8 Overlaying Procedures

Lack of memory can be a very restrictive and annoying problem with large programs. One way of getting round this is to use several smaller programs, and CHAIN them together (like the 'Welcome' cassette). However, this RUNs each program which is loaded in, so all the variables (apart from the resident integers) are lost.

Another method is to 'overlay' FNs and PROCs. If the program consists of a number of large sections, which will not be in memory at the same time as one another, these sections can be loaded in on top of each other when one is required. Since only one of the sections will be active at any particular time, the same memory can be used for all of them.

By intercepting the 'No such FN/PROC' error, an overlay file can be loaded in, and executed as if it was a normal FN or PROC. When the FN or PROC has finished, the memory that it loaded into is free for another call. This sort of overlaying is more useful on a system with discs, because of its random access ability; but it can be used with cassettes as well if the order in which the overlay files will be required is known (so that they can be saved in that order on the tape).

This chapter describes how to overlay FNs and PROCs, JMPing back in to BASIC to continue when the file has been loaded.

8.1 The 'No such FN/PROC' error

This error (error number 29) is generated by the FN/PROC handler when it failed to find the definition of the FN or PROC in the program. See section 5.3 for the operation of the FN/PROC handler. The sequence of actions taken when the FN/PROC handler comes across an undefined call is as follows:

1 The 6502 stack, from &1FF to the item on top of the stack, is saved on the BASIC STACK. The 6502 stack pointer is saved as the byte on top of the BASIC stack, so that the correct number of bytes can be retrieved after the call. After saving, the 6502 stack pointer is re-set to &1FF.

2 The FN or PROC token is saved as the first item on the 6502 stack, at &1FF, so that ENDPROC or the '=' statement know which type of call they are in. The FN token is &A4, and the PROC token is &F2.

3 PTRA is saved on the 6502 stack, from &1FE to &1FC. The stack pointer now points to &1FB (at the next free byte).

4 If there was no name after the FN/PROC token, a 'Bad call' error is generated. Otherwise, the FN/PROC handler searches through the list of already used FNs or PROCs for the name.

5 If it wasn't found in the list (which it won't be, if it is not in the program), the FN/PROC handler searches through the program for the definition. When it doesn't find it, it restores the base of PTRA from the 6502 stack, so that ERL will be set up properly by the BASIC error handler, and generates a 'No such FN/PROC' error.

When this error ocurrs, the prevailing conditions on entry to the BRK handler are:

&FD,&FE	points to the error number (29)		
6502 stack:	&1FB	RTI info.	3 bytes
	&1FE	PTRA offset	1 byte
	&1FF	FN/PROC token	1 byte

BASIC STACK contains old 6502 stack.

&37,&38	points 1 byte before the FN/PROC token
&39	length of name (+1 for token)

The FN/PROC can be re-entered to force it to use an overlayed file as the FN or PROC it was looking for, but first the 6502 stack must be restored to the state immediately before the error was generated. The 3 bytes of RTI information must be pulled from the stack, and the base of PTRA must be pushed back on (&B first, then &C).

At this point the overlay file can be loaded. When the overlay file is in memory, the FN/PROC handler can be re-entered, as if the overlay is a FN or PROC which it has just found.

To re-enter the FN/PROC handler, set the base of PTRA (in &B,&C) to point to the first character which would be after the name of the FN/PROC in the definition, and JMP to &B223 (BASIC1) or &B1F4 (BASIC2).

Jumping to this address will continue with the FN/PROC handler, and the name will not be added to the list of used FNs or PROCs. If the name had been added to the list, difficulties would arise when the overlay had been finished with; the FN/PROC handler would still think that it knew where the overlayed FN or PROC was, but the memory may have already been used by a different overlay file.

8.2 Static overlaying

A very simple method of overlaying a FN or PROC is to load a file into a fixed position in memory (hence 'static') whenever a 'No such FN/PROC' error is generated.

The routine in this section will load the file 'OVERLAY' into memory at &6000 (this can be changed by altering line 600), and then re-enter the FN/PROC handler to use this file as the FN or PROC which could not be found.

The 'OVERLAY' file should be saved as if it is a normal BASIC program: it should not contain the 'DEF PROCname' (but it must have the 'ENDPROC' or '=' statement). If parameters are to be passed to it, the '(' should be the first character on the first line of the program. For example, the following overlay file will print the SIN of the number passed to it:

```
10(number)
20PRINT SIN(number)
30ENDPROC
```

If this program is saved as the file 'OVERLAY', any unrecognised FN or PROC call will be passed to it. For example, 'PROCFRED(PI/2)' will print '1'.

This overlay routine cannot tell the difference between FNs and PROCs; it will load the file ‘OVERLAY’ whenever the error is generated. So, if the file is saved as above, ‘X=FNA(3)’ will give a ‘No PROC’ error, when it finds the ‘ENDPROC’ statement on the end of what it thinks is a FN.

If the overlay does not need any parameters, the first character on the first line could be the start of the first statement, or a space.

```
   4 REM This is a simpte program to overtay procedures.
   6 REM
   8 REM          M D Plumbley 1984
  10 REM
  12 REM Once this is initilaised, if a FN or PROC is not
  14 REM  found in a program, generating the
  16 REM  "No such FN/PROC" error, then the fite called
  18 REM  "OVERLAY" will be loaded from disc, and
  20 REM  executed.
  22 REM
  24 REM The overlay file shoutd not contain the name of
  26 REM  the PROC or FN, but any parameters should be
  28 REM  inside brackets on the first line of the file.
  30 REM  If used, the open bracket must be the first
  32 REM  character on the first line of the file.
  90 REM
  95
 100 PROCsetup :REM Set up correct ROM entry points
 390
 395 REM *** OS vectors ***
 400 brkv = &0202
 410 oldbrk = !brkv AND &FFFF
 490
 495 REM *** OS routines ***
 500 oscli = &FFF7
 590
 600 ldslot = &6000 :REM Area to load overlay into
 799
 900 start% = &0C00 :REM Assembte into user char space
 905
 910 FOR opt% = 0 TO 3 STEP 3
 920 P% = start%
 950 [OPT opt%
 960
1000 .newbrk
1005     PHA        \Save A and Y on 6502 stack
1010     TYA
1015     PHA
1020
1025     LDY #0     \Get error number
1030     LDA (&FD),Y
```

```
1035
1040     CMP #29    \If "No such FN/PROC", go
1045     BEQ noproc \ to overtay routine.
1050
1055 .giveup        \Otherwise, restore A and Y and go
1060     PLA        \ to the defautt BRK handter.
1065     TAY
1070     PLA
1075     JMP oldbrk
1080
2000 .noproc
2005     PLA        \Remove the saved A and Y from the
2010     PLA        \ 6502 stack.
2015
2020     PLA        \Remove the RTI information from the
2025     PLA        \ 6502 stack.
2030     PLA
2035
2040     LDA &B     \Push the base of PTRA, ready for
2045     PHA        \ the return from the FN/PROC.
2050     LDA 8C
2055     PHA
2060
2065     LDX #lodtxt MOD &100   \Tell the fiting system to
2070     LDY #lodtxt DIV &100   \ load the overlay file
2075     JSR oscli
2080
2085     LDA #ldslot MOD&100+4  \Set PTRA to point to the
2090     STA &B                 \ 1st char of the fite
2095     LDA #ldslot DIV &100   \ (not CR, line num, or
2100     STA 8C                 \ length)
2105
2110     JMP prefnd \Continue with the FN/PROC handter
2115
2120 .lodtxt        \DFS command to load the overtay
2125 ]$P% = "LOAD OVERLAY ":P%=P%+LEN$P%
2130 $P% = STR$~ldslot :P%=P%+LEN$P%
2135 ?P% = &0D :P%=P%+1
2140
3010 @%=0
8000 NEXT
8020 PRINT'"Code length =&"~P%-start%
8030
8040 REM *** Link new routine in to BRK vector ***
8050 IF newbrk=oldbrk PRINT"Already set up":END
8060 brkv?0 = newbrk MOD &100
8070 brkv?1 = newbrk DIV &100
8080 END
8090
9000 REM *** Set up R0M entry points, attowing for ***
9010 REM ***           BASIC1 and BASIC2           ***
9020 DEFPROCsetup
```

```
9030 IF ?&8015=ASC"1" THEN PROCset1 ELSE PROCset2
9040 ENDPROC
9050
9300 REM *** Set up BASIC1 entry points            ***
9310 DEFPROCset1
9320 prefnd = &B223 :REM Return to FN/PROC handler
9330 ENDPROC
9340
9500 REM *** Set up BASIC2 entry points            ***
9510 DEFPROCset2
9520 prefnd = &B1F4 :REM Return to FN/PROC handter
9530 ENDPROC
```

The general operation of the routine is as follows:

1 If the error number is not 29, the default BRK handler is called (lines 1000 to 1080). If the error number is 29, the 3 bytes of RTI information are removed from the stack (as well as the 2 registers saved by the BRK handling routine at 1000 to 1015).

2 The base of PTRA is pushed back on the 6502 stack (lines 2040 to 2055), for the return when the call is finished.

3 The overlay file is loaded by sending the line 'LOAD OVERLAY 6000' to the Operating System Command Line Interpreter (OSCLI). This will be interpreted just as if a '*LOAD' had been typed at the keyboard. Note the use of the hexadecimal version of the STR$ function (line 2130). This is in BASICI and BASIC2, but is not mentioned in the *User Guide*.

4 The base of PTRA is set to point to the fifth character of the file (at &6004). If the file has been entered as a BASIC program, the first character of the file will be a &0D, followed by a 2-byte line number, followed by the line length byte (see section 2.4 for the program storage format).

5 A JMP is made to re-enter the FN/PROC handler. It will then think that the call definition has been found, and that the base of PTRA points to the first charcter after the name in the definition. If this character is a '(', it will handle any parameters which are listed. It will then start executing statements in the file as if it was a proper FN or PROC.

8.3 Dynamic overlaying

The routine in the last section is a bit limited. It can't tell the difference between different FNs or PROCs, as it doesn't do any name checking; and it always loads into the same area of memory (which must be decided when it is assembled), so only one PROC or FN can operate at a time.

The routine in this section shows how FNs and PROCs can be recognised and loaded onto the BASIC STACK, completely invisible to the main program (except for the amount of memory required to load them). If there is not enough memory to load the FN or PROC, a 'No room' error will be generated. FNs and PROCs loaded like this can call others inside them to be overlayed, and these will also be loaded onto the STACK. The program in section 8.2 would just load the other overlay on top of the first one.

The exit from the FN or PROC is trapped by changing the token byte on the 6502 stack at &1FF, so that a 'No FN' or 'No PROC' error will be generated. This allows the overlayed file to be removed from the STACK when it is finished with, by intercepting these errors.

The overlay files are created in the same manner as the ones in section 8.2, with the '(' as the first character on the first line if necessary. However, the routine will check the name of the FN or PROC, and will load in 'P.fred' if 'PROCfred' is called, arid 'F.fred' if 'FNfred' is called. Note that the operating system will treat upper and lower case letters as the same, so 'F.FRED' is the same as 'F.fred' as far is it is concerned.

```
10 REM *** Program to overlay PROCs and FNs **
12 REM
14 REM     M D Plumbley        1984
16 REM
18 REM Once this is run, if a FN or PROC is not found in
20 REM  a program, generating the "No such FNIPROC"
22 REM  error, then the file with the same name
24 REM  as the FN or PROC will be loaded from disc (or
26 REM  tape). The P directory will be used for PROCS,
28 REM  the F directory for FNs.
30 REM
32 REM The FN or PROC will be loaded on the BASIC
```

```
 34 REM  STACK, and will be removed then it exits.
 36 REM
 38 REM The overlay file should not contain the name of
 40 REM the PROC or FN, but any parameters should be
 42 REM inside brackets on the first line of the file.
 44 REM If used, the open bracket must be the first
 46 REM character on the first line of the file.
 48 REM
 50 REM Before using with BASIC 1, all EQU directives
 52 REM shoutd be reptaced by indirections:
 54 REM "EQUB X"  => "]?P%=X:P%=P%+1:[OPTopt%"
 55 REM "EQUW X"  => "]!P%=X:P%=P%+2:[OPTopt%"
 56 REM "EQUD X"  => "]!P%=X:P%=P%+4:[OPTopt%"
 57 REM "EQUS A$" => "]$P%=A$:P%=P%+LEN$P%:[OPTopt%"
 90 REM
 95
100 PROCsetup :REM Set up correct ROM entry points
390
395 REM *** OS vectors ***
400 brkv = &0202
410 oldbrk = !brkv AND &FFFF
490
495 REM *** OS routines ***
500 oscli  = &FFF7
505 osfile = &FFDD
590
690 REM *** BASIC registers ***
700 stack  = &0004
705 inta   = &002A
799
800 parms  = &0070 :REM Temp for number of parameters
899
900 start% = &0B00 :REM User defined character area
905
910 FOR opt% = 0 TO 3 STEP 3
920 P% = start%
950 [OPT opt%
960
1000 .newbrk
1005     PHA             \Save A and Y on 6502 stack
1010     TYA
1015     PHA
1020
1025     LDY #0          \Get error number
1030     LDA (&FD),Y
1035
1040     CMP #29         \If "No such FN/PROC", go
1045     BEQ nofnpr      \ to overtay routine.
1047
1050     CMP #7          \If "No FN" see if it is
1055     BEQ jnofn       \ to be thrown away.
1057
```

```
1060     CMP #13          \If "No PROC" see if it is a PROC
1065     BEQ jnoprc       \ to be thrown away
1070
1075 .ospace
1080 .giveup              \Otherwise, restore A and Y and go
1085     PLA              \ to the defautt BRK handler.
1090     TAY
1095     PLA
1100     JMP oldbrk
1105
1110 .jnofn               \Jump to the "No FN" handler
1115     JMP nofn
1117
1120 .jnoprc              \Jump to the "No PROC" handler
1125     JMP noproc
1127
1990 \ *** If we get here, a FN or PROC is to be       ***
1992 \ *** over layed, after a "No such FN/PROC" error ***
2000 .nofnpr
2005     PLA              \Remove the saved A and Y from the
2010     PLA              \ 6502 stack.
2015
2020     PLA              \Remove the RTI information from the
2025     PLA              \ 6502 stack.
2030     PLA
2035
2040     LDA &B           \Push the base of PTRA, ready for
2045     PHA              \ the return from the FN/PROC.
2050     LDA 8C
2055     PHA
2060
2065     LDY &39          \If the length of the name of the
2070     CPY #9           \ FN/PROC, with the token, is > 8,
2075     BCS giveup       \ it is too big to be a filename.
2080
2085     LDA #&0D         \Put a CR on the end of the
2090     STA filnam+1,Y   \ area, ...
2095
2100 .txnmlp              \ and transfer the name from the
2105     LDA (&37),Y      \ text into the fitename area.
2110     STA filnam,Y
2115     DEY
2120     BNE txnmlp
2125
2130     LDX #ASC"P"      \If the token on the front of the
2135     CMP #&F2         \ name (the last byte transfered)
2140     BEQ proc         \ was a PROC token, put a "P" on
2145     LDX #ASC"F"      \ the front of the filename;
2150 .proc                \ otherwise use an "F".
2155     STX filnam
2160
2165     LDA #ASC"."      \Put a "." between the P/F and the
```

```
2170     STA filnam+1    \ FN/PROC name.
2175
2180     LDX #pblock MOD &100   \Call OSFILE to find
2185     LDY #pblock DIV &100   \ the length of the
2190     LDA #5                 \ file.
2195     JSR osfile
2200
2205     CMP #1          \If it didn't exist, jump to the
2210     BNE giveup      \ default error handter.
2215
2220     LDA stack       \Save the BASIC STACK pointer in
2225     STA inta        \ IntA, and move the STACK pointer
2230     SEC             \ dam ready to load the overlay,
2235     SBC pblock+&0A  \ by subtracting the length of the
2240     STA stack       \ fite from it. The fite length
2245     STA pblock+2    \ is returned by OSFILE 5 in
2250                     \ pblock+&A and pblock+&B.
2255     LDA stack+1     \
2260     STA inta+1      \ A copy of the new stack pointer
2265     SBC pblock+&0B  \ is loaded into pblock+2 and
2270     STA stack+1     \ pblock+3, to tell OSFILE &FF
2275     STA pblock+3    \ where to load the file uhen it
2277                     \ is called.
2280     BCC ospace      \ If the STACK wrapped round,
2282                     \ give an error.
2285
2290     JSR pushi       \Push the old STACK pointer on
2292                     \ the STACK.
2295
2300     LDA #0          \Set the "addr" flag for OSFILE to
2305     STA pblock+6    \ load the file at the given addr
2310
2315     LDX #pblock MOD &100   \Call OSFILE to load
2320     LDY #pblock DIV &100   \ the overlay file into
2325     LDA #&FF               \ the space allocated
2330     JSR osfile             \ on the STACK.
2335
2340     LDA stack       \Set the base of PTRA to point to
2345     CLC             \ the first character in the BASIC
2350     ADC #8          \ file (4 up to miss over IntA,
2355     STA &B          \ and another 4 up to miss the
2360     LDA stack+1     \ &0D, line number, and length
2365     ADC #0          \ byte as before).
2370     STA &C
2375
2380     LDA filnam      \Set the FNIPROC identifier byte
2385     STA &1FF        \ on the stack to a "P" or "F"
2390
2395     JMP prefnd      \Jump into the FN/PROC handler.
2990
3000 .pblock             \OSFILE parameter btock
3005     EQUW filnam
```

```
3010     EQUD 0
3015     EQUD 0
3020     EQUD 0
3025     EQUD 0
3030     EQUB 0
3032
3035 .filnam             \Filename area (max 9 characters)
3040     EQUS "123456789"
3045     EQUB &0D
3990
3992 \ ** No FN error   **
4000 .nofn
4005     LDA &1FF        \If the item on the stack was not
4010     CMP #ASC"F"     \ Left by the overtay routine,
4015     BNE jgivup      \ there isn't a FN on the STACK.
4017
4020     CPX #&F5        \If the 6502 stack pointer wasn't
4025     BNE jgivup      \ &F5, we're not in a FN.
4027
4030     JSR getnsa      \Get the value of the FN follcwing
4035     JSR chksdb      \ the "=", check end of statement,
4040     JMP doret       \ and jump to do the FN return.
4045
4090 \
4100 .jgivup
4105     JMP giveup      \Jump to the old BRK handler
4110
4990                     \ ** No PROC error **
5000 .noproc
5005     LDA &1FF        \If the item on the stack was not
5010     CMP #ASC"P"     \ left by the over lay routine,
5015     BNE jgivup      \ there isn't a PROC on the STACK.
5020
5025     CPX #&F5        \If the 6502 stack pointer wasn't
5030     BNE jgivup      \ &F5, were not in a PROC.
5032
5035     JSR chksda      \Check end of statement after the
5036                     \ "ENDPROC".
5037
5040 .doret
5045     PLA             \Remove the saved A and Y from the
5050     PLA             \ 6502 stack.
5055
5060     PLA             \Remove the RTI information from
5065     PLA             \ the 6502 stack
5070     PLA
5075
5080     PLA             \Remove the return addr to the
5085     PLA             \ FN/PROC handter.
5090
5095     PLA             \Restore PTRB
5100     STA &1A
```

```
5105     PLA
5110     STA &19
5115     PLA
5120     STA &1B
5125
5130     PLA             \If there were no parameters,
5135     BEQ noparm      \ don't restore any.
5140
5145     STA parms       \Otherwise, restore the saved
5150 .doparm             \ vatue of each parameter by
5155     JSR popi1       \ popping the variabte descriptor
5160     JSR poppar      \ block and vatue from the BASIC
5165     DEC parms       \ stack.
5170     BNE doparm
5175
5180 .noparm
5185     PLA             \Restore PTRA
5190     STA &C
5195     PLA
5200     STA 8B
5205     PLA
5210     STA &A
5215
5220     LDY #0          \Restore the BASIC stack pointer
5225     LDA (stack),Y   \ to the value it was before the
5230     TAX             \ FN or PROC was loaded onto it:
5235     INY             \ this had been pushed on the
5240     LDA (stack),Y   \ STACK when the file was loaded.
5245     STX stack
5250     STA stack+1
5253
5255
5260     LDY #0          \Restore the 6502 stack from the
5265     LDA (stack),Y   \ BASIC STACK. The first byte
5270     TAX             \ gives the old value of the 6502
5275     TXS             \ S register, the rest of the
5280 .txstk              \ bytes are the actual stack
5285     INY             \ contents.
5290     INX
5295     LDA (stack),Y
5300     STA &100,X
5305     CPX #&FF
5310     BNE txstk
5315
5320     TYA             \Move the STACK pointer up to
5325     ADC stack       \ remove the 6502 stack contents
5330     STA stack       \ from it.
5335     BCC stkok
5340     INC stack+1
5345 .stkok
5347
5350     LDA &27         \Set the 6502 flags according to
5352                     \ &27 (in case we're in a FN).
```

```
5253
5355     RTS             \Exit
9000 ]
9010 NEXT
9020 @%=0
9030 PRINT'"Code length =&"~P%-start%
9040
9045 REM *** Link new routine in to BRK vector ***
9050 IF newbrk=oldbrk PRINT"Already set up":END
9060 brkv?0 = newbrk MOD &100
9070 brkv?1 = newbrk DIV &100
9075 END
9080
9500 REM *** Set up ROM entry points, allwing for ***
9510 REM ***           BASIC1 and BASIC2          ***
9520 DEFPROCsetup
9530 IF ?&8015=ASC"1" THEN PROCset1 ELSE PROCset2
9540 ENDPROC
9550
9600 REM *** Set up BASIC1 entry points          ***
9610 DEFPROCset1
9615 prefnd = &B223 :REM Return to FNIPROC handter
9620 pushi  = &BDAC :REM Push IntA on the BASIC STACK
9625 popi1  = &BE23 :REM Pop &37-&3A from the STACK
9630 poppar = &8C5B :REM Pop parameter vatue from STACK
9635 getnsa = &9AF7 :REM Get <numeric> or <string>
9640 chksda = &9810 :REM Check end of statement (PTRA)
9645 chksdb = &980B :REM Check end of statement (PTRB)
9650 ENDPROC
9670
9800 REM *** Set up BASIC2 entry points          ***
9810 DEFPROCset2
9815 prefnd = &B1F4 :REM Return to FNIPROC handter
9820 pushi  = &BD94 :REM Push IntA on the BASIC STACK
9825 popi1  = &BE0B :REM Pop &37-&3A from the STACK
9830 poppar = &8CC1 :REM Pop parameter vatue from STACK
9835 getnsa = &9B1D :REM Get <numeric> or <string>
9840 chksda = &9857 :REM Check end of statement (PTRA)
9845 chksdb = &9852 :REM Check end of statement (PTRB)
9850 ENDPROC
```

The general operation of the routine is as follows:

1 It creates a filename using the name of the FN or PROC, which is left 1 byte after (&37). If it is a FN, 'F.' is put on the front: otherwise 'P.' is put on the front.

2 OSFILE is called to find the length of the overlay file, and the BASIC STACK is moved down by a corresponding amount. The old value of the STACK pointer is pushed onto the STACK so that it can be restored to its original value afterwards. This action also checks that the STACK

has not gone below the level of the HEAP (and produces a 'No room' error if it has).

3 OSFILE is called again, but this time to load the file into the space created for it on the STACK.

4 A 'P' or an 'F' is put in the token slot on the 6502 stack at &1FF. This will cause a 'No FN' or 'No PROC' error when the FN or PROC exits, so that the STACK can be restored, removing the overlayed file.

5 PTRA is pointed to the first character of the overlay and a JMP is made to continue with the FN/PROC handler.

When a 'No FN' or 'No PROC' error is generated on the return from the overlayed call (caused by the substitution of the call type identifier token at stage 4) the routine must not only do the job normally performed by end of the FN/PROC handler, but also remove the overlayed file from the BASIC STACK.

The action performed when this happens is as follows:

1 If it is the exit from a FN, the value is evaluated, and a check is made for the end of the statement. If it is the exit from a PROC, the end of statement chack only is made. These actions were not performed by the FN or PROC return statements before the error was generated.

2 The return address to the FN/PROC handler is pulled from the stack. The rest of this routine will do its job instead.

3 PTRB is restored from the stack.

4 The parameter values, pushed on the BASIC STACK when the FN/PROC call was made, are restored.

5 PTRA is restored from the stack.

6 The BASIC STACK, which is now in the same state which it was just after the overlay file was loaded, is restored to its previous value (which was pushed onto the STACK by the overlaying routine).

7 The 6502 stack is restored from the BASIC STACK.

8 The flags are set according to the byte in &27. If we are returning from a PROC, this has no effect; but if we are returning from a FN, the 6502 flags need to reflect the type of the value of the FN.

9 The routine exits, either to the PROC statement handler, or to the code which asked for the FN value.

For more details on the general operation of PROCs and FNs, see section 5.3. For more details on the 'No FN' (error number 7) and 'No PROC' (error number 13) see chapter 11.

This overlay routine is very much better than the one in section 8.2. However, there are still improvements which could be made to it. For example, if a recursive FN or PROC is used, it will load m another new version each time a call is made. Perhaps a linked list of overlayed files could be used to get round this.

Another way of overlaying may be to shift the STACK down bodily, and load the file between HIMEM and the bottom of the screen. A file loaded in this way could be left in memory until a 'No room' error was generated, and then it could be removed (providing it wasn't being executed at the time). In fact, there are many alternatives and improvements which can be made to this general idea.