ปีการศึกษา 2533

การวางแผนความต้องการวัสดุ

(Material Requirement Planning - MRP)

โดย

นาย บุญชัย ชาญภิญโญ เลขประจำตัว 301126

นาย บุณยฤทธิ์ บันธไชย เลขประจำตัว 301131

นาย วิญญู กิ่งหิรัญวัฒนา เลขประจำตัว 301251

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ดร. ชม กิ้มปาน

ปริญญานิพนธ์ ปีการศึกษา 2533

ภาควิชา คอมพิวเตอร์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง การวางแผนความต้องการวัสดุ

ผู้จัดทำ

1. นาย บุญชัย ชาญภิญโญ
2. นาย บุณยฤทธิ์ ขันธไชย
3. นาย วิญญ กิ่งหิรัญวัฒนา

อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ สิ้มโภคา)

อาจารย์ที่ปรึกษา

(ดร. ชม กิ้มปาน)

การวางแผนความต้องการวัสดุ

นาย บุญชัย ชาญภิญโญ
นาย บุณยฤทธิ์ ขันธไชย
นาย วิญญ กิ่งหิรัญวัฒนา

ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร. ชม ก้มปาน อาจารย์ที่ปรึกษา

ปีการศึกษา 2533

บทคัดย่อ

การวางแผนความต้องการวัสดุ คือ เทคนิคการวางแผนความต้องการสินค้าวัตถุดิบ ตลอดจนชิ้นส่วนประกอบต่างๆทั้งหมด ให้สอดคล้องกับความต้องการที่กำหนดในตารางการผลิตหลัก เทคนิคดังกล่าวจำเป็นต้องพึ่งคอมพิวเตอร์ในการคำนวณความต้องการ และเวลาที่ต้องการของสิ่งของคงคลัง เพื่อที่จะได้สั่งซื้อหรือผลิตเพิ่มเติมให้ได้จำนวนที่ต้องการและทันเวลา

Material Requirement Planning

Mr. Boonchai Charnphinyo

Mr. Boonyarit Kanthachai

Mr. Winyu Kinghiranwatana

Dr. Worawat Limphoka Advisor

Dr. Chom Kimparn Advisor

Abstract

MRP, the plan on demand of goods, materials including all other factors, is prepared to be harmonized with demand presented in main preduction schedule. This technic means using computer to calculate demand and time of products demand for the stock in order to keep place order period or additional production period

สารบัญ

บทนำ

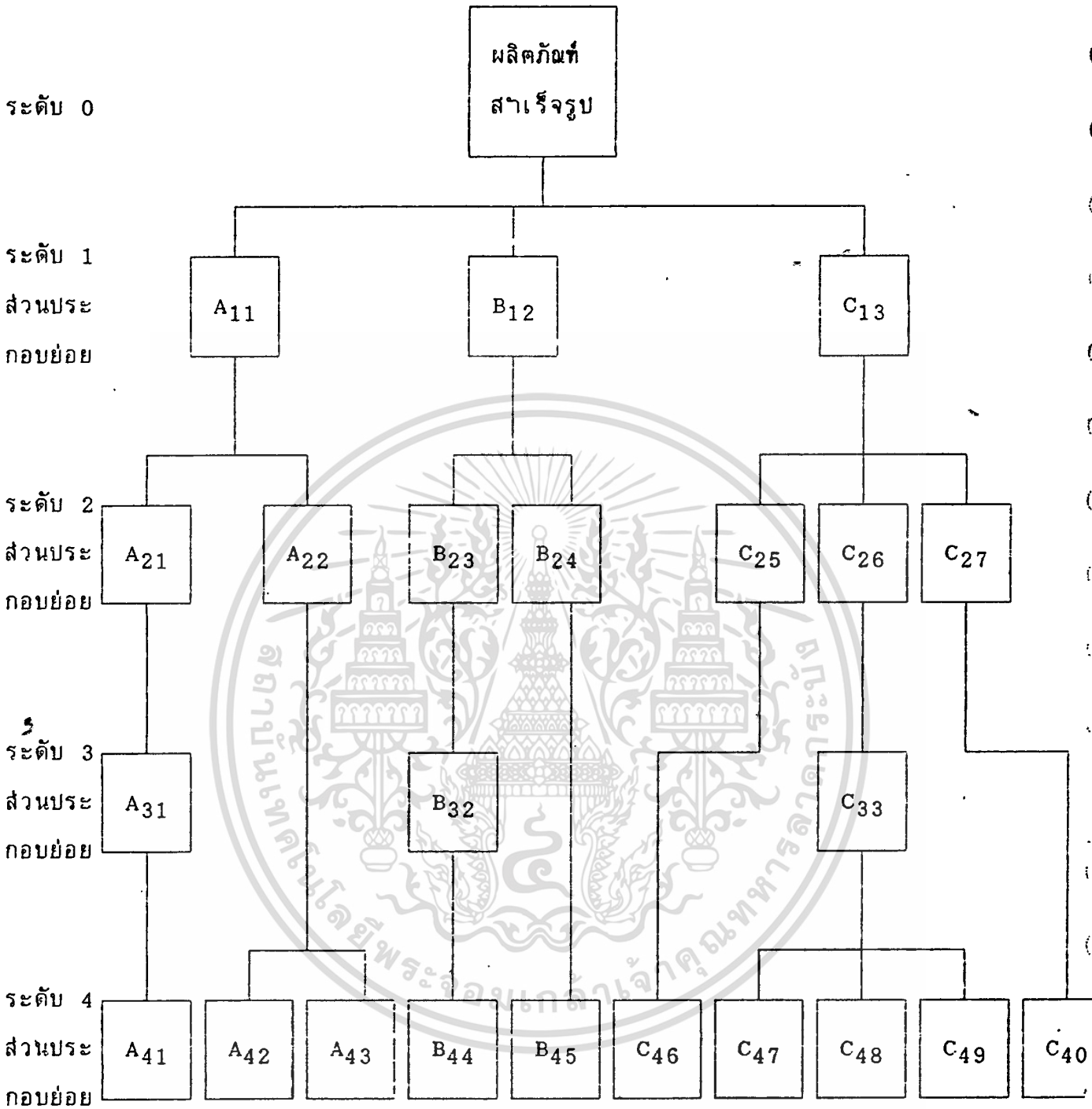
เมื่อเราต้องการจะทำการผลิตผลิตภัณฑ์ชนิดหนึ่งเช่น เครื่องจักร คอมพิวเตอร์ นาฬิกา วิทยุ และอื่น ๆ ปัญหาที่มักจะพบเสมอ ๆ ก็คือจำนวนที่ถูกต้องของชิ้นส่วน ไม่ว่าจะผลิตขึ้นเอง หรือสั่งซื้อจากภายนอก และมีของพร้อมทันเวลากับที่จะใช้ประกอบเป็นผลิตภัณฑ์สำเร็จรูปหรือไม่ สำหรับผลิตภัณฑ์สำเร็จรูปที่ประกอบด้วยส่วนประกอบหลาย ๆ อย่าง และมีขั้นตอนการประกอบเป็นผลิตภัณฑ์สำเร็จรูปซับซ้อนการวางแผนมีความยุ่งยากเกินไปที่จะทำบนกระดาษได้ แต่ปัจจุบันงานดังกล่าวสามารถทำได้ง่ายโดยการใช้คอมพิวเตอร์เข้าช่วย วิธีของ MRP จึงได้เริ่มมีการนำไปใช้ในโรงงานอุตสาหกรรมอย่างแพร่หลาย และมีการพัฒนาให้ก้าวหน้ายิ่งขึ้นไปพร้อม ๆ กับความก้าวหน้าทางเทคโนโลยีคอมพิวเตอร์คอมพิวเตอร์ในปัจจุบันสามารถเก็บข้อมูลได้ปริมาณมากมายมหาศาล การคำนวณก็ทำได้รวดเร็ว และมีความถูกต้องยิ่งขึ้น ขณะเดียวกันค่าใช้จ่ายก็ต่ำลงมาก

จุดมุ่งหมายของวิทยานิพนธ์นี้ เพื่ออธิบายหลักการ ของการวางแผนความต้องการวัสดุ ,ขั้นตอนการสร้างซอฟท์แวร์วางแผนการผลิต ตั้งแต่ การเลือกใช้ tool จนถึง ขั้นการทดสอบระบบ โดยบทที่ 1 จะเป็นทฤษฎีและวิธีทำ MRP , บทที่ 2 แสดงการออกแบบและสร้างทุกส่วนประกอบของระบบ , บทที่ 3 เป็นผลการทำงานของระบบ เมื่อป้อนความต้องการขั้นต้นเข้าไป แล้ววิจารณ์กับสรุปในบทสุดท้าย ส่วนภาคผนวก เป็นการแนะนำให้รู้จักกับ tool ที่นำมาใช้ในการพัฒนาโปรแกรม กับ source code ของ Pro*C สำหรับผู้สนใจ

บทที่ 1

ทฤษฎี

การวางแผนความต้องการวัสดุ เป็นวิธีการคำนวณเพื่อจัดหาวัสดุ (สินค้าสำเร็จรูป ชิ้นส่วนประกอบต่าง ๆ วัตถุดิบ) ให้เพียงพอกับความต้องการทั้งจำนวน และทันต่อเวลาที่มีความต้องการที่เกิดขึ้นในทุก ๆ ระดับของการผลิต หรือกล่าวอีกนัยหนึ่งก็คือ การจัดหาวัสดุให้เพียงพอ และทันเวลากับความต้องการในทุกขั้นตอนของการผลิต จนเป็นสินค้าสำเร็จรูป ดังนั้นเราจึงต้องรู้รายการวัสดุที่ต้องการในการผลิตสินค้าหรือเรียกอีกอย่างหนึ่งว่า โครงสร้างของสินค้าสำเร็จรูปอย่างละเอียด ดังแสดงในรูป



จากรูปจะเห็นว่า สินค้าสำเร็จรูปประกอบด้วยส่วนประกอบย่อย A11 , B12 , C13 ซึ่งถือว่าเป็นส่วนประกอบย่อยขั้นต้น หรือระดับที่ 1 และชิ้นส่วนประกอบย่อยเหล่านี้ยังประกอบด้วยชิ้นส่วนย่อยลงไปอีกเป็นชั้นที่ 2 ที่ 3 ลงไปเรื่อย ๆ จนกระทั่งถึงระดับต่ำสุด ซึ่งเป็นการซื้อชิ้นส่วนหรือวัตถุดิบจากบุคคลภายนอก ชิ้นส่วนประกอบย่อยบาง

รายการใช้มากกว่า 1 หน่วย เพื่อใช้ในการผลิตชิ้นส่วนที่อยู่ในระดับที่สูงกว่า ดังนั้นเราจึงต้องคำนวณการใช้ชิ้นส่วนแต่ละชิ้นให้เพียงพอ นอกจากนี้ชิ้นส่วนประกอบย่อยบางชนิดยังสามารถใช้ในการประกอบผลิตภัณฑ์สำเร็จรูปได้มากกว่าหนึ่งชนิด ดังนั้นความต้องการของชิ้นส่วนที่ใช้ร่วมกันสำหรับผลิตภัณฑ์หลายชนิดนี้จึงต้องสะสมให้เพียงพอกับความต้องการที่ผลิตภัณฑ์แต่ละชนิดจะต้องใช้ทั้งหมด ในการคำนวณหาชิ้นส่วน หรือวัตถุดิบในขั้นตอนการผลิตต่าง ๆ ว่าในช่วงเวลาใดจะต้องจัดหามาจำนวนเท่าไร ควรจะสั่งซื้อหรือสั่งผลิตเมื่อใด ในขั้นแรกของการคำนวณจึงต้องพิจารณาข้อมูลจากตารางการผลิตหลัก (Master Production Schedule) จากนั้นจึงมาพิจารณาโครงสร้างของสินค้าว่า ในแต่ละขั้นตอนใดใช้วัตถุดิบ หรือชิ้นส่วนใดเป็นจำนวนเท่าไร ต่อการผลิตผลิตภัณฑ์ที่กำหนดในตารางการผลิตหลักหนึ่งหน่วย

1.1 ความหมายของคำบางคำที่ใช้ใน MRP (Definitions)

เพื่อให้เข้าใจพื้นฐานของการวางแผนความต้องการวัสดุได้ดียิ่งขึ้น จึงควรที่จะเข้าใจความหมายของคำบางคำที่ใช้ใน MRP ดังนี้

1. ความต้องการขั้นต้น (Gross Requirements) หมายถึงยอดรวมทั้งหมดของความต้องการของของคงคลังแต่ละชนิดในแต่ละช่วงเวลา ความต้องการขั้นต้นของของคงคลังแต่ละชนิดนี้ จะทำให้เราสามารถคำนวณหาจำนวนชิ้นส่วนประกอบ ชิ้นส่วนประกอบย่อย หรือวัตถุดิบ ที่ต้องนำมาใช้ทำเป็นของคงคลังดังกล่าวนี้ และชิ้นส่วนประกอบหน่วยเหล่านั้นก็จะกลายเป็นความต้องการขั้นต้น เพื่อให้หาชิ้นส่วนที่จะมาทำชิ้นส่วนประกอบย่อยอีกทีหนึ่ง และจะเป็นเช่นนี้ไปจนกระทั่งถึงวัตถุดิบหรือชิ้นส่วนที่ต้องสั่งซื้อจากบุคคลภายนอก

2. จำนวนของที่ได้รับตามกำหนดเวลา (Schedul Receipts) หมายถึงจำนวนของทั้งหมดที่เราได้สั่งซื้อหรือสั่งผลิตไปแล้ว และคาดหมายว่าจะได้รับของจำนวนนั้นตามกำหนดเวลาที่วางไว้

3. จำนวนที่มีอยู่ในคลัง (On Hand) หมายถึงจำนวนของคงคลังแต่ละชนิดที่มีอยู่ทั้งหมด ซึ่งได้ทำการตรวจสอบก่อนที่จะเริ่มทำการวางแผนความต้องการวัสดุ ทั้งนี้ก็เพื่อให้การวางแผนมีความถูกต้อง และเกิดประโยชน์ต่อการผลิต

4. จำนวนที่สามารถนำไปใช้ได้ (Available) ในบางครั้งจำนวนของที่มีอยู่ในคลังอาจจะไม่สามารถนำไปใช้ได้ทั้งหมด ทั้งนี้เพราะเราอาจจะต้องเผื่อไว้จำนวนหนึ่ง

เพื่อป้องกันของขาดมือ ซึ่งจะเป็นจำนวนเท่าใดนั้นขึ้นอยู่กับนโยบายของบริษัท จำนวนของที่เผื่อไว้นี้จึงต้องให้มีอยู่ในคลังตลอดเวลา หรือในบางครั้งเราจำเป็นต้องจัดสรรไว้บางส่วนให้กับใบเบิกที่ได้แจ้งไว้แล้วแต่ยังไม่ได้นำของออกจากคลัง เราจึงจำเป็นต้องกันของคงคลังส่วนนี้เอาไว้ จำนวนของคงคลังที่สามารถจะนำไปใช้ได้ จึงเป็นจำนวนที่ได้หักส่วนต่าง ๆ ดังที่ได้กล่าวมาแล้ว แต่ในบางช่วงเวลาจำนวนของที่สามารถจะนำไปใช้ได้ก็อาจจะเพิ่มขึ้นได้ เนื่องจากได้รับของที่ได้สั่งไปก่อนหน้านี้

5. ความต้องการสุทธิ (Net Requirements) สำหรับของคงคลังชนิดใดชนิดหนึ่งที่กำหนดไว้ตามช่วงเวลาใด ๆ ของแผน ความต้องการสุทธิก็คือ จำนวนที่จะต้องทำการสั่งซื้อหรือสั่งผลิต การสั่งซื้อหรือสั่งผลิตจะไม่เกิดขึ้นถ้าจำนวนของคงคลังที่สามารถนำไปใช้ได้ (Available) ในช่วงใด ๆ มีมากกว่าความต้องการขั้นต้นที่มีอยู่ในช่วงเวลานั้น ๆ ในกรณีเช่นนี้ความต้องการสุทธิจึงมีค่าเป็น 0

6. แผนหมายกำหนดการรับของที่สั่ง (Planed Order Receipts) เป็นแผนที่กำหนดว่าของที่ต้องการนั้นจะได้รับในวันใด สำหรับแผนหมายกำหนดการรับของที่สั่งจะถูกใช้อ้างอิงเพื่อวางแผนหมายกำหนดการสั่งของ

7. แผนหมายกำหนดการสั่งของ (Planned Order Releases) เป็นการวางแผนกำหนดเวลาสั่งของ เพื่อจะให้ของที่สั่งไปนั้นได้รับตามแผนหมายกำหนดการรับของแผนหมายกำหนดการสั่งของจะต้องพิจารณาควบคู่ไปพร้อมกันกับแผนหมายกำหนดการรับของ

8. ปริมาณที่จัดสรรไว้ (Allocated Quantities) หมายถึงปริมาณของคงคลังที่จะต้องกันเอาไว้เนื่องจากบัญชีค้างเบิก ในบางครั้งขณะที่ทำการตรวจนับของคงคลังที่มีอยู่ในมือสุทธิ เพื่อวางแผนการสั่งซื้ออาจจะมีใบเบิกของบางรายการที่ได้ทำการเบิกของไว้แล้วแต่ยังไม่ได้นำของออกจากคลัง ทำให้การคำนวณอาจผิดพลาดไปได้ ถ้าไม่นำรายการดังกล่าวมาพิจารณาด้วย ฉะนั้นจำนวนของที่มีอยู่ในมือจะต้องถูกหักด้วยจำนวนที่ต้องจัดสรรไว้

1.2 ธรรมชาติของอุปสงค์ (Nature of Demand)

สำหรับอุปสงค์ที่กล่าวถึงใน MRP มีลักษณะพอที่จะแบ่งให้เห็นธรรมชาติของอุปสงค์ได้ 2 ลักษณะคือ อุปสงค์อิสระและอุปสงค์ตาม (Independent and Dependent Demand) ซึ่งพื้นฐานของอุปสงค์อิสระและอุปสงค์ตาม เป็นหลักการพื้นฐานที่สำคัญของ MRP

โดยลักษณะของอุปสงค์ทั้งสองจะมีความแตกต่างกัน ดังนี้

1. อุปสงค์อิสระ เป็นความต้องการที่มาจากภายนอก ไม่มีความสัมพันธ์กับความต้องการสินค้าหรือของคงคลังชนิดอื่น หรือเป็นอุปสงค์ที่มาจากความต้องการของลูกค้า ตัวอย่างของอุปสงค์อิสระได้แก่ ผลิตภัณฑ์สำเร็จรูปของบริษัท ชิ้นส่วนของอะหลั่ยต่าง ๆ เพื่อไว้บริการลูกค้า (Sevice parts) ชิ้นส่วนอะหลั่ยที่ใช้ในงานซ่อมบำรุง เพื่อเตรียมไว้เปลี่ยนให้กับเครื่องจักรที่เกิดเสียขึ้นมา (Maintenance part use on breakdown) เครื่องมือ อุปกรณ์ต่าง ๆ ที่ใช้ในการผลิต (Production Supplies) ตลอดจนสิ่งของต่าง ๆ ที่ใช้ในสำนักงาน (Office Supplies) สำหรับชิ้นส่วนที่ใช้ในการซ่อมบำรุงป้องกัน (Maintenance part use on PM and Planned repair) เราไม่ถือเป็นอุปสงค์อิสระ แต่จะเป็นอุปสงค์ตามเช่น น้ำมันหล่อลื่น เป็นต้น ในการทำ MRP อุปสงค์อิสระก็คือ ความต้องการที่ปรากฏอยู่ในตารางการผลิตหลัก

2. อุปสงค์ตาม สำหรับอุปกรณ์ที่จะพิจารณาว่าเป็นอุปสงค์ตามนั้น จะต้องมีความสัมพันธ์โดยตรงหรือถูกผลักดันให้เป็นไปตามความต้องการของวัสดุชนิดอื่น ๆ กล่าวอีกอย่างหนึ่งก็คือ ชิ้นส่วนที่ต้องการใช้ในการทำผลิตภัณฑ์ที่เป็นอุปสงค์อิสระเราเรียกว่าอุปสงค์ตาม ดังนั้น ในการคำนวณหาปริมาณความต้องการในอุปสงค์ตาม เราก็คำนวณจากอุปสงค์อิสระผลที่ได้จะทำให้เราทราบว่า จะต้องใช้ส่วนประกอบย่อย (Subassemblies) ชิ้นส่วน (Component parts) และวัตถุดิบในแต่ละขั้นตอนเป็นจำนวนเท่าไร จึงจะทำให้ได้สินค้าสำเร็จรูปตามจำนวนที่ต้องการ โดยในการคำนวณเราจะคำนวณย้อนกลับจากผลิตภัณฑ์ที่เป็นอุปสงค์อิสระ ยกตัวอย่างเช่น โรงงานผลิตเครื่องดูดฝุ่นแห่งหนึ่ง ได้ทำการพยากรณ์ว่าจะสามารถขายเครื่องดูดฝุ่นได้ 1000 เครื่องในช่วงเวลาอนาคตอันใกล้นี้ ค่าดังกล่าวนี้ถือว่าเป็นอุปสงค์อิสระ เพื่อให้ได้จำนวนเครื่องดูดฝุ่น ตามที่ต้องการจะต้องมีอุปสงค์ตามดังนี้ จำนวนล้อ 4000 ล้อ ตัวถังของเครื่อง 1000 ตัว หัวครอบพลาสติกคลุมปลายด้านหน้า 1000 ชิ้น และหัวครอบพลาสติกคลุมปลายด้านหลัง แต่เป็นพลาสติกคนละชนิดกัน จำนวน 1000 ชิ้น และนอกจากนี้ยังจะต้องมีจำนวนน็อต โบลท์ และชิ้นส่วนอื่น ๆ ให้เพียงพอที่จะนำมาประกอบเป็นเครื่องดูดฝุ่น 1000 เครื่อง จะเห็นได้ว่าจำนวนชิ้นส่วนต่าง ๆ ดังที่กล่าวมานี้จะแปรผันไปตามจำนวนเครื่องดูดฝุ่นที่จะต้องผลิต

สำหรับชิ้นส่วนต่าง ๆ ตามที่กล่าวมานี้ เราอาจหามาได้จากการสั่งผลิต หรือสั่งซื้อจากบุคคลภายนอก หรือผลิตขึ้นเอง ถ้าเป็นการสั่งซื้อจากบุคคลภายนอก เราจะต้องพิจารณาถึงจำนวนที่จะต้องสั่งซื้อทั้งหมด ซึ่งก็คือผลรวมของจำนวนที่ต้องใช้เพื่อผลิตเป็น

ผลิตภัณฑ์สำเร็จรูป บวกกับจำนวนที่สั่งไว้เป็นอะหลั่ยคอยบริการลูกค้า แต่ถ้าชิ้นส่วนนั้นเราผลิตขึ้นเอง เช่น ตัวถังของเครื่องดูดฝุ่น เราก็ต้องใช้โลหะแผ่นมาผลิต ซึ่งเราสามารถคำนวณหาจำนวนโลหะแผ่นที่ต้องการได้ เมื่อเรารู้ว่าแต่ละตัวถังของเครื่องดูดฝุ่นต้องใช้โลหะแผ่นเท่าไร ในทำนองเดียวกัน ถ้าเราผลิตหัวครอบพลาสติกขึ้นเอง อุปสงค์ตามก็คือ จำนวนของเม็ดพลาสติกซึ่งเป็นวัตถุดิบที่จะต้องคำนวณหา

1.3 วัสดุใช้ร่วม (Common Use Items)

ในสภาพการณ์ของอุตสาหกรรมการผลิตโดยทั่ว ๆ ไป มักจะมีชิ้นส่วนบางอย่างและสินค้าหลาย ๆ ชนิดที่ใช้วัตถุดิบชนิดเดียวกัน ในการผลิตให้เป็นผลิตภัณฑ์ หรือสินค้าที่ต้องการ ดังนั้นผลรวมของความต้องการของวัตถุดิบดังกล่าวก็คือ การบวกความต้องการที่เกิดจากแหล่งต่าง ๆ ความต้องการจากหลาย ๆ แหล่งที่จะใช้วัสดุร่วมกันนั้น ทั่ว ๆ ไป จะถูกรวบรวมเพื่อทำการสั่งซื้อหรือสั่งผลิตเพียงครั้งเดียว ไม่แยกกันทั้งนี้เพื่อให้ประหยัดค่าใช้จ่ายในการสั่งซื้อหรือสั่งผลิต (Save on ordering and setup costs) ในขั้นตอนของ MRP จะทำการรวบรวมความต้องการเหล่านี้ เพื่อหาความต้องการสุทธิของแต่ละวัสดุ ยกตัวอย่างเช่น เหล็กเส้นชนิดหนึ่งอาจผลิตเป็นสกรูได้หลายชนิด และสกรูเหล่านี้แต่ละชนิดก็อาจจะใช้ในการประกอบสินค้าสำเร็จรูปได้หลายแบบหลายอย่างเช่นเดียวกัน ซึ่ง MRP จะรวบรวมรายการวัสดุที่ใช้ร่วมกันเหล่านี้ สำหรับการสั่งซื้อวัตถุดิบหรือสำหรับการผลิตชิ้นส่วนต่าง ๆ แต่ละครั้งให้มีประสิทธิภาพและประหยัดค่าใช้จ่าย

1.4 ช่วงเวลานำ (Lead Times)

ช่วงเวลานำก็คือเวลาที่ใช้สำหรับทำงานอย่างใดอย่างหนึ่ง สำหรับวัสดุที่เราใช้ในช่วงเวลานำก็คือ เวลาที่ใช้ทำงานตั้งแต่การเตรียมงานที่จำเป็นบนกระดาษ บวกเวลาที่ใช้ในการเตรียมการปฏิบัติงาน สำหรับวัสดุที่สั่งซื้อจากภายนอกหรือวัตถุดิบ ช่วงเวลานำก็คือ เวลาตั้งแต่ออกใบสั่งซื้อจนกระทั่งได้รับสินค้าที่สั่ง

สำหรับ MRP แล้วช่วงเวลานำดังกล่าวจะมีความสำคัญมาก เพราะจะถูกนำไปใช้สำหรับพิจารณาหาเวลาที่ควรเริ่มทำการประกอบชิ้นส่วน วันเริ่มต้นของการผลิตชิ้นส่วนและสำหรับกำหนดวันสั่งซื้อวัตถุดิบ

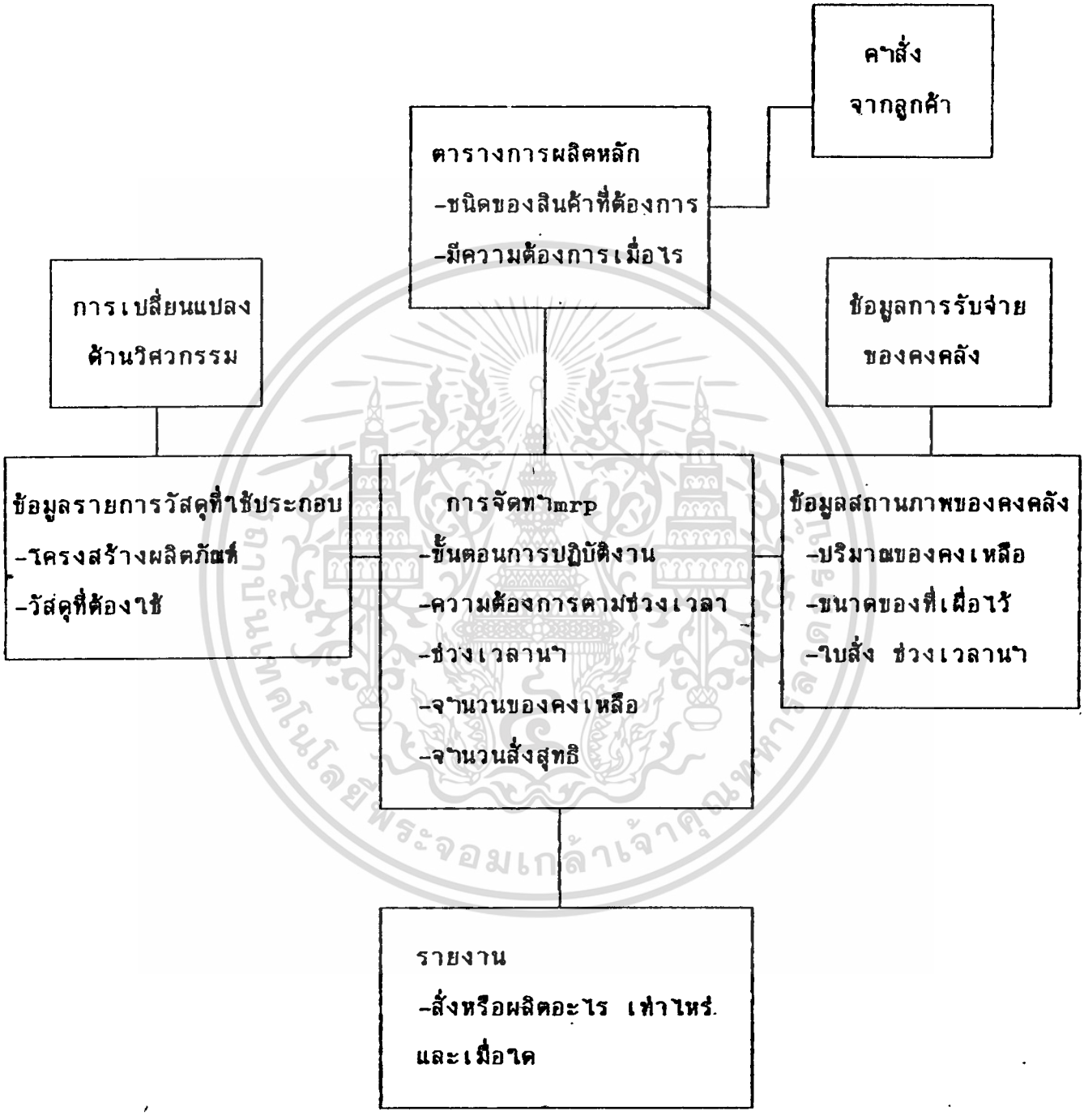
1.5 เครื่องมือหรือข้อมูลที่จำเป็นสำหรับ MRP

เครื่องมือหรือข้อมูลที่จำเป็นสำหรับ MRP มีอยู่ 3 ประการคือ

1.ตารางการผลิตหลัก(Master Production Scheduling)

2.ข้อมูลการเปลี่ยนแปลงด้านวิศวกรรม(Engineering Changes)

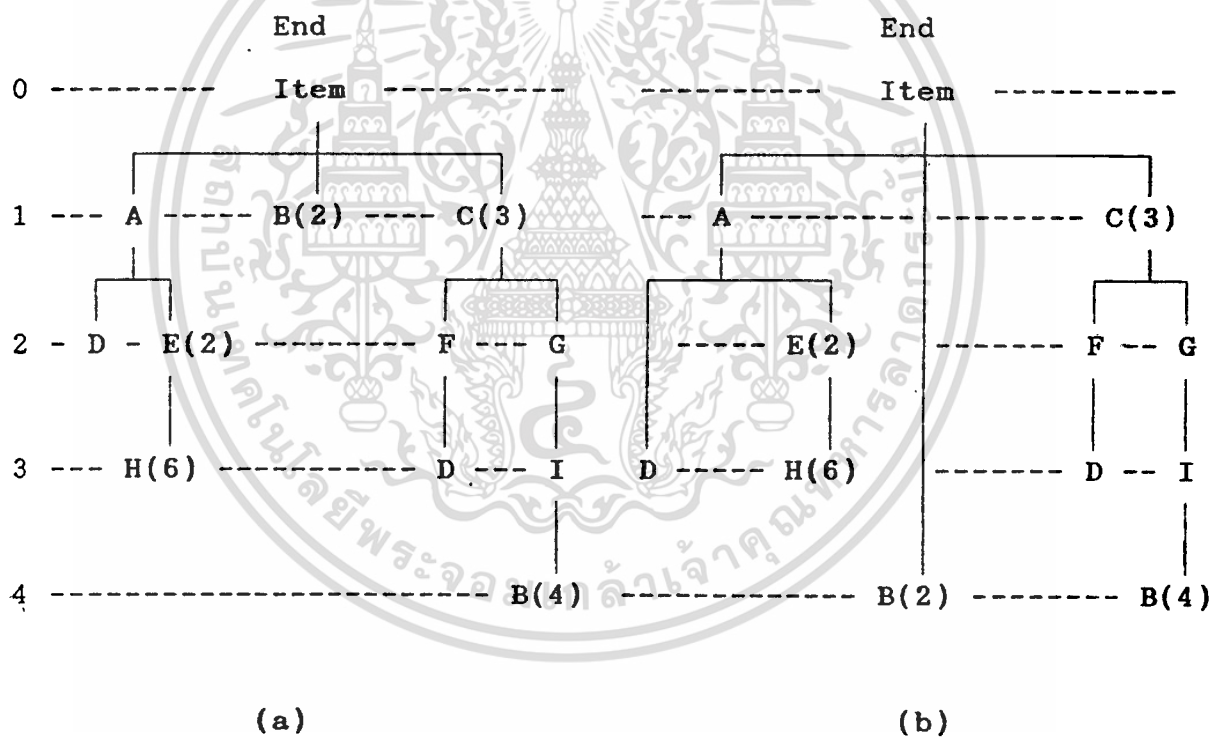
3.ข้อมูลรับจ่ายของคงคลัง(Inventory Transaction Data)



ภาพเครื่องมือหรือข้อมูลที่จำเป็นสำหรับ MRP

ก.ตารางการผลิตหลัก เป็นตารางที่แสดงให้ทราบว่ามีสินค้าชนิดใดบ้างที่ต้องทำการผลิต จำนวนผลิตสินค้าแต่ละชนิดเป็นเท่าไร และเวลาที่ต้องการสำหรับสินค้าแต่ละชนิดคือช่วงใด สินค้าที่บรรจุในตารางการผลิตหลักเป็นสินค้าสำเร็จรูปที่บริษัทจำหน่ายให้แก่ลูกค้า ดังนั้นจึงจัดอยู่ในพวกอุปสงค์อิสระ

ข.ข้อมูลการเปลี่ยนแปลงด้านวิศวกรรม คือข้อมูลแสดงถึงรายการวัสดุหรือโครงสร้างของผลิตภัณฑ์ ตลอดจนเครื่องมือและอุปกรณ์ต่างๆที่ใช้ในการผลิต ในการที่จะหารายการวัสดุของสินค้าชนิดใดชนิดหนึ่ง เราจำเป็นต้องรู้โครงสร้างของสินค้านั้นก่อนว่ามีส่วนประกอบของวัสดุอะไรบ้าง ต้องใช้ในขั้นตอนใดหรือลำดับใดและต้องใช้เวลาเท่าไรในแต่ละขั้นตอน เพื่อจะได้วางแผนการสั่งวัสดุได้ถูกต้องทั้งชนิด จำนวนและเวลา



(a) (b)

ในแต่ละรายการวัสดุหรือโครงสร้างสินค้าจะมีการกำหนดรหัสของวัสดุแต่ละชนิดตามขั้นตอนการผลิตสินค้า โดยเริ่มจากผลิตภัณฑ์ขั้นสุดท้าย ไปยังส่วนประกอบและชิ้นส่วนจนถึงวัตถุดิบตามลำดับ ซึ่งจะเริ่มต้นจาก 0,1,2... ไปเรื่อยๆ

จากรูป a แสดงการให้ระดับตามขั้นตอนที่แสดงในรายการวัสดุ หรือโครงสร้างของสินค้า ซึ่งคอมพิวเตอร์ต้องรู้หมายเลขระดับของชิ้นส่วน หรือส่วนประกอบทุกๆชิ้น เพื่อ

สามารถพิจารณาย้อนกลับตามช่วงเวลานำ จากวันกำหนดเสร็จของผลิตภัณฑ์ขั้นสุดท้ายไปยังวันที่ควรจะเริ่มผลิตชิ้นส่วน ซึ่งสามารถทำให้เราทำตารางหมายกำหนดการทำชิ้นส่วนต่างๆ ได้

จากรูป (b) เป็นการจัดโครงสร้างของสินค้าตามรหัสต่ำจะช่วยทำให้ขั้นตอนการพิจารณาหาความ ต้องการสุทธิของวัสดุและการแตกกระจายวัสดุไปสู่ระดับต่ำกว่าไม่สับสนโดยจะเริ่มพิจารณาจากวัสดุที่มีรหัสระดับต่ำ ที่สูงให้หมดก่อนจึงเริ่มพิจารณาในระดับต่ำรองลงมา

<u>ค. ข้อมูลรับ-จ่ายของคงคลัง</u> ส่วนที่เป็นข้อมูลรับจ่ายของคงคลังหรือวัสดุชนิดใดๆ จะช่วยทำให้เราทราบสถานภาพของของคงคลังหรือวัสดุชิ้นนั้น ๆ ได้อย่างถูกต้อง และเป็นปัจจุบัน ในส่วนที่แสดงสถานภาพของวัสดุใดก็ตามจะมีรายละเอียดของชิ้นส่วนนั้นๆปรากฏอยู่ เช่น เลขที่ชิ้นส่วน ช่วงเวลานำ ต้นทุนมาตรฐาน ของที่กำหนดให้เผื่อไว้ ของคงคลังที่มีอยู่ ขนาดของการสั่งซื้อ เวลาเตรียมการผลิต ช่วงเวลาการผลิต และข้อมูลด้านสถิติเกี่ยวกับการใช้ในปีที่แล้ว

จากเครื่องมือหรือข้อมูลทั้ง 3 ในการทำ MRP จะเห็นได้ว่าเราจะต้องปรับปรุงข้อมูลที่มีอยู่ให้ทันสมัยและถูกต้องอยู่เสมอเพื่อให้เป็นเครื่องมือในการทำ MRP ที่มีประสิทธิภาพ เช่น ในรายการวัสดุที่เป็นโครงสร้างสินค้า จะต้องทำการปรับปรุงให้ถูกต้องกับที่ทางฝ่ายวิศวกรรมได้ทำการเปลี่ยนแปลงแบบหรือขั้นตอนไปจากเดิม และโดยเฉพาะอย่างยิ่งสำหรับข้อมูลความต้องการในตารางการผลิตหลัก จะมีผลต่อฝ่ายควบคุมการผลิตเป็นอย่างมาก ในการตัดสินว่าจะผลิตอะไรในช่วงเวลาใด ทั้งนี้เพราะความต้องการที่เกิดขึ้นมีความผันแปรตลอดเวลา จึงทำให้ตัวเลขในตารางการผลิตหลักต้องเปลี่ยนไปด้วย ซึ่งการเปลี่ยนแปลงที่เกิดขึ้นต้องส่งข้อมูลให้ฝ่ายจัดทำ MRP ทราบอย่างทันเวลาเพื่อสถานภาพการผลิตดำเนินไปอย่างถูกต้องสอดคล้องกับความต้องการที่เป็นจริงมากที่สุด

1.6 MRP ทำงานอย่างไร

การทำ MRP เริ่มจากตารางการผลิตหลัก ซึ่งจะกำหนดยอดความต้องการผลิตภัณฑ์สำเร็จรูปในช่วงเวลาต่างๆให้ทราบ เพื่อที่จะได้ให้ฝ่ายควบคุมการผลิตตัดสินใจได้ว่าจะผลิตอะไรในแต่ละช่วงเวลา ต่อจากนั้นก็มาพิจารณาว่าในการผลิตสินค้าสำเร็จรูปแต่ละช่วงเวลานั้นต้องมีลำดับขั้นตอนอย่างไร และกำหนดขั้นตอนการทำงานในแต่ละช่วงเวลา เพื่อจะรู้ว่าจะทำงานอะไรในช่วงเวลาใด ซึ่งข้อมูลที่ใช้ในช่วงนี้คือ ข้อมูลโครง

สร้างสินค้า ข้อมูลดังกล่าวสามารถทำให้เราคำนวณได้ว่าต้องใช้วัตถุดิบหรือชิ้นส่วนจำนวนเท่าไร สำหรับการผลิตสินค้าแต่ละชนิดต้องมีการเบิกจ่ายวัสดุและรับวัสดุเข้าออกจากคลังเพื่อให้วัสดุทุกชนิดมีอย่างเพียงพอและทันเวลา จึงต้องอาศัยข้อมูลที่แสดงสถานภาพของการผลิตช่วยในการตัดสินใจว่า จะสั่งซื้อ สั่งผลิตวัสดุชนิดใด ในช่วงเวลาใด เป็นจำนวนเท่าใดเป็นต้น

บทที่ 2

การออกแบบ และ การสร้าง

2.1 MRP PROCESSING MODULE

เนื่องจากโปรแกรมการวางแผนความต้องการวัสดุ เป็นงานที่ยุ่งยากซับซ้อน แรกๆ การเขียนโปรแกรมจะมุ่งประเด็นที่ส่วนการคำนวณการวางแผนอย่างเดียว จากนั้นจึงจะพยายามแยกโปรแกรมเป็นฟังก์ชั่นๆ โดยมีการผ่านค่าพารามิเตอร์ระหว่างฟังก์ชั่น ซึ่งทำให้โปรแกรมง่ายแก่การตรวจสอบและแก้ไข

ฟังก์ชั่นต่างๆมีการทำงานพอสรุปได้ดังนี้

mrp(current time)

อธิบาย เป็นขบวนการทำงานหลักของส่วนการคำนวณความต้องการวัสดุ

พารามิเตอร์ current_time แสดงสัปดาห์ปัจจุบัน

leadtime check (current time)

อธิบาย ตรวจสอบว่าเวลาที่ต้องการผลิตภัณฑ์เป็นไปได้หรือไม่

พารามิเตอร์ current_time แสดงสัปดาห์ปัจจุบัน

check time (current time)

อธิบาย ตรวจสอบว่าเวลาที่ต้องการน้อยกว่าเวลาปัจจุบันหรือไม่

พารามิเตอร์ current_time

enough time (current time)

อธิบาย ตรวจสอบว่าเวลาที่ใช้ในการผลิตทั้งหมด น้อยกว่า ช่วงเวลานำสูงสุดหรือไม่ ถ้าน้อยกว่าแสดงว่าเวลาที่ใช้ผลิตไม่เพียงพอ

พารามิเตอร์ current_time

cal maxleadtime ()

อธิบาย ทำการคำนวณช่วงเวลานำสูงสุด โดยรวมเวลานำของแต่ละกิ่ง แล้วหากิ่งที่มีเวลานำสูงสุด

พารามิเตอร์ ไม่มีพารามิเตอร์

sum (time temp,count,max leadtime)

อธิบาย รวมเวลานำของแต่ละกิ่งทำงานร่วมการฟังก์ชั่น cal_maxleadtime

พารามิเตอร์ time_temp เป็นอาร์เรย์เก็บชื่อผลิตภัณฑ์แต่ละกิ่ง

count จำนวนผลิตภัณฑ์ในแต่ละกิ่ง

max_leadtime เก็บเวลานำสูงสุดของผลิตภัณฑ์ที่ต้องการ

product total (buf total,num,max) , connect tree(buf total,num,max)

อธิบาย หาองค์ประกอบของผลิตภัณฑ์ที่ต้องการทราบทำงานร่วมกับฟังก์ชั่น connect_tree

พารามิเตอร์ buf_total เก็บรายชื่อและระดับขององค์ประกอบทั้งหมดของผลิตภัณฑ์ที่ต้องการทราบ

num จำนวนองค์ประกอบทั้งหมดของผลิตภัณฑ์

max ระดับสูงสุดขององค์ประกอบของผลิตภัณฑ์

product level ()

อธิบาย คำนวณหาระดับต่ำของวัสดุทุกตัว ข้อมูลที่ได้เก็บใส่ตาราง

พารามิเตอร์ ไม่มีพารามิเตอร์

sort by level(min,max,buf level,count,list name,ind)

อธิบาย หาระดับต่ำของวัสดุทุกตัวโดยดูจากข้อมูลในตารางเก็บระดับต่ำที่หามาจากฟังก์ชั่น product_level แล้วทำการจัดเรียงวัสดุตามระดับต่ำของวัสดุ ข้อมูลดังกล่าวจะนำมาใช้ในการคำนวณหาความต้องการวัสดุ เพราะจะต้องทำการคำนวณเป็นระดับๆไป

พารามิเตอร์ min ระดับต่ำสุดของวัสดุที่นำมาคำนวณหาความต้องการวัสดุ

max ระดับสูงสุดของวัสดุที่นำมาคำนวณหาความต้องการวัสดุ

buf_level เป็นอาร์เรย์เก็บชื่อวัสดุเรียงตามระดับต่ำของวัสดุโครงสร้างข้อมูล
เป็นอาร์เรย์ 2 มิติ มิติแรก แสดงระดับ มิติสอง แสดงชื่อผลิตภัณฑ์
เช่น ที่ระดับ 0 มีผลิตภัณฑ์ A,B,C,D
ดังนั้น buf_total[0][0] = A

count แสดงจำนวนวัสดุในแต่ละระดับว่ามีจำนวนเท่าไหร่

list_name รายชื่อวัสดุทั้งหมดที่จะนำมาคำนวณหาความต้องการวัสดุ

ind จำนวนวัสดุทั้งหมด

level total (buf total,min,max,num)

อธิบาย เป็นฟังก์ชั่นทำงานร่วมกับ sort_by_level ทำหน้าที่หาระดับต่ำของวัสดุทุกตัวโดยดูจากข้อมูลในตารางเก็บระดับต่ำ

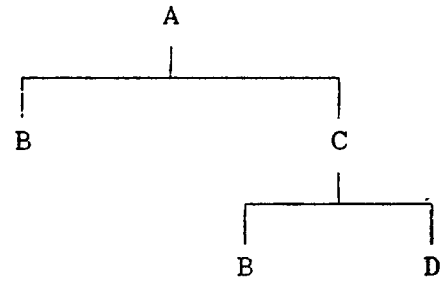
พารามิเตอร์ num คือ จำนวนองค์ประกอบของผลิตภัณฑ์เป็นค่าที่ได้จากการเรียกฟังก์ชั่น product total
พารามิเตอร์อื่นๆดูจากฟังก์ชั่น sort_by_level

calculate qualtity (product,temp,seq,list name)

อธิบาย คำนวณและวางแผนความต้องการวัสดุ โดยใช้ข้อมูลจาก mps,receive schedule และสถานภาพในสินค้าคงคลัง มาคำนวณหา ความต้องการสุทธิ , วันสั่งของ

พารามิเตอร์ product เป็นที่เก็บข้อมูลที่ได้จากการคำนวณการวางแผนความต้องการวัสดุ
โครงสร้างข้อมูลของ product เป็นอาร์เรย์ขนาด 2 มิติ มิติแรก
มี 6 ตัวมีความหมายดังนี้
0 ความต้องการขั้นต้น
1 ของที่ได้รับตามกำหนดเวลา
2 ของที่สามารถนำมาใช้ได้จริง
3 ความต้องการสุทธิ
4 แผนหมายกำหนดการรับของ
5 แผนหมายกำหนดการสั่งของ
มิติที่2 แสดงสัปดาห์ เช่น product[0][5] = 2 หมายความว่า สัปดาห์ที่ 5 มีความต้องการขั้นต้น 2 หน่วย เป็นต้น

temp เป็นที่เก็บองค์ประกอบของผลิตภัณฑ์ เนื่องจากวัสดุตัวหนึ่งอาจเป็นองค์ประกอบของผลิตภัณฑ์หลายตัว ยกตัวอย่างเช่น



วัสดุ B เป็นองค์ประกอบของ A,C การคำนวณหาความต้องการวัสดุจะทำเป็นระดับ ดังนั้น เมื่อหาจำนวนวัสดุ B ที่เป็นองค์ประกอบของ A ได้แล้วต้องเก็บใส่ buffer เพื่อว่าเมื่อถึงระดับของมันข้อมูลดังกล่าวจะถูกนำมาใช้โดยไม่สูญหาย buffer นี้จะมีโครงสร้างข้อมูลเป็น structure ที่มีฟิลด์คือ date และ qualtity ถ้าวันที่ต้องการของวัสดุ B มีอยู่แล้วใน buffer ฟิลด์quantity จะถูกบวกซ้ำเท่านั้น ข้อมูลใน temp จะเป็นความต้องการขั้นต้นขององค์ประกอบนั้นๆ

seq เป็นจำนวนคู่ใน temp ฟิลด์ date และ ฟิลด์ qualtity ถือว่าเป็น 1 คู่ โครงสร้างข้อมูลเป็นอาร์เรย์มิติเดียว เพราะผลิตภัณฑ์ทุกตัวจะมี seq เป็นของตัวเอง อาร์เรย์เป็นตัวแสดงว่าเป็นของผลิตภัณฑ์ไหนโดยดูคู่กับ list_name

เช่น list_name = [A,B,C,D] seq[0]

หมายถึง seq ของ A เป็นต้น

list_name เป็นรายชื่อของวัสดุทั้งหมดที่ทำการวางแผนความต้องการวัสดุ

prod inventory()

อธิบาย ปริมาณของวัสดุว่ามีในคลังสินค้าที่สามารถนำมาใช้ได้เท่าไหร่

พารามิเตอร์ ไม่มีพารามิเตอร์

product receive (product)

อธิบาย วัสดุที่ได้รับตามกำหนดเวลาว่ามีในวันไหน จำนวนเท่าไหร่

พารามิเตอร์ product ดูรายละเอียดจากฟังก์ชัน calculate_quantity

save in buf(seq,temp,list name)

อธิบาย หาปริมาณองค์ประกอบของผลิตภัณฑ์และเก็บค่าที่หาได้ลง bufferชื่อ temp

พารามิเตอร์ ดูรายละเอียดค่าพารามิเตอร์จากฟังก์ชั่น calculate_quantity

gotoxy(x,y)

อธิบาย ทำการเลื่อนเคอร์เซอร์ไปยังตำแหน่ง x,y

พารามิเตอร์ x,y ค่าโคออร์ดิเนต

clrscr()

อธิบาย เป็นการ clear จอภาพ

พารามิเตอร์ ไม่มีพารามิเตอร์

report (current time,num of product)

อธิบาย แสดงผลมีให้เลือกว่าให้ออกเครื่องพิมพ์หรือจอภาพ

พารามีเตอร์ current_time วันปัจจุบัน

num_of_product จำนวนวัสดุทั้งหมดที่ทำการวางแผนความต้องการวัสดุ

to printer(current time,num of product)

อธิบาย แสดงผลออกเครื่องพิมพ์

พารามิเตอร์ พารามิเตอร์ดูรายละเอียดที่ฟังก์ชั่น report

to moniter (current time,num of product)

อธิบาย แสดงผลออกจอภาพ

พารามิเตอร์ พารามิเตอร์ดูรายละเอียดที่ฟังก์ชั่น report

read file (product,fp1,max d,str)

อธิบาย อ่านข้อมูลจากแฟ้มข้อมูล

พารามิเตอร์ product ดูรายละเอียดที่ฟังก์ชั่น calculate_quantity

fp1 ไฟล์พอย์เตอร์

max_d เป็นสัปดาห์สูงสุด เป็นการควบคุมว่าพิมพ์สัปดาห์สูงสุดที่สัปดาห์ไหน

str ตัวแปรเก็บว่าผลิตภัณฑ์อะไร

write file(fp1,product)

อธิบาย บันทึกข้อมูลที่ได้จากการหาความต้องการลงแฟ้มข้อมูล

พารามิเตอร์ fp1 ไฟล์พอยเตอร์

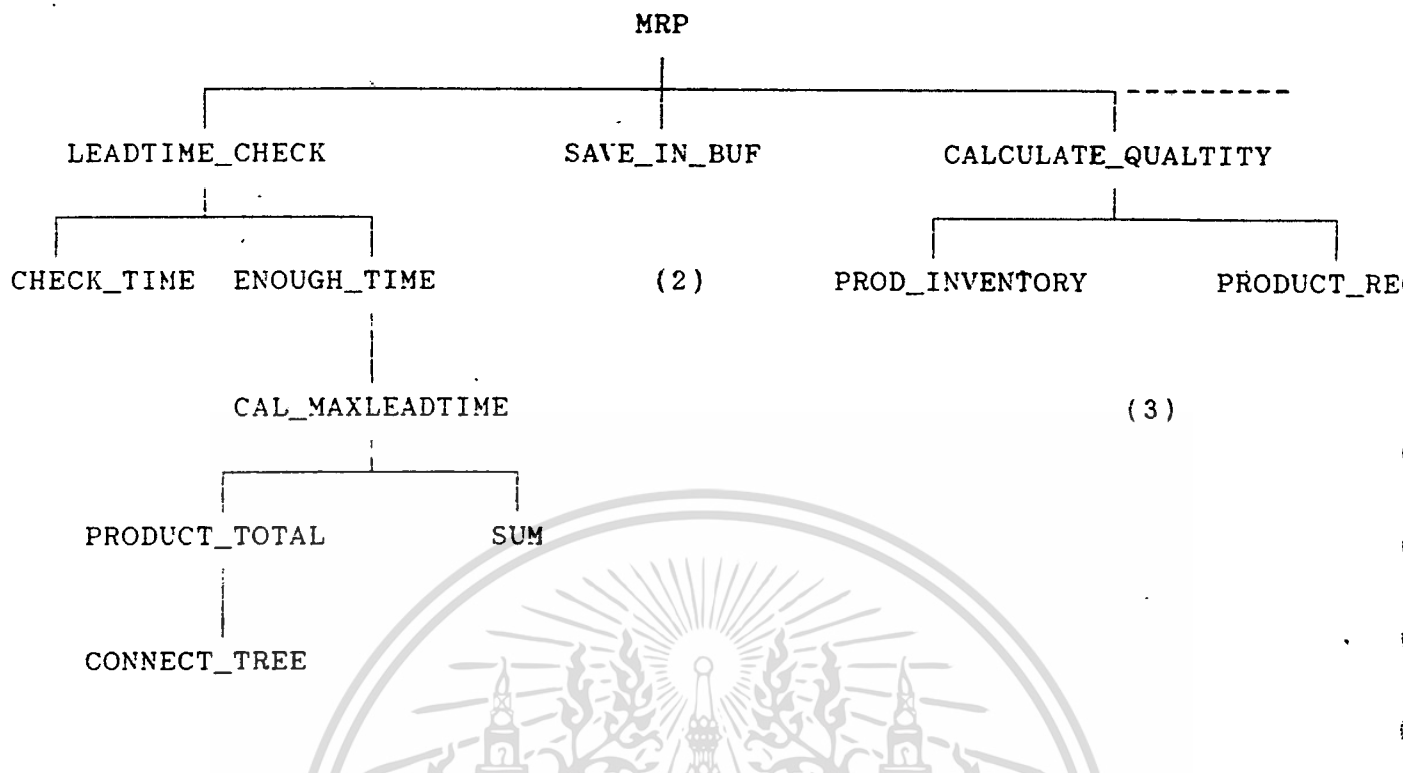
product ดูรายละเอียดที่ฟังก์ชั่น calculate_qualtity

inquiry()

อธิบาย ดูโครงสร้างของผลิตภัณฑ์

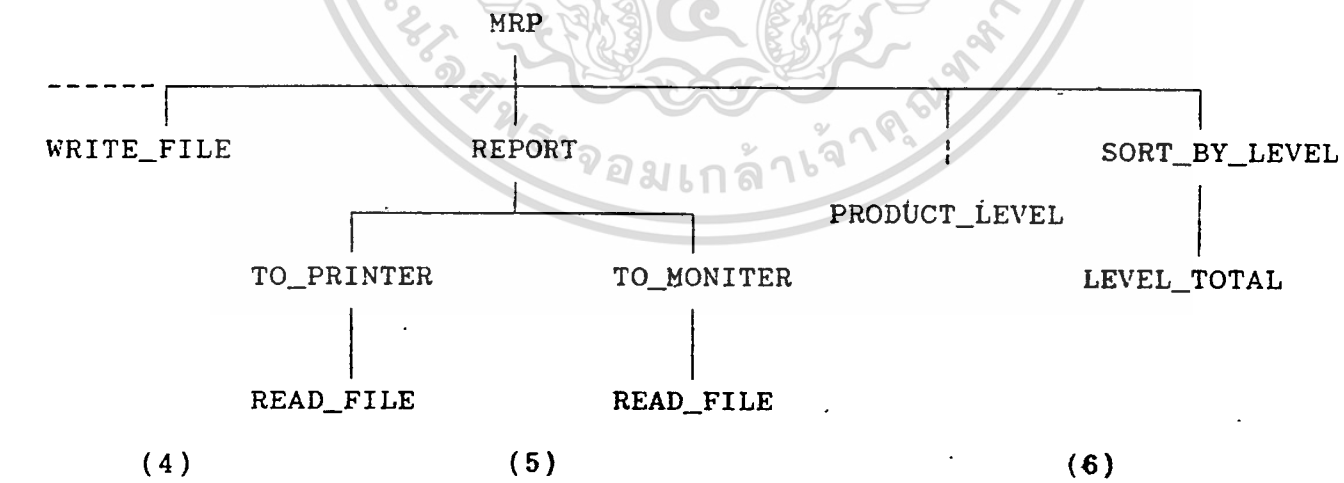
พารามิเตอร์ ไม่มีพารามิเตอร์

ลักษณะการเรียกใช้ฟังก์ชั่นของส่วนการคำนวณ MRP เป็นดังรูป



ส่วนที่ (1) อยู่ในไฟล์ชื่อ leadt.pc

ส่วนที่ (2)และ(3) อยู่ในไฟล์ชื่อ math.pc



ส่วนที่ (4)และส่วนที่ (5) อยู่ในไฟล์ชื่อ rep.pc

ส่วนที่ (6) อยู่ในไฟล์ชื่อ level.pc

mrp() คือ โปรแกรมส่วนที่ทำหน้าที่คิดคำนวณแผนการผลิต โดยมี INPUT คือ ข้อมูลรายละเอียดของผลิตภัณฑ์ในฐานข้อมูล และ OUTPUT คือแผนการสั่งซื้อ และ สั่งผลิต , อธิบายด้วย PSUEDO CODE ในหน้าถัดไป

mrp() เริ่มเช็ค leadtime ว่าเพียงพอแก่การผลิตหรือไม่ คือ เวลาในการผลิตอย่างน้อยต้องเท่ากับ leadtime สูงสุดของผลิตภัณฑ์นั้น หากเวลาน้อยกว่าแสดงว่า ผลิตไม่ได้ นอกจากนี้ยังเช็คอีกว่า วันที่ต้องการผลิตภัณฑ์น้อยกว่า current time หรือไม่ ถ้าน้อยกว่า จะแสดง error การเช็คดูได้จากค่า possible คือ possible เป็น 0 ผลิตได้ , เป็น 1 แสดงว่าที่ MPS มีผลิตภัณฑ์บางตัวที่วันที่ต้องการน้อยกว่าเวลาปัจจุบัน ซึ่งเป็นไปไม่ได้ , ถ้า possible = 2 แสดงว่า ให้เวลาไม่พอแก่การผลิต โปรแกรมจะแสดง max leadtime ของผลิตภัณฑ์นั้น

ในกรณี possible เป็น 0 คือ ผลิตได้ ผลิตภัณฑ์ทุกตัวที่จะถูกนำมาวางแผน จะถูกนำมาเทียบกับตาราง low level ว่ามันมี level ประจำตัวมันเท่าไหร่ ซึ่ง routine product_planning_level จะทำหน้าที่นี้ และ ให้ค่า min,max ออกมา min คือ level ต่ำสุด , max คือ level สูงสุด ที่จะใช้ในการวางแผน จาก level ที่ได้ทั้งหมด level ต่ำสุดจะถูกนำมาคำนวณหาปริมาณ การคำนวณทำเป็นระดับเพื่อแก้ปัญหาการใช้ของใน inventory กล่าวคือ ของใน inventory จะถูกหักเริ่มจากวันที่ต้องการใช้ต่ำสุดก่อน การคำนวณเป็นระดับนี้ เมื่อถึงระดับประจำตัวมันปริมาณความต้องการขั้นต้น จะถูกหาไว้ทั้งหมด ซึ่งจะสามารถแก้ปัญหานี้ได้ จากนั้นเอาข้อมูลใน temp ทั้งหมดมาคำนวณหาความต้องการสุทธิ ,วันที่สั่งของ ,วันที่รับของ routine calculate_quality จะทำดังที่กล่าวไว้ทั้งหมด ตัวแปร product ใน argument นั้น เก็บค่าที่คำนวณไว้ก่อนที่จะนำมาเก็บใส่ file ใน routine write_file จากนั้นจะหาว่า product นั้นมีลูกเป็นอะไร ปริมาณเท่าไร ข้อมูลเหล่านี้จะถูกเก็บใส่ buffer โดยองค์ประกอบย่อยทุกตัว จะมี buffer ประจำเก็บปริมาณ และ วันที่

0	1	2	3	4	- - - - - - - - - - - - - - -	50	51	52
1	4	7	2	6	- - - - - - - - - - - - - - -	6	4	9

MRP Processing Module

```
{
 possible = check_leadtime(current_time)
 if (possible ==0)
  {
    product_planning_level(min,max,buf_level)
    openfile(fp1)
    for (level=min;level<=max,level++)
     {
      name = name_product(level,buf_level)
      while (not end)
       {
         calculate_quality(name,temp,product)
         write_file(fp1,product)
         save_chile_in_buf(temp,name)
       }
     }
    closefile(fp1)
    report(current_time)
  }
 else
  if (possible == 1)
    printf("\n ERROR TIME INCORRECT , LESS THAN CURRENT TIME")
  else if (possible == 2)
     {
      printf("\n ERROR , REQUIRE TIME AT LEAST ITS MAXIMUM LEADTIME")
      printf("PRODUCT %s HAS LEADTIME = %d",name,max_leadtime)
     }
}
```

ถ้าวันไหนมีปริมาณอยู่แล้ว ปริมาณใหม่จะถูกบวกเพิ่มเข้าไป การทำงานจะวน loop จนหมดทุกระดับทุกผลิตภัณฑ์ จากนั้น เป็นการแสดงผล โดยมีให้เลือกว่า จะออก printer หรือ monitor การแสดงผลเริ่มจาก current time ที่ป้อนเข้ามา แสดงผลทาง printer ครั้งละ 20 คอลัมน์ หรือทาง monitor ครั้งละ 7 คอลัมน์

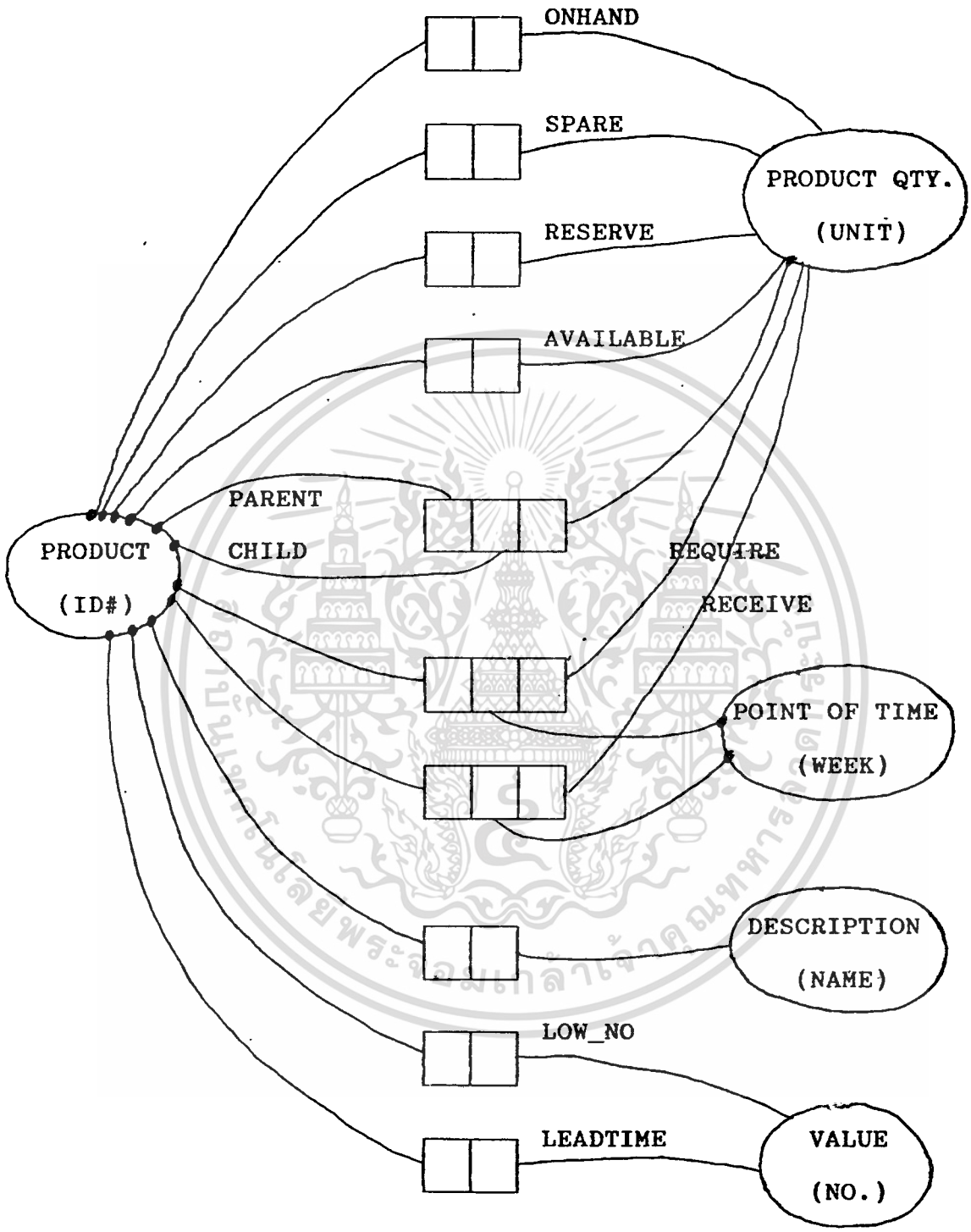
2.2 ฐานข้อมูล

สำหรับระบบฐานข้อมูล เลือกใช้ DATABASE MANAGEMENT SYSTEM (DBMS) เพราะว่า DBMS เป็นซอฟท์แวร์ที่ช่วยจัดการฐานข้อมูล เกี่ยวกับการเก็บและดึงข้อมูล โดยที่มีฟังก์ชั่นการทำงานต่างๆพร้อมอยู่แล้ว และสามารถเรียกใช้ได้โดยคำสั่งของ DBMS เอง ซึ่งเป็นการสะดวกกว่าวิธีการ MAINTENANCE วิธีอื่นๆ อีกประการหนึ่งคือ ระบบ DBMS นี้ สามารถทำงานบนเครื่องหลายระดับ เช่น เมนเฟรม, มินิคอมพิวเตอร์, จนถึงระดับ ไมโครคอมพิวเตอร์ ซึ่งกำลังเป็นที่สนใจ และ เริ่มมีใช้กันอย่างแพร่หลาย

และสาเหตุที่เลือกใช้ RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) คือ ORACLE นั้น มาจากจุดดีต่างๆ ดังสรุปเป็นข้อๆ ต่อไปนี้

- ลดความซ้ำซ้อนของข้อมูล (Redundancy can be reduced)
- สามารถหลีกเลี่ยงความไม่ตรงกันของข้อมูล (Inconsistency can be avoided)
- สามารถใช้ข้อมูลร่วมกันได้ (The data can be shared)
- สามารถควบคุมความเป็นมาตรฐานได้ (Standard can be enforced)
- สามารถจัดระบบความปลอดภัยที่รัดกุมได้ (Security restrictions can be applied)
- สามารถควบคุมความถูกต้องของข้อมูลได้ (Integrity can be maintained)
- สามารถสร้างสมดุลในความขัดแย้งของความต้องการได้ (Requirement can be balanced)
- เกิดความอิสระของข้อมูล (Data independence)

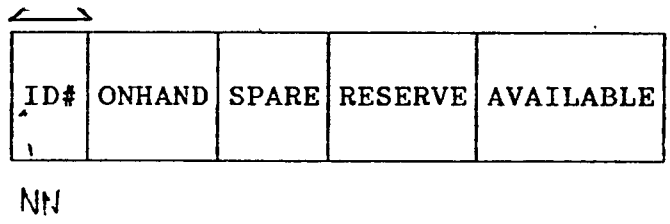
ขั้นตอนการออกแบบ เริ่มจากเขียน NIAM DIAGRAM แสดง ENTITY และความสัมพันธ์ระหว่าง ENTITY ทั้งหมด



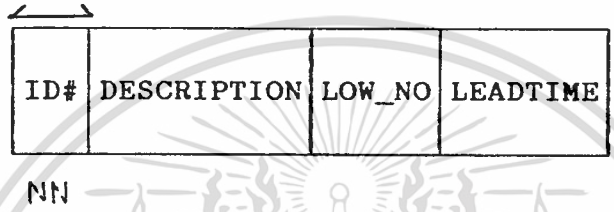
NIAM DIAGRAM แสดง ENTITY และ ความสัมพันธ์

หลังจากนั้น จึงสร้าง BASE TABLE ขึ้นจากการวิเคราะห์ NIAM DIAGRAM ดังรูป

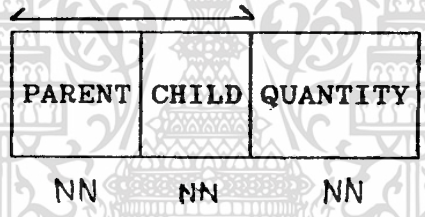
ชื่อ TABLE : INVENTORY



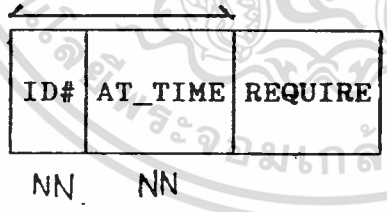
ชื่อ TABLE : SPEC



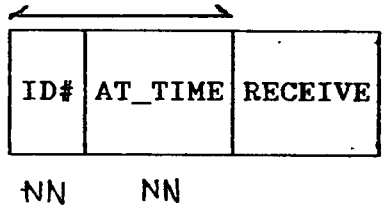
ชื่อ TABLE : BOMI



ชื่อ TABLE : MPS



ชื่อ TABLE : RECEIVE



BASE TABLE ในระบบ MRP

บทที่ 3

วิธีใช้

ผู้ใช้พิมพ์คำสั่ง

C:\>MAIN

จะปรากฏ MAIN MENU ให้เลือก

```
                    MRP  PROJECT

                1. COMPONENT SPECIFICATION
                2. COMPONENT RELATION
                3. INQUIRY PARENT PRODUCT
                4. INVENTORY
                5. MASTER PRODUCT SCHEDULING
                6. PRODUCT WILL RECEIVE
                7. MRP
                8. REPORT
                9. EXIT

Which one do you want :
```

ซึ่งแต่ละข้อ มีหน้าที่การทำงานแตกต่างกันไป ดังนี้

1. COMPONENT SPECIFICATION

รับข้อมูล ได้แก่ คำอธิบาย(DESCRIPTION) ช่วงเวลานำ(LEADTIME)

ไม่อนุญาติให้รับข้อมูล ได้แก่ LOW LEVEL NUMBER

2. COMPONENT RELATION

รับข้อมูลองค์ประกอบของผลิตภัณฑ์ ว่าประกอบด้วยอะไรบ้าง และใช้จำนวนเท่าไหร่

3. INQUIRY PARENT PRODUCT

แสดงองค์ประกอบของผลิตภัณฑ์

4. INVENTORY

รับข้อมูลสถานภาพ สินค้าคงคลังในปัจจุบัน

5. MASTER PRODUCT SCHEDULING

รับข้อมูลตารางการผลิตหลัก ได้แก่ ชนิดสินค้าที่ต้องการ ต้องการเมื่อไหร่

6. PRODUCT WILL RECEIVE

รับข้อมูลขององค์ประกอบ ที่คาดว่าจะได้รับตามเวลาที่กำหนด

7. MRP

วางแผนความต้องการวัสดุ ว่าต้องสินค้าที่เวลาใด จำนวนเท่าใด

8. REPORT

แสดงผลที่ได้จากการวางแผน ออกจอภาพ หรือเครื่องพิมพ์ หรืออาจเป็น

ผลจากการวางแผนครั้งก่อน

9. EXIT

ออกจากโปรแกรม เลิกการทำงาน

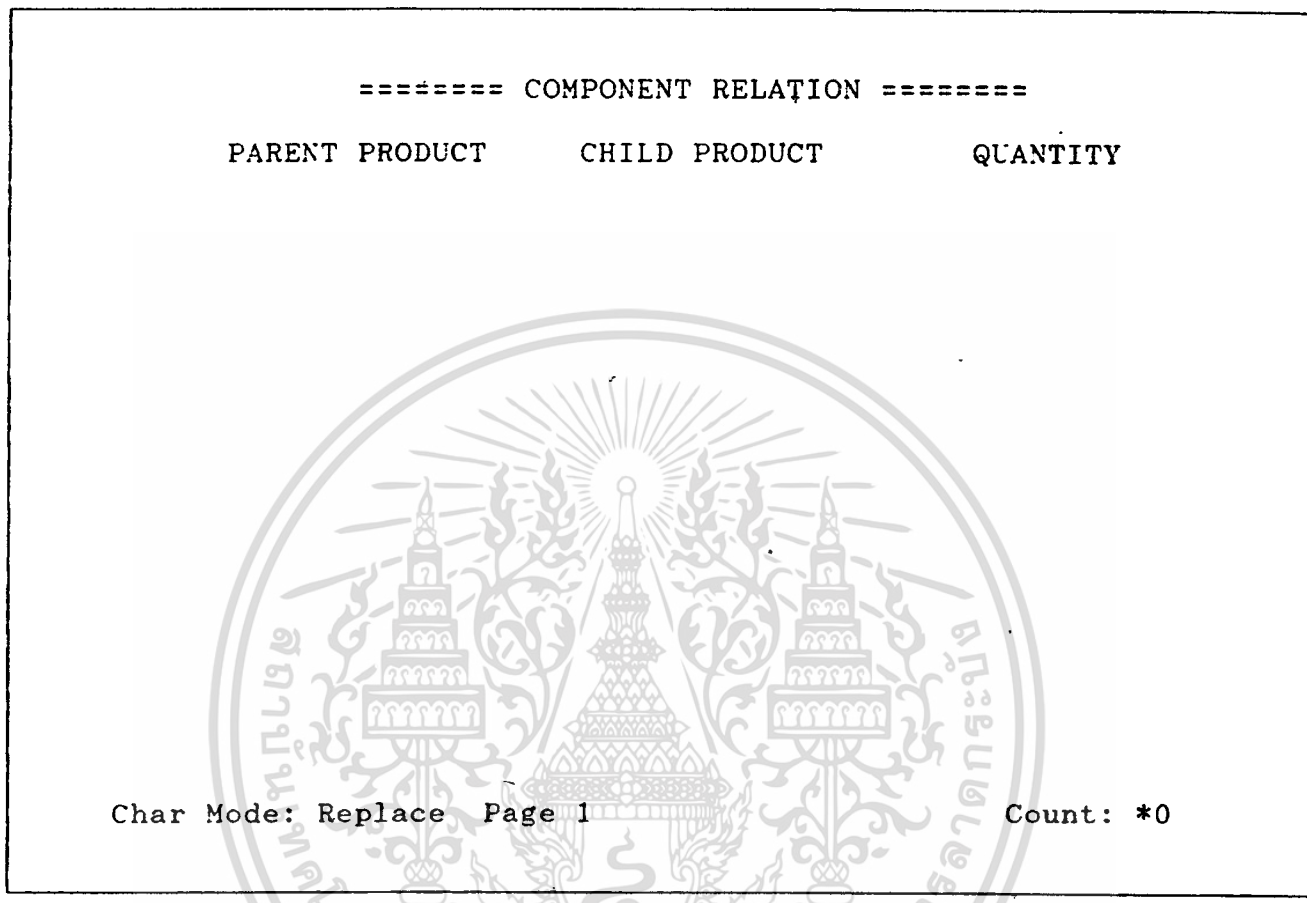
ผู้ใช้ควรจะป้อนข้อมูล เรียงตามลำดับตั้งแต่ข้อแรกลงมา เมื่อมีการวางแผนการผลิตผลิตภัณฑ์ตัวใหม่ หรือ อาจเข้าไปแก้ไขข้อมูลเพียงบางข้อ ในกรณีเปลี่ยนแผนการผลิตที่เคยวางไว้แต่เดิม

1.COMPONENT SPECIFICATION

```
========  COMPONENT SPECIFICATION  ========

PRODUCT ID=          DESCRIPTION          LOW LEVEL NUMBER     LEADTIM
                                                0                0

Char Mode: Replace  Page 1                          Count: *0
```

- ชื่อ FORM : SPEC
- รับข้อมูล รหัส,ชื่อ,และช่วงเวลานำของผลิตภัณฑ์ ตามลำดับ
- LOW LEVEL NUMBER คือ เลขระดับต่ำประจำผลิตภัณฑ์ ซึ่งใช้ในการคำนวณ MRP และ ผู้ใช้ป้อนข้อมูลเข้าไปไม่ได้
- ต้องป้อนข้อมูลของผลิตภัณฑ์ใน FORM นี้ก่อน มิฉะนั้น จะป้อนข้อมูลใส่ FORM อื่นๆ ไม่ได้

2. COMPONENT RELATION



- ชื่อ FORM : BOMI
- รับข้อมูล องค์ประกอบของผลิตภัณฑ์ ว่าประกอบด้วยอะไรบ้าง และใช้จำนวนเท่าไร
- รหัสผลิตภัณฑ์ที่ป้อนเข้ามา จะต้องมีอยู่ก่อนใน COMPONENT SPECIFICATION
- ไม่อนุญาตให้ป้อนรหัสผลิตภัณฑ์ PARENT สลับกับ รหัสผลิตภัณฑ์ CHILD ในกรณีที่มี RECORD ของ PARENT กับ CHILD นั้นแล้ว
- ก่อนออกจาก FORM จะมีการถามให้คำนวณ LOW LEVEL NUMBER หรือไม่ ซึ่งผู้ใช้ควรจะคำนวณใหม่ทุกครั้งที่มีการแก้ไขข้อมูล

3. INQUIRY PARENT PRODUCT

```
What COMPONENT RELATION do you want to see ?
:

PRODUCT STRUCTURE                    LEVEL      QUANTITY
-------------------------------------------------------
```

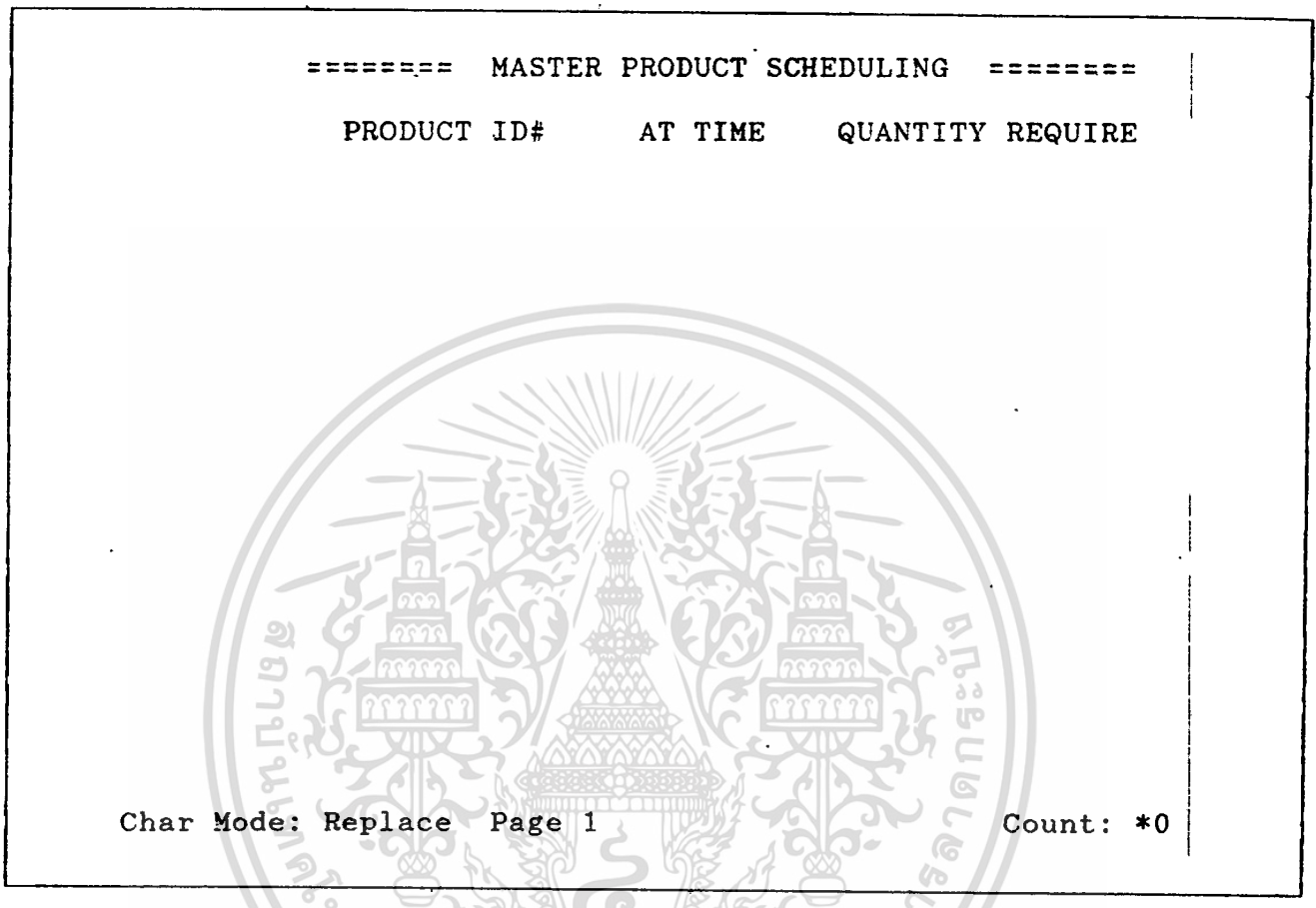
- สอบถามโครงสร้างผลิตภัณฑ์ใดๆ ที่มีอยู่ใน COMPONENT RELATION
- ผู้ใช้ป้อนรหัสผลิตภัณฑ์ แล้วโปรแกรมจะแสดงโครงสร้างให้เห็น เป็นลำดับขั้นลดหลั่นกันไป
- ถ้าไม่แสดงผล หมายความว่า ผลิตภัณฑ์นั้นไม่มีองค์ประกอบอยู่ใน BOM

4. INVENTORY

```
                ========  INVENTORY  ========

      ID#                                ONHAND 0

                                          SPARE 0

                                        RESERVE 0

                                      AVAILABLE 0

Char Mode: Replace  Page 1                          Count: *0
```

- ชื่อ FORM : MATERIAL
- รับข้อมูลสถานภาพสินค้าคงคลังของผลิตภัณฑ์ ได้แก่ ของคงคลังที่มีอยู่ (ON HAND), ของที่ต้องเผื่อไว้ (SPARE), ปริมาณที่ต้องจัดสรรไว้ (RESERVE) และ จำนวนที่สามารถนำไปใช้ได้ (AVAILABLE)
- รหัสผลิตภัณฑ์ที่ป้อนเข้ามา จะต้องมีอยู่ก่อนใน COMPONENT SPECIFICATION
- จำนวนทั้งหมดต้องมากกว่า หรือ เท่ากับ 0 เท่านั้น
- เฉพาะจำนวน AVAILABLE ไม่ต้องป้อน เพราะโปรแกรมจะคำนวณให้โดยอัตโนมัติ

5. MASTER PRODUCT SCHEDULING



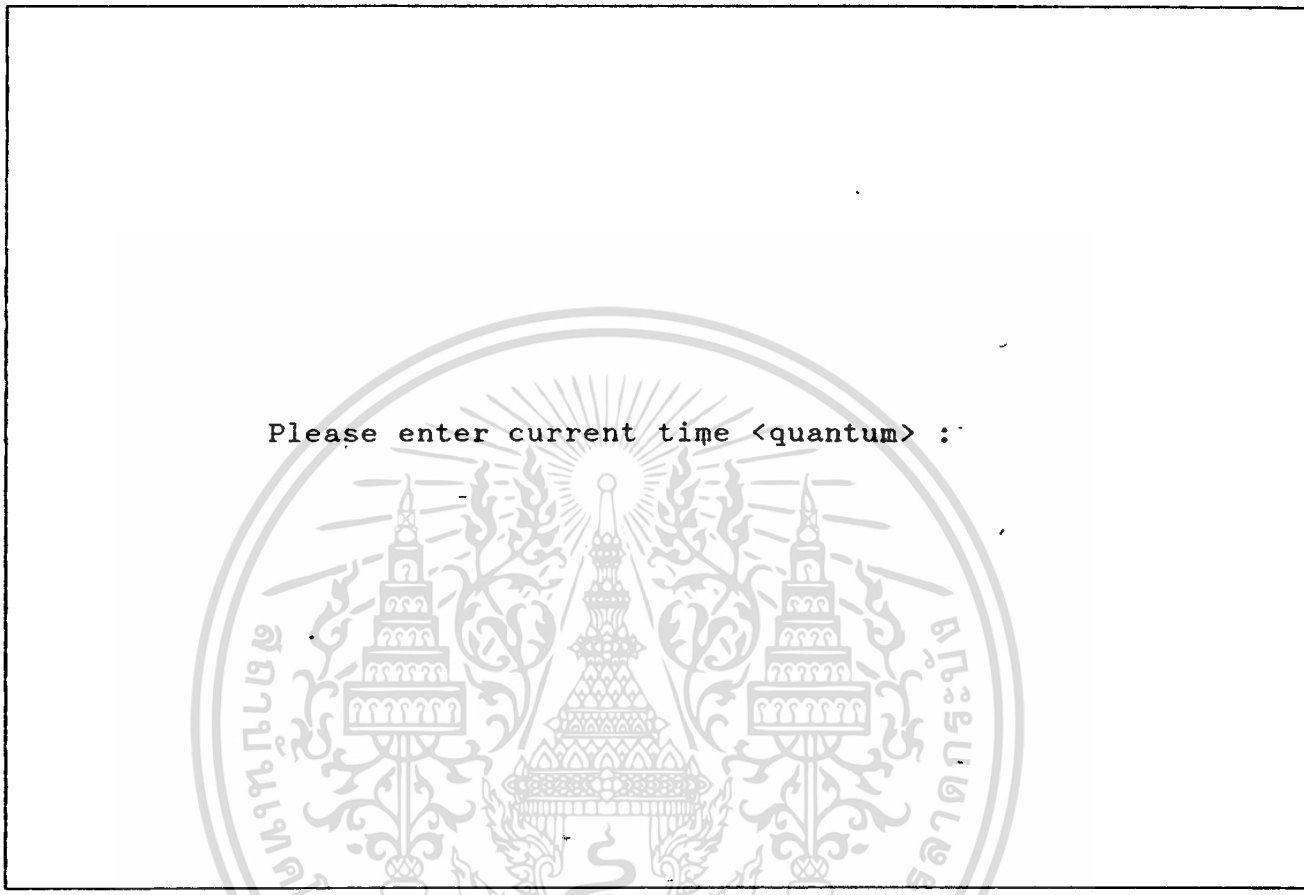
- ชื่อ FORM : MPS
- รับข้อมูลตารางการผลิตหลัก ได้แก่ ชนิดสินค้าที่ต้องการ และ ต้องการเมื่อไหร่
- รหัสผลิตภัณฑ์ที่ป้อนเข้ามา จะต้องมีอยู่ก่อนใน COMPONENT SPECIFICATION

6. PRODUCT WILL RECEIVE

```
        ========  PLANNING TO RECEIVE  ========
       PRODUCT ID#        AT TIME      QUANTITY TO RECEIVE




Char Mode: Replace  Page 1                          Count: *0
```

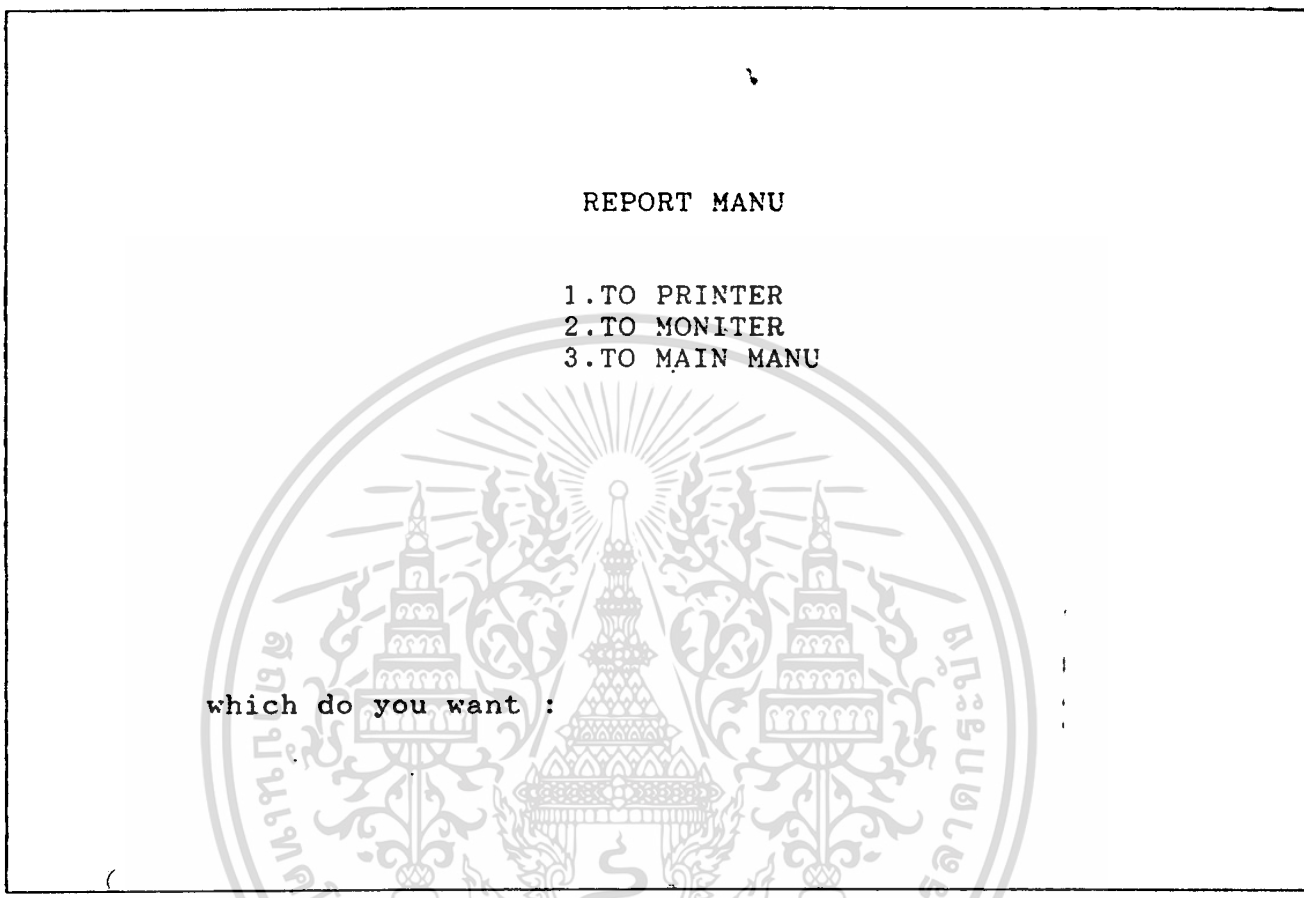
- ชื่อ FORM : RECEIVE
- รับข้อมูลขององค์ประกอบ ที่คาดว่าจะได้รับตามเวลาที่กำหนด
- รหัสผลิตภัณฑ์ที่ป้อนเข้ามา จะต้องมีอยู่ก่อนใน COMPONENT SPECIFICATION

7. MRP



- ส่วนของ MRP PROCESSING MODULE ทำการคำนวณแผนการผลิต หลังจากที่ได้ป้อนข้อมูลทุกอย่างเกี่ยวกับผลิตภัณฑ์ เรียบร้อยแล้ว
- ก่อนการคำนวณ จะถามเวลาปัจจุบัน (CURRENT TIME) ก่อน เพื่อให้ได้แผนการผลิตออกมา สอดคล้องกับเวลาจริง
- แผนการผลิตที่ได้ จากการคำนวณ ถูกเก็บไว้ในไฟล์ ดังนั้น เมื่อวางแผนการผลิตครั้งหนึ่งแล้ว สามารถเรียกดูผลได้โดยไม่ต้องคำนวณใหม่

8. REPORT

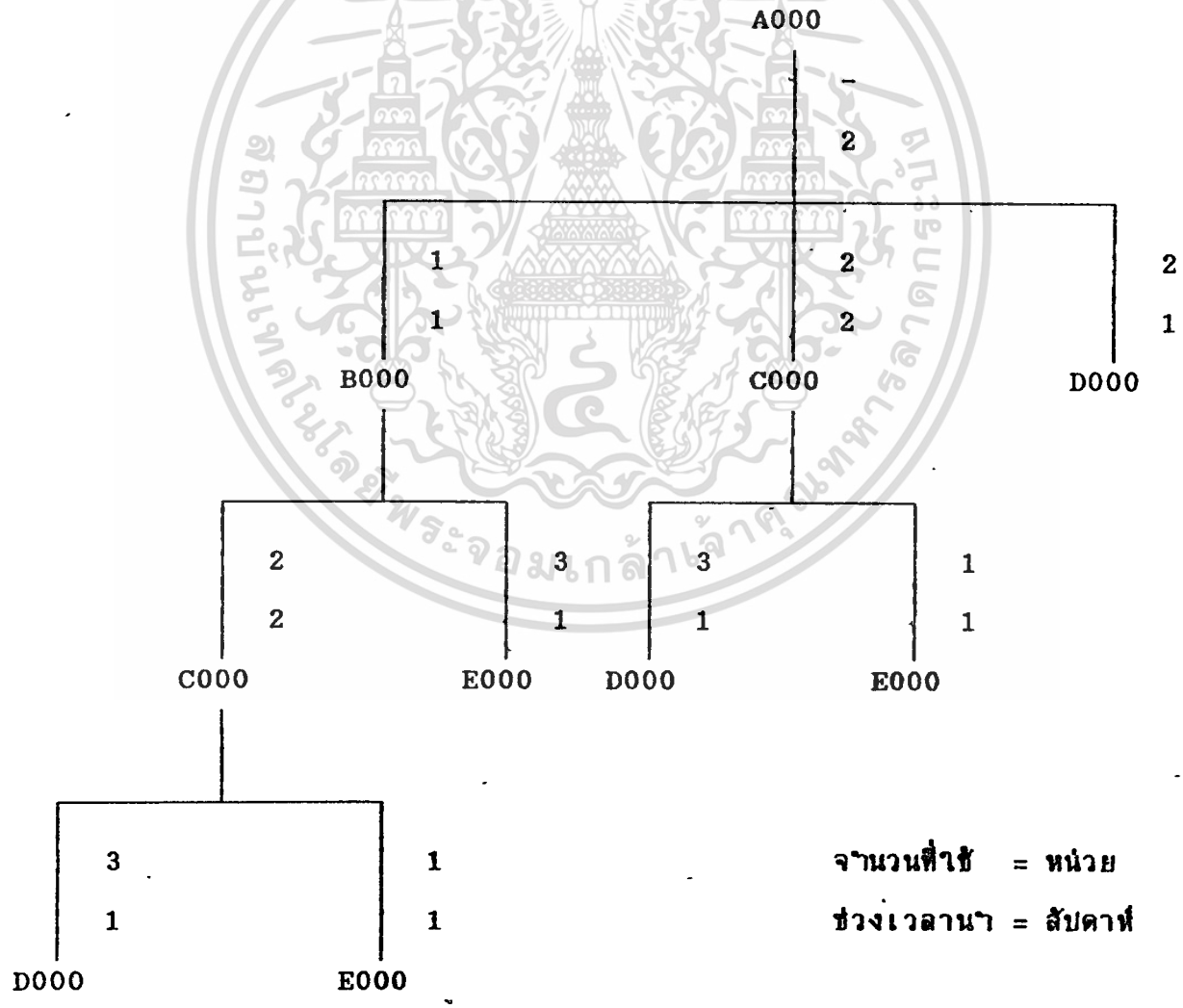


- ผู้ใช้เลือกดูแผนการผลิตครั้งล่าสุด ทางหน้าจอ หรือ ทางเครื่องพิมพ์
- ถ้าตารางการผลิตหลัก ยาวเกิน 1 หน้าจอ ผู้ใช้จะต้องกดปุ่มเพื่อดูหน้าจอถัดไป
- กระดาษพิมพ์ควรใช้อย่างหน้ายาว เพื่อให้เพียงพอกับ ตารางการผลิตหลัก

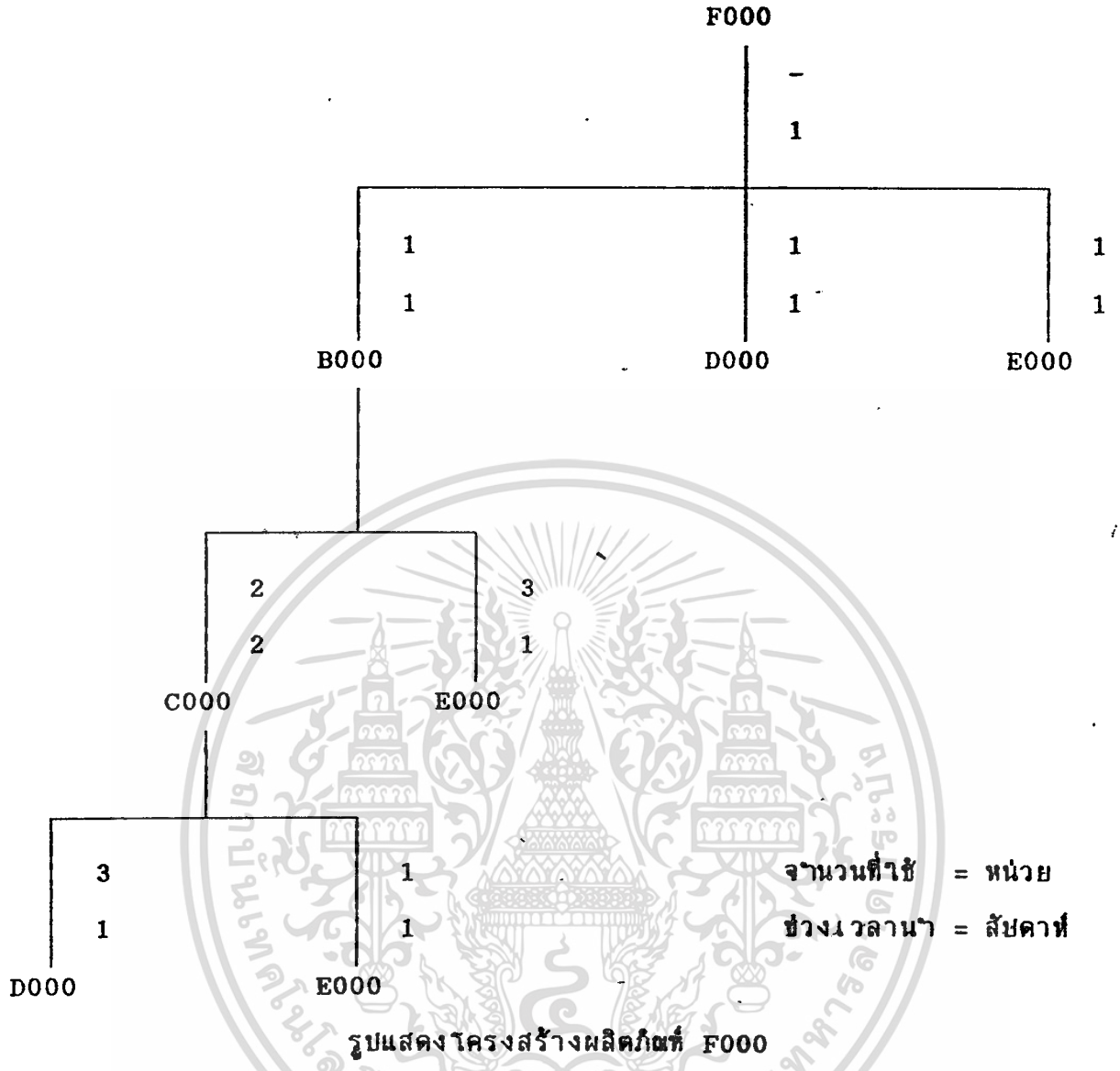
บทที่ 4

การทดลอง และ ผลการทดลอง

ในบทนี้ จะเป็นการทดลองวางแผนการผลิต โดยมีโครงสร้างของสินค้า (PRODUCT STRUCTURE) ตามรูป ใช้แสดงแทนรายการวัสดุ ที่ใช้ในการผลิตสินค้าของตัวอย่างนี้ ซึ่งเป็นรายการวัสดุของผลิตภัณฑ์ 2 ผลิตภัณฑ์ที่แตกต่างกัน คือ A000 และ F000 ตามลำดับ ช่วงเวลานำ (LEAD TIME) ของวัสดุแต่ละหน่วยได้แสดงไว้ด้านข้างของวัสดุนั้นๆ พร้อมทั้งจำนวนที่ต้องใช้ (USAGE QUANTITY) ของวัสดุนั้นๆต่อการผลิตวัสดุ ในระดับที่สูงกว่า 1 หน่วย



รูปแสดงโครงสร้างผลิตภัณฑ์ A000



รูปแสดงโครงสร้างผลิตภัณฑ์ F000

เริ่มแรกสุด ผู้ใช้ป้อนข้อมูลลงใน COMPONENT SPECIFICATION ก่อน คือ ชื่อ และช่วงเวลานำประจำตัว ตามที่ได้แสดงไว้ในโครงสร้าง ของผลิตภัณฑ์ที่เกี่ยวข้องทั้งหมด ดังรูป

======== COMPONENT SPECIFICATION ========

PRODUCT ID#	DESCRIPTION	LOW LEVEL NUMBER	LEADTIME
A000	PRODUCT 'A'	0	2
B000	MATERIAL 'B'	1	1
C000	MATERIAL 'C'	2	2
D000	MATERIAL 'D'	3	1
E000	MATERIAL 'E'	3	1
F000	PRODUCT 'F'	0	1
ROOT	PRODUCT'S ROOT	0	0

แล้วจึงมาป้อนองค์ประกอบของผลิตภัณฑ์ โดยที่ใน 1 RECORD นั้น หมายถึง ความสัมพันธ์ 1 คู่, PARENT PRODUCT คือ ผลิตภัณฑ์ที่ระดับใดๆ, CHILD PRODUCT คือ องค์ประกอบตัวหนึ่ง ซึ่งใช้ผลิต PARENT PRODUCT ตัวนั้น จะสังเกตุได้ว่า ความสัมพันธ์ที่ซ้ำกัน ระหว่างโครงสร้างสองรูป จะถูกป้อนลงไปเพียงครั้งเดียว ไม่จำเป็นต้องป้อนซ้ำ

======== COMPONENT RELATION ========

PARENT PRODUCT	CHILD PRODUCT	QUANTITY
ROOT	A000	1
A000	B000	1
A000	C000	2
A000	D000	2
B000	C000	2
B000	E000	3
C000	D000	2
C000	E000	1
ROOT	F000	1
F000	B000	1
F000	D000	1
F000	E000	1

ผู้ใช้สามารถตรวจดูความถูกต้องของโครงสร้างของผลิตภัณฑ์ได้ จาก INQUIRY PARENT PRODUCT

PRODUCT STRUCTURE	LEVEL	QUANTITY
A000	1	1
B000	2	1
C000	3	2
D000	4	2
E000	4	1
E000	3	3
C000	2	2
D000	3	2
E000	3	1
D000	2	2

PRODUCT STRUCTURE	LEVEL	QUANTITY
F000	1	1
B000	2	1
C000	3	2
D000	4	2
E000	4	1
E000	3	3
D000	2	1
E000	2	1

จากนั้น มาป้อนข้อมูลสถานภาพสินค้าคงคลัง ของผลิตภัณฑ์ ตามรายละเอียด ดังนี้

ID#	ONHAND	SPARE	RESERVE	AVAILABLE
A000	0	0	0	0
B000	150	10	0	140
C000	20	20	0	0
D000	700	50	10	640
E000	50	0	0	50
F000	0	0	0	0

ต่อไป เป็นการระบุความต้องการผลิตภัณฑ์ ที่เวลาต่างๆ ในที่นี้ คือ ผลิตภัณฑ์ A000 และ ผลิตภัณฑ์ F000 ส่วน C000 เป็นอะไหล่ที่ต้องมีพร้อมไว้ทุกๆสัปดาห์ ซึ่งถือว่าเป็นความต้องการเช่นเดียวกัน

======== MASTER PRODUCT SCHEDULING ========

PRODUCT ID#	AT TIME	QUANTITY REQUIRE.
A000	8	150
C000	4	10
C000	5	10
C000	6	10
C000	7	10
C000	8	10
F000	7	125

ท้ายที่สุด ผู้ใช้ป้อน ของที่จะได้รับตามกำหนดเวลา ที่สามารถนำมาใช้ในแผนการผลิต ดังรูป

======== PLANNING TO RECEIVE ========

PRODUCT ID#	AT TIME	QUANTITY TO RECEIVE
B000	2	130
C000	1	250

หลังจากที่ได้ป้อนข้อมูลที่จำเป็นแก่การวางแผนการผลิต อย่างครบถ้วนสมบูรณ์แล้ว ผู้ใช้จึงมาเลือก ข้อ 7 จาก MAIN MENU คือ MRP PROCESSING MODULE ทำการคำนวณ และ วางแผนการสั่งผลิต แล้วเลือกข้อ 8 เพื่อดูแผนการสั่งผลิตและสั่งซื้อของ

ผลิตภัณฑ์ทั้งหมด ออกทางหน้าจอ หรือพิมพ์ออกมาทางเครื่องพิมพ์ ดังแสดงในหน้าถัดไป

<u>หมายเหตุ</u> ในระหว่างป้อนข้อมูล ผู้ใช้สามารถกด F8 เพื่อดู FUNCTION KEY ต่างๆได้

FUNCTION	KEYSTROKE(S)	FUNCTION	KEYSTROKE(S
Right	-> Arrow	Duplicate Field	Shift-F7
Left	<- Arrow	Duplicate Record	F7
Scroll Right	Ctrl ->	Enter Query	F1
Scroll Left	Ctrl <-	Count Query Hits	Shift-F1
Next Field	<W Enter	Execute Query	F2
	Tab	Commit Transaction	End
Next Primary Key Fld	Escape TAB	Exit/Cancel	F3
Next Record	Down Arrow	Print	Shift-F9
Next Set of Records	Shift-F2	Redisplay Page	Shift-F8
Next Block	Pg Dn	Help	F10
Previous Field	Back Tab	List Field Values	F4
Previous Record	Up Arrow	Display Error	Shift-F10
Previous Block	Pg Up	Block Menu	Ctrl-Home
Insert/Replace	Ins	Show Function Keys	F8
Delete Character	Del		
Delete Backward	Backspace		
Clear Field	F6		
Clear Record	F5		
Clear Block	Shift-F6		
Clear Form/Rollback	Home		
Delete Record	Shift-F5		
Create Record	F9		

Press any function key to return to form

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6	7	8
0	0	0	A000	DEMAND		0	0	0	0	0	0	0	150
				RECEIVED		0	0	0	0	0	0	0	0
				READY FOR USE		0	0	0	0	0	0	0	0
				NET		0	0	0	0	0	0	0	150
				WILL RECEIVE		0	0	0	0	0	0	0	150
				ORDER		0	0	0	0	0	150	0	0

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6	7
0	0	0	F000	DEMAND		0	0	0	0	0	0	125
				RECEIVED		0	0	0	0	0	0	0
				READY FOR USE		0	0	0	0	0	0	0
				NET		0	0	0	0	0	0	125
				WILL RECEIVE		0	0	0	0	0	0	125
				ORDER		0	0	0	0	0	125	0

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6
150	10	0	B000	DEMAND		0	0	0	0	0	275
				RECEIVED		0	130	0	0	0	0
				READY FOR USE		140	270	270	270	270	270
				NET		0	0	0	0	0	5
				WILL RECEIVE		0	0	0	0	0	5
				ORDER		0	0	0	0	5	0

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6	7	8
20	20	0	C000	DEMAND		0	0	0	10	20	310	10	10
				RECEIVED		250	0	0	0	0	0	0	0
				READY FOR USE		250	250	250	250	240	220	0	0
				NET		0	0	0	0	0	90	10	10
				WILL RECEIVE		0	0	0	0	0	90	10	10
				ORDER		0	0	0	90	10	10	0	0

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6
700	50	10	D000	DEMAND		0	0	0	180	20	445
				RECEIVED		0	0	0	0	0	0
				READY FOR USE		640	640	640	640	460	440
				NET		0	0	0	0	0	5
				WILL RECEIVE		0	0	0	0	0	5
				ORDER		0	0	0	0	5	0

ONHAND	SPARE	RESERVE	PART		WEEK	1	2	3	4	5	6
50	0	0	B00(	DEMAND		0	0	0	90	25	135
				RECEIVED		0	0	0	0	0	0
				READY FOR USE		50	50	50	50	0	0
				NET		0	0	0	40	25	135
				WILL RECEIVE		0	0	0	40	25	135
				ORDER		0	0	40	25	135	0

บทที่ 5
วิจารณ์ และ สรุป

โปรแกรมส่วนทำหน้าที่ MRP Processing มีจุดดีหลายประการ เช่น

- ผลิตภัณฑ์มีโครงสร้างของส่วนประกอบได้ไม่จำกัด ทั้งแนวกว้าง คือ ชิ้นส่วนมากมายประกอบกันเป็นชิ้นส่วนเดียว และแนวลึก คือ สายการผลิตที่ยาวไกล
- คำนวณหาเวลานำโดยรวมของผลิตภัณฑ์ได้
- วางแผนการผลิตได้ทุกระดับในโครงสร้าง
- วางแผนการผลิตได้หลายผลิตภัณฑ์ในคราวเดียวกัน
- พิมพ์รายงานแผนการผลิต เพื่อการสั่งผลิตและสั่งซื้อ ตามเวลา

Use Interface ช่วยอำนวยความสะดวกในการป้อนข้อมูล ในขณะเดียวกัน ก็ควบคุมความถูกต้องของข้อมูลที่ป้อนเข้ามา และยังป้องกันความปลอดภัยในชั้นแรกสุดด้วย User Password

ส่วนแนวทางพัฒนานั้น มีอยู่หลายทาง เช่น ให้สามารถรายงานแยกวัสดุสำหรับผลิตภัณฑ์ต่างประเภทกัน ในกรณีใช้วัสดุชนิดเดียวกัน , ช่วยคำนวณงบประมาณที่ต้องใช้ในการผลิต , มีการวางแผนการผลิตโดยคำนึงถึงขนาดของการสั่งซื้อหรือสั่งผลิต (Lot size) ด้วย

ภาคผนวก

ภาคผนวก ก.

PRO*C

PRO*C ได้รับการออกแบบมาเพื่อแปลงโปรแกรมภาษา C ซึ่งรวมเอาคำสั่งของ SQL ไปเป็นภาษา C ธรรมดาที่สามารถจัดการข้อมูลในระบบฐานข้อมูลของ ORACLE

ในขั้นตอนของการ Precompile PRO*C จะเปลี่ยนชุดคำสั่งที่ขึ้นต้นด้วย EXEC SQL ให้เป็นชุดคำสั่งของภาษา C ซึ่งสามารถจะ complied , linked และ executed ได้ตามปกติของภาษา C ทั่วไป

ขั้นตอนในการใช้งาน PRO*C

1. เขียนโปรแกรม PRO*C โดยใช้ EXEC SQL แล้วตามด้วย SQL Query
2. Precompile โปรแกรมด้วย PRO*C จะได้ Standard C Program
3. Compile โปรแกรมและจะได้ . Object File
4. Link Object File จะได้โปรแกรมที่ทำงานได้

การใช้งาน PRO*C

โปรแกรมทุกโปรแกรมของ PRO*C จะประกอบด้วยส่วน 3 ส่วนที่สำคัญ ดังนี้

1. The DECLARE Section เป็นส่วนที่จองตัวแปรซึ่งใช้ในส่วนของการติดต่อกับ ORACLE

2. The INCLUDE SQLCA เป็นส่วนของ SQL Communication Area ซึ่ง SQL จะส่งผลบางอย่างและข้อผิดพลาดที่เกิดในขณะ execute โปรแกรม มายัง area ส่วนนี้ เป็นหน้าที่ของ Programer ที่จะคอยการตรวจสอบการทำงานของโปรแกรมโดยใช้ตัวแปรซึ่งกำหนดไว้ในส่วนนี้

3. The CONNECT statement เป็นชุดคำสั่งที่ใช้ในการเข้าถึง ORACLE RDBMS

The DECLARE Section

ตัวแปรทุกตัวที่ถูกใช้ในการติดต่อกับ ORACLE จะต้องถูกจองไว้ในส่วนนี้ โดยมีรูปแบบของการกำหนด ดังนี้

```
EXEC SQL BEGIN DECLARE SECTION ;
         int   pempno ;
         char  pname[11] ;
         int   pdeptno ;
EXEC SQL END DECLARE SECTION ;
```

ตัวแปรทุกตัวที่ถูกจองในส่วนนี้ เมื่อถูกใช้ใน SQL statement จะต้องขึ้นต้นด้วย ':' เช่น

```
EXEC SQL SELECT DEPTNO , ENAME
         INTO   :pdeptno , :pname
         FROM   EMP
         WHERE  EMPNO = :pempno ;
```

ข้อกำหนดของการใช้ตัวแปรที่ใช้ใน SQL statement

1. ถูกจองใน The DECLARE Section
2. ใช้ในลักษณะ upper/lower case เดียวกับเมื่อจอง
3. นำหน้าด้วย ':' ใน SQL statement
4. ไม่นำหน้าด้วย ':' ใน C statement
5. ต้องไม่ใช้ SQL Reserved word

The SQL Communication Area

PRO*C โปรแกรม จะต้องกำหนด SQL Communication Area เพื่อใช้ตรวจสอบการทำงานของโปรแกรม

```
EXEC SQL INCLUDE SQLCA ;
```

โครงสร้างของ SQLCA มีดังนี้

```
struct sqlca {
        char    sqlcaid[8] ;
        long    sqlcabc ;
        long    sqlcode ;
        struct {
            unsigned short sqlerrml ;
            char           sqlerrmc ;
            } sqlerrm ;
        char    sqlerrp[8] ;
        long    sqlerrd[6] ;
        char    sqlwarn[8] ;
        char    sqlext[8] ;
        } ;
struct sqlca sqlca ;
```

<u>ความหมายและการใช้งานของ sqlca</u>

สามารถอธิบายได้ดังนี้

sqlca.sqlcaid	ใช้สำหรับ FORTRAN เป็น character string เมื่อเริ่มต้นจะมีค่าเป็น "SQLCA"
sqlca.sqlcabe	ใช้สำหรับ FORTRAN เป็นเลขจำนวนเต็มขนาด 4 bytes เก็บขนาด structure ของ SQLCA
sqlca.sqlcode	เลขจำนวนเต็มขนาด 4 bytes เก็บผลสรุปของการ execute SQL statement 0 หมายถึง การ execute ประสบผลสำเร็จ

+ หมายถึง การ execute ประสบผลสำเร็จ code ปกติคือ 1403 หมายถึง "no row found" หรือ "last row returned"
- หมายถึง พบข้อผิดพลาดขณะการ execute และให้ code แสดงข้อผิดพลาด

sqlca.sqlerrm.sqlerrml

หมายถึง ความยาวของข้อความใน sqlca.sqlerrm.sqlerrmc ซึ่งมีค่าสูงสุด 70 ตัวอักษร

sqlca.sqlerrm.sqlerrmc

หมายถึง ข้อความที่บอกข้อผิดพลาดที่เกิดจากการ execute โดยมีความสัมพันธ์กับ error code ใน sqlca.sqlcode

sqlca.errp สามารถใช้เก็บข้อความได้ แต่ปัจจุบันไม่ใช้

sqlca.sqlerrd เป็น array ของจำนวนเต็ม 6 จำนวน ขนาด 4 bytes ปัจจุบันนี้ใช้เพียง element[2] ซึ่งจะเก็บจำนวน row ที่ได้จากการ INSERT และ UPDATE

sqlca.sqlwarn ประกอบด้วยตัวอักษร 8 ตัวเรียงกัน แต่ละตัวมีความหมายแตกต่างกัน

เมื่อไม่มีความผิดพลาด bit แรก มีค่าเป็น blank

เมื่อเกิดความผิดพลาด bit แรก มีค่าเป็น 'w'

เพื่อให้ programer สามารถตรวจสอบความผิดพลาดได้จากตัวอักษร 7 ตัวถัดไป

sqlca.sqlwarn[0] เป็น blank เมื่อไม่มีข้อผิดพลาด

เป็น 'w' เมื่อมีข้อผิดพลาดเกิดขึ้น

sqlca.sqlwarn[1] เป็น 'w' เมื่อตัวแปรที่จองไว้เพื่อเก็บข้อความมีค่าน้อยเกินไป

sqlca.sqlwarn[2] เป็น 'w'เกิดการ ignore ค่า NULL ในการ

	คำนวน AVG , SUM , MIN , MAX ซึ่ง programer สามารถใช้ function NVL เพื่อใส่ 0 ลงไปในตำแหน่ง NULL
sqlca.sqlwarn[3]	เป็น 'W' เมื่อจำนวน data ที่ได้จากการ SELECT มีค่าไม่เท่ากับจำนวนตัวแปรที่จองไว้ใน INTO clause
sqlca.sqlwarn[4]	เป็น 'W' เมื่อ UPDATE หรือ DELETE statement โดยไม่มี Where Clause ซึ่งส่งผลให้เกิดการกระทำกับทุก row ซึ่ง ORACLE จะไม่อนุญาติให้การกระทำนั้นเกิดขึ้น
sqlca.sqlwarn[5]	ปัจจุบันนี้ไม่ใช้
sqlca.sqlwarn[6]	เป็น 'W' เมื่อ ORACLE เกิด ROLLBACK โดยทางอ้อมเช่นเกิด Deadlock แต่จะไม่เป็น 'W' เมื่อ ORACLE Rollback โดยตรงจากประโยค ROLLBACK WORK
sqlca.sqlwarn[7]	เป็น 'W' เมื่อ data ใน row ต่างจาก data ในปัจจุบัน อันเกิดเนื่องจากการ UPDATE row นั้นหลังจากมีการ Fetch data จาก row ไปแล้ว
sqlca.sqlext	ปัจจุบันนี้ไม่ใช้

Connecting to ORACLE

PRO*C program จะต้องทำการ connect กับ ORACLE ก่อนจึงจะสามารถใช้งาน Database ได้ ซึ่งการ connect กับ ORACLE สามารถทำได้จากคำสั่ง

```
EXEC SQL CONNECT :id IDENTIFIED BY :password ;
```

เมื่อเราสามารถติดต่อกับ ORACLE แล้ว เราก็สามารถจะใช้ฐานข้อมูลได้โดยใช้คำสั่ง EXEC SQL ตามด้วย คำสั่งของ SQL ทั่วไป

USING CURSOR

cursor เป็นบริเวณเนื้อที่เก็บผลลัพท์จากการ SELECT จาก table โดย ขนาดเนื่อที่ของ cursor นั้นไม่จำกัด ขึ้นกับขนาดของหน่วยความจำ คำสั่งที่ใช้จัดการมีดังนี้

```
DECLARE  CURSOR
OPEN  CURSOR
FETCH
CLOSE  CURSOR
```

The DECLARE CURSOR statement

เป็นการกำหนดชื่อ cursor และให้ cursor รับค่าจากการ Select

```
EXEC SQL DECLARE cursorname CURSOR FOR
         SELECT ......
         FROM   ......
```

ตัวอย่างเช่น

```
EXEC SQL DECLARE  C1 CURSOR FOR
         SELECT   ENAME , EMPNO , JOB , SAL
         FROM     EMP
         WHERE    DEPTNO = :deptno ;
```

The OPEN CURSOR Statement

เมื่อ declare cursor แล้ว ต้อง open cursor ด้วย เพื่ออนุญาตให้ program สามารถดึงข้อมูลมาจาก cursor ได้

```
EXEC SQL OPEN cursorname;
```

ตัวอย่าง

```
EXEC SQL OPEN C1;
```

FETCHING ROWS of the ACTIVE SET

the FETCH statement อ่านข้อมูลออกมาจาก cursorname ทีละแถว แล้วนำข้อมูลนั้นใส่ในตัวแปรที่นำมารับค่า

```
EXEC SQL FETCH cursorname INTO :hostvar 1, :hostvar 2,...
```

ตัวอย่าง

```
EXEC SQL FETCH C1 INTO  :pename,:pempno,:pjob,:psal;
```

การ Fetch ข้อมูลจาก cursor เป็นลักษณะไปข้างหน้าแต่เพียงอย่างเดียว ด้วยเหตุนี้ ถ้าต้องการได้ข้อมูลก่อนหน้านี้ จะต้องทำการ close cursor แล้ว re-open อีกครั้ง

ถ้าต้องการเปลี่ยน query หรือ ตัวแปรใด ๆ ใน query จะต้องเปลี่ยนตัวแปรก่อน จากนั้น close แล้ว open อีกครั้ง

ถ้าการ Fetch ถึง row สุดท้ายใน cursor แล้วค่าใน SQL CODE เป็น +1403 หากต้องการทำการใด ๆ กับ cursor อีก ต้อง close แล้ว open ใหม่

The CLOSE CURSOR Statement

เมื่อไม่ต้องการปฏิบัติการกับ cursor แล้ว จะต้อง CLOSE cursor เพื่อคืน resources ต่าง ๆ ให้กับระบบ ด้วยคำสั่ง

```
EXEC SQL CLOSE cursorname ;
```

ภาคผนวก ข.

ชื่อ FILE โปรแกรมของระบบ

ระบบการวางแผนการผลิต ประกอบด้วย FILE โปรแกรม และ USER INTERFACE ดังต่อไปนี้

1. SOURCE PROGRAM ก่อนการคอมไพล์
   - LEADT.PC
   - LEVEL.PC
   - MAIN.PC
   - MATH.PC
   - REP.PC
2. SQL * FORM FILE
   - BOMI.INP , BOMI.FRM
   - MATERIAL.INP , MATERIAL.FRM
   - MPS.INP , MPS.FRM
   - RECEIVE.INP , RECEIVE.FRM
   - SPEC.INP , SPEC.FRM

ภาคผนวก ค.

Source Code

```
/*-------------- LEADT.PC ---------------*/
#include    <c:\msc\include\dos.h>
#include    <c:\msc\include\stdio.h>
#include    <c:\msc\include\stdlib.h>
#include    <c:\msc\include\process.h>
#include    <c:\msc\include\math.h>
#include    <c:\oracle5\pro\a.h>

EXEC SQL BEGIN DECLARE SECTION;
            VARCHAR   name[5];
            VARCHAR   goods[5];
            VARCHAR   buf_child[30][5];
            int       levels[30];
            int       times;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

/*------------------------ Main process of mrp ---------------------------------*/

mrp(current_time)
int current_time ;
{
int       possible,flag;
int       max,min,level,body;
int       ind =0;
int       count[20];
int       product[6][100];
int       seq[20];
string    buf_level[10][20];
string    list_name[50];
buf_prod  temp[50][20];
FILE      *fp1;

exec sql whenever not found continue;
EXEC SQL WHENEVER SQLERROR GOTO errrpt;

possible = leadtime_check(current_time);
if(possible == 0)
   {
   sort_by_level(&min,&max,buf_level,count,list_name,&ind);
   initial(seq,temp);
   clear_prod(product);
   if((fp1 = fopen("data","wb"))==NULL)
      {
      printf("\nCANNOT OPEN DATA FILE FOR WRITING");
      exit(-1);
      }
      fwrite(&current_time,sizeof(int),1,fp1);
      fwrite(&ind,sizeof(int),1,fp1);
   for(level=min; level <=max; level++)
      for(body=0; body<count[level]; body++)
         {
         strcpy(name.arr,buf_level[level][body]);
         name.len = strlen(name.arr);
         calculate_qualtity(product,temp,seq,list_name,ind);

         write_file(fp1,product);
         clear_prod(product);
         save_in_buf(seq,temp,list_name);
         }
      fclose(fp1);
      report();
      }
   else
```

```
      if(possible == 1)
         printf("\nERROR BECAUSE TIME INCORRECT LESS THAN CURRENT TIME");
      else
         if(possible == 2)
           printf("\nERROR BECAUSE REQUIRE TIME AT LEAST ITS MAXIMUM LEADTIME");
     return ;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*------------------ check time possible to do mrp process --------------------*/

leadtime_check(current_time)
int current_time;
   {
   int flag;

   flag = check_time(current_time);
   if(flag == 1)
      return(flag);
   flag = enough_time(current_time);
      return(flag);

errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*-------------------- Is time less than current time ? -------------------------*/

check_time(current_time)
int current_time;
   {
   int flag;
   flag=0;
   exec sql declare d1 cursor for
            select distinct(id#)
            from   mps;
   exec sql open d1;
   exec sql whenever not found goto endloop1;

   for(; ;)
      {
      exec sql fetch d1 into :name;
      name.arr[name.len] = '\0';
      exec sql declare d2 cursor for
               select at_time
               from   mps
               where  id# = :name;
      exec sql open d2;
      exec sql whenever not found goto endloop2;

      for(; ;)
         {
         exec sql fetch d2 into :times;
         if(times < current_time)
            {
            flag = 1;
            break;
            }
         }
endloop2:
      exec sql close d2;
```

```
      if (flag==1)
      goto endloop1;
      }
endloop1:
   exec sql close d1;
   return(flag);
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*---------- Is total time to do mrp process less than its max leadtime? ----------*

enough_time(current_time)
int current_time;
   {
   int   flag,max_leadtime,least_time;
   flag=0;
   exec sql declare d3 cursor for
            select distinct(id#)
            from   mps;
   exec sql open d3;
   exec sql whenever not found goto endloop1;

   for(; ;)
      {
      exec sql fetch d3 into :name;
      name.arr[name.len] = '\0';
      strcpy(goods.arr,name.arr);
      goods.len = strlen(goods.arr);
      max_leadtime = cal_maxleadtime();
      exec sql whenever not found goto endloop1;
      least_time = max_leadtime+current_time;
      exec sql declare d4 cursor for
               select at_time
               from   mps
               where  id# = :goods;
      exec sql open d4;
      exec sql whenever not found goto endloop2;

      for(; ;)
         {
         exec sql fetch d4 into :times;
         if(times < least_time)
            {
            flag = 2;
            break;
            }
         }
endloop2:
   exec sql close d4;
   if(flag==2)
      goto endloop1;
   exec sql whenever not found goto endloop1;
   }
   endloop1:
   exec sql close d3;
   return(flag);
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }
/*---------------------- calculate maximum leadtime ----------------------*/
```

```
cal_maxleadtime()
   {
   buffer buf_total[50];
   int    num,count,min,max,body,level,i;
   int    max_leadtime;
   string buf_level[10][30],time_temp[20];
   num = 0;
   max_leadtime=0;
   product_total(buf_total,&num,&max);
   buf_total[num].level_item = 0;
   strcpy(time_temp[0],buf_total[0].name_item);
   for(body=0; body<num; body++)
      {
      if(buf_total[body+1].level_item > buf_total[body].level_item)
         {
         level = buf_total[body+1].level_item;
         strcpy(time_temp[level],buf_total[body+1].name_item);
         }
      else
         {
         count=buf_total[body].level_item;
         sum(time_temp,count,&max_leadtime);
         level = buf_total[body+1].level_item;
         strcpy(time_temp[level],buf_total[body+1].name_item);
         }
      }
      return(max_leadtime);
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*---------------------- sum time each branch ------------------------------*/

sum(time_temp,count,max_leadtime)
int count,*max_leadtime;
string  time_temp[];
   {
   int i;
   int branch_time=0;
   exec sql whenever not found continue;
   for(i=0; i<=count; i++)
      {
      strcpy(name.arr,time_temp[i]);
      name.len = strlen(name.arr);
      exec sql select leadtime into :times
               from   spec
               where  id# = :name;

      branch_time = times+branch_time;
      }
   if (*max_leadtime < branch_time)
   *max_leadtime = branch_time;
   return ;

errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*-------------- Total component item's name of product ------------------*/

product_total(buf_total,num,max)
```

```
    {
    int i;
    strcpy(buf_total[*num].name_item,name.arr);
    buf_total[*num].level_item = 0;
    (*num)++;
    connect_tree(buf_total,num,max);
    return;
errrpt:
    printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
    }

/*------------------- connect all component of product -------------------------*/

connect_tree(buf_total,num,max)
int *num,*max;
buffer buf_total[];
    {
    int row_return,body,i;
    *max=0;
    exec sql whenever not found continue;
    exec sql select child,level into :buf_child,:levels
             from   bomi
             connect by bomi.parent = prior child
             start with bomi.parent = :name;
    row_return = sqlca.sqlerrd[2];

    for(body=0; body<row_return; body++)
       {
       strcpy(buf_total[*num].name_item,buf_child[body].arr);
       buf_total[*num].level_item = levels[body];
       if(*max < buf_total[*num].level_item)
          *max = buf_total[*num].level_item;
       (*num)++;
       }
    return;
errrpt:
    printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}
```

```
/*---------------------- LEVEL.PC --------------*/
#include    <c:\msc\include\dos.h>
#include    <c:\msc\include\stdio.h>
#include    <c:\msc\include\stdlib.h>
#include    <c:\msc\include\process.h>
#include    <c:\msc\include\math.h>
#include    <c:\oracle5\pro\a.h>

EXEC SQL BEGIN DECLARE SECTION;
            VARCHAR   name[5];
            VARCHAR   chd[5];
            int       lev,lowlevel;
EXEC SQL END DECLARE SECTION;
exec sql include sqlca;

/*--------------------- calculate product's level ----------------------------*/

int product_level()
   {
   buffer buf_total[50];
   int    num = 0;
   int    body,max,i;

   exec sql declare e1 cursor for
            select child
            from   bomi
            where  parent = 'ROOT';
   exec sql open e1;
   exec sql whenever not found goto endloop1;
   for(; ;)
      {
      exec sql fetch e1 into :name;
      name.arr[name.len]='\0';

      product_total(buf_total,&num,&max);

      for(body=0; body<num; body++)
         {
         strcpy(chd.arr,buf_total[body].name_item);
         chd.len = strlen(chd.arr);
         lev = buf_total[body].level_item;
         EXEC SQL WHENEVER NOT FOUND goto putintable;
         EXEC SQL SELECT low_no INTO :lowlevel
                  FROM spec
                  WHERE id# = :chd;
         if (lowlevel <= lev)
            {
            EXEC SQL UPDATE spec SET low_no=:lev
                     WHERE  id# = :chd;
            goto endloop;
            }
         else goto endloop;
putintable:
         EXEC SQL INSERT INTO spec
                  (id#,low_no)
                  VALUES(:chd,:lev);
endloop:
         exec sql commit work;
         }
      }
endloop1:
   exec sql close e1;
   return;
errrpt:
```

```
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*------------ sort each component of product in buffer by its level ----------*/

int sort_by_level(min,max,buf_level,count,list_name,ind)

int *min,*max,*ind;
string list_name[];
string buf_level[][20];
int   count[];
   {
   buffer buf_total[50];
   int    num=0;
   int    seq,body,level,index,i;
   int    flag=0;
   for(i=0; i<20; i++)
      count[i]=0;

   exec sql declare e2 cursor for
            select distinct(id=)
            from   mps;
   exec sql open e2;
   exec sql whenever not found goto endloop1;

   for(; ;)
      {
      exec sql fetch e2 into :name;
      name.arr[name.len] = '\0';
      product_total(buf_total,&num,max);
      }
   endloop1:
   exec sql close e2;

   level_total(buf_total,min,max,num);
   exec sql whenever not found goto endloop1;

   for(seq=0; seq<num; seq++)
      {
      level = buf_total[seq].level_item;
      for(index=0; index<count[level]; index++)
         {
         if(strcmp(buf_level[level][index],buf_total[seq].name_item)==0)
            {
            flag=1;
            break;
            }
         }
      if(flag!=1)
         {
         body = count[level];
         strcpy(buf_level[level][body],buf_total[seq].name_item);
         (count[level])++;
         }
      else flag=0;
      }
   for(level=*min; level<=*max; level++)
      {
      for(i=0; i<count[level]; i++)
         {
         strcpy(list_name[*ind],buf_level[level][i]);
         (*ind)++;
         }
      }
```

```
   return ;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*-------------------------------------------------------------------------*/
      select level of all component of product which want to do mrp process
                              from spec table
/*-------------------------------------------------------------------------*/

level_total(buf_total,min,max,num)
int  *min,*max,num;
buffer buf_total[];
   {
   int body,i;
   exec sql whenever not found continue;
   *min=100; *max=0;
   for(body=0; body<num; body++)
      {
      strcpy(name.arr,buf_total[body].name_item);
      name.len = strlen(name.arr);
      exec sql select low_no into :lev
               from spec
               where id# = :name;
      buf_total[body].level_item = lev;
      if(*max<lev)
         *max = lev;
      if(*min>lev)
         *min = lev;
      }
   return ;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }
```

```
/*----------------- MAIN.PC -----------------*/
#include    <c:\msc\include\dos.h>
#include    <c:\msc\include\stdio.h>
#include    <c:\msc\include\stdlib.h>
#include    <c:\msc\include\process.h>
#include    <c:\msc\include\math.h>
#include    <c:\msc\include\string.h>
#include    <c:\oracle5\pro\a.h>

EXEC SQL BEGIN DECLARE SECTION;
            VARCHAR   uid[10];
            VARCHAR   pwd[10];
            VARCHAR   prod_name[20] , child[6] ;
            int       level , quantity ;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

 /*---------------------------  Main of program ------------------------------*/

main()
      {
      int    select,in,current_time ;
      strcpy(uid.arr,"PROJECT4"); uid.len=strlen(uid.arr);
      strcpy(pwd.arr,"MRP"); pwd.len=strlen(pwd.arr);
      do
         {
         clrscr();
         gotoxy(34,5)  ; printf("MRP  PROJECT") ;
         gotoxy(29,8)  ; printf("1. COMPONENT SPECIFICATION");
         gotoxy(29,9)  ; printf("2. COMPONENT RELATION");
         gotoxy(29,10) ; printf("3. INQUIRY PARENT PRODUCT");
         gotoxy(29,11) ; printf("4. INVENTORY");
         gotoxy(29,12) ; printf("5. MASTER PRODUCT SCHEDULING");
         gotoxy(29,13) ; printf("6. PRODUCT WILL RECEIVE");
         gotoxy(29,14) ; printf("7. MRP");
         gotoxy(29,15) ; printf("8. REPORT") ;
         gotoxy(29,16) ; printf("9. EXIT") ;
         printf("\n\nWhich one do you want : ") ; select = getche() ;
         switch (select)
             {
             case '1' : system("cd\\oracle5\\bin");
                        system("runform spec project4/mrp");
                        system("cd\\msc\\tmp");
                        gotoxy(0,23) ; printf("\nPress any key to continue.");
                        getch();
                        break ;
             case '2' : system("cd\\oracle5\\bin");
                        system("runform bomi project4/mrp");
                        clrscr();
                        gotoxy(15,10);
                        printf("Do you want to create new product level(y/n)");
                        in = getche();
                        if( (in == 'y') || (in == 'Y') )
                           {
                           EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
                           EXEC SQL WHENEVER SQLERROR GOTO errrpt;
                           product_level();
                           EXEC SQL COMMIT WORK RELEASE;
                           }
                        system("cd\\msc\\tmp");
                        gotoxy(0,23) ;printf("\nPress any key to continue.");
                        getch();
                        break ;
             case '3' : clrscr();
```

```
                    inquiry();
                    gotoxy(0,23) ; printf("Press any to continue");
                    getch();
                    break ;
            case '4' : system("cd\\oracle5\\bin");
                    system("runform material project4/mrp");
                    system("cd\\msc\\tmp");
                    gotoxy(0,23) ;printf("\nPress any key to continue.");
                    getch();
                    break ;
            case '5' : system("cd\\oracle5\\bin");
                    system("runform mps project4/mrp");
                    system("cd\\msc\\tmp");
                    gotoxy(0,23) ;printf("\nPress any key to continue.");
                    getch();
                    break ;
            case '6' : system("cd\\oracle5\\bin");
                    system("runform receive project4/mrp");
                    system("cd\\msc\\tmp");
                    gotoxy(0,23) ;printf("\nPress any key to continue.");
                    getch();
                    break ;
            case '7' : EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
                    EXEC SQL WHENEVER SQLERROR GOTO errrpt;
                    clrscr();
                    gotoxy(20,10);
                    printf("Please enter current time <quantum> : ");
                    scanf("%d",&current_time);
                    gotoxy(0,23);printf("press any key to continue");
                    getch();
                    clrscr();
                    gotoxy(24,10); printf("please wait...");
                    mrp(current_time);
                    EXEC SQL COMMIT WORK RELEASE;
                    gotoxy(0,23) ;printf("\nPress any key to continue.");
                    getch();
                    break ;
            case '8' : EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
                    EXEC SQL WHENEVER SQLERROR GOTO errrpt;
                    report();
                    EXEC SQL COMMIT WORK RELEASE;
                    gotoxy(0,23) ;printf("\nPress any key to continue.");
                    getch();
                    break ;
             default : break;

           }
        }
        while (select != '9') ;
        printf("\n THANK YOU \n");
        goto eend;
errrpt:
  printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
  EXEC SQL ROLLBACK WORK RELEASE;
  exit(1);
eend: ;
     }

/*----------------- Inquiry the relation component of product-------------------*/

inquiry()
  {
  int   i , y , x;
  strcpy(uid.arr,"PROJECT4"); uid.len=strlen(uid.arr);
  strcpy(pwd.arr,"MRP"); pwd.len=strlen(pwd.arr);
```

```
    EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
    EXEC SQL WHENEVER SQLERROR GOTO errrpt;

    printf("\nWhat COMPONENT RELATION do you want to see ?\n: ");
    gets(prod_name.arr) ; prod_name.len = strlen(prod_name.arr) ;
    prod_name.arr[prod_name.len]='\0' ;
    EXEC SQL DECLARE C1 CURSOR FOR
             SELECT  CHILD , LEVEL , QUANTITY
             FROM    BOMI
             CONNECT BY PARENT = PRIOR CHILD
             START   WITH CHILD = :prod_name ;
    EXEC SQL OPEN C1;
    clrscr() ;
    printf("\nPRODUCT STRUCTURE                    LEVEL     QUANTITY");
    printf("\n-----------------------------------------------------");
    EXEC SQL WHENEVER NOT FOUND GOTO endloop;
    for( ; ; )
         {
         EXEC SQL FETCH C1 INTO :child , :level , :quantity ;
         child.len = strlen(child.arr) ;
         printf("\n") ;
         y = read_y() ; x = 3*level ;
         gotoxy(x,y) ; printf("%s",child.arr) ;
         gotoxy(37,y) ; printf("%d",level) ;
         gotoxy(49,y) ; printf("%d",quantity) ;
         }
endloop:
       EXEC SQL CLOSE C1;
       EXEC SQL COMMIT WORK RELEASE;
       return ;
errrpt:
    printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
    }

/*-------------------------------------------------------------------------*/
/*  READY()                                                                */
/*  return row of cursor position                                          */
/*-------------------------------------------------------------------------*/

int read_y()
     {
     int   out ;
     union REGS regis ;
     regis.h.ah = 3 ;
     regis.h.bh = 0 ;
     int86(0x10,&regis,&regis) ;
     return(regis.h.dh) ;
     }
```

```
/*---------------------- MATH.PC -------------*/
#include    <c:\msc\include\dos.h>
#include    <c:\msc\include\stdio.h>
#include    <c:\msc\include\stdlib.h>
#include    <c:\msc\include\process.h>
#include    <c:\msc\include\math.h>
#include    <c:\oracle5\pro\a.h>


EXEC SQL BEGIN DECLARE SECTION;
     extern VARCHAR   name[5];
     extern int       times;
     extern VARCHAR   chd[5];
            int       quantity;
            int       pre,spr,rsv;
EXEC SQL END DECLARE SECTION;
EXEC SQL include sqlca;

/*---------------- Set initial value of variable to 0 ------------------------*/

initial(seq,temp)

int  seq[],temp[][20];
   {
   int i,j;
   for(i=0;i<20; i++)
   seq[i]=0;

   for(i=0; i<50; i++)
      for(j=0;j<20; j++)
         temp[i][j]=0;

   return;
   }

clear_prod(product)
int product[][100];
   {
   int i,j;
   for(i=0; i<6; i++)
   for(j=0; j<100; j++)
   product[i][j]=0;
   return;
   }

/*---------------------- Mrp calculate planning ------------------------------*/

calculate_qualtity(product,temp,seq,list_name,ind)
int      ind;
int      product[][100];
buf_prod temp[][20];
int      seq[];
string   list_name[];
   {
   int  max_d = 0;
   int  i,j,demand_d;
   int  order_d,eg,value,enough;
   int  flag=0;

   prod_inventory();
   for(i=0; strcmp(list_name[i],name.arr)!=0; i++);
      for(j=0; j<seq[i]; j++)
         {
         demand_d = temp[i][j].dem_d;
```

```
        max_d=demand_d;
        }
    EXEC SQL DECLARE f1 CURSOR for
             SELECT  at_time,require
             FROM    mps
             WHERE   id# = :name;

    EXEC SQL OPEN f1;
    EXEC SQL WHENEVER NOT FOUND GOTO endloop1;
    for(; ;)
       {
       EXEC SQL FETCH f1 INTO :times,:quantity;
       flag=0;
       product[0][times] = product[0][times]+quantity;
       for(j=0;j<seq[i];j++)
          {
          if(temp[i][j].dem_d== times)
             {
             temp[i][j].amount = temp[i][j].amount+quantity;
             flag=1;
             break;
             }
          }
       if(flag!=1)
          {
          temp[i][j].dem_d=times;
          temp[i][j].amount=quantity;
          seq[i]++;
          }
       if(max_d< times)
          max_d=times;
       }
endloop1:
   EXEC SQL CLOSE f1;
   product_receive(product);
   product[2][1]=pre-spr-rsv+product[1][1];
   for(j=2; j <= max_d; j++)
      {
      eg=product[2][j-1]-product[0][j-1];
      if(eg<0)
         eg=0;
      product[2][j]=  product[2][j]+eg+ product[1][j];
      if(product[2][j] < 0)
         product[2][j]=0;
      }
   value=product[2][max_d];
   find_leadtime();
   for(j=0; j < seq[i]; j++)
      {
      demand_d = temp[i][j].dem_d;
      order_d = temp[i][j].dem_d-times;
      enough=product[2][demand_d]-product[0][demand_d];

      if(enough >= 0)
         product[3][demand_d] =  0;
      else
         product[3][demand_d] = abs(enough);
      temp[i][j].amount=product[3][demand_d];
      product[4][demand_d]=product[4][demand_d]
      +product[3][demand_d];
      product[5][order_d]=product[5][order_d]
      +product[3][demand_d];
      }
   return;
```

```
   exit(1);
   }

/*-------------------- Product's qualtity in inventory -------------------- */

prod_inventory()
   {
   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   EXEC SQL SELECT onhand,spare,reserve INTO :pre,:spr,:rsv
            FROM   inventory
            WHERE  id# = :name;
   return;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*-------------------- Scheldul receipts from receive table --------------------*/

product_receive(product)
int product[][100];
   {
   EXEC SQL DECLARE f2 CURSOR for
            SELECT  at_time,receive
            FROM    receive
            WHERE   id# = :name;

   EXEC SQL OPEN f2;
   EXEC SQL WHENEVER NOT FOUND GOTO endloop1;
   for(; ;)
      {
      EXEC SQL FETCH f2 INTO :times,:quantity;
      product[1][times]=product[1][times]+quantity;
      }
endloop1:
   EXEC SQL CLOSE f2;
   return;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*------------------------------- Product's leadtime ------------------------------*/

find_leadtime()
   {
   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   EXEC SQL SELECT leadtime INTO :times
            FROM   spec
            WHERE  id# = :name;
   return;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*---------- Scan component of product and store its qualtity in buffer -------*/

save_in_buf(seq,temp,list_name)
int   seq[];
```

```
buf_prod temp[][20];
string   list_name[];
   {
   int i,j,pind,cind;

   EXEC SQL DECLARE f3 CURSOR FOR
            SELECT child,quantity
            FROM bomi
            WHERE parent=:name;
   EXEC SQL OPEN f3;
   EXEC SQL WHENEVER NOT FOUND GOTO endloop1;

   for(; ;)
      {
      EXEC SQL FETCH f3 INTO :chd,:quantity;

      chd.arr[chd.len] = '\0';
      for (i=0; strcmp(list_name[i],name.arr) != 0;i++);
      for(j=0; strcmp(list_name[j],chd.arr) != 0;j++);
      find_leadtime();

       for(pind=0; pind<seq[i];pind++)
         {
         for(cind=0; cind<seq[j];cind++)
            {
            if((temp[i][pind].dem_d-times) == temp[j][cind].dem_d)
               {
               temp[j][cind].amount = temp[j][cind].amount
                                    +(temp[i][pind].amount*quantity);
               goto eend;
               }
            }
         if(temp[i][pind].amount!=0)
            {
            temp[j][cind].amount=temp[i][pind].amount*quantity;
            temp[j][cind].dem_d=temp[i][pind].dem_d-times;
            seq[j]++;
            cind++;
            eend: ;
            }
         }
      }
endloop1:
   EXEC SQL CLOSE f3;
   return;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }
```

```
/*---------------------- REP.PC ----------------*/
#include    <c:\msc\include\dos.h>
#include    <c:\msc\include\stdio.h>
#include    <c:\msc\include\stdlib.h>
#include    <c:\msc\include\process.h>
#include    <c:\msc\include\math.h>
#include    <c:\oracle5\pro\a.h>


EXEC SQL BEGIN DECLARE SECTION;
     extern VARCHAR   name[5];
     extern int       pre,spr,rsv;
EXEC SQL END DECLARE SECTION;
exec sql include sqlca;
char        *list[] = {
                      "DEMAND       ",
                      "RECEIVED     ",
                      "READY FOR USE",
                      "NET          ",
                      "WILL RECEIVE ",
                      "ORDER        "
                      } ;

extern int current_time ;

/*--------------------- jump to coordinate (x,y) ----------------------------*/


gotoxy(x,y)
unsigned char x,y ;
      {
      union  REGS regis ;
      regis.h.ah = 2 ;
      regis.h.dh = y ;
      regis.h.dl = x ;
      regis.h.bh = 0 ;
      int86(0x10,&regis,&regis) ;
      }

/*------------------------- clear screen ---------------------------------*/


clrscr()
      {
      union  REGS  regis ;
      regis.h.ah = 6 ;
      regis.h.al = 0 ;
      regis.h.bh = 7 ;
      regis.h.ch = 0 ;
      regis.h.cl = 0 ;
      regis.h.dh = 24 ;
      regis.h.dl = 79 ;
      int86(0x10,&regis,&regis) ;
      gotoxy(0,0) ;
      }
/*---------------------- show output to moniter or printer -------------------*/


report()
   {
   int   select;
   clrscr();
   gotoxy(30,7); printf("REPORT MANU");
   gotoxy(29,10); printf("1.TO PRINTER");
```

```
   gotoxy(29,11);printf("2.TO MONITER");
   gotoxy(29,12);printf("3.TO MAIN MANU");
   gotoxy(10,23);printf("which do you want : ");

   select = getch();
   switch(select)
      {
      case '1'  : to_printer();
                  break;
      case '2'  : to_moniter();
                  break;
      case '3'  : return;
      default   : report();
      }
   return;
   errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);

   }

/*----------------------- show output to printer -------------------------------*/

to_printer()
   {
   int    in , count1 , i , j , k ;
   int    max_d=0;
   int    max=0;
   FILE   *fp1;
   char    str[10];
   int    product[6][100];
   int     week=20;
   int      current,begin;
   int      count = 1;
   int      current_time,num_of_product;

   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   clear_prod(product);
   clrscr();
   gotoxy(20,10);printf("Make sure that your printer is onlines.");
   gotoxy(24,12);printf("Then press ANY key to continue.") ;
   getch() ;
   if((fp1 = fopen("data","rb")) == NULL)
      {
      printf("\nCANNOT OPEN DATA FILE FOR WRITING");
      return(-1);
      }
      fread(&current_time,sizeof(int),1,fp1);
      fread(&num_of_product,sizeof(int),1,fp1);
   while(count <= num_of_product)
      {
      current = current_time;
      max=0;
      begin=current_time;
      read_file(product,fp1,&max_d,str);
      strcpy(name.arr,str);
      name.len=strlen(name.arr);
      prod_inventory();
      clrscr();
      max_d = max_d-current_time+1;
      while(max_d > 0)
         {
         if(max_d<week)
            {
            max=max_d;
```

```
            {
            max_d=max_d-week;
            max=week;
            }
         fprintf(stdprn,"\n\nONHAND SPARE RESERVE    PART") ;
         fprintf(stdprn,"                    WEEK") ;

         for (count1=current ; count1< current+max; count1++)
            fprintf(stdprn,"%4d",count1) ;
         fprintf(stdprn,"\n\n%6d%6d%8d%8s",pre,spr,rsv,name.arr) ;
         for (j=0;j<6;j++)
            {
            if (j != 0)
               fprintf(stdprn,"                                  ") ;
            fprintf(stdprn,"   %-15s      ",list[j]) ;
            for (k=current ; k<current+max ; k++)
               fprintf(stdprn,"%4d",product[j][k]) ;
            fprintf(stdprn,"\n\n") ;
            }
         current = max+current;
         gotoxy(1,23);printf("press any key to continue...");
         getch();
         clrscr();
         }
      clear_prod(product);
      count++;
      }
   fclose(fp1);
   return ;
errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*------------------------ show output to moniter ----------------------------*/

to_moniter()
   {
   int    in , count1 , i , j , k ;
   int    max_d=0;
   int    max=0;
   FILE   *fp1;
   char    str[10];
   int    product[6][100];
   int     week=6;
   int      current;
   int     count = 1;
   int     current_time,num_of_product;

   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   clear_prod(product);
   if((fp1 = fopen("data","rb")) == NULL)
      {
      printf("\nCANNOT OPEN DATA FILE FOR WRITING");
      return(-1);
      }
      fread(&current_time,sizeof(int),1,fp1);
      fread(&num_of_product,sizeof(int),1,fp1);
      while(count <= num_of_product)
         {
         current = current_time;
         max=0;
```

```
        prod_inventory();
        clrscr();

        max_d=max_d-current_time+1;
        while(max_d > 0)
           {
           if(max_d<week)
              {
              max=max_d;
              max_d=0;
              }
           else
              {
              max_d=max_d-week;
              max=week;
              }
           printf("\n\nONHAND SPARE RESERVE    PART") ;
           printf("                        WEEK") ;
           for (count1=current ; count1< current+max; count1++)
              printf("%4d",count1) ;
           printf("\n\n%6d%6d%8d%8s",pre,spr,rsv,name.arr) ;
        for (j=0;j<6;j++)
           {
           if (j != 0)
              printf("                                ") ;
           printf("   %-15s      ",list[j]) ;
           for (k=current ; k<current+max ; k++)
              printf("%4d",product[j][k]) ;
           printf("\n\n") ;
           }
        current = max+current;
        printf("\npress any key to continue...");
        getch();
        clrscr();
        }
     clear_prod(product);
     count++;
     }
  fclose(fp1);
  return ;
errrpt:
  printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
  EXEC SQL ROLLBACK WORK RELEASE;
  exit(1);
  }
/*---------------------------- read file ----------------------------------*/

read_file(product,fp1,max_d,str)
int    product[][100];
FILE   *fp1;
int    *max_d;
char   str[];
  {
  int    x,k,i,num_loop,j;


  fread(str,sizeof(char),5,fp1);
  for(i=0;str[i] != ' ';i++);
  str[i-1]='\0';

  for(i=0; i<6; i++)
     {
     fread(&num_loop,2,1,fp1);
```

```
      for(k=0; k<num_loop; k++)
         {
         fread(&j,2,1,fp1);
         fread(&product[i][j],2,1,fp1);
         if(*max_d<j)
         *max_d=j;
         }
      }
   return ;

errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }

/*-------------------------- save data in file -------------------------------*/

write_file(fp1,product)
FILE  *fp1;
int    product[][100];
   {
   int  i,j;
   int end = -2;
   long num_loop;

   fwrite(name.arr,sizeof(char),5,fp1);
   for(i=0;i< 6; i++)
      {
      num_loop = 0L;
      fseek(fp1,2L,1);
      for(j=0; j<100; j++)
         {
         if(product[i][j] != 0)
            {
            fwrite(&j,sizeof(int),1,fp1);
            fwrite(&product[i][j],sizeof(int),1,fp1);
            num_loop++;
            }
         }
      fseek(fp1,-(long int)(sizeof(int)*2*num_loop+2),1);
      fwrite(&num_loop,sizeof(int),1,fp1);
      fseek(fp1,0L,2);
      }
   clear_prod(product);
   return ;

errrpt:
   printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK RELEASE;
   exit(1);
   }
```

กิตติกรรมประกาศ

ผู้จัดทำขอขอบพระคุณอาจารย์ที่ปรึกษาทั้งสองท่าน คือ ดร. ชม กิ้มปาน และ ดร. วรวัฒน์ ลิ้มโภคา ที่ได้ให้ความรู้และคำแนะนำด้วยดีตลอดมา โดยเฉพาะ ดร. วรวัฒน์ ที่กรุณาเอื้อเฟื้ออุปกรณ์ทั้งฮาร์ดแวร์ ซอฟท์แวร์อย่างครบครัน อีกทั้งสถานที่ตลอดระยะเวลาพัฒนาระบบ , ผู้จัดทำ ขอขอบพระคุณบริษัท ออราเคิล (ประเทศไทย) จำกัด ที่ให้ความสนับสนุน DBMS Oracle, SQL*Plus, SQL*Form , กลุ่มสารสนเทศที่ให้ยืมหนังสือค้นคว้า และท้ายที่สุด สำหรับทุกท่านที่ได้ให้คำชี้แนะในทุกๆเรื่อง