MANIFEST ! Plain SuperBasic (PSB) had it all !

Here, as an example, my “generic” boot file (derived from an initial ”spaghetti” boot that was developped and increased progressively, as I stepped from MDVs to SuperQboard with FLPs and finally to Tetroïd CF card).
.
You surely know that plain SuperBasic allows and understands three kinds of statemernts :
1/ statements without line numbers that are interpreted and executed immediately as commands ; for instance the first and last lines of this BOOT file.
2/ statements with line numbers outside of any PROCedure or FuNction, that are executed only once the LOAD (or LRUN) or MERGE (or MRUN) is complete : at line 32741 we have such a statement.
3/ statements with line numbers inside a PROCedure or FuNction : these are executed only when the PROCedure or FuNction is called. In this BOOT file, there is one PROCedure called BYE and two FuNctions called ALN and MWOLN.

```
DEV$="FLP1_" : MODE 4 : PROG_USE DEV$ : DATA_USE DEV$

32741 MRUN mdv1_TOOB

32742 DEFine FuNction aln(f$,f,d)
32743 input_file$  = dev$ & f$ & "_txt"
32744 output_file$ = dev$ & f$ & "_bas"
32745 OPEN_IN#5,input_file$:OPEN_OVER#6,output_file$
32746 l = f
32747 REPeat number
32748    INPUT#5,l$ : PRINT#6,IDEC$(l,5,0);" ";l$
32749    IF EOF(#5) THEN EXIT number : ELSE l=l+d
32750 END REPeat number
32751 CLOSE#6:CLOSE#5:RETurn l
32752 END DEFine aln

32753 DEFine FuNction mwoln(f$,ff,d)
32754 ll=aln(f$,ff,d): ff$ = dev$ & f$ & "_bas"
32755 MERGE ff$ : RETurn ll
32756 END DEFine mwoln

32757 DEFine PROCedure BYE
32758 PAPER#3,0:INK#3,4:CSIZE#3,0,0:CLS#3:AT#3,0,7
32759 PRINT#3,' F1',,'F2',,'F3',,'F4',,'F5':RESTORE OK
32760 FOR i = 5 TO 69 STEP 16
32761    READ p,q,p$,Q$:PAPER#3,p:INK#3,q
32762    AT#3,1,i:PRINT#3,p$:AT#3,2,i:PRINT#3,Q$
32763 END FOR i
32764 p = CODE(INKEY$(-1))/4
32765 IF p<58 OR p>62 : GO TO 32764
32766 CLS#0:CLS:CLS#2:GO TO p+OK-53
32767 END DEFine BYE

PRINT aln("MANIFEST",1,1)
MRUN mdv1_MANIFEST_bas
```

What are these two FuNctions for ? Here a more sensible version of them :

```
DEFine FuNction add_line_numbers(f$,first,interval)
input_file$  = dev$ & f$ & "_txt"
output_file$ = dev$ & f$ & "_bas"
OPEN_IN#5,input_file$
OPEN_OVER#6,output_file$
l = first
REPeat number
   INPUT#5,l$ : PRINT#6,IDEC$(l,5,0);" ";l$
   IF EOF(#5) THEN EXIT number : ELSE l = l+interval
END REPeat number
CLOSE#6 : CLOSE#5 : RETurn l
END DEFine add_line_numbers
```

```
DEFine FuNction remove_line_numbers(f$,column,suffix)
input_file$  = f$
output_file$ = f$(1 to (len(f$)-suffix)) & "_txt"
OPEN_IN#5,input_file$
OPEN_OVER#6,output_file$
l = 1
REPeat number
   INPUT#5,l$ : PRINT#6,l$(column to)
   IF EOF(#5) THEN EXIT number : ELSE l=l+1
END REPeat number
CLOSE#6 : CLOSE#5 : RETurn l
END DEFine remove_line_numbers

DEFine FuNction merge_without_line_numbers(f$,first,interval)
ll = add_line_numbers(f$,first,interval)
MERGE dev$ & f$ & "_bas" : RETurn ll
END DEFine merge_without_line_numbers
```

The FuNction ADD_LINE_NUMBERS needs three parameters :
1/ a file name without any device (prefix provided by DEV$) nor type suffixe (that actually should be _txt)
2/ a value for the first line number to add
3/ a value for the step between successive line numbers
and will output a file with line numbers, called DEV$ & “f$” & “_bas” and RETURN the last line number.
To be noted that blank lines are allowed in the input _txt file : they will become blank numbered lines and when LOADed or MERGEd, will simply be ignored by SuperBasic.

The FuNction REMOVE_LINE_NUMBERS will do the opposite ; it needs also three parameters :
1/ a complete file name, with a device and eventually a type suffixe.
2/ number of the column where the text begins : ALN would format _BAS files with 5 digits for the line numbers so that this parameter should then be 7 (reminding the old times of FORTRAN punched cards where you had 5 caracters for the label field, and one column for an extension mark)
3/ the number of characters of an eventual type suffixe to be replaced by suffixe _TXT.

The function MERGE_WITHOUT_LINE_NUMBERS is an extension of ADD_LINE_NUMBERS that will automatically merge a file without line numbers. It needs the same parameters, and produces a _bas file that could be kept and used separately. ADD_LINE_NUMBERS(“AboveFile”,1,1) will get you this numbered file :

```
 1 DEFine FuNction add_line_numbers(f$,first,ijnterval)
 2 input_file$  = dev$ & f$ & "_txt"
 3 output_file$ = dev$ & f$ & "_bas"
 4 OPEN_IN#5,input_file$
 5 OPEN_OVER#6,output_file$
 6 l = first
 7 REPeat number
 8    INPUT#5,l$ : PRINT#6,IDEC$(l,5,0);" ";l$
 9    IF EOF(#5) THEN EXIT number : ELSE l = l+interval
10 END REPeat number
11 CLOSE#6 : CLOSE#5 : RETurn l
12 END DEFine add_line_numbers
13
14 DEFine FuNction remove_line_numbers(f$,column,suffix)
15 input_file$  = f$
16 output_file$ = f$(1 to (len(f$)-suffix)) & "_txt"
17 OPEN_IN#5,input_file$
18 OPEN_OVER#6,output_file$
19 l = 1
20 REPeat number
21    INPUT#5,l$ : PRINT#6,l$(column to)
22    IF EOF(#5) THEN EXIT number : ELSE l=l+1
23 END REPeat number
24 CLOSE#6 : CLOSE#5 : RETurn l
25 END DEFine remove_line_numbers
26
```

```
 27 DEFine FuNction merge_without_line_numbers(f$,first,interval)
 28 ll = add_line_numbers(f$,first,interval)
 29 MERGE dev$ & f$ & "_bas" : RETurn ll
 30 END DEFine merge_without_line_numbers
```

You may add numbers to any unnumbered text file again and again by changing the parameters and/or modifying it, the only _bas file that will be finally kept is the last one.

Now back to my generic BOOT file, why is it “generic”. ?

Because the two last lines will ADD_LINE_NUMBERS and MRUN a MANIFEST file to specialize it :

```
LRESPR 'mdv1_ptr_gen'
LRESPR 'mdv1_MacMouse11'
LRESPR 'mdv1_wman'
rem LRESPR 'mdv1_sys_hot_rext'
LRESPR 'mdv1_Qptr'
rem LRESPR 'mdv1_sys_Qpac2'
rem LRESPR 'mdv1_fileinfo2_bin'

window#0,512,256,0,0:paper#0,0;:ink#0,0;: cls#0
CLS#0:WINDOW#0,388,42,110,0:PAPER#0,2;:INK#0,7:BORDER#0,1,7
WINDOW#1,94,225,14,0:PAPER#1,0;:INK#1,7:BORDER#1,1,7
WINDOW#2,388,182,110,43:PAPER#2,0;:INK#2,4:BORDER#2,1,7
OPEN#3,'scr_512x31a0x225':PAPER#3,0;:INK#3,4

CFO=32601
R = ALN("CFORMS",CFO,1)
TMO = R+1
R = ALN("TOP_MENU",TMO,1)
PMO = R+1
R = aln("PSION_MENU",PMO,1)
LMO = R+1
R = ALN("LANG_MENU",LMO,1)
CMO = R+1
R = ALN("C68_MENU",CMO,1)

MERGE mdv1_C68_MENU_bas
MERGE mdv1_LANG_MENU_bas
MERGE mdv1_PSION_MENU_bas
MERGE mdv1_TOP_MENU_bas
MERGE mdv1_CFORMS_bas

INITFORM
OK = TMO : KO = CFO-1 : BYE
```

It is transformed before MRUNning it into a _bas file that has only numbered lines outside PROCedures or FuNctions ! So that they will be executed immediately when their MERGEing is finished.

However, statement 32741 of BOOT will execute too and erase this file (that is thus executed only once).

```
  1 LRESPR 'mdv1_ptr_gen'
  2 LRESPR 'mdv1_MacMouse11'
  3 LRESPR 'mdv1_wman'
  4 rem LRESPR 'mdv1_sys_hot_rext'
  5 LRESPR 'mdv1_Qptr'
  6 rem LRESPR 'mdv1_sys_Qpac2'
  7 rem LRESPR 'mdv1_fileinfo2_bin'
  8
  9 window#0,512,256,0,0:paper#0,0;:ink#0,0;: cls#0
 10 CLS#0:WINDOW#0,388,42,110,0:PAPER#0,2;:INK#0,7:BORDER#0,1,7
 11 WINDOW#1,94,225,14,0:PAPER#1,0;:INK#1,7:BORDER#1,1,7
 12 WINDOW#2,388,182,110,43:PAPER#2,0;:INK#2,4:BORDER#2,1,7
 13 OPEN#3,'scr_512x31a0x225':PAPER#3,0;:INK#3,4
 14
```

```
15 CFO=32601
16 R = ALN("CFORMS",CFO,1)
17 TMO = R+1
18 R = ALN("TOP_MENU",TMO,1)
19 PMO = R+1
20 R = aln("PSION_MENU",PMO,1)
21 LMO = R+1
22 R = ALN("LANG_MENU",LMO,1)
23 CMO = R+1
24 R = ALN("C68_MENU",CMO,1)
25
26 MERGE mdv1_C68_MENU_bas
27 MERGE mdv1_LANG_MENU_bas
28 MERGE mdv1_PSION_MENU_bas
29 MERGE mdv1_TOP_MENU_bas
30 MERGE mdv1_CFORMS_bas
31
32 INITFORM
33 OK = TMO : KO = CFO-1 : BYE
```

When the MERGEs of this file are done, BOOT will be able to execute the BYE PROCedure with OK set to TMO and show this screen presenting the TOP_MENU :

The INK and PAPER of the function key labels are free ; my choice is “purple” for the sub menus firery red for programs and red lined for going up the menu tree ; going up off top menu means accessing the SuperBasic console. The file for TOP_MENU is thus this one :

```
DATA   208,7,'   PSION   ','   SUITE   '
DATA   208,7,'  PROLOG   ',' and FORTH '
DATA   208,7,'    C68    ','programming'
DATA     2,7,'   QJUMP   ','  Q R A M  '
DATA    80,7,'   SUPER   ','   BASIC   '
OK=PMO : PAPER#2,0:INK#2,4:CLS#2 : bye
OK=LMO : PAPER#2,0:INK#2,4:CLS#2 : bye
OK=CMO : PAPER#2,0:INK#2,4:CLS#2 : C68 : bye
EXEC DEV$ & 'QRAM' : bye
CLS:CLS#0:PRINT#0,'Say BYE to quit SUPER BASIC':STOP
```

As you can see, each menu file has exactly ten statement lines : the first five define the key labels and the following five define the behaviour behind the function keys when pressed. All this is managed by the BYE PROCedure. A difficulty when MERGEing unnumbered text files is the localisation of the DATA statements when RESTOREing before READing them. This was solved by introducing the variables TMO PMO LMO and CMO values by MANIFEST, and grouping the five DATA statements at the beginning of the menu files.

There is another unnumbered text file with the same issue : CFORMS provides a simple graphical interface for the C68 compiler and loader :


```
DATA 18,4,0,2,0,2,2,0,7,7,12
DATA 12,2,"3K"," -=3072",5,18,2
DATA 12,2,"5K"," -=5120",5,18,3
DATA 12,2,"7K"," -=7168",5,18,4
DATA 12,2,"9K"," -=9216",5,18,1
DATA 24,2,"VERBOSE"," -v",7,1,6
DATA 24,2,"LACONIC","",7,1,5
DATA 41,2,"ANSI"," -unproto",9,5,8
DATA 41,2,"K&R ","",9,5,7
DATA 53,2,"16 bits"," -Qshort",13,7,10
DATA 53,2,"32 bits","",13,7,9
DATA 30,7,""," flp1_mydir_",12,14,4
DATA 30,8,""," myprog_c",15,11,1
DATA 30,4,""," -Iflp1_myINCs_",14,9,2
DATA 30,5,""," -Lflp1_myLIBs_",11,13,3
DATA 10,12,"Floating point & Maths flp1_LIB_LIBM_a"," -lm",16,12,15
DATA 10,13,"Dynamic allocations flp1_LIB_LIBMALLOC_a"," -lmalloc",17,15,16
DATA 10,14,"debug support flp1_LIB_LIBDEBUG_a"," -ldebug",18,16,17
DATA 10,15,"Semaphores and tasking flp1_LIB_LIBSEM_a"," -lsem",1,17,18
DATA 0,1,0,0,1,0,0,1,0,1,1,1,1,1,0,0,0,0,11
DATA 1,1,0,4,"COMPILE AND LINK OPTIONS :"
DATA 4,2,0,4,"Stack : Mode : Norm : Int :"
DATA 0,3,0,7,"-----------------------------------------------------------------"
DATA 7,4,0,4,"My Include Directory :"
DATA 7,5,0,4,"My Library Directory :"
DATA 0,6,0,7,"-----------------------------------------------------------------"
DATA 7,7,0,4,"My Project Directory :"
DATA 7,8,0,4,"My Current File Name :"
DATA 0,9,0,7,"-----------------------------------------------------------------"
DATA 1,10,0,4,"Standard Libraries to scan :"
DATA 10,11,0,7,"Standard C LIBRARY flp1_LIB_LIBC_a"
DEFine PROCedure INITFORM
o=CFO:RESTORE o:READ m:DIM f(m):READ n:DIM r$(n,32)
DIM CC(7):FOR i=0 TO 7:READ CC(i):NEXT i:READ cp
RESTORE o+m+1:FOR i=1 TO m:READ f(i):NEXT i:READ MMM
FOR i=1 TO m:READFLD(o+i):IF m$="":r$(jt)=n$
END DEFine
```

Same file to be continued on next page without any blank lines !

```
DEFine PROCedure EDITFORM
o=CFO:CLS#2:RESTORE o+m+2:p$=""
FOR i=1 TO MMM:READ x,y,jp,jt,m$:PAPER#2,jp:INK#2,jt:AT#2,y,x:PRINT#2,m$
c=cp:READFLD(c+o)
c=NEXTFLD(c):IF c<>cp:GO TO 32642
REPeat ScanKeyb
a=CODE(INKEY$(-1))
SELect ON a
ON a=208:c=PREVFLD(c)
ON a=216:c=NEXTFLD(c)
ON a=32:c=TOGGLE(c)
ON a=10:DISPFLD c,0:EXIT ScanKeyb
ON a=27:C68:bye
END SELect
END REPeat ScanKeyb
FOR i=1 TO m
READFLD(o+i):IF m$="":n$=r$(jt)
IF ((m$<>"") OR ((jt>1) AND (jt<n))) AND (f(i)=1):p$=p$&n$
END FOR i
dat$=r$(n,4 TO LEN(r$(n))): REM p$ = p$ & " -tmp" & dat$
END DEFine
DEFine FuNction CCC$(opt)
EDITFORM:INK#2,4:CLS#2:nnn="_c"INSTR r$(1)
IF nnn=0:nnn=len(r$(1))+1:r$(1)=r$(1)&"_c"
IF opt=0:p$="-c "& dat$ & r$(1,4 TO nnn+1)&" "&p$
IF opt=1:p$="-o"&dat$&r$(1,4 TO nnn-1)&" "&dat$&"*_o "&p$
EXEC_W CC,#2,#2,#2;p$
PRINT#0," Done... Hit any key !";:return inkey$(-1)
END DEFine
DEFine FuNction TOGGLE(i)
IF m$="" THEN
EDITFLD(jt)
ELSE
IF jt=i :f(i)=1-f(i)
IF jt<>i:f(i)=0:i=jt:f(i)=1:READFLD(i+o)
END IF
DISPFLD i,1:RETurn i
END DEFine
DEFine FuNction PREVFLD(i)
DISPFLD i,0:i=jp:i=FINDFLD(i):DISPFLD i,1:RETurn i
END DEFine
DEFine FuNction NEXTFLD(i)
DISPFLD i,0:i=jn:i=FINDFLD(i):DISPFLD i,1:RETurn i
END DEFine
DEFine PROCedure DISPFLD(i,s)
PAPER#2,CC(2*f(i)+s):INK#2,CC(4+2*f(i)+s)
AT#2,y,x:PRINT#2,m$;:IF m$="":PRINT#2,r$(jt,4 TO)
END DEFine
DEFine PROCedure EDITFLD(ii)
PAPER#2,CC(2*f(i)):INK#2,CC(4+2*f(i))
AT#2,y,x:PRINT#2,FILL$(" ",29)
AT#2,y,x:INPUT#2,O$:r$(ii)=r$(ii,1 TO 3)&O$
END DEFine
DEFine PROCedure READFLD(l)
RESTORE l:READ x:READ y:READ m$:READ n$:READ jn:READ jp:READ jt
END DEFine
DEFine FuNction FINDFLD(i)
READFLD(i+o):IF (jt<>i)AND(f(i)=0):i=jt:GO TO 32695
RETurn i
END DEFine
DEFine PROCedure C68
PAPER#2,0:INK#2,4:CLS#2:VIEW#2,dev$ &'C68_help'
END DEFine C68
MRUN mdv1_TOOB
```

As with the menu files, MANIFEST gives to this file a start number kept in the variable CFO, all the DATA statements are referenced by CFO. Moreover, by reexamining MANIFEST you will notice how these variables let chain the MERGEing naturally, using the number of the last lines RETURNed by ADD_LINE_NUMBERS !

To let you really test this software I have to show you also the three submenus (unnumbered).

File mdv1_PSION_MENU_txt

```
DATA  80,7,'  M A I N  ','  M E N U  '
DATA   2,7,'   PSION   ','   QUILL   '
DATA   2,7,'   PSION   ','  ABACUS   '
DATA   2,7,'   PSION   ','  ARCHIVE  '
DATA   2,7,'   PSION   ','   EASEL   '
OK=TMO : bye : REM back to main menu
EXEC dev$ & 'quill'      : bye
EXEC dev$ & 'abacus'     : bye
EXEC dev$ & 'archive'    : bye
EXEC dev$ & 'easel'      : bye
```

File mdv1_LANG_MENU_txt

```
DATA   2,7,'    QED    ','Text EDIT '
DATA  80,7,'  M A I N  ','  M E N U  '
DATA   2,7,'Edimburgh ','   PROLOG  '
DATA   2,7,'ComputerOne','   FORTH   '
DATA   2,7,'DigitalPrec','Super FORTH'
EXEC dev$ &'qed':bye
OK=TMO : bye : REM back to main menu
EXEC dev$ &'prolog':bye
EXEC dev$ &'forth':bye
EXEC dev$ &'forth83_job':bye
```

File mdv1_C68_MENU_txt

```
DATA   2,7,' C68   QED ','Text EDIT '
DATA   2,7,' C68       ','      MAKE '
DATA  80,7,'  M A I N  ','  M E N U  '
DATA   2,7,' C68       ','   COMPILE '
DATA   2,7,' C68       ','      LINK '
EXEC QED:C68:bye
EXEC_W MAKE,#2,#2,#2;p$ & " -t " & Q$:C68:bye
OK=TMO : CLS#2 : bye : REM back to main menu
CLS#0:PRINT#0,"Compiling...";:PRINT#0,CCC$(0):C68:bye
CLS#0:PRINT#0,"Linking...";:PRINT#0,CCC$(1):C68:bye
```

and a text file called mdv1_C68_HELP

```
When using COMPILE (F2) or LINK (F3), a setup form will replace
this HELP screen. Some  of the options will be set or disabled,
depending on what this program assumes when booting and on what
you have already done since that time. The field you would most
likely modify will be highlighted: that is the filename.

Remember that you can step through the fields using the ¾ and ¿
keys. ENTER key starts compiling or linking, ESC key exits.

The SPACE BAR has different functions depending on the fields:
- sometimes it will let you step through different options.
- sometimes it will let you activate or inhibit an option.
- sometimes it will erase a field, and let you input something.

All this was meant to be intuitive. Feel free to experiment !!!
```

You need also this little mdv1_TOOB file with commands to erase MANIFEST after performing.

```
DLINE 1 to KO
BYE
```