 This is part of **Family API** which allow to create dual-os version of program runs under OS/2 and DOS

Note: This is legacy API call. It is recommended to use 32-bit equivalent

2021/09/17 04:47 · prokushev · 0 Comments
2021/08/20 03:18 · prokushev · 0 Comments

VioRegister

This call registers an alternate video subsystem within a session.

Syntax

```
VioRegister (ModuleName, EntryPoint, FunctionMask1, FunctionMask2)
```

Parameters

- 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
31	VioPrtScToggle
30	VioEndPopUp
29	VioPopUp
28	VioSavRedrawUndo
27	VioSavRedrawWait
26	VioScrUnLock
25	VioScrLock
24	VioPrtSc
23	VioGetAnsi
22	VioSetAnsi
21	VioScrollRt
20	VioScrollLf

BIT	REGISTERED FUNCTION
19	VioScrollDn
18	VioScrollUp
17	VioWrtCellStr
16	VioWrtCharStrAtt
15	VioWrtCharStr
14	VioWrtTTY
13	VioWrtNCell
12	VioWrtNAttr
11	VioWrtNChar
10	VioReadCellStr
9	VioReadCharStr
8	VioShowBuf
7	VioSetMode
6	VioSetCurType
5	VioSetCurPos
4	VioGetPhysBuf
3	VioGetBuf
2	VioGetMode
1	VioGetCurType
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	Description
31-9	Reserved, set to zero
8	VioSetState
7	VioGetState
6	VioSetFont
5	VioGetCp
4	VioSetCp
3	VioGetConfig
2	VioGetFont
1	VioModeUndo
0	VioModeWait

Return Code

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

- 0 NO_ERROR
- 349 ERROR_VIO_INVALID_MASK
- 403 ERROR_VIO_INVALID_ASCIIZ
- 426 ERROR_VIO_REGISTER

- 430 ERROR_VIO_ILLEGAL_DURING_POPUP
- 465 ERROR_VIO_DETACHED
- 494 ERROR_VIO_EXTENDED_SG

Remarks

An alternate video subsystem must register which video calls it handles. The default OS/2 video subsystem is the Base Video Subsystem.

When any of the registered functions are called, control is routed to EntryPoint. When this routine is entered, four additional values (5 words) are pushed onto the stack.

The first value is the index number (Word) of the routine being called. The second value is a near pointer (Word). The third value is the caller's DS register(Word). The fourth value 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:

```
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
```

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

Index	Function
0	VioGetPhysBuf
1	VioGetBuf
2	VioShowBuf
3	VioGetCurPos
4	VioGetCurType
5	VioGetMode
6	VioSetCurPos
7	VioSetCurType
8	VioSetMode
9	VioReadCharStr
10	VioReadCellStr
11	VioWrtNChar
12	VioWrtNAttr
13	VioWrtNCell
14	VioWrtCharStr
15	VioWrtCharStrAtt
16	VioWrtCellStr

Index	Function
17	VioWrtTTY
18	VioScrollUp
19	VioScrollDn
20	VioScrollLf
21	VioScrollRt
22	VioSetAnsi
23	VioGetAnsi
24	VioPrtSc
25	VioScrLock
26	VioScrUnLock
27	VioSavRedrawWait
28	VioSavRedrawUndo
29	VioPopUp
30	VioEndPopUp
31	VioPrtScToggle
32	VioModeWait
33	VioModeUndo
34	VioGetFont
35	VioGetConfig
36	VioSetCp
37	VioGetCp
38	VioSetFont
39	VioGetState
40	VioSetState

When a registered function returns to the video router, the return code is interpreted as follows:

- Return code = 0 : No error. Do not invoke the corresponding Base Video Subsystem routine. Return to caller with Return code = 0.
- Return code = -1 : No error. Invoke the corresponding Base Video Subsystem routine. Return to caller with Return code = return code from Base Video Subsystem.
- Return code = error (not 0 or -1) : Do not invoke the corresponding Base Video Subsystem routine. Return to caller with Return code = error.

When an application registers a replacement for VioPopUp within a session, the registered routine is only invoked when that session is in the foreground. If VioPopUp is issued when that session is in the background, the OS/2 default routine is invoked.

An alternate video subsystem should be designed so the routines registered do not cause any hard errors when they are invoked. Otherwise, a system lockout occurs. Code and data segments of registered routines, that might be loaded from diskette, must be preloaded.

All VIO functions within a session are serialized on a thread basis. That is, when an alternate video subsystem receives control, it can safely assume that it is not called again from the same session until the current call has completed.

http://www.edm2.com/index.php/VioRegister

Family API		
DOS	Process Manager	DosBeep DosExit DosSleep DosExecPgm
	File Manager	DosChDir DosChgFilePtr DosClose DosDelete DosDupHandle DosMkDir DosMove DosQCurDir DosQCurDisk DosSetFileMode DosOpen DosQFileInfo DosRead DosQFileMode DosQFSInfo DosQVerify DosRmDir DosSelectDisk DosFindClose DosFindFirst DosFindNext DosSetFileInfo DosSetVerify DosWrite DosFileLocks DosSetFHandState DosNewSize DosBufReset DosQFHandState DosSetFSinfo
	Memory Manager	DosFreeSeg DosSubAlloc DosSubFree DosSubSet DosAllocHuge DosAllocSeg DosReallocHuge DosReallocSeg DosGetHugeShift DosCreateCSAlias
	NLS	DosCaseMap DosGetCtryInfo DosGetDBCSEv DosSetCtryCode DosGetCollate DosGetMessage DosInsMessage DosPutMessage
	Date and Time	DosSetDateTime DosGetDateTime
	Devices	DosDevConfig DosDevIOCtl DosDevIOCtl2
	Signals	DosHoldSignal DosSetSigHandler
	Misc	BadDynLink DosGetEnv DosGetMachineMode DosGetVersion DosError DosErrClass DosSetVec
KBD		KbdCharIn KbdFlushBuffer KbdGetStatus KbdSetStatus KbdStringIn KbdPeek
VIO		VioGetBuf VioGetConfig VioGetCurPos VioGetCurType VioGetPhysBuf VioReadCellStr VioReadCharStr VioScrollUp VioScrollDn VioScrollLf VioScrollRt VioScrUnLock VioSetCurPos VioSetCurType VioSetMode VioGetMode VioShowBuf VioWrtCellStr VioWrtCharStr VioWrtCharStrAtt VioWrtNAttr VioWrtNCell VioWrtNChar VioWrtTTY VioScrLock VioPopUp
Tools		BIND
Modules		DOSCALLS.DLL VIOCALLS.DLL KBDCALLS.DLL MSG.DLL
Libraries		API.LIB OS2386.LIB FAPI.LIB DOSCALLS.LIB SUBCALLS.LIB

2018/08/25 15:05 · prokushev · 0 Comments