 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

OS/2 Video Subsystem

Family API (OS/2 mode)

OS/2 Video Subsystem (VIO in short) aimed to provide text-mode and graphics-mode (full screen sessions only) video output. Original VIO supports 43 functions which mostly improvement of Video BIOS INT 10H API. Functions are much richer comparing to Video BIOS. It is supports Logical Video Buffer, which allow to have output in background sessions, while screen no accesable. Temporary video session allow to build message notifications system to handle errors and exceptions. Most of functions (41 function) can be replaced by user-defined functions by Alternate Video System. Alternate Video System can hanlde function and then return to caller or pass control lower, to Base Video System. You can register not all but some Alternate Video Subsystem functions, and user Base Video Subsystem for othe functions. Some VIO functions supported in Dual-mode applications by Family API. Under DOS functions simplier but powerfull yes. Many command line tools are Dual Mode applications, as well as many MS-DOS base Microsoft products also dual-mode applications. VIO also provides limited set of graphic mode support. VIO originalli is 16-bit and implemented starting from fist version of OS/2.

Application
VIOCALLS
VIO router
Alternate Video Subsystem
VIO router
Base Video Subsystem

OS/2 Video subsystem placed in VIOCALLS.DLL. Called VIO function passes control to internal VIO router function which, depends on registered functions replacement, routes call to Base Video Subsystem (BVSCALLS.DLL) or to registered Alternate Video Subsystem.

In pseudo code it looks like this

```
Application calls VioGetAnsi in VIOCALLS
VioGetAnsi calls VioRoute
if AVS registered then
  VioRoute calls AVS VioGetAnsi
  if AVS VioGetAnsi requires to call BVS then
```

```
    VioRouter calls BVS VioGetAnsi
else
  VioRouter calls BVS VioGetAnsi
Return to application
```

OS/2 1.1 introduced Program Manager and VIO was extended to support windowed OS/2 text-based sessions. This extended Vio was named as Advanced VIO.

Such route specific to OS/2 1.0 and 1.1. OS/2 1.2 includes additional functionality. Base Video Subsystem now uses Base Video Handlers to access hardware. Also Global Video Sybsystem hook was added. Base Vide Handler is a sort of video hardware abstraction layer. Global Video Subsystem allow to add notification hooks after VioRoute call.

Application
VIOCALLS
VIO router
Alternate Video Subsystem
VIO router
Base Video Subsystem
Base Video Handlers
Base Video Subsystem
VIO router
Global Video Subsystem

In pseudo code it looks like this

```
Application calls VioGetAnsi in VIOCALLS
VioGetAnsi calls VioRoute
if AVS registered then
  VioRoute calls AVS VioGetAnsi
  if AVS VioGetAnsi requires to call BVS then
    VioRouter calls BVS VioGetAnsi
else
  VioRouter calls BVS VioGetAnsi
If GVS registered then
  VioRouter calls GVS VioGetAnsi
Return to application
```

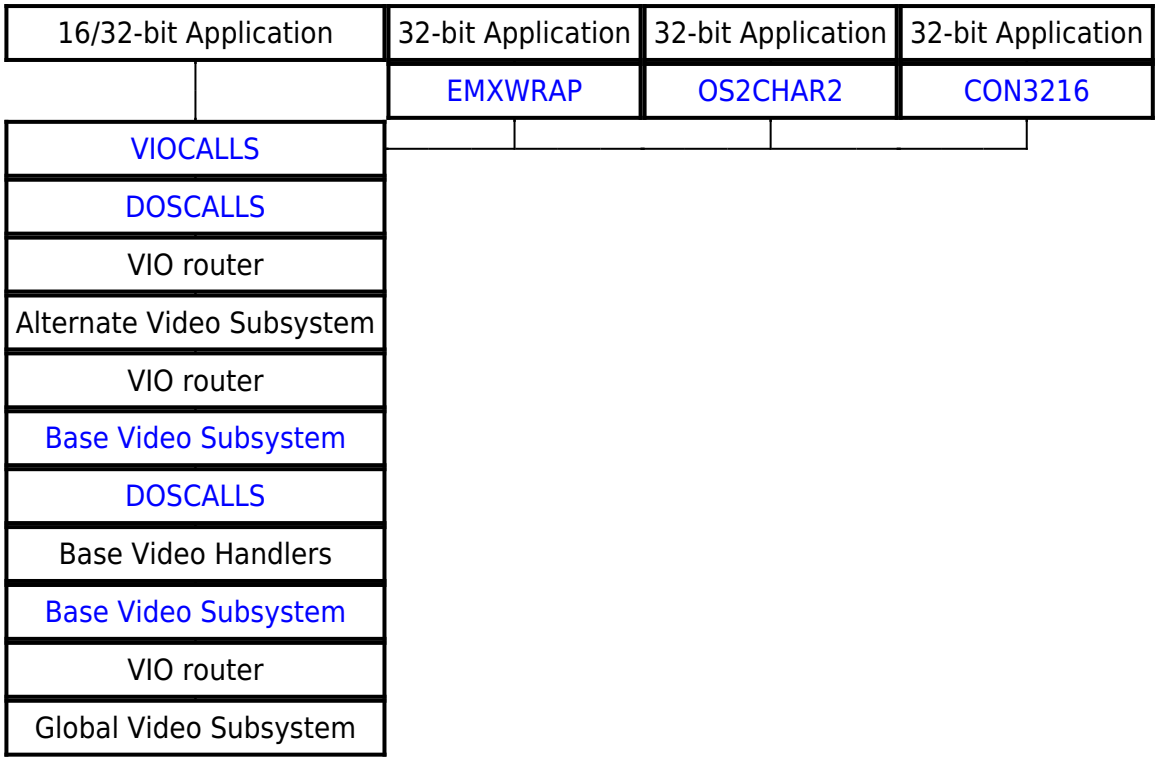
In OS/2 2.0 and later VIO still 16-bit. In late OS/2 versions (3.x?) actual code of Video subsystem and Base Video Subsysten was moved to DOSCALLS.DLL (see Fig). VIOCALLS and BVSCALLS is a forwarders to DOSCALLS.

```
Application calls VioGetAnsi in VIOCALLS
VIOCALLS forwards to DOSCALLs
VioGetAnsi calls VioRoute
if AVS registered then VioRoute calls AVS VioGetAnsi
```

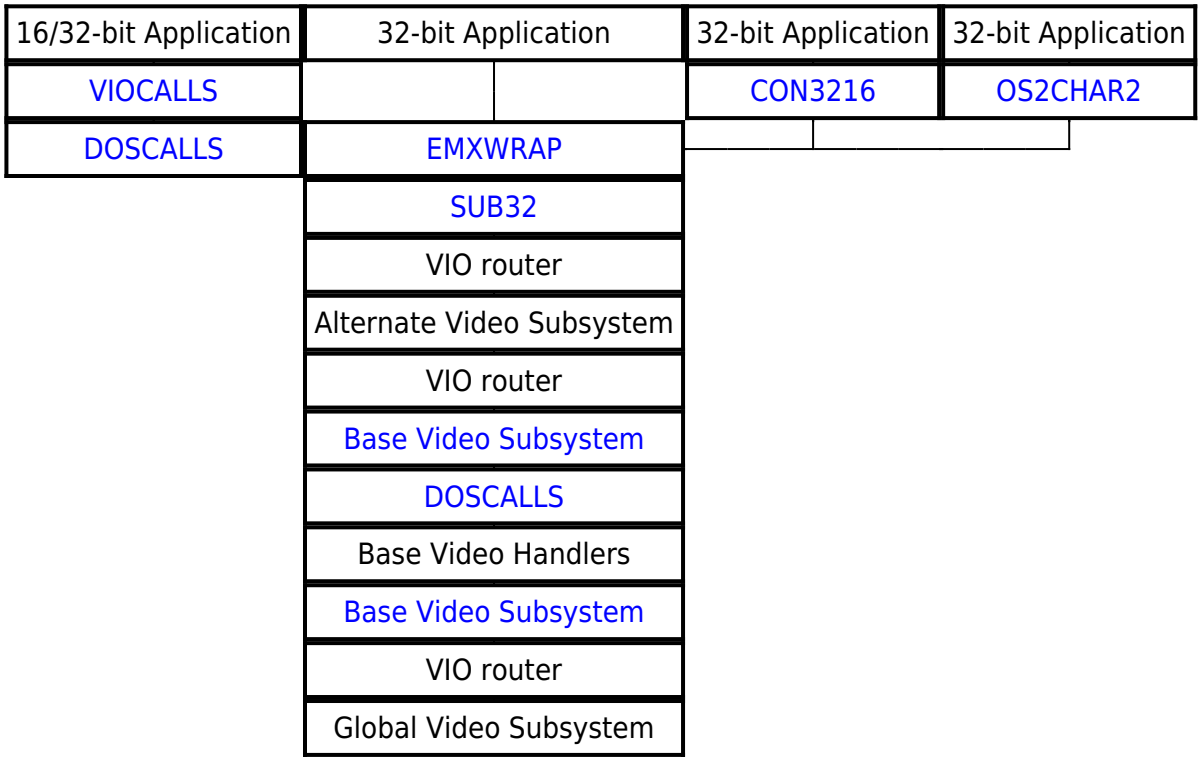
```
  if AVS VioGetAnsi requires to call BVS then VioRouter calls BVS VioGetAnsi
    BVSCALLS forwards to DOSCALLS
else VioRouter calls BVS VioGetAnsi
  BVSCALLS forwards to DOSCALLS

Return to application
```

As not part of official OS/2 distribution three independed versions of 32-to-16 bit wrapper subsystems was developed (See Fig). EXMWRAP.DLL was part of eComstation 1.0 and later releases.



osFree attempts to combine all aproaches and provide following callflow



Family API (DOS mode)

Under DOS original IBM Family API or JdeBP's Family API applications uses following call route

Application
API.LIB/FAPI.LIB/FAMAPI.LIB

Under HX DOS Extender OS/2 Emulation Family API applications uses following route

Application
VIOCALLS

Under DOS osFree Family API supports following route

Application
VIOCALLS
VIO router
Alternate Video Subsystem
VIO router
Base Video Subsystem