Note: This API calls are shared between DOS and Win16 personality.

DPMI is a shared interface for DOS applications to access Intel 80286+ CPUs services. DOS DMPI host provides core services for protected mode applications. Multitasking OS with DOS support also provides DMPI in most cases. Windows standard and extended mode kernel is a DPMI client app. Standard and extended mode kernel differs minimally and shares common codebase. Standard Windows kernel works under DOSX extender. DOSX is a specialized version of 16-bit DPMI Extender (but it is standard DPMI host). Standard mode is just DPMI client, exnhanced mode is DPMI client running under Virtual Machime Manager (really, multitasker which allow to run many DOS sessions). Both modes shares DPMI interface for kernel communication. The OS/2 virtual DOS Protected Mode Interface (VDPMI) device driver provides Version 0.9 DPMI support for virtual DOS machines. Win16 (up to Windows ME) provides Version 0.9 DPMI support. Windows in Standard Mode provides DPMI services only for Windows Applications, not DOS sessions.

DPMI host often merged with DPMI extender. Usually DPMI extender provide DPMI host standard services and DOS translation or True DPMI services.

2021/08/05 10:15 · prokushev · 0 Comments

Int 31H, AH=05H, AL=08H

Version

1.0

Brief

Map Device in Memory Block

Input

```
AX = 0508H
ESI = memory block handle
EBX = offset within memory block of page(s) to be mapped (must be page-
aligned)
ECX = number of pages to map
EDX = physical address of device (must be page-aligned)
```

Return

```
if function successful
Carry flag = clear

if function unsuccessful
Carry flag = set
AX = error code Capability not supported)
8001H   unsupported function (Device Mapping
8003H   system integrity (invalid device address)
8023H   invalid handle (in ESI)
8025H   invalid linear address (specified range is not within specified
block or EBX/EDX is not page-aligned)
```

Notes

Maps the physical addresses assigned to a device onto the linear addresses of a memory block previously allocated with Int 31H Function 0504H.

16-bit DPMI hosts will not support this function. A 16-bit client of a 32-bit DPMI 1.0 host can use this function.

Support of this call by 32-bit DPMI hosts is optional. Application programs or DOS Extenders which require this call in order to run are not DPMI Compliant.

Any committed or mapped pages resided in the linear address range that is being mapped into will be uncommitted or unmapped automatically by the host.

All pages created by this call have the mapped bit (bit 2) set in the attributes returned by the Get Page Attributes function (Int 31H Function 0506H).

This function differs from the Create Physical Address Mapping function (Int 31H Function 0800H) in that this function supports mapping of physical devices within an existing memory block, rather than at an arbitrary linear address. Use of an existing memory block gives 32-bit programs the ability to access physical devices with NEAR pointers, which is often highly desirable for performance reasons.

Unlike Int 31H Function 0800H, this function allows mapping of addresses below 1 MB that do not lie within RAM available for use by programs; e.g. this function can be used to map the refresh buffers of IBM-compatible display adapters.

If the DPMI host is not virtualizing the device, it must disable any memory caching on the mapped pages; in particular, on the 486 or later, the PCD (page cache disable) bit must be set in the page table entries.

DPMI hosts that do not virtualize physical devices can support this function by creating page table entries that map the physical device. The page table entries must be marked as mapped so that the host knows not to attempt freeing of physical memory for the pages when the memory block is freed.

DPMI hosts are allowed to support this function for some physical devices and not for others, because

mapping of virtualized devices requires page aliasing in the host - a complex task. DPMI hosts with partial support for this function may fail the function call on virtualized devices (such as displays), and allow the call on non-virtualized devices (such as the Weitek coprocessors). Allowing the client to map a physical device so that it can be accessed with NEAR references, for example, may help the client achieve considerably better performance.

See also

Note

Text based on http://www.delorie.com/djgpp/doc/dpmi/

DPMI	
Process manager	**INT 2FH** 1680H, 1687H
Signals	
Memory manager	
Misc	**INT 2FH** 1686H, 168AH
Devices	

2021/08/13 14:23 · prokushev · 0 Comments