บทที่ 4

ภาคขับอุปกรณ์ของวินโดวส์

ภาคขับอุปกรณ์คือ (Dror,1994) ส่วนของโปรแกรมที่ทำงานในระดับล่างสุดที่คอยควบคุมฮาร์ดแวร์โดยมีหน้าที่แปลงความต้องการของโปรแกรมประยุกต์ในการขอใช้อุปกรณ์ฮาร์ดแวร์ เช่น"ต้องการ วาดเส้นลงบนจอภาพ" ให้ไปเป็นชุดคำสั่งที่เหมาะสมกับฮาร์ดแวร์ที่เชื่อมต่ออยู่ โดยลักษณะนี้ ภาคขับอุปกรณ์จะทำหน้าที่กันโปรแกรมประยุกต์ (หรือแม้แต่ระบบปฏิบัติการเอง) ออกไปจากรายละเอียดของฮาร์ดแวร์ที่ติดต่อด้วย ทำให้การออกแบบโปรแกรมประยุกต์ไม่ผูกติดกับฮาร์ดแวร์ (Device independent) และสามารถเพิ่มเติมฮาร์ดแวร์ระบบอื่นๆ ได้ง่าย

โดยเหตุที่ภาคขับอุปกรณ์มักทำงานในระดับล่างสุดเสมอ (เช่น ติดต่อกับหน่วยความจำโดยตรง หรือมีการใช้อินเตอร์รัพท์) จึงเป็นที่ยอมรับกันว่า มันเป็นส่วนขยายของระบบปฏิบัติการ และมักจะถูกจัดให้ทำงานในระดับเอกสิทธิ์สูงสุด เพื่อให้ภาคขับอุปกรณ์สามารถทำงานได้อย่างมีประสิทธิภาพ

ในวินโดวส์ 3.1 จะมีภาคขับอุปกรณ์อยู่ 2 ชนิด คือ ภาคขับอุปกรณ์ธรรมดา (Ordinary Device Driver) กับ ภาคขับอุปกรณ์เสมือน (Virtual Device Driver หรือ VxD) ทั้ง 2 ชนิดนี้แตกต่างกันอย่างสิ้นเชิง ดังจะได้กล่าวต่อไปนี้

ภาคขับอุปกรณ์ธรรมดา

เนื่องจากในระบบวินโดวส์ 3.1 วงแหวนป้องกันจะถูกใช้งานเพียง 2 วง (จากทั้งหมด 4 วง ที่ หน่วยประมวลผลกลางมีให้ใช้) โปรแกรมประยุกต์ DLL หรือแม้แต่บางส่วนของวินโดวส์เอง ทำงานอยู่ที่วงนอกซึ่งมีระดับเอกสิทธิ์ต่ำสุด เฉพาะส่วนแกนกลางที่สำคัญเท่านั้นที่ถูกจัดไว้วงใน (หมายเลขศูนย์) ซึ่งมีระดับเอกสิทธิ์สูงสุด

ภาคขับอุปกรณ์แบบธรรมดานี้ก็ทำงานในวงนอกสุดเช่นเดียวกับโปรแกรมประยุกต์ซึ่งนั่นหมายถึงโปรแกรมประยุกต์เองสามารถออกแบบให้ทำงานติดต่อกับอุปกรณ์ฮาร์ดแวร์ได้เช่นเดียวกับภาคขับอุปกรณ์

ภาคขับอุปกรณ์ชนิดนี้มักจะถูกออกแบบมาให้อยู่ในรูป DLL ทั้งนี้เพื่อให้สามารถให้บริการแก่โปรแกรมประยุกต์อื่น (รวมทั้งวินโดวส์เองด้วย) ได้หลายๆชุด ยิ่งไปกว่านั้นการใช้ DLL เป็นภาคขับอุปกรณ์ แทนที่จะติดต่อกับฮาร์ดแวร์โดยตรงจากภายในโปรแกรมประยุกต์ยังทำให้โปรแกรมประยุกต์ไม่ผูกติดกับฮาร์ดแวร์อีกด้วย

ในการติดต่อกับฮาร์ดแวร์นั้นภาคขับอุปกรณ์อาจมีการติดต่อกับหน่วยความจำ I/O พอร์ต หรือแม้แต่ใช้อินเตอร์รัพท์จากภายนอก ภาคขับอุปกรณ์ที่มีการใช้งานอินเตอร์รัพท์จะต้องออกแบบเป็นพิเศษ เพื่อรองรับการประมวลผลในสภาวะเช่นนี้ เช่น อาจต้องกำหนดให้อยู่ในหน่วยความจำคงที่ (Fixed memory) หรืออาจต้องออกแบบให้ถูก เรียกใช้ซ้ำได้ (Reentrance) ประการสุดท้ายห้ามเรียกใช้ API ของวินโดวส์เนื่องจาก API ส่วนใหญ่ไม่สามารถเรียกใช้ซ้ำได้

ในวินโดวส์ 3.1 ภาคขับอุปกรณ์ธรรมดา สามารถแบ่งออกเป็น 2 กลุ่ม คือ

1. ภาคขับอุปกรณ์มาตรฐาน (Standard Device Driver) ในวินโดวส์มีการสร้างภาคขับอุปกรณ์สำหรับอุปกรณ์พื้นฐานของระบบเช่น จอภาพและการ์ดแสดงผล แป้นพิมพ์ เมาส์ พอร์ตสื่อสาร และอื่นๆ ภาคขับอุปกรณ์เหล่านี้จัดอยู่ในกลุ่มภาคขับอุปกรณ์มาตรฐาน ซึ่งให้มากับวินโดวส์อยู่แล้ว

2. ภาคขับอุปกรณ์แบบติดตั้งได้ (Installable Device Driver) วินโดวส์ 3.1 ได้เพิ่มการรองรับภาคขับอุปกรณ์ชนิดใหม่ที่เรียกว่า เป็นภาคขับอุปกรณ์แบบติดตั้งได้ เพื่อรองรับอุปกรณ์ชนิดอื่นๆ ที่ไม่ใช่อุปกรณ์มาตรฐาน (แป้นพิมพ์ เมาส์ หรืออื่นๆ ที่กล่าวมา) ถ้ามีการเพิ่มฮาร์ดแวร์ชนิดใหม่เข้าไปในระบบ วินโดวส์กำหนดให้เราต้องติดตั้ง DLL เข้าไปเพื่อทำหน้าที่เป็นภาคขับอุปกรณ์สำหรับฮาร์ดแวร์นั้น ในความเป็นจริงภาคขับอุปกรณ์สามารถเขียนขึ้น ให้อยู่ในรูปแบบใดก็ได้ ไม่จำเป็นต้องเป็น DLL แต่วินโดวส์เห็นความจำเป็นในการมีมาตรฐาน เพื่อติดตั้งเพิ่มเติมภาคขับอุปกรณ์ใหม่ๆ จึงมีการกำหนดวิธีการ (Protocol) ติดต่อระหว่างวินโดวส์ ภาคขับอุปกรณ์ที่ติดตั้งเข้าไป และโปรแกรมประยุกต์ที่ทำการขอใช้บริการของภาคขับอุปกรณ์นั้นๆ

ภาคขับอุปกรณ์เสมือน

โดยเหตุที่ในภาวะเสริม วินโดวส์จะทำการสร้างเครื่องจักรเสมือนให้กับโปรแกรมประยุกต์แต่ละชนิด (ทั้งโปรแกรมประยุกต์แบบวินโดวส์ และโปรแกรมประยุกต์แบบดอส) ปัญหาที่อาจเกิดขึ้นได้ คือมีการแย่งกันใช้ฮาร์ดแวร์ตัวเดียวกันจากเครื่องจักรเสมือนหลายๆ ตัว

วินโดวส์แก้ปัญหานี้โดยการสร้างภาคขับอุปกรณ์เสมือนเพื่อจำลองฮาร์ดแวร์ให้กับเครื่องจักรเสมือนแต่ละตัว โดยอาศัยการดักจับการติดต่อกับอุปกรณ์ภายนอกของโปรแกรมประยุกต์ การทำเช่นนี้จะทำให้เสมือนกับโปรแกรมประยุกต์ทุกๆโปรแกรมมีฮาร์ดแวร์ของตนเอง แต่ในความเป็นจริงแล้วภาคขับอุปกรณ์ของแต่ละเครื่องจักรเสมือนจะทำการติดต่อกับภาคขับอุปกรณ์เสมือน แล้วในอีกด้านหนึ่งภาคขับอุปกรณ์เสมือนจะติดต่อกับฮาร์ดแวร์จริงอีกทอดหนึ่ง

ภาคขับอุปกรณ์เสมือนจะถูกสร้างเป็นโปรแกรม 32 บิต แบบแฟลตโมเดล ทำงานที่ระดับเอกสิทธิ์สูงสุด และสามารถทำหน้าที่อย่างอื่นได้อีกมากมาย นอกจากการจำลองฮาร์ดแวร์

จากความจริงที่ว่าภาคขับอุปกรณ์ของวินโดวส์ทำงานที่ระดับเอกสิทธิ์ต่ำ ดังนั้นการติดต่อกับอุปกรณ์ภายนอกจึงไม่สามารถทำได้โดยตรง จะมีขั้นตอนที่แอบแฝงมากมาย โดยเฉพาะการอินเตอร์รัพท์ ในการใช้งานบางอย่างอาจพบว่า ผู้ออกแบบจะต้องหันมาใช้ภาคขับอุปกรณ์ที่สามารถทำงานในระดับเดียวกับภาคขับอุปกรณ์เสมือน ทั้งนี้เพื่อเพิ่มประสิทธิภาพของระบบ นั่นคือ ต้องออกแบบภาคขับอุปกรณ์ที่ทำงานในระดับเอกสิทธิ์ 0 นั่นเอง

การออกแบบภาคขับอุปกรณ์

ในการออกแบบและจัดสร้าง ภาคขับอุปกรณ์สำหรับฮาร์ดแวร์ใหม่ๆนั้นควรจะใช้ชุดพัฒนาภาคขับอุปกรณ์ของ บริษัทไมโครซอฟต์ (Microsoft Device Driver Development Kit : DDK) ซึ่งภายในประกอบด้วยเครื่องมือ สำหรับ พัฒนาพิเศษ เช่น MASM และ LINK สำหรับสร้างโปรแกรมแบบ 32 บิต แฟลตโมเดล, เครื่องมือช่วยในการตรวจสอบหาความผิดพลาด (Debug) ต่างๆ, เครื่องมือสำหรับสร้างภาคขับอุปกรณ์สำหรับเครื่องพิมพ์พร้อมทั้งชุดทดสอบการทำงาน สิ่งที่มีค่ามากสำหรับนักพัฒนาภาคขับอุปกรณ์ก็คือ ตัวอย่างของภาคขับอุปกรณ์มากมายที่ให้มากับ DDK (พร้อมทั้งต้นฉบับโปรแกรม)

ตัวอย่างเหล่านี้เป็นแนวทางที่ดีในการศึกษาการออกแบบ พัฒนาและปรับปรุงภาคขับอุปกรณ์ต่างๆ ยิ่งไปกว่านั้นยังเป็นตัวอย่างที่ดีที่แนะนำวิธีการที่ควรสร้าง (และไม่ควรสร้าง) ภาคขับอุปกรณ์ในลักษณะต่างๆ

แต่การใช้ตัวอย่างเหล่านี้ในการศึกษาก็มีข้อต้องระวังอย่างหนึ่งคือ ภาคขับอุปกรณ์หลายตัวในชุด DDK มีการออกแบบที่ค่อนข้างล้าสมัย และใช้เทคนิคที่ยากจะเข้าใจหลายอย่าง (เพราะถูกเขียนขึ้นหลายปีก่อน และมีการเพิ่มเติม เทคนิคพิสดารต่างๆ เข้าไปมากมาย) ดังนั้น หากนักพัฒนาต้องการภาคขับอุปกรณ์ที่มีการออกแบบที่ดีก็ให้ระวังในจุดนี้ไว้ด้วย

ในการวิจัยครั้งนี้ผู้วิจัยไม่ได้ออกแบบภาคขับอุปกรณ์โดยใช้ชุดพัฒนา DDK แต่เลือกที่จะออกแบบขึ้นมาเองโดยเฉพาะ ทั้งนี้เนื่องจากชุดพัฒนา DDK มีราคาแพงมาก อีกทั้งยังพบว่าสำหรับงานวิจัยนี้ยังไม่จำเป็นมากนัก แต่อย่างไรก็ตามภาคขับอุปกรณ์ที่ได้จากงานวิจัยนี้ก็ใช้มาตรฐานการออกแบบเดียวกันกับภาคขับอุปกรณ์ใน DDK ซึ่งมีรายละเอียดดังนี้

การออกแบบภาคขับอุปกรณ์แบบติดตั้งได้

เรามักกำหนดภาคขับอุปกรณ์แบบนี้ให้เป็น DLL ชนิดหนึ่ง, ให้ทำหน้าที่ควบคุมฮาร์ดแวร์ที่ต้องการ โดยที่วินโดวส์กำหนดไว้ว่า ต้องมีฟังค์ชันส่งออกไว้เรียกกลับ (Export Callback Function) ที่ตั้งชื่อว่า "DriverProc" ซึ่งจะทำหน้าที่คอยรับข้อความจากวินโดวส์ หรือโปรแกรมประยุกต์ที่จะใช้ภาคขับอุปกรณ์นี้, ทำนองเดียวกับที่วินโดวส์โพรซีเยอร์ได้รับข้อความนั่นเอง ตามรูปแบบ ดังต่อไปนี้

```
LRESULT CALLBACK DriverProc (dwDrvID, hDrv, msg,lParam1, lParam2);
DWORD     dwDrvID;     // driver ID
HDRVR     hDrv;        // driver handle
UNIT      msg;         // message ID
LPARAM    lParam1;     // parameter 1 - msg specific
LPARAM    lParam2;     // parameter 2 - msg specific
Returns:
```

Message - specific value, but generally TRUE (or non-zero messagespecific value) if successful and FALSE if failed

ภาคขับอุปกรณ์จะต้องผ่านข้อความที่ไม่ใช้ไปให้ API ที่ชื่อ DefDriverProc เพื่อให้ทำการประมวลผลต่อ , ทำนองเดียวกับที่วินโดวส์โพรซีเยอร์ต้องเรียก DefWindowProc รูปแบบการใช้งานเป็นดังนี้

```
LRESULT WINAPI DefDriverProc(dwDrvID, hDrv, msg, lParam1, lParam2);
DWORD       dwDrvID;     // driver ID
HDRVR       hDrv;        // driver handle
UNIT        msg;         // message ID
LPARAM      lParam1;     // parameter 1 - msg specific
LPARAM      lParam2;     // parameter 2 - msg specific
Returns:
        Message specific value, but generally returns TRUE.
```

สำหรับการติดต่อกับภาคขับอุปกรณ์นั้น วินโดวส์เตรียมไว้ให้ทั้งในรูปของ API และข้อความดังนี้

API สำหรับติดต่อสื่อสารกับภาคขับอุปกรณ์

ก่อนที่โปรแกรมประยุกต์จะสามารถใช้งานภาคขับอุปกรณ์ได้นั้น จะต้องทำการเปิดภาคขับอุปกรณ์ก่อนโดยใช้ API ชื่อ OpenDriver รูปแบบการใช้งานเป็นดังนี้

```
HDRVR WINAPI OpenDriver (szDriverName, szSectionName, lParam2);
LPCSTR      szDriverName;     // driver filename or entry in SYSTEM.INI.
LPCSTR      szSectionName;    // section name in SYSTEM.INI where DriverName
                              // appear or NULL for [drivers]
LPARAM      lParam2;          // driver specific parameter (pass to driver as /
                              // lParam2 in DRV_OPEN message)
Returns :
        If successfull, hDrv otherwise FALSE.
```

OpenDriver จะทำการเปิดอินสแตนท์ (Instance) ใหม่ของภาคขับอุปกรณ์ที่กำหนดโดยดูจากชื่อไฟล์ของภาคขับอุปกรณ์ หรือชื่อเอนทรีและเซ็คชัน (Entry and Section) ในไฟล์ SYSTEM.INI

เมื่อเริ่มเปิดภาคขับอุปกรณ์ในครั้งแรก วินโดวส์จะโหลดภาคขับอุปกรณ์ลงสู่หน่วยความจำ แล้วส่งข้อความ DRV_LOAD, DRV_ENABLE และ DRV_OPEN (เรียงตามลำดับ) ไปให้กับ ภาคขับอุปกรณ์นั้น จากนั้นจะให้ "ค่าการใช้" (USE_COUNT) ของภาคขับอุปกรณ์เป็น "1" ในการเปิดครั้งต่อไปของภาคขับอุปกรณ์ตัวนี้ วินโดวส์จะเพิ่มค่าการใช้ขึ้นครั้งละ "1" และส่งเฉพาะข้อความ DRV_OPEN ให้แก่ภาคขับอุปกรณ์นั้น

เมื่อโปรแกรมประยุกต์ไม่ต้องการใช้ภาคขับอุปกรณ์อีกก็ควรปิดภาคขับอุปกรณ์ โดยการใช้ API CloseDriver รูปแบบการใช้งานเป็นดังนี้

```
LRESULT WINPAI CloseDriver (hDrv, lParam1, lParam2);
HDRV        hDrv;       // driver handle.
LPARAM      lParam1;    // driver specific parameter, pass in DRV_CLOSE message.
LPARAM      lParam2 ;   // driver specific parameter, pass in DRV_CLOSE message.
Returns :
        If successful , TRUE otherwise FALSE.
```

CloseDriver จะทำการปิดอินสแตนท์ของภาคขับอุปกรณ์ที่กำหนด โดยการส่งข้อความ DRV_CLOSE ให้กับภาคขับอุปกรณ์ แล้วลดค่าการใช้ลง "1" เมื่อค่าการใช้ถูกลดลงจนเป็นศูนย์ วินโดวส์จะส่งข้อความ DRV_DISABLE และ DRV_FREE ให้กับภาคขับอุปกรณ์และปลดภาคขับอุปกรณ์ออกจากหน่วยความจำ

หลังจากที่โปรแกรมประยุกต์เปิดภาคขับอุปกรณ์ขึ้นมาแล้ว การติดต่อสื่อสารทำได้โดยการส่งข้อความถึงกันโดยใช้ API SendDriverMessage รูปแบบการใช้งานเป็นดังนี้

```
LRESULT WINAPI SendDriverMessage (hDrv, msg, lParam1, lParam2);
HDRV        hDrv;       // driver handle.
UNIT        msg;        //message ID.
LPARAM      lParam1;    // driver specific parameter, pass in DRV_CLOSE message.
LPARAM      lParam2;    // driver specific parameter, pass in DRV_CLOSE message.
Returns :
```

The conventional is to return a non-zero message specific value for success and FALSE for failure.

ข้อความในระบบภาคขับอุปกรณ์แบบติดตั้งได้

มีลักษณะการใช้งานข้อความอยู่ 3 ช่วง คือ

1. ข้อความ ที่มีค่าน้อยกว่า DRV_RESERVED ลงมา หมายถึง เป็นข้อความที่จะใช้ร่วมกันในภาคขับอุปกรณ์ทุกตัว
2. ข้อความ ที่มีค่าระหว่าง DRV_RESERVED จน ถึง DRV_USER หมายถึงข้อความที่ใช้ในการกำหนดขั้นตอนพิธีการ (Protocol) ในการติดต่อกัน เช่น ที่ใช้ในภาคขับอุปกรณ์มีเดียคอนโทรลอินเตอร์เฟซ (Media Control Interface)
3. ข้อความ ที่มีค่ามากกว่า DRV_USER ขึ้นไป หมายถึงข้อความเฉพาะที่ตัวภาคขับอุปกรณ์ใช้เอง

ในงานวิจัยนี้จะกล่าวถึงเฉพาะข้อความในแบบแรกเท่านั้น ซึ่งข้อความเหล่านี้จะถูกส่งมาจากวินโดวส์เอง ในระหว่างทำการติดตั้งภาคขับอุปกรณ์ และมีบางส่วนถูกส่งมาจากโปรแกรมประยุกต์ที่ทำหน้าที่ติดตั้งภาคขับอุปกรณ์นั้นๆ เช่น คอนโทรลพาเนล (Control Panel) เป็นต้น

1. ข้อความที่ส่งโดยวินโดวส์สู่ภาคขับอุปกรณ์

ข้อความแรกที่ส่งให้แก่ภาคขับอุปกรณ์หลังการโหลดลงในหน่วยความจำคือ DRV_LOAD, เมื่อได้รับข้อความนี้โดยทั่วๆไป ภาคขับอุปกรณ์จะทำการอ่านค่าติดตั้ง (Configuration Setting) ที่อยู่ในไฟล์ SYSTEM.INI และดำเนินการกำหนดค่าเริ่มต้นครั้งแรกของโครงสร้างภายในที่จำเป็น รูปแบบของข้อความเป็นดังนี้

Load Driver

msg : DRV_LOAD

lParam10

lParam20

returns :

If successful TRUE otherwise FALSE (if DRV_LOAD fails Windows will unload the driver with out sending it a DRV_FRRE message

วินโดวส์จะส่งข้อความ DRV_ENABLE ไปให้ภาคขับอุปกรณ์ หลังจากที่ภาคขับอุปกรณ์เสร็จสิ้นการเตรียมตัวหลังจากถูกโหลด หรือ ทุกครั้งที่กลับเข้าสู่วินโดวส์ เมื่อจบการรันโปรแกรมประยุกต์ของดอสในสแตนดาร์ดโหมด (รายละเอียดเกี่ยวกับเรื่องนี้จะได้กล่าวต่อไป) โดยปกติ เมื่อได้รับข้อความนี้ ภาคขับอุปกรณ์จะทำการกำหนดค่าฮาร์ดแวร์ และติดตั้งระบบอินเตอร์รัพท์ รูปแบบของข้อความเป็นดังนี้

Enable Driver

msg : DRV_ENABLE

lParam10

lParam20

Returns :

Value ignored.

สำหรับข้อความ DRV_OPEN วินโดวส์จะส่งให้กับภาคขับอุปกรณ์ทุกครั้งที่มีการสร้างอินสแตนท์ใหม่ (โดยการเรียก OpenDriver) ซึ่งอาจเป็น อินสแตนท์แรก หรือต่อมาก็ได้ เมื่อได้รับข้อความนี้ ภาคขับอุปกรณ์ จะทำการจัดเตรียมพื้นที่สำหรับเก็บข้อมูลเฉพาะของอินสแตนท์นั้นๆ รูปแบบของข้อความเป็นดังนี้

Open Driver Instance

msg : DRV_OPEN

lParam1 Point to an ASCII string containing parameters found after the device driver's filename in SYSTEM.INI (NULL if none).

lParam2 lParam2 from OpenDriver API.

Returns :

If successful , dwDrvID otherwise FALSE.

Note : DefDriverProc returns FALSE (DRV_OPEN failed !)

ถ้าไม่มีข้อผิดพลาดในการทำงานหลังจากได้รับ DRV_OPEN ข้อความให้ภาคขับอุปกรณ์ส่งค่าหมายเลขที่ใช้ในการระบุอินสแตนท์นี้ในรูป "dwDrvID" หมายเลขนี้จะถูกใช้โดยภาคขับอุปกรณ์ในการระบุข้อมูลของแต่ละอินสแตนท์ วินโดวส์จะทำการเก็บหมายเลขนี้ไว้ภายใน และจะสร้างหมายเลขเฉพาะที่เรียกว่า hDdrv ส่งคืนให้แก่โปรแกรมประยุกต์ เมื่อโปรแกรมประยุกต์ต้องการส่งข้อความถึงภาคขับอุปกรณ์ วินโดวส์จะทำการเพิ่มหมายเลข dwDrvID ให้ด้วย (ใช้ประกอบกับ hDrv) ดังนั้น เราจะพบว่าในระดับโปรแกรมประยุกต์ ภาคขับอุปกรณ์แต่ละอินสแตนท์จะมีหมายเลขแฮนเดิลเฉพาะ และในขณะเดียวกับตัวภาคขับอุปกรณ์เองก็มีการใช้กลไกอย่างอื่น (dwDrvID) ที่ใช้ในการระบุอินสแตนท์ของตนเอง, ค่าหมายเลขนี้ใช้เองภายในภาคขับอุปกรณ์ และอาจมีค่าซ้ำกันก็ได้วินโดวส์ไม่ได้ห้ามไว้

แต่ในการส่งข้อความดังต่อไปนี้ คือ DRV_LOAD, DRV_ENABLE, DRV_OPEN DRV_DISABLE และ DRV_FREE วินโดวส์จะไม่กำหนดค่า dwDrvID (ให้เป็นศูนย์) ส่วนในการส่งข้อความ DRV_OPEN ค่า dwDrvID จะมีค่าเท่ากับ hDrv

ให้สังเกตว่า API DefDriverProc จะให้ค่า FALSE ซึ่งเป็นการบังคับให้ ภาคขับอุปกรณ์ต้องรับข้อความ DRV_OPEN มิฉะนั้นการเรียก OpenDriver วินโดวส์จะส่งข้อความ DRV_CLOSE มาที่ภาคขับอุปกรณ์พร้อมทำการลดค่าการใช้ลง "1" เมื่อค่าการใช้ลดลงจนเป็นศูนย์ วินโดวส์จะปลดภาคขับอุปกรณ์ออกจากจากหน่วยความจำ (หลังที่ได้ส่งข้อความ DRV_DISABLE และ DRV_FREE ไปแล้ว)

ในการปิดอินสแตนท์ของภาคขับอุปกรณ์วินโดวส์จะส่งข้อความ DRV_CLOSE รูปแบบของข้อความเป็นดังนี้

Close Driver Instance

msg : DRV_CLOSE

lParam1 lParam1 from CloseDriver

lParam2 lParam2 from CloseDriver

Returns :

If successful , TRUE else FALSE.

Note : DefDriverProc returns FALSE (DRV_CLOSE failed !)

ภาคขับอุปกรณ์จะต้องทำการยกเลิกพื้นที่สำหรับอินสแตนท์ที่ปิดแล้วเอง (จะเห็นว่า ถ้าต้องการออกแบบ ภาคขับอุปกรณ์ที่รองรับการทำงานแบบหลายอินสแตนท์ การจัดการเกี่ยวกับอินสแตนท์ดาต้า จะต้องรับผิดชอบโดยภาคขับอุปกรณ์เองไม่เหมือนกับโปรแกรมประยุกต์ที่การจัดการส่วนนี้เป็นไปโดยอัตโนมัติ) และเช่นเดียวกันกับ DRV_OPEN ข้อความ ภาคขับอุปกรณ์จะต้องรับข้อความนี้ไปดำเนินการ มิฉะนั้นการเรียกใช้ CloseDriver จะให้ผลลัพธ์เป็น FALSE ซึ่งจะทำให้อินสแตนท์นั้นไม่ถูกยกเลิกและค่าการใช้ก็ไม่ถูกลดค่าลงด้วย

ส่วนข้อความ DRV_DISABLE จะถูกส่งให้แก่ภาคขับอุปกรณ์ในตอนที่ภาคขับอุปกรณ์กำลังจะถูกปลดจากหน่วยความจำ หรือในขณะที่วินโดวส์กำลังจะเข้าประมวลผลโปรแกรมประยุกต์ของดอส ในสแตนดาร์ดโหมด ซึ่งเป็นผลมาจากการที่วินโดวส์จะหยุดทำงานชั่วคราว ดังนั้นภาคขับอุปกรณ์ต่างๆ ในวินโดวส์ก็ต้องหยุดชั่วคราวด้วย (รีเซ็ตฮาร์ดแวร์ และหยุดระบบอินเตอร์รัพท์)

เมื่อออกจากโปรแกรมประยุกต์ของดอสแล้ว วินโดวส์จะส่งข้อความ DRV_ENABLE เพื่อแจ้งให้ภาคขับอุปกรณ์เริ่มต้นทำงานต่อได้ (เซ็ตฮาร์แวร์ และติดตั้งระบบอินเตอร์รัพท์) มีข้อต้องระวังอยู่ว่าในช่วงระหว่างได้รับ DRV_DISABLE จนถึง DRV_ENABLE นั้นวินโดวส์อาจเคลื่อนย้ายภาคขับอุปกรณ์ออกจากหน่วยความจำไปแล้วก็ได้

ในบางกรณีภาคขับอุปกรณ์อาจได้รับข้อความ DRV_DISABLE ในระหว่างการทำงานปกติ เช่น ในระหว่างการปรับฮาร์ดแวร์อยู่เป็นต้น ในกรณีเช่นนี้ ให้ภาคขับอุปกรณ์หยุดการทำงานก่อนชั่วคราวจนกว่าจะได้รับข้อความ DRV_ENABLE อีกครั้ง รูปแบบของข้อความเป็นดังนี้

Disable Driver

msg : DRV_DISABLE

lParam10

lParam20

Return :

Value ignored.

ข้อความสุดท้ายที่วินโดวส์จะส่งให้แก่ภาคขับอุปกรณ์ก่อนที่จะยกเลิกภาคขับอุปกรณ์นั้น คือ DRV_FREE ซึ่งเมื่อได้รับข้อความนี้ภาคขับอุปกรณ์จะทำการเคลียร์ตัวเองขั้นสุดท้ายตามที่จำเป็น รูปแบบของข้อความเป็นดังนี้

Free Driver

msg : DRV_FREE

lParam10

lParam20

returns

Value ignored

ในกรณีที่มีโปรแกรมประยุกต์เลิกการทำงานวินโดวส์จะส่งข้อความ DRV_EXITAPPLICATION ไปยังภาคขับอุปกรณ์ที่เปิดอยู่ทุกตัว รูปแบบของข้อความเป็นดังนี้

Application exit

msg : DRV_EXITAPPLICATION

lParam1 Type of exit, possible values are

1) DRV_NORMALEXIT

2) DRV_ABNORMALEXIT

lParam20

Returns :

Value ignored.

ในกรณีที่ตัววินโดวส์เองกำลังจะจบการทำงาน วินโดวส์จะส่งข้อความ DRV_EXITSESSION ไปยังภาคขับอุปกรณ์ที่เปิดอยู่ทุกตัว ภาคขับอุปกรณ์เหล่านั้นจะได้รับข้อความนี้ก่อนที่จะถูกยกเลิกการทำงาน และในขณะที่สภาพแวดล้อมทุกอย่างยังใช้งานอยู่ได้ รูปแบบของข้อความเป็นดังนี้

Windows session terminates

msg : DRV_EXITSESSION

lParam10

lParam20

Returns :

Value ignored.

สำหรับระบบที่สนับสนุนระบบการจัดการพลังงาน (Advance Power Management : APM) ภาคขับอุปกรณ์จะได้รับข้อความ DRV_POWER เมื่อระบบกำลังจะเข้าหรือออกจากภาวะประหยัดพลังงาน รูปแบบของข้อความเป็นดังนี้

Power management notification

msg : DRV_POWER

lParam1 Event description; possible values are

1) PWR_SUSPENDREQUEST The system is about to enter suspend mode.

2) PWR_SUSPENDRESUME The system is about to resume operation after being in suspended mode.

3) PWR_CRITICALRESUME The system is about to resume operation after being in suspended mode. Suspended mode was entered without frist sending a PWR_SUSPENDREQUEST message to the driver.

lParam2 0

Returns :

For PWR_SUSPENDRESUME and PWR_CRITICALRESUME, the return value is ignored. For PWR_SUSPENDREQUEST, possible return value are

1) PWR_OK OK to enter suspended mode

2) PWR_FAIL Do not enter suspended mode

2. ข้อความที่ส่งโดยโปรแกรมที่ทำการติดตั้งภาคขับอุปกรณ์

ข้อความดังต่อไปนี้เป็นกลุ่มที่จะถูกส่งไปยังภาคขับอุปกรณ์ โดยโปรแกรมที่ทำหน้าที่ติดตั้งภาคขับอุปกรณ์ เช่น คอนโทรลพาแนล เป็นต้น และเนื่องจากข้อความเหล่านี้ถูกส่งออกมาจากโปรแกรมประยุกต์ (มิใช่วินโดวส์) ดังนั้นค่าพารามิเตอร์ต่างๆ จึงอาจจะแตกต่างกันไปในแต่ละโปรแกรม ซึ่งทำให้ภาคขับอุปกรณ์ที่รับข้อความนั้นต้องมีการตรวจสอบพารามิเตอร์ก่อนนำไปใช้เสมอ เช่น การตรวจดูว่าค่าของตัวชี้ตำแหน่งถูกต้อง เป็นต้น

เมื่อมีการติดตั้งภาคขับอุปกรณ์โปรแกรมประยุกต์ที่เป็นผู้ติดตั้งจะทำการส่งข้อความ DRV_INSTALL ตามปกติเมื่อภาคขับอุปกรณ์ได้รับข้อความ ก็จะทำการติดตั้งตัวเอง และปรับปรุงค่าในไฟล์ SYSTEM.INI ตามจำเป็น รูปแบบของข้อความเป็นดังนี้

Install Device Driver

msg : DRV_INSTALL

lParam1 0

lParam2 Optional configuration information. The Control Panel Drivers program passes a pointer to a DRVCONFIGINFO structure.

Returns :

Possible values are

1) DRVCNF_OK Installation OK.

2) DRVCNF_CANCEL Installation failed. The Control Panel Drivers installation program seems to treat this same as a DRVCNF_OK.

3) DRVCNF_RESTART Installation will take effect after Windows is restarted.

สำหรับโครงสร้างข้อมูล DRVCONFIGINFO มีรูปแบบดังนี้

```
typdef struct tagDRVCONFIGINFO
{
        DWORD       dwDCSize;           // size of structure.
        LPCSTR      lpszDCISectionName; // section name.
        LPCSTR      lpszDCIAliasName;   // entry name.
} DRVCONFIGINFO;
```

ก่อนการติดตั้งโปรแกรมประยุกต์ที่ทำการติดตั้งภาคขับอุปกรณ์จะทำการส่งข้อความ DRV_QUERYCONFIGURE มาเพื่อสอบถามว่า ภาคขับอุปกรณ์นี้เป็นแบบที่ปรับแต่งได้หรือไม่ ถ้าได้ให้ตอบกลับเป็น TRUE รูปแบบของข้อความเป็นดังนี้

Query if driver is user-configurable

msg : DRV_QUERYCONFIGURE

lParam1 Not used by Control Panel.

lParam2 Not used by Control Panel.

Returns :

TRUE if the driver supports user configuration, FALSE if the driver does not.

Note: DefDriverProc returns FALSE.

ภาคขับอุปกรณ์บางตัวสนับสนุนลักษณะการปรับแต่งโดยให้ผู้ใช้เลือกได้ ซึ่งมักจะทำโดยการใช้ไดอลอกบอกซ์ (Dialog box) ซึ่งจะแสดงขึ้นเมื่อได้รับข้อความ DRV_CONFIG ซึ่งแตกต่างกับ DRV_QUERYCONFIGURE ตรงที่ DRV_CONFIG นั้นใช้เมื่อโปรแกรมประยุกต์ แน่ใจแล้วว่าภาคขับอุปกรณ์ที่กำลังจะติดตั้งนี้ปรับแต่งได้ โดยทราบจากผลการตอบรับ DRV_QUERYCONFIGURE

ในบางครั้งผู้ใช้อาจขอปรับแต่งภาคขับอุปกรณ์ใหม่ระหว่างการทำงาน ซึ่งก็สามารถทำได้โดยการส่งข้อความ DRV_CONFIG

ตามปกติภาคขับอุปกรณ์มักจะใช้ลักษณะการปรับแต่งเช่นนี้กับคุณสมบัติที่ต้องการให้ผู้ใช้เปลี่ยนแปลงได้ เช่น หมายเลขพอร์ต หรือ อินเตอร์รัพท์ เป็นต้น ส่วนคุณสมบัติอื่นๆ ที่ผู้ใช้ไม่ควรปรับแต่งควรบรรจุอยู่ในการตอบรับข้อความ DRV_INSTALL แทนที่จะใช้ DRV_CONFIG รูปแบบของข้อความเป็นดังนี้

Manually Configure a Driver

msg : DRV_CONFIG.

lParam1 LOWORD (lParam1) == hWindows of the configuration dialog box

lParam2 Optional configuration information. The Control Panel Drivers program passes a pointer to a DRVCONFIGINFO structure.

Returns :

Possible values are

1) DRVCNF_OK Configuration OK.

2) DRVCNF_CANCEL Configuration failed. The Control Panel Drivers configuration program seems to treat this the same as a DRVCNF_OK.

3) DRVCNF_RESTART Configuration will take effect after Windows is restarted.

สุดท้าย เมื่อจะมีการปลดภาคขับอุปกรณ์ออกจากรายการที่ติดตั้งไว้ ภาคขับอุปกรณ์นั้นจะได้รับข้อความ DRV_REMOVE เมื่อได้ข้อความนี้ภาคขับอุปกรณ์จะทำการลบข้อมูลที่เกี่ยวข้องออกจากไฟล์ SYSTEM.INI รูปแบบของข้อความเป็นดังนี้

Remove (uninstall) a driver

msg : DRV_REMOVE

lParam10

lParam20

Returns :

Possible values are DRVCNF_OK DRVCNF_CANCEL and DRVCNF_RESTART. The control Panel seems to treat return values asDRVCNF_RESTART.

การติดตั้งภาคขับอุปกรณ์แบบติดตั้งได้

ในการติดตั้งภาคขับอุปกรณ์ของวินโดวส์นั้น มีวิธีการทำได้หลายแบบ แบบที่เร็วที่สุดคือ การเข้าไปแก้ไขข้อมูลในไฟล์ SYSTEM.INI ในรูปแบบดังนี้

entry_name = installable_driver_filename

เช่น 3D-Stereo = c:\windows\3dsystem\stereo.dll

entry_name นั้นอาจจะอยู่ในเซ็คชัน [drivers] (ซึ่งในกรณีนี้พารามิเตอร์ที่สองของ OpenDriver จะถูกเซ็ตเป็น NULL) หรือเอาไว้ในเซ็คชันอื่น (ซึ่งกรณีนี้ต้องกำหนดชื่อเซ็คชันเอาไว้ในพารามิเตอร์ ที่ 2 ของ OpenDriver ด้วย) เช่น

[3D]

3D-Stereo = c:\windows\3dsystem\stereo.dll