Bibliothek Sprintf

Die Bibliothek stellt einen Ersatz für die Funktion „sprintf“ der „C“-Bibliothek stdio dar. Die Funktion „sprintf“ belegt einen String. Der in „format“ angegebene String enthält dabei Text, Platzhalter für als Parameter übergebene Werte sowie Ersatzdarstellungen für Sonderzeichen.

Prototyp der „C“- Funktion:

int sprintf(char **buffer*, const char **format*, ...);

Beispiel

sprintf(Puffer,“James Bond %3i“,7); Ergibt: “James Bond 007”

Eine nicht genau definierte Parameterübergabe wird vom PVI und damit auch von Automation-Basic aus zwei Gründen nicht unterstützt:

1. Beliebige Anzahl von Parameter
2. Nicht definierter Parametertyp

Die Portierung wurde wie folgt realisiert:

1. Mehrere Funktionen mit verschiedener Anzahl von Parametern.
2. Zusätzliche Funktion, die die Adresse eines Parameterfeldes akzeptiert.
3. Durch spezielle Syntax im Formatstring kann die Adresse eines Wertes übergeben werden.

Bei der Umsetzung wurde auch die Syntax des Format-Strings modifiziert.

Syntax des Formatstrings

```
+--------  Jeder Platzhalter beginnt mit dem Zeichen %
|+-------  ,+' Vorzeichen erzwingen bei Zahlen, ,-' rechtsbündig bei Strings
||+------  Zahl mit Nullen Füllen  %03i -> 001
|||+-----  Anzahl Gesamtstellen, bei * ist die Anzahl der Wert des nächsten Parameters.
||||+----  Trennung Gesamt-/Nachkommastellen, muß immer Punkt sein.
|||||+---  Keine nicht benötigten Nachkommastellen
||||||+--  Anzahl Nachkommastellen
|||||||+-  @ bedeutet Parameter ist ein Zeiger auf den benötigten Wert
||||||||+  Konverterzeichen
|||||||||
%+03.-4@f
```

Konverterzeichen

i Dezimalzahl, Integer, 32 Bit
unterstützt: Vorzeichen, Gesamtstellen, Nachkomma, Nullfüllen, keine nicht benötigten Nachkommastellen, Zeiger.

u Dezimalzahl, unsigned Integer, 32 Bit
unterstützt: Gesamtstellen, Nachkomma, Nullfüllen, keine nicht benötigten Nachkommastellen, Zeiger

f Gleitkommazahl, float
unterstützt: Vorzeichen, Gesamtstellen, Nachkomma, Nullfüllen, keine nicht benötigten Nachkommastellen, Zeiger
Einschränkungen: Kommastellen müssen fest vorgegeben werden! Keine Exponentendarstellung! Maximal 18 Ziffern! (z.B. 1234567890.12345678)

d Gleitkommazahl, double
unterstützt: Vorzeichen, Gesamtstellen, Nachkomma, Nullfüllen, Keine nicht benötigten Nachkommastellen, Zeiger
Einschränkungen: Kommastellen müssen fest vorgegeben werden! Keine Exponentendarstellung! Maximal 18 Ziffern! (z.B. 1234567890.12345678)

b Binärzahlen, unsigned Integer, 32 Bit
unterstützt: Gesamtstellen, Zeiger

x Hexadezimalzahlen, unsigned Integer, 32 Bit, kleine Buchstaben
unterstützt: Gesamtstellen, Zeiger

X Hexadezimalzahlen, unsigned Integer, 32 Bit, große Buchstaben
unterstützt: Gesamtstellen, Zeiger

z Zahlen mit Basis 36 (Ziffern 0-9,a-z), unsigned Integer, 32 Bit, kleine Buchstaben
unterstützt: Gesamtstellen, Nachkomma, Zeiger

Z Zahlen mit Basis 36 (Ziffern 0-9,A-Z), unsigned Integer, 32 Bit, große Buchstaben
unterstützt: Gesamtstellen, Nachkomma, Zeiger

s String
unterstützt: Gesamtstellen, rechtsbündig

c Zeichen
unterstützt: -

t,T Text aus Liste. Liste ist String, Texte durch „|“ getrennt.
unterstützt: Gesamtstellen ist Textnummer, * ist sinnvoll!
Beispiel sprintf2("%*t",Button,"Ok|Abbruch");

[Bedingte Ausgabe, wenn Parameter = 1

] Ende bedingte Ausgabe

Ersatzdarstellung für Sonderzeichen

%% wird gewandelt zu %

\b -> bs \e -> esc \f -> ff \n -> cr + lf
\xx -> Zeichen mit Hexcode xx ausgeben
Wegen eingebauter Konvertierung des Kompilers muß
„\“ durch „\\“ ersetzt werden!

Funktionen

Funktionen für ASCII:

1 Parameter
```
UDINT sprintf1( UDINT ziel,
                UDINT format,
                UDINT p1);
```
2 Parameter
```
UDINT sprintf2( UDINT ziel,
                UDINT format,
                UDINT p1,
                UDINT p2);
```
4 Parameter
```
UDINT sprintf4( UDINT ziel,
                UDINT format,
                UDINT p1,
                UDINT p2,
                UDINT p3,
                UDINT p4);
```
8 Parameter
```
UDINT sprintf8( UDINT ziel,
                UDINT format,
                UDINT p1,
                UDINT p2,
                UDINT p3,
                UDINT p4,
                UDINT p5,
                UDINT p6,
                UDINT p7,
                UDINT p8);
```

Adresse eines Parameterfeldes
```
UDINT sprintfar(UDINT ziel,
```

```
                UDINT format,
                UDINT parameterliste);
```

```
UDINT skipwhitespace(UDINT udiStartadresse);
```
Liefert die Adresse des ersten nicht „Whitespace“-Zeichens (Blank, Tab, cr, lf, ff) nach der Startadresse

```
DINT strncpyz(  UDINT udiZieladresse,
                UDINT udiQuelladresse,
                UDINT udiAnzahlZeichen);
```
Kopiert eine begrenzte Anzahl Zeichen von Quelladresse nach Zieladresse. D.h. Kein Programmabsturz, wenn das Endezeichen im Quellstring fehlt. An den Zielstring wird immer ein 0-Endezeichen angefügt, auch wenn die Zeichenbegrenzung eintritt.

```
UDINT trim( UDINT udiZieladresse,
            UDINT udiQuelladresse);
```
Entfernt Whitespaces, d.h. Leerzeichen, Tabulatoren, Carridge-Return und Linefeed vom Stringanfang und Stringende.
Das Ergebnis wird in den String bei der mit `udiZieladresse` übergebenen Adresse eingetragen. Ist `udiZieladresse` gleich 0 wird die mit `udiQuelladresse` übergebenen Adresse auch als Zieladresse verwendet.
Rückgabewert ist die Adresse des Zielstrings.

Funktionen für UNICODE

Die Funktionen für UNICODE sind durch ein vorangestelltes „w“ gekennzeichet.
Alle Strings werden als UNICODE betrachtet. D.h. bei den wsprintf-Funktionen ist der Ziel- und der Formatstring UNICODE. Eine Mischung ist nicht möglich.

```
UDINT wsprintf1(UDINT ziel,
                UDINT format,
                UDINT p1);
UDINT wsprintf2(UDINT ziel,
                UDINT format,
                UDINT p1,
                UDINT p2);
UDINT wsprintf4(UDINT ziel,
                UDINT format,
                UDINT p1,
                UDINT p2,
                UDINT p3,
                UDINT p4);
UDINT wsprintf8(    UDINT ziel,
                UDINT format,
```

```
                UDINT p1,
                UDINT p2,
                UDINT p3,
                UDINT p4,
                UDINT p5,
                UDINT p6,
                UDINT p7,
                UDINT p8);
UDINT wsprintfar(UDINT ziel,
                UDINT format,
                UDINT parameterliste);
UDINT wskipwhitespace(UDINT udiStartadresse);
DINT  wstrncpyz(UDINT udiZieladresse,
                UDINT udiQuelladresse,
                UDINT udiAnzahlZeichen);
UDINT wtrim(UDINT udiZieladresse,
            UDINT udiQuelladresse);
```

Funktionen für die Konvertierung von ASCII in UNICODE und umgekehrt

```
UDINT AsciiInUnicode(   UDINT pUnicode,
                        UDINT pAscii);
```

Wandelt einen ASCII-String in einen UNICODE-String.
In `pUnicode` wird die Adresse des UNICODE-Zielstrings übergeben,
in `pAscii` wird die Adresse des ASCII -Quellstrings übergeben.
Die Adressen dürfen auch gleich sein.

```
UDINT UnicodeInAscii (  UDINT pAscii,
                        UDINT pUnicode);
```

Wandelt einen UNICODE -String in einen ASCII-String.
In `pAscii` wird die Adresse des ASCII-Zielstrings übergeben.
In `pUnicode` wird die Adresse des UNICODE-Quellstrings übergeben,
Die Adressen dürfen auch gleich sein.