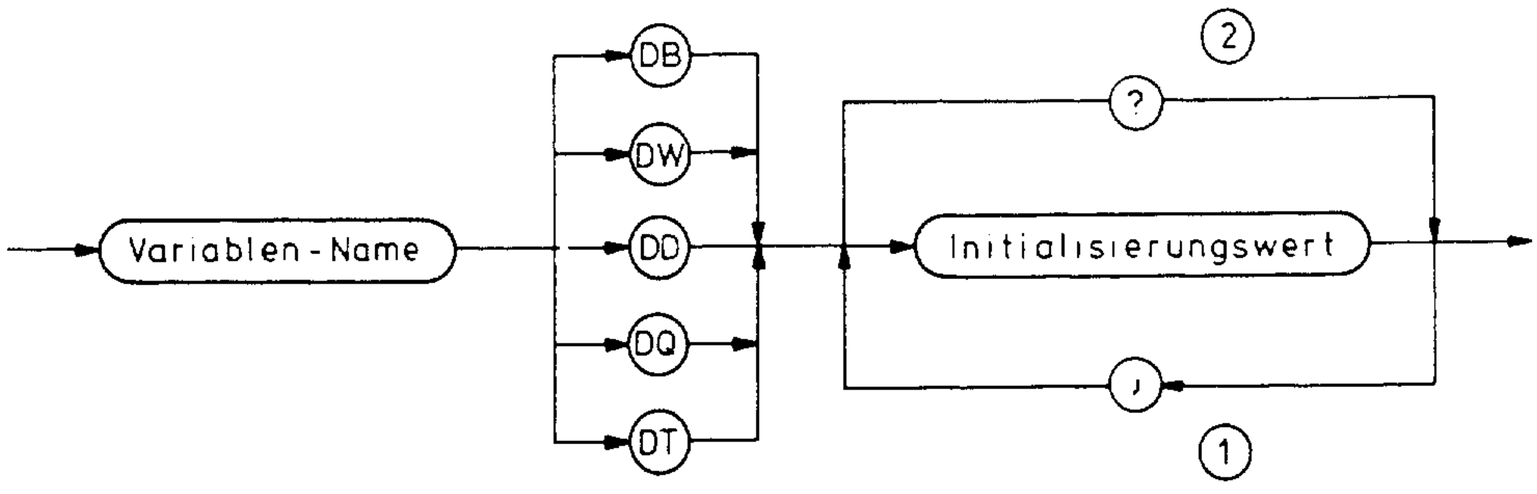
Die Mikroprozessoren 8086/80186/80286 zusammen mit der numerischen Prozessorerweiterung unterstützen fünf Datentypen:

| | | | |
|---|---|---|---|
| 1. | BYTE | – ein Byte | – ist ein 8086/80186/80286-Datentyp |
| 2. | WORD | – ein Word (zwei Bytes) | – ist ein 8086/80186/80286-Datentyp |
| 3. | DWORD | – ein Double Word (vier Bytes) | – ist ein 8086/80186/80286-oder 80X87-Datentyp |
| 4. | QWORD | – ein Quad Word (acht Bytes) | – ist ein 80X87-Datentyp |
| 5. | TBYTE | – ein Ten Byte (zehn Bytes) | – ist ein 80X87-Datentyp |

Damit die angegebenen Datentypen in einem Anwenderprogramm benützt werden können, sind Sie zunächst mit Hilfe des Assembler-Datenzuweisungs-Statements zu spezifizieren.

Im Folgenden soll die allgemeine Syntax des Datenzuweisungs-Statements gezeigt werden. Weiterhin wird diskutiert, wie dieses Statement Initialisierungswerte für Programmdaten spezifiziert.

## Sytaxdiagramm des Datenzuweisungs-Statements




**Variablen-Name** ist der metasprachliche Ausdruck eines Namens, den der Anwenderprogrammierer für eine Variable vergeben kann.

Die Schlüsselwörter **DB**, **DW**, **DD**, **DQ** und **DT** müssen angegeben werden. Sie reservieren die notwendige Anzahl von Bytes für eine zu definierende Variable.

| | | | |
|---|---|---|---|
| 1-Byte-Initialisierung: Byte<br>[Variablen-Name] | DB | Initialisierungswert | [,.....] |
| 2-Byte-Initialisierung: Word<br>[Variablen-Name] | DW | Initialisierungswert | [,.....] |
| 4-Byte-Initialisierung: Double Word<br>[Variablen-Name] | DD | Initialisierungswert | [,.....] |
| 8-Byte-Initialisierung: Quad Word<br>[Variablen-Name] | DQ | Initialisierungswert | [,.....] |
| 10-Byte-Initialisierung: Ten Byte<br>[Variablen-Name] | DT | Initialisierungswert | [,.....] |

**Initialisierungswert** ist der metasprachliche Ausdruck für einen Wert, der einer einzelnen Speichereinheit zugewiesen werden kann. Sollen mehrere Speichereinheiten initialisiert werden, sind die Werte mit Komma voneinander zu trennen ①.

Für den Fall, daß eine oder mehrere Speichereinheiten reserviert, also nicht mit Werten belegt werden sollen, ist für Initialisierungswert das Sonderzeichen ? zu verwenden ②.

### 3.2.1 Hinweise zur Darstellung von Initialisierungswerten

Byte-Integer, Word-Integer, Double-Word-Integer und Quad-Word-Integer können in binärer, dezimaler oder hexadezimaler Notation dargestellt werden.

**Beispiele:**
```
3EH, -23D              ; Byte-Variable
10011101B, -20C8H      ; Word-Variable
-834537212             ; Double-Word-Variable
97653482735D           ; Quad-Word-Variable
```

Durch Angabe einer geeigneten Nachsilbe (Suffix) im Initialisierungswert wird dem Assembler die gewählte Notation mitgeteilt:
D bedeutet dezimale Notation (Zahlenbasis 10)
B bedeutet binäre Notation (Zahlenbasis 2)
H bedeutet hexadezimale Notation (Zahlenbasis 16)

**Hinweis:** Initialisierungswerte, bei denen kein Notationszeichen angegeben ist, werden vom Assembler als dezimale Werte interpretiert. Jeder Initialisierungswert, unabhängig von seiner Notation, wird vom Assembler in die hexadezimale Darstellungsform übertragen.

Gepackte oder ungepackte BCD-Zahlen wird der Anwender aus praktischen Gründen in dezimaler Notation darstellen.

**Beispiele:**
```
-2399327655            ; Packed Decimal (max. 18 Digits)
33H, 44H, 55H          ; Packed BCD
3D, 8, 9D              ; Unpacked BCD
```

Alphanumerische Zeichen und Sonderzeichen sind in Apostrophe einzuschließen. Der Assembler interpretiert die so gekennzeichneten Initialisierungswerte als ASCII-Zeichen und übersetzt sie in ihre hexadezimale Darstellungsform.

**Beispiele:**
```
'H', 'I', 'G', 'H'     ; Einzelne ASCII-Zeichen
'HIGH'                 ; String
```

Short Real, Long Real und Temporary Real sind in folgender Weise zu notieren:

1. Im Festpunktformat, d. h. als dezimale Realzahlen **ohne** Exponentenangabe

**Beispiele:**
```
1.234
0.0314
1.
8257.0029
```

oder

2. im Gleitpunktformat (Floating Point), d.h. als dezimale Realzahl **mit** Exponentenangabe

**Beispiele:** $6.8E-27 = 6.8x10^{-27}$

$-1.67E\ 308 = -1.67x10^{308}$

Pointer können durch eine Double-Word-Variable dargestellt werden. Dies bedeutet, daß der initialisierte Wert einer kompletten Base:Offset-Adresse entspricht.

**Beispiel:**

```
0100    0200H ; Double-Word-Variable
Base    Offset
```

Dabei entspricht das Word unter der niederen Adresse der Offsetkomponente und das Word unter der höheren Adresse der Basekomponente.

## 3.2.2 Initialisierungsbeispiele mit Übersetzungsprotokoll

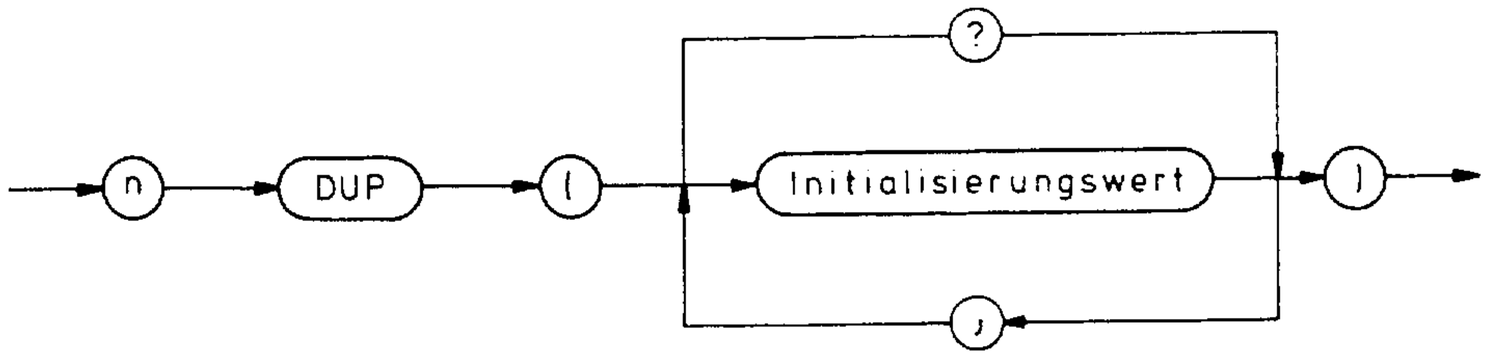
```
LOC  OBJ                  LINE     SOURCE

                             1             NAME    BEISPIEL_5
                             2
                             3     ASSUME DS:DATEN_PROG
                             4
----                         5     DATEN_PROG SEGMENT
                             6
0000 3E                      7     BYTE_INTEGER          DB      3EH,-8AH,23D        ;Byte Variable
0001 76
0002 17
0003 0900                    8     WORD_INTEGER          DW      1001B,-20C8H        ;Word Variable
0005 38DF
0007 04F941CE                9     DOUBLE_WORD_INTEGER   DD      -834537212D         ;Double Word Variable
000B EFE499BC160000          10    QUAD_WORD_INTEGER     DQ      97653482735D        ;Quad Word Variable
     00
0013 D46D8F03                11    SHORT_REAL            DD      8.43E-37            ;Double Word Variable
0017 1566BACD1BBAED          12    LONG_REAL             DQ      -1.67E+308          ;Quad Word Variable
     FF
001F 40EEBCFB45C300          13    TEMPORARY_REAL        DT      0.0000000156        ;Ten Byte Variable
     86E53F
0029 55763299230000          14    PACKED_DECIMAL        DT      -2399327655         ;Ten Byte Variable
     000080
0033 48                      15    ASCII                 DB      'H','E','L'         ;ASCII Zeichen
0034 45
0035 4C
0036 48454C4C4F              16    STRING                DB      'HELLO'             ;Zeichenkette
003B 4548                    17    WORD_STRING           DW      'HE'                ;Zeichenkette
003D 03                      18    UNPACKED_BCD          DB      3D,8D,9D            ;Byte Variable
003E 08
003F 09
0040 33                      19    PACKED_BCD            DB      33H,44H,55H         ;Byte Variable
0041 44
0042 55
0043 20000001                20    POINTER               DD      01000020H           ;Double Word Variable
                             21                                                      ;enth.  BASE:OFFSET
                             22                                                      ;Adresse
0047 ??                      23    BYTE_RESERVIEREN      DB      ?,?,?,?             ;4 Bytes reservieren
0048 ??
0049 ??
004A ??
004B ????                    24    WORD_RESERVIEREN      DW      ?,?,?               ;3 Words reservieren
004D ????
004F ????
                             25
----                         26    DATEN_PROG ENDS
                             27
                             28            END
```

### 3.2.3 Initialisierung mit Wiederholungswerten

Um mehrere Speichereinheiten mit identischen Inhalten zu spezifizieren, stellt der Assembler das DUP-Statement zur Verfügung.

**Syntaxdiagramm:**

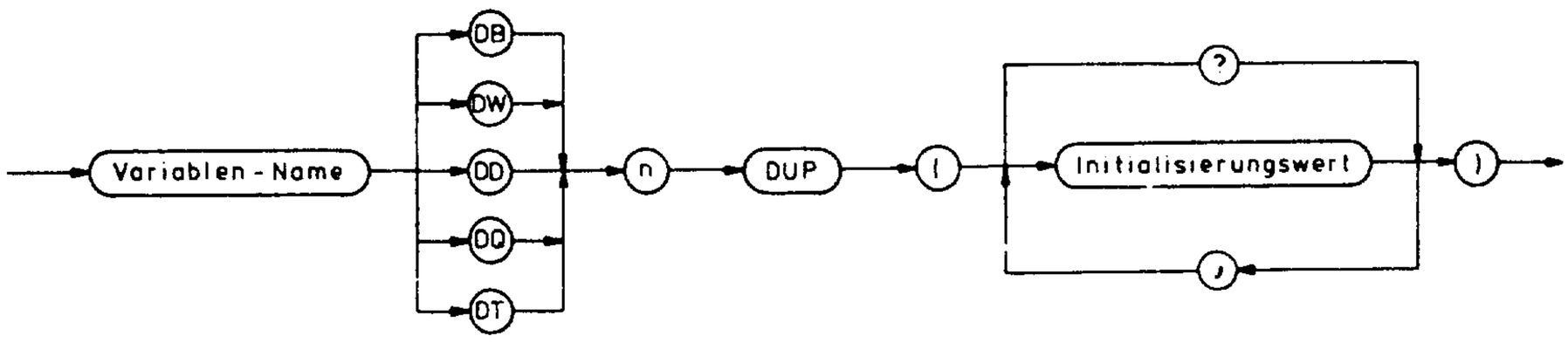



Das DUP-Statement besagt, daß **n** Kopien der in runden Klammern angegebenen Initialisierungswerte hergestellt werden sollen. Dabei kann **n** positive Zahlen im Bereich von 1 bis 65536 annehmen.

- **DUP** ist ein Schlüsselwort und muß angegeben werden.
- **Initialisierungswert** ist der metasprachliche Ausdruck für einen Wert, der n mal dupliziert werden soll.
- Ist ein bestimmter Speicherbereich zu reservieren, können mit Hilfe des Sonderzeichens **?** nichtinitialisierte Speichereinheiten dupliziert werden.

**Hinweis:** DUP-Statements können verschachtelt werden (nested DUP).

**Syntaxdiagramm der wiederholenden Datenzuweisung:**



# Initialisierungsbeispiele mit Übersetzungsprotokoll:

```
LOC  OBJ                  LINE     SOURCE

                             1             NAME    BEISPIEL_6
                             2
                             3     ASSUME DS:DATEN_PROG
                             4
----                         5     DATEN_PROG SEGMENT
                             6
                             7     ;Die folgenden Spezifikationen reservieren Word Variable
                             8     ;mit unbestimmten Werten.
                             9
0000 (2                     10     WORD1           DB          2 DUP(?)
     ??
     )
0002 (1                     11     WORD2           DW          1 DUP(?)
     ????
     )
0004 (1                     12     WORD3           DB          1 DUP(?,?)
     ??
     ??
     )
0006 (1                     13     WORD4           DB          1 DUP(?),1 DUP(?)
     ??
     )
0007 (1
     ??
     )
                            14
                            15     ;Von den angegebenen Strings werden je zwei Kopien angelegt.
                            16
0008 (2                     17     STRING1         DB          2 DUP('HELLO')
     48454C4C4F
     )
0012 (2                     18     STRING2         DB          2 DUP('HELLO','GOODBYE')
     48454C4C4F
     474F4F44425945
     )
                            19
                            20     ;Die folgenden Spezifikationen zeigen verschachtelte
                            21     ;DUP Statements.
                            22
002A (2                     23     LONG_STRING     DB          2 DUP('HELLO',3 DUP('GOODBYE'))
     48454C4C4F
      (3
     474F4F44425945
      )
     )
005E (3                     24     NESTED_DUP      DB          3 DUP(2 DUP(5 DUP(1,4 DUP(0))))
      (2
       (5
     01
        (4
     00
         )
       )
      )
     )
                            25
----                        26     DATEN_PROG ENDS
                            27
                            28             END
```

### 3.2.4 Zusammenfassung

Eine Variable ist eine Dateneinheit mit einem symbolischen Namen. Jede Variable ist mit drei Attributen verbunden:

- Typ-Attribut
- Segment-Attribut
- Offset-Attribut

Der Typ einer Variablen wird durch die spezifizierte Dateneinheit festgelegt (Byte, Word, Double Word, Quad Word oder Ten Byte) D.h. der Typ kann sich als eine bestimmte Anzahl von Bytes in der Variablen gedacht werden.

Das **Segment**-Attribut gibt das logische Segment an, in dem die Variable definiert ist.

Der **Offset** einer Variablen ist der Abstand vom Beginn des Segments, in dem die Variable definiert ist.

Das folgende Beispiel illustriert die Typ-, Segment- und Offset-Attribute einer Variablen.

```
LOC  OBJ                  LINE     SOURCE

                             1            NAME BEISPIEL_7
                             2
                             3     ASSUME DS:DATEN_PROG
                             4
----                         5     DATEN_PROG SEGMENT
                             6
0000 0000                    7     VAR_1           DW      0
0002 (2                      8                     DD      2 DUP(?)
     ????????
     )
000A 06                      9     VAR_2           DB      6,7,8
000B 07
000C 08
                            10
----                        11     DATEN_PROG ENDS
                            12
                            13            END
```

Das Segment enthält die Variablen VAR_1 und VAR_2. Die Variable VAR_01 hat den Typ WORD, da sie mit dem DW-Statement definiert wurde. Ihr Segment-Attribut ist DATEN_PROG, das Segment in dem sie definiert wurde. Ihr Offset ist Null. Das Symbol VAR_1 wird benützt, um das erste Wort im DATEN_PROG-Segment zu kennzeichnen.

Die Variable VAR_2 hat den Typ BYTE und das Segment-Attribut DATEN_PROG. Ihr Offset ist 10, da VAR_2 nach einem Word (2 Bytes) und zwei Double Words (je 4 Bytes) im DATEN_PROG-Segment definiert ist.

Der Initialisierungswert von VAR_2 ist 6. Die anderen beiden Bytes, die mit dem DB-Statement definiert sind, enthalten die Werte 7 und 8. Da sie ohne Namen sind, werden sie, genauso wie die beiden Double Words, die mit dem DW-Statement definiert sind, **nicht** als Assembler-Variable betrachtet.