

어셈블리어로 작성된

능동적 램상주 프로그램 툴

글·안 철 수(서울의대 생리학교실)

지난 호에서는 어떤 프로그램이 수행중일지라도 예약키만 누르면 당장 수행하던 프로그램의 실행을 멈추고 자기 자신이 실행되는 능동적 램상주 프로그램의 제작기법에 대해 설명하였다. 이번 호에서는 능동적 램상주 프로그램의 제작기법을 이용하여 능동적 램상주 프로그램의 기본 형식을 제시하고자 한다. 또한 이번에는 어셈블리어로 작성된 기본 형식만 선보이지만 정리되는대로 터보 파스칼, 터보 C로 작성된 능동적 프로그램의 기본 형식도 소개할 예정이다.

이번 달에는 지금까지 설명한 램상주 프로그램의 제작 기법을 사용하여 능동적 램상주 프로그램의 기본 골격을 제시하고자 한다. 램상주 프로그램은 주로 어셈블리어로 작성되는 것이 보통이지만, 램상주 프로그램에서 사용되는 기법을 잘 이해하기만 한다면 파스칼이나 C 언어와 같은 고수준 언어(high-level language)로도 램상주 프로그램을 작성할 수 있다.

본고에서는 지난 호에 예고한 바와 같이 어셈블리어로 작성된 능동적 램상주 프로그램의 기본 형식을 제공하고 자 한다. 터보 파스칼 4.0(또는 5.0) 및 터보 C로 작성된 능동적 램상주 프로그램의 기본 형식 또한 정리가 되는대로 독자 여러분께 선보일 예정이다.

능동적 램상주 프로그램 툴 사용방법

어셈블리어로 작성된 능동적 램상주 프로그램의 기본 형식은 <리스트 1>과 같다. 이것은 지난달에 설명한 내용을 그대로 프로그램한 것이므로 이해하는데 별 어려움이 없을 것으로 생각된다. 하지만 리스트를 완전히 이해하지 못하더라도 어셈블리어에 대한 기본적인 지식만 있다면, 이 기본 형식을 사용하여 누구나 쉽게 램상주 프로그램을 작성할 수 있을 것이다.

능동적 램상주 프로그램 툴을 사용하여 램상주 프로그램을 작성하려면, 먼저 프로그램의 처음 부분에 나오는 상수들을 정의해 주어야 한다.

HandlerID와 ExistID는 램상주 프로그램이 기억장소에 이미 존재하고 있는지를 검사할 때 사용되는 상수들이다.

HandlerID와 ExistID의 값은 0~FFFFh 이내의 어떤 수를 사용해도 되지만, 서로 다른 값으로 정의해 주어야 한다. 만약 같은 값을 사용하게 되면, 램상주 프로그램이 기억장소에 존재하지 않을 경우에도 존재하는 것으로 오인하게 되어 램상주 프로그램을 기억장소에 상주시킬 수 없게 된다.

ScanCode와 ShiftState는 램상주 프로그램에서 사용할 예약키(hot key)를 지정해 주는 상수이다. 램상주 프로그램에서 사용되는 예약키는 보통 쉬프트형 키(<Shift>, <Ctrl>, <Alt>)와 다른 키와의 조합으로 구성되는 경우가 많다. ScanCode는 쉬프트형 키와 함께 사용되는 키의 스캔 코드(scan code) 값이며, 각 키들의 스캔 코드 값들에 대해서는 지난달의 기사를 참조하기 바란다. ShiftState는 사용할 쉬프트형 키의 조합을 표시하며, 사용할 조합에 따른 값은 <표 1>과 같다. 예를 들어 <A> 키와 왼쪽 <Shift> 키, 오른쪽 <Shift> 키를 동시에 눌렀을 때 램상주 프로그램을 실행하고 싶다면, ScanCode의 값은 1Eh (<A>의 스캔코드 값), ShiftState의 값은 3으로 정의해 두면 된다.

StackSize는 램상주 프로그램이 사용할 스택의 크기를 나타내며, 보통의 경우에는 리스트에서와 같이 100h로 정의해 두는 것으로 충분하지만, 스택을 많이 사용하는 프로그램의 경우에는 이 크기를 필요한 만큼 늘리는 것이 좋다.

상수들을 정의해 준 후에는 램상주시키고 싶은 프로그램을 TSRMain 프로시저 내에 위치시키고, 사용할 데이터를 RES1_DATA 세그먼트 내에 위치시키는 것으로 램

상주 프로그램이 완성된다. 단, 램상주시킬 프로그램을 작성할 때 주의할 점이 두 가지 있다. 첫째, 디스크 입출력 기능-즉, 21h 번 인터럽트의 11h, 12h, 14h, 15h, 21h, 22h, 27h, 28h, 4Eh, 4Fh 번 기능을 사용할 때는, 리스트에서 표시해 둔 바와 같이 그런 프로그램의 DTA(disk transfer area)를 대피시키고 램상주 프로그램이 사용할 DTA를 새로 지정해 주어야 한다. 하지만 프로그램에서 디스크 입출력을 사용하지 않을 때는 이러한 루틴은 생략할 수 있다. 둘째, 램상주시킬 프로그램 내에서 21h 번 인터럽트의 1~Ch 번 기능을 사용할 수 없다. 따라서 키보드로부터 입력을 받거나 화면에 출력할 경우에는 롬 BIOS 루틴을 사용하거나, 입출력 루틴을 직접 만들어서 사용하여야 한다. 단, 21h 번 인터럽트의 다른 기능들이나 롬 BIOS는 자유롭게 사용할 수 있다.

완성된 프로그램은 마이크로소프트 어셈블러나 볼랜드의 터보 어셈블러를 사용하여 어셈블할 수 있다. 화일이름을 TSR.ASM이라고 할 때, 마이크로소프트 어셈블러를 사용하여 어셈블하는 방법은 다음과 같다.

A)masm tsr ;

A)link tsr ;

터보 어셈블러의 사용법은 다음과 같다(화일이름 다음에 ;를 사용할 필요가 없음에 유의하라).

A)tasm tsr

A)tlink tsr

어느 어셈블러를 사용하더라도 실행화일인 TSR.EXE를 얻을 수 있다.

이렇게 해서 작성된 램상주 프로그램은 보통의 프로그램들처럼 도스 프롬프트하에서 프로그램의 이름을 입력함으로써 기억장소 내에 상주시킬 수 있다. 이 때 만약 프로그램이 이미 상주하고 있다면 'Already Installed'라는 메시지를 출력시키고 수행을 중지하게 된다. 또한 프로그램에서 도스의 긴급에러 표시기(critical error flag)의 위치를 찾지 못하는 경우에도 'Unable to locate MS-DOS ErrorMode flag'라는 메시지를 출력시키고 수행을 멈추게 된다.

이미 상주하고 있는 프로그램을 기억장소에서 제거하고자 할 때에는 '-'를 인자(argument)로 사용하면 된다.

A)tsr -

와 같은 형식으로 램상주 프로그램을 제거시키게 되며, 성공적으로 제거되었을 때는 'Deinstalled'라는 메시지를 볼 수 있을 것이다.

〈표 1〉 ShiftState의 값

〈Alt〉	*****														
〈Ctrl〉	*****					*****									
left 〈Shift〉	**		**			**		**			**				
right 〈Shift〉	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ShiftState	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E

(*, pressed)

프로그램 설명

프로그램은 'The MS-DOS Encyclopedia'에 나와 있는 루틴을 기본으로 하여 램상주 프로그램 제거 등의 여러가지 기능을 첨가하고 불필요한 루틴들을 변형시키거나 삭제하였다.

능동적 램상주 프로그램은 프로그램을 기억장소에 상주시키는 부분, 예약키를 인지하여 램상주 프로그램을 호출하는 부분, 램상주 프로그램 고유의 역할을 수행하는 부분으로 크게 나눌 수 있다.

프로그램을 기억장소에 상주시키는 부분(Install 프로시저)에서는 램상주 프로그램이 기억장소 내에 존재하는지를 먼저 확인하는 작업을 수행한다. 만약 프로그램이 존재하지 않으면 사용중인 도스 버전, 도스 수행 표시기(InDOS flag)의 주소 및 긴급에러 표시기(critical error flag)의 주소를 저장시키고, 필요한 인터럽트 벡터를 바꾸어 주고, 프로그램의 정보영역(environment)을 제거한 후, 프로그램을 상주시키게 된다. 만약 램상주 프로그램이 이미 존재한다면, 인자를 검사하여 프로그램을 제거하거나 에러 메시지를 출력시킨다.

예약키는 9h 번 인터럽트(keyboard-action interrupt)를 가로채서 검사하게 되며, 8h 인터럽트(timer interrupt)가 호출될 때마다 하드웨어, BIOS, 도스가 안전한 상태에 있는지를 확인한 후 램상주 프로그램을 호출하게 된다. 또한 28h 번 인터럽트(idle interrupt) 처리 루틴내에서도 시스템이 안전한 상태이고 예약키가 눌러져 있으면 램상주 프로그램을 실행시키게 된다.

램상주 프로그램 고유의 역할을 수행하는 부분(Setup 프로시저)에서는 레지스터, 예외 처리 루틴, 확장 에러 정보, PSP, DTA를 대피시키고 〈Ctrl-Break〉 검사를 하지 않게 만든 다음에 고유의 역할(TSRMain 프로시저)을 수행시킨다. 또한, 수행을 끝나치고 원래의 프로그램으로 돌아가기 전에 이들을 다시 그전의 상태로 복구시키게 된다.

마지막으로 프로그램의 루틴중에서 언급하고 싶은 점이 세 가지 있다. 첫째, 프로그램이 기억장소 내에 상주하고 있는지를 검사하기 위하여 2Fh 번 인터럽트를 사용하

지 않고 11h 번 인터럽트를 사용하였다. 또한 11h 번 인터럽트를 호출할 때 램상주 프로그램의 제거에 필요한 정보도 같이 반환되도록 인터럽트 처리 루틴을 작성하였다. 둘째, 프로그램 내에서 21h 번 인터럽트의 50h 번 기능(set PSP address) 및 51h 번 기능(get PSP address)을 사용할 때는 도스 버전을 검사하여 2.xx의 경우에는 도스의 긴급에러 표시기를 억지로 1로 만들어서 보조 스택을

사용하게 함으로써 입출력 스택에 있는 데이터의 파괴를 막았다. 3.xx의 경우에는 50h 및 51h 번 기능이 호출 프로그램의 스택을 사용하기 때문에 아무런 문제가 없다. 셋째, 프로그램의 길이를 줄이기 위하여 도스 1.xx의 처리루틴을 작성하지 않았다. 하지만 만약 상업용 프로그램을 만든다면 이러한 루틴도 첨가해야 할 것이다.

<리스트 1> 능동적 램상주 프로그램의 기본 형식

```

001 HandlerID EQU ? ; *****
002 ExistID EQU ? ; *
003 ScanCode EQU ? ; *
004 ShiftState EQU ? ; *
005 StackSize EQU 100h ; *****
006
007 CR EQU 0Dh
008 LF EQU 0Ah
009 TRUE EQU -1
010 FALSE EQU 0
011
012 RESIDENT GROUP RESI_CODE, RESI_DATA, RESI_STACK
013
014 ;-----
015
016 RESI_CODE SEGMENT
017
018 ASSUME cs:RESIDENT, ds:RESIDENT
019
020 ;-----
021
022 ISR5 PROC far
023
024 inc cs:InISR5
025
026 pushf
027 cli
028 call cs:OldISR5
029
030 dec cs:InISR5
031 iret
032
033 ISR5 ENDP
034
035 ;-----
036
037 ISR8 PROC far
038
039 pushf
040 cli
041 call cs:OldISR8
042
043 cmp cs:InISR8,
044 jne Quit8
045
046 inc cs:InISR8
047 sti
048 call CheckSystem
049 jc NoSafe
050
051 mov byte ptr cs:InTSR, TRUE
052
053 call Setup
054
055 mov byte ptr cs:InTSR, FALSE
056
057 NoSafe: dec cs:InISR8
058 Quit8: iret
059
060 ISR8 ENDP
061
062 ;-----
063
064 ISR9 PROC far
065
066 push ds
067 push ax
068 push bx
069
070 in al, 60h
071
072 pushf
073 cli
074 call cs:OldISR9
075
076 mov ah, cs:InISR9
077 or ah, cs:HotFlag
078 jnz Quit9
079
080 inc cs:InISR9
081 sti
082
083 cmp al, ScanCode
084 jne Exit9
085
086 xor ax, ax
087 mov ds, ax
088 mov al, ds:[417h]
089 and al, 00001111b
090 cmp al, ShiftState
091 jne Exit9
092
093 mov byte ptr cs:HotFlag, TRUE
094
095 Exit9: dec cs:InISR9
096
097 Quit9: pop bx
098 pop ax
099 pop ds
100 iret
101
102 ISR9 ENDP
103
104 ;-----
105
106 ISR10 PROC far
107
108 inc cs:InISR10
109
110 pushf
111 cli
112 call cs:OldISR10
113
114 dec cs:InISR10
115 iret
116
117 ISR10 ENDP
118
119 ;-----
120
121 ISR11 PROC far
122
123 cmp cx, HandlerID
124 jne Quit11
125
126 mov cx, ExistID
127 mov si, seg RESI_DATA
128 mov ds, si
129 mov si, offset RESI_DATA:TSRPSP
130
131 Quit11: jmp cs:OldISR11
132
133 ISR11 ENDP
134
135 ;-----
136
137 ISR13 PROC far
138
139 inc cs:InISR13
140
141 pushf
142 cli
143 call cs:OldISR13
144
145 dec cs:InISR13
146
147 sti
148 iret
149
150 ISR13 ENDP
151
152 ;-----
153
154 ISR1B PROC far
155
156 mov byte ptr cs:Trap1B, TRUE
157 iret
158
159 ISR1B ENDP
160
161 ;-----
162
163 ISR23 PROC far
164

```

```

165      mov byte ptr cs:Trap23, TRUE
166      ired
167
168 ISR23      ENDP
169
170 ;-----
171
172 ISR24      PROC far
173
174      mov byte ptr cs:Trap24, TRUE
175
176      cmp cs:DOSVersion, 300h
177      jae DOS3
178      xor al, al
179      ired
180
181      DOS3: mov al, 3
182      ired
183
184 ISR24      ENDP
185
186 ;-----
187
188 ISR28      PROC far
189
190      pushf
191      cli
192      call cs:OldISR28
193
194      cmp cs:InISR28,
195      jne Quit28
196
197      inc cs:InISR28
198
199      call CheckSystem
200      jc Unsafe
201
202      mov byte ptr cs:InTSR, TRUE
203      call Setup
204      mov byte ptr cs:InTSR, FALSE
205
206      Unsafe: dec cs:InISR28
207
208      Quit28: ired
209
210 ISR28      ENDP
211
212 ;-----
213
214 CheckSystem PROC near
215
216      rcl cs:HotFlag, 1
217      cmc
218      jc Check
219
220      ror cs:InTSR, 1
221      jc Check
222
223      lds bx, cs:ErrorModeAddr
224      mov ah, [bx]
225      lds bx, cs:InDOSAddr
226      mov al, [bx]
227      xor bx, bx
228      cmp bl, cs:InISR28
229      rcl bl, 1
230      cmp bx, ax
231      jc Check
232
233      mov ax, 0Bh
234      out 20h, al
235      jmp here
236      here: in al, 20h
237      cmp ah, al
238      jc Check
239
240      xor al, al
241      cmp al, cs:InISR5
242      jc Check
243      cmp al, cs:InISR9
244      jc Check
245      cmp al, cs:InISR10
246      jc Check
247      cmp al, cs:InISR13
248
249      Check: ret
250
251 CheckSystem ENDP
252
253 ;-----
254
255 Setup      PROC near
256
257      push ds
258      push cs
259      pop ds
260
261      mov OldSP, sp
262      mov OldSS, ss
263      mov sp, seg RESI_STACK
264      mov ss, sp
265      mov sp, StackSize

```

```

266      push es
267      push ax
268      push bx
269      push cx
270      push dx
271      push si
272      push di
273      push bp
274      cld
275
276      mov cx, NTrap
277      mov si, offset RESIDENT:StartTrapList
278
279 SetTrap: lodsb
280      mov byte ptr [si], FALSE
281      push ax
282      mov ah, 35h
283      int 21h
284      mov [si + 1], bx
285      mov [si + 3], es
286      pop ax
287      mov dx, [si + 5]
288      mov ah, 25h
289      int 21h
290      add si, 7
291      loop SetTrap
292
293      mov ah, 33h
294      xor al, al
295      int 21h
296      mov OldBreak, dl
297      xor dl, dl
298      mov ah, 33h
299      mov al, 1
300      int 21h
301
302      cmp DOSVersion, 30Ah
303      jb SetPSP
304      push ds
305      xor bx, bx
306      mov ah, 59h
307      int 21h
308      mov cs:OldExtErrDS, ds
309      pop ds
310      mov OldExtErrAX, ax
311      mov OldExtErrBX, bx
312      mov OldExtErrCX, cx
313      mov OldExtErrDX, dx
314      mov OldExtErrSI, si
315      mov OldExtErrDI, di
316      mov OldExtErrES, es
317
318 SetPSP: mov ah, 51h
319      call Int21
320      mov OldPSP, bx
321      mov bx, TSRPSP
322      mov ah, 50h
323      call Int21
324
325      mov ah, 2Fh
326      int 21h
327      mov word ptr OldDTA, bx
328      mov word ptr OldDTA, es
329      push ds
330      mov ds, TSRPSP
331      mov dx, 80h
332      mov ah, 1Ah
333      int 21h
334      pop ds
335
336      call TSRMain
337
338      push ds
339      lds dx, OldDTA
340      mov ah, 1Ah
341      int 21h
342      pop ds
343
344      mov bx, OldPSP
345      mov ah, 50h
346      call Int21
347
348      mov ax, DOSVersion
349      cmp ax, 30Ah
350      jb ResetBrk
351      mov dx, offset RESIDENT:OldExtErrInfo
352      mov ax, 5D0Ah
353      int 21h
354
355 ResetBrk: mov dl, OldBreak
356      mov ah, 33h
357      mov al, 1
358      int 21h
359
360      mov cx, NTrap
361      mov si, offset RESIDENT:StartTrapList
362      push ds
363      ResetTrp: lodsb byte ptr cs:[si]
364      lds dx, cs:[si + 1]
365      mov ah, 25h
366      int 21h
367      add si, 7

```

```

*****
*
*
* 램상주 프로그램*
* 예시 디스크 입출*
* 력을 수행할 때만*
* 필요하다.*
*
*
*****

*****
*
*
* 디스크 사용 때*
* 필요*
*
*
*****

```

```

367      loop ResetTrp
368      pop  ds
369
370      pop  bp
371      pop  di
372      pop  si
373      pop  dx
374      pop  cx
375      pop  bx
376      pop  ax
377      pop  es
378      mov  ss, OldSS
379      mov  sp, OldSP
380      pop  ds
381
382      mov  byte ptr cs:HotFlag, FALSE
383
384      ret
385
386 Setup      ENDP
387
388 ;-----
389
390 Int21      PROC  near
391
392      cmp  DOSVersion, 300h
393      jb   Under30
394      int  21h
395      ret
396
397      Under30: push ds
398              push bx
399              lds  bx, ErrorModeAddr
400              inc  byte ptr [bx]
401              pop  bx
402              pop  ds
403
404              int  21h
405
406              push ds
407              push bx
408              lds  bx, ErrorModeAddr
409              dec  byte ptr [bx]
410              pop  bx
411              pop  ds
412
413      ret
414
415 Int21      ENDP
416
417 ;-----
418
419 TSRMain    PROC  near
420
421      ; *****
422      ; *
423      ; *   램상주 프로그램이 이 곳에 둔다
424      ; *
425      ; *
426      ; *   *****
427
428      ret
429
430 TSRMain    ENDP
431
432 ;-----
433
434 RESI_CODE  ENDS
435
436 ;-----
437
438 RESI_DATA  SEGMENT
439
440      DOSVersion    DW  ?
441      ErrorModeAddr DD  ?
442      InDOSAddr     DD  ?
443
444      HotFlag       DB  FALSE
445      InTSR         DB  FALSE
446
447      TSRPSP        DW  ?
448
449      NISR          DW  (EndISRList-StartISRList)/8
450
451      StartISRList  DB  05h
452      InISR5        DB  FALSE
453      OldISR5       DD  ?
454      OldISR5       DW  offset RESIDENT:ISR5
455
456
457      InISR8        DB  08h
458      InISR8        DB  FALSE
459      OldISR8       DD  ?
460      OldISR8       DW  offset RESIDENT:ISR8
461
462
463      InISR9        DB  09h
464      InISR9        DB  FALSE
465      OldISR9       DD  ?
466      OldISR9       DW  offset RESIDENT:ISR9
467
468      DB  10h

```

```

468      InISR10      DB  FALSE
469      OldISR10     DD  ?
470      OldISR10     DW  offset RESIDENT:ISR10
471
472      InISR11      DB  11h
473      InISR11      DB  FALSE
474      OldISR11     DD  ?
475      OldISR11     DW  offset RESIDENT:ISR11
476
477
478      InISR13      DB  13h
479      InISR13      DB  FALSE
480      OldISR13     DD  ?
481      OldISR13     DW  offset RESIDENT:ISR13
482
483
484      InISR28      DB  28h
485      InISR28      DB  FALSE
486      OldISR28     DD  ?
487      OldISR28     DW  offset RESIDENT:ISR28
488
489      EndISRList   LABEL  BYTE
490
491      NTrap        DW  (EndTrapList-StartTrapList)/8
492
493      StartTrapList DB  1Bh
494      Trap1B       DB  FALSE
495      OldISR1B     DD  ?
496      OldISR1B     DW  offset RESIDENT:ISR1B
497
498
499      Trap23       DB  23h
500      Trap23       DB  FALSE
501      OldISR23     DD  ?
502      OldISR23     DW  offset RESIDENT:ISR23
503
504
505      Trap24       DB  24h
506      Trap24       DB  FALSE
507      OldISR24     DD  ?
508      OldISR24     DW  offset RESIDENT:ISR24
509
510      EndTrapList  LABEL  BYTE
511
512      OldSP        DW  ?
513      OldSS        DW  ?
514      OldBreak     DB  ?
515      OldPSP       DW  ?
516      OldDTA       DD  ?
517
518      OldExtErrInfo LABEL  BYTE
519      OldExtErrAX  DW  ?
520      OldExtErrBX  DW  ?
521      OldExtErrCX  DW  ?
522      OldExtErrDX  DW  ?
523      OldExtErrSI  DW  ?
524      OldExtErrDI  DW  ?
525      OldExtErrDS  DW  ?
526      OldExtErrES  DW  ?
527      OldExtErrES  DW  3 dup(0)
528
529      ; *****
530      ; *
531      ; *   프로그램의 데이터를 이 곳에 둔다
532      ; *
533      ; *   *****
534
535      RESI_DATA    ENDS
536
537      RESI_STACK   SEGMENT  stack
538
539      DB  StackSize dup(?)
540
541      RESI_STACK   ENDS
542
543      ;-----
544
545      TRAN_CODE    SEGMENT
546
547      ASSUME  cs:TRAN_CODE,ds:RESI_DATA,ss:RESI_STACK
548
549      ;-----
550
551      Install      PROC  far
552
553      ASSUME  ds:RESI_DATA
554
555      mov  ax, seg RESI_DATA
556      mov  ds, ax
557      mov  TSRPSP, es
558
559      mov  cx, Handler1D
560      int  11h
561      cmp  cx, ExistID
562      jne  NoExist
563
564      mov  ah, 2Fh
565      int  21h
566      mov  al, es:[bx]
567      cmp  al, 0
568      je   NoArg

```



```

569      mov al, es:[bx + 2]
570      cmp al, '-'
571      jne NoArg
572      jmp DeInstall
573
574 NoArg: mov al, 1
575      call FatalError
576
577 NoExist: mov ah, 30h
578          int 21h
579
580          xchg ah, al
581          mov DOSVersion, ax
582
583          call GetDOSFlags
584
585          push es
586          mov cx, NISR
587          mov si, offset StartISRList
588 SetISR: lodsb
589          push ax
590          mov ah, 35h
591          int 21h
592          mov [si + 1], bx
593          mov [si + 3], es
594          pop ax
595          push ds
596          mov dx, [si + 5]
597          mov bx, seg RESIDENT
598          mov ds, bx
599          mov ah, 25h
600          int 21h
601          pop ds
602          add si, 7
603          loop SetISR
604
605          pop es
606          push es
607          mov es, es:[2Ch]
608          mov ah, 49h
609          int 21h
610
611          pop ax
612          mov dx, cs
613          sub dx, ax
614          mov ah, 31h
615          xor al, al
616          int 21h
617
618 Install      ENDP
619
620 ;-----
621
622 GetDOSFlags PROC near
623
624      ASSUME ds:RESI_DATA
625
626      push es
627      mov ah, 34h
628      int 21h
629      mov word ptr InDOSAddr, bx
630      mov word ptr InDOSAddr[2], es
631
632      mov word ptr ErrorModeAddr[2], es
633      mov ax, DOSVersion
634      cmp ax, 30Ah
635      jb Below31
636
637      dec bx
638      mov word ptr ErrorModeAddr, bx
639      jmp Quit
640
641 Below31: mov cx, 0FFFFh
642          xor di, di
643 Search:  mov ax, word ptr cs:MN2
644 Again:  repne scasb
645          jne Error
646          cmp ah, es:[di]
647          jne Again
648          mov ax, word ptr cs:MN1 + 1
649          cmp ax, es:[di][MN1 - MN2]
650          jne Other
651          mov ax, es:[di][MN1 - MN2 + 2]
652          jmp Find
653
654 Other:   mov ax, word ptr cs:MN3 + 1
655          cmp ax, es:[di][MN3 - MN4]
656          jne Search
657
658          mov ax, es:[di][MN3 - MN4 + 2]
659 Find:   mov word ptr ErrorModeAddr, ax
660
661 Quit:   pop es
662
663          ret
664
665 Error:  mov al, 2
666          call FatalError
667
668 MNnear  LABEL near
669 MNbyte  LABEL byte

```

```

670 MNword  LABEL word
671 MN1:    cmp ss:MNbyte, 0
672          jne MNnear
673 MN2:    int 28h
674
675 MN3:    test ss:MNbyte, 0FFh
676          jne MNnear
677          push ss:MNword
678 MN4:    int 28h
679
680 GetDOSFlags ENDP
681
682 ;-----
683
684 DeInstall PROC near
685
686          mov es, ds:[si]
687
688          mov cx, ds:[si + 2]
689          add si, 4
690 ResetISR: lodsb
691          push ds
692          mov dx, [si + 1]
693          mov bx, [si + 3]
694          mov ds, bx
695          mov ah, 25h
696          int 21h
697          pop ds
698          add si, 7
699          loop ResetISR
700
701          mov ah, 49h
702          int 21h
703
704          mov ah, 9
705          mov dx, seg Mess
706          mov ds, dx
707          mov dx, offset Mess
708          int 21h
709
710          xor al, al
711          mov ah, 4Ch
712          int 21h
713
714 Mess     DB CR, LF, 'Deinstalled', CR, LF, 's'
715
716 DeInstall ENDP
717
718 ;-----
719
720 FatalError PROC near
721
722          mov ah, 9
723          cmp al, 1
724          jne Err2
725
726          mov dx, seg ErrMess1
727          mov ds, dx
728          mov dx, offset ErrMess1
729          int 21h
730          jmp Exit
731
732 Err2:    mov dx, seg ErrMess2
733          mov ds, dx
734          mov dx, offset ErrMess2
735          int 21h
736
737 Exit:    mov ah, 4Ch
738          int 21h
739
740 ErrMess1 DB CR, LF, 'Already Installed', CR, LF, 's'
741 ErrMess2 DB CR, LF, 'Unable to locate MS-DOS ErrorMode
742          flag'
743          DB CR, LF, 's'
744 FatalError ENDP
745
746 ;-----
747
748 TRAN_CODE ENDS
749
750 ;=====
751
752      END Install

```