

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=0DH, AL=02H

Version

1.0

Brief

Serialize on Shared Memory

Input

```
AX = 0D02H
SI:DI = shared memory block handle
DX = option flags
```

Bit	Significance
0	0 = suspend client until serialization available 1 = return immediately with error if serialization not available
1	0 = exclusive serialization requested 1 = shared serialization requested
2-15	reserved, must be zero

Return

```
if function successful
Carry flag = clear

if function unsuccessful
Carry flag = set
AX = error code
8004H   deadlock (host detected a deadlock situation)
8005H   request cancelled with Int 31H Function 0D03H
8017H   lock count exceeded
8018H   exclusive serialization already owned by another client
8019H   shared serialization already owned by another client
8023H   invalid handle
```

Notes

Requests serialization of a shared memory block. Successful serialization symbolizes ownership or right of access to a block, and can be used by DPMI clients to synchronize the inspection or modification of a shared memory block.

For each client, the DPMI host maintains four different local (virtual machine) serialization counts (exclusive, shared, pending shared, and pending exclusive) for each shared memory block, as well as a global serialization count. The global serialization count is only updated when the sum of a virtual machine's exclusive and shared serialization counts goes from 0 to 1 (serialize) or 1 to 0 (free).

A successful exclusive serialization blocks any serialization request (exclusive or shared) for the same block by another virtual machine. Exclusive serialization should be regarded as “ownership for writing,” and should only be requested if the client intends to modify the block. A successful shared serialization will only block requests for exclusive serialization by another client. Shared serialization can be thought of as “read-only access,” and should used when the client only intends to inspect the block and will not change its contents.

Setting bit 0 of DX to 1 when the serialization request is made allows a client to determine whether a shared memory area is serialized without being suspended. Clients which “poll” for the availability of a resource in this manner are encouraged to yield the CPU with Int 2FH Function 1680H at appropriate intervals.

A serialization call that causes a client to be suspended can be canceled by a client interrupt service routine (such as a keyboard or timer interrupt handler) requesting the Free Serialization function (Int 31H Function 0D03H). In such cases, the original serialization request will return with the Carry flag set and AX = 8005H.

A client that has been suspended while waiting for serialization of a shared memory block can still service interrupts. Some hosts may need to reissue the serialization request on behalf of the client after the interrupt service routine returns, but this event will be invisible to the client.

Hosts are not required to detect deadlock. Clients that terminate and stay resident in order to function as resident service providers, executing in the context of other clients, must be careful to

avoid deadlocks and incorrect sequencing in acquiring and releasing resources.

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

From:
https://osfree.org/doku/ - **osFree wiki**

Permanent link:
https://osfree.org/doku/doku.php?id=en:docs:dpmi:api:int31:0d:02

Last update: **2021/08/27 06:54**