# บทที่ 2

## โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1

### ข้อมูลทั่วไป

โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 (Adobe Systems Incorporated, 1990) คือโปรแกรมภาษาโพสต์สคริปต์ประเภทหนึ่งที่ประกอบด้วยคำสั่งพิเศษที่บรรยายเกี่ยวกับลักษณะโครงร่างแบบอักษร โดยคำสั่งดังกล่าวจะเป็นอิสระต่ออุปกรณ์การแสดงผลต่างๆ คำสั่งของรูปแบบอักษรโพสต์สคริปต์ประเภทที่ 1 นี้ จะเป็นเพียงส่วนหนึ่งของภาษาโพสต์สคริปต์ทั้งหมด และจะมีคำสั่งพิเศษเพิ่มเติมขึ้นมาเพื่อให้มีความสามารถเพิ่มขึ้น โดยเฉพาะอย่างยิ่งในการบรรยายการสร้างแบบอักษรที่อยู่ในรูปของข้อความที่มีขนาดเล็กมาก ๆ

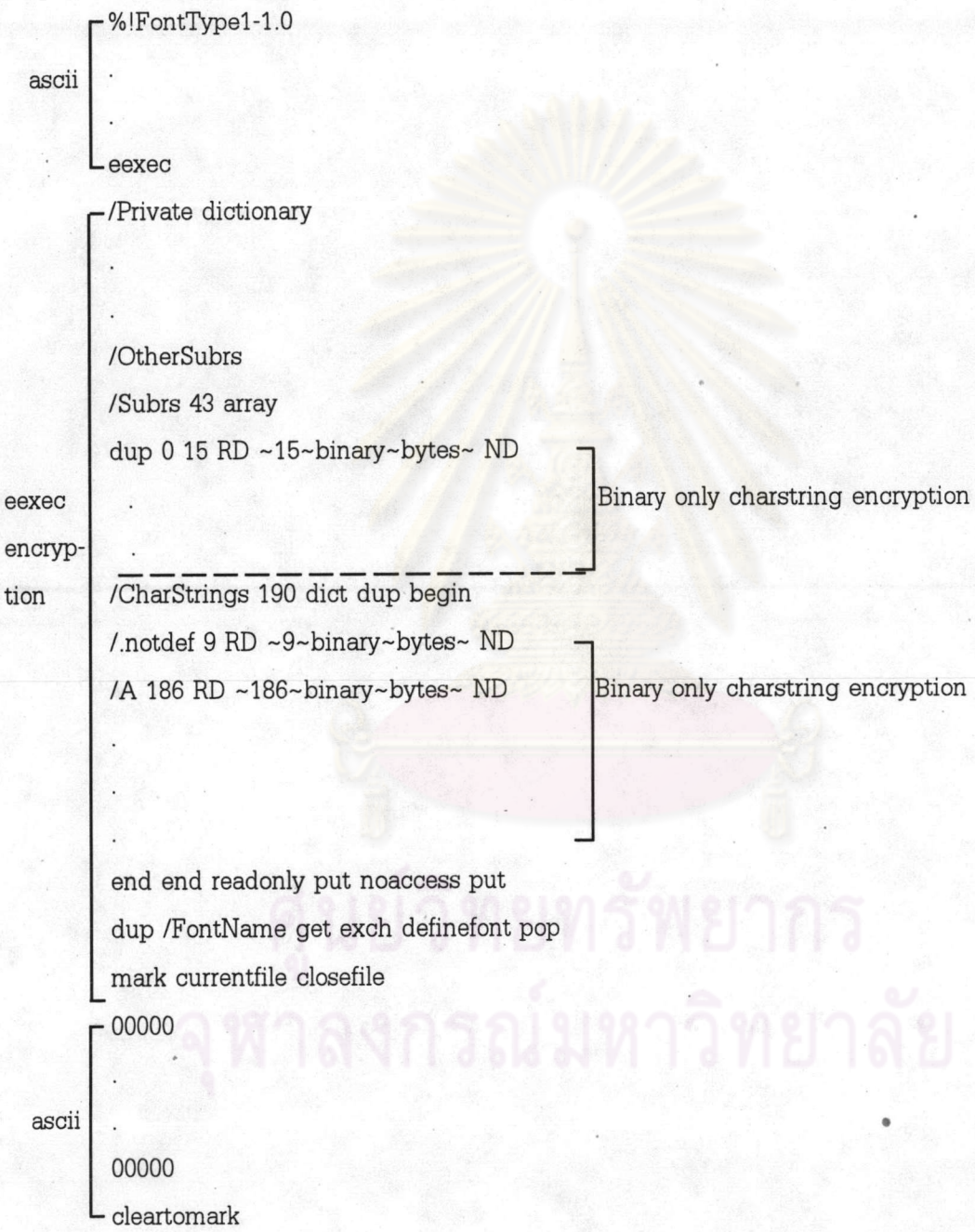
จุดเด่นที่สำคัญและถือได้ว่าเป็นความแตกต่างจากโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 3 ก็คือความสามารถในการแสดงผลตัวอักษรต่างๆได้อย่างถูกต้อง โดยเฉพาะตัวอักษรที่มีขนาดเล็กและแสดงผลบนอุปกรณ์ที่มีความละเอียดต่ำเช่น จอภาพ กล่าวคือจะมีส่วนที่เรียกว่าฮินต์(hint) ซึ่งเป็นความสามารถพิเศษเป็นเครื่องช่วยในการแสดงผล โดยตัวแปลภาษาโพสต์สคริปต์จะใช้ขั้นตอนการทำงานพิเศษสำหรับโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 นอกจากนี้ยังมีความสามารถในการกำหนดโปรแกรมย่อยเพื่อใช้เก็บคำสั่งที่บรรยายลักษณะของแบบอักษรที่ซ้ำ ๆ กันไว้ ทำให้โปรแกรมแบบอักษรมีขนาดเล็กและมีความเหมาะสมในการพัฒนาโปรแกรมแบบอักษรเพื่อใช้บรรยายการสร้างแบบอักษรภาษาไทย เพราะว่าลักษณะของแบบอักษรภาษาไทยนั้น จะมีลักษณะของอักษรบางส่วนที่คล้ายคลึงกัน เช่น ลักษณะหัวอักษรหงาย อาทิ ต ด ลักษณะหัวอักษรคว่ำบน อาทิ น พ บ ม ลักษณะหัวอักษรหงายล่าง อาทิ ถ ฤ ญ ณ ฌ

องค์ประกอบของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 นั้น ประกอบด้วยส่วนสำคัญ 3 ส่วน(ดูรูปที่ 2.1) คือ

1. ส่วนที่เป็นข้อความแอสกี ซึ่งจะเป็นส่วนบรรยายเกี่ยวกับลักษณะของโปรแกรมแบบอักษรนั้นๆ เช่น รุ่นไทป์เฟซ แฟมิลี่ น้ำหนักของเส้นของแบบอักษร ความเอียงของตัวอักษร

2. ส่วนที่เป็นข้อความเข้ารหัสลับ ซึ่งจะเป็นส่วนที่บรรยายเกี่ยวกับลักษณะของแบบอักษร โปรแกรมย่อยของแบบอักษรและฮินต์พารามิเตอร์

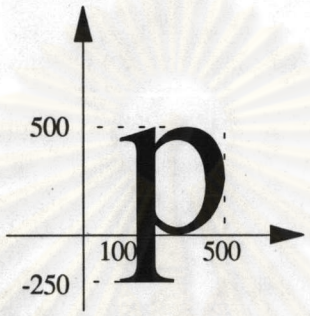
3. ส่วนที่เป็นข้อความแอสกี ซึ่งเป็นส่วนท้ายสุดของโปรแกรมแบบอักษร ใช้เป็นการภายในสำหรับตัวแปลคำสั่งโพสต์สคริปต์




รูปที่ 2.1 แสดงองค์ประกอบของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1

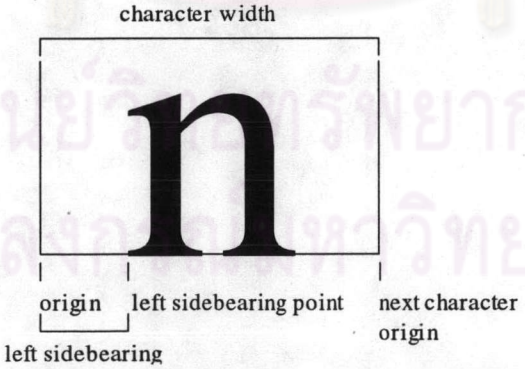
ลักษณะของแบบอักษร

การกำหนดแบบอักษรของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 จะถูกกำหนดในคาเร็คเตอร์สเปซยูนิต (character space unit) (ดูรูปที่ 2.2) ซึ่งเป็นระบบคู่ลำดับที่ใช้ในภาษาโพสต์สคริปต์ ตัวอักษรแต่ละตัวจะถูกสร้างขึ้นได้ภายในขอบเขตของคาเร็คเตอร์สเปซยูนิตนี้เท่านั้น โดยค่าขอบเขตจะอยู่ระหว่าง +2000 ถึง -2000 หน่วยทั้งในแนวแกน x และ y โดยค่าของ x จะเพิ่มขึ้นทางด้านขวา และค่าของ y จะเพิ่มขึ้นทางด้านบน




รูปที่ 2.2 แสดงถึงระบบคาเร็คเตอร์สเปซยูนิต

ตัวอักษรที่ถูกสร้างขึ้น(ดูรูปที่ 2.3) จะใช้ระบบการอ้างอิงจากจุดเริ่มต้น(character's origin) โดยจุดแรกที่เป็นส่วนซ้ายสุดของแบบอักษรที่ถูกสร้างจะเรียกว่าจุด left sidebearing ระยะทางตามแนวนอนจากจุดอ้างอิงเริ่มต้นจนถึงจุดดังกล่าวเรียกว่า left sidebearing และขนาดความกว้างของตัวอักษร (character width) คือ ระยะทางจากจุดอ้างอิงเริ่มต้นจนถึงจุดอ้างอิงเริ่มต้นของตัวอักษรถัดไป



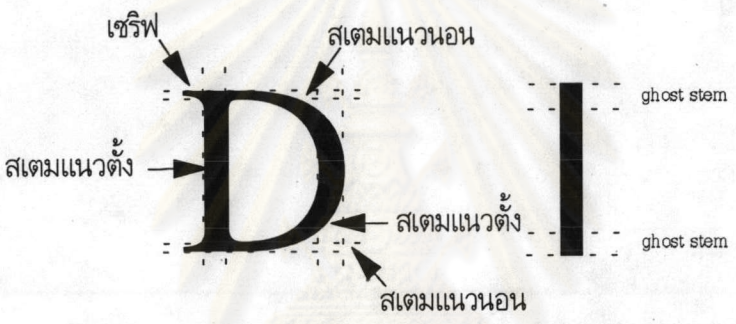

รูปที่ 2.3 แสดงถึงจุดเริ่มต้น left sidebearing และความกว้างของตัวอักษร

ลักษณะเด่นของแบบอักษรที่ถือว่าเป็นหัวใจของแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ประกอบด้วย

1. สเตมแนวตั้ง(vertical stem) เป็นเส้นหลักของตัวอักษรในแนวดิ่ง

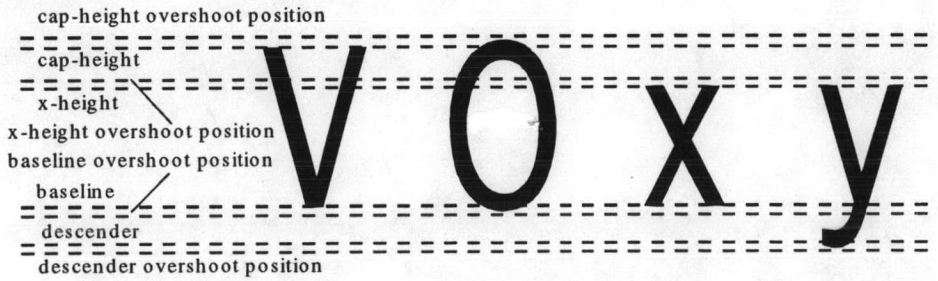
2. สเตมแนวนอน(horizontal stem) เป็นเส้นหลักของตัวอักษรในแนวนอน

สำหรับบางรูปแบบอักษรเช่น แบบอักษรโรมัน จะมีส่วนของตัวอักษรที่เรียกเฉพาะว่า "เซริฟ" หรือตัวอักษรภาษาอังกฤษตัวใหญ่ I ของบางแบบอักษร ซึ่งไม่มีลักษณะของสเตมเด่นชัด จะเรียกสเตมเหล่านี้ว่า "ghost stems" ซึ่งทั้งสองแบบนี้จะถูกพิจารณาในรูปแบบของสเตมแนวนอน (ดูรูปที่ 2.4) ทั้งนี้เพื่อช่วยให้การแสดงผลตัวอักษรที่มีขนาดเล็กให้มีความสวยงาม




รูปที่ 2.4 แสดงเส้นหลักของตัวอักษร

การกำหนดสเตมทั้งแนวตั้งและแนวนอนนี้ จะถูกนำมาใช้พิจารณาร่วมกับการกำหนดอไลเมนต์(alignment) ซึ่งเรียกว่าการทำฮินต์ เพื่อให้การแสดงผลตัวอักษรที่มีขอบบนและล่างที่แบนเรียบ กับตัวอักษรที่มีขอบบนและล่างที่มีความโค้งมน มีความสวยงามสำหรับตัวอักษรที่มีขนาดเล็ก โดยการกำหนดอไลเมนต์นี้จะประกอบด้วยเลขสองจำนวน ขอบเขตของเลขสองจำนวนนี้จะเรียกว่าอไลเมนต์โซน ค่าของเลขจำนวนแรกจะแสดงถึงขอบเขตของตำแหน่งที่แบนราบ(flat position) ในขณะที่ค่าของเลขจำนวนที่สองจะแสดงถึงขอบเขตของตำแหน่งที่มีความโค้งมน (overshoot position) ค่าความแตกต่างระหว่างเลขสองจำนวนนี้จะเรียกว่าอไลเมนต์โซนไฮต์(alignment zone height) ซึ่งค่าปรกติจะอยู่ในช่วง 10 - 20 หน่วย




รูปที่ 2.5 แสดงอไลเมนต์โซนระดับต่างๆ

อไลเมนต์โซนนี้จะมีชื่อเรียกแตกต่างกันไปตามตำแหน่งในคาเร็คเตอร์โคออร์ดิเนตสเปซดังนี้ (ดูรูปที่ 2.5)

1. เบสไลน์(base line) คือตำแหน่งในทางแกน y ที่เป็นฐานของตัวอักษรที่มีส่วนฐานแบนราบ

2. เบสไลน์โอเวอร์ชูทโพซิชัน(base line overshoot position) คือตำแหน่งในทางแกน y ที่อยู่ใต้เส้นเบสไลน์เป็นฐานของตัวอักษรที่มีส่วนฐานโค้งมน

3. แคปไฮต์(cap-height) คือตำแหน่งในทางแกน y ที่เป็นจุดสูงสุดของตัวอักษรที่มีส่วนสูงและส่วนบนแบนราบ(ตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็กที่มีหางบน เช่น d)

4. แคปไฮต์โอเวอร์ชูทโพซิชัน(cap-height overshoot position) คือตำแหน่งในทางแกน y ที่เป็นจุดสูงสุดของตัวอักษรที่มีส่วนสูงและส่วนบนโค้งมน(ตัวพิมพ์ใหญ่)

5. เอ็กซ์ไฮต์(x-height) คือตำแหน่งในทางแกน y ที่เป็นจุดสูงสุดของตัวอักษร(ตัวพิมพ์เล็ก) ที่มีส่วนบนแบนราบ

6. เอ็กซ์ไฮต์โอเวอร์ชูทโพซิชัน(x-height overshoot position) คือตำแหน่งในทางแกน y ที่เป็นจุดสูงสุดของตัวอักษรที่มีส่วนบนโค้งมน(ตัวพิมพ์เล็ก)

7. เดสเซ็นเดอร์(descender) คือตำแหน่งในทางแกน y ที่เป็นจุดต่ำสุดของตัวอักษรที่มีส่วนล่างแบนราบ (ตัวพิมพ์เล็กที่มีหางล่าง เช่น p)

8. เดสเซ็นเดอร์โอเวอร์ชูทโพซิชัน(descender overshoot position) คือตำแหน่งในทางแกน y ที่เป็นจุดต่ำสุดของตัวอักษรที่มีส่วนล่างโค้งมน(ตัวพิมพ์เล็กที่มีหางล่าง เช่น g)

นอกจากสเตมที่เป็นเส้นหลักแล้ว ลักษณะพิเศษของตัวอักษรที่สำคัญอีกส่วนได้แก่ เฟลกซ์*(flex) ซึ่งคือส่วนของเส้นโค้งของตัวอักษรที่มีความโค้งน้อยมากๆจนแทบจะเป็นเส้นตรงทั้งแนวตั้งหรือแนวนอน โดยเส้นโค้งที่จะแทนด้วยเฟลกซ์ได้นั้น จะต้องมีคุณสมบัติดังนี้(ดูรูปที่ 2.6)

---

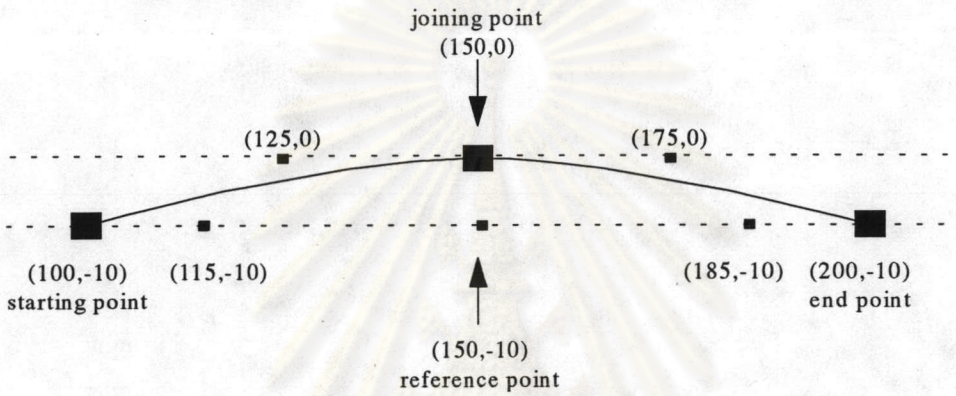
*ดูรายละเอียดการกำหนดการใช้เฟลกซ์เพิ่มเติมที่หัวข้อโปรแกรมย่อยของแบบอักษร, หน้า 25.

3.1 เป็นเส้นโค้งที่ประกอบด้วยเส้นโค้งเบเซีย(Bezier) 2 เส้น

3.2 จุดปลายของทั้ง 2 เส้น จะต้องอยู่ในแนวเดียวกัน

3.3 จุดที่เชื่อมต่อและจุดควบคุมความโค้งที่ใกล้กับจุดเชื่อมต่อของเส้นทั้ง 2 จะต้องอยู่ในแนวเดียวกัน

3.4 ความสูงของส่วนโค้งจะต้องน้อยกว่าหรือเท่ากับ 20 หน่วย




รูปที่ 2.6 แสดงเส้นโค้งที่มีคุณสมบัติของเฟลกซ์

กลไกการทำงานของเฟลกซ์นี้ จะเป็นการพิจารณาว่าในการแสดงผลควรจะเลือกใช้เส้นโค้งหรือควรจะแทนด้วยเส้นตรง โดยการคำนวณความสูงของเฟลกซ์ในหน่วยของอุปกรณ์แสดงผลว่าการแสดงผลตัวอักษรในขณะนั้นค่าความสูงของเฟลกซ์มีค่าน้อยหรือมากกว่าค่าพารามิเตอร์ความสูงของเฟลกซ์ที่กำหนดไว้อย่างไร ถ้ามีค่าน้อยกว่าก็จะพิจารณาเป็นเส้นตรง แต่ถ้ามากกว่าก็จะพิจารณาเป็นเส้นโค้ง



ข้อจำกัดของจำนวนเส้น

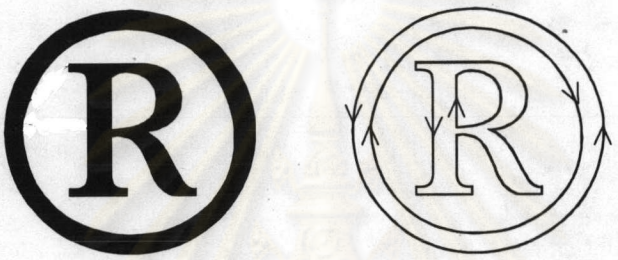
ตัวอักษรโพสต์สคริปต์ประเภทที่ 1 ประกอบขึ้นจากเส้นตรงและเส้นโค้งซึ่งโดยทั่วไปจำนวนเส้นเหล่านี้จะมีได้ไม่จำกัด ยกเว้นแต่ในบางรุ่นของตัวแปลคำสั่งโพสต์สคริปต์ได้กำหนดจำนวนเส้นสูงสุดของแต่ละตัวอักษรไว้ไม่เกิน 1,500 เส้น ทำให้หากตัวอักษรใดมีจำนวนเส้นเกินขีดจำกัดนี้ จะเกิดความผิดพลาดขึ้นได้ ดังนั้นผู้สร้างโปรแกรมแบบอักษรจึงจำเป็นที่จะต้องคำนึงถึงข้อจำกัดนี้ และควรที่จะทำการทดสอบตัวอักษรที่มีความสลับซับซ้อนมากๆกับตัวแปลคำสั่งรุ่นต่างๆรวมทั้งซอฟต์แวร์เอทีเอ็มด้วย

ทิศทางของเส้นที่บรรยายการสร้างตัวอักษร

การบรรยายการสร้างตัวอักษรโพสต์สคริปต์ประเภทที่ 1 นั้น จะต้องคำนึงถึงทิศทางของเส้นโดยยึดหลักดังนี้(ดูรูปที่ 2.7)

1. ทิศทางการบรรยายการสร้างตัวอักษรที่เป็นขอบเขตด้านนอก จะต้องมีทิศทาง"ทวนเข็มนาฬิกา"เสมอ

2. ทิศทางการบรรยายการสร้างตัวอักษรที่เป็นขอบเขตด้านใน จะต้องมีทิศทาง"ตามเข็มนาฬิกา"เสมอ



รูปที่ 2.7 แสดงทิศทางของเส้นและการระบาย

ให้สังเกตว่าไม่ควรให้มีการตัดกันของเส้นตรงหรือเส้นโค้งของตัวอักษร เพราะโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 สามารถที่จะแสดงผลแบบตัวอักษรโปร่งได้ และจะทำให้เห็นส่วนที่ตัดกันในการแสดงผลซึ่งอาจไม่ตรงกับแบบตัวอักษรโปร่งที่ต้องการ

โครงสร้างของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1

โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ประกอบด้วย

1. หมายเหตุ (comment)

ใช้สำหรับการอธิบายรายละเอียดต่างๆของโปรแกรมเช่น วันที่สร้าง ชื่อแบบอักษร ผู้สร้าง โดยหมายเหตุ 3 บรรทัดแรกจะต้องเป็นไปตามรูปแบบดังนี้

%!FontType1-*version*: *Fontname Fontversion*

%%CreationDate: *date*

%%VMUsage: n1 n2

โดย *version* เป็นรุ่นของรูปแบบอักษรโพสต์สคริปต์ประเภทที่ 1

*Fontname* เป็นชื่อแบบอักษร

*Fontversion* เป็นรุ่นของโปรแกรมแบบอักษร

*date* เป็นวันเดือนปีที่สร้างแบบอักษร

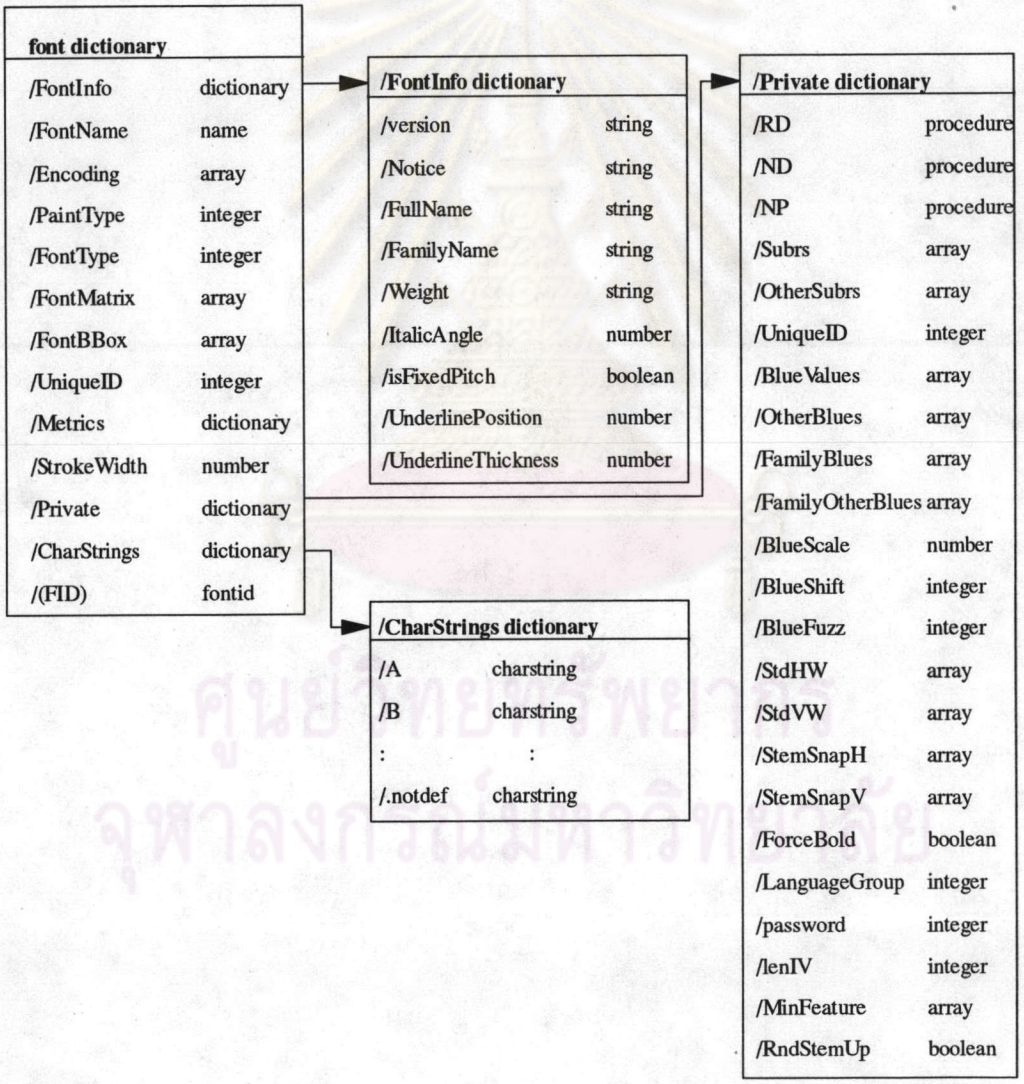
n1 เป็นค่าของขนาดหน่วยความจำเสมือนที่น้อยที่สุดที่ต้องใช้ในการเก็บแบบอักษร

n2 เป็นค่าของขนาดหน่วยความจำเสมือนที่มากที่สุดที่ต้องใช้ในการเก็บแบบอักษร

ค่าของขนาดหน่วยความจำเสมือนนี้ จะถูกใช้โดยโปรแกรมประยุกต์เพื่อใช้ในการเก็บแบบอักษรลงบนตัวแปลคำสั่งโพสต์สคริปต์

ส่วนหมายเหตุหลังจากนี้จะขึ้นต้นด้วย % แล้วตามด้วยข้อความ

2. พจนานุกรมแบบอักษร เป็นส่วนสำคัญที่สุดของโปรแกรมแบบอักษร ประกอบไปด้วยส่วนต่างๆ(ดูรูปที่ 2.8)ดังต่อไปนี้




รูปที่ 2.8 แสดงโครงสร้างของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1

2.1 FontInfo dictionary ประกอบด้วย

2.1.1 version คือรุ่นของแบบอักษร

2.1.2 Notice คือข้อความเกี่ยวกับลิขสิทธ์ต่างๆ

2.1.3 FullName คือชื่อเต็มของแบบอักษร

2.1.4 FamilyName คือชื่อของแฟมิลี่ของแบบอักษร

2.1.5 Weight คือขนาดความหนาของแบบอักษร มีค่าได้ดังนี้

100 Thin

200 Extra light

300 Light

400 Normal

500 Medium

600 Semi-bold

700 Bold

800 Extra blod

900 Heavy

2.1.6 ItalicAngle คือความลาดเอียงของแบบอักษร

2.1.7 isFixedPitch คือการกำหนดให้แบบอักษรมีขนาดความกว้างคงที่หรือไม่

2.1.8 UnderlinePosition คือตำแหน่งของเส้นใต้นับจากเส้นเบสไลน์

2.1.9 UnderlineThickness คือขนาดความหนาของเส้นใต้

2.2 FontName คือชื่อของแบบอักษร

2.3 Encoding คือตัวแปรแถวลำดับของรหัสและชื่อของตัวอักษร ในกรณีที่เป็นรหัสและชื่ออักษรมาตรฐานของบริษัทอโดบี จะต้องกำหนดว่า

/Encoding StandardEncoding

2.4 PaintType คือค่าที่กำหนดเพื่อให้ตัวแปลคำสั่งโพสต์สคริปต์แสดงผลตัวอักษรแบบทึบ (PaintType = 0) หรือแบบโปร่ง(PaintType = 2)

2.5 FontType คือค่าที่แสดงว่าเป็นแบบอักษรโพสต์สคริปต์ประเภทใด(ต้องเป็น 1 เสมอสำหรับแบบอักษรประเภทที่ 1)

2.6 FontMatrix คือค่าที่ใช้ในการปรับขนาดจากคาเร็คเตอร์สเปซยูนิตไปเป็นยูเซอร์สเปซยูนิต(user space unit) ซึ่งเป็นระบบคู่ลำดับสำหรับโปรแกรมภาษาโพสต์สคริปต์สำหรับการจัดวางวัตถุบนหน้ากระดาษ โดยทั่วไปมีค่าเท่ากับ 1000 ต่อ 1

2.7 FontBBox คือตัวแปรแถวลำดับที่เก็บขอบเขตของตัวอักษรในโปรแกรมแบบอักษร

2.8 UniqueID คือตัวเลขที่กำหนดขึ้นสำหรับโปรแกรมแบบอักษรหนึ่งๆแต่ละแบบมีค่าต่างกัน ค่านี้มีความสำคัญมากในกรณีที่มีการทำฟอนต์แคชชิ่ง เพราะว่าถ้าโปรแกรมแบบอักษรมีค่านี้เหมือนๆกัน การแสดงผลของโปรแกรมประยุกต์ต่างๆจะเกิดความผิดพลาด ค่าที่เป็นไปได้ของ UniqueID จะอยู่ระหว่าง 0 ถึง 16,777,215

บริษัทอโดบีได้ตั้งข้อกำหนดสำหรับการกำหนด UniqueID ไว้ 2 ประการคือ

2.8.1 ค่าระหว่าง 4,000,000 ถึง 4,999,999 เป็นค่าที่ใช้สำหรับหน่วยงานที่ออกแบบและใช้โปรแกรมแบบอักษรภายในหน่วยงานนั้นๆ

2.8.2 กรณีที่ผู้สร้างโปรแกรมแบบอักษรต้องการเผยแพร่ จะต้องทำการขอตัวเลข UniqueID จากบริษัท

2.9 StrokeWidth คือค่าความกว้างของเส้นขอบตัวอักษรในกรณีที่แสดงผลแบบโปร่ง

2.10 Private dictionary เป็นส่วนที่ใช้เก็บข้อมูลเกี่ยวกับอไลเมนต์โซนต่างๆ รวมทั้งส่วนที่กำหนดโปรแกรมย่อยของแบบอักษร ประกอบด้วย

2.10.1 RD, ND และ NP เป็นการกำหนดคำย่อที่ใช้แทนกระบวนคำสั่งภาษาโพสต์สคริปต์เพื่อให้โปรแกรมแบบอักษรมีขนาดเล็ก กระบวนคำสั่งเหล่านี้จะถูกใช้โดยตัวแปลคำสั่งโพสต์สคริปต์

2.10.2 Subrs เป็นส่วนที่ใช้กำหนดโปรแกรมย่อยของแบบอักษร

2.10.3 OtherSubrs เป็นส่วนที่ใช้สำหรับการทำการเปลี่ยนแปลงสเตมและการใช้เฟลกซ์

2.10.4 UniqueID เหมือนกับส่วน UniqueID ที่กล่าวไปแล้ว

2.10.5 BlueValues คือตัวแปรแถวลำดับที่ประกอบด้วยคู่ของเลขจำนวนเต็มที่ใช้กำหนดอไลเมนต์โซนในส่วนที่เกินเบสไลน์ขึ้นไป โดยมีกฎเกณฑ์ดังนี้

2.10.5.1 จำนวนเต็มค่าแรกของแต่ละคู่นั้น จะต้องมีค่าน้อยกว่าจำนวนเต็มค่าหลัง

2.10.5.2 เลขจำนวนเต็มคู่แรกจะต้องเป็นค่าของเบสไลน์โอเวอร์ชูทและเบสไลน์

2.10.5.3 เลขจำนวนเต็มคู่ถัดไปจะต้องเป็นค่าอไลเมนต์โซนในส่วนที่เกินเบสไลน์ขึ้นไป

2.10.5.4 จำนวนคู่สูงสุดคือ 7 คู่

2.10.5.5 ค่าความแตกต่างระหว่างคู่จะต้องมีค่าอย่างต่ำไม่น้อยกว่า 3

2.10.6 OtherBlues คือตัวแปรแถวลำดับที่ประกอบด้วยคู่ของเลขจำนวนเต็มที่ใช้กำหนดอไลเมนต์โซนในส่วนที่ต่ำกว่าเบสไลน์ลงมา โดยมีกฎเกณฑ์เหมือนกับ BlueValues ยกเว้นจำนวนคู่สูงสุดจะมีได้ 5 คู่และจะต้องเรียงลำดับจากน้อยไปหามาก

2.10.7 FamilyBlues และ FamilyOtherBlues จะเหมือนกับ BlueValues และ OtherBlues ทุกประการ วัตถุประสงค์ของการกำหนดค่านี้ก็เพื่อใช้ช่วยในการแสดงผลตัวอักษรในกรณีที่มีการผสมกันระหว่างแบบอักษรเดียวกัน แต่ต่างกันที่ style ตัวแปลคำสั่งโพสต์สคริปต์จะเลือกใช้อไลเมนต์โซนที่เหมาะสมเพื่อให้การแสดงผลมีความสวยงาม โดยกำหนดว่าถ้าความแตกต่างระหว่างอไลเมนต์โซนของโปรแกรมแบบอักษรนั้น (Blue และ OtherBlues) กับอไลเมนต์โซนที่กำหนดโดย FamilyBlues และ FamilyOtherBlues มีความแตกต่างกันเกินกว่า 1 จุดภาพแล้ว ตัวแปลคำสั่งโพสต์สคริปต์จะเลือกใช้อไลเมนต์โซนของโปรแกรมแบบอักษรนั้น เช่น สมมติว่าแบบอักษรโรมันขนาด 10 จุด อักษร x ตัวปรกติมีเอ็กซ์ไฮต์เท่ากับ 5.4 จุดภาพ ในขณะที่ตัวหนามีค่าเท่ากับ 5.6 จุดภาพ ก็จะเลือกใช้อไลเมนต์โซนมาตรฐาน(Family) ในทางตรงกันข้าม ถ้าแบบอักษรโรมันขนาด 100 จุด ค่าความแตกต่างจะเท่ากับ 2 จุดภาพ ดังนั้นก็จะเลือกใช้อไลเมนต์โซนของโปรแกรมแบบอักษรนั้น

2.10.8 BlueScales เป็นค่าที่ใช้ในการควบคุมขนาดตัวอักษรที่จะทำโอเวอร์ชูท ซึ่งขนาดตัวอักษรที่ควบคุมนี้จะมีค่าเปลี่ยนแปลงไปตามความละเอียดของอุปกรณ์การแสดงผล หลักเกณฑ์ที่ใช้ในการพิจารณาคือ ถ้าตัวอักษรนั้นมีขนาดน้อยกว่าขนาดของตัวอักษรที่กำหนด BlueScales การแสดงผลตัวอักษรจะไม่มีการทำโอเวอร์ชูท ในทางตรงกันข้ามถ้าขนาดตัวอักษรมีขนาดมากกว่าขนาดของตัวอักษรที่กำหนด BlueScales การแสดงผลตัวอักษรจะทำโอเวอร์ชูท(ขนาดความสูงจะอยู่ในแนวโอเวอร์ชูทโพซิชัน)

ค่าปรกติของ BlueScales จะมีค่าเท่ากับ .039625 ซึ่งถูกคำนวณมาจากตัวอักษรขนาด 10 จุด บนอุปกรณ์แสดงผลที่มีความละเอียด 300 จุดต่อนิ้ว ดังนี้

BlueScales = (ขนาดตัวอักษร - 0.49) / 240

2.10.9 BlueShift เป็นค่าที่กำหนดขึ้นเพื่อเพิ่มความสามารถในการจัดการกับการทำโอเวอร์ชูท โดยค่า BlueShift นี้จะเป็นเลขจำนวนเต็มที่แสดงถึงระยะห่างจากตำแหน่งแบนราบของอไลเมนต์โซนจนถึงจุดที่ต้องการจะทำโอเวอร์ชูทกล่าวคือถ้าตัวอักษรใดที่เกินจากส่วนแบนราบของอไลเมนต์โซนมากกว่าค่า BlueShift ก็จะถูกทำโอเวอร์ชูท หรือถ้าตัวอักษรอยู่ใกล้กับส่วนแบนราบของอไลเมนต์โซนมาก จะถูกทำโอเวอร์ชูทก็ต่อเมื่อจำนวนจุดภาพที่ใช้บนอุปกรณ์แสดงผลมากกว่าครึ่งจุดภาพ

2.10.10 BlueFuzz เป็นเลขจำนวนเต็มที่ใช้กำหนดระยะเพื่อยืดขยายระยะของอไลเมนต์โซน เช่น หากส่วนของตัวอักษรเลยขอบเขตของอไลเมนต์โซนของแต่ละคู่แล้ว แต่ยังอยู่ในขอบเขตของ BlueFuzz ก็จะจัดการส่วนของตัวอักษรเสมือนอยู่ในอไลเมนต์โซนนั้นๆ

ค่าของ BlueFuzz จะมีผลกระทบต่อระยะห่างของระหว่างอไลเมนต์โซนแต่ละคู่ด้วย เนื่องจากค่าของ BlueFuzz ที่ไม่ใช่ 0 จะขยายขนาดของอไลเมนต์โซนออกไปดังนี้

ค่าของอไลเมนต์โซนที่ขยาย = BlueFuzz * 2 + 1

ดังนั้นการกำหนดค่า BlueValues, OtherBlues, FamilyBlues และ FamilyOtherBlues จะต้องคำนึงถึงระยะห่างระหว่างคู่จากค่า BlueFuzz ด้วย

2.10.11 Stem Width เป็นกลไกเพื่อบอกตัวแปลคำสั่งโพสต์สคริปต์เกี่ยวกับขนาดความกว้างมาตรฐานของสเตม เพื่อที่จะสามารถแสดงผลตัวอักษรที่มีขนาดเล็กๆได้อย่างถูกต้อง

หลักการคือถ้าความแตกต่างระหว่างความกว้างมาตรฐานที่กำหนดโดย Stem Width กับความกว้างของสเตมของตัวอักษรนั้นๆมีความแตกต่างกันเพียงเล็กน้อย ตัวแปลคำสั่งโพสต์สคริปต์จะเลือกใช้ความกว้างของสเตมตามค่ามาตรฐาน

ค่ามาตรฐานของสเตมสามารถกำหนดได้ดังนี้

2.10.11.1 ค่าขนาดความกว้างมาตรฐานเป็นค่าที่บอกถึงขนาดของความกว้างโดยส่วนใหญ่ของตัวอักษรในโปรแกรมแบบอักษรนั้นๆ โดย

StdHW คือค่าในแนวนอน และ StdVW คือค่าในแนวตั้ง

2.10.11.2 ค่าขนาดความกว้างอื่นๆเป็นค่าที่บอกถึงขนาดของความกว้างอื่นๆของตัวอักษร ที่มีการใช้บ่อย โดย

StemSnapH คือค่าในแนวนอน และ StemSnapV คือค่าในแนวตั้ง
จำนวนสูงสุดของค่าขนาดความกว้างที่กำหนดได้คือ 12 จำนวน

2.10.12 ForceBold เป็นตัวที่ใช้ในการกำหนดให้การแสดงผลตัวอักษรตัวหนา บนอุปกรณ์แสดงผลที่มีความละเอียดต่ำเช่น จอภาพ สามารถแสดงผลตัวอักษรตัวหนาให้แตกต่างจากตัวอักษรปรกติได้ เช่น สมมติว่าการแสดงผลตัวอักษรตัวหนามีความกว้าง 1 จุดภาพ ซึ่งเท่ากับการแสดงผลตัวอักษรปรกติทำให้ไม่เห็นความแตกต่าง ก็สามารถกำหนดให้ตัวแปลภาษาใช้เทคนิคพิเศษในการแสดงผลเพื่อให้ตัวอักษรมีความหนามากขึ้นได้ ค่าที่เป็นไปได้คือ true (ใช้เทคนิคพิเศษ) และ false (ไม่ใช้เทคนิคพิเศษ)

2.10.13 LanguageGroup เป็นการบอกตัวแปลคำสั่งโพสต์สคริปต์ว่าแบบอักษรในโปรแกรมจัดอยู่ในกลุ่มอักษรใด มีค่าที่เป็นไปได้ 2 ค่า คือ 0 เป็นภาษาทั่วไป และ 1 สำหรับภาษาจีน เกาหลี คันจิ

2.10.14 RndStemUp เป็นค่าที่ใช้ควบคู่กับ LanguageGroup โดยถ้า LanguageGroup เป็น 0 ค่า RndStemUp จะมีค่า true

2.10.15 password และ MinFeature
จะต้องกำหนดไว้ดังนี้เสมอ
/MinFeature {16 16} def
/password 5839 def

2.10.16 lenIV เป็นค่าที่ใช้กำหนดจำนวนของตัวเลขสุ่มเพื่อใช้ในการเข้ารหัส(เพื่อความเข้ากันได้กับตัวแปลคำสั่งโพสต์สคริปต์รุ่นต่างๆ ควรกำหนดให้มีค่าเป็น 4 เสมอ)

2.11 CharStrings dictionary เป็นส่วนที่มีความสำคัญที่สุดของโปรแกรมแบบอักษร เพราะใช้เป็นส่วนที่เก็บข้อมูลคำสั่งที่บรรยายการสร้างตัวอักษรและการกำหนดสเตมของตัวอักษรนั้นๆ คำสั่งต่างๆที่ใช้บรรยายการสร้างตัวอักษรนี้จะจัดเก็บแบบสัญกรณ์เติมหลัง กล่าวคือจะจัดเก็บค่าพารามิเตอร์ก่อนคำสั่งดังนี้

parameter ... parameter command

ในการจัดเก็บคำสั่งและค่าพารามิเตอร์นั้น จะถูกเข้ารหัส ซึ่งมีวิธีการดังนี้

2.11.1 การเข้ารหัสค่าพารามิเตอร์
ค่าพารามิเตอร์ซึ่งเป็นเลขจำนวนเต็มจะถูกแทนด้วยรหัส(v และ w) unsigned character byte ดังนี้

- รหัสตั้งแต่ 32 ถึง 246 จะใช้แทนเลขจำนวนเต็มระหว่าง -107 ถึง 107 โดยสูตร v - 139
- รหัสตั้งแต่ 247 ถึง 250 จะแสดงว่ารหัสที่ใช้แทนมี 2 ค่า โดยแทนเลขจำนวนเต็มระหว่าง 108 ถึง 1131 โดยสูตร [(v - 247) * 256] + w + 108 เมื่อ w คือรหัสที่ตามหลัง v
- รหัสตั้งแต่ 251 ถึง 254 จะแสดงว่ารหัสที่ใช้แทนมี 2 ค่า โดยแทนเลขจำนวนเต็มระหว่าง -1131 ถึง -108 โดยสูตร [(v - 247) * 256] + w + 108 เมื่อ w คือรหัสที่ตามหลัง v
- รหัส 255 จะแสดงว่ารหัสที่ตามมาอีก 4 ไบต์จะแทนด้วยเลขจำนวนเต็มที่เป็นแบบ 2's complement

2.11.2 การเข้ารหัสคำสั่ง

คำสั่งที่ใช้ในการบรรยายการสร้างแบบอักษร จะถูกแทนด้วยรหัส 0 ถึง 31 ยกเว้นรหัส 12 จะหมายถึงคำสั่งที่ถูกแทนด้วยรหัสจำนวน 2 ไบต์*

<u>รูปแบบคำสั่งของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1</u>

คำสั่งของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 จะแบ่งออกเป็น 5 กลุ่มย่อยประกอบด้วย คำสั่งในการเริ่มและจบการบรรยายตัวอักษร คำสั่งการบรรยายการลากเส้น คำสั่งเกี่ยวกับฮินต์ คำสั่งทางคณิตศาสตร์ และคำสั่งเกี่ยวกับโปรแกรมย่อย โดยค่าพารามิเตอร์ของคำสั่งที่เกี่ยวกับระยะทางจะใช้วิธีการสัมพัทธ์จากจุดปัจจุบันของตัวอักษร ดังรายละเอียดต่อไปนี้

1. คำสั่งในการเริ่มและจบการบรรยายตัวอักษร

1.1 endchar [*format :* endchar]

เป็นคำสั่งที่ใช้ในการจบการบรรยายการสร้างตัวอักษร โดยต้องเป็นคำสั่งสุดท้ายของการบรรยายตัวอักษรทุกตัว ยกเว้นตัวอักษรแบบแอ็คเซนต์(ซึ่งเป็นการกำหนดตัวอักษรโดยนำตัวอักษรสองตัวที่ได้บรรยายไว้แล้วมาประกอบกัน)ที่กำหนดโดยคำสั่ง seac

1.2 hsbw [*format :* sbx wx hsbw]

เป็นคำสั่งที่ใช้ในการเริ่มการบรรยายการสร้างตัวอักษรโดยต้องเป็นคำสั่งแรกของการบรรยาย คำสั่งนี้จะเป็นการกำหนด left sidebearing point ที่ตำแหน่ง (sbx, 0) และกำหนดความกว้างของตัวอักษรเท่ากับ wx (แต่จะยังไม่วางจุดเริ่มต้นที่ sbx จะต้องใช้คำสั่ง rmoveto เพื่อกำหนดจุดเริ่มต้นของตัวอักษร)

1.3 sbw [*format :* sbx sby wx wy sbw]

คำสั่งนี้เหมือนกับคำสั่ง hsbw แต่สามารถกำหนดจุด left sidebearing ที่จุด sbx, sby และกำหนดความกว้างรวมทั้งความสูงของตัวอักษรด้วย wx, wy

1.4 seac [*format :* asb adx ady bchar achar seac]

สำหรับโปรแกรมแบบอักษรที่กำหนดชื่อตัวอักษรและรหัสตัวอักษรตามมาตรฐานของบริษัทอโดบี** จะสามารถสร้างตัวอักษรโดยการนำตัวอักษรที่กำหนดไว้แล้วมาประกอบกันขึ้นเป็นตัวอักษรใหม่ได้ โดยเรียก

---

*ดูรายละเอียดเพิ่มเติมที่ภาคผนวก ก, หน้า 128.

**ดูรายละเอียดเพิ่มเติมที่ภาคผนวก ข, หน้า 129.

ตัวอักษรนี้ว่าตัวอักษรแบบแอ็คเซนต์ ค่าพารามิเตอร์ asb คือค่า x left sidebearing ของตัวอักษรที่จะเป็นส่วนแอ็คเซนต์ (เป็นค่าเดียวกับค่า left sidebearing ซึ่งกำหนดด้วยคำสั่ง hsbw หรือ sbw) จุดเริ่มต้นของส่วนที่เป็นแอ็คเซนต์จะเริ่มที่ adx, ady สัมพัทธ์จากจุดเริ่มต้นของส่วนที่เป็นเบสซึ่งกำหนดโดยรหัสตัวอักษร bchar ในขณะที่ส่วนที่เป็นแอ็คเซนต์กำหนดโดยรหัสตัวอักษร achar

สิ่งสำคัญในการใช้คำสั่งนี้ก็คือขอบเขตของตัวอักษร(FontBBox) จะต้องมีขนาดใหญ่พอที่จะสามารถแสดงผลได้ทั้งตัวอักษรแอ็คเซนต์และเบส และการใช้คำสั่งนี้ไม่ต้องจบด้วยคำสั่ง endchar สำหรับโปรแกรมแบบอักษรที่มีชื่อและรหัสตัวอักษรไม่ตรงตามมาตรฐานของบริษัทอโดบี ให้ใช้โปรแกรมย่อยแทนการใช้คำสั่งนี้

2. คำสั่งการลากเส้น

2.1 closepath [*format :* closepath]

เป็นคำสั่งที่ใช้เมื่อมีการจบการบรรยายส่วนของตัวอักษร บริษัทอโดบีแนะนำว่าควรจะต้องใช้คำสั่งนี้เสมอทุกครั้งที่จบการบรรยายส่วนของตัวอักษร

2.2 hlineto [*format :* dx hlineto]

เป็นคำสั่งในการลากเส้นตามแนวนอนเป็นระยะทางเท่ากับ dx

2.3 hmoveto [*format :* dx hmoveto]

เป็นคำสั่งในการเปลี่ยนตำแหน่งจุดปัจจุบันตามแนวนอนเป็นระยะทางเท่ากับ dx

2.4 hvcurveto [*format :* dx1 dx2 dy2 dy3 hvcurveto]

เป็นคำสั่งในการลากเส้นโค้งเบเซียที่มีจุดควบคุมความโค้ง 4 จุด ที่สัมพัทธ์จากตำแหน่งปัจจุบันที่ dx1 0 (dx1+dx2) dy2 (dx1+dx2) (dy2+dy3)

2.5 rlineto [*format :* dx dy rlineto]

เป็นคำสั่งการลากเส้นตรงไปตามแกน x เท่ากับ dx และตามแกน y เท่ากับ dy

2.6 rmoveto [*format :* dx dy rmoveto]

เป็นคำสั่งการเปลี่ยนตำแหน่งจุดปัจจุบันไปตามแกน x เท่ากับ dx และตามแกน y เท่ากับ dy

2.7 rrcurveto [*format :* dx1 dy1 dx2 dy2 dx3 dy3 rrcurveto]

เป็นคำสั่งในการลากเส้นโค้งเบเซียที่มีจุดควบคุมความโค้ง 4 จุด ที่สัมพัทธ์จากตำแหน่งปัจจุบันที่ dx1 dy1 (dx1+dx2) (dy1+dy2) (dx1+dx2+dx3) (dy1+dy2+dy3)

2.8 vhcurveto [*format* : dy1 dx2 dy2 dx3 vhcurveto]

เป็นคำสั่งในการลากเส้นโค้งเบเซียที่มีจุดควบคุมความโค้ง 4 จุด ที่สัมพัทธ์จากตำแหน่งปัจจุบันที่ 0 dy1 dx2 (dy1+dy2) (dx2+dx3) (dy1+dy2)

2.9 vlineto [*format* : dy vlineto]

เป็นคำสั่งในการลากเส้นตรงตามแนวตั้ง

2.10 vmoveto [*format* : dy vmoveto]

เป็นคำสั่งในการเปลี่ยนตำแหน่งปัจจุบันไปตามแนวตั้ง

3. คำสั่งฮินต์

3.1 dotsection [*format* : dotsection]

เป็นคำสั่งที่ใช้ครอบระหว่างคำสั่งในการบรรยายการลากเส้นในส่วนที่เป็นจุดของตัวอักษร เช่น i, j โดยคำสั่งแรกจะต้องตามหลังคำสั่ง rmoveto และคำสั่งที่สองจะต้องตามหลังคำสั่ง closepath ของการบรรยายการลากเส้นส่วนที่เป็นจุด

3.2 hstem [*format* : y dy hstem]

เป็นคำสั่งที่ใช้กำหนดความกว้างของสเตมแนวนอน ระหว่างค่า y กับ y+dy โดย y เป็นค่าสัมพัทธ์จากจุดเริ่มต้นของตัวอักษร

การกำหนดสเตมแนวนอนในแต่ละตัวอักษรจะเหลื่อมล้ำกันไม่ได้ ดังนั้นถ้ามีการกำหนดสเตมที่เหลื่อมล้ำกัน จะต้องใช้วิธีการเปลี่ยนสเตมก่อน(ซึ่งเรียกการเปลี่ยนสเตมนี้ว่าการเปลี่ยนสเตมฮินต์)

3.3 hstem3 [*format* : ymin dymin ymid dymid ymax dymax hstem3]

เป็นคำสั่งที่ใช้กำหนดความกว้างของสเตมแนวนอน 3 สเตม ระหว่างค่า ymin กับ ymin+dymin ymid กับ ymid+dymid ymax กับ ymax+dymax โดย ymin, ymid, ymax เป็นค่าสัมพัทธ์จากจุดเริ่มต้นของตัวอักษร โดยค่าของสเตมเหล่านี้จะต้องเรียงลำดับจากค่าน้อยไปมากเสมอและต้องเป็นไปตามกฎ

- ค่า dymin = dymax
- ระยะห่างจาก ymin + dymin/2 ถึง ymid + dymid/2 ต้องมีค่าเท่ากันกับระยะห่างจาก ymid + dymid/2 ถึง ymax + dymax/2
- เมื่อใช้คำสั่งนี้แล้ว จะต้องไม่ใช้คำสั่ง hstem

3.4 vstem [*format* : x dx vstem]

เป็นคำสั่งที่ใช้กำหนดความกว้างของสเตมแนวตั้ง ระหว่างค่า x กับ x+dx โดย x เป็นค่าสัมพัทธ์จากจุดเริ่มต้นของตัวอักษร

การกำหนดสเตมแนวตั้งในแต่ละตัวอักษรจะเหลื่อมล้ำกันไม่ได้ ดังนั้นถ้ามีการกำหนดสเตมที่เหลื่อมล้ำกัน จะต้องใช้วิธีการเปลี่ยนสเตมฮินต์ก่อน

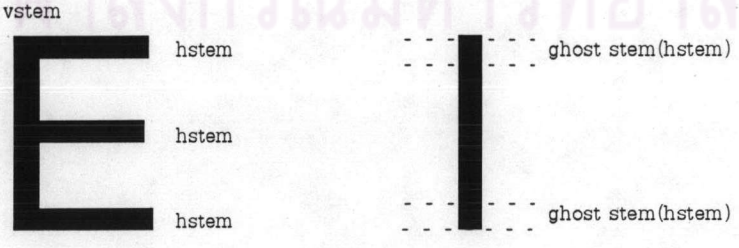
3.5 vstem3 [*format* : xmin dxmin xmid dxmid xmax dxmax vstem3]

เป็นคำสั่งที่ใช้กำหนดความกว้างของสเตมแนวตั้ง 3 สเตม ระหว่างค่า xmin กับ xmin+dxmin xmid กับ xmid+dxmid xmax กับ xmax+dxmax โดย xmin, xmid, xmax เป็นค่าสัมพัทธ์จากจุดเริ่มต้นของตัวอักษร โดยค่าของสเตมเหล่านี้จะต้องเรียงลำดับจากค่าน้อยไปมากเสมอ และต้องเป็นไปตามกฎ

- ค่า dxmin = dxmax
- ระยะห่างจาก xmin + dxmin/2 ถึง xmid + dxmid/2 ต้องมีค่าเท่ากันกับระยะห่างจาก xmid + dxmid/2 ถึง xmax + dxmax/2
- เมื่อใช้คำสั่งนี้แล้ว จะต้องไม่ใช้คำสั่ง vstem

สเตมฮินต์ซึ่งกำหนดโดยคำสั่งฮินต์ข้างต้น จะถูกนำไปใช้ร่วมกันกับอไลเมนต์โซนที่กำหนดไว้ในส่วน Private dictionary เพื่อใช้ช่วยในการแสดงผล

ดังที่ได้กล่าวแล้วว่าสำหรับบางตัวอักษรที่ไม่มีลักษณะของสเตมที่เรียกสเตมประเภทนี้ว่า "ghost stems" เช่น ตัวอักษรภาษาอังกฤษตัวใหญ่ I ของบางแบบอักษร เพื่อที่จะให้ระบบฮินต์ที่ช่วยในการแสดงผลตัวอักษรบริเวณส่วนบนและล่างของตัวอักษรสามารถทำงานได้จะต้องกำหนดghost stemsบริเวณแคปไฮต์อไลเมนต์โซนและเบสไลน์อไลเมนต์โซน(ดูรูปที่ 2.9)




รูปที่ 2.9 แสดงการกำหนดสเตมของแต่ละตัวอักษร

4. คำสั่งทางคณิตศาสตร์ มีเพียงคำสั่งเดียวที่โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ยอมให้ใช้ได้คือ คำสั่ง div [*format :* num1 num2 div quotient] เพื่อใช้ในกรณีที่ค่าพารามิเตอร์ของคำสั่งต่างๆ มีค่าเกินกว่า 32,000

5. คำสั่งที่เกี่ยวกับโปรแกรมย่อย

5.1 callothersubr [*format :* arg1....argn n othersubr# callothersubr]

เป็นคำสั่งที่จะถูกใช้โดยตัวสร้างแบบอักษรประเภทที่ 1 เพื่อจัดการแสดงผลเฟลกซ์ของตัวอักษร รวมทั้งการเปลี่ยนแปลงสเตมภายในตัวอักษร

5.2 callsubr [*format :* subr# callsubr]

เป็นคำสั่งที่ใช้ในการเรียกใช้โปรแกรมย่อยหมายเลข subr# จาก Subrs ที่กำหนดในส่วน Private dictionary

5.3 pop [*format :* pop number]

เป็นคำสั่งที่ใช้ร่วมกับคำสั่ง callothersubr เพื่อดึงค่าออกจาก operand stack ของตัวแปลคำสั่งโพสต์สคริปต์และเก็บค่านั้นไว้ใน operand stack ของตัวสร้างอักษร

5.4 return [format : return]

เป็นคำสั่งสุดท้ายของส่วนที่ใช้บรรยายของโปรแกรมย่อย เพื่อคืนการทำงานให้กับส่วนที่เรียก

5.5 setcurrentpoint [*format :* x y setcurrentpoint]

เป็นคำสั่งที่ใช้ในส่วนของโปรแกรมย่อยแบบ OtherSubrs เพื่อกำหนดจุดปัจจุบันที่ตำแหน่ง (x,y)

<u>การเข้ารหัสและถอดรหัสลับโปรแกรมแบบอักษร</u>

โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 นั้น จะมีการเข้ารหัสลับส่วนของโปรแกรม 2 ส่วน คือ

1. การเข้ารหัสลับส่วน charstring
   เป็นการเข้ารหัสลับส่วนที่บรรยายการสร้างตัวอักษรแต่ละตัว

2. การเข้ารหัสลับส่วน eexec
   เป็นการเข้ารหัสลับส่วนของ Private และ CharString dictionary ทั้งหมด

หลักการในการเข้ารหัสลับทั้ง2ส่วนนี้จะเหมือนกันแต่แตกต่างกันที่ค่าของจำนวนตัวเลขสุ่มและค่าเริ่มต้นของกุญแจ ซึ่งมีขั้นตอนดังนี้

1. สุ่มตัวเลขขึ้นมาตามจำนวนที่กำหนด แล้วรวมเข้ากับไบต์ของข้อมูลโดยไว้ที่ส่วนเริ่มต้น

1.1 จำนวนตัวเลขสุ่มสำหรับการเข้ารหัสลับส่วน charstring โดยปรกติจะมีจำนวนเท่ากับ 4 ตัว ยกเว้นแต่ว่าได้มีการกำหนดจำนวนไว้ด้วยคำสั่ง lenIV ใน Private dictionary (ตัวแปลคำสั่งโพสต์สคริปต์รุ่นที่ 23.0 กำหนดให้จำนวนตัวเลขสุ่มเท่ากับ 4)

1.2 จำนวนตัวเลขสุ่มสำหรับการเข้ารหัสลับส่วน eexec กำหนดไว้เท่ากับ 4 และตัวเลขที่สุ่มจะต้องเป็นไปตามกฎเกณฑ์ดังนี้

1.2.1 เลขสุ่มตัวแรกจะต้องไม่ใช่ตัวอักษรว่างเปล่าของรหัสแอสกี อันประกอบด้วย blank, tab, carriage return และ line feed

1.2.2 อย่างน้อย 1 ใน 4 ตัวของเลขสุ่ม จะต้องไม่ใช่ตัวอักษรที่ใช้แทนเลขฐาน 16 ประกอบด้วยรหัสแอสกีของ 0 - 9 A - F a - f

2. กำหนดกุญแจเริ่มต้น(R)ที่จะใช้เข้ารหัส โดยต้องเป็นเลขจำนวนเต็มไม่มีเครื่องหมายขนาด 16 บิต

2.1 กุญแจเริ่มต้นสำหรับการเข้ารหัสลับส่วน charstring คือ 4330

2.2 กุญแจเริ่มต้นสำหรับการเข้ารหัสลับส่วน eexec คือ 55665

3. สำหรับแต่ละไบต์ของข้อมูลที่รวมตัวเลขสุ่มแล้ว(P)

3.1 กำหนดค่าให้กับตัวแปรชั่วคราว(T)ด้วยค่า 8 บิตสูงของ R

3.2 ทำ exclusive OR(XOR) P กับ T จะได้ไบต์ที่เข้ารหัสลับแล้ว

3.3 คำนวณค่าของกุญแจ(R)ใหม่ด้วยสูตร
((C + R)*C1 + C2) mod 65536 เมื่อ C1 = 52845 และ C2 = 22719

3.4 กลับไปทำข้อ 3.1 ใหม่ จนกว่าข้อมูลจะหมด

ส่วนหลักการในการถอดรหัสลับทั้ง 2 ส่วน จะทำดังนี้

1. กำหนดกุญแจเริ่มต้น(R)ที่จะใช้ถอดรหัสลับ(เหมือนกับกุญแจที่ใช้ตอนเข้ารหัสลับ)

2. สำหรับแต่ละไบต์ของข้อมูล(C)

   2.1 กำหนดค่าให้กับตัวแปรชั่วคราว(T) ด้วยค่า 8 บิตสูงของ R

   2.2 ทำ exclusive OR(XOR) C กับ T จะได้ไบต์ที่ถอดรหัสแล้ว

   2.3 คำนวณค่าของกุญแจ(R)ใหม่ด้วยสูตร
   ((C+R)*C1+C2) mod 65536 โดยค่า C1 และ C2 เหมือนกับ C1 และ C2 ที่ใช้ตอนเข้ารหัสลับ

   2.4 กลับไปทำข้อ 2.1 ใหม่ จนกว่าข้อมูลจะหมด

3. ตัดไบต์ที่อยู่หน้าสุดออกตามจำนวนเดียวกันกับที่รวมเข้ากับไบต์ข้อมูลตอนเข้ารหัสลับ ก็จะได้ส่วนของข้อมูลที่ถอดรหัสลับแล้ว

<u>โปรแกรมย่อยของแบบอักษร</u>

วัตถุประสงค์ของการใช้โปรแกรมย่อยของแบบอักษรนอกจากเพื่อที่จะลดขนาดของโปรแกรมโดยการรวบรวมส่วนที่บรรยายการสร้างตัวอักษรที่ใช้บ่อยๆไว้เป็นโปรแกรมย่อยแล้ว การใช้โปรแกรมย่อยยังมีวัตถุประสงค์เพื่อทำการเปลี่ยนสเตมของตัวอักษรรวมทั้งการเรียกใช้เฟลกซ์อีกด้วย ซึ่งโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ได้กำหนดโปรแกรมย่อยไว้ 2 ประเภทคือ Subrs และ OtherSubrs และใช้หมายเลขสำหรับการกำหนดหรือเรียกใช้โปรแกรมย่อย

การกำหนดโปรแกรมย่อยประเภท Subrs นั้น หากในโปรแกรมแบบอักษรที่สร้างมีการทำงานเกี่ยวกับเฟลกซ์หรือการเปลี่ยนแปลงเสตม โปรแกรมย่อยหมายเลข 0 - 3 จะต้องกำหนดดังนี้

Subrs 0: 3 0 callothersubr pop pop setcurrentpoint return
Subrs 1: 0 1 callothersubr return
Subrs 2: 0 2 callothersubr return
Subrs 3: return

Subrs หมายเลข 0-2 จะใช้ทำงานเกี่ยวกับเฟลกซ์ และ Subrs หมายเลข 3 จะทำงานเกี่ยวกับการเปลี่ยนแปลงฮินต์ โดย Subrs หมายเลข 0 จะเป็นการเรียกโปรแกรมย่อย OtherSubrs หมายเลข 0 โดยส่งค่าพารามิเตอร์ 3 ค่า สำหรับการเรียกใช้เฟลกซ์ หลังจากนั้นจะดึงค่าออกจาก operand stack เพื่อกำหนดจุดปัจจุบันใหม่ แล้วส่งการทำงานกลับ Subrs หมายเลข 1 และ 2 เป็นการกำหนดให้เรียกใช้โปรแกรมย่อย OtherSubrs หมายเลข 1 และ 2 เป็นการประหยัดเนื้อที่ในการเก็บคำสั่งการเรียกใช้เฟลกซ์ ส่วน Subrs หมายเลข 3 ใช้กรณีที่ตัวสร้างอักษรไม่สามารถทำการเปลี่ยนแปลงสเตมได้ สำหรับ Subrs หมายเลข 4 เป็นต้นไป เป็นโปรแกรมย่อยที่ผู้สร้างโปรแกรมแบบอักษรสามารถกำหนดขึ้นมาเพื่อใช้งานทั่วๆไปได้ แต่ในกรณีที่โปรแกรมแบบอักษรไม่ได้มีการทำงานเกี่ยวกับเฟลกซ์หรือการเปลี่ยนแปลงสเตม ผู้สร้างแบบอักษรจะสามารถกำหนดโปรแกรมย่อยไว้ใช้ได้ตั้งแต่ Subrs หมายเลข 0 เป็นต้นไป

โปรแกรมย่อยแบบ Subrs นี้ จะถือว่าเป็น charstring ประเภทหนึ่งที่จะต้องมีการแทนด้วยรหัสและเข้ารหัสเช่นเดียวกับ charstring ทุกประการ ส่วนโปรแกรมย่อยประเภท OtherSubrs นั้น จะใช้ในการทำงานเกี่ยวกับเฟลกซ์ (0-2) และการเปลี่ยนแปลงสเตม(3) เมื่อตัวแปลคำสั่งโพสต์สคริปต์ไม่มีขั้นตอนการทำงานของตนเอง ผู้สร้างโปรแกรมแบบอักษรไม่สามารถกำหนดโปรแกรมย่อยประเภทนี้ไว้ใช้งานอื่นได้ โดยโปรแกรมย่อยประเภทนี้จะประกอบด้วยคำสั่งต่างๆของภาษาโพสต์สคริปต์ ซึ่งทุกโปรแกรมแบบอักษรที่มีการทำงานเกี่ยวกับเฟลกซ์หรือการเปลี่ยนแปลงฮินต์จะต้องรวมโปรแกรมย่อยนี้เข้าเป็นส่วนหนึ่งของโปรแกรมไว้ด้วยเสมอ

หลักเกณฑ์การใช้งานโปรแกรมย่อยเพื่อใช้งานต่างๆมีดังนี้

1. โปรแกรมย่อย Subrs ที่ผู้สร้างออกแบบไว้บรรยายแบบอักษร มีหลักการเหมือนกันกับการกำหนด charstring ทุกประการ

2. การเปลี่ยนแปลงสเตม เนื่องจากการกำหนดสเตมของตัวอักษรจะมีการเหลื่อมล้ำกันไม่ได้ จึงจำเป็นที่จะต้องทำการเปลี่ยนแปลงสเตมใหม่ โดยสเตมใหม่นี้จะต้องถูกเก็บอยู่ในโปรแกรมย่อยแบบ Subrs หมายเลขใดๆ ก็ได้ยกเว้นหมายเลข 0-3 เมื่อต้องการเปลี่ยนแปลงสเตม ให้แทรกคำสั่งใน charstring ตรงตำแหน่งที่ต้องการดังนี้

subr# 1 3 callothersubr pop callsubr เมื่อ subr# คือหมายเลขของโปรแกรมย่อยของสเตมใหม่

3. การทำงานเกี่ยวกับเฟลกซ์(ดูรูป2.6) สำหรับเส้นโค้งเบเซียที่ต้องการเปลี่ยนให้เป็นเฟลกซ์จะต้องทำตามขั้นตอนดังนี้

3.1 หาจุดเริ่มต้นของเส้นโค้ง

3.2 คำนวณระยะห่างสัมพัทธ์จากจุดเริ่มต้นจนถึงจุดอ้างอิง ซึ่งเป็นจุดที่อยู่ในแนวเดียวกันกับจุดเริ่ม-จุดปลาย และจุดเชื่อมต่อ

3.3 ตัดคำสั่ง rrcurveto ออกไป จะเหลือคู่ลำดับ 6 คู่(12 ค่า)

3.4 คำนวณค่าของคู่ลำดับคู่แรกใหม่ โดยให้สัมพัทธ์กับจุดอ้างอิง

3.5 แทรกค่าที่คำนวณได้จากข้อ 2 ไว้ข้างหน้าคู่ลำดับ 6 คู่

3.6 แทรกคำสั่ง 1 callsubr ไว้ข้างหน้าคู่ลำดับ 7 คู่ และแทรกคำสั่ง rmoveto และ 2 callsubr ไว้ที่ข้างท้ายของแต่ละคู่ลำดับ

3.7 ต่อท้ายคู่ลำดับด้วยค่าพารามิเตอร์ที่ควบคุมความสูงของเฟลกซ์(กำหนดในหน่วยของอุปกรณ์แสดงผล)และจุดสิ้นสุดสัมบูรณ์ของเส้นโค้ง แล้วตามด้วยคำสั่ง 0 callsubr

<u>รูปแบบพิเศษของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1</u>

โปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ยังมีรูปแบบพิเศษนอกเหนือไปจากที่กล่าวมาข้างต้นแล้วอีก 2 รูปแบบ ดังนี้

1. Synthetic Fonts เป็นโปรแกรมแบบอักษรที่แก้ไขค่าพารามิเตอร์บางอย่างของอีกโปรแกรมแบบอักษร เพื่อให้แบบอักษรมีลักษณะที่เปลี่ยนแปลงไป เช่น เปลี่ยนเป็นตัวเอียง โดยที่ไม่ต้องสร้างคำสั่งการบรรยายอักษรใหม่ ซึ่งการใช้ synthetic font นี้ จะมีประโยชน์ในการประหยัดหน่วยความจำของตัวแปลคำสั่งโพสต์สคริปต์

กลไกการทำงานของโปรแกรมแบบอักษรนี้ จะใช้ข้อมูลส่วน charstring ของโปรแกรมแบบอักษรที่อยู่ในหน่วยความจำแล้ว โดยจะใช้หน่วยความจำเพิ่มเพื่อเก็บข้อมูลส่วน font dictionary ของตัวมันเองเท่านั้น อย่างไรก็ตามมีความเป็นไปได้ว่าในหน่วยความจำอาจจะยังไม่มีข้อมูลcharstringของแบบอักษรที่ต้องการอยู่เลย ดังนั้น โปรแกรมแบบอักษรนี้จึงจำเป็นที่จะต้องรวมเอาส่วน charstring ของแบบอักษรที่ต้องการแก้ไขไว้ด้วย

2. Hybrid Fonts เป็นโปรแกรมแบบอักษรที่ประกอบด้วยคำสั่งในการบรรยายการสร้างแบบอักษร 2 ชุด โดยแต่ละชุดจะถูกเลือกใช้โดยพิจารณาจากความละเอียดของอุปกรณ์แสดงผล โดยทั่วไปการออกแบบคำสั่งในการบรรยายชุดแรกจะออกแบบเพื่อให้ใช้กับอุปกรณ์แสดงผลที่มีความละเอียดต่ำ ส่วนชุดที่สองจะออกแบบเพื่อให้ใช้กับอุปกรณ์การแสดงผลที่มีความละเอียดสูง เช่น การใช้เส้นโค้งหลายๆเส้นแทนการใช้เส้นตรงเส้นเดียว

ในโปรแกรมแบบอักษรจะประกอบด้วยส่วนของ Subrs และ charstring 2 ชุด โดยแต่ละชุดจะถูกคั่นด้วยคำสั่งของภาษาโพสต์สคริปต์ เพื่อใช้ในการตรวจสอบความละเอียดของอุปกรณ์การแสดงผล

ความเข้ากันได้กับซอฟต์แวร์เอทีเอ็ม

เนื่องจากซอฟต์แวร์เอทีเอ็มไม่ได้ถูกสร้างขึ้นมาอย่างสมบูรณ์เหมือนกับตัวแปลคำสั่งโพสต์สคริปต์ ทำให้ไม่สามารถทำงานได้กับคำสั่งหรือรูปแบบคำสั่งบางอย่างที่ภาษาโพสต์สคริปต์ยินยอมให้ใช้ได้ ดังนั้นโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ที่จะใช้กับซอฟต์แวร์เอทีเอ็ม จึงจำเป็นที่จะต้องจัดทำขึ้นตามข้อกำหนดต่างๆ อย่างเคร่งครัด ดังต่อไปนี้

1. ความยาวของแต่ละ charstring จะต้องไม่ยาวเกิน 65,535 ตัวอักษร

2. ส่วน Private และ CharString dictionary จะต้องอยู่หลังสุดของพจนานุกรมแบบอักษร

3. การกำหนดค่าต่างๆภายหลัง /Private จะเป็นการกำหนดค่าของส่วน Private dictionary เท่านั้น

4. ถ้ามีคำ eexec จะต้องอยู่ก่อน /Private และข้อความที่ตามหลังจะต้องเข้ารหัสเสมอ

5. การกำหนดค่าพารามิเตอร์ จะต้องอยู่ในรูปแบบดังนี้เท่านั้น
   คำสำคัญ ค่าพารามิเตอร์ที่กำหนด เช่น /FontType 1 def

6. ค่าของตัวแปรแถวลำดับ จะต้องอยู่ภายใต้เครื่องหมาย [ ] หรือ { } เสมอ

7. ข้อมูลแบบเลขฐานสอง จะปรากฏในโปรแกรมแบบอักษรได้ในส่วนของ Subrs และ CharStrings เท่านั้น

8. จำนวนเต็มที่ตามหลัง /Subrs จะต้องเป็นจำนวนของโปรแกรมย่อยแบบ Subrs ที่ปรากฏอยู่จริง

9. การกำหนดโปรแกรมย่อยจะต้องอยู่ภายใต้ /Subrs เท่านั้น

10. จำนวนเต็มที่ตามหลัง /CharStrings จะต้องมีค่ามากกว่าหรือเท่ากับจำนวน charstring ที่ใช้บรรยายตัวอักษร

11. ข้อความส่วน /CharStrings จะต้องปิดท้ายด้วยคำสั่ง end เสมอ

12. ถ้าโปรแกรมแบบอักษรมีคำว่า FontDirectory ภายหลัง /Private แสดงว่าโปรแกรมแบบอักษรนี้เป็น synthetic font แต่ถ้ามีคำว่า hires ภายหลัง /Private แสดงว่าโปรแกรมแบบอักษรนี้เป็น hybrid font

โครงสร้างแฟ้มข้อมูลพีเอฟเอ็ม

สำหรับโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ที่จะใช้กับซอฟต์แวร์เอทีเอ็มนั้น จำเป็นที่จะต้องมีแฟ้มข้อมูลพีเอฟเอ็มร่วมด้วย โดยแฟ้มข้อมูลพีเอฟเอ็มนี้จะใช้เป็นที่เก็บข้อมูลเกี่ยวกับความกว้างของแต่ละตัวอักษรและความกว้างของช่องไฟระหว่างตัวอักษรต่างๆเพื่อให้โปรแกรมประยุกต์บางโปรแกรมเรียกใช้ เช่น โปรแกรมการจัดเรียงพิมพ์เอกสาร โครงสร้างแฟ้มข้อมูลพีเอฟเอ็มจะประกอบด้วย

1. โครงสร้างส่วนหัวของพีเอฟเอ็ม(PFMHEADER)
2. โครงสร้างส่วนขยายของพีเอฟเอ็ม(PFMEXTENSION)
3. โครงสร้าง EXTTEXTMETRIC
4. ตัวแปรแถวลำดับที่เก็บข้อมูลความกว้างของแต่ละตัวอักษร
5. ตัวแปรแถวลำดับที่เก็บข้อมูลความกว้างของช่องไฟระหว่างตัวอักษรต่างๆ
6. ชื่อของแบบอักษร ชื่อของโปรแกรมขับอุปกรณ์(driver)

รายละเอียดของค่าต่างๆในแฟ้มข้อมูลพีเอฟเอ็ม แสดงดังตารางที่ 2.1 ถึง 2.4

ตารางที่ 2.1 แสดงโครงสร้าง PFMHEADER

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| WORD | dfVersion | แสดงรุ่นของแฟ้มข้อมูล |
| WORD | dfSize | ขนาดของแฟ้มข้อมูล |
| char | dfCopyright[60] | ข้อมูลเกี่ยวกับลิขสิทธ์ |
| WORD | dfType | แบบของแฟ้มข้อมูลตัวอักษร(กำหนดต้อง = 1) |
| WORD | dfPoints | ขนาดปรกติของตัวอักษร |
| WORD | dfVertRes | ความละเอียดของอุปกรณ์แสดงผลในแนวตั้ง(ค่าปรกติ = 300 จุดต่อนิ้ว) |
| WORD | dfHorizRes | ความละเอียดของอุปกรณ์แสดงผลในแนวนอน(ค่าปรกติ = 300 จุดต่อนิ้ว) |
| WORD | dfAscent | ระยะห่างจากจุดสูงสุดของตัวอักษรถึงเบสไลน์ |
| WORD | dfInternalLeading | ไม่ใช้ |
| WORD | dfExternalLeading | ไม่ใช้ |
| BYTE | dfItalic | ค่าที่แสดงว่าเป็นแบบอักษรตัวเอียง(ค่า = 1) |
| BYTE | dfUnderline | ค่าที่แสดงว่าเป็นแบบอักษรที่ขีดเส้นใต้(ค่า = 1) |
| BYTE | dfStrikeOut | ค่าที่แสดงว่าเป็นแบบอักษรที่มีขีดกลาง(ค่า = 1) |
| WORD | dfWeight | ค่าที่บอกขนาดความหนาของตัวอักษร |
| BYTE | dfCharSet | ค่าที่บอกกลุ่มของตัวอักษร<br>0 = ANSI character set<br>2 = Symbol character set<br>255 = IBMPC character set |
| WORD | dfPixWidth | ไม่ใช้ |
| WORD | dfPixHeight | ไม่ใช้ |
| BYTE | dfPitchAndFamily | ใช้กำหนด pitch และ family โดยค่า 4 บิตต่ำต้องไม่ใช่ 0 และค่า 4 บิตสูงจะเป็นดังนี้<br>0 = DONTCARE<br>1 = ROMAN<br>2 = SWISS<br>3 = MODERN<br>4 = SCRIPT |
| WORD | dfAvgWidth | ค่าความกว้างเฉลี่ย(ปรกติใช้ค่าความกว้างของ X) |

ตารางที่ 2.1 แสดงโครงสร้าง PFMHEADER(ต่อ)

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| WORD | dfMaxWidth | ค่าความกว้างสูงสุดของแบบอักษร |
| BYTE | dfFirstChar | รหัสตัวอักษรที่เป็นตัวแรกของโปรแกรมแบบอักษร |
| BYTE | dfLastChar | รหัสตัวอักษรที่เป็นตัวสุดท้ายของโปรแกรมแบบอักษร |
| BYTE | dfDefaultChar | รหัสตัวอักษรที่เป็นตัวอักษรปรกติของโปรแกรมแบบอักษร โดยเป็นค่าสัมพัทธ์กับรหัสตัวอักษรตัวแรก(รหัสตัวอักษรปรกติ = dfDefaultChar - dfFirstChar) |
| BYTE | dfBreakChar | รหัสตัวอักษรที่ใช้เป็นตัว word break ในการตัดคำ โดยเป็นค่าสัมพัทธ์กับรหัสตัวอักษรตัวแรก(รหัสตัวอักษร word break = dfDefaultChar - dfBreakChar) |
| WORD | dfWidthBytes | ไม่ใช้ |
| DWORD | dfDevice | ออฟเซ็ทที่ชี้ไปยังชื่อของอุปกรณ์แสดงผล (PostScript เท่านั้น) |
| DWORD | dfFace | ออฟเซ็ทที่ชี้ไปยังชื่อของแบบอักษร |
| DWORD | dfBitsPointer | ไม่ใช้ |
| DWORD | dfBitsOffset | ไม่ใช้ |

**ที่มา** : Microsoft Staff, **Microsoft Windows device driver adaptation guide** (Washington; Microsoft Press, 1990), p.12-28 - 12-32.

ตารางที่ 2.2 แสดงโครงสร้าง PFMEXTENSION

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| WORD | dfSizeFields | ขนาดของข้อมูลส่วนนี้ |
| DWORD | dxExtMetricsOffset | ออฟเซ็ทที่ชี้ไปยังโครงสร้าง EXTTEXTMETRIC |
| DWORD | dfExtentTable | ออฟเซ็ทที่ชี้ไปยังตัวแปรแถวลำดับของความกว้างของแต่ละตัวอักษร ซึ่งมีขนาดเท่ากับ dfLastChar - dfFirstChar + 1 WORD |
| DWORD | dfOriginTable | ไม่ใช้ |
| DWORD | dfPairKernTable | ออฟเซ็ทที่ชี้ไปยังตัวแปรแถวลำดับที่เก็บข้อมูลของความกว้างช่องไฟระหว่างตัวอักษร |

ตารางที่ 2.2 แสดงโครงสร้าง PFMEXTENSION(ต่อ)

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| DWORD | dfTrackKernTable | ไม่ใช้ |
| DWORD | dfDriverInfo | ออฟเซ็ทที่ชี้ไปยังชื่อของแบบอักษร |
| DWORD | dfReserved | ไม่ใช้ |

**ที่มา** : Microsoft Staff, **Printer and Fonts Kit** (Washington; Microsoft Press, 1990), p.2-6.

ตารางที่ 2.3 แสดงโครงสร้าง EXTTEXTMETRIC

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| short | etmSize | ขนาดของข้อมูลส่วนนี้ |
| short | etmPointSize | ขนาดปรกติของแบบอักษรในหน่วย twips |
| short | etmOrientation | ฟอนท์โอเรียนเตชัน ( 0 = ทั้ง potrait และ land-scape, 1 = potrait, 2 = landscape) |
| short | etmMasterHeight | กำหนดความสูงของแบบอักษรในรูปของจำนวนจุดของอุปกรณ์แสดงผล โดยความสูงของแบบอักษร = etmMasterHeight * 72 points-per-inch / dfVertRes |
| short | etmMinScale | กำหนดจำนวนจุดของอุปกรณ์แสดงผลเพื่อควบคุมขนาดที่เล็กที่สุดของตัวอักษรที่จะสามารถแสดงผลได้โดย ขนาดเล็กสุด = etmMinScale * 72 / dfVertRes |
| short | etmMaxScale | กำหนดจำนวนจุดของอุปกรณ์แสดงผลเพื่อควบคุมขนาดที่ใหญ่ที่สุดของตัวอักษรที่จะสามารถแสดงผลได้โดย ขนาดใหญ่สุด = etmMaxScale * 72 / dfVertRes |
| short | etmMasterUnits | อัตราส่วนระหว่างขนาดของแบบอักษรต่อค่า etmMasterHeight (แบบอักษรโพสต์สคริปต์กำหนด = 1000) |
| short | etmCapHeight | ความสูงของตัวอักษรตัวใหญ่(ทั่วไปคือ H) |
| short | etmXHeight | ความสูงของตัวอักษรตัวเล็ก(ทั่วไปคือ x) |
| short | etmLowerCaseAscent | ความสูงของส่วนหางที่สูงกว่าเบสไลน์(ทั่วไปคือ d) |
| short | etmUpperCaseDescent | ความสูงของส่วนหางที่ต่ำกว่าเบสไลน์(ทั่วไปคือ p) |
| short | etmSlant | ความเอียงของตัวอักษรคิดเป็นองศาวัดตามเข็มนาฬิกา |

ตารางที่ 2.3 แสดงโครงสร้าง EXTTEXTMETRIC(ต่อ)

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| short | etmSuperScript | ตำแหน่งของตัวอักษรตัวยกขึ้นนับจากเบสไลน์ |
| short | etmSubScript | ตำแหน่งของตัวอักษรตัวห้อยล่างนับจากเบสไลน์ |
| short | stmSuperScriptSize | ขนาดของตัวอักษรตัวยกขึ้น |
| short | etmSupScriptSize | ขนาดของตัวอักษรตัวห้อยล่าง |
| short | etmUnderlineOffset | ตำแหน่งของเส้นใต้นับจากเบสไลน์ |
| short | etmUnderlineWidth | ขนาดความกว้างของเส้นใต้ |
| short | etmDoubleUpperUnderlineOffset | ตำแหน่งของเส้นใต้คู่เส้นบนนับจากเบสไลน์ |
| short | etmDoubleLowerUnderlineOffset | ตำแหน่งของเส้นใต้คู่เส้นล่างนับจากเบสไลน์ |
| short | etmDoubleUpperUnderlineWidth | ขนาดความกว้างของเส้นใต้คู่เส้นบนนับจากเบสไลน์ |
| short | etmDoubleLowerUnderlineWidth | ขนาดความกว้างของเส้นใต้คู่เส้นล่างนับจากเบสไลน์ |
| short | etmStrikeOutOffset | ตำแหน่งของเส้นกลางตัวอักษรนับจากเบสไลน์ |
| short | etmStrikeOutWidth | ขนาดความกว้างของเส้นกลางตัวอักษร |
| WORD | etmKernPairs | จำนวนคู่ของตัวอักษรที่กำหนดความกว้างของช่องไฟ (มีได้สูงสุด 512 คู่) |
| WORD | etmKernTracks | ไม่ใช้ |

**ที่มา** : Microsoft Staff, **Printer and Fonts Kit** (Washington; Microsoft Press, 1990), p.2-7.

ตารางที่ 2.4 แสดงโครงสร้างของความกว้างช่องไฟระหว่างตัวอักษร

| ประเภทข้อมูล | ชื่อเขตข้อมูล | ความหมาย |
|---|---|---|
| BYTE | kpPairs.each[0] | รหัสตัวอักษรตัวแรก |
| BYTE | kpPairs.each[1] | รหัสตัวอักษรที่สอง |
| short | kpKernAmount | ค่าความกว้างของช่องไฟ<br>ถ้ามีค่า < 0 แสดงว่าตัวอักษรจะอยู่ชิดกันมากกว่าปรกติ<br>ถ้ามีค่า > 0 แสดงว่าตัวอักษรจะอยู่ห่างกันมากกว่าปรกติ |

**ที่มา** : Microsoft Staff, **Printer and Fonts Kit** (Washington; Microsoft Press, 1990), p.2-13.