# VioGlobalReg

**Bindings**: C, MASM

VioGlobalReg allows a subsystem to receive notification at the completion of VIO calls issued by all applications running in full-screen sessions.

*VioGlobalReg* (ModuleName, EntryPoint, FunctionMask1, FunctionMask2, 0)

*ModuleName* (**PSZ**) - input Address of the ASCIIZ string containing the 1-8 character file name of the subsystem. The maximum length of the ASCIIZ string is 9 bytes including the terminating byte of zero. The module must be a dynamic link library but the name supplied must not include the .DLL extension.

*EntryPoint* (**PSZ**) - input Address of the ASCIIZ name string containing the dynamic link entry point name of the routine in the subsystem to receive control when any of the registered functions is called. The maximum length of the ASCIIZ string is 33 bytes including the terminating byte of zero.

*FunctionMask1* (**ULONG**) - input A bit mask where each bit identifies a video function being registered. The bit definitions are shown below. The first word pushed onto the stack contains the high-order 16 bits of the function mask, and the second word contains the low-order 16 bits.

| BIT | REGISTERED FUNCTION | BIT | REGISTERED FUNCTION |
|---|---|---|---|
| 31 | VioPrtScToggle | 15 | VioWrtCharStr |
| 30 | VioEndPopUp | 14 | VioWrtTTY |
| 29 | VioPopUp | 13 | VioWrtNCell |
| 28 | VioSavRedrawUndo | 12 | VioWrtNAttr |
| 27 | VioSavRedrawWait | 11 | VioWrtNChar |
| 26 | VioScrUnLock | 10 | VioReadCellStr |
| 25 | VioScrLock | 9 | VioReadCharStr |
| 24 | VioPrtSc | 8 | VioShowBuf |
| 23 | VioGetAnsi | 7 | VioSetMode |
| 22 | VioSetAnsi | 6 | VioSetCurType |
| 21 | VioScrollRt | 5 | VioSetCurPos |
| 20 | VioScrollLf | 4 | VioGetPhysBuf |
| 19 | VioScrollDn | 3 | VioGetBuf |
| 18 | VioScrollUp | 2 | VioGetMode |
| 17 | VioWrtCellStr | 1 | VioGetCurType |
| 16 | VioWrtCharStrAtt | 0 | VioGetCurPos |

*FunctionMask2* (**ULONG**) - input A bit mask where each bit identifies a video function being registered. The bit mask has the format shown below. The first word pushed onto the stack contains the high order 16 bits of the function mask, and the second word contains the low order 16 bits. Unused bits are reserved and must be set to zero.

| Bit | Registered Function |
|---|---|
| 31-11 | Reserved, must be set to zero. |
| 10 | VioDeRegister |
| 9 | VioRegister |

| Bit | Registered Function |
|---|---|
| 8 | VioSetState |
| 7 | VioGetState |
| 6 | VioSetFont |
| 5 | VioGetCp |
| 4 | VioSetCp |
| 3 | VioGetConfig |
| 2 | VioGetFont |
| 1 | VioModeUndo |
| 0 | VioModeWait |

*Reserved* (**LONG**) - input Reserved and must be zero.

*rc* (**USHORT**) - return Return code descriptions are:

| | |
|---|---|
| 0 | NO_ERROR |
| 349 | ERROR_VIO_INVALID_MASK |
| 403 | ERROR_VIO_INVALID_ASCIIZ |
| 426 | ERROR_VIO_REGISTER |
| 494 | ERROR_VIO_EXTENDED_SG |

**Remarks**

Notification of VIO calls issued within the hard error handler and DOS (real mode) sessions is not provided.

When control is routed to *EntryPoint*, the stack appears as it did after the original VIO call except that four additional values have been pushed onto the stack. The first is the index number (**WORD**) of the routine called. The second is a near pointer (**WORD**). The third is the caller's DS register (**WORD**). The fourth is the return address (**DWORD**) to the VIO router.

For example, if VioSetCurPos were a registered function, the stack would appear as if the following instruction sequence were executed if VioSetCurPos were called and control routed to *EntryPoint*:

<wiki> ┌──────────────────────────────────────────────────────────────┐ │ PUSH WORD Row │ │ PUSH WORD Column │ │ PUSH WORD VioHandle │ │ CALL FAR VioSetCurPos │ │ PUSH WORD Index │ │ CALL NEAR Entry point in Vio router │ │ PUSH WORD Caller's DS │ │ CALL FAR Dynamic link entry point │ └──────────────────────────────────────────────────────────────┘ </wiki>

The index numbers that correspond to the registered functions are listed below:

| | | | |
|---|---|---|---|
| 0 | VioGetPhysBuf | 22 | VioSetAnsi |
| 1 | VioGetBuf | 23 | VioGetAnsi |
| 2 | VioShowBuf | 24 | VioPrtSc |
| 3 | VioGetCurPos | 25 | VioScrLock |
| 4 | VioGetCurType | 26 | VioScrUnLock |
| 5 | VioGetMode | 27 | VioSavRedrawWait |
| 6 | VioSetCurPos | 28 | VioSavRedrawUndo |
| 7 | VioSetCurType | 29 | VioPopUp |
| 8 | VioSetMode | 30 | VioEndPopUp |

| | | | |
|---|---|---|---|
| 9 | VioReadCharStr | 31 | VioPrtScToggle |
| 10 | VioReadCellStr | 32 | VioModeWait |
| 11 | VioWrtNChar | 33 | VioModeUndo |
| 12 | VioWrtNAttr | 34 | VioGetFont |
| 13 | VioWrtNCell | 35 | VioGetConfig |
| 14 | VioWrtCharStr | 36 | VioSetCp |
| 15 | VioWrtCharStrAtt | 37 | VioGetCp |
| 16 | VioWrtCellStr | 38 | VioSetFont |
| 17 | VioWrtTTY | 39 | VioGetState |
| 18 | VioScrollUp | 40 | VioSetState |
| 19 | VioScrollDn | 41 | VioRegister |
| 20 | VioScrollLf | 42 | VioDeRegister |
| 21 | VioScrollRt | | |

On entry to the global subsystem, AX contains the return code that is returned to the application that issued the VIO call. The global subsystem must return with all stack parameters and all general purpose registers, including AX, restored to the same values as on entry.

All VIO functions within a session are serialized on a thread basis. That is, when a global subsystem receives control, it can safely assume that it is not called again from the same session until the current call has completed. Note, however, that VIO calls across different sessions are not serialized.

VioGlobalReg may only be issued during system initialization. After system initialization, VioGlobalReg returns ERROR_VIO_REGISTER. A globally registered subsystem is active for the life of the system.

If multiple global subsystems are registered, they are given control in the order that they are registered.

A globally registered subsystem receives control on VIO calls issued from all full-screen sessions except the hard error handler and DOS (real mode) sessions.

### C bindings

```
#define INCL_VIO

USHORT  rc = VioGlobalReg(ModuleName, EntryPoint, FunctionMask1,
                          FunctionMask2, 0);

PSZ          ModuleName;        /* Module name  */
PSZ          EntryPoint;        /* Entry point name */
ULONG        FunctionMask1;     /* Function mask 1  */
ULONG        FunctionMask2;     /* Function mask 2  */
LONG         0;                 /* Reserved (must be zero) */

USHORT           rc;            /* return code */
```

### MASM bindings

```
EXTRN VioGlobalReg:FAR
INCL_VIO            EQU 1

PUSH@ ASCIIZ ModuleName      ;Module name
PUSH@ ASCIIZ EntryPoint      ;Entry point name
PUSH  DWORD  FunctionMask1   ;Function mask 1
PUSH  DWORD  FunctionMask2   ;Function mask 2
PUSH  DWORD  0               ;Reserved (must be zero)
CALL  VioGlobalReg

Returns WORD
```

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

Permanent link:
**https://osfree.su/doku/doku.php?id=en:ibm:prcp:vio:globalreg&rev=1473917492**

Last update: **2016/09/15 05:31**