EASYPTR Manual Updates

APPMAN

The QLiberator compiler instruction for EASYAPP
files is:

```
 REMark $$asmb=?,0,60
```

EASYEXT

EASYEXT is now V3.02
EASYEXT_txt gives a full description

EASYMENU

Version 3.07

Info and Application windows with border can not be set larger
than the main window in the list menu.

EASYMEN/PTRMEN Extensions

Version 3.50

New function MCALLT added, which needs two additional parameters for timeout
job events. Otherwise the sysntax is the same as for MCALL

```
item=MCALLT #ch%,eve%,time% [,...
```

eve% and time% are obligatory, i.e. they must be present. eve% must be given as
a name, not only as value, as the event which caused the return will be returned
in eve%. eve% can be used to pass both window events and job events to force a
return. The job events are masked into the upper nibble of eve%.

```
eve%    event to return
 = winevevents+jobevents*256
time%   timeout to return
```

MCALLT only works with SMSQ/E from V2.84 or WMAN from V1.52.

ATTENTION:

```
*******************************************************
*                                                     *
* !!!!! MCALLT does not make any version tests  !!!!! *
*                                                     *
*******************************************************
```

Version 3.07

MITEM redraws the correct item (not the next one)

Version 3.06
++++++++++++

Cosmetics

Version 3.05
++++++++++++

Changes made for SMSQ (MCALL works in SBASIC job 0)

Version 3.04
++++++++++++

MWINDOW with a backslash after the channel number, just sets the window, i.e. the contents is not affected.
If the menu element number is given as 0, the window is set to the main menu window size within the main window border.

```
e.g.    MWINDOW #3\1
```

sets the window to application sub-window no. 1, the window contents and all window attributes (contents, colour, ink, over status, etc.) are unchanged.

```
e.g.    MWINDOW #3\0
```

sets the main window area

WSARS   now accepts the backslash separator to preserve the save area at the right place behind the address parameter as described in the manual.
(the previous version expected the backslash before the address parameter)

****

From Version 2.06 two new commands are available:

MOBJA
+++++

```
adr=MOBJA ([#ch%][{,}{\}]{num}{info%,inob%})
```

Parameters as for MTEXT$

```
separator       ,       object address is returned
                \       item address is returned
```

This function gives the real address of the object of a loose menu item or an info object element.
By this, an object set for an item can be used by the programm and must not be present twice.

For specialists:
++++++++++++++++

With a \ separator, the element address is returned. If the structure (QPTR technical desription) of the element is known, direct changes are possible.
This should be used with great care, if at all!

Remark: With MWDEF the address of the menu working definition can be obtained.
Thus it it possible to get direct access to the complete menu definition.
This should be used with great care, if at all!

MAWBAR
xxxxxx

The implementation of new command MAWBARR (see below) caused a change to the syntax.

The special item number returned by MCALL on an action on the bars has been changed. The upper word of the MCALL function value (the lower word is the application sub-window number as before) now gives an operation code and the pixel position of the HIT masked as follows:

bits	0 - 3	operation
bits	4 - 15	pixel position

bits 0 - 3 set mean:

bit	0	section 1
bit	1	join
bit	2	PAN
bit	3	DO

The evaluation can be made with MAWNUM, e.g.

```
action=MCALL (#ch%) winum=action
mpnum=MAWNUM(#ch%,winum,x_st%,y_st%)
```

Then, if winum matches the application window number where the bars where installed with MAWBAR, mpnum is the operation code, x_st% gives the pixel position of the HIT on the bar while y_st% holds the length of the bar.
If mpnum==0, x_st%==0 and y_st%==0, i.e. the special item number is 0, then the window itself has been hit.

IF x_st% and y_st% are not present in the MAWNUM call,
mpnum=MAWNUM(#ch%,winum)

then mpnum is the full item number with the operation code and the pixel position masked as described above.

Where   op_code=mpnum && 15 and pix_pos=INT(mpnum/16)
will give the values.

The operation code in bits 0 - 3 gives the following meanings:

```
op_code=0       HIT on SCROLL bar in section 0
        1       HIT on SCROLL bar in section 1
                (if split)
        8       DO on SCROLL bar, split
       10       DO on SCROLL bar in section 0, join
       11       DO on SCROLL bar in section 1, join


op_code=4       HIT on PAN bar in section 0
        5       HIT on PAN bar in section 1
                (if split)
       12       DO on PAN bar, split
       14       DO on PAN bar in section 0, join
       15       DO on PAN bar in section 1, join
```

MAWBARR
+++++++

As MAWBAR and additionally the arrows are drawn. The syntax is the same as for MAWBAR, but no split is possible.

The special item number returned by MCALL is as described above, if an action occured on the bars. If the arrows have been hit, then the pixel position masked in bits 4 - 15 is set to -1 (all bits set), and the operation code masked in bits 0 - 3 is as follows:

bit 0	always 0
bit 1	scroll down or pan right
bit 2	pan
bit 3	DO

This gives the following meanings:

op_code=0	HIT up
2	HIT down
4	HIT left
6	HIT right
8	DO up
10	DO down
12	DO left
14	DO right

If MCALL returned from an action in an application sub-window where bars and arrows are set up with MAWBARR, the an immediately following MWINDOW called with the MCALL function value, will set the window size to the area within the arrows.

END
***