บทที่ 6

การออกแบบและพัฒนาโปรแกรมแสดงภาพสเตอริโอสลับเชิงเวลา

จากหลักการที่ได้กล่าวมาข้างต้นถึงวิธีการแสดงภาพสเตอริโอแบบภาพสสลับเชิงเวลา และอุปกรณ์ต่างๆ ที่เกี่ยวข้อง รวมไปถึงการทำงานของสภาพปฏิบัติการวินโดวส์ และการออกแบบภาคขับอุปกรณ์ชนิดติดตั้งได้ เราจะนำความรู้ที่ได้ทั้งหมดมาประกอบเป็นแนวทางที่ใช้ในการออกแบบและพัฒนางานวิจัยนี้ และเพื่อเตรียมการสำหรับการพัฒนาในอนาคต ผู้วิจัยได้ทำการแยกส่วนองค์ประกอบโปรแกรมออกจากกันตามหน้าที่การทำงานหลักของแต่ละส่วน เพื่อให้สามารถเลือกพัฒนาเฉพาะส่วนที่ต้องการได้โดยง่าย โดยไม่มีผลกระทบต่อส่วนอื่นที่ไม่เกี่ยวข้อง ซึ่งจะเป็นประโยชน์ต่อการพัฒนาและปรับปรุงในอนาคตต่อไป

โปรแกรมจะถูกออกแบบให้แยกกันระหว่างภาคขับสเตอริโอและส่วนที่เป็นโปรแกรมแสดงภาพ ภาคขับสเตอริโอจะถูกออกแบบให้เป็นแฟ้มแบบ DLL ซึ่งสามารถนำไปใช้กับงานวิจัยอื่นๆที่ต้องการภาคขับอุปกรณ์สำหรับแสดงผลแบบสเตอริโอ 3 มิติได้ ส่วนโปรแกรมแสดงภาพสามารถเปลี่ยนเป็นโปรแกรมประยุกต์อื่นที่เหมาะสมตามแต่ลักษณะการนำไปใช้งาน

การออกแบบภาคขับสเตอริโอ

โปรแกรมในส่วนนี้จะถูกออกแบบให้เป็นภาคขับอุปกรณ์แบบติดตั้งได้ โดยมีโครงสร้างแฟ้มเป็นแบบ DLL โดยภายในแบ่งออกเป็น 3 หน้าที่หลัก คือ

1. ส่วนควบคุมจังหวะการแสดงผล
2. ส่วนกำหนดค่าเริ่มต้นและควบคุมการ์ดแสดงผล
3. ส่วนบรรจุฟังก์ชันส่งออก (Exported Function)

รายละเอียดในการออกแบบ แต่ละส่วนเป็นดังต่อไปนี้

1. การออกแบบส่วนควบคุมจังหวะการแสดงผล

ในการแสดงภาพแบบสเตอริโอคู่สลับนั้นเพื่อให้ได้ภาพที่นุ่มนวลไม่มีอาการสั่นพริ้วหรือกระตุกเป็นห้วงๆ ควรจะแสดงภาพด้วยความเร็วสูง และกำหนดช่วงเวลาการสลับภาพด้วยความเร็วที่แน่นอนและสม่ำเสมอ สำหรับระบบที่เราออกแบบขึ้นในงานวิจัยนี้จะทำการแสดงภาพสำหรับดวงตาแต่ละข้างด้วยความเร็วประมาณ 30 ภาพ/วินาที โดยสัญญาณที่เรานำมาใช้เป็นตัวกำหนดจังหวะจะนำมาจากสัญญาณการกวาดภาพทางแนวตั้ง (Vsync) (ดูรูปที่ 5.2) ซึ่งจะเกิดเป็นพัลซ์ (Pulse) บวก เฉพาะในขณะที่จอภาพอยู่ในสภาพปลอดสัญญาณ (Blanking) และปืนอีเลคตรอน กำลังสบัดจากจุดล่างสุดทางมุมขวาล่างของจอภาพ กลับไปยังจุดเริ่มต้นที่มุมซ้ายบนของจอภาพ

การนำสัญญาณนี้มาใช้กำหนดจังหวะการแสดงผลมีข้อดี 2 อย่าง คือ

1. ทำให้ไม่เกิดสัญญาณรบกวนแบบจุดขาว (Snow) ขณะทำการแสดงผลทั้งนี้เนื่องจากการเปลี่ยนภาพจะทำในช่วงปลอดสัญญาณ

2. สามารถเพิ่มความถี่ในการแสดงภาพได้ เพียงแต่เปลี่ยนการ์ดและจอแสดงผลที่มีความถี่ในการกวาดภาพสูงขึ้นโดยไม่ต้องแก้ไขโปรแกรมแต่อย่างใดทั้งสิ้น

สัญญาณการกวาดภาพทางแนวตั้งนี้จะถูกต่อมาจากขั้วต่อสายสัญญาณของจอภาพ นำมาผ่านวงจรกรองกระแส (Rectify) ซึ่งประกอบด้วยไดโอดสัญญาณเบอร์ 1N-4148 เพื่อให้สัญญาณเฉพาะซีกบวกเท่านั้นผ่านไปเข้าที่ขาสัญญาณ RI (Ring Indicator) ของพอร์ตสื่อสารแบบอนุกรม เพื่อใช้ในการกระตุ้นกลไกการสลับภาพต่อไป

ขั้นตอนต่อมาเราจะทำการสร้างโปรแกรม ที่ทำหน้าที่รับสัญญาณ RI ไปดำเนินการต่อ โดยโปรแกรมส่วนนี้จะถูกแบ่งออกเป็น 3 ส่วน คือ

1.1 ส่วนการติดตั้งโปรแกรมบริการอินเตอร์รัพท์

1.2 ส่วนการรื้อถอนโปรแกรมบริการอินเตอร์รัพท์

1.3 ส่วนที่เป็นโปรแกรมบริการอินเตอร์รัพท์

1.1 การออกแบบส่วนการติดตั้งโปรแกรมบริการอินเตอร์รัพท์

หน้าที่หลักของโปรแกรมในส่วนนี้คือการกำหนดตำแหน่งของโปรแกรมบริการอินเตอร์รัพท์ที่ต้องการลงไปในตารางอินเตอร์รัพท์เดสคริปเตอร์ (IDT) เพื่อที่เมื่อเกิดการอินเตอร์รัพท์ขึ้นจะไปกระตุ้นให้โปรแกรม บริการที่ถูกต้องทำงาน หน้าที่อีกประการหนึ่งคือกำหนดค่ารีจีสเตอร์ควบคุมต่างๆ ให้เหมาะสม

ในระบบพีซีคอมพิวเตอร์ตั้งแต่รุ่น เอที เป็นต้นมาการจัดสัญญาณขออินเตอร์รัพท์จากพอร์ตสื่อสารแบบอนุกรมนั้น ใช้ IRQ 3 สำหรับ COM2 และ IRQ 4 สำหรับ COM1 ซึ่งจะตรงกับหมายเลขอินเตอร์รัพท์ที่ 0bh และ 0ch ตามลำดับ การติดตั้งโปรแกรมบริการจะต้องตรวจสอบหมายเลขพอร์ตสื่อสารที่ใช้งานก่อน

เมื่อทราบถึงหมายเลขพอร์ตแล้วโปรแกรมจะต้องเตรียมการกำหนดบิตควบคุมการเปิด-ปิด การอินเตอร์รัพท์ภายนอก ซึ่งถูกควบคุมโดย วงจรควบคุมอินเตอร์รัพท์ (PIC) โดยใช้รีจีสเตอร์ควบคุม ที่เรียกว่า อินเตอร์รัพท์มาสก์รีจีสเตอร์ (Interrupt Mask Register : IMR) อยู่ที่แอดเดรส 21 h , มีโครงสร้างดังนี้

บิตที่	7	6	5	4	3	2	1	0
IRQ หมายเลข	7	6	5	4	3	2	1	0

บิตใดมีค่าเป็น 0 = เปิด อินเตอร์รัพท์

บิตใดมีค่าเป็น 1 = ปิด อินเตอร์รัพท์

สำหรับการแจ้งให้วงจรควบคุมอินเตอร์รัพท์ทราบถึงการเสร็จสิ้นอินเตอร์รัพท์ในแต่ละครั้งจะทำโดยการส่งคำสั่ง EOI (End of Interrupt) ซึ่งมีค่าเท่ากับ 20h ไปที่แอดเดรส 20h ซึ่งจะต้องทำก่อนที่จะออกโปรแกรมบริการทุกครั้ง มิฉะนั้นระบบจะสูญเสียการควบคุมทันที

อีกส่วนหนึ่งคือการกำหนดรีจีสเตอร์ควบคุมของพอร์ตสื่อสาร ซึ่งจะมีอยู่ 3 ตัว คือไลน์คอนโทรลรีจีสเตอร์ (Line Control Register : LCR) โมเด็มคอนโทรลรีจีสเตอร์ (Modem Control Register : MCR) และอินเตอร์รัพท์อีนาเบิลรีจีสเตอร์ (Interrupt Enable Register : IER) ในการควบคุมรีจีสเตอร์เหล่านี้เพื่อให้พอร์ตสื่อสารร้องขออินเตอร์รัพท์ เมื่อมีสัญญาณ RI เกิดขึ้นมีขั้นตอนดังนี้

1. กำหนดให้บิต 3 ของ MCR เป็น 1 เพื่อเข้าสู่สภาวะร้องขออินเตอร์รัพท์ได้
2. กำหนดให้บิต 7 ของ LCR เป็น 0 เพื่อขอติดต่อกับ IER

3. กำหนดให้บิต 3 ของ IER เป็น 1 เพื่อให้อินเตอร์รัพท์เมื่อมีสัญญาณ RI เกิดขึ้น

หน้าที่ที่สำคัญมากอีกอันหนึ่งของโปรแกรมส่วนนี้คือการกำหนดตำแหน่งที่อยู่ของโปรแกรมบริการลงไปในตาราง IDT ซึ่ง ทำได้โดยการใช้บริการ INT 21h หมายเลข 25/35h แต่เนื่องจาก ตัวแปลภาษาส่วนมาก จะมีบริการลักษณะนี้ให้ด้วย ซึ่งเราสามารถใช้ได้เช่นกัน สำหรับการวิจัยครั้งนี้ ตัวแปลภาษาที่ใช้ คือ Borlandc++ for Window รุ่น 3.1 ซึ่งมีฟังก์ชันบริการลักษณะนี้อยู่ 2 ฟังก์ชัน คือ

```
void interrupt (*getvect(int interruptno))();
void setvect(int interruptno, void interrupt (*isr) ( ));
```

ซึ่งจะเห็นว่า สามารถใช้งานได้ค่อนข้างสะดวกกว่าบริการ INT 21h หมายเลขบริการย่อยที่ 25/35h มาก

ในการทำความเข้าใจกับการทำงานและลักษณะการออกแบบของโปรแกรมในส่วนนี้ การนำเสนอตัวอย่างของโปรแกรมจะทำให้ผู้ศึกษามีความเข้าใจถึงลำดับขั้นและกลไกที่ใช้ได้ดีขึ้น จึงได้นำเสนอไว้ ณ ที่นี้ด้วย

```
void Install(void)
{
    if (PORT== 0x3fc) {
        BASE=COM1BASE;
        comnum=0;
        intnum=0x0c;
        mask=0xef;
    } else {
        BASE=COM2BASE;
        comnum=1;
        intnum=0x0b;
        mask=0xf7;
    }
```

```
        disable();
        oldint=getvect(intnum);                    // COM2 old interrupt vector
        setvect(intnum,MyInt);
        outportb(MCR,0x08);                        // clear diagnostic bit
        outportb(LCR,inportb(LCR) & 0x7f);         // Set DLAB(bit 7)=0 select IIR
        outportb(IER,0x08);                        // Set bit 3 for Modem Status Int
        outportb(MASK8259,inportb(MASK8259) & mask);
        enable();
}
```

1.2 การออกแบบส่วนการรื้อถอนโปรแกรมบริการอินเตอร์รัพท์

ในการรื้อถอนโปรแกรมบริการออกจากระบบจะต้องอาศัยความช่วยเหลือในการเก็บค่าตำแหน่งของโปรแกรมบริการเดิมก่อนทำการติดตั้งโปรแกรมบริการของเราเอาไว้หากปราศจากขั้นตอนนี้จะไม่สามารถรื้อถอนโปรแกรมบริการได้

ลำดับแรกก่อนทำการรื้อถอน จะต้องทำการปิดกั้นอินเตอร์รัพท์ก่อนโดยใช้คำสั่ง CLI หรือ ฟังก์ชันของตัวแปรภาษาที่ชื่อ disable() ทั้งนี้เนื่องจากระหว่างการรื้อถอนโปรแกรมบริการไม่สามารถยอมให้มีการเรียกใช้โปรแกรมบริการนี้ได้ ตัวอย่างของการรื้อถอนแสดงได้ดังนี้

```
void de_Install(void)
{
        disable();          // Disable all interupt
        outportb(MASK8259,(inportb(MASK8259) | ~mask)); // Set bit 3 for disabling IRQ3
        setvect(intnum,oldint);
        enable();
}
```

1.3 การออกแบบส่วนที่เป็นโปรแกรมบริการ

หน้าที่หลักของโปรแกรมบริการนี้คือทำหน้าที่ควบคุมการ์ดแสดงผลให้ทำการแสดงภาพสำหรับตาซ้ายและขวาสลับไปมา ในแต่ละครั้งที่เกิดการอินเตอร์รัพท์ขึ้น ขณะเดียวกันก็ทำการปิด-เปิด เลนส์ของแว่นผลึกเหลวให้สัมพันธ์กับการแสดงผล เมื่อดำเนินการทั้ง 2 เสร็จ ก็จะต้องส่งคำสั่ง EOI ไปให้วงจรควบคุมอินเตอร์รัพท์ทราบแล้วจึงจบการทำงาน

ตัวแปลภาษาที่เลือกใช้ในงานวิจัยนี้ มีขีดความสามารถในการกำหนดชนิดของฟังก์ชันให้เป็นแบบที่เหมาะสมในการใช้เป็นฟังก์ชันบริการอินเตอร์รัพท์เป็นอย่างดี โดยสามารถใช้คีย์เวิร์ด interrupt นำหน้าชื่อฟังก์ชัน เมื่อตัวแปรภาษาพบว่าฟังก์ชันใดเป็นชนิด interrupt จะทำการสร้างชุดคำสั่งเสริมพิเศษไว้ที่ส่วนหัว (Prolog) และส่วนท้าย (Epilog) ซึ่งมีโครงสร้างเป็นดังนี้

```
void interrupt Myint (...)
{
        push    ax
        push    bx
        push    cx
        push    dx
        push    es          ———— กลุ่มคำสั่งเสริมส่วนหัว (Prolog)
        push    ds
        push    si
        push    di
        push    bp
        mov     bp,xxxx
        mov     ds,bp
        mov     bp,sp
        pop     bp

    Body of service routine

        pop     di
        pop     si
        pop     ds
        pop     es          ———— กลุ่มคำสั่งเสริมส่วนท้าย (Epilog)
        pop     dx
        pop     ex
        pop     bx
        pop     dx
        reti
```

จุดที่น่าสนใจของเรื่องนี้ คือ หลังจากการเก็บค่ารีจีสเตอร์ต่างๆ ลงสู่สแตกแล้ว ตัวแปลภาษาจะทำการสร้าง ชุดคำสั่ง สำหรับการเข้าถึงดาต้าเซกเมนต์ให้กับโปรแกรมบริการโดยอัตโนมัติ โดยใช้คำสั่ง

```
mov     bp,xxxx
mov     ds,bp
```

xxxx เป็นค่า ซีเลคเตอร์ สำหรับดาต้าเซกเมนต์ของโปรแกรมบริการ ที่จะถูกหาค่าสุดท้ายขณะที่โปรแกรมถูกโหลดลงในหน่วยความจำ ทำให้เราสามารถเข้าถึงดาต้าเซกเมนต์ได้ทันทีที่โปรแกรมบริการถูกเรียกใช้

คุณสมบัตินี้เป็นสิ่งที่สำคัญมากในการสร้างโปรแกรมบริการอินเตอร์รัพท์ที่จะทำงานในระบบวินโดวส์ ทั้งนี้เนื่องจากขณะที่หน่วยประมวลผลกลาง ถ่ายทอดการควบคุมระบบมาให้แก่โปรแกรมบริการมีเพียงรีจีสเตอร์ CS:IP เท่านั้นที่มีค่าแน่นอน สำหรับรีจีสเตอร์อื่นๆ ไม่สามารถระบุค่าได้เลยทำให้ผู้ออกแบบโปรแกรมบริการต้องหาวิธีเข้าถึงดาต้าเซกเมนต์ของโปรมแกรมบริการเอง การที่ตัวแปลภาษาเข้ามามีบทบาทในการแก้ปัญหานี้จึงเป็นข้อดีอย่างมาก

อย่างไรก็ตามเนื่องจากการกำหนดค่าซีเลคเตอร์ลงไปในโค๊ดเซกเมนต์โดยตรง ทำให้มีปัญหาว่าไม่สามารถมีโปรแกรมบริการ หลายๆ อินส์แตนท์ได้เนื่องจากทุกๆ อินสแตนท์ของโปรแกรมจะมีดาต้าเซกเมนต์เดียวกัน ซึ่งไม่ถูกต้องในหลักการของการมีหลายอินสแตนท์ ที่ดาต้าเซกเมนต์ของแต่ละอินสแตนท์จะแยกจากกัน แต่ใช้โค๊ดเซกเมนต์เดียวกัน

จากข้อจำกัดดังกล่าวการพัฒนาโปรแกรมบริการอินเตอร์รัพท์จึงควรให้อยู่ในรูป DLL จะเหมาะสมที่สุด

2. การออกแบบโปรแกรมส่วนกำหนดค่าเริ่มต้นและควบคุมกับการ์ดแสดงผล

ในการออกแบบ ได้จัดให้หน้าที่ส่วนนี้อยู่ในส่วนของโปรแกรมที่จะถูกเรียกใช้เฉพาะตอนเริ่มทำงานครั้งแรกสุดเท่านั้น โดยอาศัยหลักการที่ว่า วินโดวส์จะโหลดภาคขับอุปกรณ์ลงสู่หน่วยความจำแล้วส่งข้อความ DRV_LOAD ให้เมื่อมีการเรียกใช้ API ชื่อ OpenDriver() เป็นครั้งแรก, การเรียกOpenDriver() ครั้งต่อๆไปของภาคขับอุปกรณ์ตัวเดิม วินโดวส์จะส่งเฉพาะข้อความ DRV_OPEN เท่านั้น

ปัญหาของการออกแบบโปรแกรมในส่วนนี้ อยู่ที่ความหลากหลายของการ์ดแสดงผลที่มีใช้ในระบบพีซี คอมพิวเตอร์ โดยที่นับจนถึงปัจจุบันยังไม่มีมาตรฐานการผลิตการ์ดแสดงผลชนิดซุปเปอร์วีจีเอ ในระดับฮาร์ดแวร์เลย (มาตรฐาน VESA เป็นเพียงมาตรฐานการเชื่อมต่อในระดับซอฟต์แวร์) ดังนั้นเมื่อต้องการพัฒนาโปรแกรมที่สามารถทำงานได้กับการ์ดแสดงผลหลายๆ ชนิด หรือมีการใช้คุณสมบัติพิเศษของการ์ดแสดงผลที่มาตรฐาน VESA ไม่สนับสนุน ผู้พัฒนาจะต้องทำการสร้างโปรแกรมย่อยที่ใช้ควบคุมการ์ดแสดงผลต่างๆ เก็บรวบรวมไว้เพื่อให้โปรแกรมหลักเรียกใช้ให้เหมาะสมกับการ์ดแสดงผลแต่ละชนิดเอง ซึ่งเป็นปัญหากับการออกแบบมากเนื่องจากข้อจำกัดทางด้านระยะเวลา และทรัพยากรที่มีในการวิจัย ถ้าต้องทำการศึกษาการควบคุมการ์ดแสดงผลทุกชนิดในท้องตลาดคงไม่สามารถทำได้ การออกแบบโปรแกรมส่วนนี้จึงต้องอาศัยโปรแกรมแชร์แวร์ (Shareware) ชื่อ UNIVBE ซึ่งมีจุดประสงค์เป็นภาคขับการ์ดแสดงผลชนิดซุปเปอร์วีจีเอแบบเอนกประสงค์

โปรแกรมนี้มีคุณสมบัติ, จุดประสงค์ในการใช้งาน และประวัติ โดยย่อ ดังนี้

UNIVBE ย่อมาจากคำว่า The Universal VESA VBE เป็นโปรแกรมฝังตัวขนาดเล็ก มีหน้าที่เพิ่มขยายการทำงานของวีดีโอไบออสในการ์ดแสดงผลชนิดซุปเปอร์วีจีเอ เพื่อให้บริการวีดีโอไบออสอินเตอร์เฟส ที่ใช้มาตรฐาน VESA รุ่น 1.2 ขึ้นไป ออกแบบและพัฒนาโดย Kendall Bennett ในปีค.ศ. 1992 ที่มหาวิทยาลัย Royal Melbourne Insititue of Technology (RMIT) ประเทศออสเตรเลีย ในขณะทำการวิจัยโปรแกรมนี้พัฒนามาถึงรุ่น 4.2 แล้ว

การที่การ์ดแสดงผลใดๆสามารถให้บริการวีดีโออินเตอร์เฟส ตามมาตรฐาน VESAได้ (ไม่ว่าจะเป็นการใช้ไบออสของการ์ด หรือการใช้โปรแกรมฝังตัว UNIVBE ก็ตาม) จะทำให้การ์ดแสดงผลนั้นสามารถทำงานร่วมกับโปรแกรมที่ใช้มาตรฐานนี้ได้ โดยไม่เกิดปัญหาความไม่เข้ากัน

ผู้สร้างโปรแกรมกล่าวอ้างว่าเหตุผลสำคัญที่โปรแกรมนี้จะเข้ามาแทนที่วีดีโอไบออสเดิมที่มีอยู่ คือ

1 วีดีโอไบออสเดิมไม่สนับสนุน หรือสนับสนุนเพียงบางส่วน ต่อมาตรฐาน VESA

2 UNIVBE เป็นโปรแกรมฝังตัวที่มีขนาดเล็กมากจะใช้พื้นที่เพียง 3 KB ในการทำงานกอปร์กับเป็นโปรแกรมที่มีการเพิ่มเติมแก้ไขได้ตลอดเวลา จึงทำให้มีความสามารถ และความถูกต้องเพิ่มขึ้นเรื่อยๆ จึงมีแต่จะดีกว่า และทันสมัยกว่าไบออสเดิมที่อยู่บนการ์ด

3 มีความสามารถพิเศษอื่นๆ ที่วีดีโอไบออสส่วนใหญ่ไม่มีคือ

3.1 สามารถทำเพจฟลิปปิง (Page Flipping) คือ สามารถทำการแสดงผลแบบหลายหน้าได้ในทุกๆ ภาวะการแสดงผล (ทั้งนี้ขึ้นอยู่กับขนาดของหน่วยความจำที่การ์ดแสดงผลมีด้วย) ทำให้โปรแกรมประยุกต์ที่มีการสร้างภาพเคลื่อนไหวในภาวะความละเอียดสูงได้ประโยชน์อย่างมาก

3.2 สามารถทำจอภาพเสมือน (Virtual Screen) คือ สามารถนำหน่วยความจำที่มีอยู่ในการ์ดแสดงผลมาทำการจัดเรียงใหม่ เพื่อให้สามารถบรรจุภาพขนาดใหญ่กว่าภาวะความละเอียดจริงไว้ในหน่วยความจำ แล้วเลื่อนภาพ (Scroll) ไปมาอย่างนิ่มนวลได้

3.3 สามารถแสดงสีในแบบ 32 K, และ 16 ล้านสีได้ นอกไปจากภาวะ 16 และ 256 สีธรรมดา

3.4 มีสมรรถนะในการสลับชุดความหน่วยจำได้เร็วกว่าไบออสทั่วๆไป (Fast Bank Switching)

3.5 มีการจัดเตรียมบริการส่วนขยาย เพื่อให้โปรแกรมที่ทำงานในภาวะป้องกันเรียกใช้ได้โดยตรง (โปรแกรมจะทำงานได้เร็วขึ้นกว่าการเรียกใช้บริการผ่านทาง INT 10h มาก เนื่องจากการตัดขั้นตอนที่เสียเวลาต่างๆ ทิ้งไป)

4 นับจนถึงรุ่นปัจจุบัน (รุ่น 4.2) โปรแกรมนี้สามารถทำงานได้กับการ์ดแสดงผลดังต่อไปนี้

- ATI Technologies 18800, 28800
- Ahead A & B
- Chips & Technologies 82c451/452/453
- Everex
- Genoa Systems GVGA
- OAK Technologies OTI-037C, OTI-067, OTI-077, OTI-087
- Paradise PVGA1A, WD90C00/10/11/20/21/30/31
- NCR 77C20/21/22E
- Trident 88/8900
- Video7 V7VGA versions 1-5
- Tseng Labs ET3000, ET4000, ET4000/W32
- AcuMos AVGA2
- S3 86c911/924/801/805/928

- Advance Logic AL2101 SuperVGA
- MXIC 86010 SuperVGA
- Primus 2000 SuperVGA
- RealTek 3106 SuperVGA
- Cirrus Logic CL-GD 5420, 5422, 5424, 5426, 5428

หมายเหตุ: ไม่ทั้งหมดของการ์ดแสดงผลเหล่านี้ที่ได้รับการทดสอบอย่างสมบูรณ์ และโปรแกรม UNIVBE ในรุ่นปัจจุบันไม่สามารถใช้ความสามารถพิเศษที่มีในการ์ดบางรุ่นเช่น การมีวงจรเร่งความเร็ว (Acceleration) ได้

จากคุณสมบัติดังกล่าวการออกแบบการทำงานของโปรแกรมในส่วนนี้ จะดำเนินการโดยใช้บริการของ UNIVBE ซึ่งนอกจากจะมีผลดีในแง่ลดเวลาและข้อผิดพลาดในการพัฒนาลงไปได้มากแล้ว ยังมีผลดีในแง่การปรับปรุงภาคขับการ์ดแสดงผลในอนาคต จะทำได้โดยง่ายเพียงปรับปรุงเฉพาะ UNIVBE เท่านั้น (ผู้สนใจสามารถดาวน์โหลด UNIVBE รุ่นล่าสุดได้โดยใช้ FTP ไปที่ godzilla.cgl.rmit.oz.au (131.170.14.2): kjb/MGL/svgakt??.zip)

การเชื่อมต่อกับ UNIVBE

UNIVBE เป็นโปรแกรมฝังตัวที่ทำงานภายใต้ระบบปฏิบัติการดอส ในการใช้งาน UNIVBE จากโปรแกรมที่ทำงานภายใต้ระบบวินโดวส์จะต้องแก้ปัญหาที่สำคัญ 2 ข้อคือ

1. ทำอย่างไรจึงจะสามารถทำให้โปรแกรมประยุกต์ของวินโดวส์ซึ่งทำงานในภาวะป้องกัน สามารถติดต่อขอใช้บริการจาก UNIVBE ซึ่งฝังตัวอยู่ในดอสและทำงานในภาวะจริงได้.
2. บริการบางอย่างของ UNIVBE จะให้ผลลัพธ์กลับมาในหน่วยความจำ ซึ่งผู้ขอใช้บริการต้องส่งค่าตำแหน่งของหน่วยความจำไปให้ แต่โปรแกรมที่ทำงานในภาวะป้องกันอ้างถึงหน่วยความจำโดยใช้รูปแบบ ซีเลคเตอร์:ออฟเซต ส่วนโปรแกรมที่ทำงานในภาวะจริงใช้รูปแบบ เซกเมนต์:ออฟเซต ยิ่งไปกว่านั้นตำแหน่งสูงสุดของหน่วยความจำที่โปรแกรมในภาวะจริงสามารถอ้างถึงจะอยู่ภายใน 1 MB. แรกเท่านั้น แต่โปรแกรมในภาวะป้องกันสามารถใช้หน่วยความจำที่อยู่สูงกว่านี้ได้ ทำอย่างไรจึงจะกำหนดพื้นที่สำหรับใช้ในการสื่อสารระหว่างโปรแกรมทั้ง 2 ได้.

จากการศึกษาไม่พบว่ามี API ใดๆ ของวินโดวส์ที่สามารถใช้ในการแก้ปัญหาข้อแรกได้ แต่พบว่าบริการในลักษณะนี้มีอยู่ในส่วนขยายของระบบปฏิบัติการที่ผนวกอยู่ในตัววินโดวส์เอง ส่วนขยายที่ว่านี้มีชื่อเรียกว่า DPMI (Dos Protected Mode Interface) สำหรับงานวิจัยนี้บริการที่ต้องการใช้ในการแก้ปัญหานี้คือ บริการ INT 31h หมายเลขที่ 300h ซึ่งมีชื่อเรียกว่า บริการจำลองอินเตอร์รัพท์ภาวะจริง (Simulate Real Mode Interrupt) ซึ่งมีรูปแบบการใช้งานดังนี้

INT 31h Function 0300h (Simulate Real Mode Interrupt)

Simulate an interrupt in real mode. The function transfers control to the address specified by the real mode interrupt vector. The real mode handler must return by executing an IRET.

Call with:

AX = 0300H

BL = interrupt number

BH = must be zero

CX = number of word to copy from protected mode to real mode stack.

ES:(E)DI = Selector:Offset of real mode register data structure in the following format

Offset	Length	Contents
00h	4	DI or EDI
04h	4	SI 0r ESI
08h	4	BP or EBP
0Ch	4	Reserved should be 0
10h	4	BX or EBX
14h	4	DX or EDX
18h	4	CX or ECX
1Ch	4	AX or EAX
20h	2	CPU status flags
22h	2	ES
24h	2	DS
26h	2	FS
28h	2	GS

2Ah	2	IP (Reserved ignored)
2Ch	2	CS (Reserved ignored)
2Eh	2	SP
30h	2	SS

Retruns:

If function successful

CF = clear

ES:(E)DI = Selector:Offset of modified real mode regidter data structure.

If function unsuccessful

CF = set

AX = error code

8012h = Linear memory unavailable (stack)

8013h = Physical memory unavailable (stack)

8014h = Backing store unavailable (stack)

8021h = Invalid value (CX too large)

ส่วนปัญหาประการที่ 2 แก้ไขโดยการกำหนดพื้นที่ของหน่วยความจำที่อยู่ภายใน 1 MB. แรก ที่มีขนาดใหญ่เพียงพอและไม่มีโปรแกรมอื่นใช้งาน ให้เป็นพื้นที่ร่วมสำหรับการสื่อสาร (ในงานวิจัยนี้ใช้พื้นที่ส่วนท้ายของหน่วยความจำสแดงผลบริเวณ A000:FF00h-A000:FFFFh) โดยโปรแกรมในภาวะจริงสามารถเข้าถึงได้โดยตรง ด้วยตัวชี้ตำแหน่งในรูปแบบ เซกเมนต์:ออฟเซต แต่สำหรับโปแกรมในภาวะป้องกันจะต้องใช้ตัวชี้ตำแหน่งในรูป ซีเลคเตอร์:ออฟเซต ซึ่งจะต้องอาศัยบริการ DPMI หมายเลขที่ 0002h ซึ่งมีชื่อเรียกว่า บริการสร้างเดสคริปเตอร์จากค่าเซกเมนต์ (Segment to Descriptor) มีรูปแบบการใช้ดังนี้

INT 31h Function 0002h (Segment to Descriptor)

Map a real mode segment (paragraph) address onto LDT descriptor that can be used by a protected mode program to access the same memory.

Call with:

AX = 0002h

BX = real mode segment.

Returns:

if function successful

CF = clear

AX = Selector for real mode segment

if function unsucessful

CF = set

AX = error code (8011h descriptor unavailable)

ตัวอย่างการโปรแกรมการใช้บริการสร้างเดสคริปเตอร์เป็นดังนี้

```
unsigned        VGA_ADDR
        asm     mov     ax,2
        asm     mov     bx,0xa000           //Segment value
        asm     int     31h
        asm     jc      error
        asm     mov     VGA_ADDR,AX         //Store Selector value
```

สำหรับตัวอย่างการใช้บริการ DPMI เพื่อติดต่อกับ UNIVBE เป็นดังนี้

```
.data
regstruc        STRUC
edi1            dd      0
esi1            dd      0
ebp1            dd      0
res1            dd      0
ebx1            dd      0
edx1            dd      0
ecx1            dd      0
eax1            dd      0
flags           dw      0
es1             dw      0
ds1             dw      0
fs1             dw      0
```

```
gs1             dw      0
ip1             dw      0
cs1             dw      0
sp1             dw      0
ss1             dw      0
regstruc        ENDS
reg1            regstruc <0>
param1          dw      ?
COMM_AREA_SEG   EQU     0A000h
AREA_LEN        EQU     256
COMM_AREA_OFF   EQU     0FF00h
.code
;Get super VGA information, by using DPMI service to communicate with UNIVBE. Using an
unused area of video RAM as communication buffer. (at A000h:FF00h)
; int getVGAinfo(VgaInfoBlock *vgaInfo)
_getVGAinfo     proc    near
                push    bp
                mov     bp,sp
                mov     ax,[bp+4]
                mov     param1,ax
                mov     word ptr reg1.eax1,4f00h                ;Get super VGA info
                mov     word ptr reg1.edi1,COMM_AREA_OFF ;Offset of DOS memory
                mov     word ptr reg1.es1,COMM_AREA_SEG  ;Segment of DOS memory
                lea     di,reg1
                mov     bx,0010h        ;Int 10h
                mov     ax,0300h        ;DPMI service # 0300h
                mov     cx,0            ;No word to copy
                int     31h             ;call DPMI
                jc      error0          ;Service failed !
                cmp     word ptr reg1.eax1,004fh        ;Check response of UNIVBE
```

```
                jne     error0
                mov     si,COMM_AREA_OFF        ;Source offset
                mov     di,param1               ;Target offset
                mov     cx,AREA_LEN             ;Lenght of this buffer
                mov     ax,_VGA_ADDR
                mov     ds,ax                   ;Store selector to DS
                push    ss
                pop     es                      ;ES = program data segment
                cld
                rep     movsb
                pop     bp
                xor     ax,ax                   ;Indicate OK.
                ret
error0:         pop     bp
                mov     ax,0ffffh               ;Indicate error
                ret
_getVGAinfo     endp
```

ในส่วนของ UNIVBE บริการพิเศษสำหรับใช้ในการเชื่อมกับโปรแกรมที่ต้องการความสามารถในส่วนเพิ่มขยายของ UNIVBE นอกเหนือไปจากที่มาตรฐาน VESA กำหนดไว้ มีรูปแบบการขอใช้บริการดังนี้

ให้ใช้ INT 10h และกำหนดค่ารีจีสเตอร์ดังนี้

AX = 4F01h

CX = FFFFh

ES:DI = เซกเมนต์:ออฟเซต ของโครงสร้างข้อมูล PmInfoBlock เพื่อเก็บผลลัพธ์ (ลักษณะโครงสร้างแสดงไว้ ในส่วนการออกแบบโครงสร้างข้อมูล)

(ให้สังเกตว่า UNIVBE กำหนดหมายเลขบริการพิเศษนี้รวมทั้งรูปแบบการใช้ ให้ตรงกับบริการสอบถามข้อมูลภาวะการแสดงผล (Get Mode Information) ซึ่งใช้ในมาตรฐาน VESA

แตกต่างกันเพียงหมายเลขภาวะการแสดงผลใช้เป็น FFFFh จึงทำให้สามารถใช้โปรแกรมย่อยในการติดต่อ UNIVBE ร่วมกันได้)

ตัวอย่างโปรแกรมเพื่อขอใช้บริการพิเศษของ UNIVBE เป็นดังนี้

```
;int getModeinfo(ModeInfoBlock *modeInfo,int mode);
_getModeinfo    proc    near
                push    bp
                mov     bp,sp
                mov     ax,[bp+4]
                mov     param1,ax
                mov     word ptr reg1.es1,COMM_AREA_SEG      ;Segment of DOS
memory
                mov     word ptr reg1.eax1,4f01h             ;Get mode
information
                mov     word ptr reg1.edi1,COMM_AREA_OFF     ;Offset of DOS
memory
                mov     ax,[bp+8]
                push    ax                  ;Save mode no.
                mov     word ptr reg1.ecx1,ax  ;Video mode
                lea     di,reg1
                mov     bx,0010h            ;Int 10h
                mov     ax,0300h            ;DPMI service #300h
                mov     cx,0                ;No word to copy
                int     31h
                pop     ax                  ;Restore mode no.
                jc      error1              ;Service fail !
                cmp     word ptr reg1.eax1,004fh     ;Check VESA response
                jne     error1
                cmp     ax, word ptr reg1.ecx1       ; Check valid mode
                jne     error1
                mov     si,COMM_AREA_OFF             ; Source offset
```

```
                mov     di,param1               ; Target offset
                mov     cx,AREA_LEN             ; Lenght of this buffer
                mov     ax,_VGA_ADDR
                mov     ds,ax                   ; Store selector to DS
                push    ss
                pop     es
                cld
                rep     movsb
                pop     ax
                pop     bp
                xor     ax,ax                   ; Indicate OK.
                ret
error1:         pop     bp
                mov     ax,0ffffh               ; Indicate error
                ret
_getModeinfo    endp
```

การใช้บริการนี้จะทำให้ UNIVBE ระบุตำแหน่งและขนาดของชุดคำสั่งที่อยู่ภายในตัว UNIVBE ที่ใช้ทำหน้าที่ควบคุมการเขียน-อ่าน หน่วยความจำ (Bank Switching) และควบคุมการสลับหน่วยความจำในการแสดงผล (Page Flipping) ของการ์ดแสดงผลที่ใช้ในระบบให้แก่โปรแกรมที่ขอใช้บริการ (ระบุลงในโครงสร้างข้อมูลที่ชี้ด้วย ES:DI) โปรแกรมที่ขอใช้บริการนี้มาจะต้องทำการคัดลอกชุดคำสั่งดังกล่าวออกมาไว้ในหน่วยความจำของตนเองเพื่อให้เรียกใช้ได้โดยตรง การออกแบบในลักษณะนี้จะทำให้โปรแกรมทำงานเร็วขึ้น เนื่องจากไม่ติดต่อกับ UNIVBE อีกต่อไป

แต่เนื่องจากตำแหน่งของชุดคำสั่งที่ UNIVBE ให้มาจะอยู่ในรูป เซกเมนต์:ออฟเซต อีกทั้งในภาวะป้องกันมีการกำหนดลักษณะการใช้หน่วยความจำให้แยกกันระหว่างดาต้าเซกเมนต์ กับโค๊ดเซกเมนต์ (เราไม่สามารถเขียนข้อมูลลงในโค๊ดเซกเมนต์โดยตรง หรือทำการเรียกใช้ชุดคำสั่งที่อยู่ในดาต้าเซกเมนต์ได้, ถ้ามีการกระทำดังกล่าวจะทำให้เกิด เอคเซบชัน หมายเลข 13 ทันที) ทำให้ต้องจัดกระบวนวิธีเพื่อนำชุดคำสั่งที่ได้มาไว้ในหน่วยความจำส่วนที่เป็นโค๊ดเซกเมนต์ ซึ่งในส่วนนี้การศึกษาจากตัวอย่างโปรแกรมจะทำให้เกิดความเข้าใจได้ดีกว่า

```
// These pointers are declared for use as CODE pointer. Their values are obtain from
AllocDStoCSAlias() API call.

void far *bankSwitch;          // Pointer to bank switch routine
void far *writeBank;           // Relocated write bank routine
void far *readBank;            // Relocated read bank routine
void far *pageFlip;            // Relocated page flip routine

// These pointer are declared for use as DATA pointer. Their value are obtain from
_farmalloc() function call.

void far *DS_bankSwitch;       // DS pointer to bank switch routine
void far *DS_writeBank;        // DS pointer for write bank routine
void far *DS_readBank;         // DS pointer for read bank routine
void far *DS_pageFlip;         // DS pointer for page flip routine

// This structure is filled by UNIVBE for a relocated small code that/ use for fast bank
switching and fast copying. But UNIVBE is designed to communicate with a large model
program which all pointer is FAR type.

typedef struct {
        short    writeBankLen;
        void far *writeBank;
        short    readBankLen;
        void far *readBank;
        short    newPageLen;
        void far *newPage;
} PMInfoBlock;

int Get_UNIVBE_Code(void)
```

```
{
PMInfoBlock    pmInfo;                    // Protected mode info
writeBank = readBank = pageFlip = NULL;       //Initailize code pointer
if (getModeinfo((ModeInfoBlock *)&pmInfo, -1) != 0)
        return GET_ROUTINE_FAIL;
// Convert Segment address return by UNIVBE into Selector offset form
pmInfo.readBank = SegmToSelc(pmInfo.readBank);
pmInfo.writeBank = SegmToSelc(pmInfo.writeBank);
pmInfo.newPage = SegmToSelc(pmInfo.newPage);
// Copy routine from real mode memory into protected mode and convert data segment to
executable segment
if (pmInfo.writeBankLen > 0) {
        DS_writeBank = farmalloc(pmInfo.writeBankLen);
        _fmemcpy(DS_writeBank,pmInfo.writeBank,pmInfo.writeBankLen);
        writeBank = MK_FP(AllocDStoCSAlias(FP_SEG(DS_writeBank)),FP_OFF
(DS_writeBank));
} else
        return GET_ROUTINE_FAIL;
if (pmInfo.readBankLen > 0) {
        DS_readBank = farmalloc(pmInfo.readBankLen);
        _fmemcpy(DS_readBank,pmInfo.readBank,pmInfo.readBankLen);
        readBank = MK_FP(AllocDStoCSAlias(FP_SEG(DS_readBank)),FP_OFF
(DS_readBank));
}
if (pmInfo.newPageLen > 0) {
        DS_pageFlip = farmalloc(pmInfo.newPageLen);
        _fmemcpy(DS_pageFlip,pmInfo.newPage,pmInfo.newPageLen);
        pageFlip = MK_FP(AllocDStoCSAlias(FP_SEG(DS_pageFlip)),FP_OFF
(DS_pageFlip));
} else   return NO_PAGE_FLIPPING;
```

```
// All OK !
return 0;
}
void far *SegmToSelc (void far *Segm_offset)
{
unsigned Selector,Segment,Offset;
        Segment = FP_SEG(Segm_offset);
        Offset = FP_OFF(Segm_offset);
        asm     mov     ax,2            // DPMI service no. 2
        asm     mov     bx,Segment      // Get Segment address
        asm     int     31h             // Call DPMI
        asm     mov     Selector,ax
        return  MK_FP(Selector,Offset);
}
```

3. การออกแบบส่วนบรรจุฟังก์ชันส่งออก

นอกเหนือไปจากการทำหน้าที่ควบคุมการ์ดแสดงผลและกำหนดจังหวะการแสดงภาพแล้ว ภายในภาคขับอุปกรณ์ยังได้ออกแบบให้มีฟังก์ชันส่งออกที่จำเป็นสำหรับระบบแสดงผลสเตอริโอไว้สำหรับให้โปรแกรมประยุกต์ภายนอกได้เรียกใช้ ในงานวิจัยนี้กำหนดให้มีฟังก์ชันส่งออกไว้ 4 ฟังก์ชัน โดยแบ่งเป็น 2 กลุ่มคือ

3.1 ฟังก์ชันควบคุมการ์ดแสดงผล

การทำงานของฟังก์ชันในกลุ่มนี้จะกระทำโดยตรงกับการ์ดแสดงผล โดยมีขนาดของการทำงานครั้งละ 1 หน้าแสดงผล (1 Display Page) เสมอ (1 หน้าแสดงผลมีขนาด = ความละเอียดทางแนวตั้ง x จำนวนไบต์ใน 1 เส้นแสกน) ชื่อฟังก์ชันและรูปแบบการใช้งานเป็นดังนี้

3.1.1 BOOL CopyScreen(int SrcPage, int DestPage)

ทำหน้าที่คัดลอกข้อมูลจากหน่วยความจำแสดงผลที่กำหนดด้วยพารามิเตอร์ SrcPage ไปไว้ที่หน่วยความจำแสดงผล ที่กำหนดด้วยพารามิเตอร์ DestPage ฟังก์ชันนี้จะถูกเรียกใช้เพื่อการ

สำเนาข้อมูลในจอแสดงผลขึ้นอีกชุดหนึ่ง เพื่อใช้เป็นพื้นหลังให้กับภาพที่จะถูกเตรียมขึ้น สำหรับดวงตาอีกข้างหนึ่ง

3.1.2 void VisualPage(int Page)

ทำหน้าที่ควบคุมให้การ์ดแสดงผลนำข้อมูลภาพในหน่วยความจำแสดงผลที่กำหนดด้วยพารามิเตอร์ Page ออกแสดงที่จอแสดงผล

3.1.3 void ActivePage(int Page)

ทำหน้าที่ควบคุมหน่วยความจำแสดงผล ที่กำหนดด้วยพารามิเตอร์ Page ให้เป็นส่วนแอคตีฟ (การเขียนหรืออ่าน ข้อมูลในหน่วยความจำของการ์ดแสดงผลจะกระทำกับส่วนที่แอคตีฟเท่านั้น) การใช้ฟังก์ชันนี้ทำให้เราสามารถการบรรจุข้อมูลภาพขนาด 1 หน้าแสดงผล ตั้งแต่ 2 ชุดขึ้นไป ลงในหน่วยความจำของการ์ดแสดงผลได้โดยไม่รบกวนซึ่งกันและกัน

3.2 ฟังก์ชันควบคุมแว่นผลึกเหลว

3.2.1 void Openeye(int eye)

ทำหน้าที่ควบคุมการเปิด-ปิดเลนส์ของแว่นผลึกเหลว โดยที่บิต 0 ของพารามิเตอร์ eye ควบคุมเลนส์ด้านขวา บิต 1 ควบคุมเลนส์ด้านซ้าย (สถานะของบิตเป็น 1 จะเป็นการสั่งให้เลนส์เปิด สถานะ 0 จะเป็นการสั่งให้เลนส์ปิด) การใช้ฟังก์ชันนี้ให้สัมพันธ์กันกับการใช้ ฟังก์ชัน VisualPage จะทำให้สามารถประสานการแสดงผลให้สอดคล้องกับการมองเห็นภาพของดวงตาแต่ละข้างได้.

การใช้ฟังก์ชันส่งออกทั้ง 4 เพื่อให้ระบบสามารถแสดงผลในแบบสเตอริโอมีขั้นตอนดังนี้

1 สร้างภาพสำหรับดวงตาแต่ละข้างไว้ในหน่วยความจำแสดงผล

1.1 ทำการสร้างภาพสำหรับตาข้างแรก (อาจเป็นข้างซ้ายหรือขวาก็ได้) แล้วนำออกแสดง ระบบแสดงผลสเตอริโอจะถือว่าข้อมูลที่แสดงผลอยู่ในขณะนั้นทั้งหมดเป็นภาพสำหรับดวงตาข้างแรกเรียกว่า "หน้าแสดงผลหมายเลข 0" (Page 0)

1.2 เรียกฟังก์ชัน CopyScreen() โดยกำหนดให้พารามิเตอร์ตัวแรกมีค่าเป็น 0 (Page 0) และพารามิเตอร์ตัวที่สองมีค่าเป็น 1 (Page 1) เพื่อทำการสำเนาภาพที่กำลังแสดงบนจอแสดงผล ไปเก็บไว้ที่ "หน้าแสดงผลหมายเลข 1" (Page 1)

1.3 เรียกฟังก์ชัน ActivePage() โดยกำหนดค่าพารามิเตอร์เป็น 1 (Page 1)

1.4 ทำการลบภาพในหน้าต่างที่กำลังแอคตีฟ (Active Window) ทิ้ง ทั้งเนื่องจากเป็นภาพของตาข้างแรก

1.5 ทำการสร้างภาพของดวงตาอีกข้างหนึ่งลงในช่องหน้าต่าง

1.6 เรียกฟังก์ชัน ActivePage() อีกครั้งแต่กำหนดค่าพารามิเตอร์เป็น 0 เพื่อกลับมาใช้หน้าแสดงผลหมายเลข 0

กระบวนการทั้งหมดสามารถเขียนเป็นคำสั่งในภาษา ซี ได้ดังนี้

```
DrawImage(page0);              //Create image for page 0
CopyScreen(Page0,Page1);       //Copy background to page 1
ActivePage(Page1);             //Active display memory for page 1
EraseImage(Page0);             //Erase page 0 image in page 1 memory
DrawImage(Page1);              //Create image for page 1
ActivePage(Page0);             //Active page 0 as normal
```

2 นำภาพออกแสดงให้เกิดเป็นภาพสเตอริโอ

ในการนำภาพออกแสดงจะต้องทำการกำหนดจังหวะให้เหมาะสม การใช้กลไกการอินเตอร์รัพท์ในการกำหนดจังหวะแล้วทำการแสดงภาพสลับไปมาให้สัมพันธ์กับการเปิด-ปิด เลนส์ของแว่นผลึกเหลว สามารถแสดงเป็นตัวอย่างได้ดังนี้

```
interrupt StereoShow(...)
{
    static int show;
    show ^= 1;                 //Toggle flag
    if (show == 0) {
        Openeye(LEFT);         //Open left eye and show left image.
        VisualPage(Page0);
    } else {
        Openeye(RIGHT);        //Open right eye and show right image.
        VisualPage(Page1);
    }
}
```

การออกแบบส่วนการแสดงภาพ

โปรแกรมที่ได้จากการพัฒนานี้จะมีคุณสมบัติในการนำภาพ แบบโครงเส้น (Wire Frame) จากแฟ้มข้อมูลประเภท .3D หรือ .3DV มาสร้างเป็นภาพสเตอริโอ สำหรับควงตาแต่ละข้างและแสดงภาพด้วยอัตราเร็วเท่ากับครึ่งหนึ่งของความถี่การกวาดภาพทางแนวตั้ง สามารถทำการหมุนภาพได้ทุกด้าน โดยการสั่งการผ่านทางเมาส์หรือแป้นพิมพ์ สามารถมีจำนวน ของโปรแกรมทำงานอยู่ได้พร้อมกันโดยไม่จำกัดจำนวนโดยในภาวะแสดงภาพ 3 มิติ ทุกๆช่องหน้าต่างจะมีสภาพเป็น 3 มิติได้พร้อมๆ กัน

ในส่วนนี้ แบ่งการออกแบบเป็น ดังนี้

1 การออกแบบจอภาพและส่วนติดต่อกับผู้ใช้

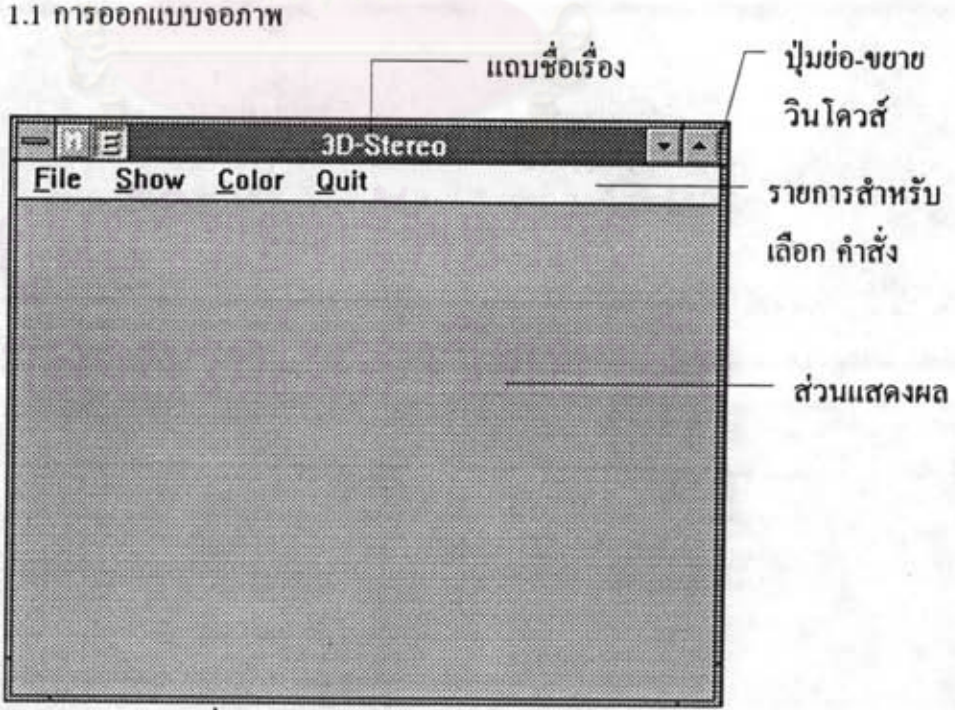
2 การออกแบบข้อความและการประมวลผล

3 การออกแบบโครงสร้างข้อมูล

4. การออกแบบการเชื่อมต่อกับภาคขับสเตอริโอ

1. การออกแบบจอภาพและส่วนติดต่อกับผู้ใช้

1.1 การออกแบบจอภาพ



รูปที่ 6.1 แสดงลักษณะของจอภาพ

จอภาพเป็นส่วนแสดงผลหลักที่สามารถย่อ-ขยาย และเคลื่อนย้ายได้ โดยแบ่งออกเป็นส่วนสำคัญ (ดูรูปที่ 6.1) ดังนี้

1.1.1 ส่วนแสดงผลสเตอริโอ เป็นพื้นที่สำหรับแสดงภาพที่ผู้ใช้สามารถย่อ-ขยาย หรือเคลื่อนย้ายไป ณ ที่ใดๆ บนจอภาพได้

1.1.2 แถบชื่อเรื่องแสดงชื่อโปรแกรม และแฟ้มข้อมูลที่กำลังแสดง

1.1.3 รายการสำหรับเลือกคำสั่ง เป็นส่วนที่ใช้แสดงรายการคำสั่งของโปรแกรม โดยจะเรียงอยู่ในแนวนอนใต้แถบชื่อเรื่อง การเลือกรายการทำได้โดยใช้เมาส์ชี้ไปยังรายการที่ต้องการ แล้วกดปุ่มซ้ายหรือกดแป้นพิมพ์ โดยการใช้ปุ่ม ALT พร้อมกับอักษณที่ขีดเส้นใต้ รายละเอียดของรายการจะปรากฏขึ้น เลือกรายการย่อยตามที่ต้องการอีกครั้ง ในแต่ละรายการ คำสั่งจะมีรายการย่อยดังนี้

1. File ใช้ในการเลือกแฟ้มข้อมูลเพื่อแสดงผล

2. Show ใช้ในการแสดงภาพสเตริโอ 3 มิติ โดยจะแบ่งออกเป็น 2 ภาวะ คือ

- Auto จะมีการหมุนภาพโดยอัตโนมัติ ซึ่งผู้ใช้สามารถปรับเปลี่ยนความเร็วในการหมุน โดย การกดปุ่ม + หรือ - เพื่อเพิ่ม-ลดความเร็วได้ตามต้องการ
- Manual ใช้แสดงภาพนิ่ง จะไม่มีการหมุนภาพโดยอัตโนมัติ แต่สามารถสั่งการได้โดยใช้เมาส์ หรือแป้นพิมพ์

3. Color ใช้ในการกำหนดสีพื้นของฉากหลังโดยจะมี 2 ลักษณะ คือ

- Default สีพื้นมาตรฐาน จะเป็นสีน้ำเงินเข้ม
- Set color สีพื้นอื่นๆ ซึ่งสามารถเลือกได้ตามแต่ผู้ใช้ต้องการ

4. Quit ใช้ในการจบการทำงาน

1.1.4 ปุ่มย่อ-ขยายวินโดวส์ ใช้ในการลดขนาดวินโดวส์จนเป็นสัญญรูป หรือขยายวินโดวส์ให้เต็มจอแสดงผล

1.2 การออกแบบส่วนติดต่อกับผู้ใช้

การออกแบบในส่วนนี้ จะเป็นไปตามรูปแบบการสร้างโปรแกรมบนไมโครซอฟต์วินโดวส์ ดังนี้

1.2.1 ระบบรายการเลือกเป็น แบบดึงลง ผู้ใช้สามารถในการเลือก รายการต่างๆ ได้

1.2.2 กล่องคำโต้ตอบ สำหรับการติดต่อกับผู้ใช้ ในการรอรับค่าพารามิเตอร์ หรือแสดงข้อมูล สำคัญ

2. การออกแบบข้อความและการประมวลผล

ข้อความที่ใช้ ในโปรแกรมมีอยู่ด้วยกัน 2 กลุ่ม คือ

1. กลุ่มที่ใช้ใน ภาคขับสเตอริโอ ซึ่งมีดังนี้

ตาราง 6.1 แสดงข้อความ ที่ใช้ในภาคขับสเตอริโอ

ชนิดของข้อความ	หน้าที่และการประมวลผล
DRV_LOAD	จัดการตรวจสอบการ์ดแสดงผล, ตรวจสอบการเชื่อมต่อกับ UNIVBE และกำหนดค่าเริ่มต้นให้กับอุปกรณ์ฮาร์ดแวร์
DRV_FREE	ปลดปล่อยหน่วยความจำและซีเลคเตอร์ที่เคยครอบครองอยู่ออก จากนั้นถอนตัวออกจากหน่วยความจำ
DRV_3D_ON	เข้าสู่ภาวะ แสดงผลแบบ สเตอริโอ
DRV_3D_OFF	ออกจากภาวะแสดงผลแบบสเตอริโอ กลับสู่ภาวะปกติ

2. กลุ่มที่ใช้ใน ส่วนการแสดงภาพ ซึ่งมีดังนี้

ตาราง 6.2 แสดงข้อความ ที่ใช้ในส่วนการแสดงภาพ

ชนิดของข้อความ	หน้าที่และการประมวลผล
WM_CREATE	สร้างส่วนแสดงผลหลัก และกำหนดค่าเริ่มต้นอื่นๆ
WM_ERASEBKGND	ลบฉากหลังของวินโดวส์ เพื่อทำการวาดภาพใหม่
WM_PAINT	วาดภาพใหม่ทั้งวินโดวส์
WM_SIZE, WM-MOVE	คำนวณตำแหน่งและ/หรือขนาดของวินโดวส์ พร้อมทั้ง ทำการปรับขนาดของภาพให้เหมาะสม
WM_COMMAND	จัดการเกี่ยวกับการเลือกรายการของผู้ใช้ ซึ่งจะมีค่าดังนี้
1. ID_FILE_OPEN	เปิดแฟ้มข้อมูลที่จะนำมาแสดง
2. ID_SHOW_AUTO	แสดงภาพสเตอริโอ โดยหมุนภาพอัตโนมัติ
3. ID_SHOW_MANUAL	แสดงภาพสเตอริโอ โดยจะไม่มีการหมุนภาพอัตโนมัติ
4. ID_QUIT	เลิกทำงาน
5. ID_SETCOLOR	กำหนดสีพื้นของวินโดวส์
6. ID_DEFAULT	กำหนดสีพื้นกลับเป็นสีน้ำเงิน

ตาราง 6.2 แสดงข้อความ ที่ใช้ในส่วนการแสดงภาพ (ต่อ)

ชนิดของข้อความ	หน้าที่และการประมวลผล
WM_KEYDOWN	ส่งรหัสของปุ่มต่างๆ บนแป้นพิมพ์ โดยมีค่าดังนี้
1. VK_LEFT	ปุ่มลูกศรซ้าย, หมุนภาพทางซ้าย
2. VK_RIGHT	ปุ่มลูกศรขวา, หมุนภาพทางขวา
3. VK_UP	ปุ่มลูกศรบน, หมุนภาพด้านบน
4. VK_DOWN	ปุ่มลูกศรล่าง, หมุนภาพด้านล่าง
5. VK_END	หยุดการแสดงภาพสเตอริโอ และกลับสู่ภาวะปกติ
6. VK_ADD	เพิ่มความเร็วในการหมุนภาพ
7. VK_SUBTRACT	ลดความเร็วในการหมุนภาพ
WM_MOUSEMOVE	หมุนภาพตามที่มีการเคลื่อนเมาส์
WM_RBUTTONDOWN	หยุดการแสดงภาพสเตอริโอ และกลับสู่สภาวะปกติ
WM_3D_ON	เตรียมภาพ เพื่อเข้าสู่ภาวะสเตอริโอ
WM_DESTROY	ยกเลิกการทำงานของโปรแกรมทั้งหมด และทำลาย วินโดวส์ต่างๆ ที่เกี่ยวข้อง

3. การออกแบบโครงสร้างข้อมูล

เนื่องจากโปรแกรมถูกแบ่งออกเป็น 2 ส่วน ดังนั้น โครงสร้างข้อมูลที่ใช้ในงานวิจัยนี้ จึงแบ่งออกเป็น 2 ส่วน คือ

3.1 โครงสร้างข้อมูล ที่ใช้ในภาคขับสเตอริโอ

3.1.1 โครงสร้างข้อมูล VgaInfoBlock

VESASignature
VESA Version
OEMStringPtr
Capabilities
VideoModePtr

TotalMemory
Reserved

โครงสร้างข้อมูลนี้มีขนาดรวม 256 ไบต์ ใช้ในการสอบถามข้อมูลของการ์ดแสดงผลชนิดซุปเปอร์วีจีเอ ซึ่งภาคขับอุปกรณ์จะไใช้เพื่อตรวจสอบว่า การ์ดแสดงผลนี้ สามารถทำงานในภาวะสเตอริโอได้หรือไม่ โดยมีรายละเอียดดังนี้

VESASignature เป็นแอร์เรย์ขนาด 4 ไบต์ บรรจุอักษร 4 ตัว คือ VESA

VESAVersion เป็นขนาด 16 บิต ใช้บอกหมายเลขรุ่นของ VESA

OEMString ตัวชี้ขนาด 32 บิต ชี้ไปยังหน่วยความจำที่ใช้เก็บข้อความระบุชื่อ ผู้ผลิต การ์ดแสดงผล, ในงานวิจัยนี้จะต้องมีข้อความ "Universal VESA VBE 4"

Capabilities มีขนาด 32 บิต ใช้บ่งชี้ความสามารถต่างๆ ของการ์แสดงผล

VideoModePrt ตัวชี้ขนาด 32 บิต ทำหน้าที่ชี้ไปยังหน่วยความจำที่เก็บหมายเลขภาวะการแสดงผลที่การ์ด นี้รองรับได้

TotalMemory มีขนาด 16 บิต ใช้ระบุขนาดของหน่วยความจำบนการ์แสดงผล มีหน่วยเป็น 64 KB.

Rererved มีขนาด 236 ไบต์ สงวนไว้ใช้ในอนาคต

3.1.2 โครงสร้างข้อมูล PmInfoBlock

writeBankLen
writeBank
readBankLen
readBank
newPageLen
newPage

โครงสร้างข้อมูลนี้ใช้ในการเชื่อต่อกับ UNIVBE ในการกำหนดตำแหน่าง และขนาดของชุดคำสั่งที่ใช้ทำหน้าที่ควบคุมการเขียน-อ่าน หน่วยความจำบนการ์แสดงผล รวมไปถึงชุดคำสั่งที่ใช้ควบคุมการสลับหน่วยความจำ มีโครงสร้างดังนี้

writeBankLen มีขนาด 16 บิต ทำหน้าที่กำหนดความยาวของชุดคำสั่งควบคุมการเขียน

writeBank	ตัวชี้ขนาด 32 บิต ทำหน้าที่ชี้ไปยังชุดคำสั่งควบคุมการเขียน
readBankLen	มีขนาด 16 บิต กำหนดความยาว ชุดคำสั่งควบคุมการอ่าน
readBank	ตัวชี้ขนาด 32 บิต ชี้ไปยัง ชุดคำสั่งควบคุมการอ่าน
newPageLen	มีขนาด 16 บิต กำหนดความยาว ชุดคำสั่งควบคุมการสลับหน่วยความจำ
newPage	ตัวชี้ขนาด 32 บิต ชี้ไปยัง ชุดคำสั่งควบคุมการสลับหน่วยความจำ

3.2 โครงสร้างข้อมูลที่ใช้ในการแสดงผล

การแสดงผลในงานวิจัยนี้ ใช้รูปแบบของแฟ้มข้อมูล .3D และ .3DV. (ดูโครงสร้างในภาคผนวก ง) ซึ่งบรรยายรายละเอียดพิกัด, เส้น และสีที่ใช้ในการสร้างภาพ โดยใช้พิกัด 3 มิติ แบบเป็นเลขทศนิยม (Floating Point Number) ทั้งหมด แต่เนื่องจากการคำณวนเลขทศนิยมนั้น ใช้เวลาค่อนข้างมาก เพื่อเป็นการเพิ่มความเร็วในการสร้างภาพ เราจึงต้องทำการเปลี่ยนพิกัดตำแหน่งจากแบบทศนิยม ลงมาเป็นพิกัดจำนวนเต็ม(Integer) ทำให้ต้องมีโครงสร้างข้อมูล 2 ชุด ที่มีลักษณะการจัดเรียงและความหมายเหมือนกันแต่บรรจุข้อมูลคนละ ชนิดกัน (ทศนิยม กับจำนวนเต็ม) ซึ่งมีโครงสร้างและความหมายดังนี้

3.2.1 โครงสร้างสำหรับเลขทศนิยม

3.2.1.1 โครงสร้างข้อมูล Tfpnt ใช้กำหนดพิกัดของจุดภาพในแบบพิกัดมือซ้าย

x	y	z	n

x	เป็นเลขทศนิยม สำหรับ พิกัดแกน x
y	เป็นเลขทศนิยม สำหรับ พิกัดแกน y
z	เป็นเลขทศนิยม สำหรับ พิกัดแกน z
n	เป็นจำนวนเต็มในที่ นี้มีค่าเท่ากับ 1

3.2.1.2 โครงสร้างข้อมูล Tfpntl. ทำหน้าที่บรรยายตำแหน่งของจุดภาพทั้งหมด มีโครงสร้างดังนี้

pnta	npnt

pnta เป็นตัวชี้ขนาด 16 บิต ไปยังกลุ่มของโครงสร้างข้อมูล TfPnt ที่เรียงติดกันในลักษณะเป็นแอร์เรย์

nput มีขนาด 16 บิต ทำหน้าที่ระบุขนาดของแอร์เรย์

3.2.1.3 โครงสร้างข้อมูล Tfsegm ทำหน้าที่บรรยายส่วนของเส้นตรงที่ใช้ในการเชื่อมระหว่างจุดภาพ 2 จุดเข้าด้วยกัน มีโครงสร้าง ดังนี้

p	color

p มีขนาด 16 บิต ระบุหมายเลขจุดที่เชื่อมต่อ

color มีขนาด 16 บิต ระบุสีของเส้นที่ให้ทำการวาด

3.2.1.4 โครงสร้างข้อมูล Tfsegml ทำหน้าที่บรรยายลักษณะของเส้นทั้งหมดในภาพ มีโครงสร้างดังนี้

segma	nsegm

segma เป็นตัวชี้ขนาด 16 บิตไปยังกลุ่มของโครงสร้างข้อมูล Tfsegm ที่เรียงติดกันในลักษณะเป็นแอร์เรย์

nsegm มีขนาด 16 บิต ใช้ระบุจำนวนเส้นที่ต้องวาดในภาพนี้

3.2.1.5 โครงสร้างข้อมูล Tffrm ทำหน้าที่บรรยายถึงจำนวนจุดและจำนวนเส้นทั้งหมดที่ใช้ในการวาดภาพ มีโครงสร้างดังนี้

segml	pntl

Segml เป็นตัวชี้ขนาด 16 บิต ชี้ไปยังโครงสร้างข้อมูล Tfsegml

pntl เป็นตัวชี้ขนาด 16 บิต ชี้ไปยังโครงสร้างข้อมูล Tfpntl

3.2.2 โครงสร้างข้อมูลสำหรับเลขจำนวนเต็ม

มีลักษณะคล้ายคลึงกับโครงสร้างข้อมูล สำหรับเลขทศนิยม ต่างกันเพียงที่โครงสร้างข้อมูล Tpnt จะระบุพิกัดเป็นเลขจำนวนเต็ม

3.2.2.1 โครงสร้างข้อมูล Tpnt ใช้กำหนดพิกัดของจุดภาพในแบบพิกัดมือซ้าย

x	y	z	n

x เป็นเลขทศนิยม สำหรับ พิกัดแกน x

y เป็นเลขทศนิยม สำหรับ พิกัดแกน y

z เป็นเลขทศนิยม สำหรับ พิกัดแกน z

n เป็นจำนวนเต็มในที่ นี้มีค่าเท่ากับ 1

3.2.2.2 โครงสร้างข้อมูล Tpntl. ทำหน้าที่บรรยายตำแหน่งของจุดภาพทั้งหมด มีโครงสร้างดังนี้

pnta	npnt

pnta เป็นตัวชี้ขนาด 16 บิต ไปยังกลุ่มของโครงสร้างข้อมูล TPnt ที่เรียงติดกันในลักษณะเป็นแอร์เรย์

nput มีขนาด 16 บิต ทำหน้าที่ระบุขนาดของแอร์เรย์

3.2.2.3 โครงสร้างข้อมูล Tsegm ทำหน้าที่บรรยายส่วนของเส้นตรงที่ใช้ในการเชื่อมระหว่างจุดภาพ 2 จุดเข้าด้วยกัน มีโครงสร้าง ดังนี้

p	color

p มีขนาด 16 บิต ระบุหมายเลขจุดที่เชื่อมต่อ

color มีขนาด 16 บิต ระบุสีของเส้นที่ให้ทำการวาด

3.2.2.4 โครงสร้างข้อมูล Tsegml ทำหน้าที่บรรยายลักษณะของเส้นทั้งหมดในภาพ มีโครงสร้างดังนี้

segma	nsegm

segma เป็นตัวชี้ขนาด 16 บิตไปยังกลุ่มของโครงสร้างข้อมูล Tsegm ที่เรียงติดกันในลักษณะเป็นแอร์เรย์

nsegm มีขนาด 16 บิต ใช้ระบุจำนวนเส้นที่ต้องวาดในภาพนี้

3.2.2.5 โครงสร้างข้อมูล Tfrm ทำหน้าที่บรรยายถึงจำนวนจุดและจำนวนเส้นทั้งหมดที่ใช้ในการวาดภาพ มีโครงสร้างดังนี้

segml	pntl

Segml เป็นตัวชี้ขนาด 16 บิต ชี้ไปยังโครงสร้างข้อมูล Tsegml

pntl เป็นตัวชี้ขนาด 16 บิต ชี้ไปยังโครงสร้างข้อมูล Tpntl

4. การออกแบบการเชื่อมต่อกับภาคขับสเตอริโอ

วิธีการที่โปรแกรมแสดงภาพเชื่อมต่อกับภาคขับสเตอริโอได้นั้น ทำโดยการเรียกใช้ API OpenDriver("Stereo", "3D", NULL) ซึ่งหลังการเรียกใช้ API ดังกล่าว จะให้ผลลัพธ์เป็นค่าแฮนเดิลของภาคขับอุปกรณ์ (Driver Handle) ซึ่งโปรแกรมแสดงภาพจะต้องเก็บค่านี้ไว้ใช้ในการติดต่อกับภาคขับสเตอริโอต่อไป ตัวอย่างการใช้งานเป็นดังนี้

```
HDRV hDrv;
hDrv = OpenDriver("Stereo", "3D", NULL);
if (hDRv == NULL)
        return ERROR;
```

สำหรับการควบคุมภาคขับสเตอริโอ จะทำโดยการส่งข้อความจากโปรแกรมแสดงภาพไปให้ภาคขับสเตอริโอ โดยจะมีข้อความอยู่ 2 ข้อความ ซึ่งโปรแกรมแสดงภาพจะต้องดำเนินการตามขั้นตอนและวิธีใช้ข้อความแต่ละข้อความดังนี้

4.1 DRV_3D_ON

ใช้สำหรับภาวะเข้าสู่การแสดงผลแบบสเตอริโอ โดยมีขั้นตอนการใช้ดังนี้

4.1.1 ทำการลบเคอร์เซอร์ออกจากจอภาพ โดยการใช้ API ShowCursor(FALSE)

4.1.2 ทำการเตรียมภาพสำหรับตาทั้งสองข้าง (ดูรายละเอียดในหัวข้อการออกแบบฟังก์ชั่นส่งออก)

4.1.3 ทำการส่งข้อความโดยการใช้ API SendDriverMessage(hDrv, DRV_3D_ON, 0, 0)

หลังขั้นตอนนี้ระบบจะเข้าสู่ภาวะการแสดงผลแบบสเตอริโอ เหตุผลของการลบเคอร์เซอร์ออกเนื่องมาจากการวาดภาพเคอร์เซอร์ของวินโดวส์นั้นทำอยู่ในหน้าแสดงผลเพียงหน้าเดียว ซึ่งในภาวะที่ต้องแสดงผลสลับไปมาทั้ง 2 หน้าจะทำให้การวาดเคอร์เซอร์ผิดพลาด และโดยเหตุนี้ขณะที่ระบบอยู่ในภาวะแสดงผลแบบสเตอริโอ จึงไม่สามารถใช้เมาส์ในการชี้ตำแหน่งบนจอภาพได้ ตัวอย่างการใช้งานแสดงได้ดังนี้

```
ShowCursor(FALSE);              //Turn off cursor
DrawImage(Page0);               //Create image for page 0
CopyScreen(Page0,Page1);        //Copy background to page 1
ActivePage(Page1);              //Active display memory for page 1
EraseImage(Page0);              //Erase page 0 image in page 1 memory
DrawImage(Page1);               //Create image for page 1
ActivePage(Page0);              //Active page 0 as normal
SendDriverMessage(hDrv, DRV_3D_ON, 0, 0)   //Enter stereo mode
```

4.2. DRV_3D_OFF

ใช้สำหรับออกจากภาวะการแสดงผลแบบสเตอริโอ โดยมีขั้นตอนการใช้ดังนี้

4.2.1 ส่งข้อความโดยใช้ API SenDriverMessage(hLDrv, DRV_3D_OFF, 0, 0)

4.2.2 กำหนดให้หน้าแสดงผลหมายเลขศูนย์แอคตีฟ โดยใช้ฟังก์ชั่นส่งออก ActivePage (Page 0)

4.2.3 กำหนดให้แสดงหน้าแสดงผลหมายเลขศูนย์ โดยใช้ฟังก์ชั่นส่งออก VisualPage (Page 0)

4.2.4 ทำการวาดเคอร์เซอร์ โดยการใช้ API ShowCursor(TRUE)

หลังขั้นตอนนี้ระบบจะกลับเข้าสู่การแสดงผลแบบธรรมดา ตัวอย่างการใช้งานแสดงได้ดังนี้

```
SendDriverMesage(hDRv, DRV_3D_OFF, 0, 0); //Exit stereo mode
ActivePage(Page 0);          //Turn page 0 active
VisualPage(Page 0);          //Display page 0
ShowCursor(TRUE);            //Turn on cursor
```