Modifications made to the bc$ function.

1. Initial Situation

This situation is applicable for the following Porting Set versions:

BaanIVc:	<= Porting Set 6.1c.06
BaanERP 5.0b:	<= Porting Set 6.2a.03.03
BaanERP 5.0c:	<= Porting Set 7.1b

The bc$ function is programmed in the bshell. So, when something is changed in the bc$ function, it affects the bshell.

Initially bc$ function supports the following arguments:

string bc$(long type, long rows, string barcode(.))

Where:

type:
This points to a shell script that generates the barcode. The script is stored in the barcode directory; this is specified in the printer information file. The script names take the form 'typexx', where xx is a two digit number. For example 'type02'. You specify the two digit number in this argument.

rows/height:
The height of the barcode, as a number of lines.

barcode:
The string that must be converted to a barcode and printed.

This text (retrieved from the BaanERP Programmers Manual) is based on using barcode scripts on an Unix system.

The program filterx.x (where x.x. is the version number, like 6.1) interprets the file containing printdata (mostly a tmp file in $BSE_TMP) , and recognizes a barcode.

When the bc$ function is used, the bshell translates this to an escape sequence. This sequence is read and interpreted by the filter program.

bc$(long type, long rows, string barcode(.)	Escape sequence
bc$(type, rows, barcode)	1B 16 2x 2y 2z ss ss ss ss ss ss ss ss .. ..

Explanation:

The escape sequence is built up as follows:

1B 16: indicates this a barcode which has to be printed
2x: this is 20 hexadecimal + offset x, offset x = type
2y: this is 20 hexadecimal + offset y, offset y = rows (height)
2z: this is 20 hexadecimal + offset z, offset z = number of data-bytes which will follow. For example, in case z=9, 9 databytes will follow. (one byte is displayed as "ss"

So, in case you have: bc$(02, 2, "12345"), the following sequence will be in the file, containing the printdata:

1B 16 22 22 25 31 32 33 34 35

1B 16: decimal: 27 22, indicating that the following bytes is about a barcode
22: decimal: 34 – 32 = 2, indicating type 2.
22: decimal: 34 – 32 = 2, indicating height 2.
25: decimal: 37 – 32 = 5, indicating height 5.
31 – 35: representing the string "12345" (31 hex. = 49 dec., and stands for the character "1")

So, filter detects barcode type 2, and therefor calls the barcode script type02, and as arguments the height and the barcode string are passed.

In a similar way the Windows variant, bwprint.exe, is working. But, instead of barcode scripts, bwprint uses a third party barcode software (barcode control), to generate the barcode image.

But, also bwprint detects in the same way the barcode type, and calls a function from this third-party barcode control, passing the arguments like height and the barcode string.

2. Adding 2 optional options to bc$ functions

Now there came in a request to have the possibility to control the following barcode options:

“calculate checksum” and “show check digit”

These 2 options are supported by the third-party barcode control, which is used by bwprint.

Therefor the bc$() function is extended with 2 optional arguments:

string bc$(long type, long rows, string barcode(.), [calc_chks], [shw_chkdgt])

calc_chks: calculate checksum, values: 0 or 1
shw_chkdgt: show check digit, values: 0 or 1

So, instead of 3 arguments, the bc$ function can have 5 arguments.

But how to “tell” filter or bwprint, that the barcode coming now, has 2 extra arguments?

This is solved as follows:
In case bc$ function is used with 5 arguments, not 1B 16 (27 22) is generated, but **1B 17**.

Example:
When doing: bc$(02, 2, “12345”, 1, 1),

the sequence becomes:

1B 17 22 22 25 21 21 31 32 33 34 35

1B 17: decimal: 27 23, indicating that the following bytes is about a barcode, using the 2 optional arguments.
22: decimal: 34 – 32 = 2, indicating type 2.
22: decimal: 34 – 32 = 2, indicating height 2.
25: decimal: 37 – 32 = 5, indicating height 5.
31 – 35: representing the string “12345” (31 hex. = 49 dec., and stands for the character “1”)

So, now when filter detects 1B 17… sequence, the bytes, containing the options are read, but filter does nothing with it further.
In case of bwprint, the options are also read, and send to the barcode control.
This control generates the barcode image, and uses these options as well.

In case 1B 16… is detected, it is still working in the old way (without these 2 extra options).

This modification of the bc$ function, which affects the bshell, is delivered from the following porting sets:

Modification of bc$, which affects the bshell:
(VHNT4530):
BaanIVc: >= Porting Set 6.1c.06.01
BaanERP 5.0b: >= Porting Set 7.1c.02
BaanERP 5.0c: >= Porting Set 7.1b.01

Modification in bwprint.exe, to interpret and handle this new options correctly:
(VHNT4565)
BaanIVc: >= Porting Set 6.1c.06.01
BaanERP 5.0b: >= Porting Set 7.1c.02
BaanERP 5.0c: >= Porting Set 7.1b.01

Modification in filter, to interpret this new options correctly:
(BDUX5935)
BaanIVc: >= Porting Set 6.1c.06.03 (modified filter already made available earlier, refer to solution 15219)
BaanERP 5.0b: >= Porting Set 7.1c.02
BaanERP 5.0c: >= Porting Set 7.1c.02

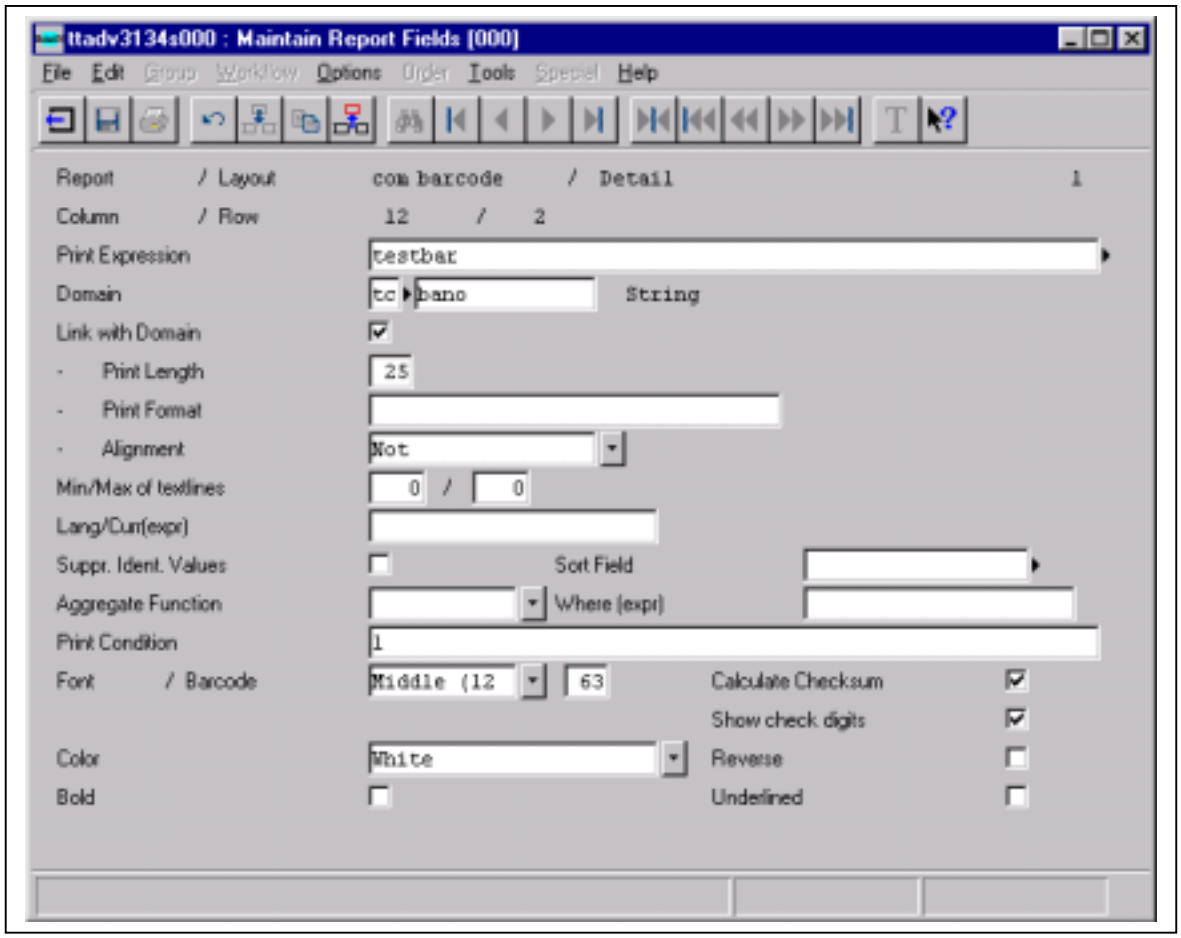
So, now using bc$() function with these 2 extra optional arguments, you can control the “Calculate Checksum” and the “Show Check Digit” features of a barcode.

3. Adding possibility to select new options also via form of “Maintain Report Fields”

There came in also a request to make it possible to select these 2 options via the form of session “Maintain Report Fields”:

BaanIVc: Maintain Report Fields (ttadv3134s000)
BaanERP 5.0b/5.0c: Report Fields (ttadv3534s000)

This is possible now, after installing solution 118107:



Here you see the options “Calculate Checksum” and “Show check digits” are added to the form. (This is the BaanIVc form)

To be able to deliver this modification without making changes to table definition(s) of tools tables, and without modifications in the report generator, these two options are combined with the barcode type.

Because, when we do not combine them, but store these options in a separate table field, a change in a tools table definition is required, as well as the report generator has to read the values from this table field.

This is solved by combining the barcode options with the barcode type.
Where :

bit 6: 01000000 = calculate checksum
bit 7: 10000000 = show check digit

and the other 6 bits (0011 1111) are reserved for the type.

So, when you select on the form of Maintain Report Fields, barcode type 37, and "calculate checksum" = 1, and "show check digits" = 0, the barcode type byte we look like:

0101 0011

this, is hexadecimal: A5 (dec. 165).

And to tell filter or bwprint, that now the barcode options are in the barcode type byte, the barcode is introduced by **1B 18**

So, the sequence for

bc$(37, 2, "12345", 1, 0) will be:

1B 18 A5 22 25 31 32 33 34 35

The disadvantage of this method is:
The barcode type can be maximal 0011 1111, this is decimal 63 !!!

So, in case your barcode type is 65, this is written as: 0100 0001.
And according the new method, this is interpreted as barcode type 1, having "show check digits" enabled. (and on Unix, will call barcode script type01).

Summarizing, this modification means:

- When specifying the barcode type via the form of Maintain Report Fields, the options (also when they are not selected, and therefor they are 0), are combined with the barcode type argument. So the barcode type may not be bigger then 63.
- When specifying the barcode using the bc$ function as follows: bc$(barcode type, height, barcode string), so without using these 2 extra options, it means the 2 extra options are set per default to 0, but still combined with the barcode type. So, also in this case the barcode type is limited to 63.
- When specifying the barcode using the bc$ function with the 2 extra arguments: bc$(barcode, type, barcode string, calc_chksm, shw_chkdgt), still the old method is used (sequence 1B 17), and the options are not mixed with the type. In this way you can types bigger then 63.
- It is also possible to specify the sequence like "1B 16 2x 2y 2z ss sss" as a string in a report field, using chr$() function. Then it depends on if you are using 1B 16, 1B 17 or 1B 18.

This last modification is delivered with the following porting sets:

(BDUX5527)
Modification in bc$ function, which affects the bshell:
BaanIVc: >= Porting Set 6.1c.06.02
BaanERP 5.0b/5.0c: >= Porting Set 7.1c.02

Modification in bwprint.exe, to interpret and handle combination of barcode type & options in 1 byte, correctly:
(BDNT5557)
BaanIVc: >= Porting Set 6.1c.06.02
BaanERP 5.0b: >= Porting Set 7.1c.02
BaanERP 5.0c: >= Porting Set 7.1c.02

Modification in bwprint.exe, to interpret and handle combination of barcode type & options in 1 byte, correctly:
(BDUX5935)
BaanIVc: >= Porting Set 6.1c.06.03 (modified filter already made available earlier, refer to solution 15219)
BaanERP 5.0b: >= Porting Set 7.1c.02
BaanERP 5.0c: >= Porting Set 7.1c.02

4. Final desired situation

The final desired situation would be, that the barcode options are stored in a separate byte.

The advantages of this method are:

- Barcode type not limited anymore to 63, but to 255
- Number of barcode options easily extendable
- Consistent approach: when using bc$ function with 3 arguments either with 5 arguments, the barcode type not limited anymore to 63.

The disadvantages of this method are:

- Changes in tools table definition (Report Fields) required, to store the options in a separate field. Only possible to deliver with new Tools version.
- Changes in the report compiler/generator required.

This final desired situation, will be an enhancement request for future releases. Refer to defect 117885.

Tuesday, October 16, 2001
Koen van den Dool
Baan Customer Services & Support
PEG Tools.