[Q]: Generic time slicing function for many multi-taskers

[A]: Serg Projzogin

;

; ; SLICE.ASM ; ; Provides a generic time slicing function for all multi-taskers I know ; or care about. ; ; Note that this library is Turbo Assembler specific, since I have long ; since weaned myself from MASM's brain-dead memory addressing syntax. ; ; This library is designed to be easily extended; for each new ; multi-tasker supported, you need to write a detect routine and a ; time-slice routine. ; ; Your detection function will take no input, and should return with ; carry set if the associated multi-tasker is detected. This routine ; may safely alter register AX. No other registers should be altered. ; ; The time-slice routine will take no input and give up a “standard” ; timeslice for the associated multi-tasker. This routine may safely ; alter registers AX, BX and DS. No other registers should be altered. ; ; Once you have such routines written, add their addresses to the ; arrays detect_func and slice_func below. Increment the ; NumMultitaskers equate, and you're done. ; ; This library placed in the public domain by Kevin Vigor, 1/5/93. ; I would, however, appreciate it if you do the following two things: ; ; 1: If you distribute an altered version of this source, please add to ; this header a log of your changes; ; ; 2: If you discover any bugs or extend this library, please send a copy ; of your changes to me at one of the below addresses: ; ; Compuserve: 72500,3705 ; Internet: kevin@wicat.com ; 72500.3705@compuserve.com ;

IDEAL ; Requires Turbo Assembler.

MODEL SMALL ; This may be changed to any model safely. Note,

```
                        ; however, that you will not be able to link
                        ; this routine to a .COM, since it makes explicit
                        ; segment refrences. This is just laziness; I
                        ; haven't bothered to do all the ifdef'ing.
```

LOCALS ; Allow local symbols starting with @@

DATASEG

; Define known multitaskers. None equ 0 DesqView equ 1 Windows_3x equ 2 OS2_2x equ 3

NumMultitaskers EQU 3 ; Do not include 'None'

current_tasker dw 0 ; Detected multi-tasker

; Table of detection routines.

detect_func DW OFFSET @code:dummy_detect

```
                DW  OFFSET @code:Desqview_detect
                DW  OFFSET @code:Windows_3X_detect
                DW  OFFSET @code:OS2_2x_detect
```

; Table of time-slicing functions.

slice_func DW OFFSET @code:dummy_slice

```
                DW OFFSET @code:Desqview_slice
                DW OFFSET @code:Win_3x_or_OS2_2x_slice
                DW OFFSET @code:Win_3x_or_OS2_2x_slice
```

CODESEG

PUBLIC _detect_multitasker, _timeslice

;; Detection routines: return with carry set if the appropiate tasker is ;; detected and clear if not.

PROC dummy_detect ; SHould never be called, but does no harm. clc ; Always fail. ret ENDP

PROC Desqview_detect ; Return with carry set if Desqview detected. ; ; This routine is based on information in the Desqview version 2.x manual. push ax push bx push cx push dx

mov cx, 'DE' mov dx, 'SQ' mov ax, 02B01h ; DOS set date function. int 021h cmp al, 0FFh ; Did DOS report the invalid date? jnz @@desqview ; If not, we've got Desqview.

clc ; Report failure.

@@clean_stack: pop dx pop cx pop bx pop ax ret

@@desqview:

; BH = Desqview major version, BL = Desqview minor version. I have no idea ; at what version the timeslicing calls became available, so I just assume ; they are supported. If this is an invalid assumption, this would be the ; place to test.

stc ; Report sucess. jmp short @@clean_stack ; and exit.

ENDP ; Desqview_detect.

PROC Windows_3X_detect ; Note: this function detects Windows 3.x in enhanced mode only. ; I am not a Windows guru (or even user), but I believe there is no ; capability for time-slicing in standard or real modes, therefore this ; function is sufficient for the purposes of this library. ; I am basing this function on the fine book PC Interrupts, which lists ; a number of magic values which mean WIndows 3.x enhanced mode is not running.

push ax

mov ax, 01600h int 02Fh

cmp al, 00h jz @@no_Windows

cmp al, 080h jz @@no_Windows

cmp al, 01h ; Windows/386 2.x; not supported. jz @@no_windows

cmp al, 0FFh ; Windows/386 2.x; not supported.

; If AL is none of the above values, it is the Windows major version number.

cmp al, 03h ; At least Win 3.0? jb @@no_windows

stc ; Yes, report sucess. pop ax ret

@@no_windows: clc ; Report failure. pop ax ret ENDP

PROC OS2_2x_detect ; I do not know of an 'official' way of testing for OS/2 presence; the ; method used here is to test the DOS version. If the major version ; is 20 or above, we assume we're in an OS/2 2.x DOS box.

push ax push cx

mov ah, 030h ; DOS get version fn. int 021h

cmp al, 014h ; 20 decimal. jb @@no_OS2

stc ; Report sucess.

@@clean_stack: pop cx pop ax ret

@@no_OS2: clc ; Report failure. jmp short @@clean_stack

ENDP

;; Time slicing routines for each tasker.

PROC dummy_slice ; Should never be called, but does no harm. ret ENDP

PROC Desqview_slice ; Give up a slice under Desqview.

ASSUME cs:@code, ds:nothing, es:nothing mov ax, 0101Ah ; Switch to DV's stack. int 015h mov ax, 01000h ; Give up time-slice. int 015h mov ax, 01025h ; Restore local stack. int 015h ret ENDP

PROC Win_3x_or_OS2_2x_slice

; This call works under either Windows 3.x in Enhanced mode, or OS/2 2.x

ASSUME ds:@code, ds:nothing, es:nothing mov ax, 01680h ; Win 3.x / OS/2 2.x timeslice call. int 02Fh ret ENDP

PROC _detect_multitasker ; Tries to find a multi-tasker. ; Returns the ID in AX, and sets up the internal data to call _timeslice. ; ; Note that this function can be safely called from Turbo/Borland C. I have ; no idea about other compilers.

push ds push bx push cx

ASSUME cs:@code, ds:nothing, es:nothing mov ax, @data mov ds, ax ASSUME ds:@data

mov cx, NumMultitaskers ; Number of routines to try. xor ax, ax

@@detect_loop: inc ax

; AX holds the number of the detection routine to try. push ax shl ax, 1 mov bx, ax ; BX = AX * 2

call [detect_func + bx] ; Call this function. pop ax ; Restore AX. jc @@found_one ; quit now if we hit

one.

loop @@detect_loop ; Go through all known detection routines.

xor ax, ax ; Signal failure. jmp short @@clean_stack ; and exit.

@@found_one: mov [current_tasker], ax

@@clean_stack: pop cx pop bx pop ds

ASSUME ds:nothing

ret ENDP

PROC _timeslice ; Give up a timeslice. Depends on having the current_tasker global set by ; a call to detect_multitasker. However, will call dummy_slice and do no ; harm if detect_multitasker has not been called. ; ; Note that this function can be safely called from Turbo/Borland C. I have ; no idea about other compilers.

push ds push ax push bx

ASSUME cs:@code, ds:nothing, es:nothing mov ax, @data mov ds, ax ASSUME ds:@data

mov ax, [current_tasker] shl ax, 1 ; BX = AX * 2 mov bx, ax

call [slice_func + bx] ; Call appropiate time-slice function.

pop bx pop ax pop ds ret ENDP

END

Cut

Cut

/* SLICE.H * * Turbo/Borland C prototypes for the functions provided by SLICE.ASM * */ #ifndef SLICE_H_ #define SLICE_H_

/* Returns zero if no known multi-tasker found, or an ID if one is. */ int detect_multitasker(void);

/* Give up a timeslice. detect_multitasker should be called first. */ void timeslice(void);

#endif

Cut

Cut

/* * TEST.C * * Stupid test-bed for the time-slicing functions in SLICE.ASM; * simply detects a multi-

tasker and then waits for a keystroke * twice, once with time-slicing and once without. */

#include <stdio.h> #include <stdlib.h> #include <string.h> #include <conio.h>

#include “slice.h”

static char *tasker_names[] = {

```
  "None",
  "DesqView",
  "Windows 3.x (enhanced)",
  "OS/2 2.x"
```

};

void main(void) {

```
  int tasker = detect_multitasker();
```

```
  printf("Multitasker found: %s\r\n", tasker_names[tasker]);
```

if (!tasker)

```
exit(1);
```

```
  puts("Waiting for keystroke (no slicing...)");
  while (!kbhit())
      ;
```

```
  getch();
```

```
  puts("Waiting for keystroke (slicing...)");
  while (!kbhit())
      timeslice();
```

```
  getch();
```

```
  exit(0);
```

}