บทที่ 3

แนวคิดและทฤษฎี

จากการวิจัยเรื่องการพัฒนาระบบการจัดการข้อสอบแบบปรนัยนี้ จุดประสงค์เพื่อทำให้การสอบหรือการทดสอบสามารถทำได้ด้วยเครื่องคอมพิวเตอร์ โดยข้อสอบนั้นสามารถแสดงผลได้ทั้งเป็นแบบข้อความ รูปภาพ และเสียง ด้วยเหตุนี้จึงมีความจำเป็นต้องทราบถึงทฤษฎีต่างๆ ที่เกี่ยวข้อง แล้วจึงนำมาประยุกต์และพัฒนาเพื่อทำให้เป็นระบบ ทฤษฎีต่างๆ มีดังต่อไปนี้

3.1 การจัดเก็บข้อมูลประเภทข้อความ (Text File)

ข้อมูลชนิดนี้จะประกอบไปด้วย ตัวอักษร ตัวเลข และตัวอักษรแบบพิเศษ เช่น ก ข A B c d 1 2 3 # % & ฯลฯ การเก็บข้อมูลชนิดนี้เป็นแบบง่ายๆ แต่ละตัวอักษรถูกเก็บเรียงกันตั้งแต่ต้นจนจบ จะมีที่น่าสังเกตตรงที่อาจจะมีอักษรพิเศษบางตัวที่ทำหน้าที่บอกความหมายภายในข้อความด้วย เช่น บอกการเริ่มต้นบรรทัดใหม่ ($0D0A_{16}$) เป็นต้น

ข้อมูลที่เป็นข้อความและจัดเก็บในระบบการจัดการข้อสอบแบบปรนัย เช่น ข้อความโจทย์ ข้อความตัวเลือก และรายงานต่างๆ ดังแสดงในรูปที่ 3.1

รายงาน เกรด

ลำดับที่	รหัส	คำนำหน้า	ชื่อ	นามสกุล	เกรด
1	C518272	นาย	กอบกิจ	สหัสรังษี	B
2	C518274	นาย	วีระชัย	แซ่ตง	B
3	C518276	นาย	สัญญา	มีความสุข	A

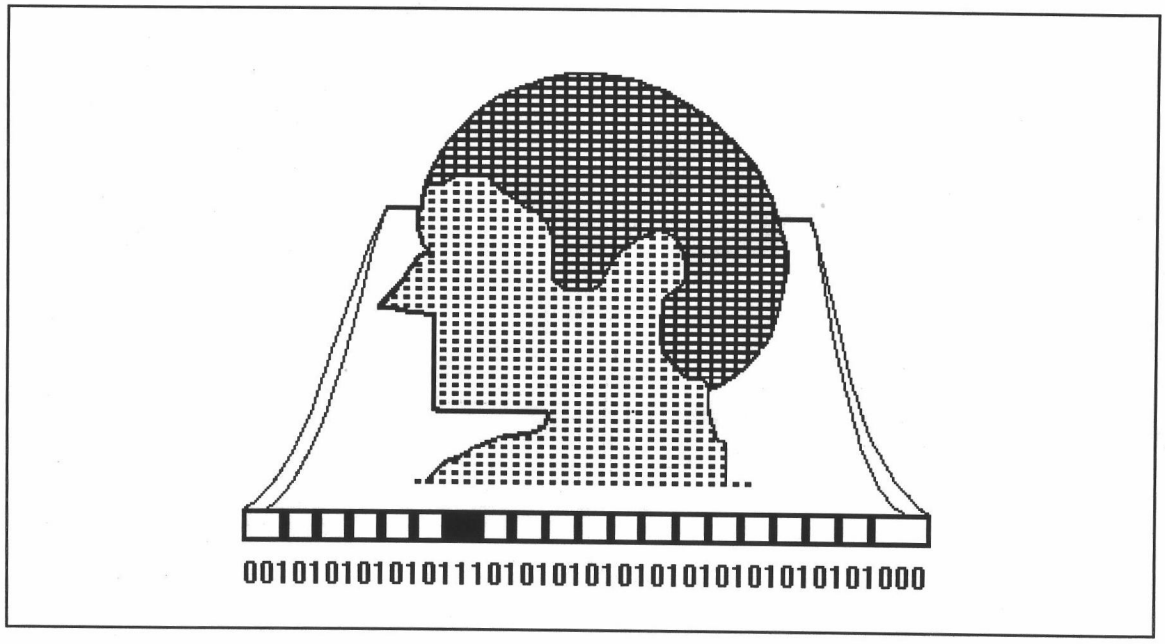
รูป 3.1 แสดงตัวอย่างข้อมูลประเภทข้อความ

3.2 ทฤษฎีภาพกราฟิก (Graphics represent theory)

การแสดงรูปภาพบนหน้าจอคอมพิวเตอร์ สามารถแสดงได้ 2 วิธีใหญ่ ๆ คือ แบบบิตแมพ (Bitmap representation) และแบบเวคเตอร์ (Vector representation)

3.2.1 แบบบิตแมพ (Bitmap representation)

เป็นวิธีการแสดงรูปภาพแบบง่ายๆ คือ การแบ่งภาพออกเป็นตาราง(Grid) และมีจุดเล็กหรือเรียกว่า พิกเซล (Pixel) ปรากฏอยู่ในแต่ละช่องของตารางดังกล่าว จุดเล็ก ๆ นั้น อาจจะเป็นสีขาว, สีดำ หรือสีอื่น ๆ ก็ได้ เมื่อรวมจุดเล็กเข้าด้วยก็จะมองเห็นเป็นรูปภาพขึ้น ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 One line of bitmap data from an image

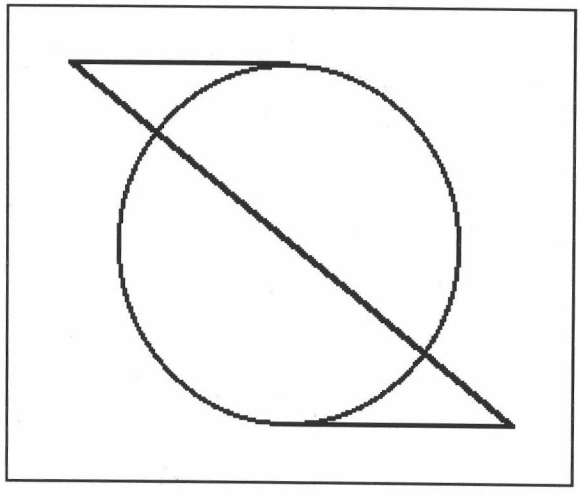
ตัวอย่าง บิตแมพที่เป็นภาพขาวดำ พิกเซลที่ปรากฏจะมีแค่ 2 สี คือ สีขาวและสีดำ ซึ่งมักจะใช้กับจอภาพแบบโมโนโครม (Monochrome) แต่ถ้าเป็นจอภาพแบบโมโนโครม

วีจีเอ (Monochrome VGA) สามารถแสดงระดับความเข้มของสีขาวและสีดำ หรือเรียกว่า "สีเทา" ("Gray Scale") โดยการกระจายความหนาแน่นของจุดสีขาวและสีดำคละกันไป ทำให้เกิดความแตกต่างของสีได้ และถ้าเป็นจอสีประเภทวีจีเอ (VGA or Vedio Graphics Adapter) หรือซุปเปอร์วีจีเอ (Supper VGA) พิกเซลที่ปรากฏบนหน้าจอสามารถแสดงสีต่าง ๆ ได้มากถึง 256 สีหรือ 16.7 ล้านสี ก็ขึ้นอยู่กับคุณสมบัติของวีจีเอการ์ด (VGA Card) และจอภาพ

3.2.2 แบบเวคเตอร์ (Vector representation)

เป็นวิธีการแสดงรูปภาพโดยการแสดงเป็นเส้นตรงเส้นโค้งหรือรูปร่างต่างๆ เช่น รูปสี่เหลี่ยม และวงกลม บางครั้งอาจจะมีส่วนของการระบุสีต่าง ๆ ลงบนภาพ เช่น การระบายสีพื้นลงในรูปวงกลม การสร้างเงาให้กับรูปภาพ เป็นต้น

ข้อมูลแบบเวคเตอร์นั้นจะมีส่วนคล้ายคำสั่งของโปรแกรม มีการเก็บในลักษณะข้อความภาษาอังกฤษหรือ ASCII CODE และนำคำสั่งต่าง ๆ มาแปลความหมาย เพื่อทราบตำแหน่งและจุดมุ่งหมายที่จะวาดภาพนั้นออกมา ตัวอย่างเช่น รูปวงกลมที่มีรัศมี 100 จุดศูนย์กลางอยู่ที่ X = 2.25 เซนติเมตร และ Y = 5 เซนติเมตร ข้อความของคำสั่ง คือ circle (100, 2250, 5000) เป็นต้น ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 A simple line drawing, suitable for vector representation

3.2.3 โครงสร้างแฟ้มข้อมูลกราฟิกรูปแบบ PCX

โครงสร้างแบบ PCX เป็นโครงสร้างชนิดหนึ่งที่มีมานานแล้วและยังเป็นที่นิยมอยู่จนถึงปัจจุบัน พัฒนาขึ้นโดยบริษัทแซดซอฟต์ (Zsoft Corporation) ซึ่งเป็นเจ้าของโปรแกรมวาดภาพพีซีเพ้นท์บรัช (PC Paintbrush) และได้มีการพัฒนาเพิ่มเติมใหม่จนปัจจุบัน โครงสร้างแฟ้มข้อมูลกราฟิกแบบ PCX จะขึ้นอยู่กับฮาร์ดแวร์ที่ใช้ในการสร้างรูปภาพ สามารถรวบรวมความหลากหลายของ Version ต่างๆ ได้ ดังแสดงในตารางที่ 3.1 - 3.2

ตารางที่ 3.1 แสดงลักษณะโครงสร้างข้อมูลแบบ PCX

Version number	Image Characteristics
0	สำหรับการแสดงภาพชนิดสีเดียว (monochrome display)
2	สำหรับการแสดงภาพ 16 สี (color display)
5	สำหรับการแสดงภาพ 256 สี (color display)

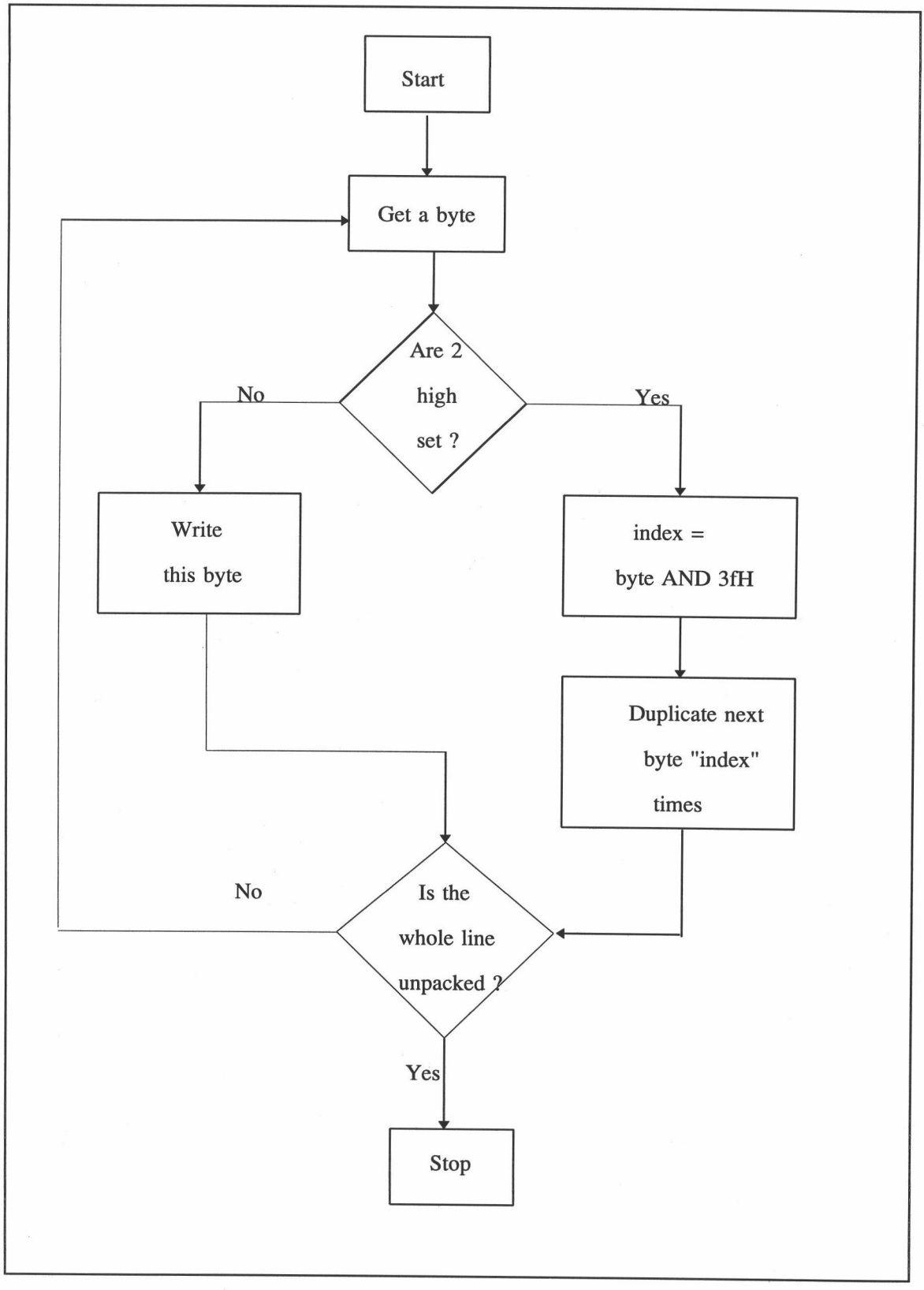
ตารางที่ 3.2 แสดงส่วนหัวของแฟ้มข้อมูล PCX

เริ่ม	ขนาด	รายละเอียด
0	1	Zsoft flag มีค่า 10 หรือ (0A hex) = ไฟล์ Zsoft PCX
1	1	Ver no. 0 = PC Paintbrush 2.5 2 = PC Paintbrush 2.8 with palette 3 = PC Paintbrush 2.8 without palette 4 = PC Paintbrush สำหรับ windows 5 = PC Paintbrush 3.0 หรือมากกว่าขึ้นไป เช่น PC Paintbrush; Publisher's Paintbrush
2	1	วิธีการเข้ารหัส (1 หมายถึง PCX)
3	1	จำนวนบิตต่อพิกเซล
4	8	ขอบเขตของภาพ มี 4 ค่า (ค่าละ 2 ไบต์) คือ Xmin,Ymin,Xmax,Ymax
12	2	ความละเอียดในการพิมพ์ภาพตามแนวนอน คือ จำนวนจุดต่อนิ้ว
14	2	ความละเอียดในการพิมพ์ภาพตามแนวตั้ง คือ จำนวนจุดต่อนิ้ว
16	48	รายละเอียดของสี
64	1	สำรอง
65	1	จำนวนบิตของ ระนาบ ที่ใช้แสดงภาพ
66	2	จำนวนไบต์ต่อบรรทัด หรือจำนวนไบต์ที่ใช้เก็บข้อมูลในแต่ละบรรทัด
68	2	ข้อมูลเกี่ยวกับสี (Header palette interpretation) 1 = สี หรือ ขาวดำ 2 = เกรย์สเกล (Gray Scale)
70	2	จำนวนพิกเซลบนหน้าจอตามแนวนอน
72	2	จำนวนพิกเซลบนหน้าจอตามแนวตั้ง
74	54	สำหรับ Fractal Programming

การบันทึกข้อมูลในแฟ้มข้อมูลแบบ PCX โดยวิธีการเข้ารหัสที่เรียกว่า Run Length Encoding (RLE) คือ เป็นวิธีการอัดข้อมูลรูปภาพ เพื่อลดขนาดของแฟ้มข้อมูล การเข้ารหัสแบบนี้มีรายละเอียดดังนี้ คือ ถ้าไบต์ใด ๆ มีค่าไม่เหมือนกับไบต์อื่น ๆ และ 2 บิตบนยังจะมีค่าไม่เท่ากับ '11' ไบต์นั้นจะเก็บลงแฟ้มทันทีโดยไม่ถูกแปลงค่าใด ๆ ทั้งสิ้น นอกนั้นที่ไม่เข้าเงื่อนไขจะมีตัวนับ (counter) นับจำนวนไบต์ที่เก็บลงแฟ้มแต่จำนวนซ้ำนั้นต้องไม่เกิน 63 ต้ว ถ้าเกินให้นำค่าตัวนับที่ได้ OR กับ 'c0H' แล้วเก็บค่าตัวนับนั้นลงบนแฟ้มแล้วตามด้วยค่าของไบต์นั้น จากนั้นจึงเริ่มนับ 1 ใหม่ ถ้ากรณีที่ไบต์เดียว(ไม่ซ้ำ)และมีค่าบิตบนเป็น '11' ให้เขียนตัวนับเท่ากับ 1 (หรือ ทำการ OR กับ 'c0H' จะได้ 'c1H') ลงในแฟ้มข้อมูลและตามด้วยค่าไบต์นั้น ดังแสดงในตารางที่ 3.3 และรูปที่ 3.4

ตารางที่ 3.3 สรุปการเข้ารหัสแบบ RLE

เงื่อนไข	การกระทำ
ค่าไบต์ที่เหมือนไบต์อื่น (หรือซ้ำ)	ทำการนับเพิ่ม Count = Count + 1
ค่าไบต์ < 'c0H' และไม่เหมือนไบต์อื่น	เก็บลงแฟ้มเลย
ค่าไบต์ > 'c0H' และไม่เหมือนไบต์อื่น	เก็บ 'c0H' ลงแฟ้มและตามด้วยค่าไบต์นั้น
ถ้า Count > 63	ให้ Count = 1 และเก็บค่า 'ffH'
	และค่าไบต์ลงแฟ้ม



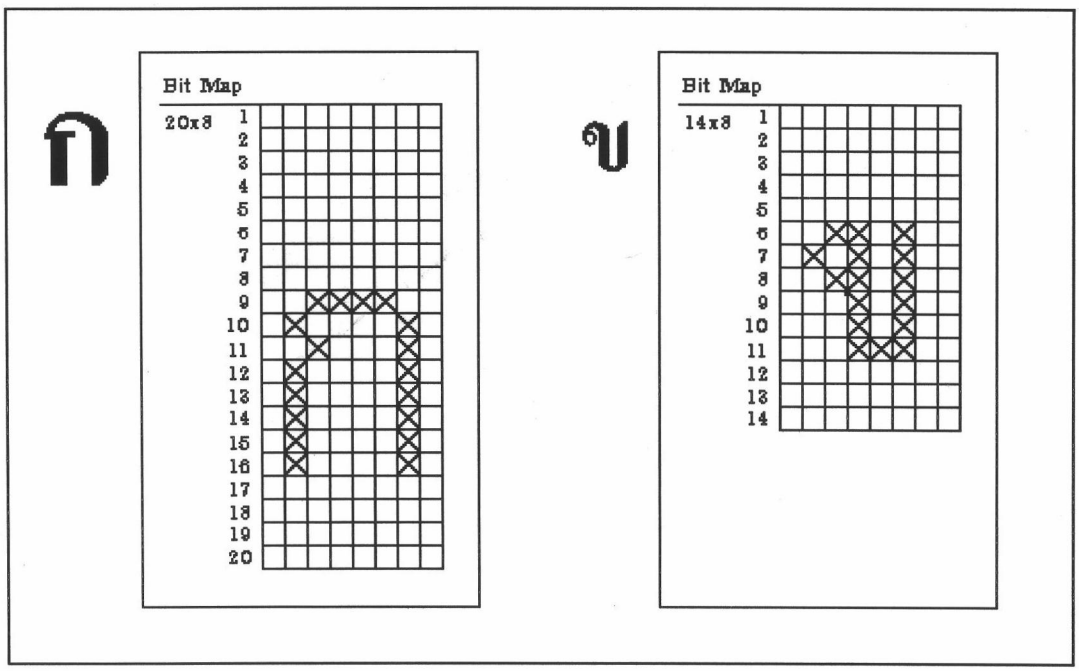
รูปที่ 3.4 แสดงการอ่านแฟ้มข้อมูลภาพ PCX

3.3 การแสดงตัวอักษรบนหน้าจอและการรับตัวอักษรจากแป้นพิมพ์ภาษาไทยในโหมดกราฟิก

3.3.1 การแสดงข้อความตัวอักษรบนจอภาพในโหมดกราฟิก

หลักการเขียนโปรแกรมเพื่อให้แสดงข้อความบนหน้าจอทั้งภาษาไทยและภาษาอังกฤษในโหมดกราฟิก มีขั้นตอนดังนี้

1) เริ่มจากการสร้างฟอนต์ของตัวอักษร และเก็บฟอนต์ลงแฟ้มข้อมูลไว้เพื่อนำกลับมาใช้ภายหลัง การสร้างฟอนต์ได้จะต้องมีโปรแกรมหรือเครื่องมือที่ทำหน้าที่เป็นเอดิเตอร์โดยการสร้างตารางที่จะบรรจุตัวอักษร และสามารถลากเส้นลงบนตารางทำให้เกิดเป็นภาพของตัวอักษรต่างๆ ผู้สร้างสามารถออกแบบรูปร่างของตัวอักษรได้ จะมีรูปแบบเป็นอย่างไรก็ได้ขึ้นอยู่กับความเห็นชอบของผู้ออกแบบเอง แต่จะต้องอยู่ภายใต้กฎเกณฑ์ของการสร้างด้วย กฎเกณฑ์ที่ว่านี้ คือ เรื่องของขนาดของตาราง อาจจะมีขนาด 20x8 (โหมด VGA) หรือ 14x8 (โหมด EGA) ในกรณีที่มีขนาด 20x8 หมายถึง ฟอนต์ที่มีขนาดจำนวนแถวนอน (row) 20 แถว และจำนวนแถวตามแนวตั้ง (column) 8 แถว ส่วนกรณีฟอนต์ที่มีขนาด 14x8 หมายถึง ฟอนต์ที่มีขนาดจำนวนแถวตามแนวนอน (row) 14 แถว และขนาดจำนวนแถวตามแนวตั้ง (column) 8 แถว ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 ตัวอย่างการแสดงตัวอักษรในรูปแบบฟอนต์ขนาด 20x8 และ 14x8

จุดหนึ่งจุดที่ในตารางจะแทนด้วยเลขฐานสอง หมายถึง ในตารางที่ลากเส้นเพื่อวาดรูปตัวอักษรไว้นั้นจะมีลักษณะเป็นจุด จุดใดที่มีการลากเส้นผ่านจุดนั้นจะแทนด้วยค่า 1 และจุดใดที่ว่างเปล่าหรือไม่มีการลากเส้นผ่านจะแทนด้วยค่า 0 ตัวอย่างเช่น ถ้ามีแถวหนึ่งในแนวนอนมีการลากเส้นผ่านที่ตำแหน่งที่ 1, 3 และ 7 จะสามารถเขียนออกมาในรูปเลขฐานสองคือ 10100010 และแปลงเป็นเลขฐาน 10 จะมีค่า 162 หรือถ้าแปลงเป็นเลขฐาน 16 จะมีค่า A2 ในคอมพิวเตอร์จะเก็บข้อมูลเหล่านี้ในรูปแบบของเลขฐานสองหรือที่เรียกว่า "บิต" และเก็บในรูปแบบของเลขฐานสิบหกหรือเรียกว่า "ไบต์" หรือกล่าวอีกในหนึ่งก็คือ ในหนึ่งไบต์จะประกอบด้วย 8 บิต และที่กล่าวมาทั้งหมดก็เพื่อที่จะชี้ให้เห็นว่า ตารางหนึ่งตารางที่มีแถวตามแนวนอนแต่ละแถวขนาด 8 จุดหรือ 8 ช่อง จะใช้เนื้อที่ 8 บิตหรือ 1 ไบต์ ฉะนั้นถ้าตัวอักษรหนึ่งตัวมีขนาด 20x8 ก็จะใช้เนื้อที่ทั้งหมด 20 ไบต์ และถ้าเป็นขนาด 14x8 ก็จะใช้เนื้อที่ทั้งหมด 14 ไบต์ ในเครื่องคอมพิวเตอร์มีการอ้างถึงตัวอักษรใดๆ ก็ตามจะแทนด้วยค่าที่เป็นตัวเลขทางคณิตศาสตร์เสมอ โดยหนึ่งตัวอักษรจะแทนด้วยขนาด 1 ไบต์หรือ 8 บิต 1 ไบต์ จะสามารถแทนค่าได้ตั้งแต่ 0 - 255 นั่นหมายความว่า 1 ไบต์สามารถแทนตัวอักษรได้ถึง 256 ตัว เพราะฉะนั้นการสร้างฟอนต์หนึ่งฟอนต์จะต้องสร้างพร้อมกันทั้งหมด 256 ตัว สรุปคือ ในกรณีที่ต้องการสร้างฟอนต์ที่มีขนาด 20 x8 จะต้องใช้เนื้อที่ทั้งหมดถึง 256x20 หรือเท่ากับ 5120 ไบต์ และในกรณีที่ต้องสร้างฟอนต์ขนาด 14x8 จะต้องใช้เนื้อที่ทั้งหมด 256x14 หรือเท่ากับ 3584 ไบต์ สำหรับงานวิจัยครั้งนี้มีฟอนต์ต่างๆ ทั้งหมด 6 ฟอนต์ ซึ่งเก็บอยู่ในแฟ้มข้อมูลต่างๆ ดังนี้ คือ NORMAL.X20, NORMAL.X14, ITALIC.X20, OFFSET.X20, COMPUTE.X20 และ LIGHT.X20 แต่การวิจัยครั้งนี้เลือกใช้ฟอนต์ที่มีขนาด 14x8 หรือ NORMAL.X20 เท่านั้น

2) หลังจากสร้างฟอนต์แล้ว ต่อไปก็ทำการเขียนฟังก์ชั่นให้อ่านฟอนต์ที่อยู่ในแฟ้มข้อมูลเข้ามาเก็บไว้ในหน่วยความจำ สำหรับบริการให้ฟังก์ชั่นภาษาไทยทั้งหลายเรียกใช้ฟังก์ชั่นการอ่านฟอนต์จากแฟ้มข้อมูล ดังแสดงในรูปที่ 3.6

```
//-----------------------------------------
//        Read data font from file *.X*
//-----------------------------------------
int readfont()
{
        FILE    *fp;
        int  i,j,handle;
        long     length;
        unsigned long size;
        char far *temp;
        char *tempfile='\0';
        for ( i=0;i<NUMFONT;i++ )  {
                fp = NULL;
                handle = 0;
                size   = 0;
                strcpy(tempfile,PATHFONT);
                strcat(tempfile,FONT[i].name);
                if ((fp = fopen(tempfile,"rb")) == NULL) {
                        box_error("ไม่สามารถเปิดแฟ้มตัวอักษรได้ !","");
                        return(NO);
                }
                handle = fileno(fp);
                if ((handle = fileno(fp)) == 0) {
                        return(NO);
                }
                FONT[i].buf_font = temp = (char far *)farmalloc(size =
                        filelength(handle));
```

รูปที่ 3.6 แสดงฟังก์ชั่นการอ่านฟอนต์จากแฟ้มข้อมูล

```
            if (temp == NULL) {
                box_error("เนื้อที่หน่วยความจำไม่พอ","ไม่สามารถ
                    เก็บแฟ้มตัวอักษรได้");
            return(NO);
            }
            fread((void *)FONT[i].buf_font,1,size,fp);
            fclose(fp);
        }
        return(YES);
}
```

รูปที่ 3.6 แสดงฟังก์ชั่นการอ่านฟอนต์จากแฟ้มข้อมูล (ต่อ)

3) เมื่อนำฟอนต์มาเก็บไว้ในหน่วยความจำแล้ว ก็เขียนฟังก์ชั่นเพื่อทำการแสดงตัวอักษรลงบนหน้าจอ แต่เนื่องจากตัวอักษรในภาษาไทยมีได้ถึง 3 ระดับ เช่นคำว่า "ที่อยู่" จึงต้องมีการจัดระดับของตัวอักษรออกเป็น 3 ประเภท คือ อักษรระดับบน อักษรระดับกลาง และอักษรระดับล่าง เมื่อจะนำอักษรต่างๆ แสดงออกทางหน้าจอต้องทำการตรวจสอบก่อนว่า ตัวอักษรนั้นอยู่ในระดับใด โดยดูจากตารางที่เขียนไว้ด้วยภาษา C ดังแสดงในรูปที่ 3.7 (Code_Level Table) เมื่อทราบระดับของตัวอักษรแล้วก็ทำการอ่านรูปแบบของฟอนต์ในหน่วยความจำ และใช้ฟังก์ชั่น putpixel เพื่อนำค่าต่างๆ ไปแสดง โดยแต้มจุดลงบนหน้าจอทำให้เกิดภาพเป็นตัวอักษร แต่การทำงานจริงๆ นั้น เพื่อให้แสดงข้อความ (ประโยคหรือคำ) จะใช้ฟังก์ชั่น put_str ซึ่งเป็นฟังก์ชั่นหลัก ฟังก์ชั่นนี้จะมีการเรียกฟังก์ชั่นต่างๆ มากมายเข้ามาร่วมด้วย ดังแสดงในรูปที่ 3.8

```
// Map each character to identify level
//        0 - not a character        1 - middle (normal) level
//        2 - top level character    3 - bottom level character
//        -1 - attribute character
signed char code_level[255] = {
        0, 0,-1, 0, 0,-1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
        0, 0,-1,-1,-1, 0,-1,-1, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 2, 1, 1, 2, 2, 2, 2, 3, 3, 1, 0, 0, 0, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
        };
```

รูปที่ 3.7 แสดงตารางระดับตัวอักษร (Code_Level Table)

```
void put_str(unsigned char *st,unsigned cur_x,unsigned cur_y,int color)
{
        int x,y;
        register unsigned char  *ptrtmp;
        char far *ptr;      // change by kk
        ptr = &FONT[cur_font].buf_font[*st * getfonth()];
        for(y=0;y<getfonth();y++) {
                for(x=0;x<getfontw();x++) {
                        if((*ptr << x) & 0x80) {
                                putpixel(cur_x+x,cur_y+y,color);
                        }
                }
                ptr++;
        }
}
```

รูปที่ 3.8 แสดงฟังก์ชันการแสดงอักษรทั้งภาษาไทยและภาษาอังกฤษ

3.3.2 การรับข้อมูลภาษาไทยจากแป้นพิมพ์ในโหมดกราฟิก

การรับข้อมูลจากแป้นพิมพ์ เมื่อไหร่ก็ตามที่มีการกดปุ่มใดๆ บนแป้นพิมพ์เครื่องคอมพิวเตอร์ จะรับค่าเข้ามาหนึ่งค่า จะต้องนำค่านั้นไปเปรียบเทียบค่าจากตาราง โดยที่ตารางนี้จะมีการเก็บรหัสภาษาไทยไว้ตามตำแหน่งของแป้นพิมพ์ จึงทำให้ทราบว่าแป้นพิมพ์ที่กดเข้ามาหมายถึงตัวอักษรใด ตัวอย่างตารางแสดงไว้ในรูปที่ 3.9 ภาษาไทยที่มีใช้อยู่ในปัจจุบันแบ่งเป็นกลุ่มใหญ่ๆ ได้ 2 กลุ่ม คือ รหัสภาษาไทยของเกษตร และรหัส สมอ. แต่ในงานการวิจัยนี้ได้เลือกใช้เฉพาะรหัส สมอ. เนื่องจากเป็นรหัสมาตรฐานของสมาคมมาตรฐานอุตสาหกรรม

```
/* Map keyboard for So-Mo-Oo code */

unsigned char key_smo[127] = {
        0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,
        160, 44, 46, 50, 51, 52, 38,167,
        54, 55, 53, 57,193,162,227,189,
        168,197, 47, 45,192,182,216,214,
        164,181,171,199,178,170,204, 63,
        49,196, 46,169,175,174,226,172,
        231,179,235,201,200, 63,236,207,
        173, 48,177,166,184,234,206, 34,
        41,237, 40,186, 96,197,217, 56,
        37,191,212,225,161,211,180,224,
        233,195,232,210,202,183,215,185,
        194,230,190,203,208,213,205,228,
        187,209,188,176,126, 44, 45
};
```

รูปที่ 3.9 แสดงตารางรหัสแป้นพิมพ์

3.4 การพัฒนาและการจัดการระบบเสียง

3.4.1 อุปกรณ์ในการทำให้เกิดเสียง

การพัฒนาระบบที่ต้องการใช้เสียงเข้ามาเป็นส่วนประกอบ เพื่อทำให้ระบบมีความสมบูรณ์มากขึ้น ต้องใช้อุปกรณ์ทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์เข้าช่วย อุปกรณ์ที่เป็นฮาร์ดแวร์ประกอบด้วย แผงวงจรเสียง (sound card) ลำโพง และไมโครโฟน ส่วนอุปกรณ์ทางด้านซอฟต์แวร์จะเป็นโปรแกรมช่วยงานหรือที่เรียกว่า "ไดร์เวอร์" ("Driver") ซึ่งไดร์เวอร์นี้จะมีความเกี่ยวพันกับแผงวงจรเสียง แผงวงจรเสียงยี่ห้อหนึ่งก็จะมีไดร์เวอร์ตัวหนึ่ง ในปัจจุบันมีหลายบริษัทที่ทำการผลิตแผงวงจรประเภทนี้ แต่การวิจัยนี้ได้เลือกบริษัท Creative เพราะเป็นบริษัทที่มีชื่อเสียงและมีผลิตภัณฑ์จำหน่ายมากมายในท้องตลาด ซึ่งผลิตแผงวงจรที่ใช้ชื่อว่า Sound Blaster และมีไดร์เวอร์ชื่อ ct-voice.drv ทั้งสองตัวนี้จะต้องทำงานร่วมกัน

3.4.2 ขบวนการทำให้เกิดเสียง

นอกจากการทำงานร่วมกันระหว่างอุปกรณ์ทางด้านฮาร์ดแวร์และซอฟต์แวร์ที่ทำให้เกิดเสียงตามที่ได้กล่าวมาแล้ว ยังต้องใช้ข้อมูลที่จะทำให้เกิดเสียง ข้อมูลดังกล่าวสามารถเก็บโทนเสียงและระดับเสียงที่แตกต่างกันได้ การเก็บข้อมูลอยู่ในเครื่องคอมพิวเตอร์จะถูกเก็บแบบดิจิตอล (Digital) และเพื่อให้สามารถแสดงออกมาเป็นเสียงได้ จะต้องอาศัยอุปกรณ์ทำการแปลงข้อมูลนั้นให้เป็นสัญญาณเสียง หรือสัญญาณอนาล็อก (Analog) ผ่านออกมาทางลำโพง เราเรียกขบวนการนี้ว่า D/A (Digital to Analog) และในทางกลับกันถ้าต้องการบันทึกเสียงลงบนแฟ้มข้อมูลก็ทำได้โดยอาศัยการนำเสียงผ่านไมโครโฟน และอุปกรณ์แปลงสัญญาณเสียงเป็นข้อมูลดิจิตอลเก็บลงแฟ้ม เรียกขบวนการนี้ว่า A/D (Analog to Digital)

3.4.3 โครงสร้างแฟ้มข้อมูลเสียง

โครงสร้างของแฟ้มข้อมูลเสียงปัจจุบันมีหลายชนิด และที่ใช้ได้กับ Sound Blaster มีด้วยกัน 4 ชนิด คือ VOC, WAV, TS และ S แต่การวิจัยครั้งนี้ใช้โครงสร้างที่เป็น VOC เท่านั้น โครงสร้างแฟ้มข้อมูลชนิดนี้ถูกแบ่งออกเป็น 2 ส่วน คือ ส่วนหัว (Head Block) และส่วนข้อมูลเสียง (Data Block)

1) ข้อมูลส่วนหัว (Head Block) เป็นส่วนที่จะบอกให้ทราบว่าแฟ้มข้อมูลนี้เป็นแฟ้มข้อมูลประเภทเสียงชนิด VOC มีขนาด 26 ไบต์ มีรายละเอียดดังนี้

(1) ไบต์ที่ 0-19 เก็บรายละเอียดของแฟ้มหรือชื่อประเภทของแฟ้ม โดยเก็บข้อความว่า "Creative Voice File" และปิดท้ายด้วย $1A_{16}$

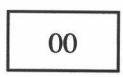
(2) ไบต์ที่ 20-21 เก็บแอ็ดเดรสของข้อมูลเริ่มต้นของเนื้อข้อมูลเสียงโดยใช้ 2 ไบต์นี้เก็บค่า 26 หรือ $001A_{16}$ แต่การเก็บจริงบนแฟ้มจะเก็บเรียงลำดับจากไบต์หลังมาก่อนไบต์หน้าเสมอเฉพาะค่าที่มีความหมายเป็นตัวเลขทางคณิตศาสตร์ จึงเก็บ $1A_{16}$ ตรงไบต์ที่ 20 และเก็บ 00_{16} ตรงไบต์ที่ 21

(3) ไบต์ที่ 22-23 เก็บเลขเวอร์ชั่น ถ้าเป็นเวอร์ชั่น 1.10 จะเก็บค่า $010A_{16}$ และถ้าเป็นเวอร์ชั่น 1.2 จะเก็บค่า 0114_{16} แต่การเก็บข้อมูลจริงจะเก็บเรียงลำดับไบต์หลังก่อนมาไบต์หน้า จากที่กล่าวมาแล้วข้างต้น

(4) ไบต์ที่ 24-25 เก็บข้อมูลเพื่อบ่งชี้ว่าแฟ้มข้อมูลนี้เป็น VOC ค่าที่ได้ ถ้าเป็นเวอร์ชั่น 1.10 จะมีค่า 1129_{16} หรือถ้าเป็นเวอร์ชั่น 1.2 จะมีค่า $111F_{16}$ การเก็บเหมือนกับไบต์ที่ 22-23

2) ส่วนข้อมูลเสียง (Data Block) คือ ข้อมูลที่มีส่วนของการแยกย่อยออกไปอีก (Sub Block) แต่ละส่วนย่อยจะมีรูปแบบและหน้าที่ที่แตกต่างกัน โดยสามารถแยกส่วนย่อยๆ ออกได้เป็น 8 ชนิด ซึ่งส่วนหัวของแต่ละชนิดจะทำหน้าที่บอกฟังก์ชั่นภายในส่วนเอง บางชนิดก็จะมีส่วนของการบอกความยาวเท่าไร และข้อมูลทางด้านเสียง ดังมีรายละเอียดของแต่ละชนิดดังนี้

(1) ชนิดที่ 0 มีขนาด 1 ไบต์ เรียกว่า Terminator Block ซึ่งจะอยู่ท้ายแฟ้มเสมอ ทำหน้าที่บอกการสิ้นสุดข้อมูลเสียง เก็บค่า 00_{16} ดังแสดงในรูปที่ 3.10



1 byte

รูปที่ 3.10 รูปแบบส่วนย่อยชนิดที่ 0

(2) ชนิดที่ 1 เรียกว่า Voice Data Block เป็นส่วนที่เก็บเนื้อข้อมูลเสียง มีขนาดไม่เกิน $0FFFFFE_{16}$ ไบต์ โดยไบต์แรกเก็บค่า 01_{16} และสามไบต์ถัดมาเรียก

ว่า Blk_len ทำหน้าที่เก็บความยาวของข้อมูล และหนึ่งไบต์ถัดมาเรียกว่า TC หรือ Time_Constant ใช้เก็บอัตราการสุ่มสัญญาณอนาล็อกเพื่อเปลี่ยนเป็นสัญญาณดิจิตอล (sampling rate) โดยมีสูตรคำนวณค่า TC คือ

$$TC = 256 - (1{,}000{,}000 / \text{sampling rate})$$

และหนึ่งไบต์ถัดมาเรียกว่า PM เก็บค่าเพื่อบอกถึงวิธีการอัดข้อมูล (Pack Method) โดยมีค่าต่างๆ ดังนี้

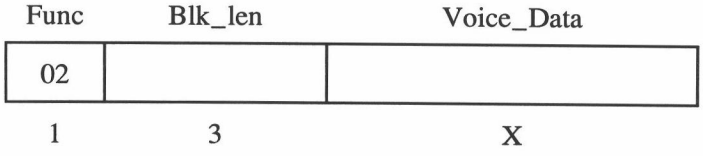
- 0 หมายถึง การเก็บข้อมูลโดยไม่ต้องทำการอัดข้อมูล
- 1 หมายถึง มีการอัดข้อมูลก่อนเก็บด้วยอัตรา 2 : 1 เช่น ข้อมูล 8 บิตจะถูกอัดเหลือ 4 บิต
- 2 หมายถึง มีการอัดข้อมูลก่อนเก็บด้วยอัตรา 3 : 1 เช่น ข้อมูล 8 บิตจะถูกอัดเหลือ 2.6 บิต
- 3 หมายถึง มีการอัดข้อมูลก่อนเก็บด้วยอัตรา 4 : 1 เช่น ข้อมูล 8 บิตจะถูกอัดเหลือ 2 บิต

ส่วนไบต์ถัดไปเรียกว่า Voice_Data คือส่วนของการเก็บข้อมูลเสียงจริงๆ ดังแสดงในรูปที่ 3.11

Func	Blk_len	TC	PM	Voice_Data
02				
1	3	1	1	X

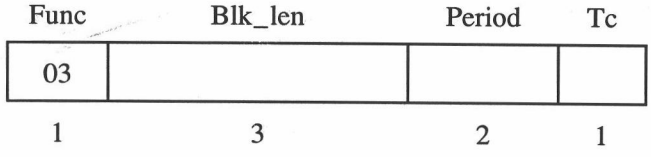
รูปที่ 3.11 รูปแบบส่วนย่อยชนิดที่ 1

(3) ชนิดที่ 2 เรียกว่า Voice Continuation Block เป็นที่เก็บเนื้อข้อมูลเสียงที่ถูกแบ่งเป็นส่วนย่อยๆ เนื่องจากขนาดข้อมูลนั้นใหญ่เกินขนาดของหน่วยความจำมีขนาดไม่เกิน $0FFFFFE_{16}$ ไบต์ โดยไบต์แรกเก็บค่า 02_{16} และสามไบต์ถัดมาเรียกว่า Blk_len ทำหน้าที่เก็บความยาวของข้อมูล ดังแสดงในรูปที่ 3.12



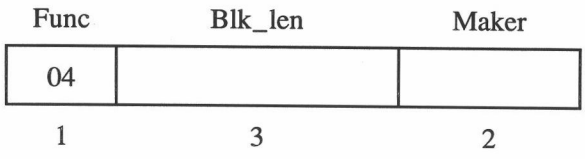
รูปที่ 3.12 รูปแบบส่วนย่อยชนิดที่ 2

(4) ชนิดที่ 3 เรียกว่า Silence Block เป็นที่เก็บข้อมูลเสียงเงียบหรือข้อมูลที่มีค่า 0 มีขนาด 7 ไบต์ โดยไบต์แรกเก็บค่า 03_{16} และสามไบต์ถัดมาเรียกว่า Blk_len เก็บค่า 3 และสองไบต์ถัดมาเรียกว่า Period เก็บจำนวนข้อมูลเสียงเงียบหรือจำนวนของค่า 0 และไบต์สุดท้ายเรียกว่า TC เก็บค่าอัตราการสุ่มสัญญาณอนาล็อกเพื่อเปลี่ยนเป็นสัญญาณดิจิตอล ดังแสดงในรูปที่ 3.13



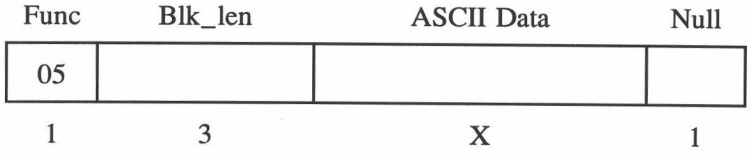
รูปที่ 3.13 รูปแบบส่วนย่อยชนิดที่ 3

(5) ชนิดที่ 4 เรียกว่า Marker Block เป็นส่วนที่ใช้กำหนดจุดเริ่มให้ฟังก์ชั่นในไดร์เวอร์เสียงให้เริ่มขับเสียง มีขนาด 5 ไบต์ โดยไบต์แรกเก็บค่า 04_{16} และสามไบต์ถัดมาเรียกว่า Blk_len เก็บค่า 2 และสองไบต์ถัดมาเรียกว่า Marker เก็บค่าได้ตั้งแต่ 1 ถึง $FFFE_{16}$ โดยไดร์เวอร์เสียงจะอ่านค่านี้ไปเก็บไว้ในตัวแปร ct_voice_status ในหน่วยความจำ ดังแสดงในรูปที่ 3.14



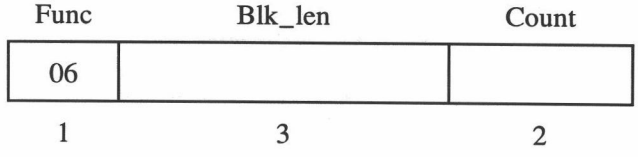
รูปที่ 3.14 รูปแบบส่วนย่อยชนิดที่ 4

(6) ชนิดที่ 5 เรียกว่า ASCII Text Block เป็นที่เก็บข้อความเพื่อแสดงบนจอภาพเป็นฟอร์กราวน์ประกอบกับการขับเสียงที่เป็นแบ็คกราวน์ มีขนาดไม่เกิน $FFFFFE_{16}$ โดยไบต์แรกเก็บค่า 05_{16} และสามไบต์ถัดมาเรียกว่า Blk_len เก็บความยาวของข้อความ ASCII Data โดยไบต์สุดท้ายเรียกว่า Null เก็บค่า 00_{16} ดังแสดงในรูปที่ 3.15



รูปที่ 3.15 รูปแบบส่วนย่อยชนิดที่ 5

(7) ชนิดที่ 6 เรียกว่า Repeat Loop Block เป็นส่วนที่บอกการเริ่มต้นของการวนซ้ำของการขับเสียง ซึ่งส่วนที่อยู่ถัดไปจะเป็นส่วนของข้อมูลเสียงที่ต้องการขับเสียงแบบวนซ้ำ มีขนาด 6 ไบต์ โดยไบต์แรกเก็บค่า 06_{16} และสามไบต์ถัดมาเรียกว่า Blk_len เก็บค่า 2 โดยสองไบต์สุดท้ายเรียกว่า Count เก็บค่าจำนวนที่ต้องการวนซ้ำ ซึ่งมีค่าได้ตั้งแต่ 1 ถึง $0FFFE_{16}$ ดังแสดงในรูปที่ 3.16



รูปที่ 3.16 รูปแบบส่วนย่อยชนิดที่ 6

(8) ชนิดที่ 7 เรียกว่า End Repeat Loop Block เป็นส่วนที่บอกการสิ้นสุดของการวนซ้ำการขับเสียง มีขนาด 4 ไบต์ โดยไบต์แรกเก็บค่า 07_{16} และสามไบต์ถัดมาเรียกว่า Blk_len เก็บค่า 0 ดังแสดงในรูปที่ 3.17

รูปที่ 3.17 รูปแบบส่วนย่อยชนิดที่ 7

3.4.4 คุณสมบัติทั่วไปของระบบเสียง

จากความสามารถของไดร์เวอร์เสียงทำงานร่วมกับฮาร์ดแวร์เสียง เพื่อทำหน้าที่จัดการกับข้อมูลที่มีโครงสร้างของเสียง สามารถสรุปความสามารถต่างๆ ของระบบเสียงได้ดังนี้

1) สามารถกำหนดให้มี อัตราการสุ่มสัญญาณเพื่อเปลี่ยนสัญญาณอนาล็อกให้เป็นดิจิตอลได้หลายอัตราในแฟ้มข้อมูลเดียวกัน

2) สามารถใช้เทคนิคการอัดข้อมูล (Compressed Data) เพื่อทำการลดข้อมูลต่างๆ รวมทั้งการอัดข้อมูลเสียงเงียบ (Packing Silence) ได้ในแฟ้มเดียวกัน

3) สามารถเก็บข้อมูลเสียงได้ทั้งในระบบเสียงแบบสเตอริโอและแบบโมโนได้ในแฟ้มเดียวกัน

4) สามารถเก็บข้อความ เพื่อใช้ในการแสดงบนจอภาพเป็นลักษณะฟอร์กราวน์และการขับเสียงเป็นแบ็คกราวน์ได้

5) สามารถทำการขับเสียงให้เสียงในช่วงใดช่วงหนึ่งเล่นซ้ำกันได้ และได้หลายรอบ