เครื่องวิเคราะห์วงจรดิจิตอล

DIGITAL CIRCUIT ANALYZER

โดย

นางสาว นวลพรรณ อรุณรัตน์ 38013361

นาย ศิษฐวัชร ปิยะศีล 38013382

นาย สมพงษ์ วิเวก 38013383

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

หัวข้อปริญญานิพนธ์	**เครื่องวิเคราะห์วงจรดิจิตอล**
	Digital Circuit Analyzer
โดย	**นางสาว นวลพรรณ อรุณรัตน์ 38013361**
	นาย ศิษฐวัชร ปิยะศีล 38013382
	นาย สมพงษ์ วิเวก 38013383
อาจารย์ที่ปรึกษา	**อาจารย์ ไพศาล สิทธิโยภาสกุล**
ภาควิชา	**เทคนิคอุตสาหกรรม**
ปีการศึกษา	**2540**

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของชบวนการศึกษาหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการปริญญานิพนธ์

_____ ประธานกรรมการ
(_____)

_____ กรรมการ
(_____)

_____ กรรมการ
(_____)

_____ กรรมการ
(_____)

_____ กรรมการ
(_____)

หัวข้อปริญญานิพนธ์ **เครื่องวิเคราะห์วงจรดิจิตอล**

โดย

นางสาว นวลพรรณ	**อรุณรัตน์**	**38013361**
นาย ศิษฐวัชร	**ปิยะศีล**	**38013382**
นาย สมพงษ์	**วิเวก**	**38013383**

อาจารย์ที่ปรึกษา **อาจารย์ ไพศาล สิทธิโยภาสกุล**

บทคัดย่อ

โครงงานนี้ เป็นการออกแบบและพัฒนาเครื่องวิเคราะห์วงจรดิจิตอล ซึ่งใช้สำหรับอ่านและวิเคราะห์หาสมการจากวงจรรวมแบบโปรแกรมอาเรย์ลอจิก (PAL) เป็นหลัก โดยมีส่วนฮาร์ดแวร์ที่ใช้สำหรับอ่านข้อมูลจากวงจรรวมแล้วส่งข้อมูลผ่านอินเตอร์เฟสการ์ดให้กับโปรแกรม ซึ่งเป็นส่วนของการคำนวณวิเคราะห์ และอ่านสมการออกมา สำหรับส่วนของซอฟแวร์ เป็นโปรแกรมในการวิเคราะห์สมการจากไอซีโปรแกรมตระกูล PAL นั้น จะใช้หลักการในการจัดการหน่วยความจำแบบเอ็กซ์เทนด์ หรือหน่วยความจำเกินพันไบต์แรก ซึ่งเป็นพื้นที่ที่เกินความสามารถของ MS-DOS ที่จะเข้าถึงในการใช้งาน เพื่อเพิ่มสมรรถนะภาพในการวิเคราะห์หาสมการที่มีความยาวและยากได้

การใช้เทคนิคในการจัดการหน่วยความจำแบบเอ็กซ์เทนด์นั้น จะทำให้เครื่องวิเคราะห์วงจรดิจิตอลมีความสามารถที่จะอ่านสมการจากไอซีโปรแกรมตระกูล PAL ในรูปแบบของ Min-term และ Max-term ได้อย่างถูกต้อง

THESIS **Digital Circuit Analyzer**

NAME **Nualpun Arunrat 38013361**

Sittawath Piyasil 38013382

Sompong Vivake 38013383

ADVISER **Mr. Phaisan Sittiyopassakoon**

Abstract

This thesis presents design and development logic analyzer. This analyzer is mainly used for read and analysis Boolean equation from Programmable Array Logic (PAL) integrated circuits. The project consists of two part, part one is hardware circuit that used for read an information from PAL IC and send data via interface card on personal computer, part two is software on personal computer that use for analysis data and find exactly Boolean equation of PAL IC. The program used Extended Memory Specification (XMS) technique for high performance analysis of the implicit Boolean equation.

The result of this logic analysis can be given in form of Min-term and Max-term.

กิตติกรรมประกาศ

ขอขอบพระคุณ บิดา มารดา ผู้ให้กำเนิดและครูบาอาจารย์ โดยเฉพาะอย่างยิ่ง อาจารย์ ไพศาล สิทธิโยภาสกุล ที่ยอมรับเป็นอาจารย์ที่ปรึกษา ที่ท่านได้กรุณาช่วยชี้ทางสว่าง ให้กลุ่มโครงงาน และ ปริญญานิพนธ์นี้สำเร็จลงได้ด้วยดี

ขอขอบคุณ รุ่นพี่ และเพื่อนๆ ที่ช่วยแนะนำข้อมูลและหนังสือในการเขียนโปรแกรม จนทำให้โปรเจคชิ้นนี้ทำงานได้อย่างมีประสิทธิภาพ

ขอบคุณ มันสมองอันชาญฉลาดของพวกเรา และขอขอบคุณเครื่องมือเครื่องใช้ต่างๆ โดยเฉพาะเครื่องคอมพิวเตอร์ พริ้นเตอร์ และซอฟท์แวร์โปรแกรมที่สำคัญจากอาจารย์ที่ปรึกษา ที่ช่วยอำนวยความสะดวกในการทำโครงงานในครั้งนี้

นางสาว นวลพรรณ อรุณรัตน์ 38013361
นาย ศิษฐวัชร ปิยะศีล 38013382
นาย สมพงษ์ วิเวก 38013383

คณะผู้จัดทำปริญญานิพนธ์

สารบัญ

บทที่ 1

บทนำ

หน่วยความจำ เป็นสิ่งที่จำเป็นสำหรับไมโครโปรเซสเซอร์มาก เพราะเป็นที่ที่เดียวเท่านั้นที่ซีพียูจะใช้เก็บข้อมูลหรือโปรแกรมชั่วคราวได้ จริงอยู่ที่ซีพียูในปัจจุบัน บางตัวมีหน่วยความจำฝังไว้ภายในชิปของตัวมันด้วย แต่นั่นก็ไม่เพียงพอที่จะทำงานได้ การมีหน่วยความจำอย่างเพียงพอจะช่วยให้ทำงานสร้างสรรค์ได้มากขึ้น และมีประสิทธิภาพมากขึ้นไปด้วย อย่างไรก็ดีในการออกแบบสร้างซีพียูนั้น ซีพียูแต่ละตัวมักมีข้อจำกัดในเรื่องการใช้หน่วยความจำ

ถ้าคุณใส่โปรแกรมดีไวซ์ไดรเวอร์ และโปรแกรมฝังตัวในหน่วยความจำมากๆ คุณอาจเหลือพื้นที่ในหน่วยความจำเพียงนิดเดียว ซึ่งบางที่อาจจะน้อยไปจนไม่พอวิ่งโปรแกรมปกติได้ สำหรับเครื่องวิเคราะห์วงจรดิจิตอลเองนั้น ก็ประสบกับปัญหาเรื่องหน่วยความจำไม่พอในการทำงาน จึงต้องหาวิธีเพื่อเพิ่มหน่วยความจำให้เพียงพอ โดยการใช้เทคนิคการจัดหน่วยความจำบน เอกซ์เทนด์เมมโมรี

ดังนั้น จะสามารถกล่าวถึงความสามารถในการทำงานของเครื่องวิเคราะห์วงจรดิจิตอลที่พัฒนาชุดโปรแกรมขึ้นมาใหม่ได้ดังนี้

การอ่านและวิเคราะห์โปรแกรม PAL

การอ่านและวิเคราะห์หาสมการจาก ไอซีโปรแกรม PAL จะเป็นส่วนที่ทำหน้าที่คล้ายกับโลจิกอนาไลเซอร์ โดย PAL ที่ต้องการอ่านนั้นจะต้องเสียบอยู่บนซ็อกเก็ต (test socket) และตัวโปรแกรม จะทำการส่งอินพุท จากค่าต่ำสุดไปจนถึงค่าสูงสุด ซึ่งค่าสูงสุดนี้จะขึ้นอยู่กับจำนวนของอินพุทที่ป้อนเข้าไป เมื่อโปรแกรมส่งค่า ไปให้กับ PAL หนึ่งครั้ง ก็จะทำการอ่านค่าเอาท์พุทของ PAL มาหนึ่งครั้ง จนกว่าจะครบ หลังจากนั้นโปรแกรมจะนำค่าเอาท์พุทต่างๆที่ได้นี้ นำมาวิเคราะห์หาสมการทางลอจิก ภายใต้การวิเคราะห์ด้วยซอฟท์แวร์ที่ถูกพัฒนาขึ้นมาใหม่ที่ทำงานสลับไปมาระหว่าง Extended Memory กับ Convertional Memory ได้อย่างมีประสิทธิภาพ โดยจะสามารถวิเคราะห์หาสมการได้ทั้งรูปแบบของ Minterm และ Maxterm ตามลักษณะการโปรแกรมของไอซี ซึ่งจะสามารถวิเคราะห์หาสมการได้สูงสุด 8 เอาท์พุท จากการรับค่าอินพุทได้สูงสุด 16 อินพุท

การวิเคราะห์ระดับสัญญาณทางลอจิก

สำหรับในส่วนของการวิเคราะห์ระดับสัญญาณทางลอจิกนั้น สามารถวัดได้สูงสุด 24 Channel และสามารถวิเคราะห์ระดับสัญญาณได้ 3 ระดับคือ โลจิก LOW , HIGH และ HI IMPEDENCE โดยจะแบ่งการแสดงผลออกเป็น 3 ครั้ง ในแต่ละครั้งจะสามารถแสดงผลได้ 8 ช่องสัญญาณซึ่งการแสดงผลทั้ง 3 ครั้งจะใช้ฐานเวลาร่วมกัน เพื่อใช้ในการเปรียบเทียบรูปคลื่นของสัญญาณ

การทดสอบสถานภาพของไอซี TTL/CMOS

สำหรับในส่วนนี้จะใช้หลักการเดียวกันกับการวิเคราะห์หาสมการจากไอซีโปรแกรม PAL ซึ่งสามารถตรวจสอบได้เฉพาะไอซีโลจิกเกทพื้นฐาน โดยผู้ใช้ต้องป้อนเบอร์ไอซีให้กับโปรแกรม หลังจากนั้น โปรแกรมจะนำเบอร์ไอซีที่เราป้อนไปเปรียบเทียบกับข้อมูลของไอซีเบอร์นั้นๆ ถ้าผลที่ได้เหมือนกันก็แสดงว่าไอซีตัวนั้นสามารถนำมาใช้งานได้อย่างถูกต้อง

บทที่ 2
ทฤษฎีที่ใช้ในโครงงาน

2.1 Programmable Array Logic (PAL)

2.1.1 แนวความคิดเกี่ยวกับ PAL

PAL เป็นอุปกรณ์ในตระกูลหน่วยความจำสำหรับ นักออกแบบเครื่องมือที่เต็มไปด้วยประสิทธิภาพที่หาที่ติไม่ได้ สำหรับใช้ในการออกแบบวงจร logic ใหม่ ๆ หรือที่มีใช้อยู่แล้ว PAL ช่วยประหยัดเวลาและประหยัดเงินโดยการแก้ปัญหาเกี่ยวกับการแบ่งระบบ และการแก้ปัญหาที่เกิดขึ้นด้วยการเพิ่มเทคโนโลยีของอุปกรณ์สารกึ่งตัวนำเข้าไป

ความเจริญก้าวหน้าด้วยความรวดเร็วในการรวมเอาเทคโนโลยี ซึ่งนำไปสู่ความโตขึ้น และโตขึ้นของฟังก์ชั่นโลจิกมาตรฐาน โดยการนำ IC หลายๆ ตัวมากระทำให้เกิดการ์ดวงจรที่สมบูรณ์ ,ขณะที่ LSI ยังต้องการอุปกรณ์ SSI/MSI เป็นจำนวนมาก สำหรับเชื่อมโยงกับผู้ใช้ระบบ นักออกแบบยังคงใช้การสุ่มโลจิกสำหรับการประยุกต์ใช้งานหลาย ๆ อย่าง

นักออกแบบต้องเผชิญหน้ากับแต่ละปัญหาความยุ่งยากซับซ้อน ในการผลิตตั้งแต่ระดับต่ำๆจนถึงระดับกลางๆในการออกแบบ บ่อยครั้งที่ฟังก์ชั่นสามารถบรรยายได้ดี และยังสามารถแสดงที่มาที่มีประโยชน์จากการประดิษฐ์วงจรรวม อย่างไรก็ดีวัฏจักรของการออกแบบสำหรับวงจรที่มีจำหน่ายก็ยังคงยาวนานและมีราคาสูงมาก นี่เป็นการแสดงถึงความเสี่ยงที่เดียวที่จะยับยั้งผู้ใช้ส่วนมาก เทคโนโลยีที่ใช้รองรับการรวมกันอย่างคล่องตัวมากที่สุด ด้วยความเร็วของการหมุนรอบวัฏจักรของโลจิกที่มีจำหน่ายอยู่ ถูกแก้ไขให้ง่ายขึ้นซึ่งหาได้ไม่ง่ายนักหน่วยความจำแบบถาวรยังช่วยแก้ปัญหาเกี่ยวกับการโปรแกรมด้วย

ตระกูล PAL ในยุคแรกจะใช้ฟิวส์เป็นตัวโปรแกรมโลจิก PAL คือการรวมแนวความคิดของอุปกรณ์ซึ่งรวมเอาความคล่องตัวของโปรแกรมด้วยความเร็วสูงและขยายตัวเลือกของทางเลือกของการเชื่อมต่อระบบ PAL ซึ่งสามารถรายการสิ่งของ,ตัดวัฏจักรของการออกแบบและให้ความยุ่งยากซับซ้อนสูง แต่จะให้ความคล่องตัวสูงสุด ลักษณะสำคัญต่างๆเหล่านี้ ได้รวมถึงการลดขนาดของชิ้นงานและให้ความแม่นยำสูง ด้วยความจริงที่ว่า วงจร PAL เป็นเสมือนเพื่อนที่ดีที่สุดของนักออกแบบ

2.1.2 PAL กับการศึกษาเกี่ยวกับ PROM แบบเก่าและศึกษากลวิธีใหม่ๆ

MMI เป็น PROM สมัยใหม่ที่ถูกพัฒนาขึ้นและให้การแนะนำเกี่ยวกับสถาปัตยกรรม และวิธีการต่างๆมากมายซึ่งปัจจุบันถูกยอมรับให้เป็นมาตราฐานของโรงงานอุตสาหกรรม ด้วยการประดิษฐ์ PROM ที่ใหญ่ที่สุดในโลก MMI เป็นเทคโนโลยีที่ทดลองได้ผลแล้ว และมีความสามารถในการผลิตเป็นจำนวนสูงมากเพื่อความต้องการในการผลิตและรองรับ PAL

PAL เป็นการออกแบบเทคโนโลยีของตัวหลอมได้หรือฟิวส์ซึ่งริเริ่มโดย หน่วยความจำแบบถาวรเพื่อใช้ใน PROM แบบไบโพล่าร์ การเชื่อม PROM แบบหลอมได้เริ่มแรกจะใช้แบบดิจิตอล นักออกแบบใช้แรงไฟเขียนบนซิลิกอน ในเวลาเพียงไม่กี่นาทีก็สามารถแปลง PROM ให้ว่างได้ จากวัตถุประสงค์โดยทั่วไปอุปกรณ์ตัวหนึ่งๆจะประกอบไปด้วยสมการพวกอัลกอริทึม ,ไมโครโปรแกรม หรือ การกระทำทางบูลีน นี่เป็นการสร้างชอบเขตใหม่เพื่อใช้สำหรับ PROM ในการควบคุมการเก็บข้อมูลของคอมพิวเตอร์,การกำหนดตัวอักษร,ตารางการเก็บรักษาข้อมูลและการนำไปประยุกต์ใช้งานอื่นๆ การยอมรับอย่างกว้างขวางของเทคโนโลยีนี้ ได้พิสูจน์ให้เห็นชัดเจนโดยตลาด PROM ทุกวันนี้ด้วยเงินหลายๆล้านเหรียญ

กุญแจสำคัญของผลสำเร็จของ PROM นั่นคือออกแบบให้เร็วและง่ายสำหรับผู้ใช้ซึ่งมีขนาดเล็กลงกับความต้องการเป็นพิเศษ PAL สามารถโปรแกรมความคล่องตัวโดยใช้ประโยชน์จากเทคโนโลยีการประสานกันเพื่อหลอมฟิวส์ด้วยเครื่องมือโลจิก การใช้ PAL ออกแบบสามารถทำได้รวดเร็วและได้ผล เครื่องมือทางโลจิกสามารถเปลี่ยนแปลงความยุ่งยากจากการสุ่มของเกท ของความยุ่งยากทางการกระทำทางคณิตศาสตร์

2.1.3 การ AND และการ OR

PAL เป็นเครื่องมือจำพวกผลรวมของผลคูณโลจิกโดยการใช้การโปรแกรมของกระบวนการ AND โดยที่เอาท์พุทเทอมจะถูกกำหนดด้วยกระบวนการ OR ตั้งแต่รูปแบบของการรวมของผลคูณสามารถแสดงผลเกี่ยวกับการแปลงการกระทำทางบูลีน การใช้ของ PAL จะถูกกำหนดด้วยจำนวนเทอม ที่หาได้ง่าย ในกระบวนการ AND - OR PAL เริ่มมีขนาดที่แตกต่างที่ยอมรับเพราะได้ผลในงานทางโลจิก

รูปที่ 2.1.1 แสดงพื้นฐานของ PAL โครงสร้างสำหรับ 2 อินพุท,1 เอาท์พุท เป็นสมการทางโลจิกโดยทั่วๆไปสำหรับส่วนนี้คือ

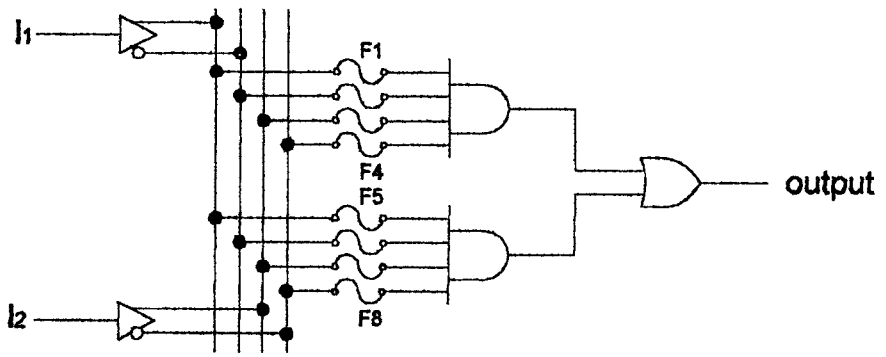
$$\text{Output} = (i_1+\overline{f}_1)(\overline{i}_1+\overline{f}_2)(i_2+\overline{f}_3)(\overline{i}_2+\overline{f}_4)+$$
$$(i_1+\overline{f}_5)(\overline{i}_1+\overline{f}_6)(i_2+\overline{f}_7)(\overline{i}_2+\overline{f}_8)$$

โดยที่ "f" คือตำแหน่งของการต่อเชื่อมของฟิวส์ในกระบวนการ AND ของ PAL การต่อเชื่อมจะไม่ถูกระเบิดถ้าเป็นโลจิก 1 ดังนี้

ฟิวส์ระเบิด , $f = 0$

ฟิวส์สมบูรณ์ , $f = 1$

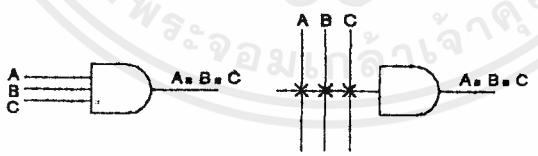
ถ้าปราศจากการโปรแกรม PAL ฟิวส์ทุกตัวจะคงสมบูรณ์



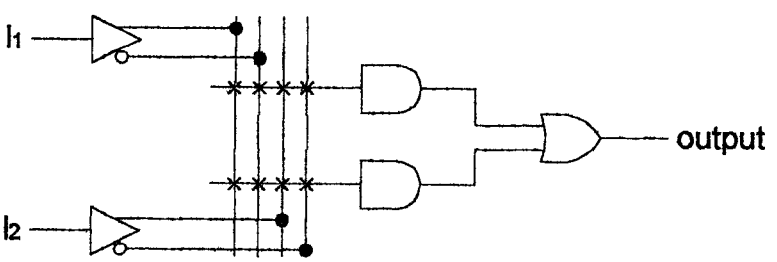
รูปที่ 2.1.1

2.1.4 การให้คำอธิบายเกี่ยวกับ PAL

ขณะที่เครื่องมือให้ความสะดวกสำหรับฟังก์ชั่นเล็กๆ ความเร็วกลายเป็นความลำบากในระบบใหญ่ๆ การลดความยุ่งยากที่เป็นไปได้,ความยุ่งยากของระบบโครงข่ายของลอจิกคือการอธิบายทั่วๆไป โดยใช้ โลจิกไดอะแกรม และตารางความจริง รูปที่ 2.1.2 แสดงแบบแผนของโลจิกที่นำมาใช้เก็บโลจิก PAL แบบง่าย ๆต่อการเข้าใจและใช้งาน ในรูป x คือการแสดงความสมบูรณ์ของฟิวส์ ในรูปแบบของฟังก์ชั่น AND (อินพุทเทอมบนเส้นร่วมที่ประกอบด้วย x คือการไม่ต่อร่วมกัน) สัญลักษณ์ทางโลจิกดังรูปที่ 2.1.2 ถูกนำมาใช้ โดยการรวมวงจรจากโรงงาน เพราะมันถูกสร้างขึ้นมาเพื่อให้เห็นได้ชัดเจน การต่อเชื่อมกันระหว่างจุดต่อจุดระหว่าง IC ที่ถูกออกแบบมาแล้วกับแผน ผังของโลจิกหรือโลจิกไดอะแกรม มันสามารถที่จะยอมรับให้โลจิกไดอะแกรมและ ตารางความจริงรวมอยู่ ดังนั้นการบริการด้วยการให้ความสะดวกสำหรับ PAL 2 อินพุท-1 เอาท์พุท ดังแสดงในรูปที่ 2.1.1 และ 2.1.3



รูปที่ 2.1.2

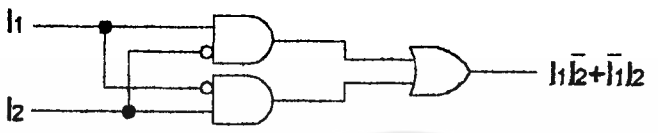


รูปที่ 2.1.3

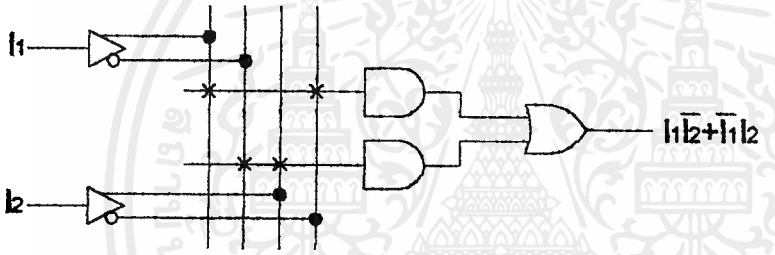
ตัวอย่างอย่างง่ายของ PAL , พิจารณาถึงผลที่เกิดจากการแปลงฟังก์ชั่น

$$\text{Output} = i_1\bar{i}_2 + \bar{i}_1 i_2$$

โดยปกติการรวมกันของโลจิกไดอะแกรม สำหรับฟังก์ชั่นนี้คือรูปที่ 2.1.4 ซึ่งแสดงวงจรสมมูลทางโลจิกแสดงดังรูปที่ 2.1.5



รูปที่ 2.1.4

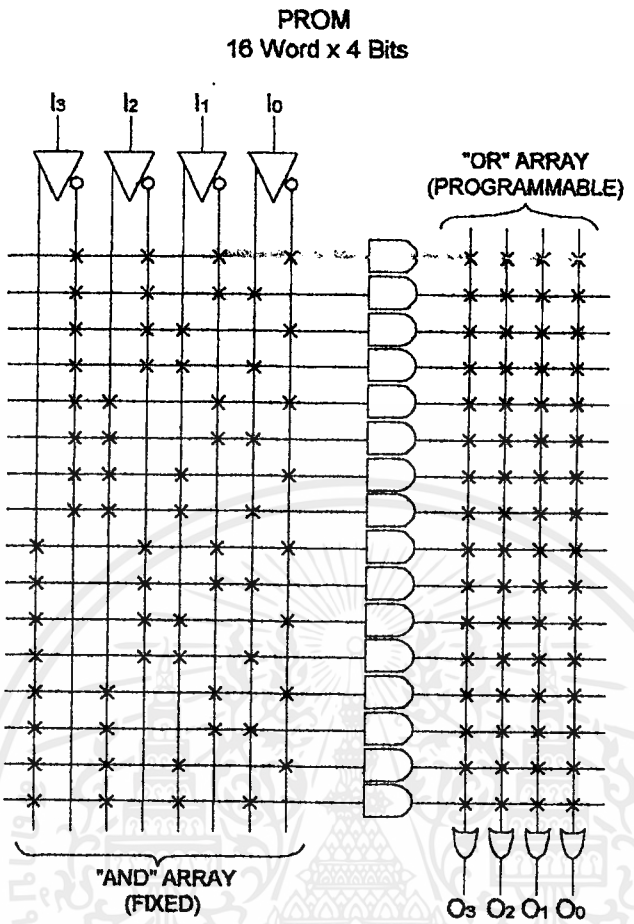


รูปที่ 2.1.5

การใช้การรวมกันของโลจิกในปัจจุบันเป็นไปได้ที่จะเปรียบเทียบ โครงสร้างของ PAL กับโครงสร้างของหลายๆตระกูลของ PROM และ PLA ซึ่งโครงสร้างพื้นฐานทางโลจก ของ PROM ประกอบไปด้วยขบวนการ AND ซึ่งเอาท์พุทจะถูกป้อนเข้าสู่ระบบของขบวนการ OR ดังรูปที่ 2.1.6 PROM ราคาต่ำ, โปรแกรมง่าย และเหมาะสำหรับใช้เปลี่ยนขนาดและการจัดระบบ โดยธรรมดาส่วนมากนิยมใช้เก็บโปรแกรมคอมพิวเตอร์และข้อมูล ในการประยุกต์ใช้งานนี้ได้ระบุอินพุทตามตำแหน่งหน่วยความจำของคอมพิวเตอร์, เอาท์พุทคือตัวบรรจุตำแหน่งหน่วยความจำ

โครงสร้างพื้นฐานทางโลจิกของ PLA ประกอบด้วยการโปรแกรมโดยขบวนการ AND ซึ่งเอาท์พุทได้จากการป้อนข้อมูลผ่านขบวนการ OR ดังรูปที่ 2.1.7 ตั้งแต่ที่นักออกแบบสามารถควบคุมทั้งอินพุทและเอาท์พุทได้อย่างสมบูรณ์, PLA ไม่สามารถที่จะทำการโปรแกรมได้อีก ถูกออกแบบให้ใช้อย่างกว้างขวางในการประยุกต์ใช้งาน อย่างไรก็ดีหลักการสร้าง PLA โดยทั่วๆไปจะแพง, ค่อนข้างจะยากต่อการเข้าใจ และค่าโปรแกรมแพง (ต้องอาศัยนักโปรแกรมที่มีความสามารถเป็นพิเศษ)

โครงสร้างพื้นฐานทางโลจิกของ PAL ประกอบด้วยการโปรแกรมโดยขบวนการ AND ซึ่งเอาท์พุทถูกระบุไว้อย่างแน่นอนด้วยขบวนการ OR ดังรูปที่ 2.1.8 PAL ได้รวมเอาข้อดีของ PLA ด้วยราคาที่ต่ำและการโปรแกรมง่ายของ PROM ตารางที่ 2.1.1 สรุปคุณสมบัติของ PROM ,PLA และ PAL



รูปที่ 2.1.6

	AND	OR	OUTPUT OPTION
PROM	Fixed	Programmabe	TS OC
FPLA	Programmable	Programmabe	TS OC Fusible Polarity
FPGA	Programmable	None	TS OC Fusible Polarity
FPLS	Programmable	Programmabe	TS Registered Feedback, I/O
PAL	Programmable	Fixed	TS Rrgistered Feedback, I/O

ตารางที่ 2.1.1

2.1.5 PAL สำหรับงานหลายๆประเภท

สมาชิกของตระกูล PAL และคุณสมบัติ สามารถสรุปได้ดังตารางที่ 2.1.2 ซึ่งถูกออกแบบให้ครอบคลุมการกระทำของโลจิก เพื่อลดราคา และขนาดโปรแกรมสำเร็จรูป นี่คือการยอมรับของการยอมรับของนักออกแบบว่า PAL เหมาะสมที่สุดที่จะนำมาประยุกต์ใช้งาน PAL จึงเข้ามาเป็นองค์ประกอบพื้นฐานต่อไป

2.1.6 ขบวนการสำหรับรับข้อมูล

ขบวนการรับข้อมูลของ PAL เหมาะที่จะใช้กับขนาด 12X10 (12 อินพุท, 10 เอาท์พุท) ถึงขนาด 20X2 โดยที่ทั้งการทำงานเมื่อเป็น high และ low เหมาะสำหรับองค์ประกอบทางเอาท์พุท ดังรูปที่ 2.1.9 นี่คือการเปลี่ยนแปลงที่กว้างขวางสำหรับการจัดรูปแบบ อินพุทและเอาท์พุท ซึ่งยอมให้ PAL แทนที่ซึ่งจะมีขนาดที่แตกต่างกันเมื่อนำมารวมกันให้เป็นชิ้นเดียวกัน

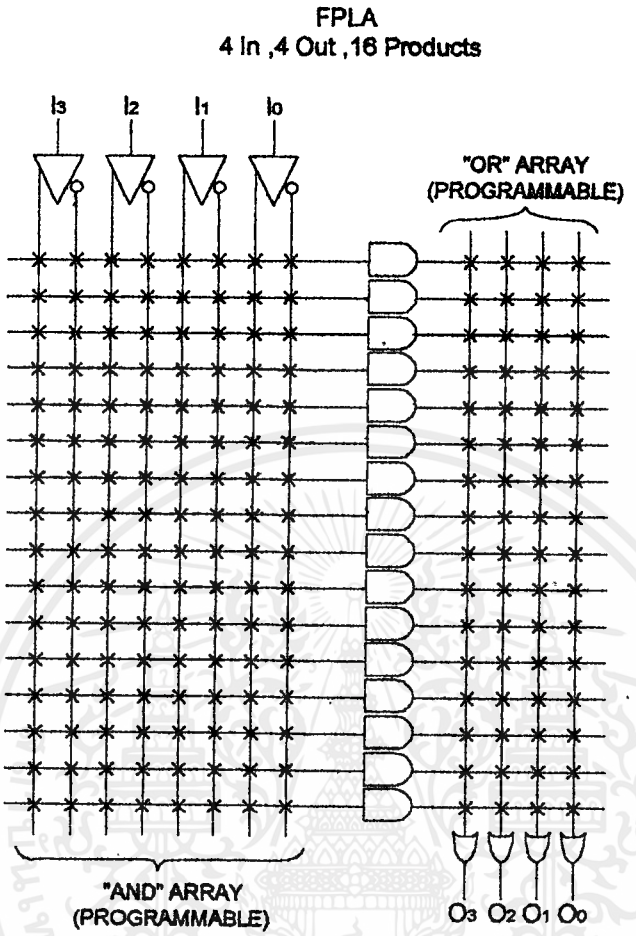
PAL Introduction

PAL Input / Output / Function / Performance Chart

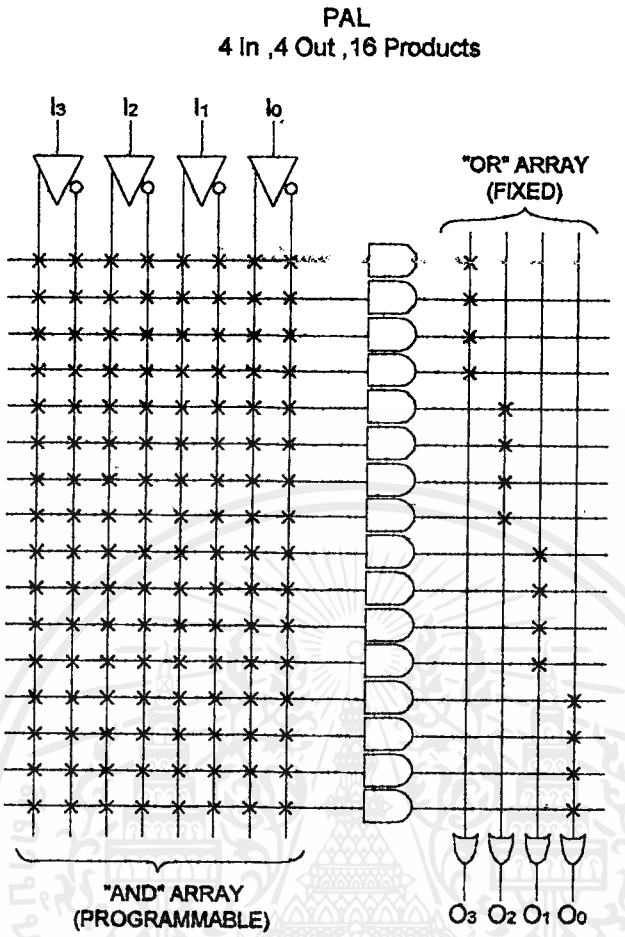
PART NUMBER	I/P	O/P	PROGRAMMABLE I/O'S	FEEDBACK REGISTER	OUTPUT POLARITY	FUNCTION	PERFORMANCE STD	A	-2	-4
PAL10H8	10	8			AND-OR	AND-OR Gate Array	X			
PAL12H6	12	6			AND-OR	AND-OR Gate Array	X			
PAL14H4	14	4			AND-OR	AND-OR Gate Array	X			
PAL16H2	16	2			AND-OR	AND-OR Gate Array	X			
PAL16C1	16	2			BOTH [1]	AND-OR Gate Array	X			
PAL20C1	20	2			BOTH [1]	AND-OR Gate Array	X			
PAL10L8	10	8			AND-NOR	AND-OR Invert Gate Array	X			
PAL12L6	12	6			AND-NOR	AND-OR Invert Gate Array	X			
PAL14L4	14	4			AND-NOR	AND-OR Invert Gate Array	X			
PAL16L2	16	2			AND-NOR	AND-OR Invert Gate Array	X			
PAL12L10	12	10			AND-NOR	AND-OR Invert Gate Array	X			
PAL14L8	14	8			AND-NOR	AND-OR Invert Gate Array	X			
PAL16L6	16	6			AND-NOR	AND-OR Invert Gate Array	X			
PAL18L4	18	4			AND-NOR	AND-OR Invert Gate Array	X			
PAL20L2	20	2			AND-NOR	AND-OR Invert Gate Array	X			
PAL16L8	10	2	6		AND-NOR	AND-OR Invert Gate Array	X	X	X	X
PAL20L10	12	2	8		AND-NOR	AND-OR Invert Gate Array	X			
PAL16R8	8	8		8	AND-NOR	AND-OR Invert Array w/Reg's	X	X	X	X
PAL16R6	8	6	2	6	AND-NOR	AND-OR Invert Array w/Reg's	X	X	X	X
PAL16R4	8	4	4	4	AND-NOR	AND-OR Invert Array w/Reg's	X	X	X	X
PAL20X10	10	10		10	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
PAL20X8	10	8	2	8	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
PAL20X4	10	4	6	4	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
PAL16X4	8	4	4	4	AND-NOR	AND-OR-XOR Invert w/Reg's	X			
PAL16A4	8	4	4	4	AND-NOR	AND-CARRY-OR-XOR Invert w/reg's	X			

[1] Simultaneous AND-OR and AND-NOR output

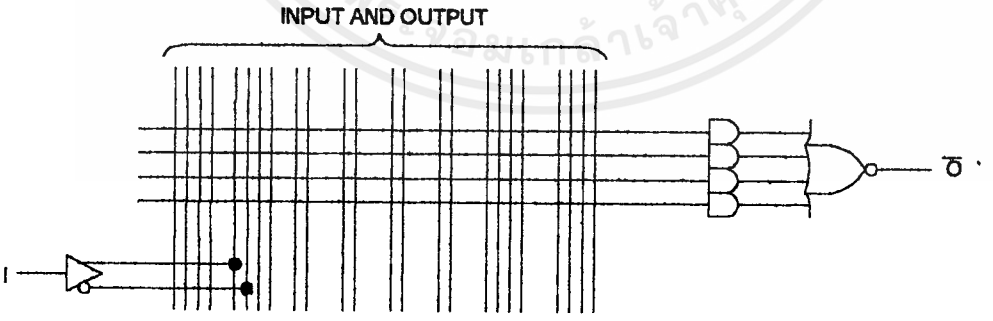
ตารางที่ 2.1.2



รูปที่ 2.1.7



รูปที่ 2.1.8

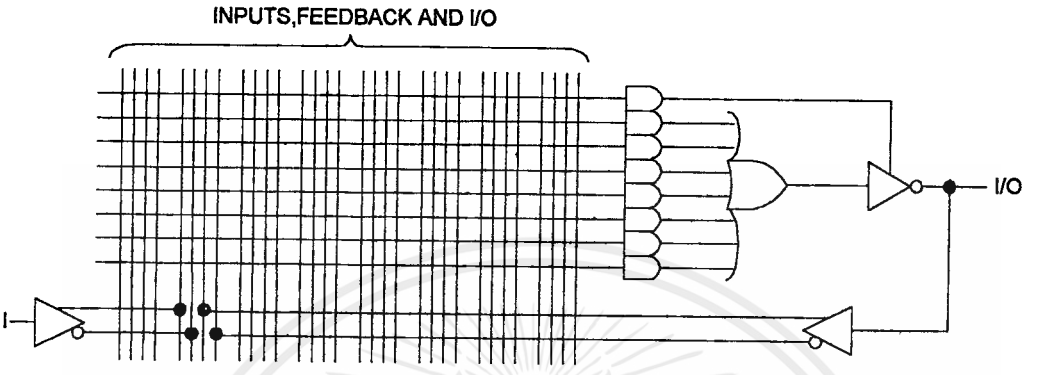


รูปที่ 2.1.9

2.1.7 การโปรแกรม อินพุท / เอาท์พุท

ลักษณะของสมาชิกที่สุดยอดของตระกูล PAL คือการโปรแกรม อินพุท / เอาท์พุท นี่คือการยอมรับเงื่อนไขการผลิตซึ่งควบคุมเอาท์พุทโดยตรงของ PAL ดังแสดงรูปที่ 2.1.10 เงื่อนไขการผลิตอันดับแรกคือกำหนดให้ตัวกั้นหรือ buffer ทำการได้ 3 สภาวะ ซึ่งในการบังคับสวิทช์เงื่อนรวมเข้ากับเอาท์พุท PIN เอาท์พุทจะย้อนกลับไปยังกระบวนการทางอินพุท ของ PAL ดังนั้น PAL จะขับอินพุท / เอาท์พุท PIN เมื่อสวิทช์ 3

สภาวะให้อยู่ในสภาวะที่สามารถต่อเชื่อมได้, อินพุท / เอาท์พุท PIN คืออินพุทของขบวนการของ PAL และเมื่อสวิทช์ 3 สภาวะให้อยู่ในสภาวะที่ไม่สามารถที่จะต่อเชื่อมได้ ลักษณะนี้สามารถใช้ PIN สำหรับจัดเตรียมเนื้อที่สำหรับใช้ในการประมวลผลของฟังก์ชั่น อินพุท / เอาท์พุท หรือกำหนดเอาท์พุท PIN ที่สามารถติดต่อโดยตรงได้ 2 ทาง สำหรับทำงาน เช่น การเลื่อนหรือการหมุน ข้อความแบบอนุกรม



รูปที่2 1.10

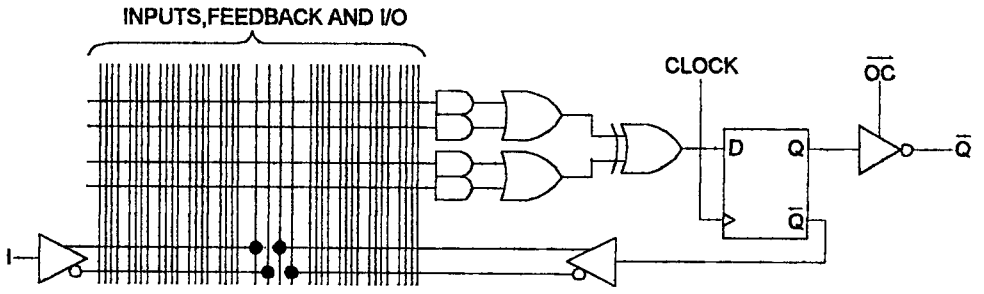
2.1.8 หน่วยความจำย่อยทางเอาท์พุทกับการป้อนกลับ

ลักษณะอื่นของความสุดยอดของตระกูล PAL คือข้อมูลในหน่วยความจำย่อยที่เอาท์พุทกับการป้อนกลับของหน่วยความจำย่อย แต่ละเงื่อนไขการผลิตจะเก็บค่าไว้ในฟลิปฟลอปชนิด D ในช่วงขอบขาขึ้นของระบบนาฬิกา โดยที่ เอาท์พุท Q ของฟลิปฟลอปผ่านสวิทช์ไปยังเอาท์พุท PIN โดยสามารถเชื่อมต่อโดยการทำงานเมื่อเป็น low ของตัวกั้นหรือ buffer 3 สภาวะ

รวมไปถึงความเหมาะสมในการส่งข้อมูล เอาท์พุท Q จะถูกส่งกลับไปยังขบวนการของทางอินพุทของ PAL การป้อนกลับนี้ทำให้ PAL จดจำสภาวะก่อนหน้า และยังสามารถดัดแปลงฟังก์ชั่นพื้นฐานบนสภาวะนั้นได้ การยินยอมดังกล่าวทำให้นักออกแบบสามารถเลือกใช้อุปกรณ์ต่างๆเพื่อสนับสนุนการทำงานของระบบคอมพิวเตอร์ ด้วยการเรียงลำดับสภาวะ ซึ่งสามารถโปรแกรมให้ได้ผลสำเร็จ เช่น การกระทำพื้นฐาน, การนับขึ้น, การนับลง, การนับข้าม, การเลื่อน และการแยกไปทำงานตามที่โปรแกรมกำหนดไว้ การกระทำดังกล่าวสามารถทำให้เป็นผลสำเร็จได้โดยหน่วยความจำย่อยของ PAL ด้วยอัตราสูงขึ้นถึง 20 MHz

2.1.9 ขบวนการ OR แบบพิเศษของ PAL

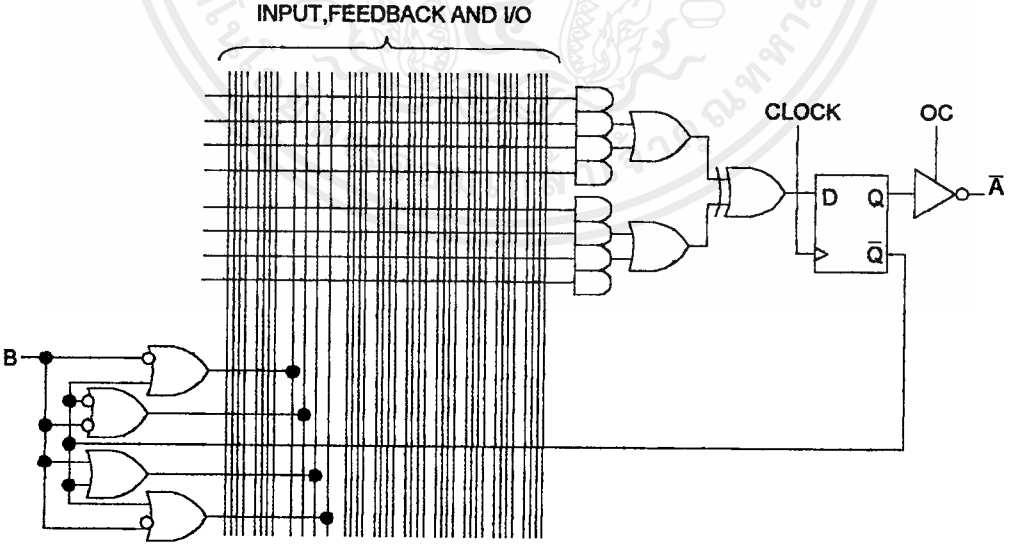
XOR PAL ชนิดนี้มีลักษณะการกระทำการ OR แบบพิเศษ เป็นการรวมของผลคูณเป็นส่วนหนึ่งในการรวมกัน 2 ครั้ง ซึ่ง XOR ที่อินพุทของฟลิปฟลอปชนิด D ดังรูปที่ 2.1.11 ลักษณะทั้งหมดของการหน่วยความจำของ PAL คือการรวมใน XOR PAL การกระทำ XOR กำหนดอุปกรณ์ง่ายๆ ซึ่งการทำงานใช้ในการนับและการจัดเรียงลำดับอื่นๆ



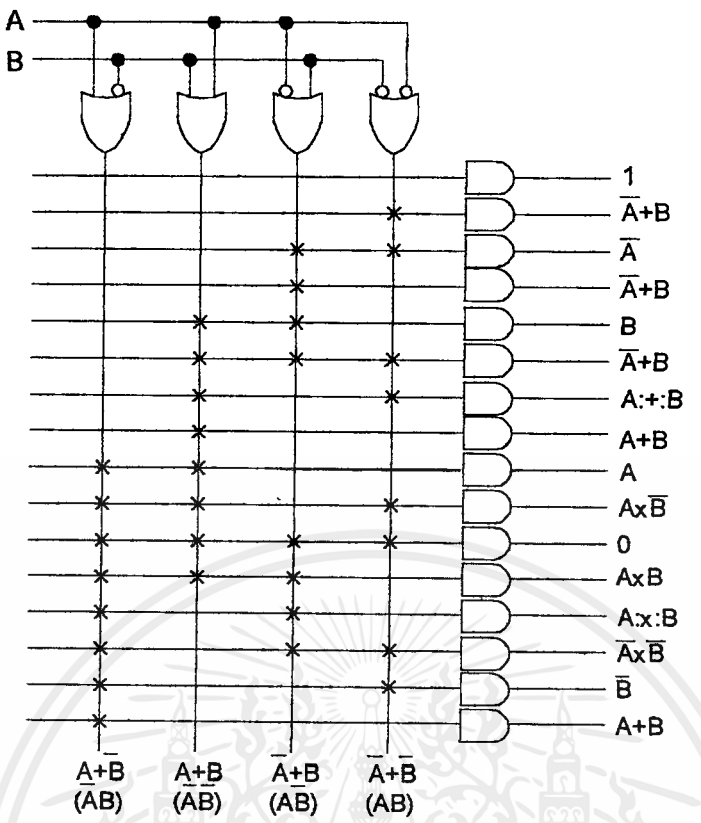
รูปที่ 2.1.11

2.1.10 การป้อนกลับเกี่ยวกับการเลขคณิต ของวงจรตรรก

การกระทำทางคณิตศาสตร์ (+,-,>,<) สามารถทำให้เกิดผลโดยการรวมกันของการป้อนกลับของวงจรตรรกในลักษณะของ XOR PAL ที่อินพุทของฟลิปฟลอปชนิด D ยอมผลักดันให้การทำงานในสภาวะก่อนของ XOR ด้วยการรวมการเปลี่ยนแปลง 2 ส่วน ผลิตโดยกระบวนการของ PAL เอาท์พุท Q ของฟลิปฟลอปที่ป้อนกลับไปยังเกทพื้นฐานโดยอินพุทของเงื่อนไข A ดังรูปที่ 2.1.12 วงจรเกทพื้นฐานนี้ได้จากกรรมวิธีการลดรูปโดยอาศัยกรรมวิธีการลดรูปแบบคาร์นอร์ (Karnaugh Map) ดังรูปที่ 2.1.14 รูปที่ 2.1.13 แสดงขบวนการของ PAL ที่สามารถโปรแกรมได้ 16 วิธีการปฏิบัติการ การกำหนดลักษณะนี้เพื่อประโยชน์ในการทำงานหลายๆอย่างของการกระทำ 2 ตัวแปร และทำให้สะดวกต่อการเปรียบเทียบที่จำเป็นเพื่อให้การทำงานทางคณิตศาสตร์เร็วขึ้น



รูปที่ 2.1.12



รูปที่ 2.1.13

(A+$\overline{B}$),(A+B) \ ($\overline{A}$+B),($\overline{A}$+$\overline{B}$)	---	-X	XX	X-
---	1	$\overline{A}$+$\overline{B}$	$\overline{A}$	$\overline{A}$+B
-X	A+B	A:+:B	$\overline{A}$• B	B
XX	A	A• $\overline{B}$	0	A• B
X-	A+$\overline{B}$	$\overline{B}$	$\overline{A}$• $\overline{B}$	A• :B

รูปที่ 2.1.14

มันทำให้เราเข้าใจอย่างชัดเจนว่า PAL สามารถแทนที่ระบบ SSI หรือ Small-Scale Intergrated ที่ใช้กันอยู่ทุกวันนี้ ดังนั้นต้นทุนการผลิตซึ่งต่ำลงและให้ความคล่องตัวสูงขึ้นในการทำงานของเครื่องมือทางโลจิก

2.1.11 การโปรแกรม PAL

PAL สามารถโปรแกรมโดยนักโปรแกรม PROM โดยทั่วไปร่วมด้วยผู้ใช้โปรแกรม PAL PAL เกิดขึ้นจากนักโปรแกรม PROM ระหว่างการโปรแกรมครึ่งหนึ่งของเอาท์พุท PAL ถูกเลือกสำหรับโปรแกรมในขณะที่เอาท์พุทอื่นๆและอินพุทใช้สำหรับอ้างตำแหน่ง เอาท์พุทเป็นสวิทช์สำหรับเลือกตำแหน่งอื่นๆการพิจารณาใช้วิธีเดียวกับการโปรแกรมในสภาวะต่ำๆ

2.1.12 PALASM (PAL Assembler)

PALASM คือการโปรแกรมอำนวยความสะดวกใช้แสดงความหมาย,จำลอง,สร้าง และทดสอบ PAL PALASM จะรับเอารายละเอียดของการออกแบบ PAL ด้วยอินพุทไฟล์ มันจะพิสูจน์ด้วยการออกแบบให้กระทำตามตารางอีกหนและบ่งบอกตำแหน่งของฟิวส์ ซึ่งจะใช้ในการโปรแกรม PAL PALASM มีประโยชน์ในโอกาสที่มีการร้องขอของรหัสหลายๆแหล่งกำเนิดและเอกสารของแนวความคิดในการออกแบบ PAL

2.1.13 HAL (Hard Array Logic)

ในตระกูล HAL คือการระบุรุ่นของโปรแกรม PAL HAL เป็นของ PAL เช่นเดียวกับที่ ROM เป็นของ PROM มาตราฐานของแผ่นซิลิกอนบางๆที่ใช้สร้างไอซีที่ประดิษฐ์โดยเครื่องจักรครอบอยู่ 6 ชั้น ชั้นที่เป็นโลหะทำขึ้นจากอะลูมิเนียมผสมสำหรับ HAL แทนที่ของการโปรแกรมด้วยกระบวนการฟิวส์ที่ใช้ใน PAL

HAL คือการแก้ปัญหาของผลกระทบทางราคา ซึ่งมีจำนวนมากและเป็นเครื่องที่ผลิตมาเป็นต้นแบบเพียงเครื่องเดียวที่ประกอบด้วยกระบวนการของเกท

HMSI (HAL Medium Scale Integration) ตระกูล HMSI เกิดมาจาก PAL ที่ใช้เทคโนโลยีของ HAL อุปกรณ์นี้ ในตระกูล TTL ที่มีอยู่เพราะพวกมันถูกผลิตมาในปริมาณจำกัด ผู้ใช้จะได้รับประโยชน์คุ้มค่า การออกแบบ HMSI PAL เป็นการประยุกต์ใช้งานกับ IC เบอร์ 74LS

2.1.14 เทคโนโลยีของ PAL

อุตสาหกรรมของ PAL ใช้การพิสูจน์ของ TTL Schottky Bipolar ของการระเบิดฟิวส์ชนิด Ti-W เพื่อสร้างเส้นที่จะถูกหลอมหรือระเบิดใน PROM รูปแบบของ emitter follower ชนิด NPN ในการโปรแกรมด้วยกระบวนการ AND ที่อินพุทของชนิด PNP มีความต้านทานทางด้านเอาท์พุทสูง (สูงสุด 0.25 mA) เอาท์พุทโดยทั้งหมดของ TTL มาตราฐานประกอบด้วยการพูลอัฟด้วยตัวต้านทานภายใน

PAL แบบที่มีใช้กันแพร่หลายมีการหน่วงเวลา 20 nS และ PAL ทั้งหมดมีขนาด 20 ขาและ 24 ขา

2.1.15 ข้อมูลความลับของ PAL

วงจรที่ถูกใช้ในการโปรแกรมและ โลจิกที่ใช้ตรวจสอบความจริง บางครั้งสามารถใช้การกำหนดโดยการเก็บค่าของแบบแผนของโลจิกในกระบวนการ PAL สำหรับความลับ PAL มีฟิวส์ตัวสุดท้ายซึ่งสามารถขยายการทำให้ไม่สามารถตรวจสอบโลจิกได้ นี้คือความสำคัญในการถ่วงความก้าวหน้าของผู้เรียนแบบและมันสามารถป้องกันได้อย่างได้ผลของลิขสิทธิ์ในการออกแบบ

2.1.16 การแบ่งแต่ละส่วนของเบอร์ PAL

แต่ละส่วนของเบอร์ PAL จะถูกระบุในส่วนที่เป็นรหัสของเบอร์ ซึ่งจะแสดงความหมายของการทำงานของโลจิก ตัวอย่างเช่น PAL เบอร์ PAL14LSCN จะมี 14 อินพุท, 4 เอาท์พุท, ทำงานเมื่อเป็น low,บอกอัตราอุณหภูมิ,ขนาด 20 ขา, ทำจากพลาสติก

ตัวอย่าง PAL เบอร์ PAL14L4-2CJ883BPO1234

PAL : บอกกระบวนการหลักว่าเป็นโลจิกตระกูลใด

14 : บอกกระบวนการทางอินพุท

L : ชนิดของเอาท์พุท

L = ทำงานเมื่อเป็น low

C = ส่วนประกอบให้สมบูรณ์

X = ตัวที่เก็บข้อมูลแบบ OR แบบพิเศษ

A = ตัวเก็บข้อมูลแบบคณิตศาสตร์

4 : จำนวนของเอาท์พุท

-2 : ความเร็ว/กำลัง

A = ความเร็วสูง

-2 = ให้กำลังครึ่งหนึ่ง

-4 = ให้กำลัง 1/4 ของกำลัง

C : อัตราของอุณหภูมิ

C = 0 ถึง 75 องศา

M = -55 ถึง 125 องศา

J : รูปแบบของตัวไอซี

N = ชนิดพลาสติก

J = ชนิดเซรามิค

F = ชนิดที่มีลักษณะแบน

883B : กระบวนการ Hi-Rel ที่มีให้เลือก

883B MIL-STD-883

METHOD 5004 & 5005 LEVEL B

883C MIL-STD-883

METHOD 5004 & 5005 LEVEL C

B = MIL-STD-883

METHOD 5004 EQUIVALENT

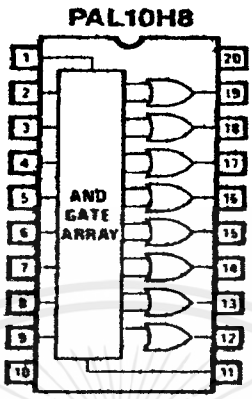
C = MIL-STD-883

METHOD 5004 EQUIVALENT

PO1234 : จำนวนรูปแบบของบิท

2.1.17 สัญลักษณ์ของโลจิก PAL

สัญลักษณ์ของแต่ละชนิดของอุปกรณ์ PAL เป็นการบรรยายการทำงานแบบกระทัดรัดของ PAL สัญลักษณ์สร้างความสะดวกในการเลือกใช้ PAL ที่ดีที่สุด ระบุข้อมูลในการประยุกต์ใช้งาน ดังรูปที่ 2.1.15 แสดงให้เห็นสัญลักษณ์ของ PAL 10H8 พร้อมกระบวนการของวงจรเกท



รูปที่ 2.1.15

2.1.18 ข้อได้เปรียบของการใช้ PAL

PAL มีลักษณะเฉพาะอย่างในโลกของการออกแบบโลจิก ไม่เท่านั้นมันยังมีข้อได้เปรียบมากมายเหนือกว่าการใช้โลจิกโดยทั่วไป ด้วยลักษณะมากมายที่ไม่สามารถพบที่ใดอีกของตระกูล PAL

- ใช้การโปรแกรมแทนที่โลจิกของ TTL ทั่วๆไป
- ลดความสำคัญของไอซีและง่ายต่อการควบคุม
- ลดไอซีของวงจรนับให้น้อยที่สุดคือ 4 ถึง 1
- ความสะดวกและปราศจากสิ่งขัดขวางของเครื่องต้นแบบและโครงการ
- ขนาดของรูปร่างไอซีจะบางและมีขนาด 24 ขาและ 20 ขา
- ความเร็วสูง , ขนาดช่วงหน่วงเวลาที่ใช้กันแพร่หลายคือขนาด 20 nS
- โปรแกรมตามมาตราฐานของ PROM
- โปรแกรมเอาท์พุท 3 สถานะ
- ลักษณะพิเศษที่สามารถที่นำออกมาได้โดยการลอกเลียนแบบของคู่แข่ง

ลักษณะทั้งหมดเหล่านี้รวมกันอยู่ในการพัฒนาการผลิตให้ต่ำลง และเพิ่มผลกระทบทางราคาของการผลิต ซึ่งเป็นวิธีการประหยัดเงินของ PAL

2.1.19 การแทนที่ลอจิกโดยตรง

ทั้งใหม่และที่มีอยู่แล้วของการออกแบบ PAL สามารถใช้แทนการกระทำทางตรรกหลายๆวงจรได้ นี่คือการยอมให้นักออกแบบมองวงจรในแง่ดีในหลายๆวิธีที่ไม่เคยทำมาก่อนที่จะเป็นไปได้ PAL เป็นผลกระทบในกรณีพิเศษเมื่อใช้กำหนดเมื่อต้องการเชื่อมต่อโดยอาศัยการกระทำหลายๆวิธีของ LSI ข้อเสียเปรียบของ PAL ถูกรวมด้วยความหนาแน่นของการกระทำ LSI ทำให้เกิดพลังงานในการรวมกัน

2.1.20 การเปลี่ยนแปลงของ PAL

PALเสนอให้ระบบการออกแบบของลอจิกขณะที่โลกใหม่ของทางเลือกจนกระทั่งปัจจุบัน การตกลงใจบนระบบเครื่องมือทางลอจิกโดยธรรมดาระหว่างการกระทำของ SSI / MSI บนมือหนึ่งและไมโครโปรเซสเซอร์ของอันอื่นๆ ในการกระทำหลายๆกรณีอาจจะเชื่องช้าเกินไปของการตอบสนองต่อวิธีแรก และง่ายเกินไปต่อการสนับสนุนวิธีที่ 2 ขณะที่ PAL ที่ถูกนำเสนอต่อนักออกแบบจะมีความหนาแน่นของการกระทำสูง, ความเร็วสูง และราคาต่ำ ถึงแม้ว่าจะดีกว่า, PAL เริ่มนิยมที่จะเปลี่ยนแปลงขนาดและการกระทำด้วยเหตุนี้เป็นการส่งเสริมให้ทางเลือกของนักออกแบบเพิ่มขึ้น

2.1.21 *การกระทำต่อพื้นที่ว่าง*

โดยการยอมให้นักออกแบบแทนที่การกระทำของลอจิกง่ายๆหลายๆอย่างด้วยไอซีตัวเดียว PAL กระทัดรัดกว่าแผนงานของบอร์ด PC ขนาดของ PAL ขนาด 20 ขามีขนาดบางช่วยในการลดขนาดของพื้นที่ของบอร์ด ในขณะที่การออกแบบบอร์ดจะง่ายลงรวมถึงการประดิษฐ์ที่ได้จากเครื่องจักร นี่หมายความว่าการ์ดหลายๆชนิดสามารถลดให้เหลือ 1 หรือ 2 การ์ดได้ และนั่นคือการสร้างความแตกต่างระหว่างผลสำเร็จทางกำไร และความสูญเสียที่เกิดจากความแพง

2.1.22 รายการอุปกรณ์ที่น้อยกว่า

ตระกูล PAL การทำงานใช้แทนที่ขึ้นถึง 90% ของการรวมเอาตระกูล TTL เพียง 25 ส่วน การลดลงอย่างมากมายนี้ทั้งอุปกรณ์ที่ใช้ทำ และต้องการรายการสิ่งของ ถึงแม้ว่าจะดีกว่า, การปรับปรุงเพียงเล็กน้อยเพื่อการกระทำให้เป็นมาตราฐานซึ่งง่ายต่อผู้ใช้ที PAL แต่ไม่ง่ายต่อมาตราฐานของผู้ใช้ TTL

2.1.23 ความเร็วสูง

ตระกูล PAL การทำงานของโปรแกรมเร็วกว่าหรือเท่าๆกับ วงจรโลจิกไบโพล่าร์ที่ดีที่สุด นี่ทำให้ทางเลือกทางการทำงานทางโลจิกมีมากที่สุด หรือควบคุมผลที่ตามมา ซึ่งต้องการความซับซ้อนขนาดกลางและความเร็วสูง ด้วยในระบบไมโครโปรเซสเซอร์หลายๆระบบ PAL สามารถความเร็วสูงของข้อมูลต่อเชื่อม นั่นคือจะใช้ไม่ได้ผลกับไมโครโปรเซสเซอร์ตัวเดียว นี่สามารถใช้สำหรับขยายความสามารถของราคาต่ำ,ความเร็วต่ำของไมโครโปรเซสเซอร์แบบ NMOS ออกไปยังบริเวณความต้องการของไมโครโปรเซสเซอร์แบบไบโพล่าร์ที่มีราคาสูงแบบก่อน

2.1.24 ความง่ายต่อการโปรแกรม

สมาชิกของตระกูล PAL สามารถโปรแกรมเร็วและง่ายด้วยการใช้ PROM มาตราฐานมาโปรแกรม นี่คือการยอมรับของนักออกแบบที่ใช้ PAL ด้วยการลงทุนที่ต่ำที่สุดและการจัดเตรียมเป็นกรณีพิเศษ หลายๆชนิดของโลจิกที่สามารถออกแบบได้ ดังเช่น FPLA ต้องให้ราคาแพงกับนักออกแบบ

2.1.25 ความปลอดภัยของข้อมูล

การพิสูจน์ข้อเท็จจริงของ PAL สามารถทำให้ไม่เสร็จสมบูรณ์โดยการระเบิดของการเชื่อมครั้งสุดท้าย นี่คือการป้องกันไม่ให้เกิดการลอกเลียนแบบค่าของข้อมูลได้ และทำให้ PAL มีความสมบูรณ์ในการใช้ประยุกต์ใช้งาน ซึ่งข้อมูลจะไม่สามารถถูกโจรกรรมได้ ซึ่งป้องกันได้อย่างปลอดภัย

2.1.26 สรุป

สมาชิกทั้ง 25 ตัวของตระกูล PAL ของอุปกรณ์ทางโลจิกซึ่งเป็นทางเลือกใหม่ของนักออกแบบในการกระทำของอุปกรณ์ทางโลจิกที่ออกแบบที่เกี่ยวกับการเรียงลำดับและการรวมกันของโลจิกตระกูลนี้จะเร็ว,กระทัดรัด,บอบบาง และง่ายต่อการใช้ ทั้งการออกแบบใหม่และที่มีอยู่แล้ว มันสามารถที่จะลดราคาในทุกๆบริเวณที่ออกแบบ และผลิต ด้วยการสอดคล้องกับการเพิ่มการผลิตซึ่งทำให้เกิดกำไรได้

2.2 การทำงานของระบบฮาร์ดแวร์

ระบบฮาร์แวร์ได้จัดตำแหน่งหมายเลขพอร์ตไว้ เพื่อเป็นการกำหนดตำแหน่งที่แน่นอนของชิปซัพพอร์ตในการใช้งาน และในการอ้างแอดเดรสของโปรแกรม ซึ่งได้จัดตำแหน่งไว้ดังแสดงในตารางที่ 2.2.1

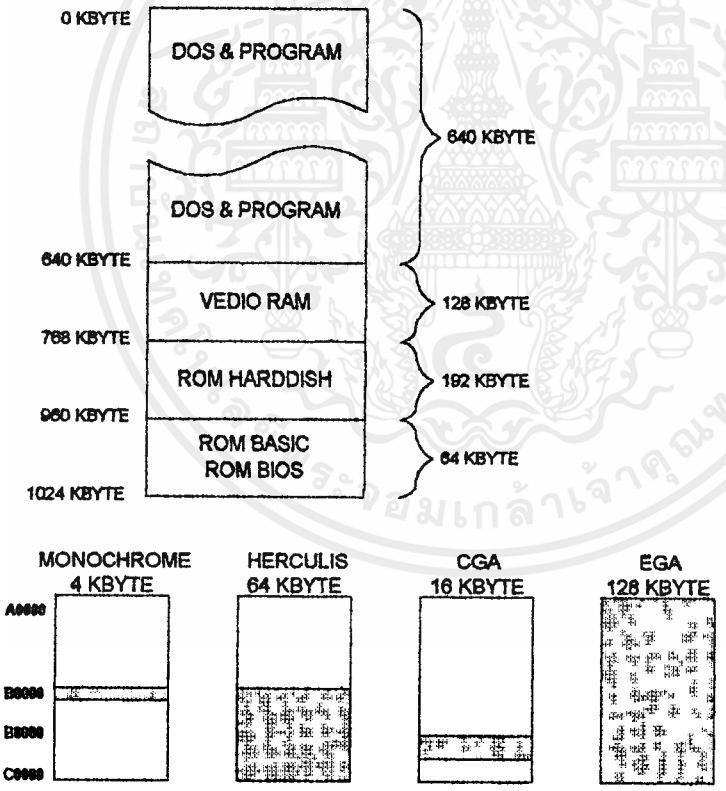
ตารางที่ 2.2.1 แสดงการจัดตำแหน่งหมายเลขพอร์ตของระบบฮาร์ดแวร์

หมายเลขพอร์ต	การใช้งาน
000-00F	ตัวควบคุมดีเอ็มเอ 8237A-5
020-021	ตัวควบคุมการอินเตอร์พัพต์ 8259-A
040-043	ไทม์เมอร์ 8253
060-063	ตัวควบคุมพอร์ต I/O 8255
080-083	ดีเอ็มเอเพจรีจิสเตอร์
0AX	NMI
0CX	สงวนไว้
0EX	สงวนไว้
200-20F	ส่วนควบคุมพอร์ตเกม
210-217	ส่วนขยาย
220-24F	สงวนไว้
278-27F	สงวนไว้
2F0-2F7	สงวนไว้
2F8-2FF	พอร์ตสื่อสาร 2 (com 2)
300-31F	การ์ดโปรโตไทร์
320-32F	ส่วนควบคุมฮาร์ดดิสค์
378-37F	เครื่องพิมพ์แบบขนาน
380-38F	พอร์ตสื่อสาร SDLC
3A0-3AF	สงวนไว้
3B0-3BF	พอร์ตแสดงผลจอภาพ/เครื่องพิมพ์
3C0-3CF	สงวนไว้
3D0-3DF	พอร์ตแสดงผลจอภาพแบบสี/กราฟฟิก
3E0-3EF	สงวนไว้
3F0-3F7	พอร์ตควบคุมดิสค์ไดร์ฟ
3F8-3FF	พอร์ตสื่อสาร 1 (com 1)

เราสามารถอ่านข้อมูลจากพอร์ตต่าง ๆ บนเมนบอร์ดของระบบฮาร์ดแวร์ได้ โดยใช้คำสั่งจากภาษาแอสแซมบลี 8080 เช่น IH AL,61 เป็นการอ่านค่าจากพอร์ต 61 มาสู่รีจิสเตอร์ AL หากต้องการจะส่งข้อมูลให้กับพอร์ต 61 ก็ต้องใช้คำสั่ง OUT,62 AL จะเป็นการนำข้อมูลจากรีจิสเตอร์ AL มาให้แก่พอร์ต 61 โดยตำแหน่งหมายเลขพอร์ตบนเมนพอร์ดจะอยู่ที่ 000H-01FFH และอีกส่วนคือ ตำแหน่งพอร์ตที่อยู่บนการ์ดต่าง ๆ เริ่มจากตำแหน่ง 0200H-03FFH

2.2.1 การใช้งานหน่วยความจำ

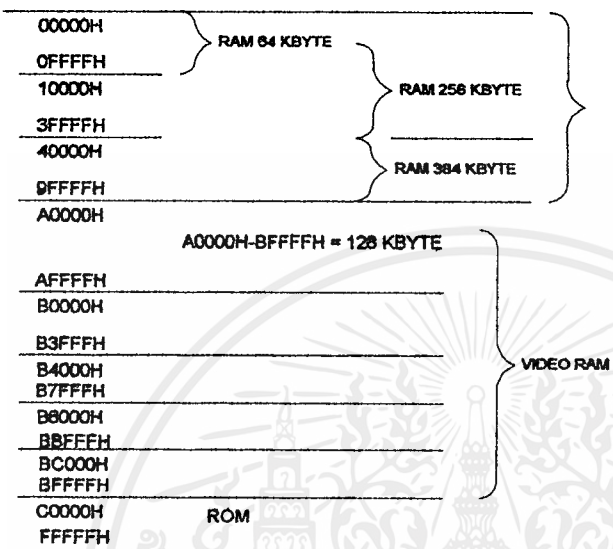
ระบบฮาร์ดแวร์ได้แบ่งหน่วยความจำออกเป็น 2 ส่วนคือ รอมหรือ ROM(read only memory) และ แรม หรือ RAM (random access memory) โดยรอมจะมีตำแหน่งแอดเดรสที่ $0F0000_H$-$0FFFFF_H$ ส่วนแรมจะเริ่มจากตำแหน่ง 00000_H-$9FFFFF_H$ ในส่วนของรอมจะมีอยู่ 2 ตัวที่จะใช้งานในระบบคือรอมเบสิก และรอมไบออส ซึ่งรอมเบสิกจะอยู่ตำแหน่งแอดเดรสที่ $F6000_H$-$FDFFF_H$ ความจุ 32 กิโลไบต์ และรอมไบออสจะอยู่ตำแหน่งแอดเดรสที่ $FE000_H$-$FFFFF_H$ ความจุ 8 กิโลไบต์ ส่วนทางด้านแรมนั้นจะมีตำแหน่งแอดเดรสที่ 00000_H-$3FFFF_H$ ความจุ 256 กิโลไบต์ และ 00000_H-$9FFFF_H$ ในกรณีความจุ 640 กิโลไบต์ ทั้งหมดนี้เป็นหน่วยความจำบนเมนบอร์ด หน่วยความจำที่ใช้งานบนการ์ด คือ ดิสเพลย์บัฟเฟอร์ (display buffer) มีตำแหน่งแอดเดรสที่ $B0000_H$-$B7FFF_H$ (โมโนโครม) และแอดเดรส $B8000_H$-$BFFFF_H$ (สี/กราฟฟิก) ระบบฮาร์แวร์ได้จัดตำแหน่งของหน่วยความจำ 1 เมกะไบต์ แต่ในระบบฮาร์แวร์นั้นไม่สามารถนำหน่วยความจำทั้งหมด 1 เมกะไบต์มาใช้งานสำหรับโปรแกรมต่าง ๆ เพราะต้องเผื่อเนื้อที่บางส่วนไว้สำหรับวิดีโอแรม (video ram) สำหรับแสดงผลทางจอภาพ รอมเบสิก(โปรแกรมภาษาเบสิก) และรอมไบออส (โปรแกรมควบคุมการทำงานของฮาร์ดแวร์)



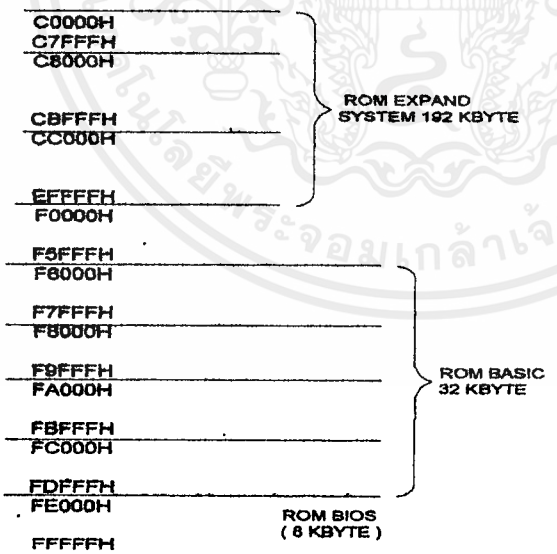
รูปที่ 2.2.1 แสดงตำแหน่งหน่วยความจำ และวิดีโอแรม

จากรูปที่ 2.2.1 จะเห็นว่าเนื้อที่ 640 กิโลไบต์แรก เป็นส่วนของโปรแกรมดอส และโปรแกรมใช้งานทั่ว ๆ ไป ตลอดจนโปรแกรมถาวร (resident program) ถัดมาอีก 128 กิโลไบต์คือ 768 กิโลไบต์ เป็นส่วนของวิดีโอแรม ถ้าหากใช้การ์ดแสดงผลโมโนโครมธรรมดา จะใช้เนื้อที่แค่ 4 ถึง 8 กิโลไบต์ (704-708) ถ้าเป็น

การ์ดเฮอร์คอวลิส (hercules) ซึ่งแสดงผลบนจอโมโนโครม และแสดงกราฟฟิกได้ จะใช้เนื้อที่ 64 กิโลไบต์ (704-768 กิโลไบต์) ส่วนการ์ด CGA (color graphic adapter) จะใช้เนื้อที่ 16 กิโลไบต์ (736-752 กิโลไบต์) และถ้าเป็นการ์ด EGA (enhanced graphics adapter) จะใช้เนื้อที่เต็ม 128 กิโลไบต์เลย จากวิดีโอแรมขึ้นมาอีก 192 กิโลไบต์ เป็นที่สำหรับหน่วยความจำรอมไบออสของฮาร์ดดิสก์ ส่วนเนื้อที่สุดท้ายจาก 960 กิโลไบต์ เป็นที่สำหรับหน่วยความจำรอมเบสิก และรอมไบออสของระบบ



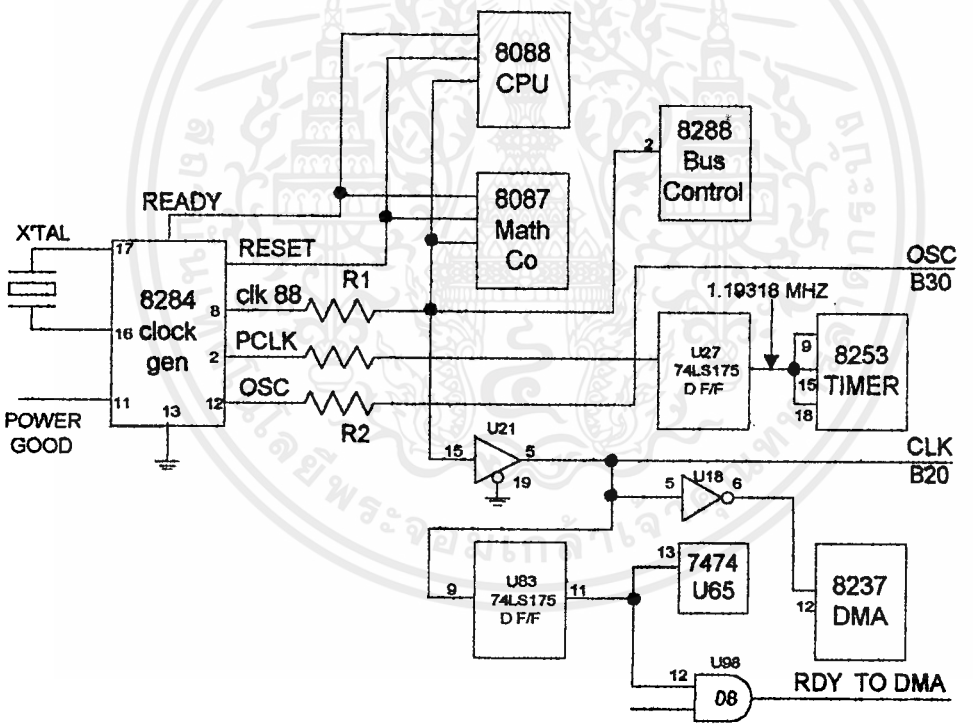
รูปที่ 2.2.2 แสดงตารางหน่วยความจำของระบบฮาร์ดแวร์



รูปที่ 2.2.3 แสดงตำแหน่งหน่วยความจำรอม

2.2.2 การทำงานของวงจรผลิตสัญญาณนาฬิกา

ระบบฮาร์แวร์จะทำงานได้นั้นจะต้องมีสัญญาณนาฬิกา มาป้อนให้กับตัวอุปกรณ์แต่ละตัวบนบอร์ด ตัวผลิตสัญญาณนาฬิกาคือ ไอซีเบอร์ 8284 (รูปที่ 2.2.4 จะกำเนิดสัญญาณนาฬิกาโดยใช้คริสตอล 14.31818 เมกะเฮิรตซ์เป็นออสซิลเลเตอร์ ทำให้ได้เอาต์พุต สัญญาณนาฬิกา 3 สัญญาณคือ OSC (14.31818mhz) CLK88 (4.77 MHz) และ PCLK(2.386 MHz) ป้อนให้อุปกรณ์บนเมนบอร์ต จะเห็นว่าขา 13 ของ 8284 ต่อบงกราวด์เพื่อให้ผลิตความถี่ตามคริสตอล 14.31818 MHz หากจะให้เป็นเทอร์โบ 8-10 MHz ขา 13 ต้องเป็น " 1 " ส่วน R1-R2 นั้นจะทำให้ความถี่ของคริสตอลคงที่ในการ อ้างอิงสัญญาณแก่ 8284 ความถี่ 14.31818 MHz ที่ออกมาจากขา 12 ของ 8284 เป็นความถี่ที่มีค่าสูงสุดบนเมนบอร์ดเรียกว่า OSC จะใช้งานร่วมกับการ์ดแสดงผลที่เป็นสี-กราฟฟิก และเอาต์พุตของ 8284 ขา 8 คือ CLK 4.77 MHz ป้อนให้แก่ซีพียู 8088 ไมโครโปรเซสเซอร์ 8087 8288 และ 74LS175 (U27) และเอาต์พุตจาก 74LS175 คือความถี่ 1.19318 เมกะเฮิรตซ์ ป้อนให้แก่ขา clock 0.1.2 ของ 8253 (ไทม์เมอร์) สัญญาณนาฬิกาทั้ง 3 คือ OSC,PLCK และ CLK88 จะต้องผ่านตัวต้านทาน 27 โอห์ม ในการเชื่อมต่อกับอุปกรณ์เหล่านั้น

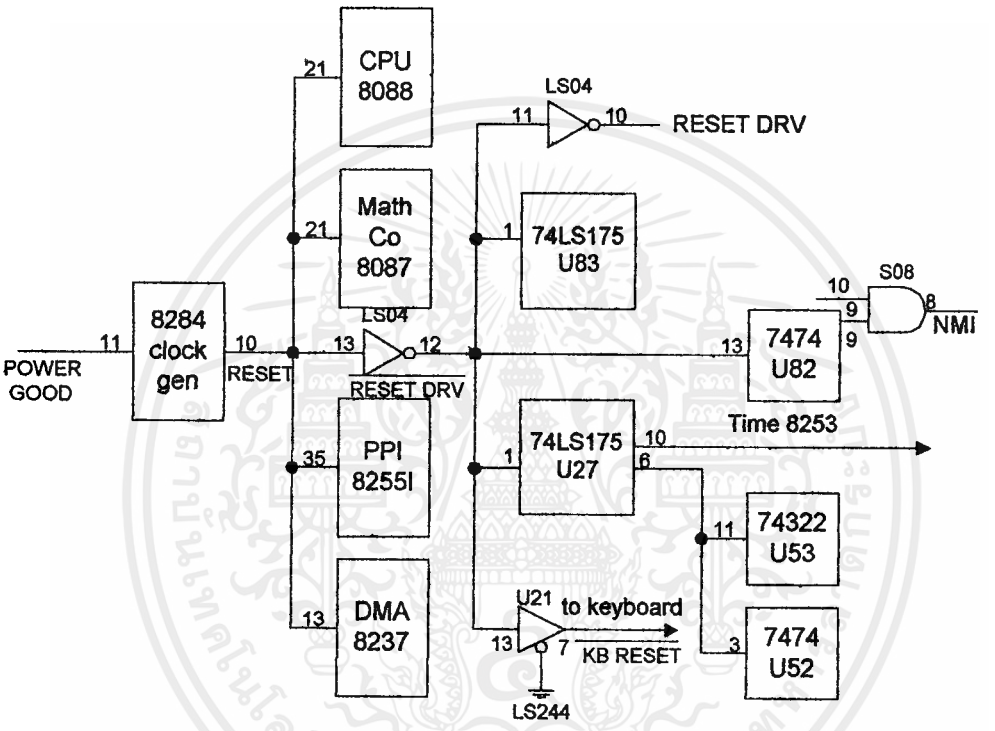


รูปที่ 2.2.4 วงจรสัญญาณนาฬิกาของระบบฮาร์ดแวร์

สัญญาณ Power Good จะป้อนให้แก่ 8284 ทางขา 11 และสัญญาณ Power Good ที่เป็นอินพุตให้ 8284 จะต้องไม่ต่ำกว่า 4.75 โวลต์ หากสัญญาณนี้ต่ำกว่าที่กำหนดไว้จะทำให้ 8284 ส่ง สัญญาณรีเซตให้แก่ 8088 ทำให้รีจิสเตอร์ภายในเป็น 0 และ CS (โค้ดเซกเมนต์) จะมีตำแหน่งแอดเดรสที่ $0FFFF_H$ ซึ่งเป็นแอดเดรสเริ่มต้นของรอมไบออส สัญญาณรีเซตนี้จะส่งไปให้แก่ 8255,8087,8237A-5 ด้วย เพื่อเคลียร์

รีจิสเตอร์ และฟลิปฟลอปภายในชิปซัพพอร์ตเหล่านี้ รวมทั้งยังส่งสัญญาณรีเซตไปให้แก่ฟลิปฟลอป และอินเวอร์เตอร์

จากรูปที่ 2.2.5 จะสังเกตเห็นว่าเมื่อสัญญาณรีเซตจาก 8284 ผ่าน 74LS07 (U18) ตัวแรกจะได้เอาท์พุทเป็นสัญญาณ $\overline{RESET\ DRV}$ เพื่อส่งไปเคลียร์ฟลิปฟลอป 74LS175 (U83,27) และ 74LS74 (U82) และสัญญาณ $\overline{RESET\ DRV}$ นี้ยังผ่าน 74LS04 (U18) อีกตัวทางขา 11 ได้เอาท์พุทเป็น RESET DRV เพื่อส่งสัญญาณนี้ออกสู่สล็อต B2 แล้วผ่านเข้าไปบนการ์ดโมโนโครมทำการเคลียร์ F/F 74LS175 บนการ์ดหมายเลขพอร์ต 3B8,3BF และรีเซต 6845 บนการ์ดด้วย

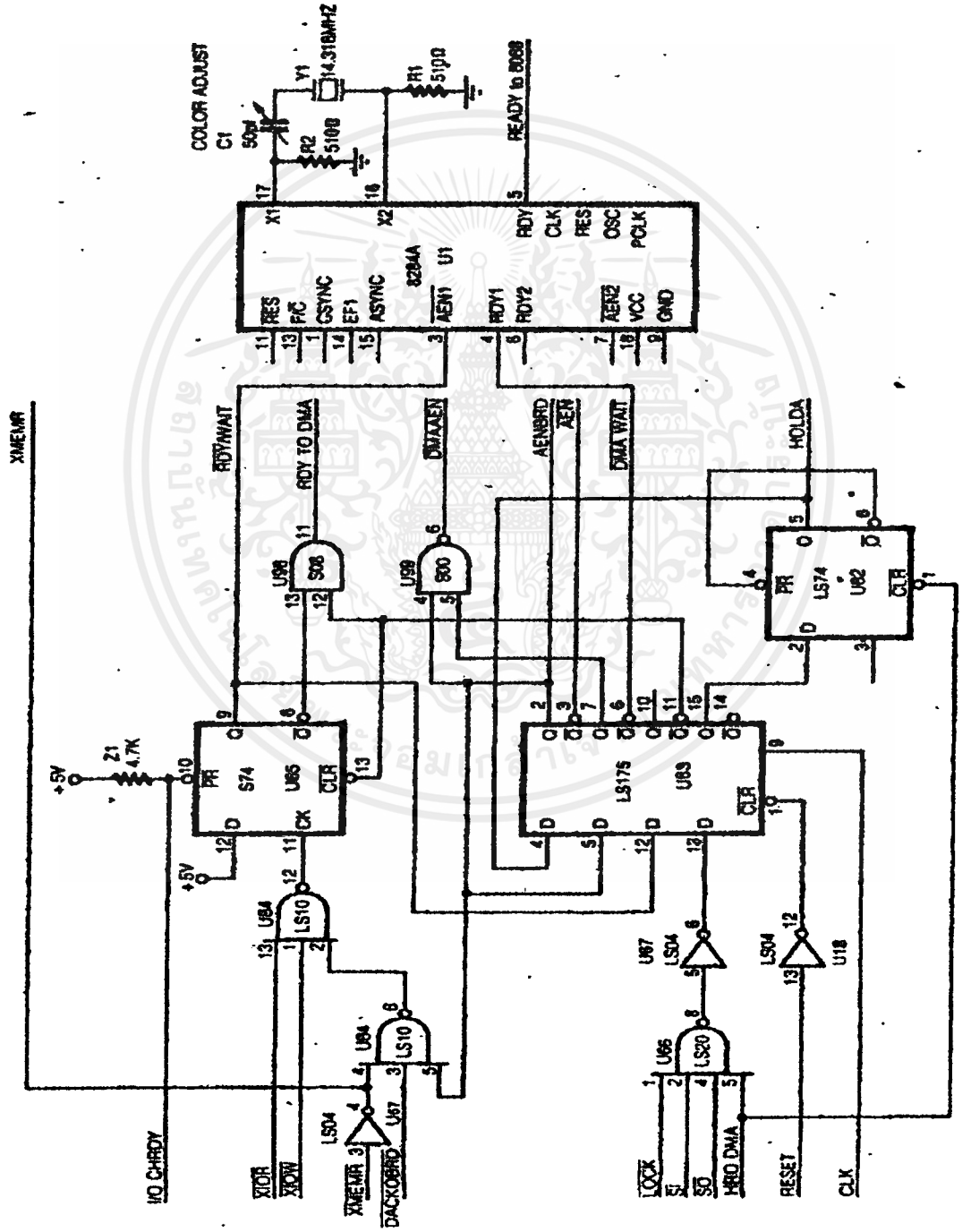


รูปที่ 2.2.5 สัญญาณรีเซตจาก 8284 กำเนิดสัญญาณนาฬิกา

จากรูปที่ 2.2.5 จะสังเกตเห็นว่าเมื่อสัญญาณรีเซตจาก 8284 ผ่าน 74LS04 (U18) ตัวแรกจะได้เอาท์พุทเป็นสัญญาณ $\overline{RESET\ DRV}$ เพื่อส่งไปเคลียร์ฟลิปฟลอป 74LS175 (U83,27) และ 74LS74 (U82) และสัญญาณ $\overline{RESET\ DRV}$ นี้ยังผ่าน 74LS04 (U18) อีกตัวทางขา 11 ได้เอาท์พุทเป็น RESET DRV เพื่อส่งสัญญาณนี้ออกสู่สล็อต B2 แล้วผ่านเข้าไปบนการ์ดโมโนโครมทำการเคลียร์ F/F 74LS175 บนการ์ดหมายเลขพอร์ต 3B8,3BF และรีเซต 6845 บนการ์ดด้วย

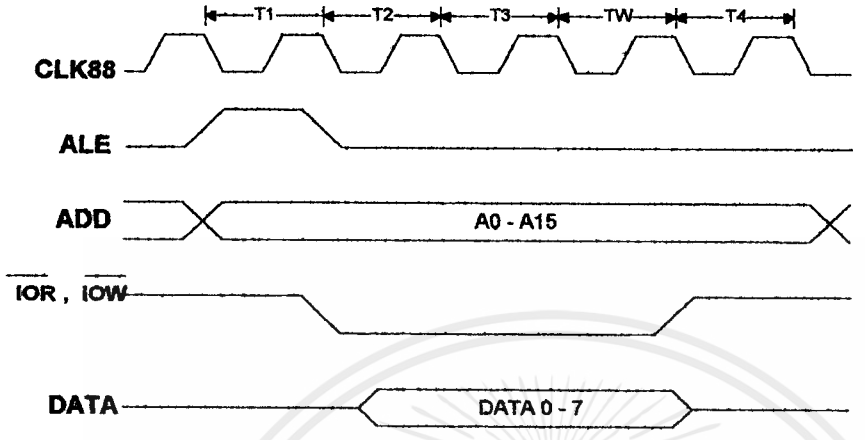
นอกจากนี้ 8284 ยังมีหน้าที่ในการควบคุมสัญญาณ READY และ wait ให้แก่ซีพียู 8088 โดยทั่วไปแล้วการทำงานของซีพียูจะเรียกว่า บัสไซเกิล หมายถึง ขั้นตอนของสัญญาณที่เกิดขึ้น ในขณะมีการส่งผ่านข้อมูลระหว่างอุปกรณ์อินพุทเอาท์พุท หน่วยความจำ และซีพียู ดังนั้นการทำงานของซีพียูในแต่ละบัสไซเกิลนั้นจะอาศัยเวลาในการทำงานเท่ากับเวลาของสัญญาณนาฬิกา 4.77 เมกะเฮิรตซ์ที่ป้อนให้ซีพียูถึง 4 ลูกด้วยกัน ใช้

คาบเวลาของสัญญาณนาฬิกา 1 ลูก ประมาณ 210 นาโนวินาที ฉะนั้น ใน 1 บัสไซเกิลจะเท่ากับ 4x210 นาโนวินาที (840 นาโนวินาที) หรือใช้ช่วงเวลาตั้งแต่ T1,T2,T3 และ T4 (Tx=210 นาโนวินาที) เมื่ออุปกรณ์ทำงานร่วมกับซีพียูทำงานช้ากว่ามาก ไม่สามารถตอบสนอง การทำงานตามบัสไซเกิล ของซีพียูได้ อุปกรณ์เหล่านั้นก็จะสามารถจะเพิ่มช่วงเวลาในการทำงานจาก T1-T4 เป็น T1-T5 ช่วงเวลาที่เพิ่มเข้ามานี้คือ Tw (Time Wait) ทำให้บัสไซเกิลเพิ่มช่วงเวลาเป็น 5x210 เท่ากับ 1.05 ไมโครวินาที วงจร Wait state ที่กล่าวถึงนี้มีลักษณะวงจรดังแสดงดังรูปที่ 2.2.6



รูปที่ 2.2.6 วงจร Wait state

ตามปกติแล้ว บัสไซเกิลของการอ่าน หรือเขียนข้อมูลลงหน่วยความจำ จะไม่มี Tw เกิดขึ้น เพราะว่าหน่วยความจำมีความเร็วพอในการตอบสนองต่อบัสไซเกิล แต่ในกรณีหน่วยความจำบนการ์ดคือ ดิสเพลย์ บัฟเฟอร์ จะต้องมี wait state (รูปที่ 2.2.7) ให้ด้วย

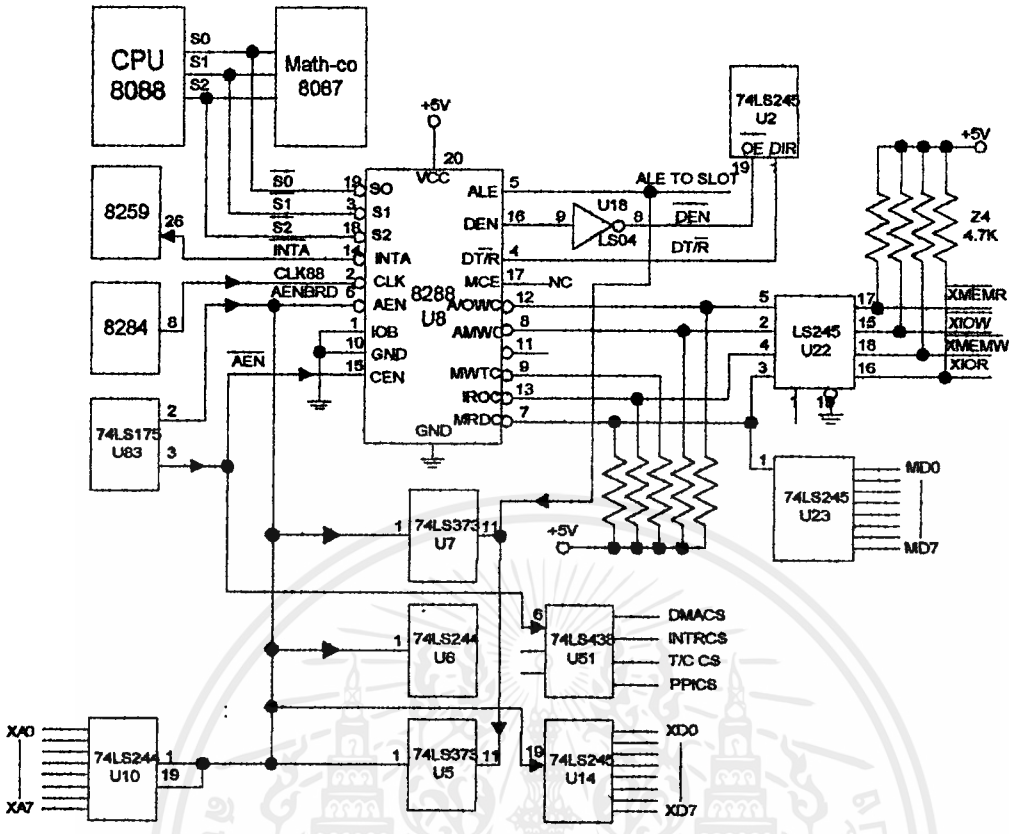


รูปที่ 2.2.7 บัสไซเกิลของซีพียูที่มี Tw

วงจร wait state นี้จะคอยตรวจสอบสัญญาณ I/O CH RDY เสมอว่ามีระดับสัญญาณเป็น "0" หรือไม่ หากมีเมื่อใดจะเป็นการสร้างสัญญาณ wait หรือ Tw ขึ้น 1 ลูกของบัสไซเกิลเพื่อให้ซีพียูอยู่ในสภาวะ wait ก่อน จากนั้นเมื่อหมด Tw แล้ว ซีพียูจะได้รับสัญญาณ $\overline{RDY}$ จาก 8284 หลังจากจบบัสไซเกิลนั้น ๆ

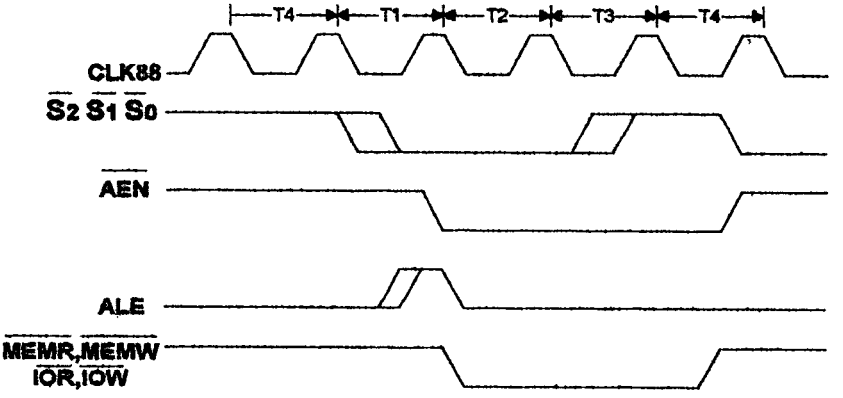
2.2.3 การจัดระบบบัส

ในการจัดระบบบัสและสัญญาณควบคุมซิสเต็มบอร์ดนั้นจะใช้ 8288 (ดูรูปที่ 2.2.8) บัสคอนโทรลเลอร์ เป็นตัวควบคุมทั้งหมด มี อินพุต 7 เส้น โดยให้ขา 1 (IOB) ต่อลงกราวน์ เพื่อทำงานในโหมดซิสเต็มบัสและสามารถจะหยุดการทำงานของ 8288 ได้โดยให้สัญญาณ $\overline{AEN}$ (แอดเดรส อีนาเบิล) ที่ขา 15 แอคตีฟ เมื่อหยุดการทำงานของ 8288 แล้ว สัญญาณ $\overline{AEN}$ จะส่งไปขา 1 ของ 74LS138 (U51) ทำให้ 74LS138 ไม่ทำงานด้วยและ 8237A-5 จะควบคุมระบบบัส ทำกระบวนการ DMA ต่อไป สัญญาณ S0,S1,S2 จากซีพียู จะถูกส่งเข้าสู่ 8288 เพื่อทำการถอดรหัส และควบคุมสัญญาณนั้น ยังมีสัญญาณสำคัญ ที่เป็นอินพุตแก่ 8288 คือ AENBRD จาก 74LS175 (U83) ขา 2 ป้อนให้ 8288 ขา 6 เพื่อให้ควบคุมสัญญาณเอาต์พุทของ 8288 เอง และสัญญาณ AENBRD ก็ยังส่งไปให้แก่ 74LS373 (U5,U7) (octal transparent latches) และ 74LS244 (U6) (octal buffer) อีกด้วย



รูปที่ 2.2.8 สัญญาณการจัดระบบบัสของ 8288 บัสคอนโทรลเลอร์

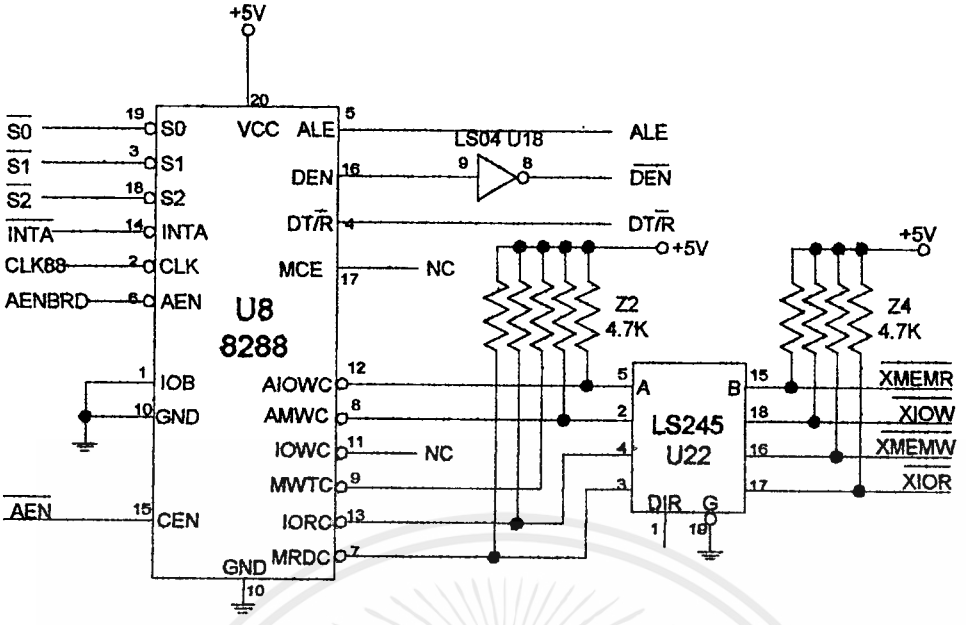
ส่วนเอาต์พุตของ 8288 ที่ใช้งานมีอยู่ 7 ขา คือ ขา 8 เป็นสัญญาณ $\overline{MEMW}$ (memory write หรือสัญญาณเขียนหน่วยความจำ) ขา 7 สัญญาณ $\overline{MEMR}$ (memory read หรือ สัญญาณอ่านหน่วยความจำ) ขา 12 (I/O write หรือสัญญาณเขียน อินพุตเอาต์พุต) ขา 13 $\overline{IOR}$ (I/O read หรือ สัญญาณอ่านอินพุตเอาต์พุต)โดยสัญญาณทั้ง 4 เส้น จะแอคตีฟโลจิก " 0 " และต่อเข้ากับ 74LS245 (U22) (ดาต้าบัสบัฟเฟอร์) เอาต์พุตขา 14 $\overline{INTR}$ (interrupt acknowage หรือสัญญาณการตอบรับอินเตอร์รัพท์ จะถูกส่งให้แก่ 8259 (โปรแกรมมิ่งอินเตอร์รัพท์) โดยสัญญาณนี้ จะเป็นตัวบอกให้ 8259 รับรู้การตอบรับการอินเตอร์รัพท์จากซีพียู สัญญาณเอาต์พุตที่ใช้ในการเชื่อมโยงกับแอดเดรสแลตช์ รับส่งข้อมูลและอินเตอร์รัพท์คอนโทรล มีดังนี้ DT/$\overline{R}$ (data transmit / receiver) จากขา 4 ของ 8288 จะเชื่อมต่อกับขา 1 ของ 74LS245 (U2) เมื่อสัญญาณ DT/$\overline{R}$ เป็น " 1" หมายความว่ากำลังส่งข้อมูลจากซีพียูผ่าน 74LS245 (U2) ไปสู่อินพุตเอาต์พุตหรือหน่วยความจำ แต่ถ้าหากสัญญาณ DT/$\overline{R}$ เป็น " 0 " หมายความว่า ซีพียูกำลังอ่านข้อมูลผ่าน 74LS245 (U2) เข้าไป สัญญาณ ALE (แอดเดรสแลตช์ อีนาเบิล) จะใช้ในการสโตรบแอดเดรสของ 74LS373 (U5,U7) เมื่อสัญญาณนี้เปลี่ยนจาก " 1 " เป็น " 0 " ณ ขอบขาลงของสัญญาณ ALE (รูปที่ 2.2.9) จะเป็นการ แลตช์แอดเดรสจากบัสแอดเดรสจากบัสข้อมูล (AD0-AD7) ของซีพียู เป็นการแยกแอดเดรส (A0-A19) และข้อมูล (AD0-AD7) ออกจากกัน แต่เมื่ออยู่ในขณะกระทำขบวนการ DMA แล้ว สัญญาณนี้จะไม่แอคตีฟ



รูปที่ 2.2.9 บัสไซเกิลของสัญญาณ ALE

สัญญาณสุดท้ายที่จะกล่าวถึงคือ สัญญาณ DEN (ดาต้าอีนาเบิล) จะต่อเข้ากับขา 19 ของ 74LS245 (U2) โดยผ่านอินเวอร์เตอร์เพื่อกลับลอจิก ของสัญญาณจาก 8288 ก่อน สัญญาณนี้จะเป็นตัวอินาเบิลให้ข้อมูลเข้าออกได้

วงจรควบคุมระบบบัสโดย 8288 (บัสคอนโทรลเลอร์) นี้ จะเห็นว่าสัญญาณ $\overline{MEMR}$,$\overline{MEMW}$,$\overline{IOR}$ และ $\overline{IOW}$ ที่ออกจาก 8288 จะต้องผ่าน IC 74LS245 (U22) ได้ เอาต์พุตเป็นสัญญาณ $\overline{XMEMR}$, $\overline{XMEMW}$, $\overline{XIOR}$ และ $\overline{XIOW}$ เพื่อส่งให้แก่อุปกรณ์ต่าง ๆ บน ซิสเต็มบอร์ด อุปกรณ์ทุกตัวบนซิสเต็มบอร์ดที่ต้องใช้สัญญาณเหล่านี้ จะได้รับเฉพาะสัญญาณ $\overline{XMEMR}$, $\overline{XIOR}$ และ $\overline{XIOW}$ เท่านั้น (ขณะที่ซีพียูกำลังทำงานอยู่) ไม่โอกาสได้รับสัญญาณ $\overline{MEMR}$,$\overline{MEMW}$,$\overline{IOR}$ และ $\overline{IOW}$ แต่จะเห็นว่ามีสัญญาณ $\overline{MEMR}$,$\overline{MEMW}$,$\overline{IOR}$ และ $\overline{IOW}$ ถูกส่งออกมายังสล็อตทั้ง 8 ของฮาร์ดแวร์ที่ขา B11-B14 เพื่อส่งสัญญาณนี้ให้แก่วงจรอินเตอร์เฟส หรืออุปกรณ์อินพุตเอาต์พุตบนการ์ดเท่านั้น รูปที่ 2.2.10 แสดงสัญญาณควบคุมจาก 8288 ออกสู่ระบบ



รูปที่ 2.2.10 แสดงสัญญาณควบคุมจาก 8288 ออกสู่ระบบ

2.3 หลักการในการจัดหน่วยความจำแบบไดนามิก

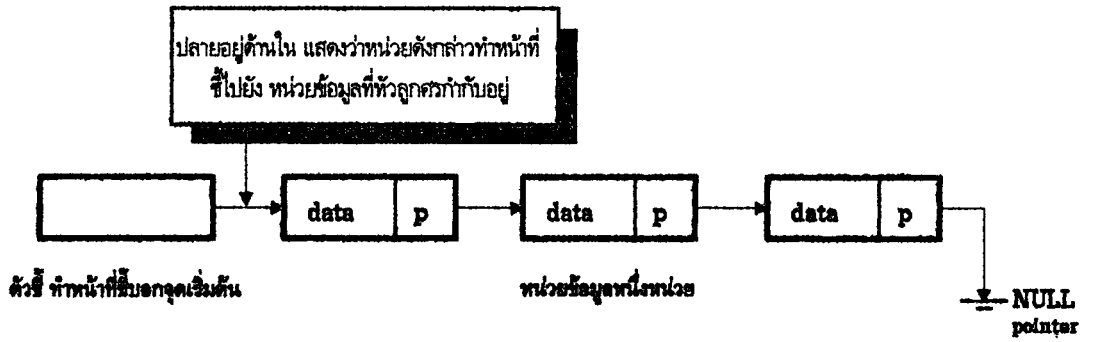
2.3.1 การจัดการกลุ่มข้อมูลแบบไดนามิก

สำหรับโครงสร้างข้อมูลหนึ่งโครงสร้างจะมีหน่วยข้อมูลเพียงหน่วยเดียวแต่หน่วยข้อมูลนั้นอาจจะมีสมาชิกมากกว่าหนึ่งตัวก็ได้ การจองพื้นที่ข้อมูลทั้งโครงสร้างจะกระทำไปในคราวเดียว ซึ่งจะมีข้อเสียในกรณีที่ไม่ได้ใช้พื้นที่ทั้งหมดทันที ทำให้ไม่สามารถใช้หน่วยความจำได้อย่างคุ้มค่า แต่ก็มีข้อดีตรงที่เราจัดการพื้นที่ได้ง่าย ในกรณีที่การใช้ข้อมูลไม่ได้ต้องการพื้นที่จำนวนมากในทันที เราอาจจะแบ่งข้อมูลออกเป็นส่วนๆ เพื่อทยอยจองเมื่อมีความต้องการก็ได้ ในลักษณะเช่นนี้ โครงสร้างข้อมูลก็จะประกอบไปด้วยหน่วยข้อมูลแบบไดนามิกหลายๆ หน่วยเชื่อมต่อกันเป็นโครงสร้างข้อมูล การเชื่อมต่อกันนี้จะนิยมใช้ตัวชี้ซึ่งอยู่เป็นส่วนหนึ่งของหน่วยข้อมูลนั้นๆ ในการอ้างถึงตำแหน่งหน่วยข้อมูลที่อยู่เป็นหน่วยถัดไป โครงสร้างซึ่งประกอบมาจากหน่วยข้อมูลนี้มีหลายชนิด ซึ่งเราจะได้กล่าวต่อไป

สำหรับโครงสร้างข้อมูลแบบไดนามิกนั้น ได้แก่ ลิงค์ลิสต์ สแต็ก คิว และทรี (สแต็กและคิวชนิดที่มีโครงสร้างแบบไดนามิก) จะมีการจัดการกับหน่วยข้อมูลที่คล้ายคลึงกัน เพียงแต่แตกต่างกันในลักษณะการเชื่อมต่อกับหน่วยข้อมูลอื่น ลิงค์ลิสต์ สแต็ก และคิว จะมีหน่วยข้อมูลที่เรียงต่อกันเป็นสายยาว การเข้าถึงหน่วยข้อมูลของสแต็กและคิว จะกำหนดวิธีการเข้าถึงโดยอาศัยตัวชี้สแต็ก ในกรณีของสแต็ก และอาศัยตัวชี้สำหรับอ่านหรือตัวชี้สำหรับเขียน ในกรณีของคิว ตามข้อกำหนดของ สแต็กและคิว ส่วนลิงค์ลิสต์ ถือเป็นกรณีทั่วไปของโครงสร้างข้อมูลไดนามิกที่เรียงต่อกันเป็นสายยาว สำหรับโครงสร้างข้อมูลแบบทรีจะมีลักษณะที่แตกต่างจากโครงสร้างอื่นๆ ในจุดที่การเชื่อมต่อกับหน่วยข้อมูลอื่น จะมีลักษณะแยกสายย่อยออกไปได้ ในลักษณะคล้ายต้นไม้ การเข้าถึงสายย่อยๆ จะเริ่มมาจากปลายใหญ่ที่มีลักษณะเป็นรากของโครงสร้าง ซึ่งจะได้กล่าวโดยละเอียดต่อไป

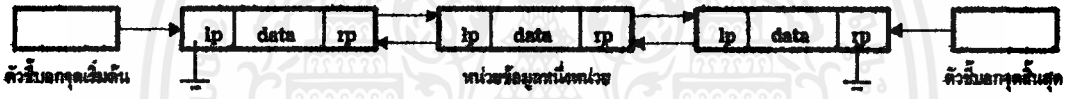
1.ลิงค์ลิสต์ (Linked Lists) คือโครงสร้างข้อมูลที่ประกอบไปด้วย หน่วยข้อมูล(Node) ที่เชื่อมต่อกันเป็นสายยาวด้วยตัวชี้ การเข้าถึงหน่วยข้อมูลแต่ละหน่วยจะอาศัยตัวชี้จากหน่วยที่อยู่ก่อนหรือถัดไป การเข้าถึงข้อมูลลิงค์ลิสต์ จำเป็นจะต้องมีตัวชี้ที่ทำหน้าที่ชี้ไปยังจุดเริ่มต้นของลิงค์ลิสต์ เพื่อให้เป็นทางเข้าไปสู่โครงสร้างข้อมูล เช่นเดียวกับโครงสร้างข้อมูลไดนามิกอื่นๆ ลิงค์ลิสต์อาจแบ่งประเภทตามจำนวนตัวชี้ไปยังหน่วยข้อมูลอื่นๆ และตามโครงสร้างได้ดังนี้

Singly Linked List ในหน่วยข้อมูลของลิงค์ลิสต์จะมีตัวชี้ไปยังหน่วยข้อมูลอื่นเพียงตัวเดียว มีโครงสร้างดังรูป



รูปที่ 2.3.1 Singly Linked List

Doubly Linked List ในหนึ่งหน่วยข้อมูลจะประกอบไปด้วยส่วนที่เป็นข้อมูล และตัวชี้ไปยังหน่วยข้อมูลอื่นเช่นเดียวกันกับ Singly Linked List แต่ในส่วนของตัวชี้ไปยังหน่วยอื่นจะมีอยู่ 2 ตัว สำหรับชี้ไปยังหน่วยข้อมูลในลำดับก่อน และหน่วยข้อมูลถัดไป การบอกตำแหน่งโครงสร้างข้อมูลอาจจะใช้ตัวชี้ 2 ตัว สำหรับชี้หน่วยข้อมูลแรก และหน่วยข้อมูลสุดท้ายในโครงสร้าง เช่น



รูปที่ 2.3.2 Doubly Linked List

Multi Linked List ในหน่วยข้อมูลจะมีตัวชี้มากกว่า 2 ตัวขึ้นไปทำให้โครงสร้างแบบนี้สามารถอ้างถึงหน่วยอื่นๆ ได้มากกว่า 1 ตัว เช่นใช้อ้างหน่วยถัดไป หน่วยก่อน หน่วยเริ่มต้น และหน่วยสุดท้ายเป็นต้น

Circular Linked List เป็นลิงค์ลิสต์ที่ตัวชี้ในหน่วยสุดท้ายจะชี้ไปยังหน่วยข้อมูลแรกในลิสต์ ดังนั้นการเข้าถึงหน่วยถัดไปจากหน่วยสุดท้ายจะทำให้เราไปถึงหน่วยข้อมูลแรก แทนที่จะชี้เข้าหาตัวเอง หรือชี้ไปที่ NULL

การจัดโครงสร้างของลิงค์ลิสต์ มีขั้นตอนการทำงานดังนี้

- การกำหนดหน่วยข้อมูลแรก
- การกำหนดหน่วยข้อมูลถัดไป หรือการต่อท้ายลิสต์ด้วยหน่วยข้อมูลใหม่
- การเข้าหาหน่วยข้อมูลแต่ละหน่วย
- การแทรก ตัด ย้ายหน่วยข้อมูล
- การลบลิสต์ทั้งลิสต์ เป็นการคืนหน่วยความจำที่จองมาทั้งหมดให้แก่ระบบ

- การกำหนดหน่วยข้อมูลหน่วยแรก

ตัวแปรทั้งหมดที่ได้กำหนดมาแล้ว ไม่ใช่ตัวเก็บข้อมูล แต่เป็นเพียงตัวชี้ไปยังหน่วยข้อมูลเท่านั้น ตัวหน่วยข้อมูลแท้จริงจะอยู่ในรูปตัวแปรไดนามิก ซึ่งจะต้องจองหน่วยความจำสำหรับหน่วยข้อมูลเสียก่อนการกำหนดหรือสร้างหน่วยข้อมูลขึ้นมาสำหรับใช้เก็บค่านั้น อาจแบ่งได้เป็น 2 ขั้นตอนคือ การกำหนดหน่วยข้อมูลหน่วยแรก และการกำหนดหน่วยข้อมูลต่อไป วิธีการทั้งสองจะแตกต่างกันอยู่เล็กน้อย การกำหนดหน่วยข้อมูลหน่วยแรกมีขั้นตอนดังนี้

1.ขอจองพื้นที่ข้อมูล 1 หน่วยตามขนาดของสตรัคเจอร์ ซึ่งคำนวณได้จากตัวกระทำ sizeof แล้วเก็บค่าตำแหน่งหน่วยความจำเริ่มต้นของหน่วยข้อมูลที่จองได้ไว้ในตัวชี้สำหรับชี้จุดเริ่มต้นและจุดสิ้นสุด

2.กำหนดให้ตัวชี้ในหน่วยข้อมูลชี้ไปยัง NULL หรือชี้เข้าหาตนเอง (ในกรณีของ Circular Linked List)

3.เมื่อกำหนดตัวหน่วยข้อมูลแรกเสร็จแล้ว ก็จะใส่ค่าข้อมูลลงในหน่วยข้อมูลแรก โดยอาศัยตัวชี้ที่กำหนดไว้สำหรับจุดเริ่มต้นของข้อมูล

- การเพิ่มหน่วยข้อมูลถัดไป (ต่อท้ายลิสต์)

หลังจากที่ได้สร้างและกำหนดหน่วยข้อมูลแรกในลิสต์แล้ว การกระทำกับหน่วยข้อมูลถัดไป ก็จะมีลักษณะเช่นเดียวกับหน่วยแรกดังนี้

1.เริ่มจากกำหนดให้ walknode มีค่าเท่ากับ headnode นั้นคือให้ walknode ชี้ไปยังจุดเริ่มต้นของลิสต์

2.ตรวจสอบค่าในตัวชี้ ของหน่วยข้อมูลว่าสิ้นสุดหรือไม่ หากยังไม่สิ้นสุดให้เลื่อนตัวชี้ walknode ไปยังตำแหน่งหน่วยข้อมูล ถัด ไป และตรวจสอบจนกว่า ตัวชี้ walknode จะชี้ที่หน่วยข้อมูลสุดท้าย

3.เมื่อพบหน่วยข้อมูลสุดท้าย เราก็จะสร้างหน่วยข้อมูลขึ้นอีก 1 หน่วย ใส่ค่าข้อมูล และกำหนดตัวชี้ในหน่วยข้อมูลใหม่นี้ชี้ไปที่ NULL ในกรณีของ Doubly Linked List จะกำหนดให้ตัวชี้ที่กำหนดไว้สำหรับชี้หน่วยสุดท้ายมาชี้ที่หน่วยนี้ด้วย (หน่วยนี้จะกลายเป็นหน่วยข้อมูลสุดท้ายตัวใหม่แทนตัวเก่า ที่กำลังจะไม่ใช่หน่วยสุดท้ายอีกต่อไป) ในกรณีที่เป็น Circular Linked List ก็จะกำหนดให้ตัวชี้ในหน่วยใหม่นี้ ชี้กลับไปที่หน่วยข้อมูลหน่วยแรก

4.ขั้นตอนสุดท้ายเป็นการเชื่อมหน่วยข้อมูลที่สร้างใหม่กับหน่วยสุดท้าย (เดิม) ในกรณีของ Singly Linked List เราจะถ่ายค่าใน newnode ให้แก่ตัวชี้ในหน่วยข้อมูลที่ walknode ชี้อยู่อันเป็นหน่วยข้อมูลสุดท้าย (เดิม) ให้ชี้ไปที่หน่วยสุดท้าย (ใหม่) ในกรณีของ Doubly Linked List เราจะถ่ายค่า newnode ให้แก่ตัวชี้ที่ใช้สำหรับชี้หน่วยถัดไป ในหน่วยที่ walknode ชี้อยู่และถ่ายค่า walknode ให้แก่หน่วยชี้ที่ใช้สำหรับชี้หน่วยก่อน ในหน่วยที่ newnode ชี้อยู่ อันเป็นการเชื่อมหน่วยทั้งสองเข้าด้วยกันและหน่วยที่ newnode ชี้อยู่นี้จะกลายเป็นหน่วยข้อมูลสุดท้ายตัวใหม่

- การเข้าหาหน่วยข้อมูล

ในการเข้าถึงสมาชิกใดๆ ในหน่วยข้อมูล เราจะใช้ตัวชี้ที่กำหนดขึ้นสำหรับใช้ในการเข้าถึงโดยเฉพาะ จะไม่ใช้ตัวชี้ที่กำหนดขึ้นสำหรับชี้จุดเริ่มต้นของลิสต์ เพราะถ้าเราใช้ตัวชี้ตัวนี้สำหรับเข้าหาข้อมูลในหน่วยต่างๆ ของลิสต์ เราจะสูญเสียค่าตำแหน่งเริ่มต้นของลิสต์และจะไม่สามารถเข้าถึงจุดเริ่มต้นของลิสต์ รวมทั้งคืนหน่วยความจำทั้งหมดของลิสต์ได้อีกต่อไป

1.เริ่มจากกำหนดให้ walknode ซึ่งเป็นตัวชี้สำหรับใช้เข้าถึงหน่วยต่างๆ ในลิสต์ ให้ชี้ไปยังจุดเริ่มต้นของลิสต์เสียก่อน

2.เลื่อนตัวชี้ walknode ไปยังหน่วยข้อมูลถัดไป โดยการถ่ายค่าตัวชี้สำหรับชี้หน่วยถัดไปในหน่วยข้อมูลที่ walknode ชี้อยู่ให้แก่ตัว walknode เอง ในกรณีของ Doubly Linked List การเลื่อนตัวชี้สามารถกระทำได้ 2 ทางคือ การเลื่อนไปยังหน่วยถัดไป และการเลื่อนไปยังหน่วยก่อนการเลื่อนไปยังหน่วยถัดไปนั้น walknode จะได้ค่าจากตัวชี้หน่วยถัดไปในหน่วยที่ walknode ชี้อยู่ ส่วนการเลื่อนไปยังหน่วยก่อน จะได้ค่าจากตัวชี้หน่วยก่อน จากตัวชี้ในหน่วยที่ walknode ชี้อยู่นั่นเอง

3.เมื่อพบหน่วยข้อมูลที่ต้องการ หรือต้องการอ่านข้อมูลจากหน่วยดังกล่าว เราก็จะใช้ตัวชี้ walknode ซึ่งกำลังชี้อยู่ที่หน่วยข้อมูลที่ต้องการนี้ ติดต่อกับสมาชิกใดๆ ในหน่วยข้อมูล

- การแทรกข้อมูล

การแทรกข้อมูลจะประกอบไปด้วยขั้นตอนการกำหนดหน่วยข้อมูลขึ้นใหม่ การใส่ข้อมูลลงในหน่วยข้อมูล และการตัดสายข้อมูลออกจากกัน เพื่อแทรกหน่วยข้อมูลใหม่ลงในสายข้อมูลเก่า

1.ในขั้นแรก เราจะสร้างหน่วยข้อมูลใหม่ และเลื่อน walknode ไปยังจุดที่ต้องการแทรกข้อมูล ใส่ข้อมูลลงในหน่วยข้อมูลใหม่ให้เรียบร้อย

2.ขั้นตอนการแทรกหน่วยข้อมูลใหม่ที่เพิ่งสร้างขึ้นให้เชื่อมเข้ากับลิสต์เก่า

ในกรณีของ Singlyd Linked List เราจะถ่ายค่าการชี้ในตัวชี้หน่วยถัดไปของหน่วยข้อมูลที่ walknode ชี้อยู่ ให้กับตัวชี้หน่วยถัดไปที่ newnode ชี้อยู่ เป็นการโอนการชี้หน่วยข้อมูลถัดไปของหน่วยข้อมูลที่ walknode ชี้อยู่ให้กับหน่วยข้อมูลที่ newnode ชี้อยู่นั้นทำการชี้แทน แล้วจึงถ่ายค่าการชี้ของ newnode ให้กับตัวชี้หน่วยถัดไปของหน่วยข้อมูลที่ walknode ชี้อยู่ เป็นการกำหนดให้หน่วยข้อมูลที่ walknode ชี้อยู่นี้มีหน่วยข้อมูลถัดไปเป็นหน่วยข้อมูลที่ newnode ชี้อยู่นั่นเอง

ในกรณีของ Doubly Linked List ค่อนข้างจะยุ่งยาก เริ่มจากการโอนค่าการชี้หน่วยถัดไปของหน่วยข้อมูลที่ walknode ชี้อยู่ให้แก่ตัวชี้หน่วยถัดไปของหน่วยข้อมูลที่ newnode ชี้อยู่และถ่ายค่าการชี้หน่วยก่อนของหน่วยข้อมูลในลำดับถัดจากหน่วยที่ walknode ชี้อยู่ให้แก่ตัวชี้หน่วยก่อนของหน่วยข้อมูลที่ newnode ชี้อยู่ ซึ่งก็คือค่าเดียวกันกับค่าการชี้ใน walknode นั่นเอง เป็นการเชื่อมหน่วยที่สร้างขึ้นใหม่ เข้ากับหน่วยข้อมูลเก่าทั้งสอง ในจุดที่หน่วยใหม่จะไปแทรก หลังจากนั้นก็จะเชื่อมหน่วยเก่าทั้งสองเข้ากับหน่วยใหม่ โดยการถ่ายค่าการชี้ของ newnode อันเป็นค่าตำแหน่งหน่วยความจำเริ่มต้นของหน่วยข้อมูลใหม่ให้กับ

ตัวชี้ "หน่วยถัดไป" ของหน่วยข้อมูลที่ walknode ชี้อยู่และถ่ายให้กับตัวชี้ "หน่วยก่อน" ของหน่วยข้อมูล "ถัดไป" จากหน่วยข้อมูลที่ newnode เพิ่งได้รับค่าการชี้มาใหม่

- การตัดข้อมูล

1.เริ่มจากการค้นหาตำแหน่งของหน่วยข้อมูลที่ต้องการตัดออก แล้วเลื่อน walknode ให้ไปชี้ที่หน่วยข้อมูลก่อนหน่วยที่จะตัดออก และเลื่อน nextnode ไปชี้ที่หน่วยข้อมูลหลังหน่วยที่จะตัดออก

2.คืนหน่วยความจำของหน่วยข้อมูลที่ต้องการจะตัดออกให้แก่ระบบ

3.หลังจากนั้นจึงเชื่อมหน่วยที่เหลือ เข้า ด้ว ยกัน

ในกรณีของ Singly Linked List จะถ่ายค่าการชี้ใน nextnode ให้แก่ตัวชี้หน่วยถัดไป ในหน่วยข้อมูลที่ walknode ชี้อยู่ ส่วนในกรณีของ Doubly Linked List จะให้ค่าการชี้ใน nextnode ให้แก่ตัวชี้หน่วยถัดไปในหน่วยข้อมูลที่ walknode ชี้อยู่ และถ่ายค่าการชี้ใน walknode ให้แก่ตัวชี้หน่วยก่อนในหน่วยข้อมูลที่ nextnode ชี้อยู่

- การย้ายข้อมูล

การย้ายข้อมูลจะประกอบไปด้วยขั้นตอน 2 ขั้นตอนคือ การตัดข้อมูล ที่ต้องการย้ายออกจากลิสต์ โดยใช้กรรมวิธีเดียวกันกับการตัดข้อมูล แต่เราจะใช้ตัวชี้อีกตัวหนึ่ง ในการจดจำตำแหน่งของหน่วยข้อมูลที่ต้องการย้ายไว้ และนำไปแทรกในตำแหน่งที่ต้องการ ตามกรรมวิธีการแทรกข้อมูล

- การลบลิสต์ออกทั้งหมด

จะกระทำได้ 2 วิธีคือ คืนจากหน่วยข้อมูลสุดท้ายกลับมายังหน่วยข้อมูลแรก หรือคืนจากหน่วยข้อมูลแรกไปยังหน่วยข้อมูลสุดท้าย ในที่นี้จะเป็นกรรมวิธีการคืนจากหน่วยแรกไปยังหน่วยสุดท้าย คือ

1.กำหนดให้ตัวชี้ walknode ชี้ยังจุดเริ่มต้น และตัวชี้ nextnode ชี้ยังหน่วยถัดไป 1 หน่วย

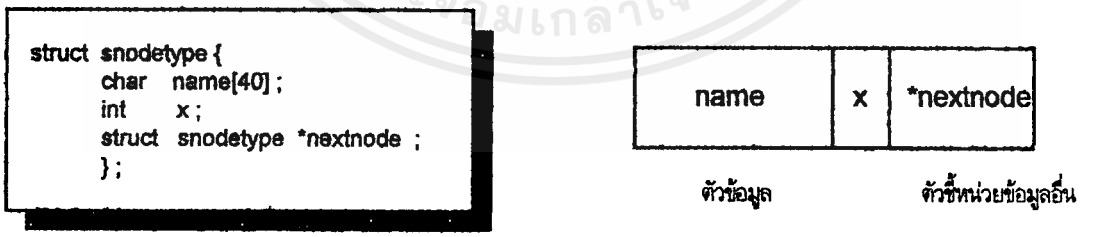
2.คืนหน่วยความจำที่ walknode ชี้อยู่ให้แก่ระบบ และถ่ายค่าการชี้ของ nextnode ให้แก่ walknode แล้วเลื่อน nextnode ให้ไปชี้ยังหน่วยถัดไปโดยการถ่ายค่าการชี้ในตัวชี้หน่วยถัดไป หน่วยข้อมูลที่ nextnode ชี้อยู่นั้น จะชี้ให้แก่ตัว nextnode เอง กระทำเช่นนี้ไปเรื่อยๆ จนกว่าจะคืนหน่วยความจำของหน่วยข้อมูลทั้งหมดที่ใช้ในลิสต์

2.หน่วยข้อมูลไดนามิก (Node) คือ หน่วยข้อมูลที่ได้มาจากการจัดหน่วยความจำแบบไดนามิก ในหนึ่งหน่วยจะประกอบไปด้วยหนึ่งข้อมูลซึ่งอาจจะมีมากกว่าหนึ่งข้อมูลก็ได้ และตัวชี้ไปยังหน่วยข้อมูลอื่นจะทำหน้าที่บอกตำแหน่งหน่วยความจำเริ่มต้นของหน่วยข้อมูลถัดไป ซึ่งก็อาจจะมีมากกว่าหนึ่งตัวก็ได้ ขึ้นอยู่กับโครงสร้างข้อมูล หน่วยข้อมูลแต่ละหน่วยจะนิยามจากโครงสร้างข้อมูลสตรัคเจอร์ ซึ่งสามารถกำหนดสมาชิกให้มีชนิดต่างกันได้ ดังรูป

รูปที่ 2.3.3 โครงสร้างหน่วยข้อมูลแบบไดนามิกที่ง่ายที่สุด

จากรูปที่ 2.3.3 หน่วยข้อมูลไดนามิก (ซึ่งต่อไปเราจะเรียกโดยย่อว่า "หน่วยข้อมูล") จะประกอบไปด้วยสองส่วนคือ **ตัวข้อมูล** เป็นตัวแปร x มีชนิดเป็น int และ **ตัวชี้ไปยังหน่วยข้อมูลอื่น** เป็นตัวชี้ชื่อ **nextnode** มีชนิดเป็น **struct snodetype** ซึ่งหมายถึง **nextnode** จะใช้สำหรับชี้ข้อมูลที่มีชนิดเดียวกันกับตัวโครงสร้างของตัวเอง หรือกล่าวง่ายๆ ก็คือ หน่วยข้อมูลหนึ่งๆ จะมีตัวชี้สำหรับชี้ไปยังหน่วยข้อมูลอื่นๆ ที่มีโครงสร้างเดียวกัน

ตัวข้อมูลอาจจะประกอบไปด้วยข้อมูลมากกว่าหนึ่งตัว และอาจมีชนิดต่างกันได้ ดังเช่น

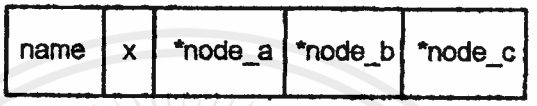


รูปที่ 2.3.4 โครงสร้างหน่วยข้อมูลแบบไดนามิกที่มีข้อมูลมากกว่าหนึ่งตัว

ประกอบไปด้วยตัวข้อมูลสองตัวคือ **name** เป็นอาร์เรย์ชนิด **char** ขนาด 40 หน่วย และ **x** เป็นข้อมูลเดี่ยวชนิด **int** และมี **nextnode** เป็นตัวชี้หน่วยข้อมูลถัดไป

ตัวชี้ข้อมูลอื่น อาจมีมากกว่าหนึ่งตัวก็ได้ ดังเช่น

```
struct snodetype {
        char   name[40] ;
        int    x ;
        struct snodetype *node_a ;
        struct snodetype *node_b ;
        struct snodetype *node_c ;
        } ;
```



ตัวข้อมูล ตัวชี้หน่วยข้อมูลอื่น

รูปที่ 2.3.5 โครงสร้างหน่วยข้อมูลแบบไดนามิกที่มีข้อมูลมากกว่าหนึ่งตัวและมีตัวชี้ข้อมูลอื่นอีกสามตัว

ประกอบไปด้วยข้อมูลสองตัวคือ **name** และ **x** และมีตัวชี้ข้อมูลสามตัวคือ **node_a node_b** และ **node_c** ทั้งสามตัวมีชนิดเดียวกัน สำหรับใช้ชี้ไปยังหน่วยข้อมูลอื่นๆ ในทิศทางที่ต่างกัน เช่นหน่วยข้อมูลก่อนหน้านี้ หรือหน่วยข้อมูลถัดไป เป็นต้น

3.สแต็ก คิว และทรี

สแต็ก (Stacks) คือโครงสร้างข้อมูลที่ประกอบไปด้วยข้อมูลหลายหน่วยการเข้าถึงหน่วยข้อมูลจะใช้ตัวชี้สแต็ก ข้อมูลใดที่ถูกเก็บเข้าไปในสแต็กก่อน จะถูกอ่านออกมาได้เป็นลำดับหลัง

ตัวชี้สแต็ก (Stack Pointers) คือตัวชี้ที่ทำหน้าที่บอกตำแหน่งหน่วยความจำปัจจุบันของข้อมูลที่จะใช้อ่านหรือเขียน

คิว (Queues) คือโครงสร้างข้อมูลที่ประกอบไปด้วยข้อมูลหลายหน่วย ข้อมูลที่ถูกนำเข้าไปเก็บจะเรียงตัวกันเป็นลำดับ ข้อมูลใดที่ใส่เข้าไปเป็นลำดับก่อนจะถูกอ่านออกมาได้ก่อน

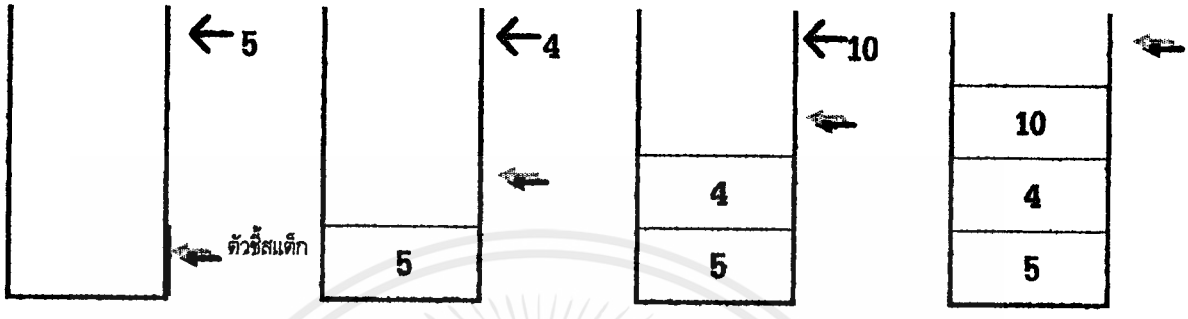
ตัวชี้สำหรับอ่าน (Read Pointers) คือตัวชี้ที่ทำหน้าที่บอกตำแหน่งหน่วยความจำปัจจุบันของหน่วยข้อมูลในคิวที่จะถูกอ่าน

ตัวชี้สำหรับเขียน (Write Pointers) คือตัวชี้ที่ทำหน้าที่บอกตำแหน่งหน่วยความจำปัจจุบันของหน่วยข้อมูลในคิวที่จะใช้เก็บข้อมูล

ทรี (Trees) คือโครงสร้างข้อมูลที่ประกอบไปด้วยหน่วยข้อมูลย่อยหลายๆหน่วยเชื่อมต่อกันในลักษณะแตกเป็นกิ่งก้านออกไปคล้ายต้นไม้ การเข้าถึงข้อมูลใดๆ ภายในโครงสร้างทรี จะต้องเข้าจากทางด้านราก(Root) หรือหน่วยข้อมูลแรกของทรี

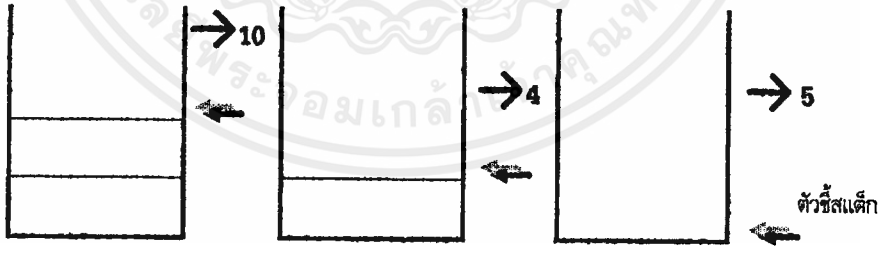
- สแต็ก

สแต็ก นับเป็นโครงสร้างข้อมูลหนึ่งที่สามารถจัดการได้ทั้งวิธีแบบสแตติก และแบบไดนามิก แต่คุณสมบัติที่สำคัญของสแต็กที่เหมือนกันก็คือ การเข้าถึงข้อมูลภายในหน่วยจะอาศัย **ตัวชี้สแต็ก** การอ่านหรือเขียนแต่ละครั้ง จะกระทำกับตำแหน่งบนสุดของสแต็กเสมอ ดังรูป



รูปที่ 2.3.6 การเขียนข้อมูลลงสแต็ก

ในการเขียนข้อมูลลงสแต็กนั้น จะมีขั้นตอนอย่างคร่าวๆ สองขั้นคือ การเลื่อนตัวชี้สแต็ก และการเขียนข้อมูลลงในตำแหน่งที่ตัวชี้สแต็กชี้อยู่ จากรูป สมมติว่าในสแต็กว่างเปล่า ไม่มีข้อมูลใดๆ อยู่ เราใส่ค่า 5 ลงในสแต็กที่ว่างเปล่า ค่า 5 จะวางตัวอยู่ล่างสุดของสแต็ก ตัวชี้สแต็กจะเลื่อนไปหนึ่งอันดับ เมื่อเราใส่ค่า 4 ลงไป ค่า 4 ก็จะไปอยู่ในตำแหน่งถัดไปจากค่า 5 และในทำนองเดียวกันกับค่า 10 ที่หลังจากใส่แล้วจะไปอยู่ข้างบนสุด



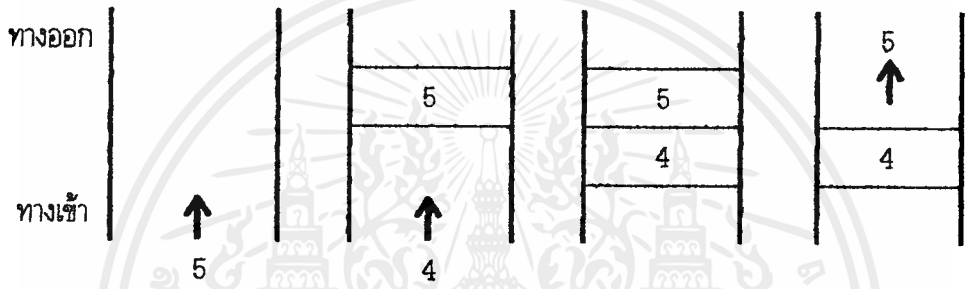
รูปที่ 2.3.7 การอ่านข้อมูลจากสแต็ก

เมื่อเราอ่านข้อมูล ก็จะเกิดการกระทำในทิศทางตรงกันข้ามค่าที่อ่านได้คือ ค่าที่จะตรงกับตำแหน่งที่ตัวชี้สแต็กชี้อยู่ ดังนั้น ในครั้งแรกเราจะอ่านได้ค่า 10 หลังจากนั้น ตัวชี้จะเลื่อนลงมาหนึ่งอันดับ เมื่อเราอ่านต่อไปจึงได้ค่า 4 และค่า 5 เป็นค่าสุดท้าย

ในกรณีที่เราใส่ข้อมูลจนพื้นที่ที่จองไว้สำหรับเก็บข้อมูลมีไม่เพียงพอ (แบบสแตติก) ข้อมูลที่จะใส่เพิ่มลงไปก็จะไม่สามารถเพิ่มลงได้อีก ลักษณะเช่นนี้เราเรียกว่า Stack Overflow ในทางกลับกัน เมื่อเราพยายามอ่านข้อมูลจากสแต็กที่ว่างเปล่า เราก็จะไม่ได้ข้อมูลใดๆ ลักษณะเช่นนี้เราเรียกว่า Stack Uynderflow

- คิว

คิวเป็นโครงสร้างข้อมูลอีกประเภทหนึ่ง ที่สามารถจำลองได้ทั้งแบบสแตติก คือจองพื้นที่ส่วนหนึ่งไว้เลย กับแบบไดนามิก เป็นการจองพื้นที่เก็บข้อมูลเมื่อมีข้อมูลใหม่เข้ามา และก็จะคืนพื้นที่ของหน่วยข้อมูลเมื่อได้อ่านหน่วยนั้นๆ ไปแล้ว การทำงานของคิวอาจจะอธิบายได้ด้วยรูปดังนี้



รูปที่ 2.3.8 การอ่านและเขียนข้อมูลในคิว

จากรูปที่ 2.3.8 คิวยังว่างอยู่ สมมติว่ามีการใส่ค่า 5 ลงไปและตามด้วยค่า 4 เมื่ออ่านข้อมูลจากคิวก็จะได้ค่า 5 ก่อน และตามด้วย 4

คิวอีกประเภทหนึ่งที่ใช้กันโดยทั่วไป ซึ่งจะมีการจัดการการเขียน และอ่านด้วยวิธีที่ใช้เวลาในการจัดการน้อยที่สุดในคิวทั่วไปก็คือ **คิววงกลม** เราอาจนิยามคิววงกลมได้ดังนี้

คิววงกลม (Circular Queues) คือโครงสร้างข้อมูลแบบคิวแบบสแตติก ที่มีหน่วยข้อมูลแรกอยู่ต่อจากหน่วยสุดท้าย การเชื่อมต่อกันของหน่วยข้อมูลนี้ พิจารณาจากตัวชี้ข้อมูลสำหรับอ่านและเขียน ซึ่งจะมีการปรับตำแหน่งค่าการชี้ให้กลับมาอยู่ในตำแหน่งข้อมูลแรก เมื่อเลื่อนตำแหน่งการชี้ถัดไปจากตำแหน่งข้อมูลสุดท้ายหนึ่งตำแหน่ง

คิววงกลมจะมีการจองพื้นที่จำนวนหนึ่งสำหรับการเก็บค่า เมื่อมีการเขียนข้อมูล ข้อมูลก็จะเข้าไปโดยอาศัยตัวชี้สำหรับเขียนของคิว หลังจากนั้น ตัวชี้ก็จะเลื่อนไปยังตำแหน่งถัดไป หากตำแหน่งถัดไปอยู่นอกขอบเขตของพื้นที่ที่จองไว้ ก็จะมีการปรับค่าการชี้ให้มาชี้อยู่ที่ตำแหน่งเริ่มต้นของพื้นที่ ในการอ่านข้อมูล ก็จะอ่านผ่านทาง**ตัวชี้สำหรับอ่าน**ของคิว เมื่ออ่านข้อมูลได้แล้ว ก็จะเลื่อนตัวชี้ไปยังตำแหน่งถัดไป เช่นเดียวกันกับตัวชี้สำหรับการเขียน เมื่อค่าการชี้ออกไปนอกขอบเขตพื้นที่ ก็จะมีการปรับตัวชี้สำหรับอ่านให้มาอยู่ในตำแหน่งเริ่มต้นของพื้นที่ โดยการกระทำเช่นนี้ เราจึงสามารถนำสายของข้อมูล ซึ่งก็คือ**สตรีม**ไปพักไว้ในคิววง

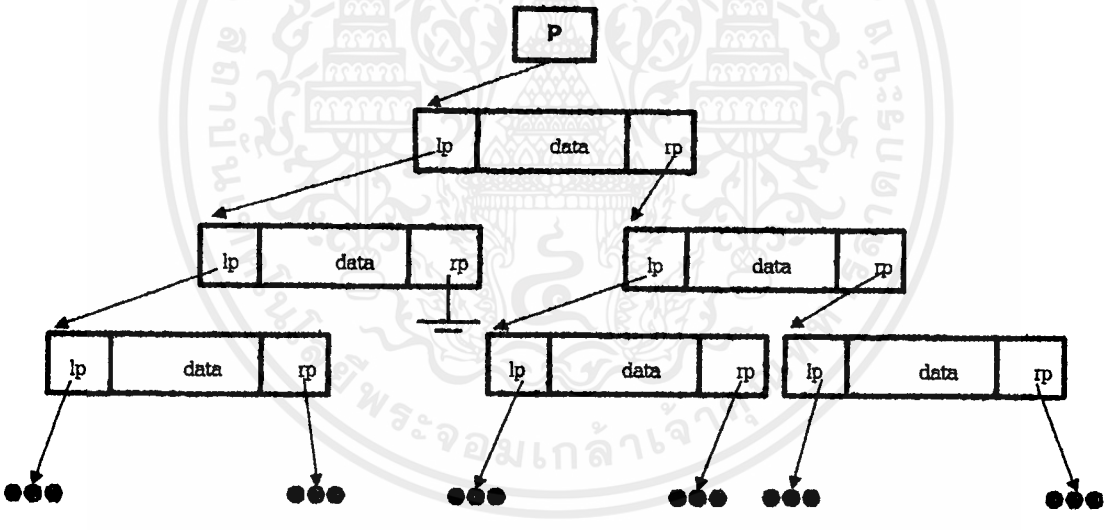
กลมได้ และเมื่อต้องการข้อมูลสตรีมดังกล่าว ก็อ่านออกมาจากคิววงกลม โดยคุณสมบัติของคิว ลำดับของข้อมูลที่ใส่เข้าไป กับที่อ่านออกมาในภายหลังจะยังคงเดิม อธิบายหลักการทำงานของคิววงกลมได้คือ

เมื่อมีการเขียนข้อมูลลงในคิว ข้อมูลจะเข้าไปโดยอาศัยตำแหน่งจากตัวชี้สำหรับเขียน หลังจากนั้นตัวชี้สำหรับเขียนก็จะเลื่อนไปยังตำแหน่งหน่วยถัดไป เมื่อมีการอ่านข้อมูล ข้อมูลที่ถูกตัวชี้สำหรับอ่านก็จะถูกนำออกมา แล้วตัวชี้สำหรับอ่านก็จะเลื่อนไปยังหน่วยถัดไป ทิศทางการเลื่อนของตัวชี้ทั้งสองจะมีทิศทางไปในทางเดียวกัน เมื่อตัวชี้ทั้งสองอยู่ในตำแหน่งเดียวกัน นั่นหมายถึงไม่มีข้อมูลอยู่ในคิว แต่ถ้าตัวชี้สำหรับเขียนอยู่ในตำแหน่งก่อนตัวชี้สำหรับอ่านหนึ่งตำแหน่ง ก็จะไม่สามารถเขียนข้อมูลลงในคิวได้อีก เพราะหากเขียนข้อมูลเพิ่มลงไป ก็จะไปทับข้อมูลเดิมที่ยังไม่ได้อ่านออกไป

- ทรี

ทรี ประกอบไปด้วยหน่วยข้อมูลไดนามิกหลายตัวที่เชื่อมต่อกันคล้ายกิ่งก้านของต้นไม้ การเข้าถึงหน่วยข้อมูลแต่ละหน่วยจะกระทำได้โดยผ่านทางหน่วยข้อมูลที่อยู่ในลำดับก่อนๆ ดังรูปของไบนารีทรี

ไบนารีทรี (Binary Trees) คือ ทรีที่มีกิ่งก้านแยกออกไปไม่เกินสองกิ่งจากหน่วยข้อมูลหนึ่งๆ



รูปที่ 2.3.9 โครงสร้างของไบนารีทรี

โครงสร้างของทรี และรวมไปถึงโครงสร้างหน่วยข้อมูลไดนามิกทุกตัว จะต้องมีจุดเริ่มต้นจาก**ตัวชี้ข้อมูลชนิดหน่วยข้อมูล**เสมอ จากตัวชี้จุดเริ่มต้น จะเก็บค่าตำแหน่งหน่วยความจำเริ่มต้นของหน่วยข้อมูลแรก หรืออาจเรียกได้ว่า **ชี้ไปยังหน่วยข้อมูลแรก** ในหน่วยข้อมูลแรกของทรี จะมีตัวชี้ตั้งแต่สองตัวขึ้นไป แต่ละตัวจะใช้ชี้หน่วยข้อมูลถัดไป ในกรณีของไบนารีทรีจะมีตัวชี้หน่วยข้อมูลถัดไปเพียงสองตัว จากรูปเราแทนด้วย lp และ rp หน่วยข้อมูลถัดไปก็อาจจะชี้ไปยังหน่วยถัดๆ ไปอีกคล้ายกิ่งก้าน การเข้าถึงหน่วยแต่ละหน่วยจะอาศัยการเริ่มจากตัวชี้โครงสร้างทั้งหมด และจากหน่วยหนึ่งๆจะเข้าไปถึงหน่วยต่อๆไปได้ โดยอาศัยตัวชี้

หน่วยถัดไปในหน่วยข้อมูลนั้นๆ ในการบอกตำแหน่ง**หน่วยข้อมูลสุดท้าย** เราจะนิยมกำหนดให้ตัวชี้หน่วยข้อมูลถัดไปในหน่วยสุดท้ายให้ชี้ไปยังหน่วยของตัวเอง หรือชี้ไปที่ **NULL** (ในรูปเราจะแทนด้วย ⊥) รายละเอียดในการจัดการเพิ่มเติม แทรก หรือ ลบหน่วยข้อมูลจะเป็นไปในลักษณะเดียวกันกับลิงค์ลิสต์

2.3.2 การจัดหน่วยความจำแบบไดนามิก

ในการจัดการโครงสร้างข้อมูลแบบไดนามิกนั้น พื้นที่ที่จะใช้เก็บข้อมูลจะยังไม่ถูกจองเมื่อโปรแกรมเริ่มทำงานเหมือนกับโครงสร้างข้อมูลสแตติก ดังนั้นจึงเป็นหน้าที่ของผู้เขียนโปรแกรม ที่จะต้องใส่ฟังก์ชั่นสำหรับจองหน่วยความจำก่อนที่จะเขียนโปรแกรมติดต่อกับโครงสร้างข้อมูลไดนามิก และเมื่อใช้งานเสร็จสิ้นก็จะเป็นหน้าที่ของผู้เขียนโปรแกรม ที่จะต้องคืนหน่วยความจำเหล่านั้นให้ระบบด้วย ลำดับการจัดหน่วยความจำแบบไดนามิก จึงอาจเขียนเป็นขั้นตอนได้ดังนี้

1.จองหน่วยความจำ เราจะกำหนดจำนวนพื้นที่ที่ต้องการสำหรับข้อมูลหนึ่งหน่วย แล้วจองพื้นที่จากระบบตามจำนวนที่ต้องการ ค่าที่ได้จากระบบก็คือค่าตำแหน่งหน่วยความจำเริ่มต้นสำหรับการใช้งาน ค่าดังกล่าวเราจะเก็บไว้ในตัวชี้ข้อมูลชนิดที่สอดคล้องกับโครงสร้างข้อมูลไดนามิกที่จองนั้นสำหรับใช้อ้างตำแหน่งเริ่มต้นเพื่อใช้งานต่อไป

2.ใช้หน่วยความจำ เราจะใช้ติดต่อกับพื้นที่ที่ต้องการโดยอาศัยตัวชี้ที่ชี้ไปยังจุดเริ่มต้นหน่วยข้อมูลนี้ นอกจากนั้น เราอาจกำหนดตัวชี้ขึ้นใหม่ เพื่อใช้ในการทำงานอีกก็ได้

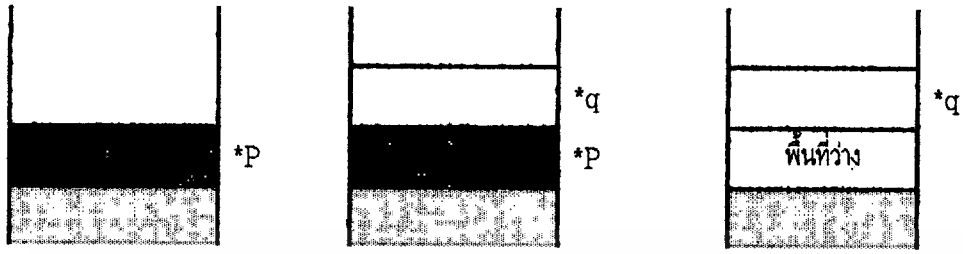
3.คืนหน่วยความจำ หลังจากใช้งานเสร็จสิ้นแล้ว เราจะคืนหน่วยความจำที่ได้จองไว้แก่ระบบ โดยอาศัยค่าการชี้ที่ได้จากการจองหน่วยความจำนั่นเอง

ในภาษาซี มีฟังก์ชั่นที่ใช้งานเกี่ยวกับการจัดการหน่วยความจำแบบไดนามิกที่น่าสนใจคือ

ฟังก์ชั่น	โมดูล	ความหมาย
void *malloc(size_t size); void far *farmalloc(unsiged long nbytes);	alloc.h, stdlib.h alloc.h	จองหน่วยความจำตามจำนวนไบต์ที่ต้องการ
void free (void *block); void farfree (void far *block);	alloc.h, stdlib.h alloc.h	คืนหน่วยความจำให้ระบบ
unsigned coreleeft (void);	alloc.h	รายงานส่วนฮีพที่เหลือ
unsigned long coreleft (void); unsigned long farcoreleft (void);	alloc.h alloc.h	รายงานหน่วยความจำระบบที่เหลือ
void *realloc (void *oldblock,size_t size); void far *farrealloc (void far *oldblock, unsigned long nbytes);	alloc.h, stdlib.h alloc.h	เปลี่ยนขนาดพื้นที่ของหน่วยความจำ

ตาราง 2.2.1 ฟังก์ชันที่ใช้งานเกี่ยวกับการจัดหน่วยความจำแบบไดนามิก

การจองพื้นที่หน่วยความจำ จะได้พื้นที่มาจากส่วนฮีพ (Heap) ซึ่งเป็นพื้นที่ที่เหลือจากการใช้งานของโปรแกรม ในการทำงานจริง พื้นที่ส่วนฮีพนี้อาจไม่ได้อยู่ติดกันเป็นส่วนเดียว แต่อาจจะกระจายแทรกอยู่ระหว่างพื้นที่ที่ใช้งานก็ได้ ซึ่งเกิดจากการคืนหน่วยความจำบางส่วนให้กับระบบไปก่อน ดังรูป



a) จองพื้นที่ให้ตัวชี้ p b) จองพื้นที่ให้ตัวชี้ q c) คืนพื้นที่ที่ p ชี้อยู่

รูปที่ 2.2.1 การทำงานที่ก่อให้เกิดพื้นที่ในฮีพที่ไม่ต่อเนื่อง

จากรูป 2.2.1a มีการจองพื้นที่จำนวนหนึ่งให้ตัวชี้ p หลังจากนั้น ในรูป 2.2.1b มีการจองพื้นที่อีกจำนวนหนึ่งให้ q การจัดพื้นที่ในหน่วยความจำของระบบปฏิบัติการดอสนั้น จะจัดสรรพื้นที่จากพื้นที่ว่างหรือฮีพให้แก่โปรแกรมตามลำดับจากตำแหน่งหน่วยความจำค่าต่ำกว่าไปยังค่าสูงกว่า ดังนั้นพื้นที่ที่ตัวชี้ p ได้รับจึงอยู่ก่อนพื้นที่ที่ q ได้รับ เมื่อโปรแกรมคืนพื้นที่ที่ p ได้รับในรูปที่ 2.2.1c พื้นที่ที่ p เคยได้รับนี้ก็กลายเป็นพื้นที่ว่างสำหรับให้โปรแกรมจองไปใช้งานได้อีก นับเป็นส่วนฮีพเช่นกัน จากรูป 2.2.1c จึงเห็นพื้นที่ของฮีพแยกออกเป็นสองส่วนไม่ต่อเนื่อง

ในกรณีเช่นนี้ หากการขอใช้พื้นที่ครั้งต่อไปมีขนาดไม่เกินพื้นที่ว่างที่ได้รับคืนจาก p ระบบก็จะตัดพื้นที่ส่วนนี้ให้แก่โปรแกรม แต่ถ้าการขอใช้พื้นที่มีขนาดพื้นที่เกินกว่านี้ ระบบก็จะตัดส่วนที่อยู่ถัดจากพื้นที่ที่ q ครอบครองอยู่ไปแทน แต่ถ้ายังไม่สามารถจัดการได้ เนื่องจากไม่มีพื้นที่ว่างต่อเนื่องที่มีขนาดใหญ่พอจะให้ได้ ฟังก์ชันที่ใช้ขอพื้นที่จากระบบก็จะรายงานเป็น NULL แม้พื้นที่รวมจะมีขนาดใหญ่พอก็ตาม

ข้อควรระวังในการจัดหน่วยความจำแบบไดนามิก

การจัดหน่วยความจำแบบไดนามิก อาศัยตัวชี้ในการเข้าถึงพื้นที่เป็นหลักในการจัดการใดๆในตัวชี้ ภาษาซีจะถือว่าค่าในตัวชี้จะเป็นค่าที่เหมาะสมในตัวชี้เสมอ ดังนั้น หากผู้ใช้กระทำใดๆ ต่อตัวชี้ที่ยังมีค่าไม่เหมาะสม จะทำให้เกิดผลเสียหายต่อระบบได้ การจัดการหน่วยความจำแบบไดนามิกจึงมีข้อควรระวังที่สำคัญดังนี้

1.อย่าเขียนข้อมูลลงในตัวชี้ที่ยังไม่ได้กำหนดค่าให้อย่างถูกต้อง การเขียนข้อมูลผ่านตัวชี้ดังกล่าวจะเป็นการเขียนลงในตำแหน่งหน่วยความจำที่เราไม่ทราบ ซึ่งอาจจะเป็นส่วนโปรแกรม ข้อมูล หรือส่วนของระบบปฏิบัติการ และจะทำให้ระบบหยุดทำงาน หรือทำงานผิดพลาดได้

2.อย่าเขียนข้อมูลมากกว่าพื้นที่ที่จองไว้ เพราะข้อมูลที่เกินไปนั้นจะไปทับข้อมูลส่วนอื่น หรือไปทับโปรแกรมอื่นจะทำให้ระบบทำงานผิดพลาดหรือหยุดการทำงานได้

3.ระวังการใช้ตัวชี้ผิดชนิด การใช้ตัวชี้ผิดชนิดจะทำให้การกระทำเลขคณิตของค่าการชี้ผิดพลาด ตำแหน่งหน่วยความจำที่เข้าถึงจะผิดไป ทำให้เขียนหรืออ่านข้อมูลผิดพลาด และหากเกิดการเขียนข้อมูลผิดพลาด และหากเกิดการเขียนข้อมูลนอกพื้นที่ที่ได้จองไว้ ก็จะทำให้ระบบทำงานผิดพลาดได้

4.อย่าหยุดโปรแกรมก่อนที่จะคืนหน่วยความจำให้ระบบ เพราะหน่วยความจำเหล่านั้นระบบปฏิบัติการจะไม่สามารถจัดสรรให้กับโปรแกรมอื่นๆ ได้อีก จนกว่าจะบูตระบบใหม่

5.อย่าใช้ฟังก์ชันคืนหน่วยความจำกับตัวชี้ที่ไม่ได้ชี้ไปยังหน่วยความจำที่ได้จองไว้ด้วยฟังก์ชันจองหน่วยความจำแบบไดนามิก ซึ่งถ้าหากตัวชี้นั้นถูกชี้ไปยังหน่วยความจำที่ได้จองไว้ด้วยกรรมวิธีอื่น จะทำให้พื้นที่นั้นถูกคืนให้แก่ระบบทั้งๆที่ยังคงมีความต้องการใช้งาน เมื่อมีการขอใช้พื้นที่ พื้นที่เหล่านั้นจะถูกนำไปใช้งานอื่นทำให้เกิดความเสียหายได้

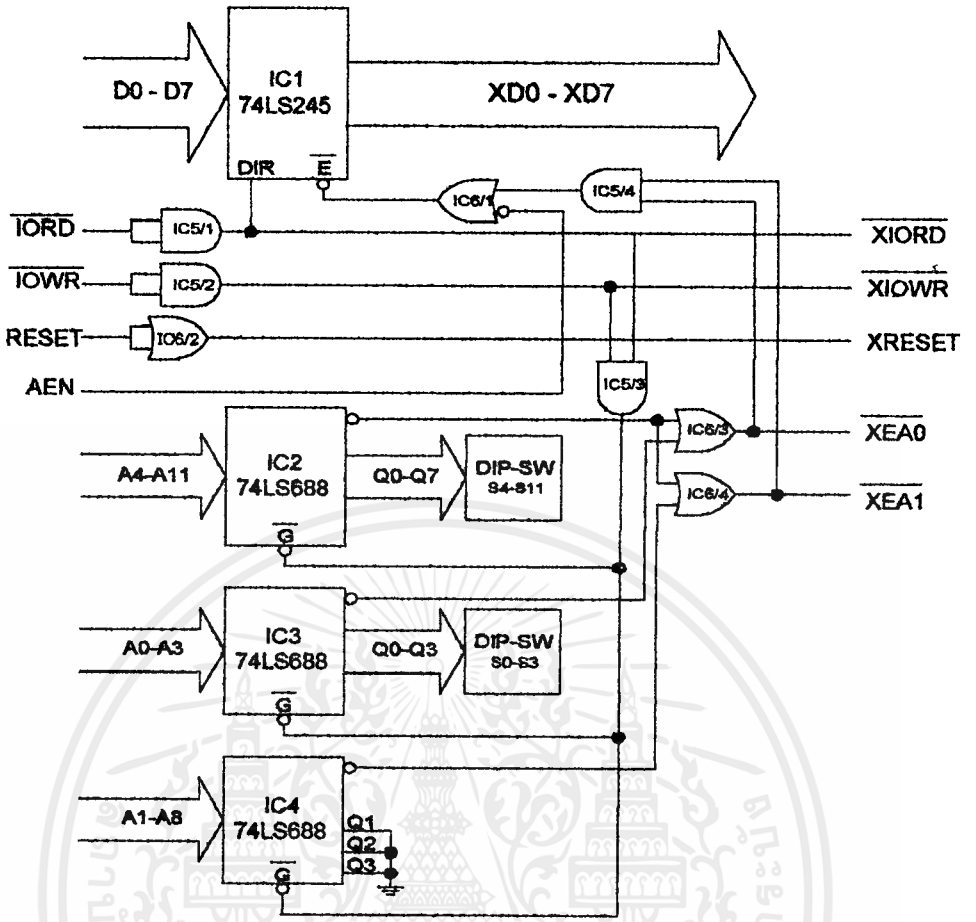
บทที่ 3
หลักการทำงานของเครื่องวิเคราะห์วงจรดิจิตอล

สำหรับตัวฮาร์ดแวร์นั้น ทางคณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์ ไพศาล สิทธิโยภาสกุล ที่ได้มอบส่วนฮาร์ดแวร์ เพื่อนำมาแก้ไขและปรับปรุงให้ใช้งานได้ดีขึ้นกว่าเดิม

สำหรับฮาร์ดแวร์ของเครื่องวิเคราะห์วงจรดิจิตอล สามารถอธิบายถึงส่วนประกอบต่างๆ ได้ ดังนี้

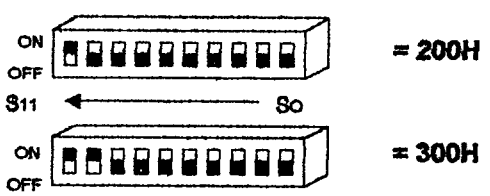
3.1 Interface Card

ในส่วนของ Interface Card นั้นจะเป็นส่วนที่ทำหน้าที่ในการติดต่อกับคอมพิวเตอร์และวงจรหลัก โดยจะอาศัยการมัลติเพล็กซ์ซึ่งจะถอดแอดเดรสพอร์ทที่ใช้ในการติดต่อมา 2 แอดเดรส โดยแอดเดรสแรกจะเป็นดัชนีในการอ้างถึงส่วนที่เราต้องการติดต่อ ส่วนแอดเดรสหลังจะใช้ในการส่งดาต้าไปยังส่วนที่เราต้องการติดต่อด้วย. เหตุที่ต้องใช้การมัลติเพล็กนั้นก็เนื่องจาก ถ้าใช้การติดต่อกับวงจรหลักโดยตรงเราจะต้องใช้สายสัญญาณจำนวนมาก ซึ่งจะทำให้ไม่สะดวกในเวลาใช้งาน แต่ก็จะมีข้อดีในแง่ของความเร็วเนื่องจากไม่ต้องอ้างดัชนีก่อนการติดต่อ รูปที่ 3.1 แสดงวงจรของ I/O Interface Card ซึ่งการ์ดนี้สามารถเสียบเข้ากับสล็อตของเครื่อง PC/XT และ PC/AT โดยอาศัยบัสตำแหน่ง (address bus), บัสข้อมูล (data bus), สัญญาณควบคุมการอ่านและเขียน, สัญญาณอินพุตและเอาต์พุต, สัญญาณรีเซ็ต และไฟเลี้ยงจากเครื่องพีซี



รูปที่ 3.1 แสดงวงจรของ Interface Card

การกำหนดแอดเดรสในการเลือกการทำงานของการ์ดนี้ สามารถกำหนดได้จากดิพสวิตช์ที่อยู่บนการ์ด ซึ่งโดยทั่วไปจะกำหนดไว้ที่แอดเดรส 201H ซึ่งเป็นแอดเดรสเกมส์พอร์ตของ 8088, 80286, 80386 และ 80486 แอดเดรสนี้เราสามารถตั้งใหม่ได้โดยการปรับที่ดิพสวิตช์ ซึ่งแอดเดรสที่เรากำหนดจากดิพสวิตช์นี้จะเป็นแอดเดรสของดัชนี ส่วนแอดเดรสของดาต้าจะมีค่ามากกว่า แอดเดรสของดัชนีอยู่หนึ่งเสมอซึ่งในส่วนนี้เราไม่ต้องกำหนด เพราะการ์ดจะรับรู้เอง รูปที่ 3.2 แสดงการกำหนดตำแหน่งแอดเดรสให้กับการ์ด



รูปที่ 3.2 แสดงการกำหนดตำแหน่งแอดเดรสให้กับการ์ด

การทำงานของวงจร

จากวงจรของ Interface Card ในรูปที่ 3.1 เมื่อมีการใช้คำสั่งอินพอร์ตหรือเอาพอร์ต จะทำให้สัญญาณ I/O READ หรือ I/O WRITE แอคตีฟ ซึ่งทั้งสองสัญญาณนี้จะไม่แอคตีฟพร้อมกัน เราจึงใช้ AND GATE เป็นตัวถอดสัญญาณ I/O ซึ่งจะไปอีนาเบิลไอซี 74LS688 เพื่อตรวจสอบดูว่าแอดเดรสที่อ้างมานั้นตรงกับที่เรากำหนดไว้ที่ดิพสวิตช์หรือไม่ ถ้าไม่ตรงก็จะไม่มีสัญญาณไปอีนาเบิลไอซี 74LS245 ซึ่งทำหน้าที่เป็นบัฟเฟอร์สองทิศทาง สำหรับทิศทางนี้จะถูกกำหนดจากสัญญาณ I/O READ แต่ถ้าแอดเดรสที่อ้างมานั้นตรงกับที่เรากำหนดไว้จะทำให้ไอซี 74LS245 ถูกอีนาเบิล สัญญาณข้อมูล D0-D7 ที่ผ่านบัฟเฟอร์แล้วนั้นจะกลายเป็นสัญญาณ XD0-XD7 และในการอีนาเบิลไอซี 74LS245 นี้จะต้องอาศัยสัญญาณ AEN ร่วมด้วย เพื่อให้แน่ใจว่าสัญญาณ I/O ที่เกิดขึ้นนั้นไม่ได้เกิดจากกระบวนการ DMA การแยกแอดเดรสของดัชนีนั้นจะใช้ไอซี 74LS688 สองตัวคือ IC2 และ IC3 โดยการเปรียบเทียบ A0-A11 กับค่าที่ตั้งไว้ที่ดิพสวิตช์แล้วนำเอาท์พุตของทั้งคู่มา OR กันที่ IC6/1 จะได้เป็นสัญญาณ XEA0 ส่วนแอดเดรสของดัชนีจะใช้ไอซี 74LS688 (IC4) ร่วมกับ IC2 โดยที่ IC4 จะไม่นำ A0 มาเปรียบเทียบ แต่จะนำ +5V มาเปรียบเทียบกับค่าที่กำหนดไว้ในดิพสวิตช์แทน แล้วนำเอาท์พุตของ IC2 และ IC 4 มา OR กันที่ IC6/2 ได้เป็นสัญญาณ XEA1 ส่วนสัญญาณ I/O READ และ I/O WRITE นั้นจะใช้ IC5/1 และ IC5/2 เป็นบัฟเฟอร์ได้เป็นสัญญาณ XI/O READ และสัญญาณ XI/O WRITE ตามลำดับ ส่วนสัญญาณ RESET จะใช้ IC6/4 เป็นบัฟเฟอร์ได้ โดยใช้เป็นสัญญาณ XRESET

3.2 วงจรหลัก

วงจรหลักนี้จะประกอบด้วยส่วนสำคัญๆอยู่ 3 ส่วนคือ

1.วงวจรดีมัลติเพล็กซ์

2.วงจร I/O

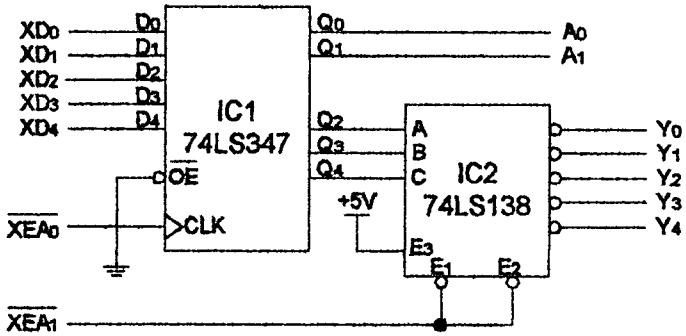
3.วงจร Comparator

3.2.1 วงจรดีมัลติเพล็กซ์

วงจรดีมัลติเพล็กซ์นั้นจะทำหน้าที่ตรงกันข้ามกับวงจรมัลติเพล็กซ์ซึ่งอยู่ในส่วนของ Interface Card ซึ่งประกอบด้วย IC1 74LS347 และ IC2 74LS138 เพียง 2 ตัว

การทำงานของวงจร

จากรูปที่ 3.3 IC1 จะมีหน้าที่แลตช์ข้อมูลที่อ้างมาจากแอดเดรสดัชนี โดยสัญญาณ XEA0 จะต่ออยู่กับขา CLK ของ IC1 สัญญาณที่นำเข้ามาแลตช์ได้แก่สัญญาณ XD0-XD4 สัญญาณ XD0 และ XD1 จะใช้เป็นสัญญาณอ้างแอดเดรสให้กับไอซี 8255 ในส่วนของวงจร I/O ส่วนสัญญาณ XD2-XD4 จะเป็นสัญญาณอินพุตให้กับไอซีดีมัลติเพล็กซ์ 74LS138 เอาท์พุตที่ได้จะเป็นสัญญาณที่ใช้ในการอีนาเบิลไอซี 8255 ในวงจร I/O โดยต้องรอให้มีการอ้างข้อมูลผ่านแอดเดรสของดาต้า ซึ่งจะทำให้สัญญาณ XEA1 แอคตีฟเป็นผลทำให้ IC2 ถูกอีนาเบิล



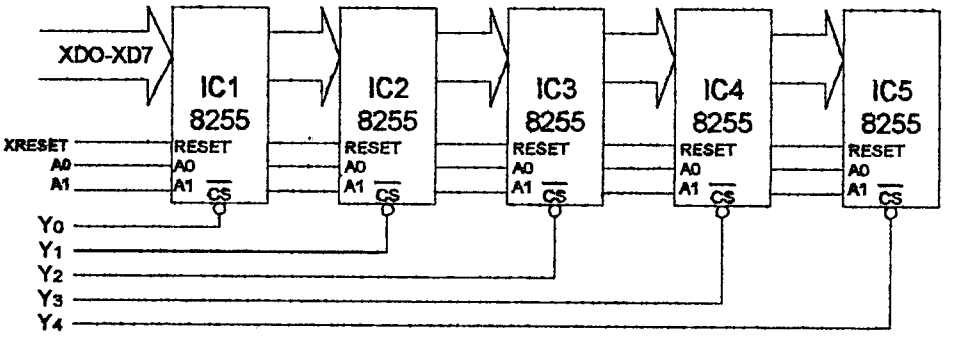
รูปที่ 3.3 แสดงวงจรดีมัลติเพล็กซ์

3.2.2 วงจร I/O

ส่วนของวงจร I/O จะประกอบไปด้วยไอซี 8255 ทั้งสิ้น 5 ตัว สาเหตุที่ต้องใช้ไอซี 8255 มากถึง 5 ตัวนั้นเนื่องจากต้องการให้ส่วนของอินพุตและเอาต์พุตเป็นอิสระต่อกันและสาเหตุที่เลือกใช้ไอซีเบอร์ 8255 ก็เพราะสายสัญญาณอินพุตและเอาต์พุททั้ง 24 เส้นมีระดับสัญญาณ TTL จึงเป็นการง่ายในการใช้ 8255 อินเตอร์เฟสกับวงจรลอจิกอื่นๆ นอกจากนี้รายังสามารถโปรแกรมให้แต่ละพอร์ตเป็นอินพุทและเอาต์พุตได้ และยังสามารถโปรแกรมให้ทำงานในโหมดต่างๆได้ถึง 3 โหมด หน้าที่ของไอซี 8255 แต่ละตัวนั้นมีดังนี้ IC1 เป็นเอาต์พุตให้กับอุปกรณ์ที่นำมาทดสอบ IC2 เป็นเอาต์พุตสำหรับควบคุมอิเลคทรอนิคส์สวิตช์ซึ่งจะใช้ไอซีเบอร์ 74LS126 เพื่อแยกส่วนของอินพุตและเอาต์พุตออกจากกัน ส่วน IC3 และ IC4 มีหน้าที่เป็นอินพุตนำสัญญาณเอาต์พุตจากวงจร Comparator เข้าไปประมวลผล สาเหตุที่ต้องใช้อินพุตขนาด 48 บิตก็เนื่องจากออกแบบให้ใช้กับอินพุตขนาด 24 ช่อง โดยในแต่ละช่องสามารถตรวจสอบระดับสัญญาณได้ 3 ระดับคือ LOW, HIGH และ HIGH IMPEDANCE โดยแต่ละช่องจะต้องใช้อินพุต 2 บิตจึงจะเพียงพอกับการใช้งาน สำหรับไอซี 8255 ตัวสุดท้ายนี้จะใช้ในการแสดงสถานะของวงจร เช่น EMPTY BUSSY เป็นต้น นอกจากนี้ยังใช้เป็นเอาต์พุตให้กับวงจร R-2R LADDER ในกรณีที่ต้องการให้วงจรสามารถปรับระดับของแรงดัน THESHOLD ได้

การทำงานของวงจร

สำหรับการทำงานของวงจรนั้นจะเริ่มจากเมื่อมีการอ้างข้อมูลผ่านแอดเดรสของดัชนี วงจรดีมัลติเพล็กซ์จะทำการแยกแอดเดรสที่ใช้สำหรับระบุว่าจะติดต่อกับพอร์ตอะไรภายในตัวของไอซี 8255 และเมื่อมีการอ้างข้อมูลผ่านแอดเดรสของดาต้าวงจรดีมัลติเพล็กซ์จะทำการอีนาเบิลไอซี 8255 ตัวที่เราต้องการจะติดต่อด้วย ทำให้สามารถรับส่งข้อมูลระหว่างตัวโปรแกรมและอุปกรณ์ภายนอกได้ จากรูปที่ 3.4 สัญญาณ RESET จากเครื่องคอมพิวเตอร์จะต่ออยู่กับขา RESET ของไอซี 8255 เพื่อทำการเคลียร์ค่าที่กำหนดไว้ในตัวของไอซี 8255 เมื่อมีการ RESET ที่เครื่องคอมพิวเตอร์



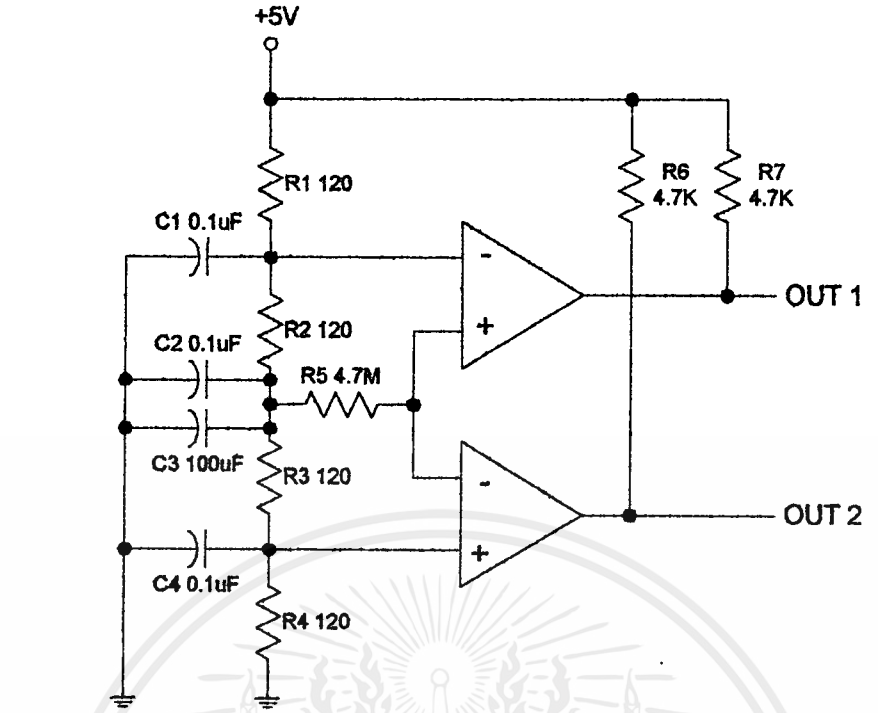
รูปที่ 3.4 แสดงวงจร I/O

3.2.3 วงจร Comparator

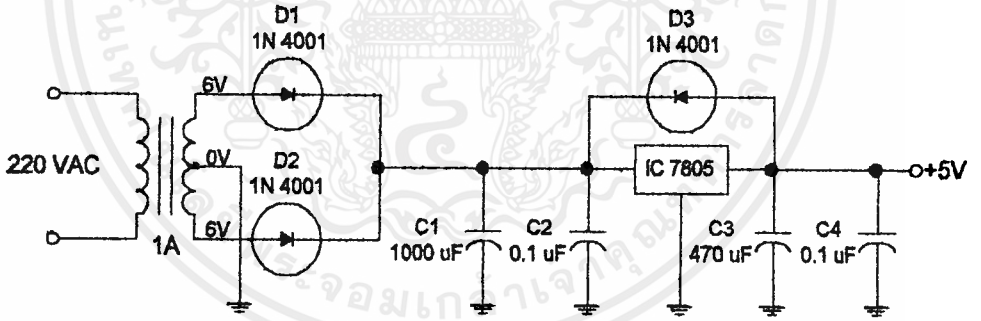
ในส่วนของวงจร Comparator นี้มีหน้าที่ในการเปรียบเทียบระดับของแรงดันที่เราต้องการทดสอบ โดยสามารถแยกระดับของแรงดันได้ 3 ระดับ ซึ่งจะอาศัยหลักการของ Window Comparator ดังแสดงในรูปที่ 3.5 ซึ่งจะประกอบด้วยไอซีเบอร์ LM339 เป็นหลักซึ่งจะมีออปแอมป์อยู่ภายใน 4 ชุด โดยถ้าแรงดันที่ทำการทดสอบนั้นมีระดับโลจิกเป็น LOW ("0") วงจรจะแสดงสถานะเป็น "01" เมื่อระดับโลจิกเป็น HIGH ("1") วงจรจะแสดงสถานะเป็น "10" แต่ถ้าเป็น HIGH IMPEDANCE คือขณะที่ไม่ได้ต่อกับอะไรไว้วงจรจะแสดงสถานะเป็น "11" วงจร Comparator นี้จะต้องสร้างทั้งหมด 24 ชุดเพื่อให้สามารถใช้วัดได้ 24 ช่องสัญญาณ

การทำงานของวงจร

วงจรจะประกอบด้วยออปแอมป์ 2 ตัวเพื่อทำเป็นวงจรเปรียบเทียบ ในส่วนของแรงดันอ้างอิงจะได้จากชุดแบ่งแรงดันที่ประกอบด้วย R1-R4 ซึ่งจะต้องมีค่าความผิดพลาดน้อยๆ จึงเลือกใช้ที่ค่าความผิดพลาด 1% เพื่อให้ได้ค่าแรงดันอ้างอิงที่ถูกต้องที่สุด ส่วนทางด้านเอาต์พุตของวงจรจะต้องทำการ Full Up ที่เอาต์พุตของออปแอมป์ทั้งสองด้วย ความต้านทานค่า 4.7K เนื่องจากเอาต์พุตของออปแอมป์เป็นแบบคอลเลคเตอร์เปิด ขณะที่ยังไม่ได้ทำการวัดแรงดันที่ขา 5 และขา 7 ของออปแอมป์จะมีค่าสูงกว่าแรงดันอ้างอิงทำให้เอาต์พุตของออปแอมป์ทั้งคู่เป็นโลจิก "1" เมื่อนำไปวัดกับจุดที่ต้องการทดสอบ ถ้าจุดนั้นเป็นโลจิก "0" แรงดันอ้างอิงที่ขา 4 จะสูงกว่าแรงดันทดสอบทำให้เอาต์พุตของออปแอมป์มีสภาวะเป็นโลจิก "0" ส่วนแรงดันอ้างอิงที่ขา 7 จะสูงกว่าแรงดันทดสอบที่ขา 6 เอาต์พุตของออปแอมป์จะมีสภาวะเป็นโลจิก "1" แต่ถ้าจุดที่ทดสอบเป็นโลจิก "1" แรงดันอ้างอิงที่ขา 4 จะต่ำกว่าแรงดันทดสอบที่ขา 5 ของออปแอมป์เอาต์พุตจะแสดงสภาวะเป็นโลจิก "1" ส่วนแรงดันอ้างอิงที่ขา 7 ก็จะต่ำกว่าแรงดันทดสอบที่ขา 6 ทำให้เอาต์พุตของออปแอมป์มีสภาวะเป็นโลจิค "0" ดังแสดงวงจรไว้ในรูปที่ 3.5



รูปที่ 3.5 แสดงวงจร Comparator



รูปที่ 3.6 แสดงวงจรแหล่งจ่ายไฟ

บทที่ 4
การทำงานของโปรแกรม

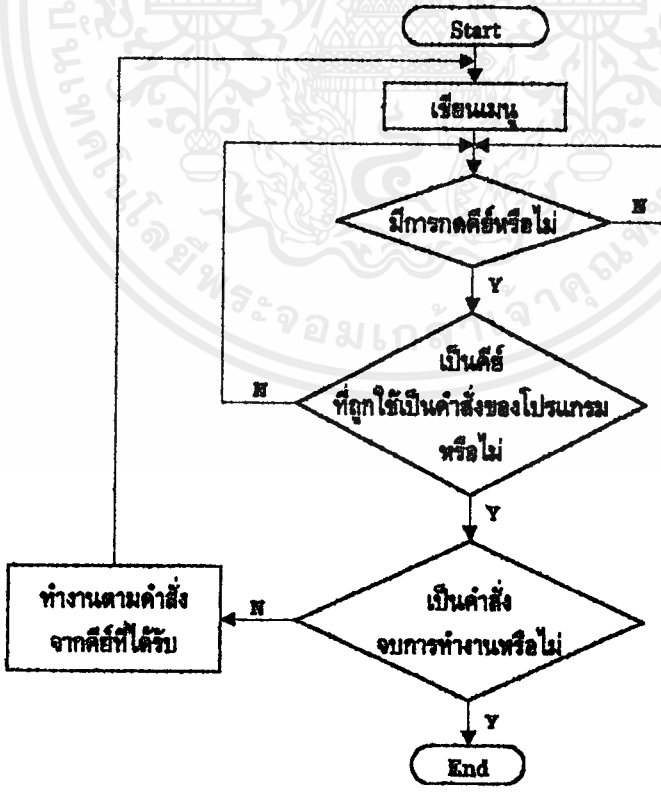
4.1 การทำงานของโปรแกรม

เนื่องจากในโปรเจคนี้ ต้องการมุ่งเน้นให้สามารถนำ hardware ที่ออกแบบไว้มาใช้ประโยชน์ได้หลายๆ อย่าง โดยที่กำหนดไว้คือ สามารถใช้งานเป็น logic analyzer ได้ (สามารถวัดสถานะ hi-impedance ได้), เป็นเครื่องอ่านหรือวิเคราะห์ฟังก์ชั่นในไอซี PAL (แบบที่ไม่มี Registor ภายใน) ซึ่งส่วนนี้ก็จะใช้โปรแกรม QM เข้ามาช่วยในการวิเคราะห์หาสมการอีกทีหนึ่ง และส่วนสุดท้ายคือ ใช้เป็นเครื่องตรวจสอบไอซี TTL และ CMOS ในตระกูล 74 series ซึ่งจะตรวจสอบได้เฉพาะ basic gate ทั่วไป เช่น AND OR NOR NAND NOT เท่านั้น

ในส่วนของ software นี้ จะแยกรายละเอียดของแต่ละโปรแกรมเป็น 4 หัวข้อคือ

4.1.1 Main Menu

เขียนด้วยภาษา C ทำหน้าที่เป็นตัวสั่งงานสำหรับโปรแกรมทั้งหมดในโปรเจค ซึ่งมีโฟลวชาร์ทการทำงานของโปรแกรมดังรูปที่ 4.1.1



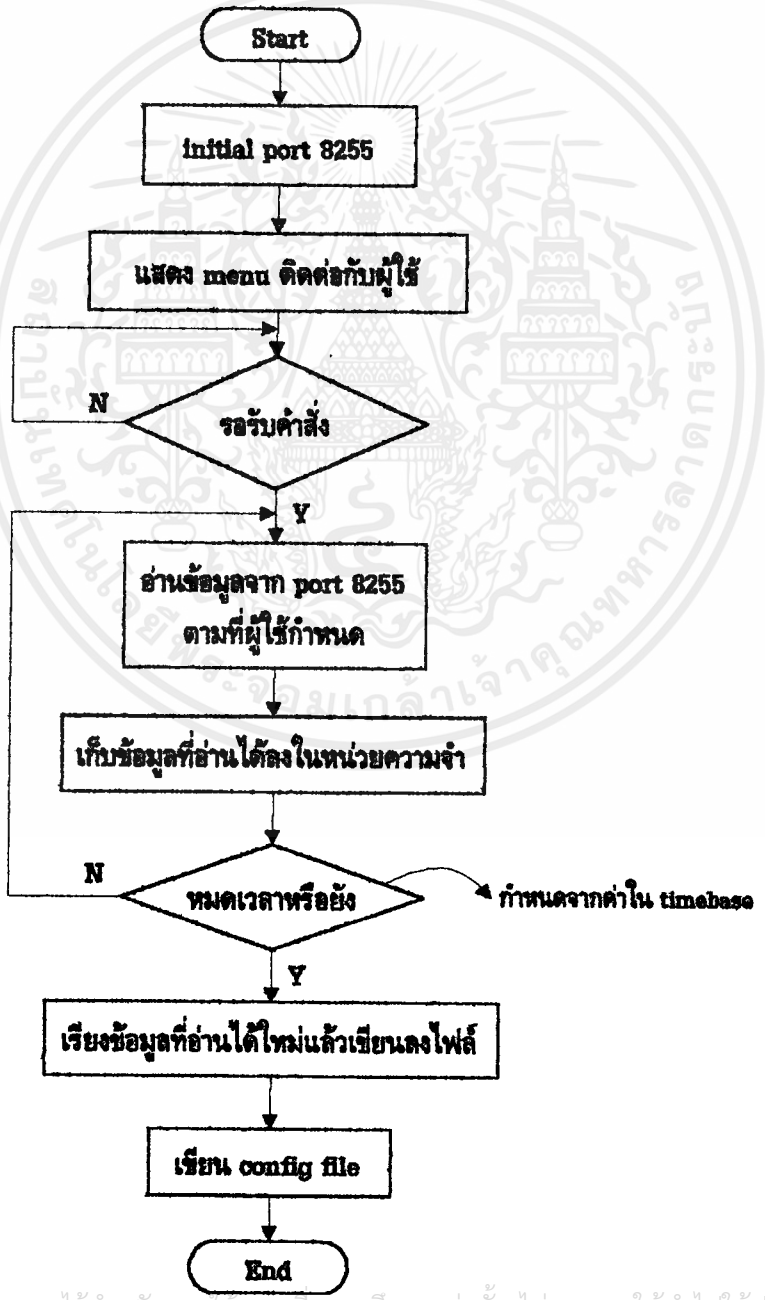
รูปที่ 4.1.1 โฟลวชาร์ทของโปรแกรม MAIN.EXE

4.1.2 Logic Analyzer

ในการใช้งานเครื่องเป็น Logic Analyzer นี้ ก็จะแยกการทำงานออกเป็น 2 ส่วน คือ ส่วนที่ทำหน้าที่อ่านและเก็บข้อมูล และส่วนที่ทำหน้าที่ประมวลผล และแสดงผล

4.1.2.1 ส่วนที่ทำหน้าที่อ่านและเก็บข้อมูล

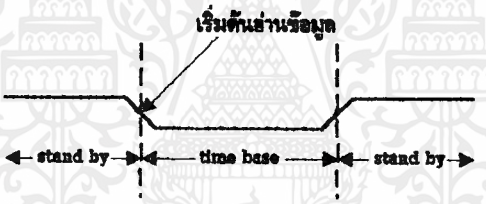
ส่วนนี้คือโปรแกรมที่กำหนดชื่อไว้เป็น CLA.EXE (Computer Logic Analyzer) เขียนด้วยภาษา Assembly มีหลักการทำงานดังนี้คือ จะรับคำสั่งต่างๆจากผู้ใช้ และอ่านข้อสถานะของสัญญาณที่วัดผ่านทางพอร์ทข้อมูล 8255 แล้วนำผลข้อมูลที่อ่านเข้ามาได้เก็บเป็น file ข้อมูล คือ LOGIC01.DAT ถึง LOGIC24.DAT และนอกจาก file ของข้อมูลแล้วยังมี file ที่เป็น config (LOGIC.CFG) สำหรับเป็นส่วนบอกให้ตัวประมวลผลทราบว่ามีข้อมูลกี่ channel โดยจะแสดงการทำงานต่างๆ ของโปรแกรมไว้ใน flowchart ดังรูปที่ 4.1.2



รูปที่ 4.1.2 โฟลวชาร์ทของโปรแกรม CLA.EXE

initial 8255 จะเป็นส่วนกำหนด control word ให้กับ 8255 แต่ละตัวในเครื่อง เมื่อทำการกำหนดค่าต่างๆ แล้วก็จะตรวจสอบข้อมูลที่ส่งออกไป ถ้าข้อมูลที่ส่งออกไปไม่ตรงกับข้อมูลที่รับเข้ามาก็แสดงว่าเครื่องทำงานผิดพลาด หรือไม่ได้ต่อเครื่องเข้ากับคอมพิวเตอร์ โปรแกรมก็จะจบการทำงานตั้งแต่ขั้นตอนนี้ ถ้าผ่านการตรวจสอบแล้ว ไฟสีเขียวที่หน้าปัดก็จะติดสว่างขึ้น แสดงว่าเครื่องพร้อมที่จะทำงานแล้ว

menu ติดต่อกับผู้ใช้ ในส่วนของ menu นี้ ผู้ใช้จะต้องเลือกว่าต้องการใช้กี่ channel โดยจำนวน channel ที่เลือกจะมีผลต่อความถี่สูงสุดที่จะทำการวัดได้ โดยถ้าเลือกจำนวน channel มากๆ ความถี่สูงสุดที่วัดได้ก็จะต่ำลง ดังนั้นถ้าผู้ใช้ต้องการวัดความถี่สูงๆ ก็ควรเลือกไว้ที่ 4 channel การเลือกจำนวน channel สามารถเลือกได้ดังนี้คือ 4, 8, 12, 16, 20 และ 24 channel ขั้นตอนต่อไปก็คือเลือก time base ว่าต้องการเวลาในการอ่านข้อมูลนานมากน้อยเพียงใด ซึ่ง time base นี้จะมีให้เลือกใช้ทั้งหมด 16 ค่า ถ้าหากผู้ใช้ต้องการวัดสัญญาณความถี่ต่ำๆ ก็คือ จะต้องเลือกค่าเวลาสูงสุดเอาไว้ก่อน ขั้นสุดท้าย ก่อนการสั่งให้อ่านข้อมูลก็คือ การ set threshold ว่าวงจรหรือสัญญาณที่เราจะวัดนั้นมีระดับแรงดันเป็นเท่าใด ถ้าเป็นวงจร logic ทั่วๆ ไป ก็ให้ set ไว้ที่ +5V เมื่อ set ค่าต่างๆ ครบแล้วจึงเริ่มสั่งให้โปรแกรมอ่านข้อมูลโดยเลือกไปที่ start ขณะเริ่มต้นการ start จะมีการส่งสัญญาณ trigger ไปยังภายนอกด้วยซึ่งผู้ใช้สามารถนำสัญญาณนี้ไปเป็นตัวสั่งให้เริ่มต้นส่งข้อมูลได้ โดยสัญญาณ trigger นี้จะ active ที่ negative edge ดังรูปที่ 4.1.3



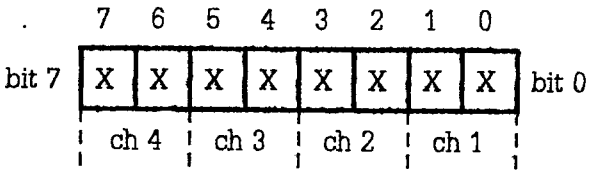
รูปที่ 4.1.3 ลักษณะของสัญญาณ trigger

การอ่านข้อมูลจาก 8255 ลักษณะการอ่านข้อมูลจะอ่านเข้ามาที่ละ 1 port โดยใน 1 port ก็จะมีข้อมูลทั้งหมด 4 channel เนื่องจาก 1 channel ต้องใช้ 2 bits ในการแทนสภาวะ 3 สภาวะ ดังตารางที่ 4.1.1

สถานะ	หมายเลข
โลจิก "0"	01
โลจิก "1"	10
Hi Impedance	00,11

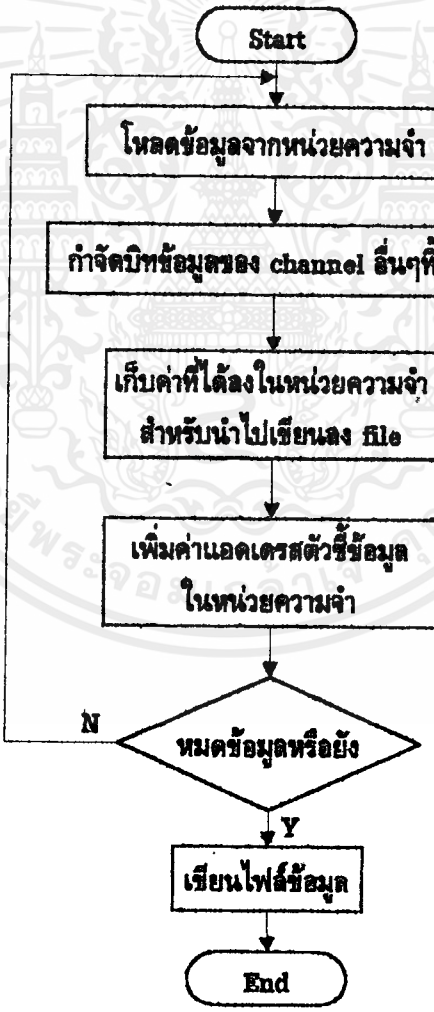
ตารางที่ 4.1.1 แสดงเลขฐานสองที่กำหนดสถานะโลจิก

เมื่ออ่านข้อมูลเข้ามาแล้ว ก็จะนำข้อมูลเก็บในหน่วยความจำ โดยไม่มีการทำอะไรกับข้อมูลทั้งสิ้น เพื่อต้องการให้ CPU ใช้เวลากับสิ่งเหล่านี้น้อยที่สุด เพื่อให้สามารถวัดสัญญาณที่มีความถี่สูงๆ ได้



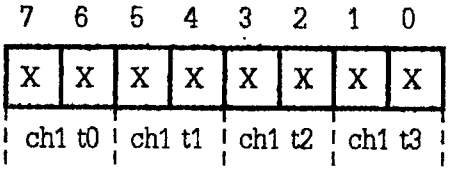
รูปที่ 4.1.4 ลักษณะของข้อมูลที่อ่านได้ 1 byte

การเรียงข้อมูลใหม่ การเรียงข้อมูลใหม่เพื่อให้โปรแกรมประมวลผลทำได้ง่ายขึ้น และเป็นการแยกข้อมูลออกมาเป็น channel ของใครของมัน เพื่อนำมาเก็บลงเป็น file ซึ่งจะมีขั้นตอนการจัดเรียงข้อมูลใหม่ดังแสดงใน flow chart รูปที่ 4.1.5



รูปที่ 4.1.5 โฟลว์ชาร์ตของการเรียงข้อมูล

ดังนั้นเราก็จะได้ format ของข้อมูลใหม่ ซึ่งใน 1 byte นั้น จะมีข้อมูลของ channel ใด channel หนึ่งเพียง channel เดียว ดังแสดงในรูปที่ 4.1.6

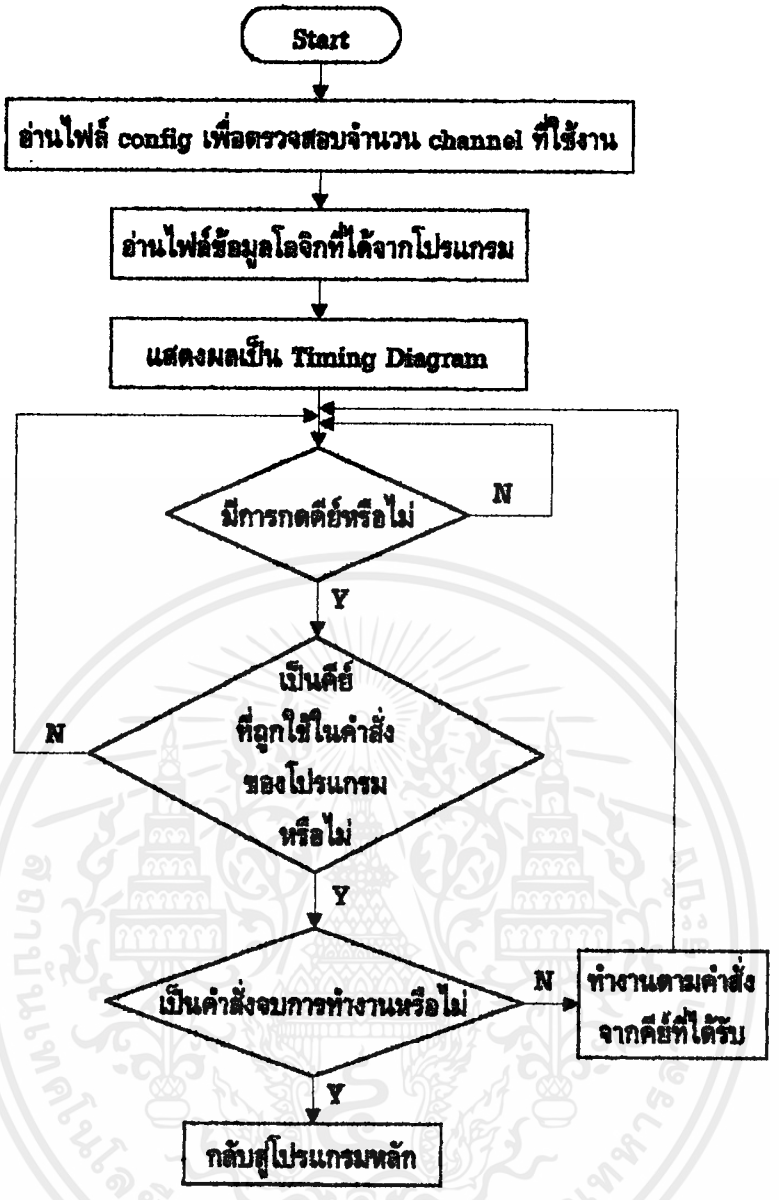


รูปที่ 4.1.6 ลักษณะข้อมูลใน 1 byte ของ channel 1

config file ถูกเขียนขึ้นมาเพื่อเป็นตัวบอกให้กับโปรแกรมส่วนแสดงผลรู้ว่ามีจำนวนของ channel ที่ใช้งานเท่าไร โดยจะเป็น binary file ซึ่งจะมีเพียง 1 byte เท่านั้น คือจะเป็นเลขฐาน 16 1 ตัว เช่นถ้าหากว่าเป็นการใช้งาน 4 channel เราก็จะเห็นรหัสตัวที่ 4 ของรหัส ASCII นั่นคือ ตัวข้าวหลามตัดอยู่ใน config file

4.1.2.2 ส่วนแสดงผลข้อมูล

ส่วนนี้คือโปรแกรมที่กำหนดชื่อไว้เป็น PULSE.EXE เขียนด้วยภาษาซี มีหลักการทำงานดังโฟลว์ชาร์ทรูปที่ 4.1.7 คือ อันดับแรกจะอ่าน config file (LOGIC.CFG) เพื่อตรวจสอบว่าเป็นการใช้งานกี่ Channel หลังจากนั้นจะทำการตรวจสอบไฟล์ต่างๆที่ใช้ในโปรแกรม และตรวจสอบว่า มีไฟล์ข้อมูลสถานะทางโลจิก(FUNCTxx.DAT) อยู่ครบตามจำนวน Channel ที่ใช้งานหรือไม่ ถ้าหากไฟล์ใด ไฟล์หนึ่งไม่ครบโปรแกรมจะหยุดการทำงานทันที และเมื่อตรวจสอบไฟล์เรียบร้อยแล้วโปรแกรมก็จะทำการอ่านข้อมูลสถานะทางโลจิกที่ถูกสร้างขึ้นโดย CLA.EXE เพื่อนำเลขฐานสองในไฟล์ข้อมูลมาเก็บไว้ในหน่วยความจำ โดยจะเก็บในรูปของตัวแปรชุด (แบ่งเป็นแต่ละ Channel) หลังจากนั้นจึงนำข้อมูลสถานะทางโลจิกดังกล่าวที่เก็บไว้ในตัวแปรชุด มาทำการแปลงให้เป็นรูปกราฟฟิก คือเป็น Timing Diagram นั่นเองโดยถ้าข้อมูลมีสถานะเป็นไฮอิมพีแดนซ์ Timing Diagram จะเป็นช่องว่าง (ไม่มีเส้นปรากฏ)



รูปที่ 4.1.7 โฟลว์ชาร์ตของโปรแกรม PULSE.EXE

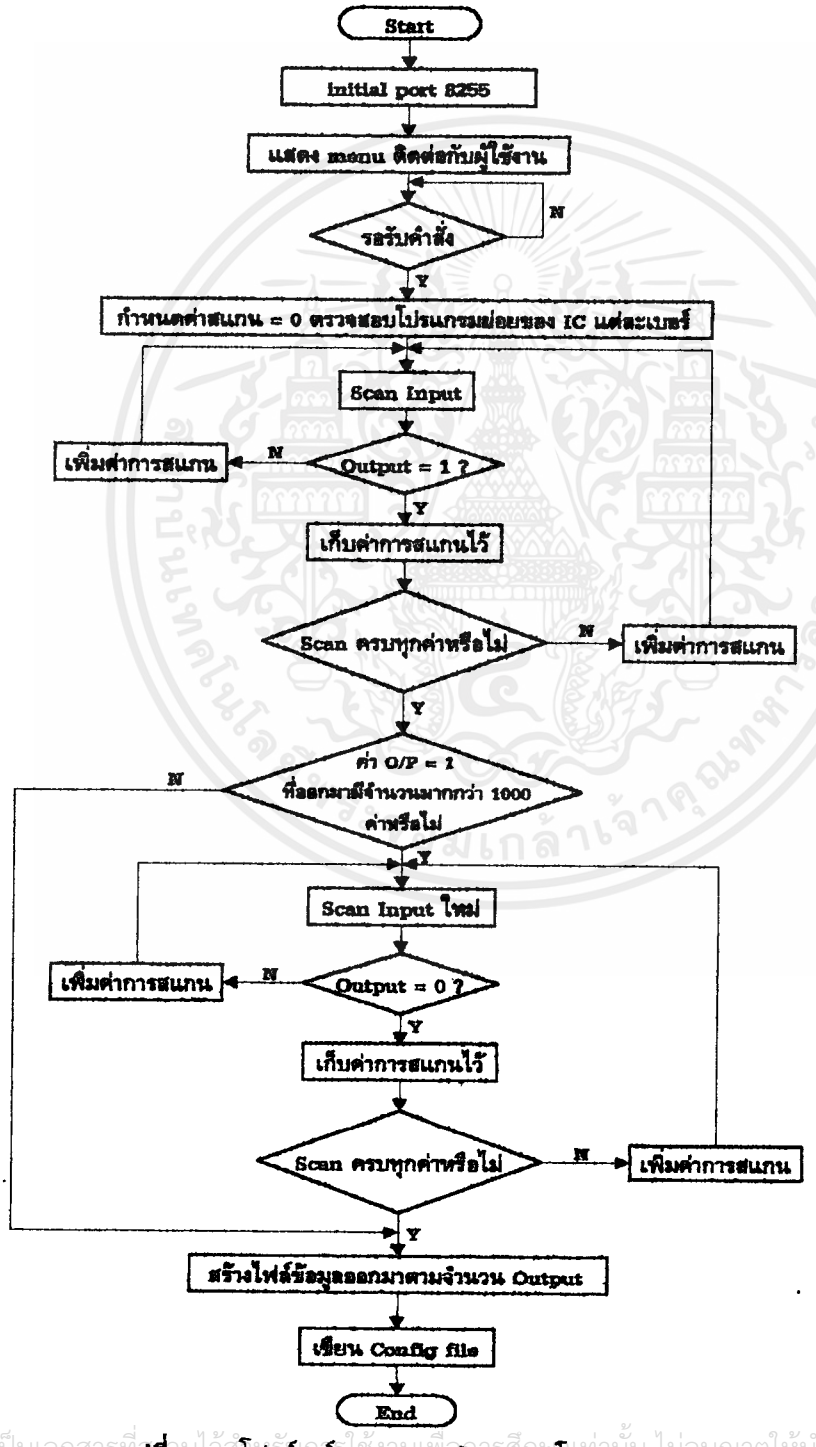
4.1.3 การอ่านหรือวิเคราะห์โปรแกรมจากไอซี PAL

การอ่านหรือวิเคราะห์โปรแกรมจากไอซี PAL นี้ก็จะมีการแบ่งแยกหน้าที่เป็น 2 ส่วนคือ ส่วนที่ทำหน้าที่อ่านข้อมูลจากไอซี PAL และส่วนที่ทำหน้าที่วิเคราะห์หาสมการออกมา

4.1.3.1 การอ่านข้อมูลจาก PAL

ในการข้อมูลจาก PAL นี้จะอาศัยหลักการป้อนค่าต่างๆ ให้กับอินพุทของไอซี PAL เช่น ถ้าเป็นไอซี PAL ที่มี 16 อินพุท ก็ต้องป้อนค่าให้กับอินพุทของไอซีทั้งหมด 2^{16} ค่า หรือ 65535 ค่า คือเริ่มจาก 000F - FFFFH (หรือถ้าหากมี 16 อินพุท แต่ใช้งานเพียง 4 อินพุท ก็จะป้อนค่าให้กับอินพุทของไอซีเพียง 2^4 ค่า ซึ่งจะทำให้การทำงานของการส่งข้อมูลและการวิเคราะห์สมการทำได้เร็วขึ้น) แล้วทำการตรวจสอบว่าที่

อินพุทแต่ละค่านั้น เอาท์พุทของมันมีค่าเป็นอะไร ซึ่งเราจะสนใจเฉพาะแต่อินพุทที่ให้เอาท์พุทเป็นโลจิก " 1 " เท่านั้น (ในรูปของมินเทอม) แล้วนำค่าอินพุทนั้นเก็บเป็น file ข้อมูลเพื่อวิเคราะห์หาสมการอีกทีหนึ่ง ซึ่งโปรแกรมที่ทำหน้าที่เหล่านี้คือ โปรแกรม PLA.EXE (เขียนด้วยภาษา Assembly) จากหลักพื้นฐานข้างต้นไอซี PAL ที่ใช้จะต้องไม่มี Register อยู่ภายในตัว เนื่องจากเราจะไม่สามารถทราบได้ว่าสภาวะเริ่มต้นเป็นอะไร (นั่นคือไม่สามารถวิเคราะห์สมการที่เป็น Sequence ได้) flow chart การทำงานของโปรแกรมแสดงในรูปที่ 4.1.8



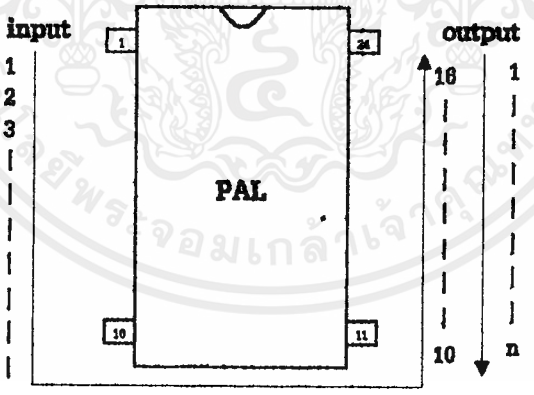
รูปที่ 4.1.8 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม PLA.EXE

เมนูติดต่อกับผู้ใช้ ก่อนการเริ่มใช้งาน ผู้ใช้จะต้องเลือกเบอร์ไอซีที่ต้องการ ซึ่งจะสามารถเลือกได้ทั้งหมด 8 บอร์ด้วยกัน คือ เบอร์ 16L8, 16L4, 16L2, 14L8, 14L4, 12L10, 12L6 และ 12L4 เมื่อเลือกเบอร์ที่ต้องการได้แล้ว ผู้ใช้ก็จะต้องเลือกค่าหน่วงเวลา เนื่องจากคอมพิวเตอร์แต่ละเครื่องจะมีความเร็วในการทำงานไม่เท่ากัน

การตรวจสอบโปรแกรมย่อย เมื่อผู้ใช้ทำการเลือกเบอร์ไอซีและกำหนดค่าการหน่วงเวลาแล้ว ทำการ start โปรแกรม จากนั้น โปรแกรมจะนำเบอร์ไอซีที่ผู้ใช้เลือกไปเรียกหาโปรแกรมย่อยของแต่ละเบอร์ แล้วเริ่มต้นการทำงานตามโปรแกรมย่อยเหล่านั้น

การ scan input และการรับค่าจาก output ปัญหาในการ scan input และรับค่าจาก output ก็คือแต่ละเบอร์จะมีตำแหน่งขา input และขา output ไม่ตรงกัน นอกจากนั้นไอซีบางเบอร์ยังใช้ขาเดียวกันทำหน้าที่เป็นทั้งอินพุทและเอาท์พุท ทำให้ต้องมีการเลื่อน bit ของค่าการ scan ให้ตรงกับ input ของไอซีซึ่งแต่ละเบอร์ก็จะมีการเลื่อน bit ข้อมูลที่ต่างกันไป ทั้งอินพุทและเอาท์พุท

การเริ่มต้นการ scan ก็จะใช้การเพิ่มค่าการ scan ขึ้นที่ละ 1 ค่า แล้วตรวจสอบว่าเอาท์พุทเป็น 1 หรือไม่ ถ้าไม่ก็จะเพิ่มค่าอินพุทขึ้นอีก 1 แล้วตรวจสอบ output ใหม่ แต่ถ้าเป็น "1" ก็จะเก็บค่าของ input ที่ทำให้เอาท์พุทเป็น 1 ไว้ แล้วจึงเพิ่มค่าการ scan ขึ้นอีก 1 ค่า แล้วเริ่มต้นการ scan ใหม่ โดยเมื่อ scan ครบทุกค่าแล้วก็จะนำค่า input ที่ทำให้ output เป็น "1" มาเขียนเป็น file ข้อมูลโดยมีชื่อว่า FUNCTXX.DAT โดยค่าตัวเลขท้าย 2 ตัว เป็นตัวบอกหมายเลขของ function ของไอซี ซึ่งการเรียงขา input และ output จะมีรูปแบบการเรียงดังรูปที่ 4.1.9



รูปที่ 4.1.9 รูปแบบการเรียงขาของไอซี

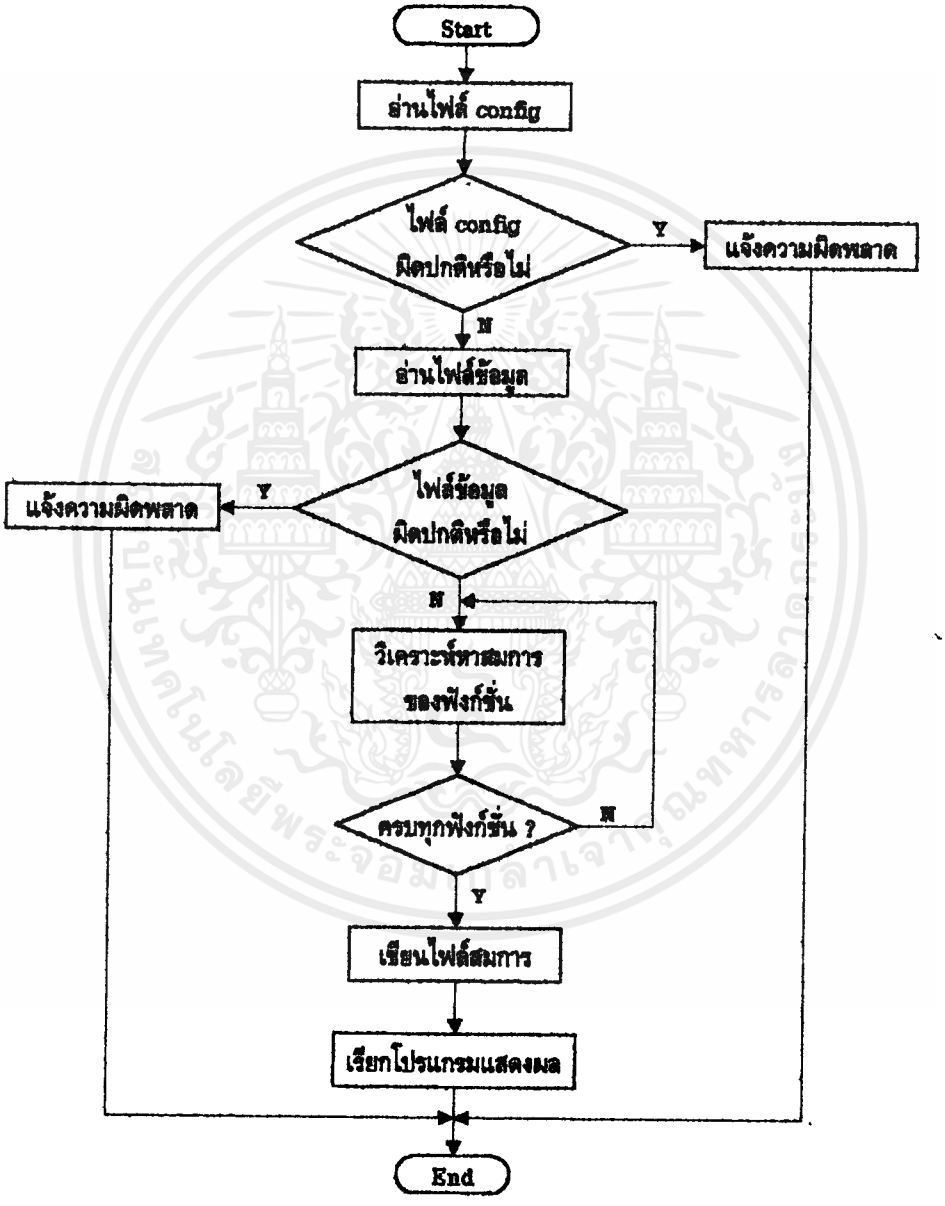
config file มีลักษณะเป็น Text file โดยจะเก็บพารามิเตอร์ของ ลักษณะของข้อมูล, จำนวนอินพุท และ จำนวนเอาท์พุท ซึ่งจะมีรูปแบบดังนี้

- พารามิเตอร์ตัวที่ 1 เป็นตัวบอกว่าวิเคราะห์ด้วยสมการแบบมินเทอม หรือแมกเทอม โดยจะแทนด้วยตัวอักษร "T" และ "A" ตามลำดับ
- พารามิเตอร์ตัวที่ 2 เป็นตัวบอกจำนวนอินพุทของ ไอซีเบอร์ที่ทำการวิเคราะห์
- พารามิเตอร์ตัวที่ 3 เป็นตัวบอกจำนวนเอาท์พุทของ ไอซีเบอร์ที่ทำการวิเคราะห์

ยกตัวอย่างเช่น config file คือ " I 16 8 " ก็จะหมายความว่า เป็นการวิเคราะห์โดยใช้มินเทอม เป็นไอซี PAL ที่มี 16 อินพุท และมี 8 เอาท์พุท เป็นต้น

4.1.3.2 ส่วนวิเคราะห์หาสมการ

ส่วนนี้จะใช้โปรแกรม QM.EXE ซึ่งได้รับมาจากอาจารย์ที่ปรึกษา เป็นโปรแกรมที่เขียนขึ้นมาโดยนักศึกษาสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยทางกลุ่มโครงงานได้นำมาปรับปรุง และปรับเปลี่ยนให้เข้ากับการทำงานที่ต้องการ ซึ่งโปรแกรมมีลำดับขั้นการทำงานดัง flow chart ดังรูปที่ 4.1.10



รูปที่ 4.1.10 แสดงโฟลว์ชาร์ตของส่วนวิเคราะห์สมการ

การอ่าน config file ขั้นแรก โปรแกรมจะทำการอ่าน config file เพื่อตรวจสอบว่าฟังก์ชั่นที่จะให้ วิเคราะห์นี้ เป็นมินเทอม หรือแมกเทอม และตรวจสอบว่ามีจำนวนอินพุทและเอาท์พุทเท่าใด เป็นค่าที่เป็น

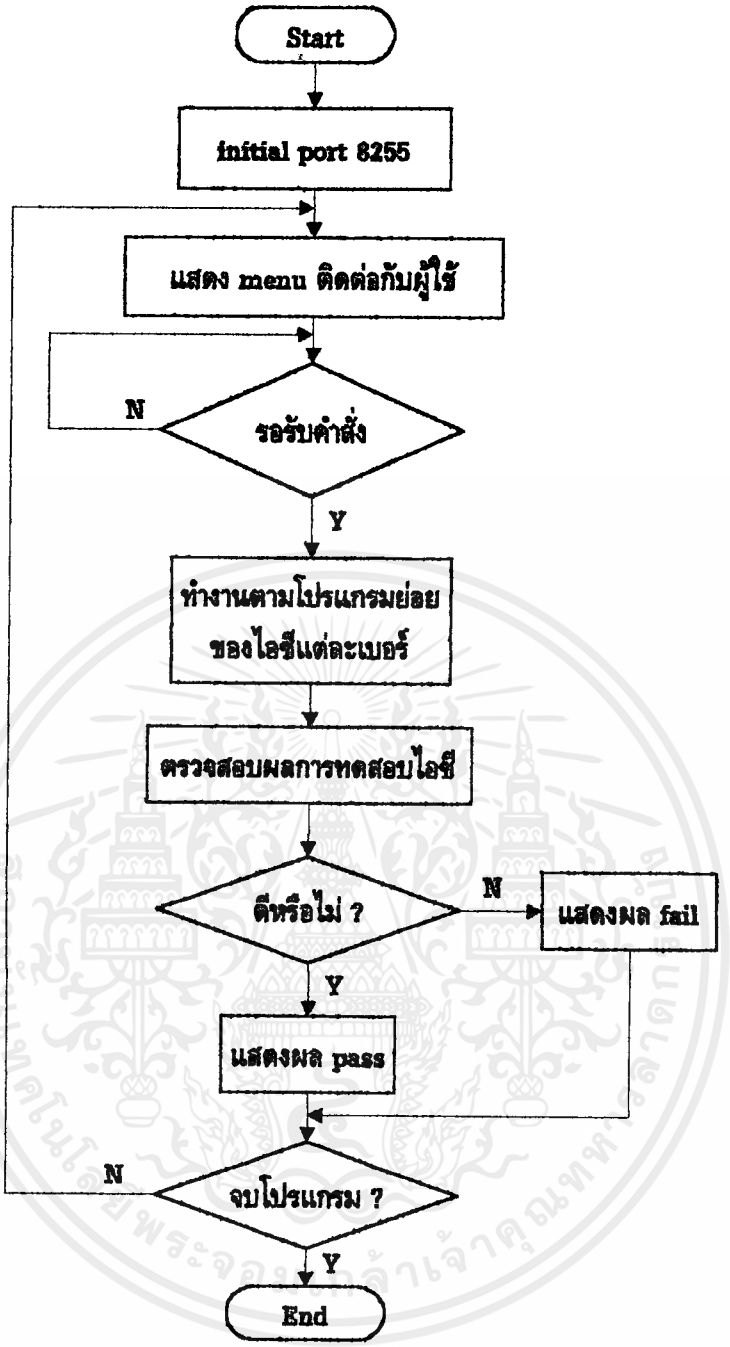
ไปได้หรือไม่ คือถ้าเป็นค่าที่ผิดปกติ โปรแกรมก็จะรายงานให้ทราบ และจบโปรแกรมลง แต่ถ้าเป็นค่าที่เป็นไปได้ โปรแกรมก็จะไปอ่านไฟล์ข้อมูลที่ถูกสร้างขึ้นโดยโปรแกรม PAL.EXE (FUNCTXX.DAT) แล้ววิเคราะห์ว่าค่าค่าเทอมต่างๆ ในฟังก์ชั่นเหล่านั้นเป็นไปได้หรือไม่ (เช่น มีค่ามินเทอมคือ 3544 แต่มี อินพุท เพียง 11 ก็จะเป็นไปไม่ได้ แต่ในความเป็นจริงแล้วไม่น่าจะมีความผิดพลาดเช่นนี้ขึ้นได้ แต่ก็ต้องทำการกันเอาไว้เพื่อความสมบูรณ์) ถ้าเป็นไปไม่ได้โปรแกรมก็จะแจ้งให้ทราบและจบโปรแกรม

การวิเคราะห์สมการ ถ้าไม่มีอะไรที่ผิดพลาดในไฟล์ข้อมูล และไฟล์ config โปรแกรมก็จะทำการวิเคราะห์หาสมการออกมาให้ โดยใช้โปรแกรม QM.EXE โปรแกรมนี้จะเป็นตัวที่ทำการวิเคราะห์หาสมการจากฟังก์ชั่นที่ได้รับมาจาก โปรแกรม PAL.EXE โดยอาศัยหลักการของ QM (Quine McCluskey method) คือเป็นหลักการที่จะใช้เทอมของแต่ละ function ร่วมกันให้มากที่สุดเพื่อให้ประหยัดอุปกรณ์ หรือ GATE แต่เนื่องจากว่า function ที่ถูกโปรแกรมมาใน PAL นั้น มิได้ใช้อาท์พุทร่วมกัน จึงได้ทำการแก้ไขให้โปรแกรมทำการวิเคราะห์ทีละฟังก์ชั่น แล้วได้สมการออกมาเป็นของใครของมันเลย (ซึ่งจากการทดสอบดูจะได้สมการที่สั้นกว่าคิดแบบ QM ปกติ) เมื่อได้สมการออกมาแล้วก็จะเขียนไว้เป็น file equation (QM.EQ)

การแสดงผล หลังจากที่ทำการวิเคราะห์สมการออกมาได้แล้ว โปรแกรมจะไปเรียกโปรแกรมแสดงผลออกมา โปรแกรมแสดงผลนี้ จะทำการอ่านไฟล์ QM.EQ แล้วนำมาแสดงผลให้บนจอภาพ ด้วยโปรแกรม READ.EXE โดยสามารถใช้ปุ่มลูกศร ขึ้น, ลง, ซ้าย, ขวา หรือ Page Up, Page Down, Home, End ในการเลื่อนดูสมการได้

4.1.4 การตรวจสอบไอซี TTL และ CMOS

โปรแกรมที่ทำหน้าที่นี้คือโปรแกรม TTL.EXE โดยการตรวจสอบไอซี TTL และ CMOS นั้น จะต้องเป็นตระกูล 74 series เท่านั้น และจะต้องเป็นไอซี logic gate พื้นฐานทั่วๆ ไปด้วย เช่น AND, OR, NOR, NAND, NOT ซึ่งจะมีการทำงานดังแสดงใน flow chart ดังรูปที่ 4.1.11

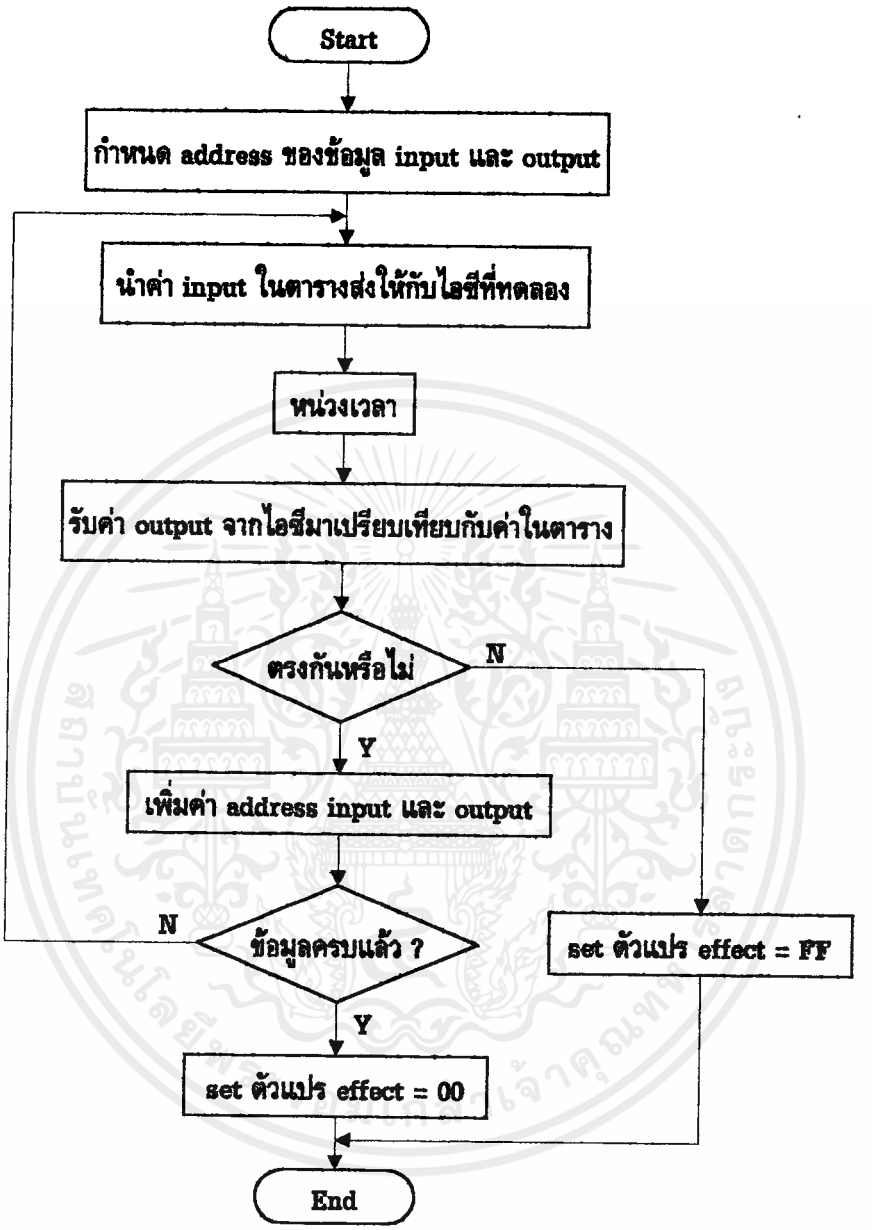


รูปที่ 4.1.11 แสดงโฟลว์ชาร์ตของโปรแกรมตรวจสอบไอซี TTL และ CMOS

เมนูติดต่อกับผู้ใช้ ผู้ใช้จะต้องกำหนดเบอร์ของไอซีลงไปว่าเป็นเบอร์อะไร แล้วโปรแกรมจะตรวจสอบว่ามีข้อมูลของไอซีเบอร์นั้นอยู่หรือไม่ ถ้ามี ขั้นต่อไปก็จะเป็นการกำหนดช่วงเวลาในการหน่วงเวลาไว้ว่าส่งข้อมูลให้นานเท่าไร จึงจะรับผลจากไอซีเข้ามาตรวจสอบ

โปรแกรมย่อยของไอซีแต่ละเบอร์ เนื่องจากไอซีแต่ละเบอร์จะมีตำแหน่งขา input และ output ไม่ตรงกัน จึงต้องแยกการทำงานเป็นไอซีแต่ละเบอร์ เพื่อให้ง่ายต่อการเขียนโปรแกรมและลดข้อผิดพลาด ในการส่งข้อมูลไปยัง input และรับ output จากไอซีที่ต้องการตรวจสอบ

การตรวจผลการทดสอบไอซี หลักการในการตรวจสอบคือ นำผลที่ได้จากการส่งค่าไปให้ input ของไอซีแต่ละเบอร์ แล้วรับเอา output ที่ถูกต้องของไอซีเบอร์นั้นๆ ซึ่งจะมีหลักการตรวจสอบดังแสดงใน flow chart ดังรูป 4.1.12



รูปที่ 4.1.12 แสดงโฟลว์ชาร์ตของหลักการตรวจไอซี

การแสดงผล โดยจะดูจากค่าในตัวแปร effect โดยถ้าค่าเป็น 00 แสดงว่าไอซีตัวนั้นดี จะแสดงผลเป็น pass แต่ถ้าค่าใน effect เป็น FF แสดงว่าไอซีตัวนั้นเสีย จะแสดงผลเป็น fail

4.2 หลักการในการเขียนโปรแกรมในการย้ายหน่วยความจำไว้บน XMS

ขั้นแรกจะต้องกำหนดขนาดหน่วยความจำที่จะใช้ก่อน หน่วยความจำที่เราจองนี้จะเป็นหน่วยความจำเหนือ 1,000 Kbytes คือเราจะจองเท่าไรขึ้นอยู่กับหน่วยความจำที่เรามีอยู่ ในโครงงานนี้ เราจะจองไว้ 470,000 Block สำหรับ record ทั่วไป และจอง 10000 Block สำหรับ record PI

ส่วนหน่วยความจำที่นำไปใช้งานได้จริงๆ นั้น อยู่ภายใต้ 640 Kbytes ที่เป็นข้อจำกัดของ MS-DOS ที่จะสามารถมองเห็นได้ แต่ภายใน 640 Kbytes เราไม่สามรถนำไปใช้ในการคำนวณสำหรับ Logic Analyzer ได้ทั้งหมด เพราะหน่วยความจำบางส่วนจะถูกจองเอาไว้เก็บโปรแกรม ไดรเวอร์ต่างๆ ที่ใช้ใน Windows และ DOS พื้นที่หน่วยที่เหลือ จะเหลือไม่ถึง 640 Kbytes เช่นอาจจะเหลือประมาณ 450 Kbytes ซึ่งปกติแล้วถ้าเป็น software เก่า จะไม่สามารถคำนวณได้ มันต้องการหน่วยความจำถึง 600 Kbytes ขึ้นไป ทางที่จะทำให้มันคำนวณได้ก็คือ ทำการเขียน Config ใหม่ เพื่อลบไดรเวอร์ต่างๆ ออกไปก่อน ขณะใช้ Logic Analyzer เพื่อให้หน่วยความจำเหลือพอคำนวณ ซึ่งเป็นวิธีที่ไม่สะดวกในการนำเครื่อง Logic Analyzer ไปใช้งาน เราจะต้องทำให้หน่วยความจำที่ใช้งานให้น้อยกว่านี้ คือน้อยกว่า 450 Kbytes ที่เหลืออยู่ เราเขียนโปรแกรมขึ้นมาหนึ่งตัวเพื่อให้เป็นตัวจัดการหน่วยความจำเสียใหม่ แทนที่โปรแกรมทั้งหมดจะอยู่ใน Conventional

จริงแล้วมันไม่จำเป็นต้องอยู่ใน Conventional ทั้งหมด เพราะมันไม่ได้ใช้งานพร้อมกัน เราก็ใช้ช่องว่างอันนี้เป็นหลักในการเขียนโปรแกรม คือทำการเขียนโปรแกรมจองหน่วยความจำบน XMS ไว้จำนวนหนึ่งเพื่อที่จะทำการย้ายข้อมูลที่ยังไม่ได้ใช้งานในขณะนั้น ของตัว Logic Analyzer ไปเก็บไว้บน XMS ก่อน ส่วนที่ใช้งานจริงๆ ก็ยังอยู่ใน Conventional เหมือนเดิม แต่ส่วนที่เหลือนี้จะใช้หน่วยความจำใน Conventional น้อยลง แต่ตัวโปรแกรมยังสามารถคำนวณผลได้เหมือนเดิม คือสามารถคำนวณผลได้ในขณะที่มีหน่วยความจำเหลือเพียง 450 Kbytes ซึ่งโปรแกรมการทำงานเก่า ไม่สามารถที่จะคำนวณได้ ในขณะที่หน่วยความจำเหลือน้อยกว่า 450 Kbytes

ต่อไปมาดูหลักการทำงานของโปรแกรมที่เขียนขึ้นมา ก่อนอื่นต้องกำหนดหน่วยความจำ ที่เราต้องการที่กล่าวมาแล้วในตอนต้นได้ดังนี้

```
# define  MaxPi     10000
# define  MaxList   470000
```

หลังจากนั้น เขียนฟังก์ชันขึ้นมา เพื่อเป็นตัวรับค่า Block ที่เราต้องการใช้งาน และคืนหน่วยความจำให้แก่ระบบในกรณีที่จองหน่วยความจำไม่ได้

```
int   allocterm1 (struct TERM1 *term1);
struct record far *Term1( struct TERM1 *term1, int index);
void freeterm1 (struct TERM1 *term1);
```

term อื่นๆ ก็เขียนได้ในทำนองเดียวกันนี้

ต่อไปต้องมาเขียนสตรัคเจอร์ หรือกลุ่มข้อมูลชนิดโครงสร้างที่ได้กำหนดไว้เป็นฟังก์ชันในตอนแรก

```
struct record {
        word  Data;
        word  Cutout;
        };
```

ตัวสตรัคเจอร์ที่กำหนดขึ้นจะใช้หน่วยความจำแล้วแต่ขนาดของตัวแปรที่กำหนด ในที่นี้สมมติให้เท่ากับ 4 ไบต์ แต่เราจองหน่วยความจำใน Conventional ไว้ 256 Block คือ ใช้หน่วยความจำไปทั้งหมด 256 x 4 = 1024 ไบต์ คือใช้หน่วยความจำไป 1024 ไบต์ จากที่มีอยู่ 640 Kbytes ต่อไปเป็นสตรัคเจอร์ที่ใช้กำหนดค่าหน่วยความจำของ XMS

```
struct Term1 {
      struct record far *term1; เป็นตัวแปร pointer ไว้ชี้ตำแหน่งเริ่มต้นของ Conventional
      int TermNum;   ใน term1 กำหนดขนาดของ XMS เป็นกี่ Block คือ 150000 Block
      int  handle;     เป็นตัวเก็บตำแหน่งของ XMS Memory
      int  oldpage;    หมายเลข Page ที่อยู่ใน Conventional Memory
            };
```

ใน term อื่นๆ ก็กำหนดแบบเดียวกันนี้

ลองทำการคำนวณหน่วยความจำที่ใช้เปรียบเทียบระหว่างโปรแกรมใหม่กับโปรแกรมเก่า ก่อนอื่นมาดูจำนวน page ที่มีได้

$$\text{คำนวณได้จาก } \frac{470000 \text{ block}}{256 \text{ block}} = 1836 \text{ page}$$

$$\text{และ } \frac{10000 \text{ block}}{64 \text{ block}} = 156 \text{ page}$$

ก่อนที่โปรแกรมนี้ จะทำการจองหน่วยความจำ หรือทำอะไรได้ จะต้องตรวจสอบก่อนว่า เครื่องที่ใช้อยู่ใน HIMEM.SYS ถูก Load อยู่หรือไม่ ถ้าไม่โปรแกรมนี้จะทำงานไม่ได้ เพราะว่าต้องให้หน่วยความจำเหนือ 1Mbyte ในการเก็บค่า ถ้าเราไม่ load ไว้ โปรแกรมนี้จะทำงานไม่ได้ โดยจะเขียนเป็นคำสั่งได้ดังนี้

```
if (! XMS installed ( ) ) {
        printf (" This program use Extended memory and HIMEM.SYS !\n ");
        Exit (1);
                        }
```

ความหมายของโปรแกรมดังกล่าวคือ ถ้าฟังก์ชัน XMS installed ( ) มีค่าเป็น 0 แสดงว่าเครื่องคอมพิวเตอร์ไม่ได้ Load HIMEM.SYS ไว้ โปรแกรมส่วนนี้จะทำงานสั่งให้โปรแกรมออกไปโดยไม่ต้องทำคำสั่งอื่นๆ ที่เหลือ แต่ถ้ามีการ Load ฟังก์ชัน XMS installed ( ) จะเป็น 1 และจะข้ามคำสั่งนี้ไป แล้วจึงทำคำสั่งอื่นต่อไปได้ หลังจากทำการตรวจสอบ HIMEM แล้ว ก็จะมาทำโปรแกรมที่ทำการจองหน่วยความจำ XMS กับ Conventional ซึ่งเขียนได้ดังนี้

```
term1.TermNum = MaxList;
if (! allocterm1 (&term1) ) {
    printf ("Error Can't Alloc Memory for TERM1 \n");
    exit (-1);
    }
```

MaxList เป็นค่า Block ที่เราต้องการกำหนดไว้ทีแรก ในที่นี้เราจะกำหนดเป็น 470000

ฟังก์ชัน allocterm จะรับค่ามาจากฟังก์ชัน int allocterm 1 (struct TERM1 * term1) อีกทีหนึ่ง โดยเมื่อฟังก์ชันส่งค่ามาบอกว่าจองหน่วยความจำได้หรือไม่ได้ คำสั่ง if จะทำการตรวจสอบ ถ้าเกิดว่าจองหน่วยความจำไม่ได้ จะส่งผลกลับมาเป็น "0" แสดงว่าเกิด error ในการจองเกิดขึ้น อาจจะจองไม่ได้ หรือหน่วยความจำไม่พอ ตัวโปรแกรมก็จะทำการแสดงผล "Error can't Alloc Memory for term1 " แล้วทำการออกจากโปรแกรมเช่นเดียวกับคำสั่งอื่นๆ หลังจากนี้ จะไม่ถูกกระทำ ถ้าเกิดกรณีไม่เกิด error คือจองหน่วยความจำได้ ก็จะทำการจองหน่วยความจำของตัวอื่นๆ ต่อไป โดยส่งผลกลับมาเป็น "1" ให้ตรวจสอบตัวโปรแกรมที่ใช้ในการจองหน่วยความจำ HIMEM และ Conventional เขียนขึ้นได้ดังนี้

```
int allocterm1(struct TERM1 *term1)
{
long size;
if((term1->term1 = (struct record far *)farmalloc(256*sizeof(struct record))) ==
NULL)
        return 0;
/* alloc buffer in conventional memory 256*sizeof(record) byte */
size = (long)((term1->TermNum*sizeof(struct record))/1000) + 1;
/* uppper line convert size to KB */
term1->handle= XMSalloc(size);
      if(XMS_ERR != 0x00) {
      printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        return 0;
     }
      if(!(term1->handle))
     {
      farfree(term1->term1);
      return 0;
     }
      term1->oldpage=0;
      return 1;
     }
```

ฟังก์ชัน term1->term1 = (struct record far *)farmalloc(256*sizeof(struct record))) == NULL)

ฟังก์ชันนี้ จะทำการคำนวณขนาดของ Conventional ที่เราต้องการ เมื่อคำนวณได้แล้วก็จะทำการจอง ถ้าจองได้ก็จะทำให้คำสั่งตรวจสอบค่านี้ไม่ทำงาน เกิดการจองไม่ประสบผลสำเร็จ จะทำให้ค่าที่ได้มีค่าเท่ากับ NULL ทำให้คำสั่ง if นี้ทำงานและส่งผลกลับไปเป็นศูนย์ เพื่อให้โปรแกรมคืนค่าหน่วยความจำที่จองไว้ก่อนหน้านี้แล้ว หลังจากนั้น ค่อยออกจากโปรแกรมแต่ถ้าเกิดการจองได้ ก็จะทำการคำนวณในส่วนของหน่วยความจำ XMS โดยใช้คำสั่ง size = (long)((term1->TermNum*sizeof(struct record))/1000)+1;

ตัวอย่างการคำนวณ เช่น

บน Conventional

256*4 = 1024 byte

บน XMS

$$\frac{470000*4}{1000} + 1 = 1880 \text{ Kbyte}$$

จะสังเกตุได้ว่าหน่วยความจำที่จองบน XMS จะมากกว่าบน Conventional

หลังจากนั้น จะทำการถ่ายค่าตัวชี้โดยใช้คำสั่ง term1->handle = XMSalloc (size); ถ้าเกิดกรณีที่มีข้อผิดพลาดในการจองหน่วยความจำโปรแกรมจะมาทำคำสั่ง

```
if (XMS_ERR != 0x00) {
        printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        return 0;
        }
if(!(term1->handle))
        {
        farfree(term1->term1);
        return 0;
        }
```

ถ้าเกิดจองหน่วยความจำได้ ให้ค่า page ที่ใช้อยู่มีค่าเท่ากับศูนย์แล้ว return ค่ากลับไปเป็น 1 เพื่อบอกว่าจองหน่วยความจำได้แล้ว

```
term1->oldpage=0;
return 1;
```

ในทำนองเดียวกัน การจองหน่วยความจำใน term อื่นๆ ก็จะมีหลักการทำงานที่คล้ายกันนี้ เพียงแต่จำนวน byte ที่จองอาจจะแตกต่างกันไปตาม record หลังจากการจองหน่วยความจำใน XMS และ Conventional เสร็จเรียบร้อยแล้วก็ถึงขั้นตอนการนำเอาหน่วยความจำที่เราจองไว้นั้น มาใช้งานโดยทำการเขียนเป็นโปรแกรมได้ดังนี้

```
struct record4 far *Term4(struct TERM4 *term4,int index)
{
  register int page=0;
  register long XMSoffset;
  /* check page fault */
  XMSoffset = (long)(index/256LU);
  page = (int)XMSoffset;
  if(page != term4->oldpage)
  {
      XMSwriteoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
          ((long)term4->oldpage*256*sizeof(struct record4)));
      XMSreadoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
          ((long)page*256*sizeof(struct record4)));
      term4->oldpage = page;
  }
return (struct record4 far *)((char far *)term4->term4 + ((index*sizeof(struct record4))
      - (page * 256LU * sizeof(struct record4))));
  }
```

โดยที่ฟังก์ชัน struct record4 far *Term4 นั้นจะทำการรับค่าตัวชี้เข้ามาว่า มันชี้อยู่ที่ Page ไหน ตัวแปร XMS Offset = (long) (index / 256LU); จะสมมุติรับค่า INDEX อยู่ที่ 250

$$250 = \frac{0.976}{256}$$ ทำการปัดเศษทิ้งจะเหลือค่า 0 แสดงว่าตอนนี้ทำงานอยู่ที่ Page 0

แล้วทำการถ่ายค่า XMS Offset = 0 ให้กับตัวแปร Page หลังจากนั้นก็จะไปทำการตรวจสอบค่าโดยใช้คำสั่ง if ดังนี้ if (Page != term4 -> old Page) ซึ่งก็คือ จะทำการตรวจสอบค่า Page กับค่า Oldpage ว่าอยู่ตำแหน่งเดียวกันหรือไม่ ถ้าอยู่ตำแหน่งเดียวกัน ก็จะห้าม คำสั่ง if ตัวนี้ไปสมมุติว่า Page กับ Old page มีค่าไม่เท่ากัน แสดงว่า ตอนนี้ ต้องการย้าย Oldpage ไปเก็บไว้บน XMS และดึง Page ใหม่ที่ต้องการลงมาที่ Conventional โดยเขียนเป็นโปรแกรมได้ดังนี้

```
if(page != term4->oldpage)
{
    XMSwriteoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
            ((long)term4->oldpage*256*sizeof(struct record4)));
    XMSreadoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
            ((long)page*256*sizeof(struct record4)));
    term4->oldpage = page;
}
```

โปรแกรมนี้จะเป็นคำสั่งย้ายข้อมูลใน Conventional ที่เราไม่ต้องการขึ้นไปเก็บไว้บนหน่วยความจำ XMS โดยที่ในหน่วยความจำ Conventional นั้นค่า Term4 -> handle จะเป็นตำแหน่งเริ่มต้นในหน่วยความจำ XMS ต่อด้วยจำนวนไบต์ที่จะเขียน 256 x 4 = 1024 Byte ต่อด้วยตำแหน่งที่จะเขียนใน XMS คำนวนได้จาก เพจที่ใช้งาน คูณด้วย ขนาดของ Record เช่น ตอนนี้อยู่ที่ Page 0 = 0 x 256 x 4 = 0 คือตำแหน่งที่จะไปเขียนอยู่ที่ Page 0

ต่อไปจะเป็น XMS read off ป้อนค่าเข้ามาเหมือนกันคือ เราทำการย้ายหน่วยความจำจาก Conventional ไปไว้บน XMS แล้วแสดงว่าตอนนี้ หน่วยความจำที่ Conventional ว่างอยู่ เราก็จะทำการย้ายข้อมูลจาก XMS ลงมาไว้บน Conventional โดยป้อนค่าได้ดังนี้ ป้อนตำแหน่งเริ่มต้นใน Conventional ป้อนตำแหน่งเริ่มต้นบน XMS หาขนาดจำนวน Byte ที่จะทำการย้าย โดยค่าสุดท้ายจะเป็นตำแหน่งของ Page ที่จะทำการย้ายลงมา เช่น ต้องการย้าย Page 1 ลงมา = 1 x 256 x 4 =1024 Byte

หลังจากย้ายเรียบร้อยแล้วให้ทำการถ่ายค่า เพจใหม่ลงบนค่าปัจจุบันที่ชี้อยู่ โดยใช้คำสั่ง

```
Term -> Oldpage = Page ;
```

หลังจากนั้นให้ทำการ Return ค่าตำแหน่ง Block ว่าเราต้องการไปชี้อยู่ที่ Block ไหน โดยใช้คำสั่ง

```
return    (struct record4 far *)((char far *)term4->term4 + ((index*sizeof(struct record4))
          -(page * 256LU * sizeof(struct record4))));
```

จากโปรแกรมเราจะคำนวน Block ได้จาก

ตำแหน่งเริ่มต้นใน Conventional + (ตำแหน่งที่ต้องการจะไป มีหน่วยเป็น Block x ขนาดของ Record) - (Page ที่อยู่ก่อนหน้านี้ x 256 x ขนาดของ Record) เช่น เราต้องการไปที่ตำแหน่ง 257 Block โดยที่ Record

มีขนาด = 4 และ Page ก่อนหน้านี้คือ 1 จะได้ 0 + (257 x 4) - (1 x 256 x 4) = 4 Byte คือ เราต้องเลื่อนตัวชี้ไป 4 Byte หรือ 1 Black นั่นเอง

ส่วนขั้นตอนในการย้ายค่าใน Term อื่นๆ ก็ทำในทำนองเดียวกันนี้ หลังจากทำการย้ายค่าที่คำนวนเสร็จเรียบร้อยแล้ว เราจะต้องทำการคืนหน่วยความจำให้กับระบบ เพราะเราทำการจองหน่วยความจำแบบไดนามิก ถ้าเราจองแบบสแตติกเหมือนโปรแกรมตัวเก่า ตัวคอมพิวเตอร์จะทำการคืนหน่วยความจำเอง แต่โปรแกรมที่เขียนขึ้นมาใหม่ เราต้องสั่งให้โปรแกรมคืนหน่วยความจำให้แก่ระบบ โดยทำการคืนหน่วยความจำบน Conventional และ XMS ให้กับระบบเพื่อไปใช้งานให้กับโปรแกรมอื่นๆต่อไป ดังนี้

```
void freeterm4(struct TERM4 *term4)
{
  farfree(term4->term4);
  XMSfree(term4->handle);
  if(XMS_ERR != 0x00)
  printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
}
```

เมื่อทำการคืนเสร็จแล้ว ต้องทำการตรวจสอบว่า ทำการคืนหน่วยความจำเรียบร้อยแล้วหรือยัง โดยใช้คำสั่ง if คือถ้าเกิด XMS_ERR มีค่าไม่เป็นศูนย์ แสดงว่ายังมีหน่วยความจำที่ยังไม่ได้คืน แสดงว่าหน่วยความจำตรงนั้นอาจเสีย หรือเกิดข้อผิดพลาดใดๆขึ้น ซึ่งมันก็จะแสดงตำแหน่งที่ยังไม่ได้คืนออกมาทางจอภาพ แต่ถ้าเราทำการคืนหน่วยความจำเรียบร้อยแล้ว ก็จะผ่านคำสั่ง if ไปสู่การจบการทำงาน

บทที่ 5
การติดตั้งและใช้งาน

5.1 การติดตั้ง

5.1.1 การติดตั้งฮาร์ดแวร์

เพียงนำส่วนที่เป็นการ์ด (interface card) เสียบลงบน ISA slot ที่ว่างๆในเครื่องคอมพิวเตอร์ แล้วต่อสายจาก connector ที่การ์ดมายังบอร์ดภายนอก ที่ช่อง card connect โดยไฟเลี้ยงของบอร์ดภายนอกนี้ จะใช้อะแดปเตอร์ 13.8 Vdc 1 A เสียบเข้าที่แจ็คบนกล่องถ้าการติดตั้งเป็นไปอย่างถูกต้อง LED Power บนกล่องวงจรจะติดสว่างขึ้น

5.1.2 การติดตั้งซอฟท์แวร์

จะใช้แผ่น Install ในการติดตั้ง ซึ่งแผ่น Install จะมีไฟล์ Logicnew.exe อยู่ (Logicnew.exe นี้จะเป็นไฟล์ที่เก็บโปรแกรมในการใช้งานทั้งหมดด้วยการบีบอัดข้อมูล) ให้ใช้ไฟล์ Install.bat ในการติดตั้งโดยการพิมพ์ Install <drive : > ,drive คือ ชื่อไดร์ฟที่ต้องการจะนำโปรแกรมไปติดตั้ง เช่น ต้องการติดตั้งที่ไดร์ฟ C ก็พิมพ์ Install C: ก็จะเป็นการติดตั้งโปรแกรมทั้งหมดลงในไดร์ฟ C ในไดเรคทอรี่ของ "LOGICNEW" ซึ่งเมื่อทำการติดตั้งเรียบร้อยแล้ว จะมีไฟล์โปรแกรมที่สำคัญในไดเรคทอรี่ "LOGICNEW" ดังนี้

- MAIN.EXE
- CLA.EXE
- PULSE.EXE
- TTL.EXE
- PLA.EXE
- QM.EXE
- READ.EXE
- FON.T

* (ในการติดตั้งนั้นจะต้องมีพื้นที่ว่างในฮาร์ดดิสก์ประมาณ 1.2 MB สำหรับไฟล์ทั้งหมด และต้องการอีกประมาณ 100 KB สำหรับไฟล์ข้อมูลที่ถูกเขียนขึ้นเมื่อเรียกใช้โปรแกรม)

5.2 การใช้งาน

5.2.1 การใช้งาน Main Menu

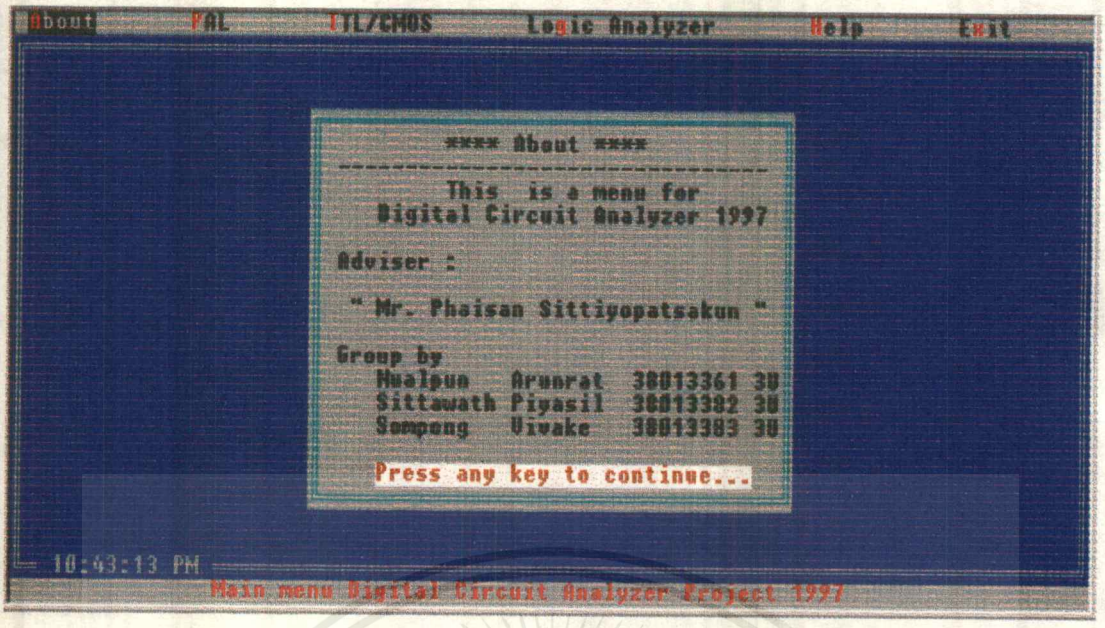
Main Menu เป็นโปรแกรมที่ใช้สั่งงานโปรแกรมทั้งหมดในโปรเจคนี้ นั่นคือ การวิเคราะห์หาสมการจากไอซีโปรแกรม PAL, Logic Analyzer และ การตรวจสอบไอซี TTL/CMOS ซึ่งโปรแกรมนี้คือ MAIN.EXE เมื่อเรียก MAIN ที่ DOS prompt ก็จะปรากฏเมนูขึ้นมาดังรูปที่ 5.2.1

รูปที่ 5.2.1 หน้าจอของ MAIN เมื่อเรียกเข้ามาครั้งแรก

จากรูปที่ 5.2.1 จะเห็นว่าเมนูหลักจะประกอบไปด้วยเมนูย่อย 6 เมนู โดยเมนูย่อยที่ Active จะมีแถบบาร์สีดำอยู่ที่เมนูย่อยนั้น ในการเลือกเมนูย่อย จะทำได้ 2 วิธีด้วยกันคือ วิธีแรกใช้ปุ่มลูกศร ซ้าย และ ขวา เพื่อเลื่อนบาร์แล้วกดปุ่มลง หรือ ENTER เพื่อเปิดเมนูย่อยนั้น หลังจากนั้นก็ใช้ปุ่ม ขึ้น , ลง หรือ SPACE BAR เพื่อเลือกทำงานตามเมนูย่อยนั้นๆแล้วกด ENTER อีกวิธีหนึ่งก็คือใช้คีย์ลัดในการเรียกเมนูย่อย โดยคีย์ลัดจะสังเกตุได้คือ มันจะมีสีสันแตกต่างจากตัวอื่นๆ เช่น เมนู About คีย์ลัดก็คือ ตัวอักษร A คือเมื่อเรากดปุ่มตัวอักษรที่เป็นคีย์ลัด บาร์ก็จะกระโดดไปยังเมนูย่อยนั้นๆทันที และเปิดเมนูให้ โดยที่เราไม่ต้องกดปุ่ม ENTER หรือปุ่มลงอีก หลังจากนั้น ในเมนูย่อยๆนั้นๆ ก็จะมีคีย์ลัดอีก เราก็สามารถที่จะกดคีย์ลัดเพื่อทำงานตามเมนูย่อยได้ทันทีโดยไม่ต้องกดปุ่ม ENTER เช่นเดียวกัน

เมนูย่อยทั้งหมดของเมนูหลักมีดังนี้

1. About : (คีย์ลัดคือ A) เป็นเมนูที่ใช้แสดงข้อมูลเกี่ยวกับโปรแกรม และคณะผู้จัดทำ แสดงดังรูปที่ 5.2.2



รูปที่ 5.2.2 หน้าจอของเมนูหลักเมื่อเลือกเมนูย่อย About

2. PAL : (คีย์ลัดคือ P) เมื่อเปิดเมนูจะมีเมนูย่อยดังรูปที่ 5.2.3

รูปที่ 5.2.3 เมนูของ PAL

- ***Analysis PAL*** (คีย์ลัดคือ N) ทำหน้าที่เรียกโปรแกรมที่ใช้อ่านและวิเคราะห์หาสมการของไอซีตระกูล PAL (PLA.EXE, QM.EXE และ READ.EXE)
- ***Read last equation*** (คีย์ลัดคือ R) ทำหน้าที่เรียกโปรแกรมที่ใช้อ่านสมการที่ได้จากการวิเคราะห์ไอซี PAL ครั้งล่าสุด (READ.EXE)

3. TTL/CMOS : (คีย์ลัดคือ T) แสดงดังรูปที่ 5.2.4 ใช้ในการเรียกโปรแกรมตรวจสอบไอซี TTL/CMOS (TTL.EXE)

รูปที่ 5.2.4 เมนูของ TTL/CMOS

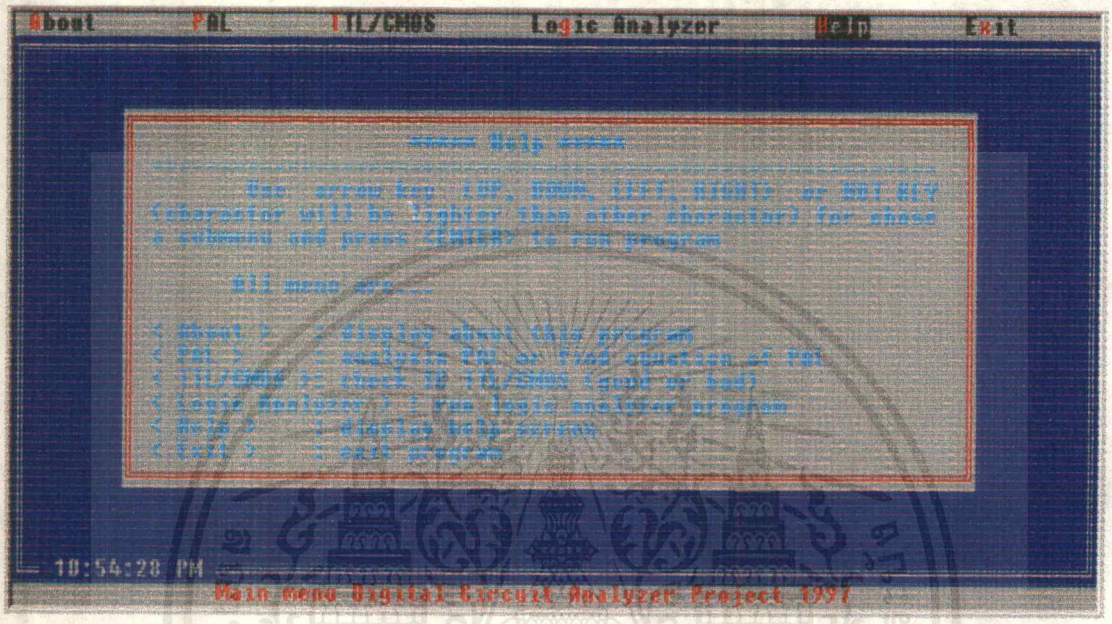
4. Logic Analyzer : (คีย์ลัดคือ G) เมื่อเปิดเมนูขึ้นมา จะมีเมนูย่อยดังนี้ (ดูรูป 5.2.5)

รูปที่ 5.2.5 เมนูของ Logic Analyzer

- ***Run logic analyzer*** (คีย์ลัดคือ R) ทำหน้าที่เรียกโปรแกรม logic analyzer (CLA.EXE และ PULSE.EXE)

- ***Display last data*** (คีย์ลัดคือ D) ใช้ในการดูข้อมูลโลจิกที่วัดได้ครั้งล่าสุดมาดูซ้ำ (PULSE.EXE)

5. Help : (คีย์ลัดคือ H) เป็นเมนูแสดงความช่วยเหลือขณะทำการใช้โปรแกรม Main Menu แสดงดังรูป 5.2.6



รูปที่ 5.2.6 หน้าจอของเมนูย่อย Help

6. Exit : (คีย์ลัดคือ X) ใช้ออกจากเมนูหลัก แสดงดังรูป 5.2.7 ซึ่งสามารถออกจากเมนูหลักได้ 2 วิธี คือ

รูปที่ 5.2.7 หน้าจอเมื่อเปิดเมนู Exit

- ***OS Shell*** (คีย์ลัดคือ O) เป็นการออกจากเมนูแบบชั่วคราว คือ ออกไปยัง DOS ชั่วคราวเท่านั้น ซึ่งสามารถกลับมายังเมนูได้โดยการใช้คำสั่ง EXIT ที่ DOS prompt

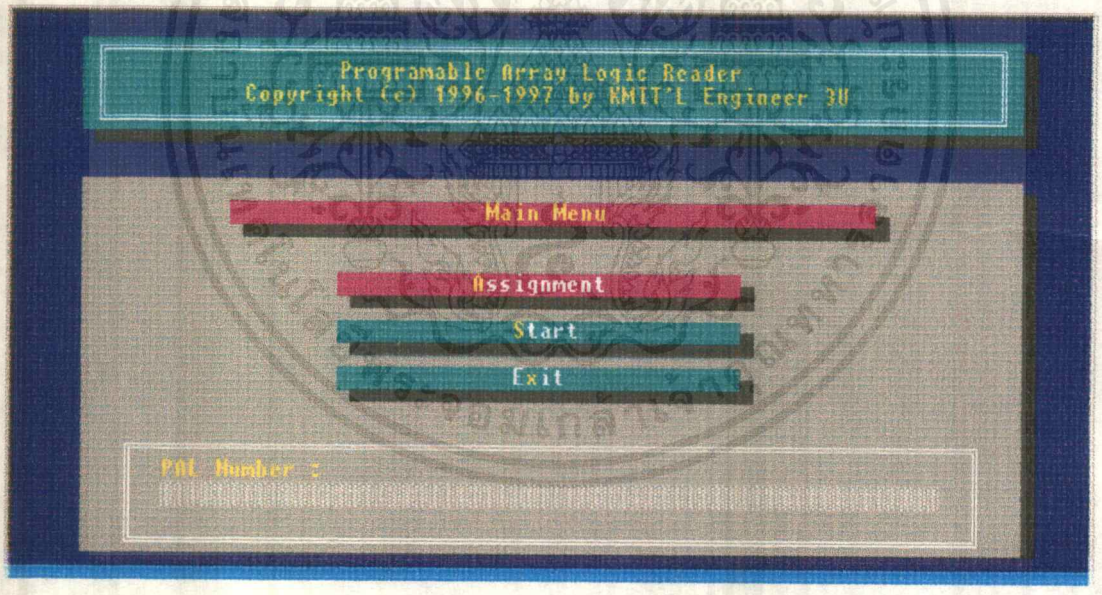
- ***Quit*** (คีย์ลัดคือ Q) เป็นการออกจากเมนู หรือกลับมายัง DOS อย่างถาวร

5.2.2 การวิเคราะห์โปรแกรมจากไอซี PAL

การทำงานนี้ ก็แยกโปรแกรมออกเป็น 3 ตัวคือ ส่วนอ่านข้อมูลจากไอซี PAL , ส่วนวิเคราะห์สมการ และโปรแกรมอ่านไฟล์สมการ ซึ่งมีการใช้งานดังต่อไปนี้

5.2.2.1 โปรแกรม PLA.EXE

โปรแกรม PLA.EXE เป็นโปรแกรมที่ใช้สำหรับการอ่าน Function ในไอซี PAL ที่อยู่ในตระกูล H, L และ C ส่วน PAL ในตระกูล R ไม่สามารถอ่านได้ ฟังก์ชั่นที่อ่านได้จะถูกเก็บไว้ในไฟล์ข้อมูล ดังนั้น ไดร์ฟที่ใช้งานโปรแกรมนี้อยู่ จะต้องมีพื้นที่ว่างไม่น้อยกว่า 100 KB การเรียกใช้งานโปรแกรม PLA ทำได้ 2 วิธีคือ เรียกผ่านเมนูหลัก ดังที่ได้กล่าวมาแล้ว และเรียกโดยตรงโดยการพิมพ์ PLA ที่ DOS Prompt ถ้าเครื่องคอมพิวเตอร์ที่ใช้งานอยู่ไม่ได้ต่อฮาร์ดแวร์ของเครื่องวิเคราะห์วงจรดิจิตอลไว้ จะมีข้อความเตือนออกมา ถ้าต่อเครื่องไว้เรียบร้อยแล้ว LED Stand By จะติดสว่าง และจะเห็นเมนูดังในรูปที่ 5.2.8



รูปที่ 5.2.8 เมนูของโปรแกรม PLA.EXE เมื่อเรียกเข้ามาครั้งแรก

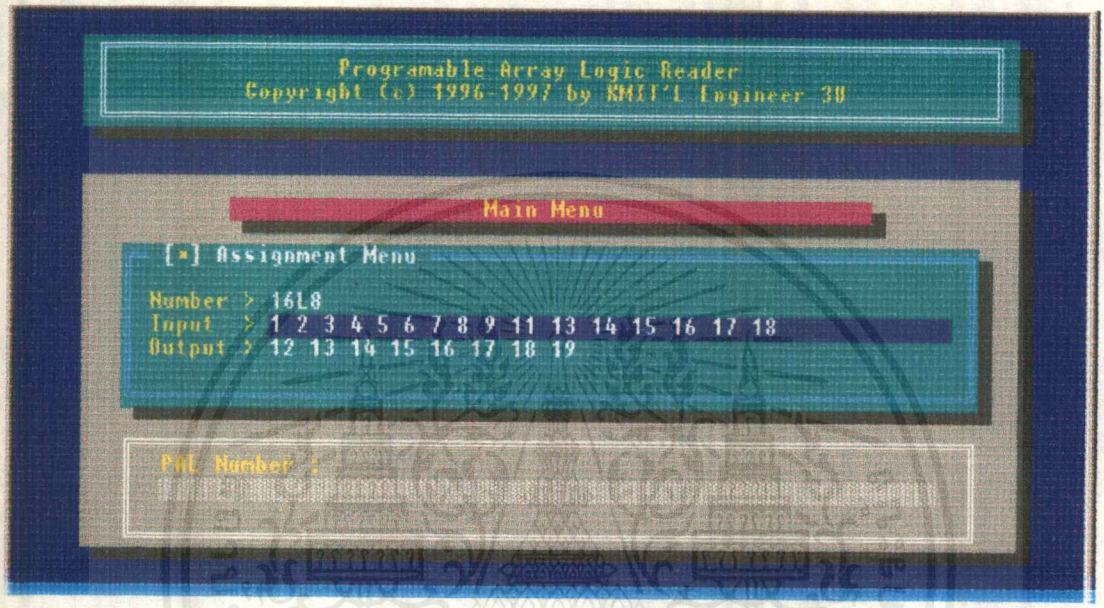
โดยในเมนูจะประกอบไปด้วย 3 เมนูย่อยคือ เมนู Assignment , Start และ Exit ส่วนด้านล่างของเมนูจะแสดงข้อความ

PAL Number :

ซึ่งในตอนเรียกเข้าโปรแกรมครั้งแรกนี้จะไม่มีค่าใดๆ ปรากฏขึ้นหลังข้อความดังกล่าว ส่วนแถบบาร์ด้านล่างจะเป็นตัวแสดงการอ่านข้อมูลจาก PAL

การใช้งานแต่ละเมนูย่อย

1. Assignment เป็นเมนูที่ใช้สำหรับกำหนดเบอร์ของ PAL ตำแหน่งขา อินพุท และเอาท์พุท สามารถเรียกใช้โดยการเลื่อนบาร์มาที่ Assignment แล้วกด ENTER หรือกดคีย์ A จะปรากฏผลดังแสดงในรูปที่ 5.2.9



รูปที่ 5.2.9 เมนู Assignment ของ PLA.EXE

การกำหนดเบอร์ไอซี PAL เราสามารถเลือกเบอร์ของ PAL ได้โดยพิมพ์เบอร์ของ PAL ตามหลังข้อความ

Number >

Input >

Output >

* โดยเบอร์ไอซีที่สามารถเลือกได้ คือ เบอร์ 10H8, 12H6, 14H4, 16H2, 10L8, 12L6, 14L4, 16L2, 12L10, 14L8, 16L6 และ 1618

เช่น ต้องการอ่านโปรแกรมจาก PAL เบอร์ 16L8 ให้พิมพ์ชื่อเบอร์ แล้วกด ENTER ดังนี้

Number > 1618 <ENTER>

Input >

Output >

จะปรากฏผลดังนี้

Number > 16L8

Input > 1 2 3 4 5 6 7 8 9 11 13 14 15 16 17 18

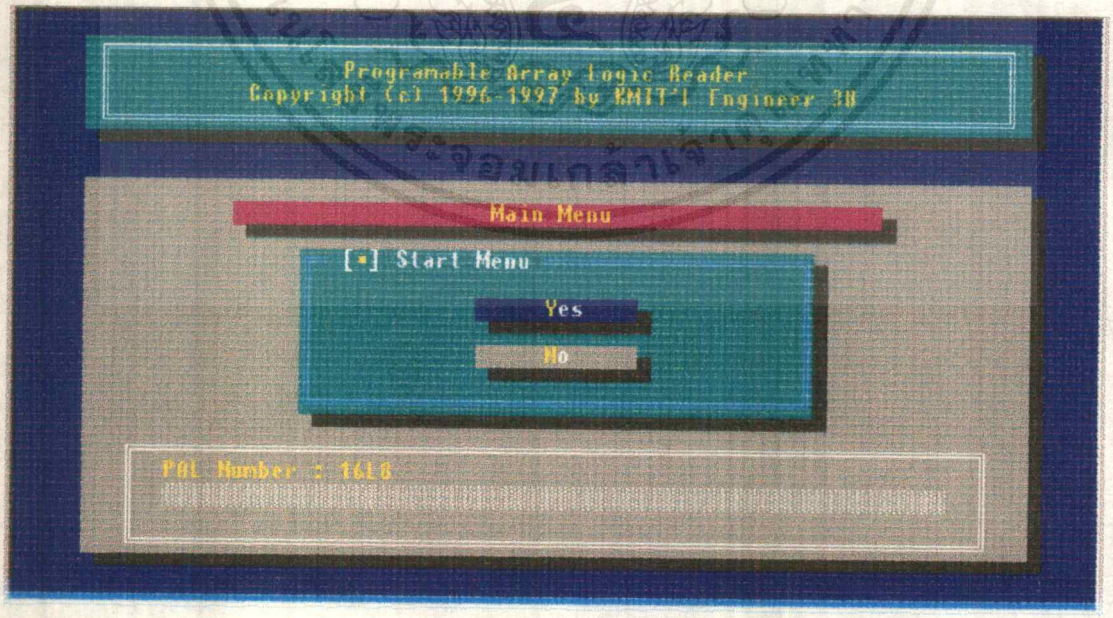
Output > 12 13 14 15 16 17 18 19

ต่อจากนี้ก็จะเป็นการกำหนดขาที่ต้องการให้เป็นอินพุท โดยขาแรกทางซ้ายมือจะเป็นค่า LSB ถ้าไม่ต้องการเปลี่ยนแปลงตำแหน่งขาให้กด ENTER เท่านั้น เพื่อผ่านการเลือกขาอินพุท แต่ถ้าต้องการกำหนดตำแหน่งขาใหม่ก็สามารถพิมพ์ตำแหน่งขาได้เลย โดยใช้ SPACE BAR เป็นตัวขั้นระหว่างอินพุทแต่ละขาโดยไม่จำเป็นต้องเลือกให้ครบทุกขา (ควรเลือกเฉพาะขาที่เป็นอินพุทจริงๆเท่านั้น เพราะจะทำให้ลดเวลาในการวิเคราะห์หาสมการลงไปได้มาก) และไม่ต้องเรียงขาตามลำดับก็ได้ เมื่อเลือกขาอินพุทครบแล้วให้กด ENTER การกำหนดขาจะต้องกำหนดตามค่าที่ปรากฏเท่านั้น ถ้ากำหนดนอกเหนือไปจากนี้จะมีเสียงเตือนให้ป้อนค่าใหม่

เมื่อผ่านขั้นตอนนี้แล้วก็จะเป็นการกำหนดขาเอาท์พุทของไอซี วิธีการป้อนค่าก็จะเหมือนกับการป้อนขาอินพุท โดยที่ค่าแรกจะให้ผลเป็น function ที่ 1 ค่าต่อไปก็จะเป็น function ถัดไปตามลำดับ เมื่อป้อนครบแล้วให้กด ENTER ก็จะกลับมายังเมนูหลักตามเดิม และจะมีข้อมูลปรากฏที่ด้านล่างของเมนูเป็น

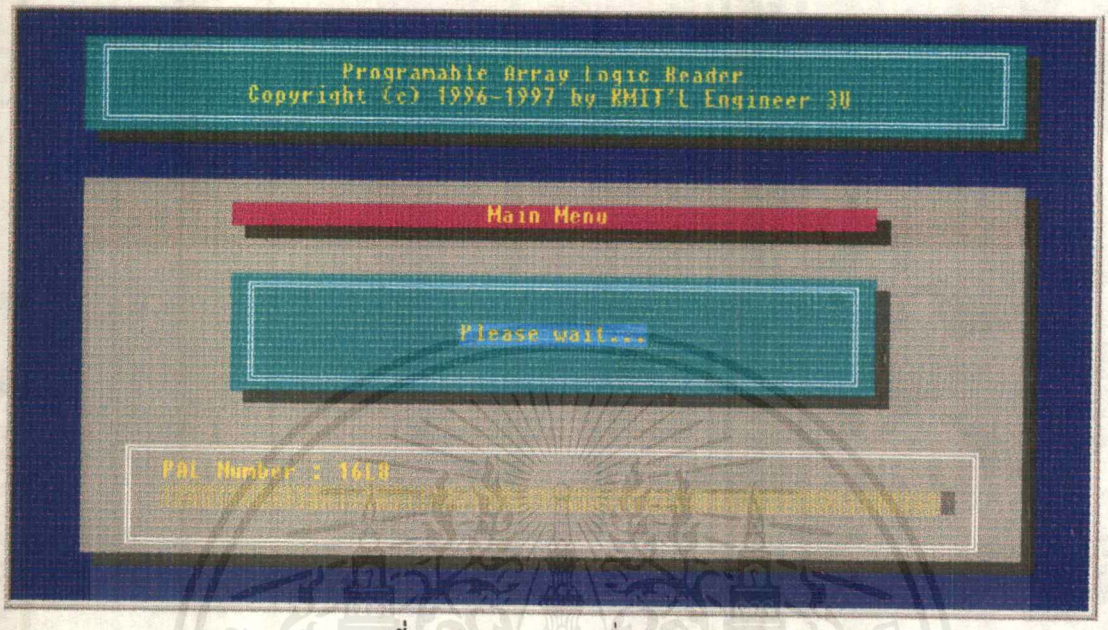
PAL Number : 16L8

2. Start เป็นเมนูสำหรับเริ่มต้นการอ่าน function จาก PAL สามารถเลือกเมนูนี้ได้จากการเลื่อนบาร์ โดยใช้คีย์ลูกศร มายังเมนู Start แล้วกด ENTER หรือการกดคีย์ S จะปรากฏผลดังรูปที่ 5.2.10



รูปที่ 5.2.10 เมนู Start ของ PLA.EXE

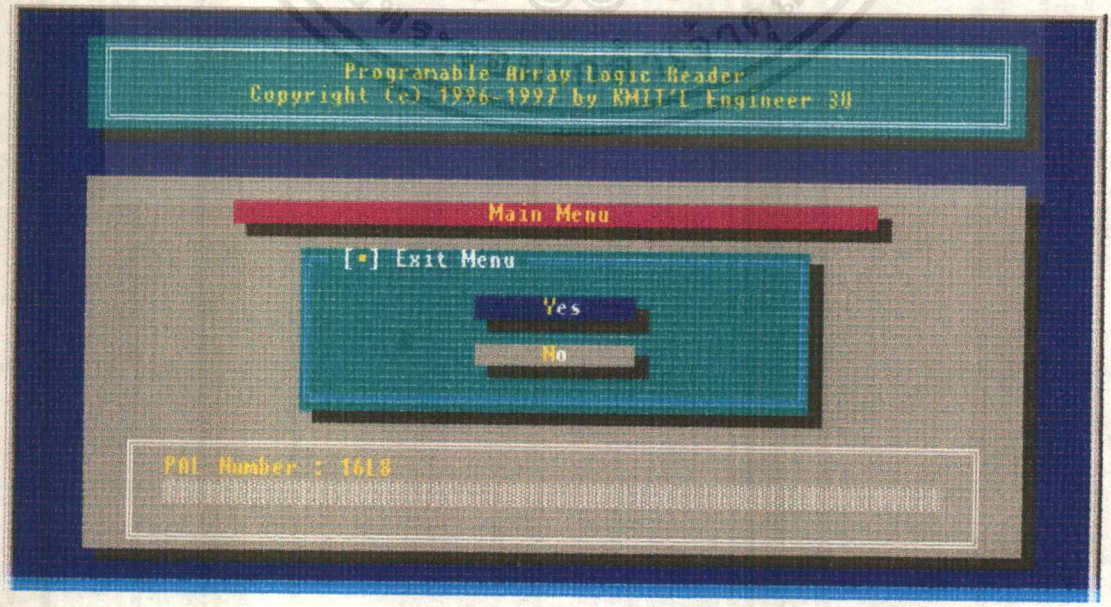
ซึ่งจะมีการถามอีกครั้งเพื่อยืนยัน ถ้าต้องการให้อ่าน ให้เลือก Yes แล้วกด ENTER หรือกดคีย์ Y แต่ถ้าไม่ต้องการให้เลือก No หรือกดคีย์ N ซึ่งเราจะไม่สามารถ Start ใหม่ได้ถ้าไม่ได้ทำการเลือกเบอร์ไอซีก่อน แต่ถ้าเลือกทุกอย่างถูกต้องเรียบร้อยแล้ว จะปรากฏข้อความดังแสดงในรูปที่ 5.2.11



รูปที่ 5.2.11 เมนู Start เมื่อเลือก Yes

หลังจากนั้น ที่ด้านล่างของเมนู จะแสดงให้เห็นว่าอ่านค่าไปแล้วเท่าไร โดยจะแสดงเป็นบาร์สีเหลืองเลื่อนไปทางขวามือ ถ้าเลื่อนไปสุดทางขวามือก็แสดงว่าอ่านข้อมูลครบแล้ว (ในขณะที่อ่านข้อมูลอยู่นี้ LED Busy จะติดสว่าง) ขั้นตอนต่อไปก็จะทำการเขียนข้อมูลลงไฟล์ แล้วจบโปรแกรมเอง

3. *Exit* เป็นเมนูสำหรับออกจากโปรแกรม ซึ่งเราสามารถออกจากโปรแกรมได้โดยการเลื่อนบาร์มาที่ Exit แล้วกด ENTER หรือกดคีย์ X หลังจากนั้นจะปรากฏผลดังรูปที่ 5.2.12



รูปที่ 5.2.12 เมนู Exit ของ PLA.EXE

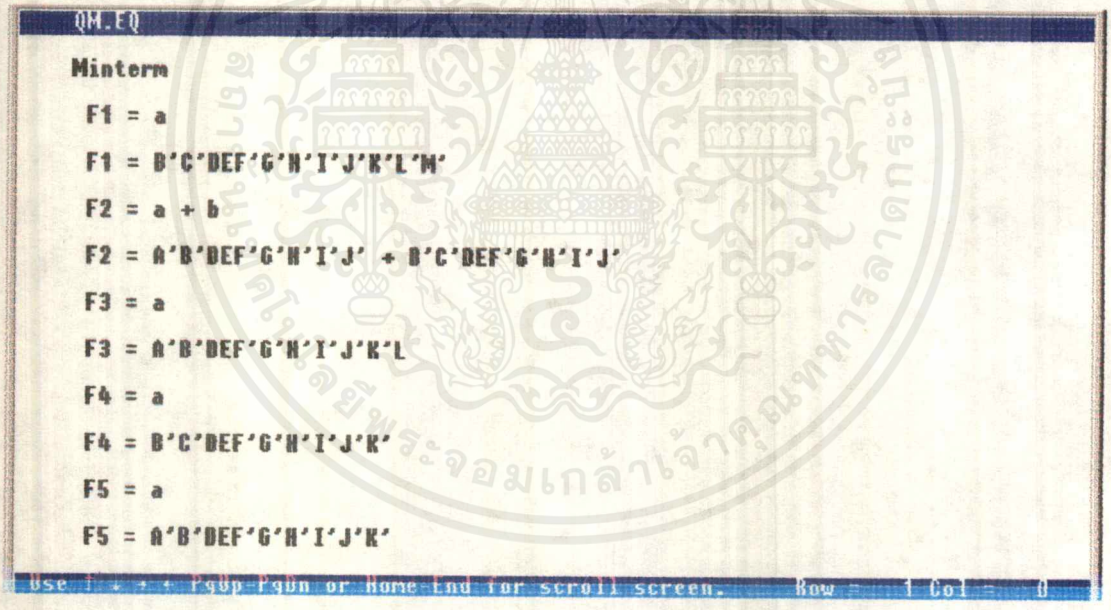
ถ้าต้องการออกจากโปรแกรมให้เลือก Yes แล้วกด ENTER หรือกดคีย์ Y แต่ถ้าไม่ต้องการออกจากโปรแกรมก็ให้เลือก No หรือกดคีย์ N

5.2.2.2 โปรแกรม QM.EXE

ใช้วิเคราะห์หาสมการจาก function ที่อ่านออกมาได้โดย PLA.EXE จะถูกสั่งให้ทำงานโดยเมนูหลัก Analysis PAL ต่อจากโปรแกรม PLA.EXE แต่ถ้าหากจะเรียกใช้ที่ DOS prompt ก็ทำได้โดยการเรียก QM แต่ต้องแน่ใจว่ามีไฟล์ FUNCTXX.DAT ครบ และถูกต้องตามที่กำหนดในไฟล์ QM.CFG ซึ่งผลก็คือจะได้สมการออกมาในไฟล์ QM.EQ

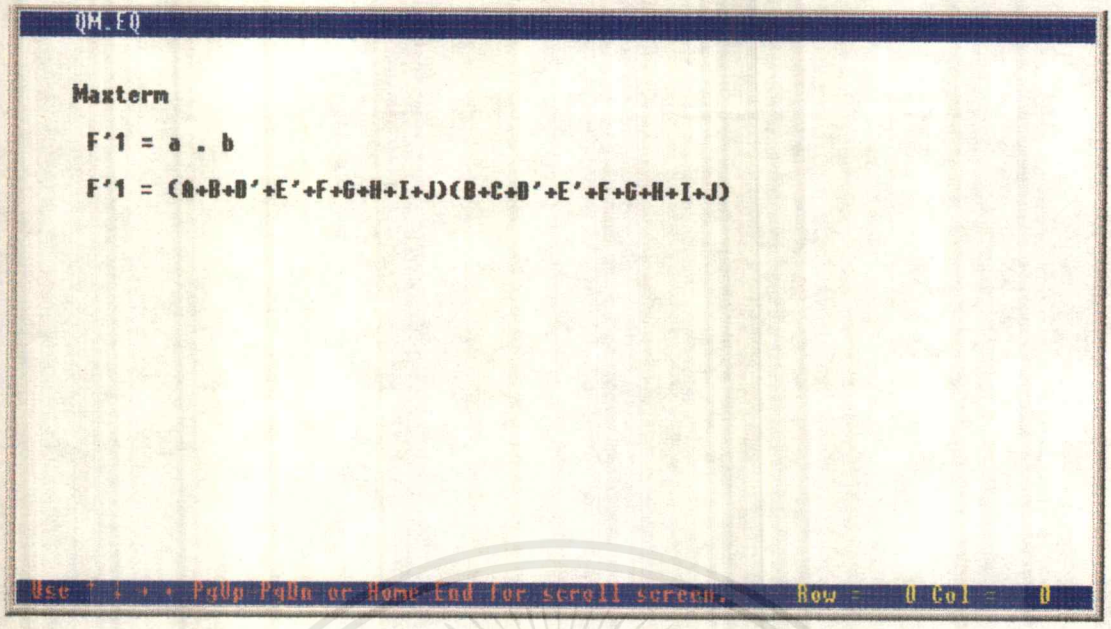
5.2.2.3 โปรแกรม READ.EXE

ใช้อ่านไฟล์สมการ (QM.EQ) แล้วนำมาแสดงผลให้ดูได้โดยง่าย ซึ่งจะถูกเรียกผ่านเมนูหลักของ PLA แต่ถ้าเรียกใช้ที่ DOS prompt ก็พิมพ์ READ QM.EQ จะได้หน้าจอดังรูปที่ 5.2.13 และทำการเลื่อนดูได้โดยใช้คีย์ลูกศร ขึ้น , ลง , ซ้าย , ขวา หรือ Page Up , Page Down , Home หรือ End และออกจากโปรแกรมได้ด้วยการกดคีย์ ESC



รูปที่ 5.2.13 หน้าจอของโปรแกรม READ.EXE ในขณะที่อ่านไฟล์ QM.EQ

ในการอ่านไฟล์ QM.EQ นั้น ถ้าเรากำหนดให้อ่านคราวละหลายๆ เอาท์พุท สมการที่อ่านได้จะอยู่ในรูปแบบของ Minterm เท่านั้น แต่ถ้าเรากำหนดให้อ่านไฟล์ QM.EQ คราวละ 1 เอาท์พุท สมการที่อ่านได้จะอยู่ในรูปแบบของ Minterm หรือ Maxterm อย่างใดอย่างหนึ่งตามลักษณะการโปรแกรมของไอซีตัวนั้นๆ ดังรูปที่ 5.2.14

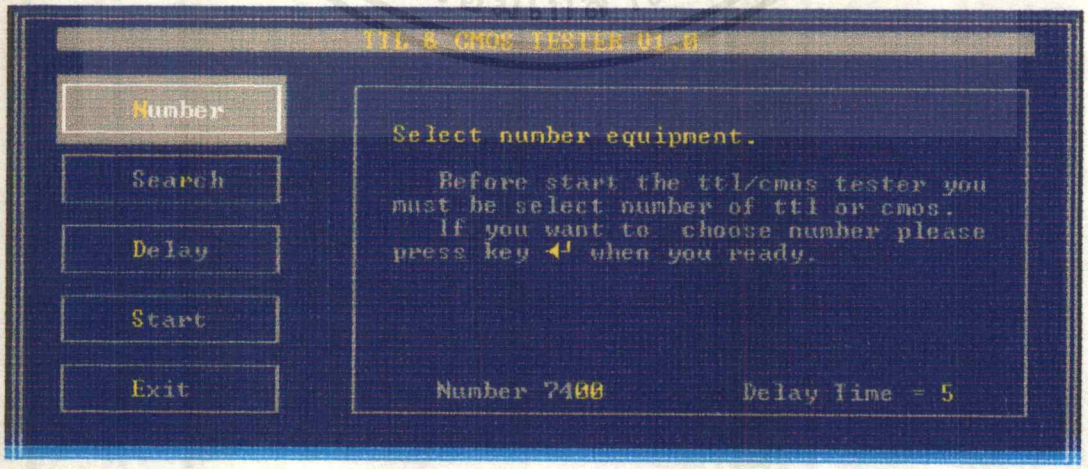


รูปที่ 5.2.14 หน้าจอของโปรแกรม READ.EXE ในขณะที่อ่านไฟล์ QM.EQ ในรูปแบบของ **Maxterm**

* การใช้โปรแกรมเหล่านี้ หากใช้ผ่านเมนูหลัก จะทำให้สะดวกกว่าการเรียกใช้ที่ DOS prompt มาก

5.2.3 การตรวจสอบไอซี TTL / CMOS

การตรวจสอบไอซี TTL/CMOS นี้ จะใช้โปรแกรม TTL.EXE ซึ่งจะตรวจสอบไอซี TTL/CMOS ในตระกูล 74 series ซึ่งเป็นการประยุกต์ใช้ Hardware ให้คุ้มค่าที่สุด โดยไอซีที่นำมาทดสอบ จะต้องเป็นไอซีเกทพื้นฐานทั่วไปที่ไม่ใช่ OC (open collecter) และไม่ใช่ไอซีประเภทฟลิปฟลอปด้วย ซึ่งการทำงานของโปรแกรมจะมีการตรวจสอบ Hardware ก่อนเสมอ หากไม่ได้ต่อ Hardware ไว้ก็จะมีข้อความเตือน แต่ถ้าต่อ Hardware ไว้เรียบร้อยแล้ว LED Stand By จะติดสว่าง และจะปรากฏหน้าจอดังรูปที่ 5.2.15



รูปที่ 5.2.15 หน้าจอของโปรแกรม TTL.EXE เมื่อเรียกเข้ามาครั้งแรก

โดยจะมีเมนูย่อยอีก 5 เมนู ซึ่งประกอบไปด้วย

- Number
- Search
- Delay
- Start
- Exit

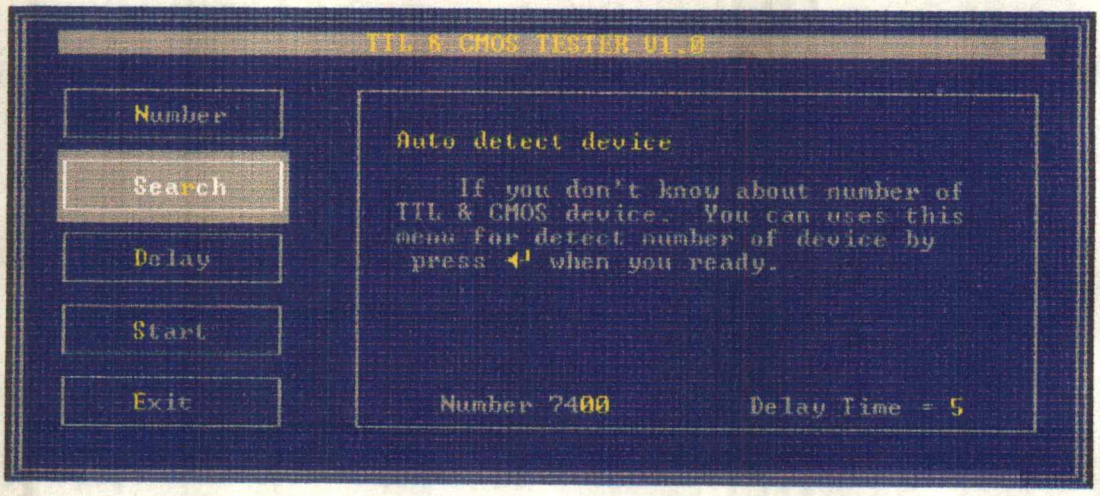
การเลือกใช้เมนูย่อยสามารถเลือกได้จากการใช้คีย์ลูกศร ขึ้น และ ลง เพื่อเลื่อนบาร์ไปยังเมนูย่อยแล้วกด ENTER หรือ เลือกได้โดยการกดตัวอักษรสีเหลืองที่ปรากฏอยู่ตามแต่ละเมนูย่อย

1. เมนู Number เมนูนี้จะใช้สำหรับการป้อนเบอร์ไอซีที่ต้องการจะทดสอบ โดยเมื่อเลือกเมนูนี้ หน้าจอจะแสดงผลดังรูปที่ 5.2.16

รูปที่ 5.2.16 เมนู Number ของ TTL.EXE

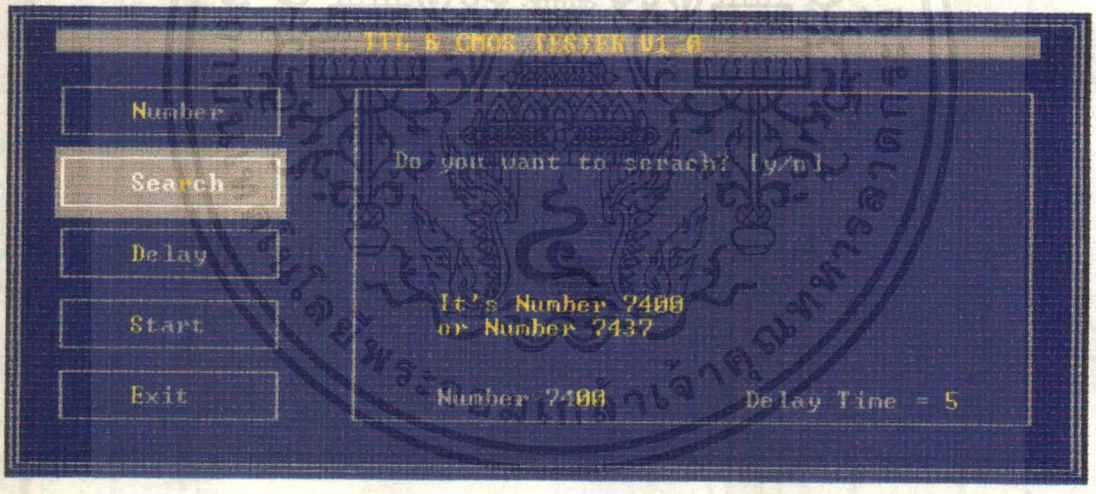
หลังจากนั้น ให้ป้อนเบอร์ไอซีที่ต้องการแล้วกด ENTER ถ้าไอซีเบอร์นั้นไม่สามารถตรวจสอบได้ หรือป้อนเบอร์ผิดพลาด จะมีเสียงเตือน และจะให้ป้อนค่าใหม่ ถ้าถูกต้องก็จะกลับมาที่เมนูหลักตามเดิม

2. เมนู Search เป็นเมนูที่ใช้สำหรับค้นหาเบอร์ไอซีในกรณีที่ไม่ทราบเบอร์ไอซี เนื่องจากการลบเลือนของเบอร์บนตัวไอซี การตรวจสอบจะทำได้ก็ต่อเมื่อ ไอซีที่นำมาตรวจสอบเป็นไอซีที่ดี และเป็นเบอร์ที่โปรแกรมสามารถตรวจสอบได้เท่านั้น เมื่อเลือกใช้เมนูนี้จะมีข้อความถามยืนยันอีกครั้ง ดังแสดงในรูปที่ 5.2.17



รูปที่ 5.2.17 การถามยืนยันเมื่อจะทำการค้นหาเบอร์ไอซี

ถ้าต้องการให้กด Y แต่ถ้าไม่ต้องการให้กด N ถ้าไอซีที่นำมาตรวจสอบดี และโปรแกรมสามารถตรวจสอบได้ โปรแกรมก็จะแสดงการค้นหาข้อมูลที่หน้าจอ เมื่อพบแล้วก็จะมีข้อความบอกให้ผู้ใช้ทราบว่าไอซีที่นำมาตรวจสอบเป็นเบอร์อะไร ดังรูปที่ 5.2.18



รูปที่ 5.2.18 การรายงานเบอร์ของไอซีที่ตรวจสอบได้

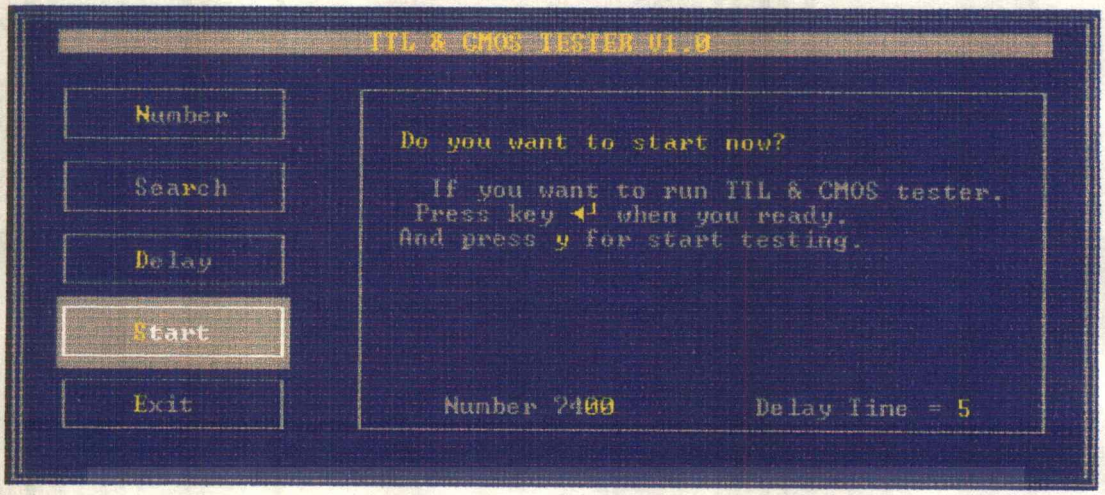
3. เมนู Delay ใช้กำหนดค่าการหน่วงเวลา เมื่อเลือกเมนูนี้โดยใช้คีย์ลูกศร จะปรากฏหน้าจอดังรูปที่ 5.2.19

รูปที่ 5.2.19 เมนู Delay ของ TTL.EXE

ซึ่งในการทดสอบไอซีแต่ละเบอร์จะมีการส่งค่าต่างๆ ไปให้กับอินพุทของไอซี เมื่อส่งค่าเสร็จก็จะมีการหน่วงเวลาไว้ก่อน (เพื่อป้องกันไอซีตอบสนองไม่ทัน เนื่องจากความเร็วของคอมพิวเตอร์ที่ใช้ อาจจะมีมากกว่าที่ไอซีจะทำงานทัน) แล้วจึงรับข้อมูลจากเอาท์พุทของไอซีกลับมาตรวจสอบ ดังแสดงในรูปที่ 5.2.20

รูปที่ 5.2.20 การป้อนค่าหน่วงเวลาในเมนู Delay

4. เมนู Start เมื่อเลือกเมนูนี้โดยการใช้คีย์ลูกศรเลื่อนมาที่เมนูนี้แล้ว จะปรากฏข้อความดังรูปที่ 5.2.21

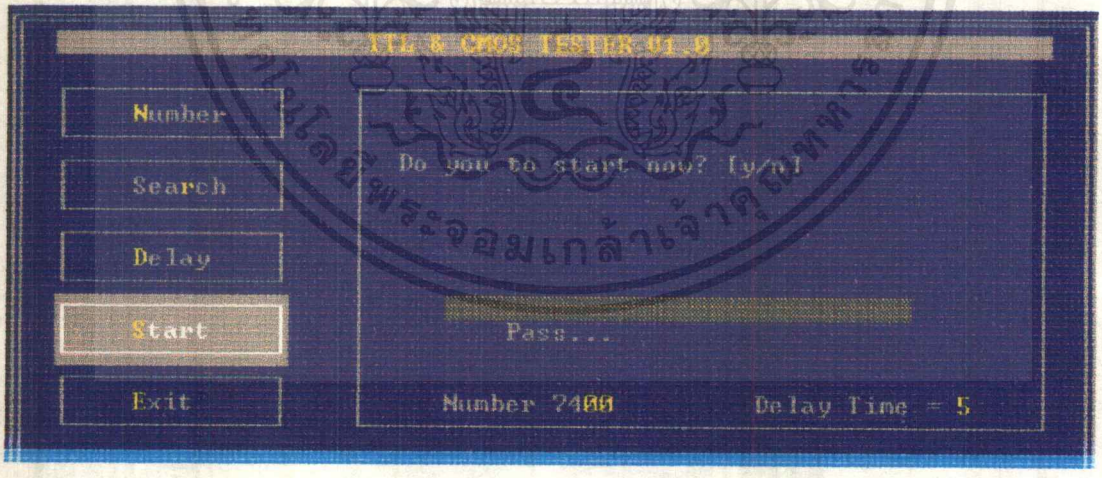


รูปที่ 5.2.21 เมนู Start ของ TTL.EXE

ถ้าเราต้องการตรวจสอบไอซีตามเบอร์ที่ปรากฏอยู่ที่ด้านล่างของเมนู คือ

< Number 7400 Delay Time = 5 >

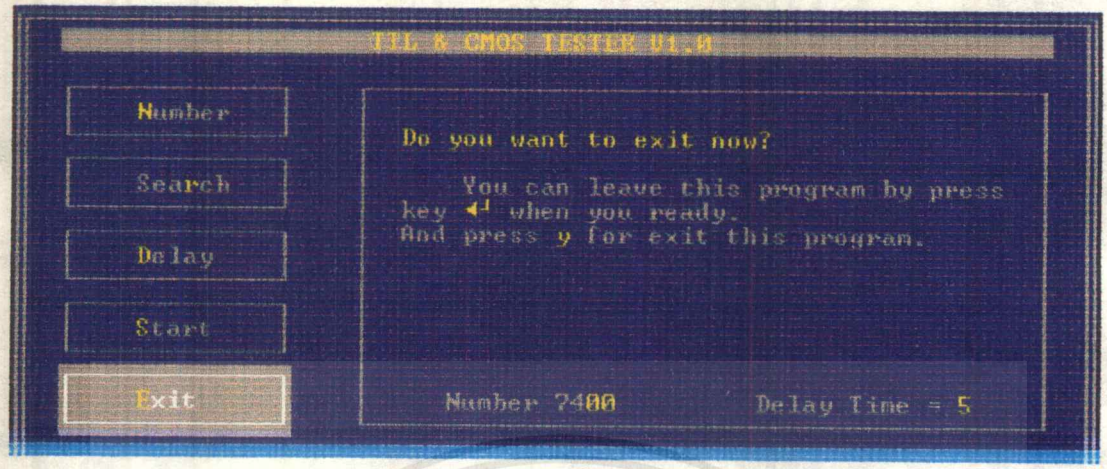
แสดงว่าผู้ใช้เลือกทดสอบไอซีเบอร์ 7400 โดยใช้ค่า Delay Time = 5 ถ้าต้องการตรวจสอบไอซีตามค่านี้ก็ให้กด Y จะปรากฏผลการตรวจสอบแจ้งให้ทราบดังรูปที่ 5.2.22



รูปที่ 5.2.22 การรายงานผลการตรวจสอบไอซี

โดยถ้าปรากฏข้อความ Failed... แสดงว่า ไอซีที่นำมาตรวจสอบนี้เสีย หรือผู้ใช้เลือกเบอร์ไอซีผิดเบอร์นั่นเอง แต่ถ้าปรากฏข้อความ Pass... แสดงว่าไอซีที่นำมาทดสอบนั้นดี

5. เมนู Exit เป็นเมนูสำหรับออกจากโปรแกรม เมื่อเลือกเมนูนี้โดยใช้คีย์ลูกศร จะปรากฏผลดังรูปที่ 5.2.23



รูปที่ 5.2.23 เมนู Exit ของ TTL.EXE

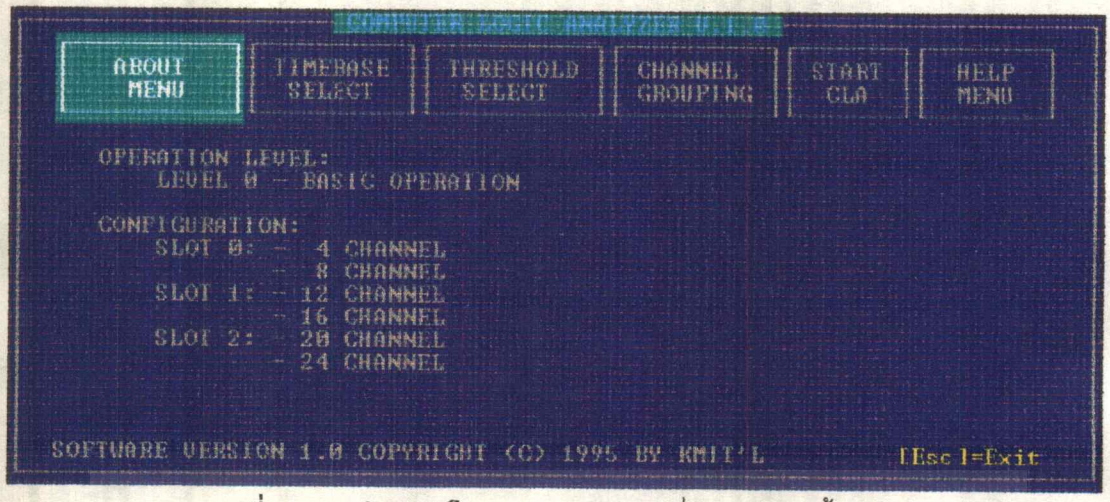
ถ้าต้องการออกจากโปรแกรมให้กด ENTER แล้วกด Y หรือสามารถออกจากโปรแกรมได้โดยการกดคีย์ E และคีย์ Y ตามลำดับ

5.2.4 Logic Analyzer

จากที่ได้กล่าวมาแล้วว่า ส่วนของ Logic Analyzer นั้นจะประกอบไปด้วยโปรแกรมส่วนของการอ่านข้อมูล และโปรแกรมส่วนของการแสดงผล นั้นคือโปรแกรม 2 ตัวดังต่อไปนี้

5.2.4.1 โปรแกรม CLA.EXE

โปรแกรม CLA.EXE เป็นโปรแกรมสำหรับใช้งานเป็น Logic Analyzer ซึ่งในการใช้งานเป็นเครื่อง Logic Analyzer นี้นอกจากผู้ใช้จะต้องติดตั้งส่วนของ Hardware ให้เรียบร้อยแล้ว จะต้องต่อสายวัดของ Logic Analyzer เข้าที่ด้านหลังของเครื่องด้วย โดยสายวัดแต่ละชุดสามารถวัดได้ 8 channel โดยจะต้องเสียบจุดสายวัดเข้ากับ slot 0 เป็น slot แรก ถ้าต้องการวัดมากกว่า 8 channel ก็ให้เสียบสายวัดเพิ่มที่ slot 1 และ slot 2 ตามลำดับ โดยที่สายวัดทั้ง 3 ชุดจะมี 1 ชุดที่แตกต่างจากชุดอื่น สังเกตุจากชุดที่เป็นสายวัดที่มีปากคีบสีเหลืองอยู่ สำหรับสายวัดชุดนี้จะใช้เสียบกับ slot 0 เท่านั้น เนื่องจากปากคีบสีเหลืองนี้จะเป็นตัว trigger สำหรับต่อกับอุปกรณ์ภายนอกเพื่อเป็นตัวบอกให้อุปกรณ์ภายนอกที่เราต้องการวัดสัญญาณเริ่มต้นทำงาน และการใช้งานโปรแกรมนี้จะต้องนำไอซีออกจาก test slot บนกล่อง Hardware ด้วย เพื่อให้การทำงานของโปรแกรมเป็นไปอย่างถูกต้อง เมื่อติดตั้ง Hardware เรียบร้องแล้ว สามารถเรียกใช้โปรแกรม CLA ได้ 2 วิธีคือเรียกผ่าน Main Menu และเรียกโปรแกรมโดยตรง โดยการพิมพ์ CLA ที่ DOS Prompt ถ้าปรากฏข้อความเตือน แสดงว่าการติดตั้ง Hardware ไม่ถูกต้อง เช่น ไม่ได้เสียบปลั๊ก Adapter ฯลฯ แต่ถ้าไม่มีอะไรผิดพลาด LED Stand By ที่เครื่องจะติดสว่าง และหน้าจอจะปรากฏดังแสดงในรูปที่ 5.2.24



รูปที่ 5.2.24 หน้าจอของโปรแกรม CLA.EXE เมื่อเรียกเข้ามาครั้งแรก

สำหรับโปรแกรม CLA นี้จะมีเมนูย่อย 6 เมนู ประกอบด้วย

- About Menu
- Timebase Select
- Threshold Select
- Channel Grouping
- Start CLA
- Help Menu

การเลือกเมนูย่อยสามารถเลือกใช้โดยการกดคีย์ TAP หรือ คีย์ลูกศร ซ้ายและขวา หรือการกดคีย์ตัวอักษรที่ปรากฏเป็นสีที่แตกต่างกันอยู่ในแต่ละเมนูย่อย ก็ได้เช่นกัน ส่วนการออกจากโปรแกรมนั้นจะใช้คีย์ ESC

การใช้งานในแต่ละเมนูย่อย

1. About Menu เป็นเมนูที่บอกให้ผู้ใช้ทราบว่าการวัด 4 Channel ต้องต่อสายวัดที่ Slot 0 หรือถ้าต้องการวัด 12 Channel ก็ต้องต่อสายวัดทั้ง Slot 0 และ Slot 1

2. Timebase Select ใช้ในการเลือกค่า Timebase ในการวัดสัญญาณ ซึ่งจะมีค่าต่างๆให้เลือกใช้ดังแสดงในรูปที่ 5.2.25

รูปที่ 5.2.25 ค่า Timebase ที่มีให้เลือกในเมนู Timebase Select

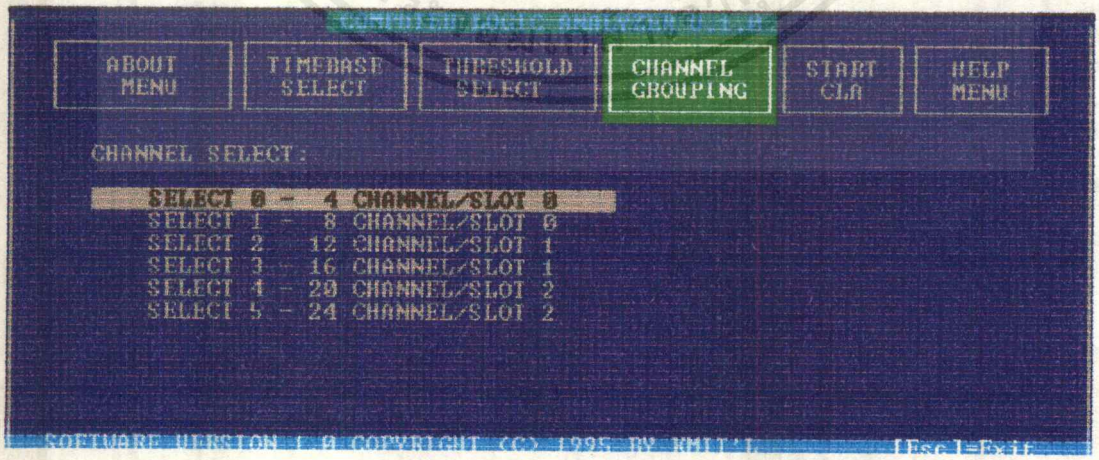
การเลือกค่า Timebase นั้น จะต้องใช้คีย์ลูกศร ขึ้นกับลงเท่านั้น ถ้าใช้คีย์ลูกศร ซ้ายและขวา จะเป็นการเปลี่ยนเมนูย่อย การเลือกค่า Timebase นั้นเพียงแต่เลื่อนบาร์ไปที่ตำแหน่งที่ต้องการ แล้วเปลี่ยนตำแหน่งเมนูย่อยไปยังเมนูอื่นๆได้เลย โดยที่ไม่จำเป็นต้องกดปุ่ม Enter

* การเลือกค่า Timebase ค่ามากๆจะทำให้ไฟล์ข้อมูลมีขนาดใหญ่กว่าการเลือกค่า Timebase น้อยๆ

3. Threshold Select ใช้สำหรับเลือกว่าระดับแรงดันที่เราจะวัดนั้นมีค่าสูงสุดเท่าไร โดยปกติจะตั้งไว้ที่ 5 V สำหรับใช้กับวงจรดิจิตอลทั่วๆไป ซึ่งจะมีค่าต่างๆให้เลือกดังแสดงไว้ในรูปที่ 5.2.26

รูปที่ 5.2.26 ค่า Threshold ต่างๆ ที่มีให้เลือกใช้ในเมนู Threshold Select

4. Channel Grouping เป็นเมนูย่อยสำหรับเลือกว่าต้องการจะวัดกี่ Channel โดยที่จำนวน Channel เราสามารถที่จะกำหนดได้จากค่าที่แสดงในรูปที่ 5.2.27



รูปที่ 5.2.27 การเลือกจำนวน Channel

ในการเลือกจำนวน Channel มากๆ จะทำให้วัดความถี่ได้ต่ำลงกว่าการเลือกจำนวน Channel น้อยๆ

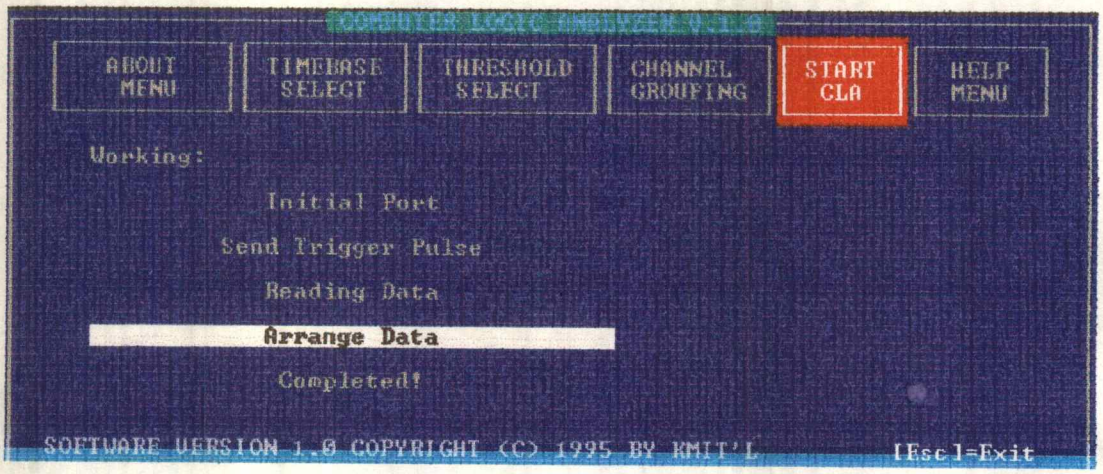
* ข้อแนะนำการใช้งาน ถ้าใช้สายวัดไม่ครบทุก Channel ที่เลือก เช่นเลือกว่าจะใช้ 8 Channel แต่ใช้วัดจริงเพียง 6 Channel , Channel ที่ไม่ได้ใช้งานควรต่อไว้กับโลจิก " 0 " เพื่อให้สามารถอ่านค่า State ได้อย่างถูกต้อง

5. Start CLA เมื่อเลือกมาที่เมนูนี้ โปรแกรมจะบอกให้ผู้ใช้กด ENTER เพื่อเริ่มต้นวัดสัญญาณ แต่ถ้าต้องการเปลี่ยนค่าต่างๆก็สามารถเลื่อนบาร์ไปยังเมนูอื่นๆได้ ที่เมนูนี้จะแสดงค่าต่างๆที่ผู้ใช้เลือกไว้ให้ดูทั้งหมดก่อน เพื่อให้ผู้ใช้ยืนยันการเริ่มวัดสัญญาณโดยใช้ค่าต่างๆเหล่านี้ ดังแสดงในรูปที่ 5.2.28 และระหว่างที่มีการติดต่อส่งและรับข้อมูล LED Busy จะติดสว่าง

รูปที่ 5.2.28 เมนู Start CLA จะแสดงค่าทั้งหมดที่ตั้งไว้ให้ดูก่อนการเริ่มต้นวัดสัญญาณ

หลังจากนั้นโปรแกรมจะเริ่มการทำงานตามลำดับดังนี้

1. ทำการ Initial port รับส่งข้อมูล
2. ส่งสัญญาณ trigger ไปยังอุปกรณ์ภายนอก
3. อ่านข้อมูลมาเก็บไว้ในหน่วยความจำ
4. ทำการเรียงข้อมูลใหม่
5. เขียนข้อมูลลงไฟล์ แล้วจบโปรแกรมซึ่งจะมีการแสดงผลดังในรูปที่ 5.2.29

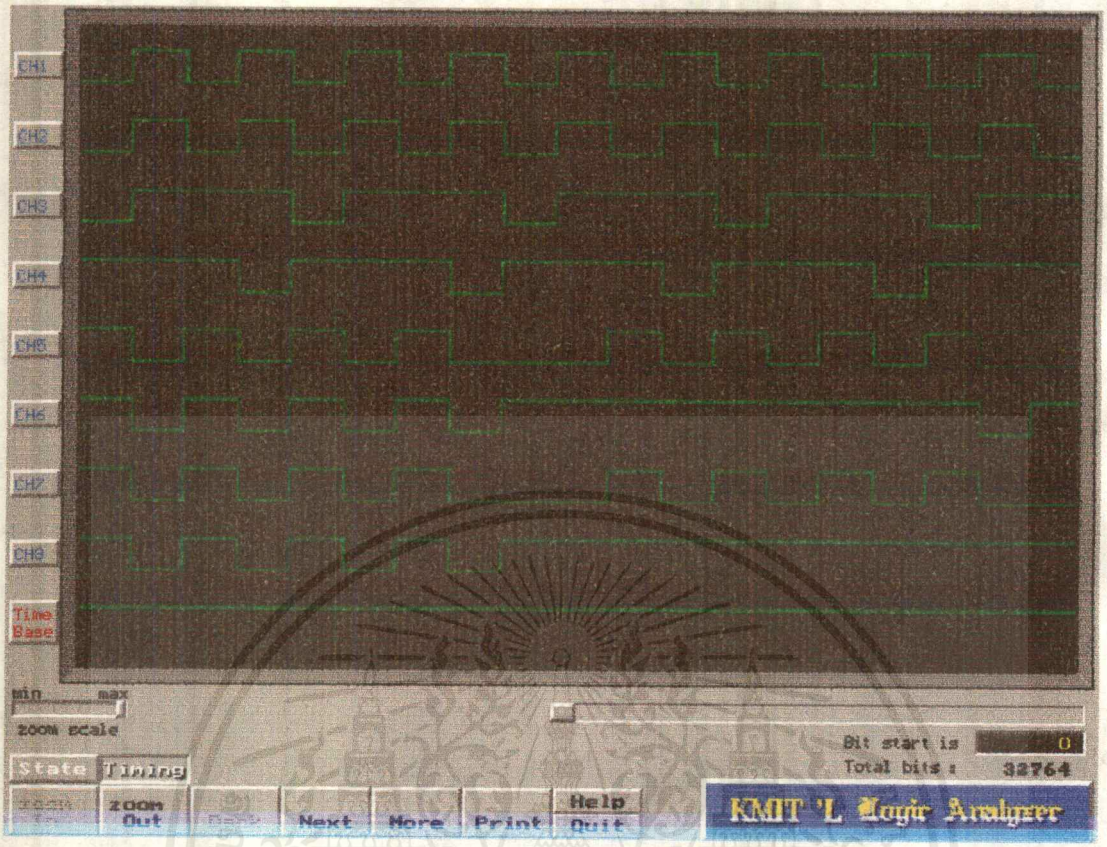


รูปที่ 5.2.29 ขั้นตอนการทำงานของโปรแกรม CLA ในขณะวัดสัญญาณ

6. Help menu เป็นส่วนอธิบายการใช้เมนูย่อยในการกำหนดค่า Time base, Threshold และ Channel

5.2.4.2 โปรแกรม PULSE.EXE

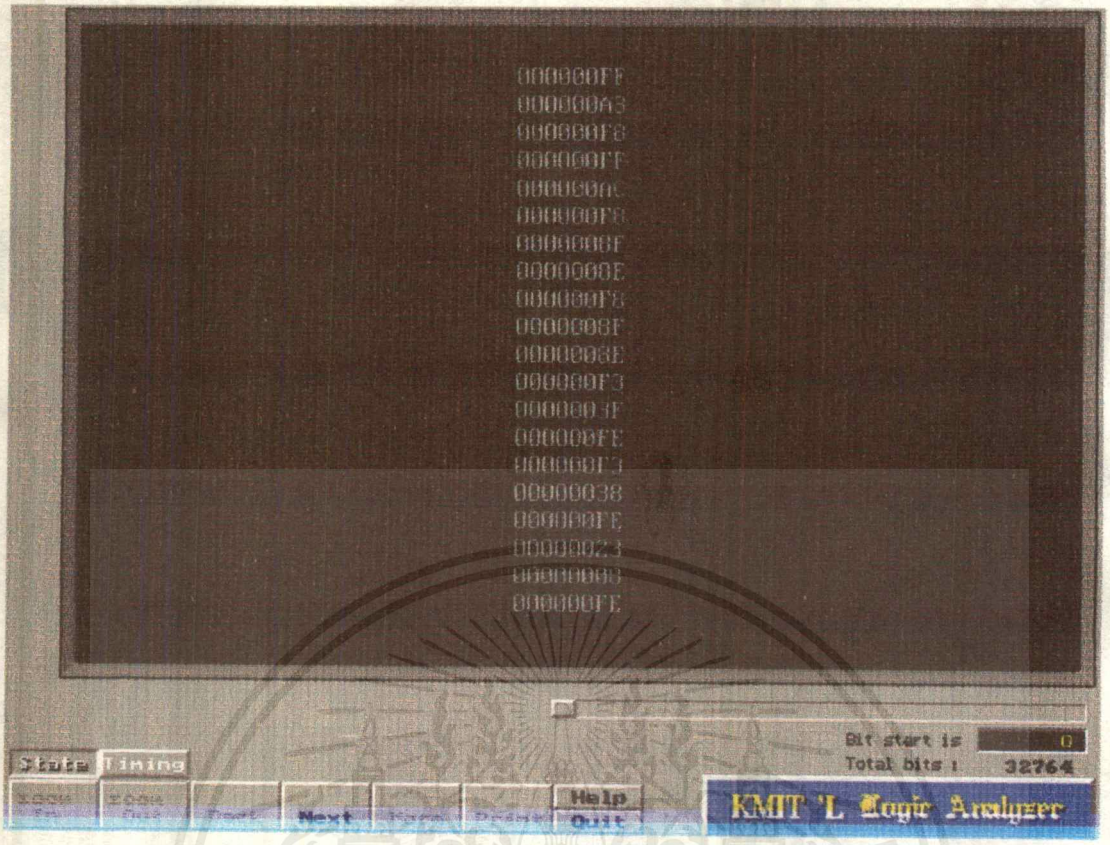
โปรแกรม PULSE นี้ เป็นโปรแกรมที่ใช้ในการประมวลผลและแสดงผลข้อมูลที่ทำการวัดได้ ซึ่งสามารถเรียกใช้ได้ด้วยการเรียกผ่านเมนูหลักดังที่กล่าวมาแล้วในหัวข้อ 5.2.1 หรือโดยการพิมพ์ PULSE ที่ DOS prompt ก็ได้ โปรแกรมนี้เป็นโปรแกรมที่แสดงผลใน graphics mode จึงเป็นโปรแกรมที่มีขนาดใหญ่ และต้องการหน่วยความจำในการ RUN โปรแกรมค่อนข้างมาก ดังนั้นหากเรียกโปรแกรมแล้วได้รับข้อความแจ้งว่าหน่วยความจำไม่พอ ก็ให้ทำการ Boot เครื่องใหม่ โดย bypass เอา device driver ที่ไม่จำเป็นในไฟล์ CONFIG.SYS ออกไปให้หมด (จากการทดลอง ถ้าติดตั้ง device driver HIMEM.SYS ลงไป จะเหลือ conventional memory ประมาณ 630 kb ก็จะทำการ RUN โปรแกรม PULSE.EXE ได้ อย่างไม่มีปัญหา) เมื่อเรียกโปรแกรม จะปรากฏหน้าจอดังรูป 5.2.30



รูปที่ 5.2.30 หน้าจอของ PULSE.EXE

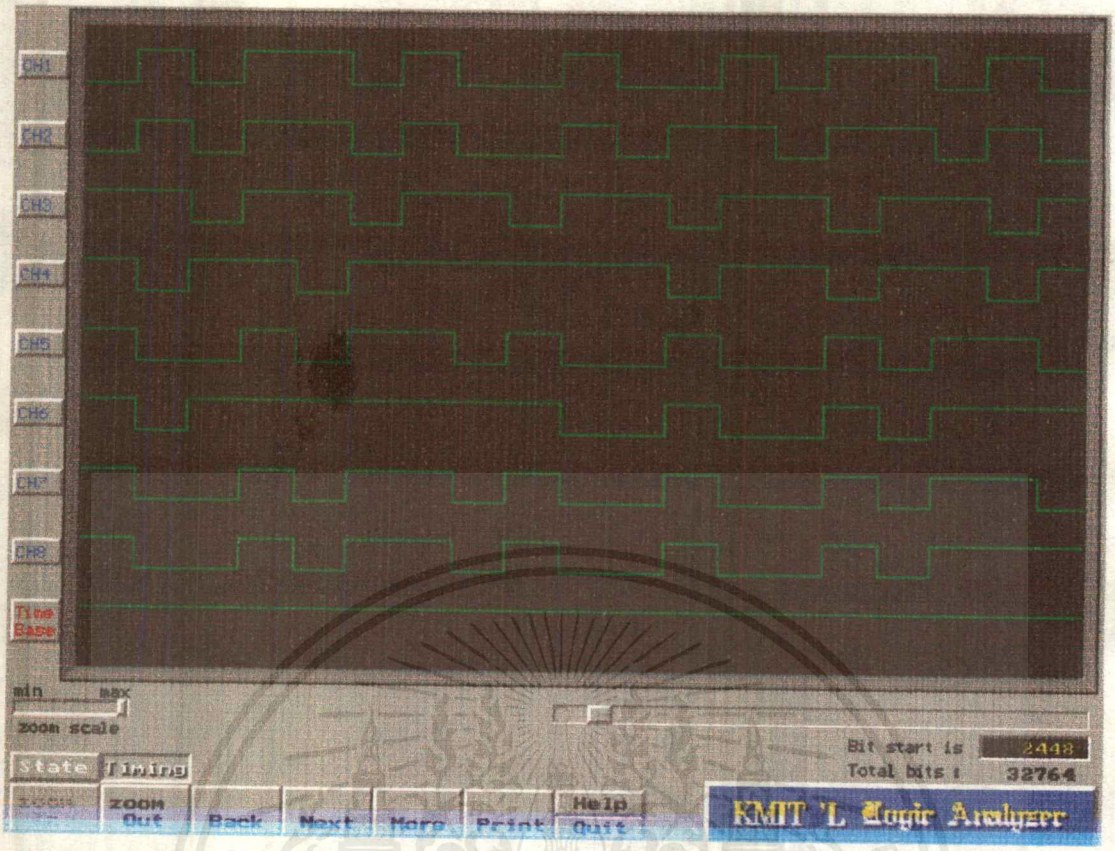
จากรูปที่ 5.2.30 จะเห็นว่ามีปุ่มที่ใช้สั่งงานต่างๆ อยู่ด้านล่างของจะภาพ ซึ่งปุ่มเหล่านี้สามารถกดได้โดยการกดคีย์ตัวอักษรประจำปุ่ม (จะเป็นตัวอักษรตัวใหญ่ในปุ่มนั้นๆ เช่น ปุ่ม ZOOM OUT คีย์ตัวอักษรที่ใช้งานก็คือตัว "O") และจะมีรายละเอียดของปุ่มต่างๆ ทั้งหมดดังนี้

1. State/Timing : (คีย์ที่ใช้คือ S และ T ตามลำดับ)จะเป็นปุ่มที่ใช้สลับโหมดการแสดงผล ระหว่าง TIMING MODE และ STATE MODE โดยใน TIMING MODE จะแสดงข้อมูลเป็นรูปสัญญาณโลจิก (ดังรูป 5.2.14) คือ HI,LOW และ HI IMPEDANCE จะเป็นช่องว่างส่วน HI และ LOW ก็จะเป็นเส้นตามปกติ) ส่วน State mode จะแสดงผลข้อมูลเป็นเลขฐาน 16 โดยใช้ channel 1 เป็น LSB และ channel สูงสุดเป็น MSB (ส่วน Timebase ไม่นำมาคิด) และถ้าในกรณีที่ channel ใด channel หนึ่งมีค่าสถานะโลจิกเป็น Hi Impedence ค่าของ state จะแสดงออกมาเป็น FFFFFFFF (ตัดสินใจไม่ได้) หน้าจอใน State mode แสดงดังรูป 5.2.31



รูปที่ 5.2.31 หน้าจอของ PULSE ใน State mode

2. Next Page : (คีย์ที่ใช้คือ N) ใช้ในการดูข้อมูลในหน้าถัดไป (ในกรณีที่ 1 หนึ่งหน้าจอ แสดงไม่ครบทั้งหมด) 1 หน้าจอ (ทั้ง Timing Diagram และ State จะใช้ปุ่มนี้ร่วมกัน) และจะทราบว่าเลื่อนดูข้อมูลมาถึงจุดใดของข้อมูลแล้วโดยการกด Bar ที่ข้างใต้จอด้านขวาซึ่งจะเลื่อนแสดงตำแหน่งของข้อมูล ดังรูป 5.2.32

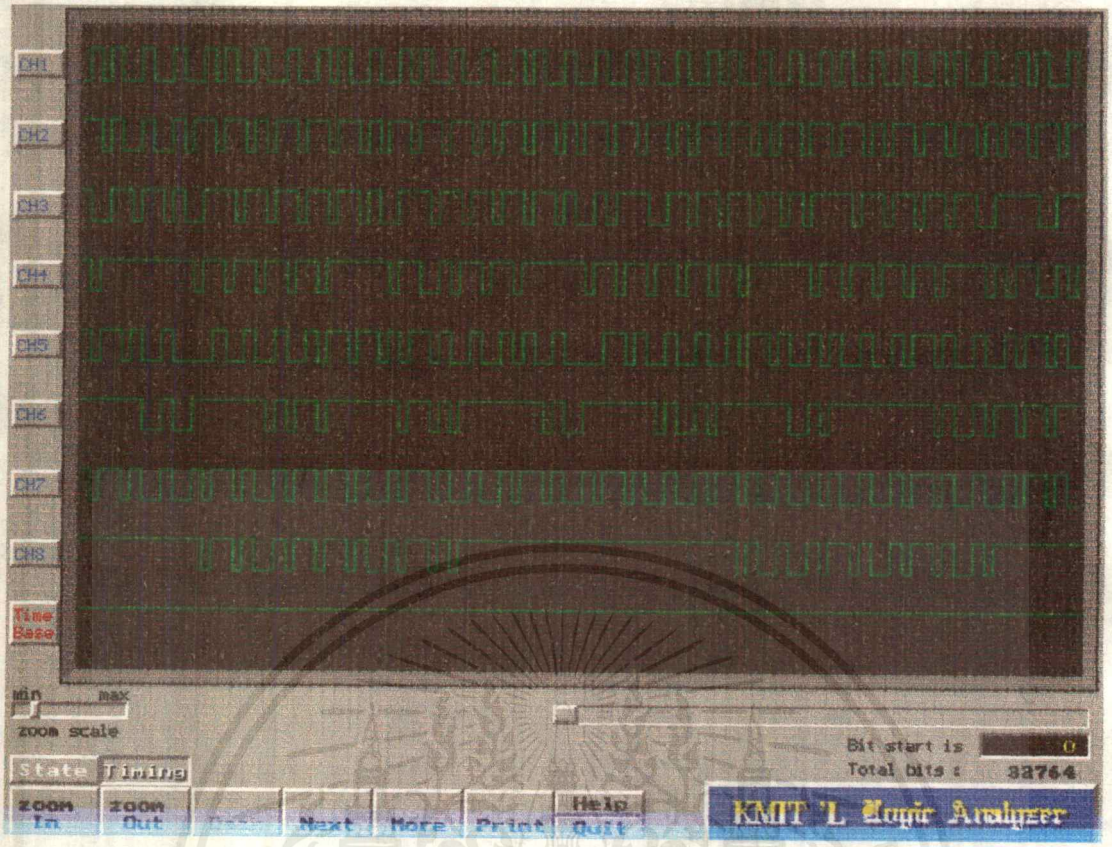


รูปที่ 5.2.32 หน้าจอของ PULSE เมื่อกดปุ่ม Next

3. Back Page : (คีย์ที่ใช้คือ B) ใช้ในการดูข้อมูลย้อนถอยหลังกลับมา 1 หน้าจอ (ทั้ง Timing Diagram และ State จะใช้ปุ่มนี้ร่วมกัน)

4. Zoom In : (คีย์ที่ใช้คือ I) ใช้ในการขยายขนาดความกว้างของช่วงเวลา ในโหมด Timing Diagram ซึ่งเราจะดูระดับของการ Zoom ได้ที่ช่อง Zoom Scale พิจารณาจากรูป 5.2.30 จะเห็นว่าเป็นการ Zoom In สูงสุด

5. Zoom Out : (คีย์ที่ใช้คือ O) ใช้ในการลดขนาดความกว้างของช่วงเวลา ในโหมด Timing Diagram แสดงดังรูปที่ 5.2.33



รูปที่ 5.2.33 หน้าจอของ PULSE.EXE เมื่อกดปุ่ม Zoom Out

6. More Channel : (คีย์ที่ใช้คือ M) เนื่องจากในโหมดของ Timing Diagram จะแสดง Timing Diagram ได้เพียงหน้าจอละ 8 Channel ดังนั้นถ้าต้องการดู Channel ที่สูงกว่านี้ ก็ทำได้โดยการใช้ปุ่ม More นั่นเอง และถ้าหากเป็นการใช้งานไม่ครบทุก Channel หากใช้ปุ่ม More ไปดูก็จะเห็นว่าไม่มีสัญญาณปรากฏที่ Channel ที่ไม่ได้ใช้งาน (ดูหมายเลข Channel ได้ที่ปุ่มด้านซ้ายสุดของจอ)

7. Print : (คีย์ที่ใช้คือ P) เป็นปุ่มที่ใช้ในการพิมพ์รูปสัญญาณในโหมดของ Timing Diagram

8. Help : (คีย์ที่ใช้คือ H) เป็นปุ่มที่ใช้ในการอธิบายการใช้งานโปรแกรม และปุ่มต่างๆ

9. Quit : (คีย์ที่ใช้คือ Q) เป็นปุ่มที่ใช้ในการออกจากโปรแกรม

และนอกจากนี้ยังมีปุ่มใช้งานที่ไม่แสดงให้เห็นที่หน้าจออีกด้วย คือ ปุ่ม Home และ ปุ่ม End ซึ่งทำหน้าที่กระโดดไปดูข้อมูลยังตำแหน่งเริ่มต้น และสิ้นสุดของข้อมูล ตามลำดับ โดยจะสามารถใช้ได้ทั้งในโหมดของ Timing Diagram และในโหมด State.

บทที่ 6

สรุป และ วิจารณ์

- การใช้งานในส่วนของการวิเคราะห์หาสมการจากไอซีตระกูล PAL

การใช้งานของเครื่องวิเคราะห์วงจรดิจิตอลด้วยซอฟท์แวร์โปรแกรมชุดใหม่ในการวิเคราะห์หาสมการ PAL มีลักษณะดังนี้

1. ไอซี PAL ที่สามารถอ่านได้ จะต้องเป็นประเภทที่ไม่มี Register อยู่ภายใน และต้องเป็นไอซีที่มีจำนวนอินพุทไม่เกิน 16 อินพุท และจำนวนเอาท์พุทสูงสุดได้ไม่เกิน 8 เอาท์พุท

2. การเลือกวิเคราะห์หาสมการจากทุกอินพุทของไอซี อาจทำให้ได้ค่า Minterm จำนวนมาก ซึ่งโปรแกรมใน PALควรจะกำหนดขาอินพุทเฉพาะขาที่ใช้งานจริงๆเท่านั้นเพื่อให้การวิเคราะห์หาสมการได้ในเวลาอันรวดเร็ว

3. ในการใช้งานบนหน่วยความจำปกติ (Conventional Memory) นั้น ไม่จำเป็นจะต้องเขียน CONFIG.SYS ใหม่ เพื่อให้ QM.EXE สามารถทำการวิเคราะห์ได้ ดังเช่น ชุดโปรแกรมเก่าที่จะต้อง RUN อยู่บนเนื้อที่ในหน่วยความจำตั้งแต่ 600 Kbyte ขึ้นไป แต่ในชุดโปรแกรมใหม่นี้จะใช้เนื้อที่เพียง 400 Kbyte ก็จะสามารถทำการวิเคราะห์วงจรได้แล้ว ซึ่งเป็นผลมาจากการจัดการหน่วยความจำแบบ ไดนามิก คือจะจองพื้นที่ใช้งานเฉพาะส่วนที่ใช้งานจริงๆ ส่วนเนื้อที่ที่เหลือก็จะเก็บอยู่ใน XMS โหมด ซึ่งเหมาะสำหรับติดตั้งบนเครื่องคอมพิวเตอร์รุ่นใหม่ๆที่ใช้เนื้อที่ในการใช้งานในหน่วยความจำ Conventional มาก โดยที่โปรแกรมชุดเก่านั้นจะไม่สามารถ RUN ได้เลย

4. สำหรับในส่วนของการวิเคราะห์หาสมการจากไอซีตระกูล PAL นั้น ในโปรแกรม QM.EXE นี้สามารถที่จะคำนวณได้ทั้ง Minterm และ Maxterm ได้อย่างถูกต้อง ด้วยการอ่านจากไฟล์ config ที่สร้างขึ้นจาก PLA.EXE ที่เขียนด้วยภาษาแอสแซมบลี ซึ่งโปรแกรม PLA.EXE นี้ สามารถที่จะสแกนค่าเอาท์พุทที่เป็น " 1 " และ " 0 " ได้ตามลักษณะของรูปแบบในการอัดโปรแกรมของไอซี PAL ตัวนั้นๆ

- การใช้งานในส่วนของการตรวจสอบไอซี TTL/CMOS

ไอซีที่ตรวจสอบได้จะต้องเป็นไอซีโลจิกเกทพื้นฐาน และไม่เป็นชนิด Open Collector เท่านั้น เนื่องจากความไม่เหมาะสมบางอย่างของตัว Hardware เอง จึงทำให้สามารถตรวจสอบไอซีได้จำนวนเบอร์ค่อนข้างน้อย

- การใช้งานในส่วนของ Logic Analyzer

1. เนื่องจาก Hardware ของเครื่องต้องใช้งานหลายอย่าง ดังนั้น ในส่วนของ Logic Analyzer จึงเป็นการสั่งให้เครื่องคอมพิวเตอร์อ่านข้อมูลจากสัญญาณที่วัดโดยตรง ไม่มีการใช้ Hardware ช่วยในการอ่านข้อมูลในส่วนนี้ ซึ่งทำให้สามารถวิเคราะห์สัญญาณในช่วงความถี่ที่ไม่สูงมากนัก (เช่น การสื่อสารอะซิงโครนัส) โดยความถี่สูงสุดที่สามารถวิเคราะห์ได้ จะขึ้นอยู่กับการเลือกจำนวน Channel และความเร็วในการทำงานของเครื่องคอมพิวเตอร์ด้วย

ความสัมพันธ์ระหว่างการเลือกจำนวน Channel กับความถี่สูงสุดที่วัดได้โดยใช้เครื่องคอมพิวเตอร์ที่ใช้ CPU เบอร์ 80486 DX2-80 จะได้ผลดังตารางที่ 6.1

จำนวน channel	ความถี่สูงสุดที่วัดได้	ช่วงที่เหมาะสมกับการใช้งาน
4	100 KHz	45 KHz
8	70 KHz	30 KHz
12	50 KHz	20 KHz
16	40 KHz	18 KHz
20	30 KHz	15 KHz
24	25 KHz	10 KHz

ตารางที่ 6.1 ความสัมพันธ์ระหว่างการเลือกจำนวน channel กับความถี่สูงสุดที่สามารถวัดได้

2. การเลือกจำนวน Channel มากกว่าการใช้งานจริง เช่นเลือกไว้ 8 channel แต่ใช้งานจริงเพียง 6 channel นั้น channel ที่เหลืออาจถูกสัญญาณรบกวนได้และการอ่านค่าใน State mode ไม่สามารถอ่านได้ (จะเป็น FFFFFFFF หมด เนื่องจากมี channel ที่เหลือเป็น Hi Impedance) ดังนั้น จึงควรจะต่อกับโลจิก " 0 " หรือกราวน์เอาไว้ เพื่อให้การอ่านสัญญาณไม่สับสน และสามารถอ่านค่าใน State mode ได้อย่างถูกต้องอีกด้วย

เอกสารอ้างอิง

1. The Monolithic Memories, Programmable Array Logic Handbook, Printed and bound by R.R. Donnelley & Sons Company.
2. Len Dorfman and More J.Neuberger, C++ Memory management, TAB Books is a division of McGraw-Hill, Inc,1994..
3. ธันวา ศรีประโมง, การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม, โครงการตำราวิชาการ มหาวิทยาลัยเทคโนโลยีมหานคร.
4. ปิติ สุคนธสุโรจน์, เทคนิคการจัดการหน่วยความจำสำหรับ MS-DOS 6.2, บริษัท ซีเอ็ด ยูเคชั่น จำกัด (มหาชน).
5. ดร.วิทยา เรืองพรวิสุทธิ์, คู่มือโปรแกรมภาษาซี, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน).
6. มนตรี พจนารถลาวัณย์, การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี, บริษัท ซีเอ็ด ยูเคชั่น จำกัด (มหาชน).

ภาคผนวก ก.

Software program

ซอฟท์แวร์ที่สำคัญสำหรับโครงงานนี้ มีดังนี้

- ***MAIN.C*** เป็นตัวสั่งงานสำหรับโปรแกรมทั้งหมดในโปรเจคนี้
- ***XMS.C*** เป็นโปรแกรมในการขอใช้หน่วยความจำเกินพันไบต์แรก (Extended Memory : XMS) (แสดงโฟลว์ชาร์ทในการทำงานสลับไปมาระหว่าง หน่วยความจำปกติ และ หน่วยความจำ .เกินพันไบต์แรก)

* โปรแกรมในส่วนของการวิเคราะห์ PAL *

- ***PLA.ASM*** ใช้อ่านข้อมูลจาก IC ตระกูล PAL ที่คีย์เข้ามา ในรูปของ file FUNCTxx.DAT
- ***20P.ASM*** ใช้อ่านข้อมูลจาก IC ตระกูล PAL ที่มี 20 ขา
- ***QM.C*** ใช้วิเคราะห์หาสมการจาก IC ตระกูล PAL
- ***READ.C*** เป็นโปรแกรมที่ใช้แสดงผลข้อมูลทางจอภาพ

* โปรแกรมในส่วนของ Logic Analyzer *

- ***CLA.ASM*** ทำหน้าที่อ่าน และเก็บข้อมูลในรูปของ file LOGICxx.DAT
- ***PULSE.C*** ทำหน้าที่ประมวลผล และแสดงผล

* โปรแกรมในส่วนของการทดสอบสถานภาพของไอซี TTL/CMOS *

- ***TTL.ASM*** เป็นโปรแกรมที่ใช้ตรวจสอบสถานภาพของไอซี TTL/CMOS

MAIN.C

```
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <process.h>
#include "nclock.h"

#define BORDER          1
#define ESC            27
#define ENTER          13
#define SPACE          32
#define UP             72
#define DOWN           80
#define LEFT           75
#define RIGHT          77

#define OTHER         150
#define ABORT         151

#define BLACK           0
#define BLUE            1
#define GREEN           2
#define CYAN            3
#define RED             4
#define MAGNETA         5
#define BROWN           6
#define HGRAY           7
#define GRAY            8
#define HBLUE           9
#define HGREEN         10
#define HCYAN          11
#define HRED           12
#define HMAGNETA       13
#define YELLOW         14
#define WHITE          15
#define BLUE_BK        16
#define GREEN_BK       32
#define CYAN_BK        48
#define RED_BK         64
#define MAGNETA_BK     80
#define BROWN_BK       96
#define GRAY_BK       112
#define BLINK         128

void goto_xy(int x,int y);
void main_ui(char *main_menu[],int main_count,int mainx,int mainy,char *main_keys,int menu_space);
int popup(char *menu[],char *keys,int count,int x,int y,int border);
int get_resp(int x,int y,int sub_count,int main_count,char *menu[],char *keys);
void save_video(int startx,int endx,int starty,int endy,unsigned int *buf_ptr);
void restore_video(int startx,int endx,int starty,int endy,unsigned char *buf_ptr);
void write_main(char *menu[],int count,int x,int y,int menu_space,int attrib,int attrib_hotkey);
void write_char_menu(int x,int y,char *p,int attrib,int attrib_hotkey);
void write_char(int x,int y,int p,int attrib);
void write_string(int x,int y,char *string,int attrib);
void display_menu(char *menu[],int x,int y,int count,int attrib,int attrib_hotkey);
void draw_border(int x,int y,int endx,int endy,int attrib);
void window(int x,int y,int endx,int endy,int attrib,int border);
int video_mode();
void cur_on();
void cur_off();
int is_in(char *s,char c);

char *mainbar[]= {
        "aAbout",
        "aPAL",
        "aTTL/CMOS",
```

```
        "cLogic Analyzer",
        "aHelp",
        "bExit",
};

char *about[]= {
        "c About Project "
};

char *pal[]= {
        "c Analysis PAL      ",
        "b Read last equation "
};

char *ttl[]= {
        "c Test IC "
};

char *logic[]= {
        "b Run Logic analyzer ",
        "b Display last data  "
};

char *help[]= {
        "d How to use..",
};

char *exitt[]= {
        "b OS Shell ",
        "b Quit    ",
};

char *out = "Type EXIT for return to menu project... ";
char *con = "Press any key to continue...";
char *ab[] = {
        "      **** About ****",
        "-----------------------",
        "      This  is a menu for  ",
        "  Digital Circuit Analyzer 1997",
        "",
        "Adviser :",
        "",
        " \" Mr. Phaisan Sittiyopatsakun \"",
        "",
        "Group by",
        "  Nualpun  Arunrat  38013361 3U",
        "  Sittawath Piyasil    38013382 3U",
        "  Sompong  Vivake  38013383 3U"
};

char *hlp[] = {
        "                ***** Help *****",
        "-------------------------------------------------",
        "      Use  arrow key  (UP, DOWN, LEFT, RIGHT)  or HOT KEY",
        "(charactor will be lighter than other charactor) for chose",
        "a submenu and press <ENTER> to run program",
        "",
        "    All menu are...",
        "",
        "< About >   : display about this program",
        "< PAL >     : analysis PAL or find equation of PAL",
        "< TTL/CMOS >: check IC TTL/CMOS (good or bad)",
        "< Logic Analyzer > : run logic analyzer program",
        "< Help >    : display help screen",
        "< Exit >    : exit program"
};

int hor_choice=1,save_hor=0,arrow_hor=0;
```

```
char far *vmem;

main()
{
  int vmode;
  for(;;)
    {
     clrscr();
     cur_off();
     vmode=video_mode();
     if(vmode==7) vmem=0xB0000000;
         else vmem=0xB8000000;
     window(0,0,80,1,BLACK | GRAY_BK,!BORDER);
     window(0,1,79,23,HGRAY | BLUE_BK,BORDER);
     window(0,24,80,25,RED | GRAY_BK,!BORDER);
     goto_xy(15,24); printf("Main menu Digital Circuit Analyzer Project 1997");
     write_main(mainbar,6,1,0,6,BLACK | GRAY_BK,RED | GRAY_BK);
     main_ui(mainbar,6,1,0,"aptghx",6);
     hor_choice=1,save_hor=0,arrow_hor=0;
    }
}

/*-------------------------------------*/

void main_ui(char *main_menu[],int main_count,int mainx,int mainy,char *main_keys,int menu_space)
{
  union inkey {
   char ch[2];
   int i;
   } c;

  int h=0;
  int resp=0;
  int startx[30];
  int loop,y_line;
  unsigned int *p;

  startx[0]=0;
  for(h=1;h<main_count;h++)
   startx[h]=strlen(main_menu[h-1])+menu_space+startx[h-1];
  write_char_menu(mainx,mainy,mainbar[arrow_hor],HGRAY | BLACK,RED | BLACK);

  LB2:

  for(;;)
   {
    clock(3,24);
    while(!bioskey(1));
    c.i=bioskey(0);
    write_char_menu(mainx+startx[hor_choice-1],mainy,mainbar[hor_choice-1],BLACK | GRAY_BK,RED | GRAY_BK
    save_hor=hor_choice;
    if(c.ch[0])
        {
         hor_choice=is_in(main_keys,tolower(c.ch[0]));
         if(hor_choice)
           {
            LB1:
            write_char_menu(mainx+startx[hor_choice-1],mainy,mainbar[hor_choice-1],HGRAY | BLACK,RED | BLA
            switch(hor_choice-1)
              {
               case 0:
                    {
                     resp=popup(about,"aptghxb",1,2,2,BORDER);
                     switch(resp)
                       {
                        case 0:
                          {
                            p=(unsigned int *) malloc((58-22+1) * (21-4+16));
```

```
                if(!p) exit(1);
                save_video(22,58,4,21,p);
                window(22,4,57,20,CYAN | GRAY_BK,BORDER);
                for(loop=0,y_line=5;loop<=12;loop++,y_line++)
                 {
                  write_string(24,y_line,ab[loop],BLACK | GRAY_BK);
                 }
                write_string(27,19,con,BROWN | BLINK | GRAY_BK);
                if(getch()==0) getch();
                restore_video(22,58,4,21,(char *)p);
                free(p);
                break;
            }
          }
        break;
       }
 case 1:
       {
       resp=popup(pal,"aptghxnr",2,10,2,BORDER);
       switch(resp)
         {
          case 0:
           {
                cur_on();
                p=(unsigned int *) malloc((100-0+1) * (27-0+16));
                if(!p) exit(1);
                save_video(0,80,0,25,p);
                window(0,0,80,25,HGRAY | BLACK,!BORDER);
                if(!(spawnl(P_WAIT,"PLA","PLA",NULL)))
                 {
                  cur_off();
                  spawnl(P_WAIT,"QM","QM",NULL);
                 }
                cur_off();
                restore_video(0,80,0,25,(char *)p);
    free(p);
                break;
           }
          case 1:
           {
                cur_off();
    p=(unsigned int *) malloc((100-0+1) * (27-0+16));
                if(!p) exit(1);
                save_video(0,80,0,25,p);
                window(0,0,80,25,HGRAY | BLACK,!BORDER);
                spawnl(P_WAIT,"READ","READ","QM.EQ",NULL);
                restore_video(0,80,0,25,(char *)p);
    free(p);
                break;
            }
          }
       break;
      }
case 2:
      {
      resp=popup(ttl,"aptghxe",1,22,2,BORDER);
      switch(resp)
        {
         case 0:
          {
    cur_on();
    p=(unsigned int *) malloc((100-0+1) * (27-0+16));
                if(!p) exit(1);
                save_video(0,80,0,25,p);
                window(0,0,80,25,HGRAY | BLACK,!BORDER);
                spawnl(P_WAIT,"TTL","TTL",NULL);
                cur_off();
    restore_video(0,80,0,25,(char *)p);
```

```
                free(p);
                break;
              }
            }
           break;
          }
      case 3:
          {
           resp=popup(logic,"aptghxrd",2,35,2,BORDER);
           switch(resp)
            {
             case 0:
              {
          cur_on();
          p=(unsigned int *) malloc((100-0+1) * (27-0+16));
                if(!p) exit(1);
                save_video(0,80,0,25,p);
                window(0,0,80,25,HGRAY | BLACK,!BORDER);
                if(!(spawnl(P_WAIT,"CLA","CLA",NULL)))
                  {
                   spawnl(P_WAIT,"PULSE","PULSE",NULL);
                  }
                cur_off();
          restore_video(0,80,0,25,(char *)p);
                free(p);
                break;
              }
             case 1:
              {
                cur_off();
          p=(unsigned int *) malloc((100-0+1) * (27-0+16));
                if(!p) exit(1);
                save_video(0,80,0,25,p);
                window(0,0,80,25,HGRAY | BLACK,!BORDER);
                spawnl(P_WAIT,"PULSE","PULSE",NULL);
                cur_off();
                restore_video(0,80,0,25,(char *)p);
          free(p);
                break;
              }
            }
           break;
          }
      case 4:
          {
           resp=popup(help,"aptghxw",1,55,2,BORDER);
           switch(resp)
            {
             case 0:
              {
                p=(unsigned int *) malloc((71-8+1) * (20-4+16));
                if(!p) exit(1);
                save_video(8,71,4,20,p);
                window(8,4,70,19,RED | GRAY_BK,BORDER);
                for(loop=0,y_line=5;loop<=13;loop++,y_line++)
                  {
                   write_string(10,y_line,hlp[loop],HCYAN | GRAY_BK);
                  }
                if(getch()==0) getch();
                restore_video(8,71,4,20,(char *)p);
                free(p);
                break;
              }
            }
           break;
          }
      case 5:
          {
```

```
                resp=popup(exitt,"aptghxoq",2,68,2,BORDER);
                switch(resp)
                 {
                  case 0:
                   {
                        clrscr();
                        cur_on();
                        write_string(0,0,out,RED | BLACK);
                        system("");
                        cur_off();
                        return;
                   }
                  case 1:
                   {
                        window(33,12,47,13,HGRAY | BLACK,!BORDER);
                        delay(30);
                        window(25,8,55,15,HGRAY | BLACK,!BORDER);
                        delay(40);
                        window(20,6,60,17,HGRAY | BLACK,!BORDER);
                        delay(50);
                        window(15,4,65,19,HGRAY | BLACK,!BORDER);
                        delay(50);
                        window(10,2,70,23,HGRAY | BLACK,!BORDER);
                        delay(50);
                        window(0,0,80,25,HGRAY | BLACK,!BORDER);
                        cur_on();
                        exit(1);
                        break;
                   }
                 }
                break;
               }
          }
        switch(resp)
          {
           case OTHER :
                {
                   write_char_menu(mainx+startx[(save_hor-1)],mainy,mainbar[(save_hor-1)],BLACK | GRAY_
GRAY_BK);
                if(hor_choice==main_count+1)
                 {
                  hor_choice=1;
                 }
                if(hor_choice==0)
                 {
                  hor_choice=main_count;
                 }
                goto LB1;
               }
           case ABORT : goto LB2;
          }
        goto LB1;
       }
      switch(c.ch[0])
       {
        case ENTER :
         {
          hor_choice=save_hor;
          goto LB1;
         }
       }
     }
  else
   {
    hor_choice=save_hor;
    switch(c.ch[1])
     {
      case LEFT : arrow_hor--; hor_choice=arrow_hor+1;
```

```
            break;
          case RIGHT : arrow_hor++; hor_choice=arrow_hor+1;
            break;
          case DOWN :
            {
             goto LB1;
            }
          }
        }
      save_hor=arrow_hor+1;
      if(arrow_hor==main_count)
        {
         arrow_hor=0; hor_choice=1; save_hor=1;
        }
      if(arrow_hor<0)
        {
         arrow_hor=main_count-1; hor_choice=main_count; save_hor=main_count;
        }
      write_char_menu(mainx+startx[arrow_hor],mainy,mainbar[arrow_hor],HGRAY | BLACK,RED | BLACK);
      hor_choice=save_hor;
    }
}

/*-----------------------------------*/

int get_resp(int x,int y,int sub_count,int main_count,char *menu[],char *keys)
{
  union inkey {
    char ch[2];
    int i;
    } c;
  int arrow_ver=0,ver_choice;
  int out=0;

  write_char_menu(x,y,menu[0],HGRAY | BLACK,RED | BLACK);
  for(;;)
    {
        clock(3,24);
    while(!bioskey(1));
    c.i=bioskey(0);

    write_char_menu(x,y+arrow_ver,menu[arrow_ver],BLACK | GRAY_BK,RED | GRAY_BK);
    save_hor=hor_choice;

    if(c.ch[0])
        {
        ver_choice = is_in(keys,tolower(c.ch[0]));
        if(ver_choice)
          {
          if(ver_choice<=main_count)
            {
             hor_choice=ver_choice;
             return OTHER;
            }
          else
            {
            ver_choice -= main_count;
            if((ver_choice-1)>arrow_ver)
            write_char_menu(x,y+((ver_choice-1)-arrow_ver),menu[(ver_choice-1)],HGRAY | BLACK,RED | BLACK
            if(ver_choice<=arrow_ver)
            write_char_menu(x,y-(arrow_ver-(ver_choice-1)),menu[(ver_choice-1)],HGRAY | BLACK,RED | BLACK
            return (ver_choice-1);
            }
          }
        switch(c.ch[0])
          {
          case ENTER : return arrow_ver;
          case SPACE : arrow_ver++;
```

```
            break;
          case ESC :
            {
             arrow_hor=hor_choice-1;
             return ABORT;
            }
         }
    }
    else
    {
        switch(c.ch[1])
         {
          case UP : arrow_ver--;
            break;
          case DOWN : arrow_ver++;
            break;
          case LEFT :
            {
             hor_choice--;
             out=1;
             break;
            }
          case RIGHT :
            {
             hor_choice++;
             out=1;
             break;
            }
         }
    }
    if(arrow_ver==sub_count) arrow_ver=0;
    if(arrow_ver<0) arrow_ver=sub_count-1;
    if(out) return OTHER;
    write_char_menu(x,y+arrow_ver,menu[arrow_ver],HGRAY | BLACK,RED | BLACK);
   }
}

/*-------------------------------*/

int popup(char *menu[],char *keys,int count,int x,int y,int border)
{
  register int i,len;
  int endx,endy,choice,menulen;
  unsigned int *p;

  if ((x>79) || (x<0) || (y>24) || (y<0))
   {
    printf("range error");
    return -2;
   }

  len=0;
  for(i=0;i<count;i++)
   {
    menulen=strlen(menu[i]);
    if(menulen>len) len=menulen;
   }
  endx=len+2+x-1;
  endy=count+1+y;
  if((endx>80) || (endy+1>24))
   {
    printf("menu won't fit");
    return -2;
   }
  p=(unsigned int *) malloc((endx-x+1) * (endy-y+16));
  if(!p) exit(1);
```

```
    save_video(x-1,endx-1,y-1,endy,p);
    window(x-1,y-1,endx-2,endy-1,YELLOW | GRAY_BK,border);
    display_menu(menu,x,y,count,BLACK | GRAY_BK,RED | GRAY_BK);
    choice=get_resp(x,y,count,6,menu,keys);
    restore_video(x-1,endx-1,y-1,endy,(char *)p);
    free(p);
    return choice;
}
/*---------------------------------------*/
void draw_border(int x,int y,int endx,int endy,int attrib)
{
  register int i;
  for(i=x+1;i<endx;i++)
   {
    write_char(i,y,205,attrib);
    write_char(i,endy,205,attrib);
   }
  for(i=y+1;i<endy;i++)
   {
    write_char(x,i,186,attrib);
    write_char(endx,i,186,attrib);
   }
  write_char(x,y,201,attrib);
  write_char(x,endy,200,attrib);
  write_char(endx,y,187,attrib);
  write_char(endx,endy,188,attrib);
}

/*---------------------------------------*/
void save_video(int startx,int endx,int starty,int endy,unsigned int *buf_ptr)
{
  union REGS r;
  register int i,j;
  for(i=starty;i<endy;i++)
   for(j=startx;j<endx;j++)
    {
        goto_xy(j,i);
        r.h.ah=8;
        r.h.bh=0;
        *buf_ptr++=int86(0x10,&r,&r);
    }
}

/*---------------------------------------*/
void restore_video(int startx,int endx,int starty,int endy,unsigned char *buf_ptr)
{
  union REGS r;
  register int i,j;
 for(i=starty;i<endy;i++)
   for(j=startx;j<endx;j++)
    {
        goto_xy(j,i);
        r.h.ah=9;
        r.h.bh=0;
        r.x.cx=1;
        r.h.al=*buf_ptr++;
        r.h.bl=*buf_ptr++;
        int86(0x10,&r,&r);
    }
```

```
    }

/*-------------------------------------------------*/

void display_menu(char *menu[],int x,int y,int count,int attrib,int attrib_hotkey)
{
  register int i;
  for(i=0;i<count;i++,y++)
    {
    goto_xy(x,y);
    write_char_menu(x,y,menu[i],attrib,attrib_hotkey);
    }
}

/*-------------------------------*/

void write_char_menu(int x,int y,char *p,int attrib,int attrib_hotkey)
{
  union REGS r;
  register int i,j;
  int hot,loop,save_attrib;
  hot=*p-96;
  *p++;
  save_attrib=attrib;
  for(i=x,loop=1;*p;i++,loop++)
    {
    if(loop==hot)
        {
        attrib=attrib_hotkey;
        }
    goto_xy(i,y);
    r.h.ah=9;
    r.h.bh=0;
    r.x.cx=1;
    r.h.al=*p++;
    r.h.bl=attrib;
    int86(0x10,&r,&r);
    attrib=save_attrib;
    }
}

/*-------------------------------*/

void goto_xy(int x,int y)
{
  union REGS r;
  r.h.ah=2;
  r.h.dl =x;
  r.h.dh=y;
  r.h.bh=0;
  int86(0x10,&r,&r);
}

/*---------------------------------------*/

void write_char(int x,int y,int p,int attrib)
{
  register int i;
  char huge *v;
  v=vmem;
  v += y*160 + x*2;
  *v++ = p;
  *v = attrib;
}

/*---------------------------------------*/

void write_main(char *menu[],int count,int x,int y,int menu_space,int attrib,int attrib_hotkey)
```

```
{
  register int i;
  int menulen;
  menulen=0;
  for(i=0;i<count;i++,x+=(menulen+menu_space))
   {
    menulen=strlen(menu[i]);
    goto_xy(x,y);
    write_char_menu(x,y,menu[i],attrib,attrib_hotkey);
   }
}
/*---------------------------------------*/

void window(int x,int y,int endx,int endy,int attrib,int border)
{
  register int i,j;
  for(j=y;j<endy;j++)
   for(i=x;i<endx;i++)
    write_char(i,j,32,attrib);
  if(border)
  draw_border(x,y,endx,endy,attrib);
}
/*---------------------------------------*/

int is_in(char *s,char c)
{
  register int i;
  for(i=0;*s;i++)
   if(*s++==c)
     return i+1;
  return 0;
}

void cur_off()
{
  union REGS r;
  r.h.ch=0x20;
  r.h.cl=0;
  r.h.ah=1;
  int86(0x10,&r,&r);
}

/*---------------------------------------*/

void cur_on()
{
  union REGS r;
  r.h.ch=5;
  r.h.cl=6;
  r.h.ah=1;
  int86(0x10,&r,&r);
}
/*---------------------------------------*/

int video_mode()
{
 union REGS r;
 r.h.ah=15;
 return int86(0x10,&r,&r)&255;
}
/*---------------------------------------*/

void write_string(int x,int y,char *string,int attrib)
{
```

```
    union REGS r;
    register int i,j;
    for(i=x;*string;i++)
      {
      goto_xy(i,y);
      r.h.ah=9;
      r.h.bh=0;
      r.x.cx=1;
      r.h.al=*string++;
      r.h.bl=attrib;
      int86(0x10,&r,&r);
      }
}
```

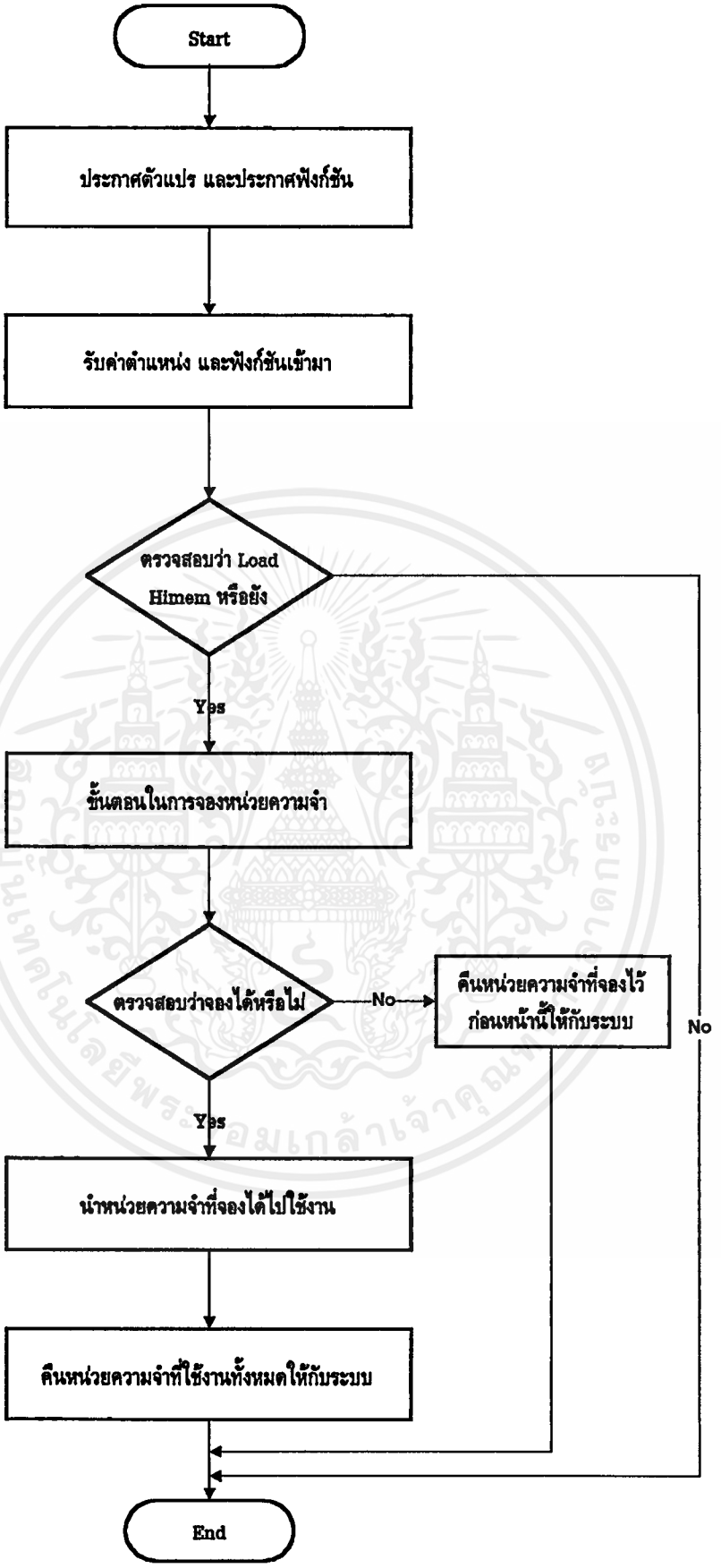
XMS.C

```
/*------- XMS.C ------------*/

#include <dos.h>

struct XMSBlock
{
  long nbytes;
  int shandle;
  long soffset;
  int dhandle;
  long doffset;
};

struct XMSBlock XMS_dat;

void far ((*XMSentry))(void);

/* error code for general use */

int XMS_ERR;
char *XMS_E80="Function no Implemented \0";
char *XMS_E81="VDISK was detected\0";
char *XMS_E82="An A20 Error occured\0";
char *XMS_E8E="A General Driver Error\0";
char *XMS_E8F="Unrecovered Driver Error\0";
char *XMS_E90="HMA Does not exist\0";
char *XMS_E91="HMA already in use\0";
char *XMS_E92="DX is less than the /HMAMIN=parameter\0";
char *XMS_E93="HMA is not allocated\0";
char *XMS_EA0="All extended memory is alocated\0";
char *XMS_EA1="All extended memory handles are alocated\0";
char *XMS_EA2="Invalid Handle\0";
char *XMS_EA3="Source handle is invalid\0";
char *XMS_EA4="Source offset is invalid\0";
char *XMS_EA5="Distination handle is invalid\0";
char *XMS_EA6="Distination offset is invalid\0";
char *XMS_EA7="Length is invalid\0";
char *XMS_EA8="Move has an invalid overlap\0";
char *XMS_EA9="Parity Error occured\0";
char *XMS_EAA="Block is not locked\0";
char *XMS_EAB="Block is not locked\0";
char *XMS_EAC="Block lock count overflow\0";
char *XMS_EAD="Lock Failed\0";
char *XMS_EB0="Only a smaller UMB is available\0";
char *XMS_EB1="No UMB are available\0";
char *XMS_EB2="UMB segment number is invalid\0";
char *XMS_E00="No Error occured\0";
char *XMS_EFF="Invalid error code,Error code is not known\0";

/* ------- Declairation Section -------- */
char far *XMSerr(void);
int      XMSinstalled(void);
int      XMSleftcon(void);
int      XMSleftall(void);
int      XMSalloc(int Kbyte);
unsigned long XMSlock(int handle);
int      XMSunlock(int handle);
int      XMSfree(int handle);
int      XMSrealloc(int handle,int newsize);
int      XMStransfer(struct XMSBlock *XM);
int      XMSread(void far *str,int handle,long n);
int      XMSwrite(void far *str,int handle,long n);
int      XMSreadoff(void far *str,int handle,long n,long offset);
int XMSwriteoff(void far *str,int handle,long n,long offset);
```

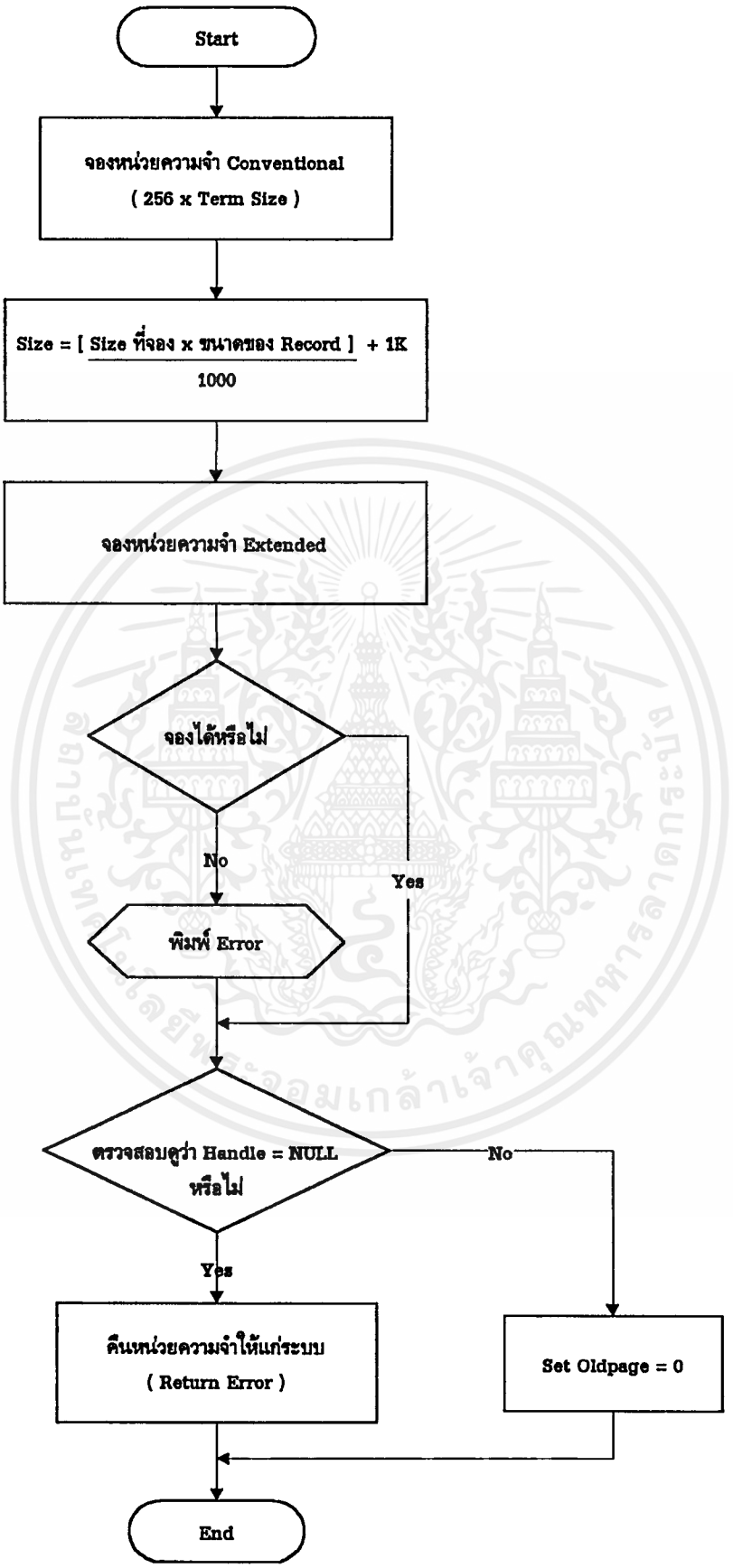
```
/* -------- Module body section ---------- */
char far *XMSerr(void)
{

        switch(XMS_ERR)
        {
                case 0x00 : return XMS_E00;
                case 0x80 : return XMS_E80;
                case 0x81 : return XMS_E81;
                case 0x82 : return XMS_E82;
                case 0x8E : return XMS_E8E;
                case 0x8F : return XMS_E8F;
                case 0x90 : return XMS_E90;
                case 0x91 : return XMS_E91;
                case 0x92 : return XMS_E92;
                case 0x93 : return XMS_E93;
                case 0xA0 : return XMS_EA0;
                case 0xA1 : return XMS_EA1;
                case 0xA2 : return XMS_EA2;
                case 0xA3 : return XMS_EA3;
                case 0xA4 : return XMS_EA4;
                case 0xA5 : return XMS_EA5;
                case 0xA6 : return XMS_EA6;
                case 0xA7 : return XMS_EA7;
                case 0xA8 : return XMS_EA8;
                case 0xA9 : return XMS_EA9;
                case 0xAA : return XMS_EAA;
                case 0xAB : return XMS_EAB;
                case 0xAC : return XMS_EAC;
                case 0xAD : return XMS_EAD;
                case 0xB0 : return XMS_EB0;
                case 0xB1 : return XMS_EB1;
                case 0xB2 : return XMS_EB2;
        }
        return XMS_EFF;
}

int XMSinstalled(void)
{
        void *int2F;

        int2F = (void *)getvect(0x2F);
        if(((unsigned long)int2F) ==
                        0x00000000UL || (*(char far *)int2F) == '\xcf')
                return 0;
        _AX = 0x4300;
        geninterrupt(0x2F);
        if(_AL != 0x80) return 0;
        else
        {
                _AX = 0x4310;
                geninterrupt(0x2F);
                XMSentry = (void far (*)())MK_FP(_ES,_BX);
        }
        XMS_ERR = 0;
        return 1;
}

int     XMSleftcon(void)
{
        register int temp;
        _AH = 0x08;
        (*XMSentry)();
        temp = _AX;
        XMS_ERR = _BL;
        return temp;
}
```

```
int XMSleftall(void)
{
        _AH = 0x08;
        (*XMSentry)();
        XMS_ERR = _BL;
        return _DX;
}

int XMSalloc(int Kbyte)
{
        _DX = Kbyte;
        _AH = 0x09;
        (*XMSentry)();
        if(_AX == 0)
        {
                XMS_ERR = _BL;
                return 0;
        }
        XMS_ERR = 0;
        return _DX;
}

unsigned long XMSlock(int handle)
{
        _DX = handle;
        _AH = 0x0C;
        (*XMSentry)();
        if(_AH == 0)
        {
                XMS_ERR = _BL;
                return 0UL;
        }
        return (unsigned long)MK_FP(_DX,_BX);
}

int XMSunlock(int handle)
{
        register int temp;
        _DX = handle;
        _AH = 0x0D;
        (*XMSentry)();
        temp = _AX;
        XMS_ERR = _BL;
        return temp;
}

int XMSfree(int handle)
{
        register int temp;
        _DX = handle;
        _AH = 0x0A;
        (*XMSentry)();
        temp = _AX;
        XMS_ERR = _BL;
        return temp;
}

int XMSrealloc(int handle,int newsize)
{
        register int temp;
        _BX = newsize;
        _DX = handle;
        _AH = 0x0F;
        (*XMSentry)();
        temp = _AX;
        XMS_ERR = _BL;
        return temp;
```

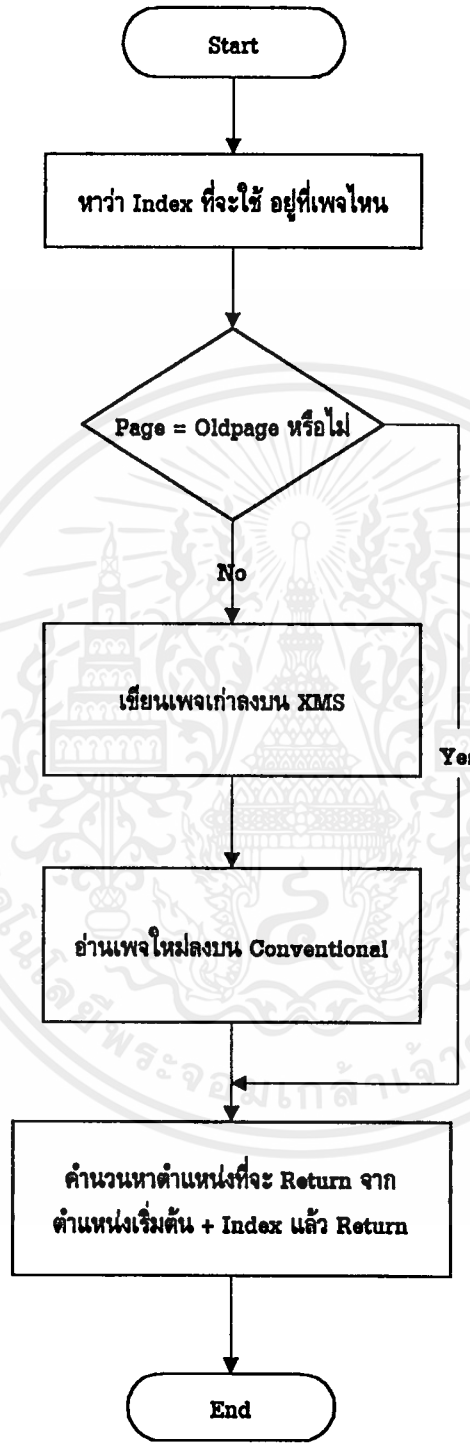
```
}

int XMStransfer(struct XMSBlock *XM)
{
        register int temp;

        _DS = FP_SEG(XM);
        _SI = FP_OFF(XM);
        _AH = 0x0B;
        (*XMSentry)();
        temp = _AX;
        XMS_ERR = _BL;
        return temp;
}

int XMSread(void far *str,int handle,long n)
{
        XMS_dat.nbytes = (unsigned long)((n>>1)<<1);
        XMS_dat.shandle = handle;
        XMS_dat.soffset = (unsigned long)0;
        XMS_dat.dhandle = 0;
        XMS_dat.doffset = (unsigned long)str;
        return XMStransfer(&XMS_dat);
}

int XMSwrite(void far *str,int handle,long n)
{
        XMS_dat.nbytes = (unsigned long)((n>>1)<<1);
        XMS_dat.shandle = 0;
        XMS_dat.soffset = (unsigned long)str;
        XMS_dat.dhandle = handle;
        XMS_dat.doffset = (unsigned long)0;
        return XMStransfer(&XMS_dat);
}

int XMSreadoff(void far *str,int handle,long n,long offset)
{
        XMS_dat.nbytes = (unsigned long)((n>>1)<<1);
        XMS_dat.shandle = handle;
        XMS_dat.soffset = (unsigned long)offset;
        XMS_dat.dhandle = 0;
        XMS_dat.doffset = (unsigned long)str;
        return XMStransfer(&XMS_dat);
}

int XMSwriteoff(void far *str,int handle,long n,long offset)
{
        XMS_dat.nbytes = (unsigned long)((n>>1)<<1);
        XMS_dat.shandle = 0;
        XMS_dat.soffset = (unsigned long)str;
        XMS_dat.dhandle = handle;
        XMS_dat.doffset = (unsigned long)offset;
        return XMStransfer(&XMS_dat);
}
```



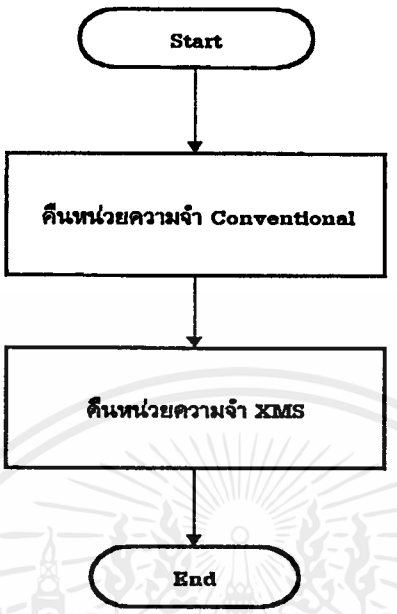
1. โฟลว์ชาร์ทของขั้นตอนการขอใช้และคืนหน่วยความจำที่ใช้ในโปรแกรม



2. โฟลว์ชาร์ทของการทำงานในการจองหน่วยความจำ



3.โฟลว์ชาร์ทของการย้ายสลับไปมาระหว่าง XMS กับ Conventional Memory



4.โฟลว์ชาร์ทของการคืนหน่วยความจำให้กับระบบ

PLA.ASM

```
outport macro port,data
                        mov  dx,300h
                        mov  al,port
                        out  dx,al
                        mov  dx,30fh
                        mov  al,data
                        out  dx,al
                        endm
inport macro port
                        mov  dx,300h
                        mov  al,port
                        out  dx,al
                        mov  dx,30fh
                        in   al,dx
                        endm
;/*************************************************************
cseg segment public
assume cs:cseg,ds:dseg
public menu
extrn mov_cursor:near,get_cursor:near,write_char_n_time:near
extrn cursor_right:near,cursor_left:near
extrn send_crlf:near,write_char:near,window2:near
extrn write_table:near,window:near,write_text:near
extrn read_key:near,write_bar_status:near,cursor_off:near
menu proc near
                        mov  ax,dseg
                        mov  ds,ax
                        mov  ax,3
                        int  10h
                        outport 13h,88h
                        outport 10h,80h
                        inport  10h
                        cmp  al,80h
                        je   work
                        jmp  nohard
work:                outport 11h,80h
                        call cursor_off
                        call del_file
                        call write_head
                        call write_main
                        call write_bar_status
                        call write_status
                        call read_key
end_main:             mov  ax,4c01h
                        int  21h
nohard:              mov  ah,9
                        lea  dx,nohard0
                        int  21h
                        lea  dx,nohard1
                        int  21h
                        lea  dx,nohard2
                        int  21h
                        lea  dx,press
                        int  21h
                        mov  ah,7
                        int  21h
                        jmp  end_main
menu endp
;/*************************************************************
write_head proc near
                        mov  bh,00010111b
                        mov  cx,0
                        mov  dx,184fh
                        call window
```

```
                        lea  si,bar2
                        call write_text
                        mov  bh,00111111b
                        mov  cx,0f18h
                        mov  dx,0f35h
                        call window2
                        mov  dx,0f22h
                        call mov_cursor
                        mov  bl,bh
                        lea  si,bar3
                        call write_text
                        ret
write_bar endp
;/*************************************************************
public write_status
write_status proc near
                        mov  dx,130bh
                        call mov_cursor
                        mov  bl,7eh
                        lea  si,number_mes
                        call write_text
                        mov  dx,140bh
                        call mov_cursor
                        mov  bl,7fh
                        mov  cx,58
                        mov  al,0b1h
                        call write_char_n_time
                        ret
write_status endp
;/*************************************************************
del_file proc near
                        mov  cx,10
                        lea  dx,funct01
del_f:          mov  ah,41h
                        int  21h
                        add  dx,12
                        loop del_f
                        ret
del_file endp
;/*************************************************************
cseg ends
dseg segment public
extrn funct01:byte
head1       db "      Programable Array Logic Reader",0
            db "Copyright (c) 1996-1997 by KMIT'L Engineer 3U",0
main_menu   db "Main Menu",0
public bar1,bar2,bar3
bar1        db "^Assignment",0
bar2        db "  ^Start",0
bar3        db "  E^xit ",0
number_mes  db "PAL Number :",0
nohard0     db 0dh,0ah,"Programable Array Logic Reader.$"
nohard1     db 0dh,0ah,"Hardware not install or device error..!$"
nohard2     db 0dh,0ah,"Copyright (c) 1996-1997 by KMIT'L Engineer. 3U",0dh,0ah,"$"
press       db "Press a key to continue...",0dh,0ah,"$"
dseg ends
end menu
```

20P.ASM

```
outport macro port,data
                mov  dx,300h
                mov  al,port
                out  dx,al
                mov  dx,30fh
                mov  al,data
                out  dx,al
                endm
inport macro port
                mov  dx,300h
                mov  al,port
                out  dx,al
                mov  dx,30fh
                in   al,dx
                endm
;/*************************************************************
cseg segment public
assume cs:cseg,ds:dseg
extrn mov_cursor:near,write_char:near,write_text:near
extrn send_crlf:near
public pal20
pal20 proc near
end_mes1:           mov  bx,1
                mov  al,value1
                mov  word ptr max_scan,0
inc_scan:         or   word ptr max_scan,bx
                dec  al
                jz   ok_scan
                shl  bx,1
                jmp  inc_scan
ok_scan:          mov  dx,0
                mov  ax,max_scan
                mov  cx,58
                div  cx
                mov  word ptr maxchar,ax
                mov  dx,140bh
                call mov_cursor
;/********************* initall switch ********************
                outport 13h,88h    ;gen
                outport 10h,80h    ;+5v
                outport 03h,80h    ;op
                outport 07h,80h    ;sw
                outport 04h,0ffh   ;off
                outport 05h,0ffh   ;off
                outport 06h,0ffh   ;off
                outport 0bh,9bh    ;ip1
                outport 0fh,9bh    ;ip2
                outport 11h,0c0h
                mov  byte ptr port1,0
                mov  byte ptr port2,0
                mov  byte ptr port3,0
                lea  si,ip_hex
                mov  dl,value1
load_more:          lodsb
                cmp  al,8
                jg   no_port1
                sub  al,1
                mov  cl,al
                mov  al,00000001b
                shl  al,cl
                or   byte ptr port1,al
                dec  dl
                jz   end_ip
                jmp  load_more
```

```
                        ;**** write header ***********
                        mov bh,00110111b
                        mov cx,105h
                        mov dx,44ah
                        call window
                        mov dh,1
                        mov dl,6
                        call mov_cursor
                        mov bl,00111111b
                        mov bh,3
                        mov cl,44h
                        call write_table
                        mov dx,211h
                        call mov_cursor
                        mov cx,2
                        lea si,head1
                        mov bl,00111110b
loop_head1:     call write_text
                        inc dh
                        call mov_cursor
                        loop loop_head1
                        ret
write_head endp
;/*************************************************************
public write_main
write_main proc near
                        mov bh,01110001b
                        mov cx,705h
                        mov dx,164ah
                        call window
                        mov bh,01010111b
                        mov cx,810h
                        mov dx,83fh
                        call window2
                        mov bl,01011110b
                        mov dx,823h
                        call mov_cursor
                        lea si,main_menu
                        call write_text
                        call write_bar
                        ;**** write status ****
                        mov dx,1208h
                        call mov_cursor
                        mov bl,7fh
                        mov bh,4
                        mov cl,40h
                        call write_table
                        ret
write_main endp
;/*************************************************************
public write_bar
write_bar proc near
                        mov bh,00111111b
                        mov cx,0b18h
                        mov dx,0b35h
                        call window2
                        mov dx,0b22h
                        call mov_cursor
                        mov bl,bh
                        lea si,bar1
                        call write_text
                        mov bh,00111111b
                        mov cx,0d18h
                        mov dx,0d35h
                        call window2
                        mov dx,0d22h
                        call mov_cursor
                        mov bl,bh
```

```
no_port1:       cmp  al,12
                jg   no_port2
                sub  al,9
                cmp  al,2
                jge  jump_nc
                mov  cl,al
                mov  al,00000001b
                shl  al,cl
                or   byte ptr port2,al
                jmp  end_port2
jump_nc:        mov  cl,al
                mov  al,00010000b
                shl  al,cl
                or   byte ptr port2,al
end_port2:      dec  dl
                jz   end_ip
                jmp  load_more
no_port2:       sub  al,13
                mov  cl,al
                mov  al,00000001b
                shl  al,cl
                or   byte ptr port3,al
                dec  dl
                jz   end_ip
                jmp  load_more
end_ip:         not  byte ptr port1
                not  byte ptr port2
                not  byte ptr port3
                and  byte ptr port2,11111101b
                outport 04h,port1
                outport 05h,port2
                outport 06h,port3
;/********************* out scan to pal ********************
                mov  word ptr f1_count,0
                mov  word ptr f2_count,0
                mov  word ptr f3_count,0
                mov  word ptr f4_count,0
                mov  word ptr f5_count,0
                mov  word ptr f6_count,0
                mov  word ptr f7_count,0
                mov  word ptr f8_count,0
                mov  word ptr f9_count,0
                mov  word ptr f10_count,0
                mov  bp,0
main_loop:      mov  bl,value1
                lea  si,ip_hex
                mov  dx,bp
                mov  byte ptr port1,0
                mov  byte ptr port2,0
                mov  byte ptr port3,0
load_ip:        mov  ax,0
                clc
                shr  dx,1
                adc  ah,0
                lodsb
                cmp  al,8
                jg   no_port1_ip
                sub  al,1
                mov  cl,al
                shl  ah,cl
                or   byte ptr port1,ah
                dec  bl
                jz   end_ip_ip
                jmp  load_ip
no_port1_ip:    cmp  al,12
                jg   no_port2_ip
                sub  al,9
                cmp  al,2
```

```
                jge  jump_nc_ip
                mov  cl,al
                shl  ah,cl
                or   byte ptr port2,ah
                jmp  end_port2_ip
jump_nc_ip:         mov  cl,al
                add  cl,4
                shl  ah,cl
                or   byte ptr port2,ah
end_port2_ip:       dec  bl
                jz   end_ip_ip
                jmp  load_ip
no_port2_ip:        sub  al,13
                mov  cl,al
                shl  ah,cl
                or   byte ptr port3,ah
                dec  bl
                jz   end_ip_ip
                jmp  load_ip
end_ip_ip:
;/********************** send scan value ******************
                push ax
                push dx
                and  byte ptr port2,11111101b
                outport 00h,port1
                outport 01h,port2
                outport 02h,port3
                call delay
                pop  dx
                pop  ax
;/********************** Keep outport **********************
                push ax
                push bx
                push cx
                push dx
                push si
                push di
                push bp
                mov  bl,value2
                lea  si,op_hex
load_op:           lodsb
                cmp  al,12
                ;jg   no_portc
                JG MM0
                JMP MM1
MM0:               JMP NO_PORTC
MM1:
                cmp  al,12
                je   f12

f11:
                CMP BYTE PTR FX1,0
                JNE L11

H11:               inport 0ch
                and  al,00110000b
                cmp  al,00100000b
                jne  f11_no1
                JMP NR11
L11:
                inport 0ch
                and  al,00110000b
                cmp  al,00100000b
                JE F11_NO1
NR11:
                mov  di,f9_count
                push es
                mov  ax,fseg3
```

```
                mov  es,ax
                mov  word ptr es:[di+4000h],bp
                pop  es
                inc  word ptr f9_count
                inc  word ptr f9_count
                CMP WORD PTR F9_COUNT,1000
                JLE F11_NO1
                MOV BYTE PTR FX1,1
                JMP OOK

f11_no1:          dec  bl
               ; jz  end_op_far
                JZ KK1
                JMP LOAD_OP
KK1:              JMP END_OP_FAR
KK2:
               ; jmp  load_op

f12:
                CMP BYTE PTR FX2,0
                JNE L12
H12:
                inport 0ch
                and  al,11000000b
                cmp  al,10000000b
                jne  f11_no1
                JMP NR12

L12:
                inport 0ch
                and  al,11000000b
                cmp  al,10000000b
                je  f11_no1

NR12:
                mov  di,f8_count
                push es
                mov  ax,fseg3
                mov  es,ax
                mov  word ptr es:[di+0000h],bp
                pop  es
                inc  word ptr f8_count
                inc  word ptr f8_count
                CMP WORD PTR F8_COUNT,1000
                JLE LESS12
                MOV BYTE PTR FX2,1
                JMP OOK

LESS12:
                jmp  f11_no1
no_portc:         cmp  al,16
                jg   no_portd_far
                cmp  al,16
                je   f16_far
                cmp  al,15
               ; je   f15
                JE SLX1
                JMP SLX0
SLX1:             JMP F15
SLX0:
                cmp  al,14
                je   f14
f13:
                CMP BYTE PTR FX3,0
                JNE L13
H13:              inport 0dh
                and  al,00000011b
```

```
                cmp  al,00000010b
                jne  f13_no1
                JMP NR13
L13:
                inport 0dh
                and  al,00000011b
                cmp  al,00000010b
                je  f13_no1
NR13:
                mov  di,f7_count
                push es
                mov  ax,fseg2
                mov  es,ax
                mov  word ptr es:[di+0c000h],bp
                pop  es
                inc  word ptr f7_count
                inc  word ptr f7_count
                CMP  WORD PTR F7_COUNT,1000
                JLE F13_NO1
                MOV BYTE PTR FX3,1
                JMP OOK

f13_no1:          dec  bl
                jz  end_op_far
                jmp  load_op
end_op_far:       jmp  end_op
no_portd_far:     jmp  no_portd
f16_far:          jmp  f16

f14:              CMP BYTE PTR FX4,0
                JNE L14
                inport 0dh
                and  al,00001100b
                cmp  al,00001000b
                jne  f13_no1
                JMP  NR14

L14:              Inport 0dh
                and  al,00001100b
                cmp  al,00001000b
                je  f13_no1

NR14:
                mov  di,f6_count
                push es
                mov  ax,fseg2
                mov  es,ax
                mov  word ptr es:[di+08000h],bp
                pop  es
                inc  word ptr f6_count
                inc  word ptr f6_count
                CMP WORD PTR F6_COUNT,1000
                JLE LESS14
                MOV BYTE PTR FX4,1
                JMP OOK
LESS14:           jmp  f13_no1

f15:
                CMP BYTE PTR FX5,0
                JNE L15

                inport 0dh
                and  al,00110000b
                cmp  al,00100000b
                jne  f13_no1
                JMP NR15
L15:              Inport 0dh
```

```
                and al,00110000b
                cmp al,00100000b
          ;     je f13_no1
                JE NY1
                JMP NR15
NY1:              JMP F13_NO1
NR15:
                mov di,f5_count
                push es
                mov ax,fseg2
                mov es,ax
                mov word ptr es:[di+04000h],bp
                pop es
                inc word ptr f5_count
                inc word ptr f5_count
                CMP WORD PTR F5_COUNT,1000
                JLE LESS15
                MOV BYTE PTR FX5,1
                JMP OOK

LESS15:           jmp f13_no1
f16:
                CMP BYTE PTR FX6,0
                JNE L16
H16:              inport 0dh
                and al,11000000b
                cmp al,10000000b
               ; jne f13_no1
                JNE FGH1
                JMP NR16
FGH1:             JMP F13_NO1


L16:              INPORT 0DH
                and al,11000000b
                cmp al,10000000b
                ;je f13_no1
                JE SSX16
                JMP NR16
     SSX16:       JMP F13_NO1

NR16:

                mov di,f4_count
                push es
                mov ax,fseg2
                mov es,ax
                mov word ptr es:[di+0000h],bp
                pop es
                inc word ptr f4_count
                inc word ptr f4_count

                CMP WORD PTR F4_COUNT,1000
                JLE LESS16      ;<1000
                MOV BYTE PTR FX6,1  ;>1000
                JMP OOK
LESS16:           jmp f13_no1
no_portd:         cmp al,19
               ; je  f19
                JE SM01
                JMP SO01
SM01:             JMP F19
SO01:
                cmp al,18
                je f18
f17:              CMP BYTE PTR FX7,0
                JNE L17
```

```
                inport 0eh
                and al,00000011b
                cmp al,00000010b
                jne f17_no1
                JMP NR17

L17:              inport 0eh
                and al,00000011b
                cmp al,00000010b
                je f17_no1

NR17:
                mov di,f3_count
                push es
                mov ax,fseg1
                mov es,ax
                mov word ptr es:[di+0c000h],bp
                pop es
                inc word ptr f3_count
                inc word ptr f3_count
                CMP WORD PTR F3_COUNT,1000
                JLE F17_NO1
                MOV BYTE PTR FX7,1
                JMP OOK

f17_no1:          dec bl
                ;jz end_op
                JZ SM02
                JMP LOAD_OP
SM02:             JMP END_OP
SM00:
              ; jmp load_op
f18:
                CMP BYTE PTR FX8,0
                JNE L18

H18:              inport 0eh
                and al,00001100b
                cmp al,00001000b
                jne f17_no1
                JMP NR18

L18:              INPORT 0EH
                and al,00001100b
                cmp al,00001000b
                JE F17_NO1

NR18:
                mov di,f2_count
                push es
                mov ax,fseg1
                mov es,ax
                mov word ptr es:[di+08000h],bp
                pop es
                inc word ptr f2_count
                inc word ptr f2_count
                CMP WORD PTR F2_COUNT,1000
                JLE LESS18
                MOV BYTE PTR FX8,1
OOK:
                MOV BYTE PTR QM_BUF,'A'
                POP BP
                POP DI
                POP SI
                POP DX
                POP CX
                POP BX
```

```
                POP AX
                JMP END_MES1
LESS18:           jmp  f17_no1

f19:
                CMP BYTE PTR FX9,0
                JNE L19
H19:              inport 0eh
                and  al,00110000b
                cmp  al,00100000b
                ;jne  f17_no1
                JNE JJ1
                JMP NR19
JJ1:             JMP F17_NO1
L19:
                inport 0eh
                and  al,00110000b
                cmp  al,00100000b
               ; je  f17_no1
                JE KL1
                JMP KL2
KL1:              JMP F17_NO1
KL2:
NR19:
                mov  di,f1_count
                push es
                mov  ax,fseg1
                mov  es,ax
                mov  word ptr es:[di+04000h],bp
                pop  es
                inc  word ptr f1_count
                inc  word ptr f1_count
                CMP WORD PTR F1_COUNT,1000
                JLE LESS19
                MOV BYTE PTR FX9,1
                JMP OOK

LESS19:           jmp  f17_no1
end_op:
                pop  bp
                pop  di
                pop  si
                pop  dx
                pop  cx
                pop  bx
                pop  ax
;/********************** test output sescess ***************
                cmp  word ptr bp,max_scan
                je   cancel
                inc  bp
                call percent
                jmp  main_loop
cancel:
                MOV BYTE PTR FX1,0

                CMP WORD PTR F1_COUNT,1
                JL VX2
                INC BYTE PTR FX1
        VX2:      CMP WORD PTR F2_COUNT,1
                JL VX3
                INC BYTE PTR FX1
        VX3:      CMP WORD PTR F3_COUNT,1
                JL VX4
                INC BYTE PTR FX1
        VX4:      CMP WORD PTR F4_COUNT,1
                JL VX5
                INC BYTE PTR FX1
        VX5:      CMP WORD PTR F5_COUNT,1
```

```
                JL VX6
                INC BYTE PTR FX1
        VX6:      CMP WORD PTR F6_COUNT,1
                JL VX7
                INC BYTE PTR FX1
        VX7:      CMP WORD PTR F7_COUNT,1
                JL VX8
                INC BYTE PTR FX1
        VX8:      CMP WORD PTR F8_COUNT,1
                JL VX9
                INC BYTE PTR FX1
        VX9:      CMP WORD PTR F9_COUNT,1
                JL VXC
                INC BYTE PTR FX1

        VXC:      CMP BYTE PTR FX1,2
                JL  CREXXX
                MOV BYTE PTR QM_BUF,'1'

    CREXXX:
                call write_file
                ret
pal20 endp
;/*************************************************************
public delay
delay proc near
                push ax
                mov  ax,01ffffh
del:             dec  ax
                jnz  del
                pop  ax
                ret
delay endp
;/*************************************************************
write_file proc near
                mov  ax,dseg
                mov  es,ax
                lea  di,qm_buf+2
                mov  ax,0
                mov  al,value1
                call hex2dec
                mov  ax,0
                mov  al,value2
                call hex2dec
                lea  dx,qm
                call create
                mov  bx,ax
                lea  dx,qm_buf
                mov  cx,7
                call write_buf
                call close
;/********************** write file funct*.dat *************
                lea  si,op_hex
                mov  al,value2
                mov  byte ptr count,al
                lea  dx,funct01
write_new:          call create
                mov  bx,ax
                lodsb
                cmp  al,12
                jg   write_portd_far
write_portc:        cmp  al,12
                je   write_12
write_11:           push ds
                push si
                mov  bp,f9_count
                cmp  bp,0
                jne  nothing_11
```

```
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far
                add  dx,12
                jmp  write_new
write_portd_far:    jmp  write_portd
nothing_11:         mov  ax,fseg3
                mov  ds,ax
                mov  si,4000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far
                add  dx,12
                jmp  write_new
write_12:         push ds
                push si
                mov  bp,f8_count
                cmp  bp,0
                jne  nothing_12
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far
                add  dx,12
                jmp  write_new
nothing_12:         mov  ax,fseg3
                mov  ds,ax
                mov  si,0000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far
                add  dx,12
                jmp  write_new
end_write_op_far:     jmp  end_write_op
write_portd:        cmp  al,16
                jg   write_porte_far
                cmp  al,16
                je   write_16_far
                cmp  al,15
                je   write_15_far
                cmp  al,14
                je   write_14
write_13:           push ds
```

```
                push si
                mov bp,f7_count
                cmp bp,0
                jne nothing_13
                call write_space
                pop si
                pop ds
                dec byte ptr count
                jz end_write_op_far
                add dx,12
                jmp write_new
write_16_far:       jmp write_16
write_15_far:       jmp write_15
nothing_13:         mov ax,fseg2
                mov ds,ax
                mov si,0c000h
                call convert_hex2dec
                mov cx,di
                sub cx,1
                mov ax,change
                mov ds,ax
                push dx
                mov dx,0
                call write_buf
                call close
                pop dx
                pop si
                pop ds
                dec byte ptr count
                jz end_write_op_far
                add dx,12
                jmp write_new
write_porte_far:    jmp write_porte
write_14:           push ds
                push si
                mov bp,f6_count
                cmp bp,0
                jne nothing_14
                call write_space
                pop si
                pop ds
                dec byte ptr count
                jz end_write_op_far2
                add dx,12
                jmp write_new
nothing_14:         mov ax,fseg2
                mov ds,ax
                mov si,8000h
                call convert_hex2dec
                mov cx,di
                sub cx,1
                mov ax,change
                mov ds,ax
                push dx
                mov dx,0
                call write_buf
                call close
                pop dx
                pop si
                pop ds
                dec byte ptr count
                jz end_write_op_far2
                add dx,12
                jmp write_new
write_15:           push ds
                push si
                mov bp,f5_count
                cmp bp,0
```

```
                jne  nothing_15
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far2
                add  dx,12
                jmp  write_new
end_write_op_far2:    jmp  end_write_op
nothing_15:         mov  ax,fseg2
                mov  ds,ax
                mov  si,4000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far2
                add  dx,12
                jmp  write_new
write_16:           push ds
                push si
                mov  bp,f4_count
                cmp  bp,0
                jne  nothing_16
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far2
                add  dx,12
                jmp  write_new
nothing_16:         mov  ax,fseg2
                mov  ds,ax
                mov  si,0000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far2
                add  dx,12
                jmp  write_new
write_porte:        cmp  al,19
                je   write_19_far
                cmp  al,18
                je   write_18
write_17:           push ds
                push si
                mov  bp,f3_count
                cmp  bp,0
                jne  nothing_17
```

```
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far3
                add  dx,12
                jmp  write_new
write_19_far:       jmp  write_19
nothing_17:         mov  ax,fseg1
                mov  ds,ax
                mov  si,0c000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op_far3
                add  dx,12
                jmp  write_new
end_write_op_far3:    jmp  end_write_op
write_18:           push ds
                push si
                mov  bp,f2_count
                cmp  bp,0
                jne  nothing_18
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op
                add  dx,12
                jmp  write_new
nothing_18:         mov  ax,fseg1
                mov  ds,ax
                mov  si,8000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz   end_write_op
                add  dx,12
                jmp  write_new
write_19:           push ds
                push si
                mov  bp,f1_count
                cmp  bp,0
                jne  nothing_19
                call write_space
                pop  si
                pop  ds
                dec  byte ptr count
```

```
                jz  end_write_op
                add  dx,12
                jmp  write_new
nothing_19:         mov  ax,fseg1
                mov  ds,ax
                mov  si,4000h
                call convert_hex2dec
                mov  cx,di
                sub  cx,1
                mov  ax,change
                mov  ds,ax
                push dx
                mov  dx,0
                call write_buf
                call close
                pop  dx
                pop  si
                pop  ds
                dec  byte ptr count
                jz  end_write_op
                add  dx,12
                jmp  write_new
end_write_op:       call cursor_on
                outport 11h,00h
                mov  ax,3
                int  10h
                mov  ah,9
                lea  dx,terminate
                int  21h
                mov  ax,4c00h
                int  21h
                ret
write_file endp
;/*************************************************************
extrn cursor_on:near
public convert_hex2dec
convert_hex2dec proc near
                push ax
                push bx
                push cx
                push dx
                push si
                push ds
                push es
                push bp
                add  bp,si
                mov  ax,change
                mov  es,ax
                mov  di,0
convert_again:      lodsw
                call hex2dec
                cmp  si,bp
                jge  stop_convert
                jmp  convert_again
stop_convert:       pop  bp
                pop  es
                pop  ds
                pop  si
                pop  dx
                pop  cx
                pop  bx
                pop  ax
                ret
convert_hex2dec endp
;/*************************************************************
public write_space
write_space proc near
                push ax
```

```
                push bx
                push cx
                push dx
                push si
                push ds
                push es
                push bp
                lea  dx,space_buf
                mov  cx,1
                call write_buf
                call close
                pop  bp
                pop  es
                pop  ds
                pop  si
                pop  dx
                pop  cx
                pop  bx
                pop  ax
                ret
write_space endp
;/*************************************************************
public create
create proc near
                push cx
                mov  cx,20h
                mov  ah,3ch
                int  21h
                pop  cx
                ret
create endp
public write_buf
write_buf proc near
                push ax         ;cx=counter
                mov  ah,40h     ;ds:dx=address
                int  21h
                pop  ax
                ret
write_buf endp
public close
close proc near
                push ax
                mov  ah,3eh
                int  21h
                pop  ax
                ret
close endp
;/*************************************************************
public percent
extrn write_char_n_time:near
percent proc near
                push ax
                push bx
                push cx
                push dx
                push si
                push bp
                cmp  byte ptr flag,0ffh
                jne  more_per
                jmp  end_per
more_per:           mov si,0
                cmp  word ptr maxchar,0
                jne  max_not0
                jmp  per100
max_not0:           cmp  byte ptr flag,52
                je   per100
                mov  cx,0
                mov  cl,flag
```

```
addsi:          add  si,maxchar
                loop addsi
                cmp  si,bp
                jl   write_1per
end_per:          pop  bp
                pop  si
                pop  dx
                pop  cx
                pop  bx
                pop  ax
                ret
write_1per:       mov  al,0b1h
                mov  bl,7eh
                call write_char
                inc  byte ptr flag
                jmp  end_per
per100:           mov  dx,140bh
                call mov_cursor
                mov  cx,58
                mov  bl,7eh
                mov  al,0b1h
                call write_char_n_time
                mov  byte ptr flag,0ffh
                jmp  end_per
percent endp
;/*************************************************************
include hex2dec.inc
cseg ends
dseg segment public
public port1,port2,port3,max_scan,count,space_buf,qm
public f1_count,f2_count,f3_count,f4_count,f5_count
public f6_count,f7_count,f8_count,f9_count,f10_count
public funct01,funct02,funct03,funct04,funct05
public funct06,funct07,funct08,funct09,funct10
public qm_buf
extrn value1:byte,value2:byte,lp_hex:byte,op_hex:byte
public maxchar,flag
flag            db 1 dup (01)
public terminate
terminate         db "Terminate normal..!$"
maxchar           dw ?
port1           db ?
port2           db ?
port3           db ?
max_scan          dw ?
FX1             DB 0
FX2             DB 0
FX3             DB 0
FX4             DB 0
FX5             DB 0
FX6             DB 0
FX7             DB 0
FX8             DB 0
FX9             DB 0
FX10            DB 0
f1_count          dw ?
f2_count          dw ?
f3_count          dw ?
f4_count          dw ?
f5_count          dw ?
f6_count          dw ?
f7_count          dw ?
f8_count          dw ?
f9_count          dw ?
f10_count         dw ?
count           db ?
space_buf         db " ",0
qm              db "qm.cfg",0
```

```
funct01             db "funct01.dat",0
funct02             db "funct02.dat",0
funct03             db "funct03.dat",0
funct04             db "funct04.dat",0
funct05             db "funct05.dat",0
funct06             db "funct06.dat",0
funct07             db "funct07.dat",0
funct08             db "funct08.dat",0
funct09             db "funct09.dat",0
funct10             db "funct10.dat",0
qm_buf              db "1 "
                    db 10 dup(20h)
dseg ends
;/*************************************************************
fseg1 segment public
        db 0ffffh dup(?)
fseg1 ends
fseg2 segment public
        db 0ffffh dup(?)
fseg2 ends
fseg3 segment public
        db 0ffffh dup(?)
fseg3 ends
change segment public
        db 0ffffh dup(?)
change ends
end
```

QM.C

```
#include <stdlib.h>
#include <conio.h>
#include <alloc.h>
#include <stdio.h>
#include <ctype.h>
/* use XMS for keep TERM all */
#include "xms.c"

#define  MaxPI          10000
#define  MaxList       470000
#define  MaxMinTerm     32000
#define  CharFile_R     65536
#define  CharFile_CFG      10
#define  MaxFunc            1

typedef  unsigned int   word;
typedef  unsigned char  byte;
typedef  unsigned long  dword;

void ProcInput();
void CFGinput();
void SetTerm();
void List();
void SetPI();
void Select();
void EqOut();
int InputFile(word *Buffer,word Range);
void SetNewTerm(word IDs,word IDd,word *IDnew,char IDx);
void TermShowData(word ID);
void FuncShowData(word ID);
void FindMin(word ID,word *BuffMin);
dword pow(dword x,dword y);
int allocterm1(struct TERM1 *term1);
struct record far *Term1(struct TERM1 *term1,int index);
void freeterm1(struct TERM1 *term1);
int allocterm2(struct TERM2 *term2);
struct record2 far *Term2(struct TERM2 *term2,int index);
void freeterm2(struct TERM2 *term2);
int allocterm3(struct TERM3 *term3);
struct record3 far *Term3(struct TERM3 *term3,int index);
void freeterm3(struct TERM3 *term3);
int allocterm4(struct TERM4 *term4);
struct record4 far *Term4(struct TERM4 *term4,int index);
void freeterm4(struct TERM4 *term4);
int allocPi(struct PIV *piv);
struct record5 far *PI(struct PIV *piv,int index);
void freePi(struct PIV *piv);

/*     Gobal Var      */
byte   MSB;
byte   Minterm;
byte   Output;
word   Func[MaxFunc][MaxMinTerm];
word   FuncEss[MaxFunc][MaxMinTerm];
word   DoCar[MaxFunc][MaxMinTerm];

/*     List  Var      */

struct record  { word Data;
                 word CutOut;
               };

struct TERM1 {
        struct record far *term1;
        int TermNum;
```

```
        int handle;
        int oldpage;
        };

struct record2 {
                word Func;
                word DoCar;
                word Flags;
            };

struct  TERM2 {
        struct record2 far *term2;
        int Term2Num;
        int handle;
        int oldpage;
        };

struct record3 {
                word Used;
                word List;
                word Group;
            };
struct TERM3 {
        struct record3 far *term3;
        int Term3Num;
        int handle;
        int oldpage;
        };

struct record4 {
                word Min1;
                word Min2;
            };
struct TERM4 {
        struct record4 far *term4;
        int Term4Num;
        int handle;
        int oldpage;
        };

word   TopData;

/*      PI Var      */
struct record5 {
                word Index;
                byte Ci;
                byte Essential;
                word Min[256];
                };
/*struct record5 PI[MaxPi];*/
struct PIV {
        struct record5 far *piv;
        int PiNum;
        int handle;
        int oldpage;
        };

word   TopPi;
word   SelectPi[MaxPi];

FILE *fp_r,*fp_cfg,*fp_eq;
int    loop;
int    outofrange=0;
int    cycle=0;
char   bar[] = { 45,92,124,47 };
struct TERM1 term1;
struct TERM2 term2;
struct TERM3 term3;
```

```
struct TERM4 term4;
struct PIV pi;

char *funct_file[] = {"","FUNCT01.DAT","FUNCT02.DAT","FUNCT03.DAT",
                      "FUNCT04.DAT","FUNCT05.DAT","FUNCT06.DAT",
                      "FUNCT07.DAT","FUNCT08.DAT","FUNCT09.DAT",
                      "FUNCT10.DAT"
                     };

void main()
{
  int k;
  /* alloc extended memory for term*/
  if(!XMSinstalled())
  {
        printf("This program use Extended memory and HIMEM.SYS !\n");
        exit(1);
  }

  term1.TermNum = MaxList;
  if(!allocterm1(&term1)) {
    printf("Error Can't Alloc Memory for TERM1\n");
    exit(-1);
  }

  term2.Term2Num = MaxList;
  if(!allocterm2(&term2)) {
    printf("Error Can't Alloc Memory for TERM2\n");
    freeterm1(&term1);
    exit(-1);
  }

  term3.Term3Num = MaxList;
  if(!allocterm3(&term3)) {
    printf("Error Can't Alloc Memory for TERM3\n");
    freeterm1(&term1);
    freeterm2(&term2);
    exit(-1);
  }

  term4.Term4Num = MaxList;
  if(!allocterm4(&term4)) {
    printf("Error Can't Alloc Memory for TERM4\n");
    freeterm1(&term1);
    freeterm2(&term2);
    freeterm3(&term3);
    exit(-1);
  }

  pi.PiNum = MaxPi;
  if(!allocPi(&pi)) {
    printf("Error Can't Alloc Memory for PI\n");
    freeterm1(&term1);
    freeterm2(&term2);
    freeterm3(&term3);
    freeterm4(&term4);
    exit(-1);
  }

  CFGinput();
  clrscr();
  for(loop=1;loop<=Output;loop++)
        {
        ProcInput();
        if(outofrange==1)
          {
        /*   gotoxy(30,12);*/
          printf("\aData file out of range !\n");
```

```
/*          gotoxy(32,13);*/
            printf("Can't analysis...\n");
            getch();
            exit(-1);
           }
         SetTerm();
         List();
         SetPI();
         Select();
         EqOut();
        }
  system("READ QM.EQ");
  freeterm1(&term1);
  freeterm2(&term2);
  freeterm3(&term3);
  freeterm4(&term4);
  freePi(&pi);
}

/********* Input Data *********/

void ProcInput()
{
int i,j;
 char Name_r[13];
 char KeyBuff[255];
dword Range;

Range=pow(2,MSB)-1;
 if((fp_r=fopen(funct_file[loop],"r"))==NULL)
    {
    gotoxy(30,12);
    printf("\aCan't open data file [%s] !\n",funct_file[loop]);
    getch();
         exit(-1);
    }

for(i=1;i<=MaxFunc;i++)
    {
    j=1;
    while(j!=0)
         {
         printf("\n F%d = ( ",loop);
         j=InputFile(Func[i],Range);
         if(j!=0) { printf("\n Data Out of Range (Max:%d)",Range);  outofrange=1; }
         }
    }
}

void CFGinput()
{
char Num;
 char far *ptr_cfg,*keystr;
char Name_cfg[13];
 char Error=0;

 keystr = (char far *)farmalloc(CharFile_R);
 ptr_cfg=keystr;
if((fp_cfg=fopen("QM.CFG","r"))==NULL)
 {
  gotoxy(25,12);
  printf("\aCan't open config file [QM.CFG] !\n");
  getch();
  exit(-1);
 }
 do
 {
   *ptr_cfg=getc(fp_cfg);
}
```

```
    ptr_cfg++;
   }
  while(*(ptr_cfg-1)!=EOF);
 Num=0,MSB=0;
 ptr_cfg=keystr;
 Minterm=*ptr_cfg;
 if(Minterm!='I' && Minterm!='A')    Error=1;
 ptr_cfg+=2;
 if((*ptr_cfg>=48)&&(*ptr_cfg<=57))
    {
    Num=1;
    while(Num!=0)
          {
          MSB*=10;
          MSB+=*ptr_cfg-48;
          if(MSB>16)  Error=1;
          ptr_cfg++;
          if(!((*ptr_cfg>=48)&&(*ptr_cfg<=57)))
            {
            Num=0;
            ptr_cfg++;
            }
          }
    Num=1;
    while(Num!=0)
          {
          Output*=10;
          Output+=*ptr_cfg-48;
          if(Output>10)  Error=1;
          ptr_cfg++;
          if(!((*ptr_cfg>=48)&&(*ptr_cfg<=57)))
             {
             Num=0;
             }
          }

    }
 fclose(fp_cfg);
 if(Error==1)
    {
    gotoxy(20,12);
    printf("\aFile config missmach or config out of range !");
    getch();
    exit(-1);
    }
}

 int InputFile(word *Buffer,word Range)
 {
 char Num;
 char far *ptr,*keystr;
 char Error=0;
 word Data;

 keystr = (char far *)farmalloc(CharFile_R);
 ptr=keystr;
 do
   {
   *ptr=getc(fp_r);
   putchar(*ptr);
   ptr++;
   }
 while(*(ptr-1)!=EOF);
 *ptr=0;
 ptr=keystr;
 *Buffer=0;ptr--;
 do
   {
```

```
  Num=0;Data=0;
  ptr++;
  if((*ptr>=48)&&(*ptr<=57))
     {
     Num=1;
     while(Num!=0)
           {
           Data*=10;
           Data+=*ptr-48;
           if(Data>Range) {
             Error=1;
             printf("\nError = 1 : because Data = %u,range = %u\n",Data,Range);
             /* xxxxx */
           }
           ptr++;
           if(!((*ptr>=48)&&(*ptr<=57)))
              {
              Num=0;
              (*Buffer)++;
              *(Buffer+*Buffer)=Data;
              if(Data>Range)   Error=1;
              }
           }
     }
  }
 while(*ptr!=0);
 fclose(fp_r);
 printf(")");
 farfree(keystr);
 printf("\nProcess Function ->[%d] in Progress Please Wait",loop);
 return(Error);
}

void InputPrint()
{
word i,j;

for(i=1;i<=MaxFunc;i++)
   for(j=0;j<=Func[i][0];j++)
      {
      printf("\n Func[");
      printf("z%d][",i);printf("%d] = ",j);
      printf("%d ;",Func[i][j]);
      }
 for(i=1;i<=MaxFunc;i++)
   for(j=0;j<=DoCar[i][0];j++)
      {
      printf("\n DoCar[");
      printf("%d][",i);printf("%d] = ",j);
      printf("%d ;",DoCar[i][j]);
      }
   printf("\n");
}

/************ Util *************/

Counter(word k)
{
char i=1,j=0;
word Bit=1;
for(i=1;i<=16;i++)
   {
   if((Bit&k)==Bit) j++;
   Bit<<=1;
   }
return(j);
}
```

```
void FindBitOn(dword k,word *ptr)
{
char y;
   *ptr=0;
   for(y=1;y<=32;y++)
     {
     if(((k>>(y-1))&1)==1)
            {
            (*ptr)++;
            *(ptr+*ptr)=y;
            }
     }
}

CmpBitOn(word x,char y)
{
word lbx=1;
char i;
for(i=1;i<y;i++) lbx<<=1;
if((x&lbx)==lbx)
          return(1);
else
          return(0);
}

Diff1Bit(word x,word y,word z)
{
word q[2];
x^=y;x&=z;
if(Counter(x)==1)
   {
   FindBitOn(x,&q[0]);
   return(q[1]);
   }
return(0);
}

void SetBitOn(word *x,int i)
{
word lbx=1;
lbx<<=(i-1);
*x|=lbx;
}

/************ LIST *************/

void SortData(word *ptr)
{
int  j,z,q;
word b;
for(j=2;j<=*ptr;j++)
  for(z=j;z<=*ptr;z++)
     if(*(ptr+j-1)>*(ptr+z))
          { b=*(ptr+z);*(ptr+z)=*(ptr+j-1);*(ptr+j-1)=b; }
j=0;
do
  {
  j++;
  if(j+1<=*ptr)
    if(*(ptr+j)==*(ptr+j+1))
      {
      for(z=j+1;z<=*ptr;z++)
           *(ptr+z)=*(ptr+z+1);
      (*ptr)--;j--;
      }
  }
while(j<*ptr);
}
```

```
void SetTerm()
{
word    z;
byte    i,j;
TopData=0;
printf("\n");
for(i=1;i<=MaxFunc;i++)
    {
    SortData(&Func[i][0]);
    SortData(&DoCar[i][0]);
    }
for(i=1;i<=MaxFunc;i++)
    {
    for(j=1;j<=Func[i][0];j++)
            {
            for(z=1;z<=TopData;z++)
                {
                if(cycle>3) cycle=0;
                printf("\r%c",bar[cycle++]);
                if(Term1(&term1,z)->Data==Func[i][j])
                        {
                        SetBitOn(&(Term2(&term2,z)->Func),i);
                        goto l1;
                        }
                }
            TopData++;
            Term1(&term1,TopData)->Data=Func[i][j];
            Term2(&term2,TopData)->Func=0;
            Term2(&term2,TopData)->DoCar=0;
            SetBitOn(&(Term2(&term2,TopData)->Func),i);
            l1:;
            }
    }

for(i=1;i<=MaxFunc;i++)
    {
    for(j=1;j<=DoCar[i][0];j++)
            {
            for(z=1;z<=TopData;z++)
                if(Term1(&term1,z)->Data==DoCar[i][j])
                        {
                        SetBitOn(&(Term2(&term2,z)->DoCar),i);
                        goto l2;
                        }
            TopData++;
            Term1(&term1,TopData)->Data=DoCar[i][j];
            Term2(&term2,TopData)->Func=0;
            Term2(&term2,TopData)->DoCar=0;
            SetBitOn(&Term2(&term2,TopData)->DoCar,i);
            l2:;
            }
    }
}

void List()
{
word    i;
/*struct record huge *ptr=Term1(&term1,0);*/ /* nonono */
word    IDs,IDd,IG;
word    IDnew;
int     IDL=1;
word    NewTop,Bottom;
word    BUFF1=1,BUFF2=0;
for(i=1;i<=MSB;i++)
    BUFF2|=(BUFF1<<(i-1));
for(i=1;i<=TopData;i++)
    {
```

```
    Term1(&term1,i)->CutOut=BUFF2;
    Term2(&term2,i)->Flags =Term2(&term2,i)->Func|Term2(&term2,i)->DoCar;
    Term3(&term3,i)->Group =Counter(Term1(&term1,i)->Data);
    Term3(&term3,i)->Used  =0;
    Term3(&term3,i)->List  =1;
    }
 Bottom=1;
 NewTop=TopData+1;
 IDnew =TopData+1;
 l1:;
 for(IDs=Bottom;IDs<=TopData;IDs++)
   {
   if(cycle>3) cycle=0;
   printf("\r%c",bar[cycle++]);
   IG=Term3(&term3,IDs)->Group+1;
   for(IDd=Bottom;IDd<=TopData;IDd++)
     {
     if(IDL==Term3(&term3,IDd)->List)
          if(IG==Term3(&term3,IDd)->Group)
/*          if((ptr+IDs)->CutOut==(ptr+IDd)->CutOut)*/
            if(Term1(&term1,IDs)->CutOut==Term1(&term1,IDd)->CutOut)
              {
/*            BUFF1=Diff1Bit((ptr+IDs)->Data,(ptr+IDd)->Data,(ptr+IDs)->CutOut);*/
                    BUFF1=Diff1Bit(Term1(&term1,IDs)->Data,Term1(&term1,IDd)->Data,Term1(&term1,ID
>CutOut);
              if(BUFF1!=0)
                    {
                    SetNewTerm(IDs,IDd,&IDnew,BUFF1);
                    IDnew++;
                    }
              }
     }
   }
 IDL++;
 Bottom=NewTop;
 NewTop=IDnew-1;
 if(NewTop!=TopData)
   {
   TopData=NewTop;
   goto l1;
   }
}

void SetNewTerm(word IDs,word IDd,word *IDnew,char IDx)
{
word    BUFF1;
word    i;

   Term1(&term1,*IDnew)->Data=Term1(&term1,IDs)->Data;
   /* Set CutOut in NewTerm            */
   /* CutOut = each Bit if 1 have 0 not  */
   BUFF1=1;
   BUFF1<<=(IDx-1);
   Term1(&term1,*IDnew)->CutOut=(Term1(&term1,IDs)->CutOut)&~BUFF1;
   Term2(&term2,*IDnew)->Func =Term2(&term2,IDs)->Func|Term2(&term2,IDd)->Func;
   Term2(&term2,*IDnew)->DoCar=Term2(&term2,IDs)->DoCar;
   /*  Set Flags   */
   BUFF1=Term2(&term2,IDs)->Flags&Term2(&term2,IDd)->Flags;
   if(BUFF1==0)
            {
            (*IDnew)--;
            goto exit11;
            }
   Term2(&term2,*IDnew)->Flags=BUFF1;
   if((BUFF1^Term2(&term2,IDs)->Flags)==0) Term3(&term3,IDs)->Used=1;
   if((BUFF1^Term2(&term2,IDd)->Flags)==0) Term3(&term3,IDd)->Used=1;
   Term3(&term3,*IDnew)->Used =0;
   Term4(&term4,*IDnew)->Min1 =IDs;
```

```
    Term4(&term4,*IDnew)->Min2 =IDd;
    Term3(&term3,*IDnew)->List =Term3(&term3,IDs)->List+1;
    Term3(&term3,*IDnew)->Group=Term3(&term3,IDs)->Group;
    for(i=1;i<=*IDnew-1;i++)
      {
      if(Term3(&term3,i)->List==Term3(&term3,*IDnew)->List)
           {
           if(Term1(&term1,i)->Data==Term1(&term1,*IDnew)->Data)
            if(Term1(&term1,i)->CutOut==Term1(&term1,*IDnew)->CutOut)
                  if((Term1(&term1,i)->Data&Term1(&term1,i)->CutOut)==(Term1(&term1,*IDnew
>Data&Term1(&term1,*IDnew)->CutOut))
                (*IDnew)--;
           }
      }
    exit11:;
}

void Printterm()
{
word i;
for(i=1;i<=TopData;i++)
    {
    printf("\n");
    printf("\n term[%d].Data  = %d;",i,Term1(&term1,i)->Data);
    printf(  " term[%d].CutOut= 0x%X;",i,Term1(&term1,i)->CutOut);
    printf("\n term2[%d].Func  = 0x%X;",i,Term2(&term2,i)->Func);
    printf(  " term2[%d].DoCar = 0x%X;",i,Term2(&term2,i)->DoCar );
    printf(  " term2[%d].Flags = 0x%X;",i,Term2(&term2,i)->Flags);
    printf("\n term3[%d].Used  = %d;",i,Term3(&term3,i)->Used);
    printf("\n term4[%d].Min1  = %d;",i,Term4(&term4,i)->Min1);
    printf(  " term4[%d].Min2  = %d;",i,Term4(&term4,i)->Min2);
    printf("\n term3[%d].List  = %d;",i,Term3(&term3,i)->List);
    printf("\n term3[%d].Group = %d;",i,Term3(&term3,i)->Group);
    }
}

void ListOut()
{
word i;
byte j;
for(j=1;j<=MSB;j++)
    {
    printf("\n\nLIST %d",j);
    for(i=1;i<=TopData;i++)
      if(Term3(&term3,i)->List==j)
            {
            printf("\n   ");
            TermShowData(i);printf("  ");FuncShowData(i);
            if(Term3(&term3,i)->Used==1) printf("  û");
            }
    }
}

void FuncShowData(word ID)
{
word i;
word lbx;
lbx=0x0001;
for(i=1;i<=16;i++)
    {
    if(Counter(Term2(&term2,ID)->Flags & lbx)==1)
       printf(" F%d",i);
    lbx<<=1;
    }
}

void TermShowData(word ID)
{
```

```
char STerm[16];
byte i;
word Ibx;
Ibx=0x8000;
for(i=1;i<=16;i++)
  {
  if(i>(16-MSB))
        {
        if((Term1(&term1,ID)->CutOut & Ibx)==0)
           STerm[i-17+MSB]='-';
        else
           {
           if((Term1(&term1,ID)->Data & Ibx)==0)
                STerm[i-17+MSB]='0';
           else
                STerm[i-17+MSB]='1';
           }
        }
  Ibx >>=1;
  }
STerm[MSB]=0;
printf("%s",STerm);
}

/************ PI *************/
void SetPI()
{
word i,j,p,e,t,count;
TopPI=0;
for(i=1;i<=TopData;i++)
  if(Term3(&term3,i)->Used==0&&Term2(&term2,i)->Func!=0)
    {
    TopPI++;
    if(TopPI>MaxPI) printf("\n PI Error");
/*     (PI+TopPI)->Index=i;*/
    PI(&pi,TopPI)->Index=i;
    PI(&pi,TopPI)->Cl=Counter(Term1(&term1,i)->CutOut);
        FindMin(PI(&pi,TopPI)->Index,(PI(&pi,TopPI)->Min));
    PI(&pi,TopPI)->Essential=0;
    }
 for(i=1;i<=MaxFunc;i++)
   for(j=1;j<=Func[i][0];j++)
     {
     if(cycle>3) cycle=0;
     printf("\r%c",bar[cycle++]);
     count=0;FuncEss[i][j]=0;
     for(p=1;p<=TopPI;p++)
          {
          for(t=1;t<=PI(&pi,p)->Min[0];t++)
            {
            if((Func[i][j]==PI(&pi,p)->Min[t])&&(1==CmpBitOn(Term2(&term2,PI(&pi,p)->Index)->Func,i)))
                {
                e=p;count++;
                if(count>1) goto ex1;
                }
            }
          }
     PI(&pi,e)->Essential=1;
     FuncEss[i][j]=1;
     ex1:;
     }
}

void Select()
{
word i,j,y,z,w;
word BuffPI;
```

```
char BuffCi1=0   ,BuffCi2;
char BuffCg1     ,BuffCg2;
char BuffCh=0;
byte BuffCn1=0xff,BuffCn2;
byte BuffCn;
byte FirstSet=1;

dword x;                /* Gobal Select Pi */
dword t;                /* Func  Select Pi */

word   SelectPiFunc[MaxPi];
dword MaxLoop;
int  px,py,pw;

/*   Set Style    */

BuffCg1=TopPi;
for(z=1;z<=TopPi;z++)  BuffCi1+=Pi(&pi,z)->Ci;
printf("\n0%%");
px=wherex()-2;py=wherey();
MaxLoop=pow(2,TopPi)-1;
pw=1;
for(x=1;x<=MaxLoop;x++)
   {

   /*   Show % Item   */

   if(pw==1)
         { pw=0;px=wherex()-2;py=wherey(); }
   gotoxy(px,py-1);
   cprintf("%d%%",(x*100)/MaxLoop);

   FindBitOn(x,&SelectPi[0]);

   /*    Check Cover ALL    */
   for(i=1;i<=MaxFunc;i++)
     for(j=1;j<=Func[i][0];j++)
           {
           z=1;w=1;
           while( (Func[i][j]!=Pi(&pi,*(SelectPi+z))->Min[w]) || (CmpBitOn(Term2(&term2,Pi(&pi,*(SelectPi+z
>Index)->Flags,i)!=1))
                 {
                 if((z==*SelectPi)&&(w==Pi(&pi,*(SelectPi+z))->Min[0]))
                       goto lb1;
                 else
                       if(w==Pi(&pi,*(SelectPi+z))->Min[0])
                          {
                          w=0;z++;
                          }
                 w++;
                 }
           }
   /* Counter Ci Cg Cn in Pi Select */
   BuffCi2=0;
   for(z=1;z<=*SelectPi;z++) BuffCi2+=Pi(&pi,*(SelectPi+z))->Ci;
   BuffCg2=Counter(x);
   if(BuffCg1<BuffCg2)
         goto lb1;
   else
         if(BuffCg1==BuffCg2)
            if(BuffCi1<BuffCi2) goto lb1;
         else
            {
            if(BuffCi1==BuffCi2&&MaxFunc>1)
                  {
                  /* Select Pi in Function
                     BuffCn1 Store ;BuffCn2 var
```

```
                    SelectPiFunc;count binary t;
                    BuffCn;                    */
                  BuffCn2=0;BuffCn=0xFF;
                  for(i=1;i<=MaxFunc;i++)
                    {
                    for(t=1;t<=pow(2,*SelectPi)-1;t++)
                      {
                      for(j=1;j<=Func[i][0];j++)
                          {
                          FindBitOn(t,&SelectPiFunc[0]);
                          z=1;w=1;
                          while(  (Func[i][j]!=Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Min[w])
(CmpBitOn(Term2(&term2,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Flags,i)!=1))
                            {
                            if((z==*SelectPiFunc)&&(w==Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Min[0]))
                                goto lb2;
                            else
                                if(w==Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Min[0])
                                    {w=0;z++;}
                            w++;
                            }
                          }
                      if(BuffCn>Counter(t)) BuffCn=Counter(t);
                      lb2:;
                      }
                    BuffCn2+=BuffCn;
                    /* Pass BuffCn2 is Number Total Func Min */
                    }
                  if(BuffCn1<BuffCn2) goto lb1;
                  }
              }
    BuffPi=x;BuffCh=1;
    BuffCg1=BuffCg2;
    BuffCl1=BuffCl2;
    BuffCn1=BuffCn2;
    if(FirstSet==1) FirstSet=0;
    lb1:;
    }
    if(BuffCh==1)
          FindBitOn(BuffPi,&SelectPi[0]);
    else
          {
          SelectPi[0]=TopPi;
          for(z=1;z<=TopPi;z++)
            SelectPi[z]=z;
          }
}

void FindMin(word ID,word *BuffMin)
{
#define MAXStack 3072
word ID1,ID2,ID3;
word IDStack,*Stack;
byte i,j;

Stack=(word*)malloc(MAXStack);
if(Stack==0) cputs("Memory Error");
IDStack=1;Stack[0]=0;
ID3=ID;
*BuffMin=0;
while((Term3(&term3,ID)->List!=1)||(IDStack!=0))
  {
  ID1=Term4(&term4,ID)->Min1;
  ID2=Term4(&term4,ID)->Min2;
  if(Term3(&term3,ID)->List==1)
    {
    (*BuffMin)++;*(BuffMin+*BuffMin)=Term1(&term1,ID)->Data;
    IDStack--;
```

```
    if((int)IDStack<0)  {printf("Stack Error Exit");exit(1);}
    If(IDStack!=0) ID=Stack[IDStack];
    }
  else
    {
    Stack[IDStack]=ID2;
    IDStack++;
    if(IDStack>MAXStack) {printf("Stack Error Exit");exit(1);}
    ID=ID1;
    }
  }
 if(Term3(&term3,ID3)->List>2)
  {
  for(i=*BuffMin;i>1;i--)
    for(j=1;j<i;j++)
        if(*(BuffMin+j)>*(BuffMin+i))
          {
          ID1=*(BuffMin+j);
          *(BuffMin+j)=*(BuffMin+i);
          *(BuffMin+i)=ID1;
          }
  }
 free(Stack);
 }

 PrintPi()
 {
 word i,j;
 word *BuffMin;

 for(i=1;i<=TopPi;i++)
  {
  printf("\n PI[%d]",i);
  printf("\n  Index= %d ",PI(&pi,i)->Index);
  printf("\n  CI   = %d ",PI(&pi,i)->CI);
  printf("\n  Func = %x \n",Term2(&term2,(PI(&pi,i)->Index))->Func);
  BuffMin=(word*)malloc(100);
  if(BuffMin==0) printf("Memory Error");
  FindMin((PI(&pi,i)->Index),BuffMin);
  for(j=1;j<=*BuffMin;j++)
    printf(" %d",*(BuffMin+j));
  free(BuffMin);
  if(*BuffMin==0xffff) printf("Memory Error");
  }
  return 0;
 }

 PiOut()
 {
 word i,j;
 word *BuffMin;

 for(i=1;i<=TopPi;i++)
  {
  printf("\n\n PI[%d]",i);
  printf(  "\n      Data: ");TermShowData(PI(&pi,i)->Index);
                            FuncShowData(PI(&pi,i)->Index);
  printf(  "\n      Min :");
  BuffMin=(word*)malloc(100);
  if(BuffMin==0) printf("Memory Error");
  FindMin((PI(&pi,i)->Index),BuffMin);
  for(j=1;j<=*BuffMin;j++)
    printf(" %d",*(BuffMin+j));
  free(BuffMin);
  if(*BuffMin==0xffff) printf("Memory Error");
  }
  return 0;
 }
```

```
void SelectOut()
{
int j;
printf("\nSelectPi \n");
for(j=1;j<=*SelectPi;j++)
    printf(" %d",*(SelectPi+j));
}

/************** EQ ***************/

void EqOut()
{
byte STerm[50],ISTerm;
static byte i,j,z,w;
word ibx;

unsigned long t,StoreSelect[16];
byte BuffCn,q;

word SelectPiFunc[100];

/* Select PI in Function
SelectPiFunc;count binary t;BuffCn;
*/

if(loop==1)
   {
   if((fp_eq=fopen("QM.EQ","w"))==NULL)
         {
         gotoxy(30,12);
         printf("\aCan't create equation file !\n");
         gotoxy(32,14);
         printf("please check disk...");
         getch();
         exit(-1);
         }
   }
else
   {
         if((fp_eq=fopen("QM.EQ","a"))==NULL)
         {
         gotoxy(30,12);
         printf("\aCan't create equation file !\n");
         gotoxy(32,14);
         printf("please check disk...");
         getch();
         exit(-1);
         }
   }


for(j=1;j<=MaxFunc;j++)
   {
   BuffCn=0xFF;
   for(t=1;t<=pow(2,*SelectPI)-1;t++)
         {
         for(q=1;q<=Func[j][0];q++)
            {
            FindBitOn(t,&SelectPiFunc[0]);
            z=1;w=1;
                  while(  (Func[j][q]!=PI(&pi,*(SelectPI+*(SelectPiFunc+z)))->Min[w])
(CmpBitOn(Term2(&term2,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Flags,j)!=1))
                  {
                  if((z==*SelectPiFunc)&&(w==PI(&pi,*(SelectPI+*(SelectPiFunc+z)))->Min[0]))
                        goto lb2;
                  else
                        if(w==PI(&pi,*(SelectPI+*(SelectPiFunc+z)))->Min[0])
```

```
                                        { w=0;z++;}
                        w++;
                        }
                    }
                if(BuffCn>Counter(t))
                        {StoreSelect[]]=t;BuffCn=Counter(t);}
                lb2:;
                }
        }
    if(loop==1)
        {
        if(Minterm=='1') fprintf(fp_eq,"\n   Minterm\n");
        else          fprintf(fp_eq,"\n   Maxterm\n");
        }

    for(j=1;j<=MaxFunc;j++)
        {
        FindBitOn(StoreSelect[j],&SelectPiFunc[0]);
        if(Minterm=='1')
              fprintf(fp_eq,"\n    F%d = ",loop);
        else fprintf(fp_eq,"\n    F'%d = ",loop);
        if(Func[j][0]==0)
              {
              if(Minterm=='1') putchar('0');
              else          putchar('1');
              }
        else
              {
              for(z=1;z<=*SelectPiFunc;z++)
                    if((Term1(&term1,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index
    >CutOut==0)&&(CmpBitOn(Term2(&term2,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Func,j)==1))
                    w=0;
                else  w=1;
              if(w!=1)
                {
                if(Minterm=='1') putchar('1');
                else          putchar('0');
                }
              else
                for(z=1;z<=*SelectPiFunc;z++)
                {
                if(CmpBitOn(Term2(&term2,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Flags,j)==1)
                      {
                      if(w!=1)
                        if(Minterm=='1') fprintf(fp_eq," + ");
                        else          fprintf(fp_eq," . ");
                      w=0;
                      fprintf(fp_eq,"%c",*(SelectPiFunc+z)+0x60);
                      }
                }
              }
        }
        fprintf(fp_eq,"\n");
    if(Minterm=='1')
        {
        for(j=1;j<=MaxFunc;j++)
              {
              FindBitOn(StoreSelect[j],&SelectPiFunc[0]);
              fprintf(fp_eq,"\n    F%d = ",loop);
              if(Func[j][0]==0) putchar('0');
              else
                {
                for(z=1;z<=*SelectPiFunc;z++)
                      if((Term1(&term1,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index
    >CutOut==0)&&(CmpBitOn(Term2(&term2,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Func,j)==1))
                      w=0;
                  else w=1;
                if(w!=1) putchar('1');
```

```
            else
              for(z=1;z<=*SelectPiFunc;z++)
              {
              if(CmpBitOn(Term2(&term2,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Flags,j)==1)
                    {
                    lbx=0x8000;
                    lSTerm=0;
                    if(w!=1) fprintf(fp_eq," + ");
                    w=0;
                    for(i=1;i<=16;i++)
                       {
                       if(i>(16-MSB))
                             {
                             if((Term1(&term1,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->CutOut
lbx)==lbx)
                                {
                                lSTerm++;
                                STerm[lSTerm-1]=64+i-16+MSB;
                                     if((Term1(&term1,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Data
lbx)==0)
                                        {
                                        lSTerm++;
                                        STerm[lSTerm-1]=39;
                                        }
                                }
                             }
                       lbx>>=1;
                       }
                    STerm[lSTerm]=0;
                    fprintf(fp_eq,"%s",STerm);
                    }
              }
            }
          }
    }
  else
    {
    for(j=1;j<=MaxFunc;j++)
     {
     FindBitOn(StoreSelect[j],&SelectPiFunc[0]);
          fprintf(fp_eq,"\n    F'%d = ",loop);
     if(Func[j][0]==0)
          putchar('1');
     else
          {
          for(z=1;z<=*SelectPiFunc;z++)
               if((Term1(&term1,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)
>CutOut==0)&&(CmpBitOn(Term2(&term2,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Func,j)==1))
                    w=0;
            else w=1;
          if(w!=1) putchar('0');
          else
           for(z=1;z<=*SelectPiFunc;z++)
           {
           if(CmpBitOn(Term2(&term2,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Flags,j)==1)
           {
           fprintf(fp_eq,"(");
           lbx=0x8000;
           lSTerm=0;
           w=1;
           for(i=1;i<=16;i++)
            {
            if(i>(16-MSB))
                  {
                  if((w!=1)&&((Term1(&term1,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->CutOut &
lbx)==lbx))
                        { lSTerm++;STerm[lSTerm-1]='+'; }
                  if((Term1(&term1,PI(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->CutOut & lbx)==lbx)
```

```
                        {
                        w=0;
                        lSTerm++;
                        STerm[lSTerm-1]=64+i-16+MSB;
                        if((~Term1(&term1,Pi(&pi,*(SelectPi+*(SelectPiFunc+z)))->Index)->Data & lbx)==0)
                          { lSTerm++;STerm[lSTerm-1]=39; }
                        }
                      }
                  lbx >>=1;
                  }
                STerm[lSTerm]=0;
                fprintf(fp_eq,"%s)",STerm);
                }
              }
            }
        }
      }
  fprintf(fp_eq,"\n");
  fclose(fp_eq);
  }


  dword pow(dword x,dword y)
  {
     dword loop;
     dword result=x;
     for(loop=1;loop<y;loop++)
      {
       result*=x;
      }
     return(result);
  }


  int allocterm1(struct TERM1 *term1)
  {
          long size;
          if((term1->term1 = (struct record far *)farmalloc(256*sizeof(struct record))) == NULL)
                  return 0;
          /* alloc buffer in conventional memory 256*sizeof(record) byte */
          size = (long)((term1->TermNum*sizeof(struct record))/1000) + 1;
          /* uppper line convert size to KB */
          term1->handle= XMSalloc(size);
          if(XMS_ERR != 0x00)
                  printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
          if(!(term1->handle))
          {
                  farfree(term1->term1);
                  return 0;
        }
          term1->oldpage=0;
          return 1;
  }

  struct record far *Term1(struct TERM1 *term1,int index)
  {
          register int page=0;
          register long XMSoffset;
          /* check page fault */
          XMSoffset = (long)(index/256LU);
          page = (int)XMSoffset;
          if(page != term1->oldpage)
          {
                  XMSwriteoff(term1->term1,term1->handle,(256LU * sizeof(struct record)),
                          ((long)term1->oldpage*256LU*sizeof(struct record)));
                  XMSreadoff(term1->term1,term1->handle,(256LU * sizeof(struct record)),
                          ((long)page*256LU*sizeof(struct record)));
                  term1->oldpage = page;
```

```
        }
        return  (struct record far *)((char far *)term1->term1 + ((index*sizeof(struct record))
                - (page * 256LU * sizeof(struct record))));
}


void freeterm1(struct TERM1 *term1)
{
        farfree(term1->term1);
        XMSfree(term1->handle);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);

}

int allocterm2(struct TERM2 *term2)
{
        long size;
        if((term2->term2 = (struct record2 far *)farmalloc(256*sizeof(struct record2))) == NULL)
                return 0;
        /* alloc buffer in conventional memory 256*sizeof(record2) byte */
        size = (long)((term2->Term2Num*sizeof(struct record2))/1000) + 1;
        /* uppper line convert size to KB */
        term2->handle= XMSalloc(size);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        if(!(term2->handle))
        {
                farfree(term2->term2);
                return 0;
        }
        term2->oldpage=0;
        return 1;
}

struct record2 far *Term2(struct TERM2 *term2,int index)
{
        register int page=0;
        register long XMSoffset;
        /* check page fault */
        XMSoffset = (long)(index/256LU);
        page = (int)XMSoffset;
        if(page != term2->oldpage)
        {
                XMSwriteoff(term2->term2,term2->handle,(256LU * sizeof(struct record2)),
                        ((long)term2->oldpage*256*sizeof(struct record2)));
                XMSreadoff(term2->term2,term2->handle,(256LU * sizeof(struct record2)),
                        ((long)page*256*sizeof(struct record2)));
                term2->oldpage = page;
        }
        return  (struct record2 far *)((char far *)term2->term2 + ((index*sizeof(struct record2))
                - (page * 256LU * sizeof(struct record2))));
}

void freeterm2(struct TERM2 *term2)
{
        farfree(term2->term2);
        XMSfree(term2->handle);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);

}

int allocterm3(struct TERM3 *term3)
{
        long size;
        if((term3->term3 = (struct record3 far *)farmalloc(256*sizeof(struct record3))) == NULL)
                return 0;
```

```
        /* alloc buffer in conventional memory 256*sizeof(record3) byte */
        size = (long)((term3->Term3Num*sizeof(struct record3))/1000) + 1;
        /* uppper line convert size to KB */
        term3->handle= XMSalloc(size);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        if(!(term3->handle))
        {
                farfree(term3->term3);
                return 0;
        }
        term3->oldpage=0;
        return 1;
}

struct record3 far *Term3(struct TERM3 *term3,int index)
{
        register int page=0;
        register long XMSoffset;
        /* check page fault */
        XMSoffset = (long)(index/256LU);
        page = (int)XMSoffset;
        if(page != term3->oldpage)
        {
                XMSwriteoff(term3->term3,term3->handle,(256LU * sizeof(struct record3)),
                        ((long)term3->oldpage*256LU*sizeof(struct record3)));
                XMSreadoff(term3->term3,term3->handle,(256LU * sizeof(struct record3)),
                        ((long)page*256LU*sizeof(struct record3)));
                term3->oldpage = page;
        }
        return  (struct record3 far *)((char far *)term3->term3 + ((index*sizeof(struct record3))
                - (page * 256LU * sizeof(struct record3))));
}

void freeterm3(struct TERM3 *term3)
{
        farfree(term3->term3);
        XMSfree(term3->handle);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);

}

int allocterm4(struct TERM4 *term4)
{
        long size;
        if((term4->term4 = (struct record4 far *)farmalloc(256*sizeof(struct record4))) == NULL)
                return 0;
        /* alloc buffer in conventional memory 256*sizeof(record4) byte */
        size = (long)((term4->Term4Num*sizeof(struct record4))/1000) + 1;
        /* uppper line convert size to KB */
        term4->handle= XMSalloc(size);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        if(!(term4->handle))
        {
                farfree(term4->term4);
                return 0;
        }
        term4->oldpage=0;
        return 1;
}

struct record4 far *Term4(struct TERM4 *term4,int index)
{
        register int page=0;
        register long XMSoffset;
        /* check page fault */
```

```
        XMSoffset = (long)(index/256LU);
        page = (int)XMSoffset;
        if(page != term4->oldpage)
        {
                XMSwriteoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
                        ((long)term4->oldpage*256*sizeof(struct record4)));
                XMSreadoff(term4->term4,term4->handle,(256LU * sizeof(struct record4)),
                        ((long)page*256*sizeof(struct record4)));
                term4->oldpage = page;
        }
        return  (struct record4 far *)((char far *)term4->term4 + ((index*sizeof(struct record4))
                - (page * 256LU * sizeof(struct record4))));
}
void freeterm4(struct TERM4 *term4)
{
        farfree(term4->term4);
        XMSfree(term4->handle);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);

}

int allocPi(struct PIV *piv)
{
        long size;
        if((piv->piv = (struct record5 far *)farmalloc(64*sizeof(struct record5))) == NULL)
                return 0;
        /* alloc buffer in conventional memory 64*sizeof(record5) byte */
        size = (long)((piv->PiNum*sizeof(struct record5))/1000) + 1;
        /* uppper line convert size to KB */
        piv->handle= XMSalloc(size);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
        if(!(piv->handle))
        {
                farfree(piv->piv);
                return 0;
        }
        piv->oldpage=0;
        return 1;
}

struct record5 far *Pi(struct PIV *piv,int index)
{
        register int page=0;
        register long XMSoffset;
        /* check page fault */
        XMSoffset = ((long)((index*sizeof(struct record5))/64LU));;
        page = (int)XMSoffset;
        if(page != piv->oldpage)
        {
                XMSwriteoff(piv->piv,piv->handle,(64LU * sizeof(struct record5)),
                        ((long)piv->oldpage*64*sizeof(struct record5)));
                XMSreadoff(piv->piv,piv->handle,(64LU * sizeof(struct record5)),
                        ((long)page*64*sizeof(struct record5)));
                piv->oldpage = page;
        }
        return  (struct record5 far *)((char far *)piv->piv + ((index*sizeof(struct record5))
                - (page * 64LU * sizeof(struct record5))));
}
void freePi(struct PIV *piv)
{
        farfree(piv->piv);
        XMSfree(piv->handle);
        if(XMS_ERR != 0x00)
                printf("Error code[%X] -> %s\n",XMS_ERR,XMSerr);
}
```

READ.C

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>

void read_file(char filename[]);
void write_char(int x,int y,int ch,int attrib);
void print_page(int sx,int sy,int ex,int ey);
void scroll(int sx,int sy,int ex,int ey,int lines,int direct);

char eq[100][500];
unsigned int row=0,col=0;
unsigned char *scr;
char *name;

main(int argc,char *argv[])
{
  clrscr();
  name=argv[1];
  if(argc<=1) { printf("\nUsage: READ filename\n"); exit(1); }
  read_file(argv[1]);
  print_page(1,2,78,22);
}

void read_file(char filename[])
{
  FILE *fp_eq;
  int ch;
  int i,j;

  if((fp_eq=fopen (filename,"r"))==NULL )
    {
     printf("Can't open file %s \n",filename);
     exit(1);
    }
   for(i=0;i!=-1;i++)
    {
     if(row<i) row=i;
     ch=0;
     for(j=0;ch!='\n';j++)
          {
          if(col<j) col=j;
          eq[i][j]=getc(fp_eq);
          ch=eq[i][j];
          if(ch==EOF)
            {
             eq[i][j+1]=-1;
             eq[i+1][0]=-10;
             i=-2;
             ch='\n';
            }
          }
    }
  fclose(fp_eq);
}

void write_char(int x,int y,int ch,int attrib)
{
  register int i;
  char far *video;
  video=(char far *)0xb8000000;
  video+=(y*160)+(x*2);
  *video++ = ch;
```

```
    *video=attrib;
}

void scroll(int sx,int sy,int ex,int ey,int lines,int direct)
{
   union REGS regs;
   if(direct==1) regs.h.ah=6;  // UP
   else regs.h.ah=7;           // DOWN
   regs.h.al=lines;
   regs.h.ch=sy;
   regs.h.cl=sx;
   regs.h.dh=ey;
   regs.h.dl=ex;
   regs.h.bh=0;
   int86(0x10,&regs,&regs);
}

void print_page(int sx,int sy,int ex,int ey)
{
   char ch=0,hor=0;
   int i,j,x,y;
   int old_i=0,old_j=0;
   int key1,key2;

   for(x=0;x<=79;x++) write_char(x,0,' ',31);
   gotoxy(5,1); printf("%s",name);
   for(x=0;x<=55;x++) write_char(x,24,' ',28);
   for(x=56;x<=79;x++) write_char(x,24,' ',30);
   gotoxy(2,25); printf("Use");
   write_char(5,24,24,28);
   write_char(7,24,25,28);
   write_char(9,24,26,28);
   write_char(11,24,27,28);
   gotoxy(14,25); printf("PgUp-PgDn or Home-End for scroll screen.");
   for(;;)
    {
     gotoxy(59,25); printf("Row = %3d Col = %3d",old_i,old_j);
     ch=0;
     if(hor) scroll(sx,sy,ex,ey,0,7);
     hor=0;

     for(y=sy,i=old_i;y<=ey;y++,i++)
          {
           for(x=sx,j=old_j;x<=ex;x++,j++)
            {
             ch=eq[i][j];
             if(ch=='\n'|| ch==-1) { x=ex; ch=0; }
             write_char(x,y,ch,11);
            }
           if(ch=='\n')  write_char(x,y,0,11);
          }

     if((key1=getch())!=0)
          {
           if(key1==27)  // ESC
            {
             for(x=0;x<=79;x++) write_char(x,24,' ',7);
             exit(0);
            }
          }
     else
     key2=getch();
     switch(key2)
          {
           case 80 :   // DOWN
            {
             if(row-old_i<=(ey-sy)|| row<=(ey-sy)) break;
             old_i++;
```

```
            scroll(sx,sy,ex,ey,1,1);
            break;
           }
          case 72 :   // UP
           {
           if(old_i<=0) break;
           old_i--;
           scroll(sx,sy,ex,ey,1,0);
           break;
           }
          case 77 :  // RIGHT
           {
           if(col-old_j<=(ex-sx) || col<=(ex-sx)) break;
           old_j+=5;
           hor=1;
           break;
           }
          case 75 :  // LEFT
           {
           if(old_j<=0) { old_j=0; break; }
           old_j-=5;
           hor=1;
           break;
           }
          case 71 :  // HOME
           {
           old_j=0;
           hor=1;
           break;
           }
          case 79 :  // END
           {
           if(col<=(ex-sx))  break;
           old_j=col-(ex-sx);
           hor=1;
           break;
           }
          case 73 :  // PAGE UP
           {
           old_i-=(ey-sy-1);
           if(old_i<=0) old_i=0;
           scroll(sx,sy,ex,ey,ey-sy+1,0);
           break;
           }
          case 81 :  // PAGE DOWN
           {
           if(row<=(ey-sy)) break;
           old_i+=(ey-sy+1);
           if(row-old_i<=(ey-sy) || row<old_i) old_i=row-(ey-sy);
           scroll(sx,sy,ex,ey,ey-sy+1,1);
           break;
           }

          default : break;
         }
    }
}
```

CLA.ASM

```
outport macro port,dat
                mov dx,300h
                mov al,port
                out dx,al
                mov dx,30fh
                mov al,dat
                out dx,al
                endm
inport macro port
                mov dx,300h
                mov al,port
                out dx,al
                mov dx,30fh
                in  al,dx
                endm
;/*************************************************************/
cseg segment public
assume cs:cseg,ds:dseg,ss:stck
extrn mov_cursor:near,get_cursor:near
extrn write_char:near,write_table:near
extrn window:near,write_text:near
main proc near
                mov  ax,dseg
                mov  ds,ax
                mov  ax,stck
                mov  ss,ax
                outport 13h,88h
                outport 10h,80h
                outport 11h,88h
                outport 12h,01h
                outport 03h,80h
                outport 0bh,9bh
                outport 0fh,9bh
                outport 07h,80h
                outport 04h,0ffh     ;set for high impedance
                outport 05h,0ffh
                outport 06h,0ffh
                inport  10h
                cmp     al,80h
                jne     nohard
                call cursor_off
                call del_file
                mov  bh,7
                mov  cx,0
                mov  dx,184fh
                call window
                mov  bh,00010111b
                mov  cx,202h
                mov  dx,164ch
                call window
                mov  dx,202h
                mov  bl,17h
                mov  bh,14h
                mov  cl,4bh
                call write_table
                mov  dx,218h
                call mov_cursor
                mov  bl,3bh
                lea  si,logo
                call write_text
                call write_bar
                call des_operation
                call get_key
nohard:           mov  cx,0
```

```
                mov  dx,184fh
                mov  bh,7
                call window
                mov  dx,0
                call mov_cursor
                mov  bl,7
                mov  cx,4
                lea  si,bye
nextline:         call write_text
                inc  dh
                call mov_cursor
                loop nextline
                mov  ah,7
                int  21h
                mov  ax,4c01h
                int  21h
main endp
;/*************************************************************/
del_file proc near
                lea  dx,filename1
                mov  cx,24
del:             mov  ah,41h
                int  21h
                add  dx,12
                loop del
                lea  dx,timebase1
                mov  ah,41h
                int  21h
                ret
del_file endp
;/*************************************************************/
clear proc near
                mov bh,17h
                mov cx,805h
                mov dx,1449h
                call window
                ret
clear endp
;/*************************************************************/
dialog proc near
                mov  dx,305h
                mov  bl,17h
                mov  bh,3
                mov  cl,13
                call write_table
                mov  dx,406h
                call mov_cursor
                lea  si,oper
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,level
                call write_text
                mov  dx,312h
                mov  bl,17h
                mov  bh,3
                mov  cl,12
                call write_table
                mov  dx,413h
                call mov_cursor
                lea  si,time
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,sel
                call write_text
```

```
                mov  dx,31eh
                mov  bl,17h
                mov  bh,3
                mov  cl,13
                call write_table
                mov  dx,41fh
                call mov_cursor
                lea  si,thes
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,seln
                call write_text
                mov  dx,32bh
                mov  bl,17h
                mov  bh,3
                mov  cl,12
                call write_table
                mov  dx,42ch
                call mov_cursor
                lea  si,chan
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,grop
                call write_text
                mov  dx,337h
                mov  bl,17h
                mov  bh,3
                mov  cl,9
                call write_table
                mov  dx,438h
                call mov_cursor
                lea  si,start
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,cla
                call write_text
                mov  dx,340h
                mov  bl,17h
                mov  bh,3
                mov  cl,10
                call write_table
                mov  dx,441h
                call mov_cursor
                lea  si,help
                mov  bl,17h
                call write_text
                inc  dh
                call mov_cursor
                lea  si,menu
                call write_text
                ret
dialog endp
;*****************************************************
des_operation proc near
                mov  dx,708h
                call mov_cursor
                mov  bl,17h
                mov  cx,10
                lea  si,mess1
onemore:        inc  dh
                call mov_cursor
                call write_text
```

```
                loop onemore
                mov dx,1505h
                call mov_cursor
                lea si,reg
                call write_text
                ret
des_operation endp
;*************************************************
des_timebase proc near
                mov dx,708h
                call mov_cursor
                mov cx,10
                mov bl,17h
                lea si,time1
timescan:         inc dh
                call mov_cursor
                call write_text
                loop timescan
                mov dx,0928h
                call mov_cursor
                mov cx,8
                lea si,time10
timescan1:        inc dh
                call mov_cursor
                call write_text
                loop timescan1
                ret
des_timebase endp
;*************************************************
des_threshold proc near
                mov dx,708h
                call mov_cursor
                mov cx,10
                mov bl,17h
                lea si,thres1
thresscan:        inc dh
                call mov_cursor
                call write_text
                loop thresscan
                ret
des_threshold endp
;*************************************************
des_channel proc near
                mov dx,708h
                call mov_cursor
                mov bl,17h
                mov cx,8
                lea si,chan1
chanscan:         inc dh
                call mov_cursor
                call write_text
                loop chanscan
                ret
des_channel endp
;*************************************************
get_key proc near
                push ax
                push bx
read_again:       mov ah,0
                int 16h       ;ah=scan code,al=ascii
                lea bx,table_key
spectial:       cmp word ptr [bx],0
                je  read_again
                cmp word ptr ax,[bx]
                je  in_table
                add bx,4
                jmp spectial
in_table:       inc bx
```

```
                inc  bx
                call word ptr [bx]
                jmp  read_again
quitt:           pop  bx
                pop  ax
                ret
get_key endp
;*************************************************
rbar proc near
                cmp  byte ptr ds:hor1,5
                jne  no_right
                mov  byte ptr ds:hor1,0
                jmp  tabbar_ok
no_right:         inc  byte ptr ds:hor1
tabbar_ok:         call write_bar
                call write_destination
                ret
rbar endp
;*************************************************
lbar proc near
                cmp  byte ptr ds:hor1,0
                jne  no_left
                mov  byte ptr ds:hor1,5
                jmp  lbar_ok
no_left:          dec  byte ptr ds:hor1
lbar_ok:           call write_bar
                call write_destination
                ret
lbar endp
;*************************************************
dbar proc near
                cmp  byte ptr ds:hor1,1
                je   barvo
                cmp  byte ptr ds:hor1,2
                je   next1
                cmp  byte ptr ds:hor1,3
                jne  ext1
                cmp  byte ptr ds:ver3,5
                je   v35
                inc  byte ptr ds:ver3
                jmp  v3
v35:              mov  byte ptr ds:ver3,0
v3:              call channel_bar
                ret
next1:            cmp  byte ptr ds:ver2,7
                je   eq7
                inc  byte ptr ds:ver2
                jmp  ext
eq7:              mov  byte ptr ds:ver2,0
ext:             call threshold_bar
ext1:             ret
barvo:            cmp  byte ptr ds:ver1,0fh
                je   more
                inc  byte ptr ds:ver1
                jmp  final
more:             mov  byte ptr ds:ver1,0
final:           call timebase_bar
                ret
dbar endp
;*************************************************
ubar proc near
                cmp  byte ptr ds:hor1,1
                je   barvou
                cmp  byte ptr ds:hor1,2
                je   next1u
                cmp  byte ptr ds:hor1,3
                jne  ext1u
                cmp  byte ptr ds:ver3,0
```

```
                je  v35u
                dec  byte ptr ds:ver3
                jmp  v3u
 v35u:            mov  byte ptr ds:ver3,5
 v3u:            call channel_bar
                ret
 next1u:          cmp  byte ptr ds:ver2,0
                je  eq7u
                dec  byte ptr ds:ver2
                jmp  extu
 eq7u:            mov  byte ptr ds:ver2,7
 extu:           call threshold_bar
 ext1u:           ret
 barvou:          cmp  byte ptr ds:ver1,0
                je  moreu
                dec  byte ptr ds:ver1
                jmp  finalu
 moreu:           mov  byte ptr ds:ver1,0fh
 finalu:         call timebase_bar
                ret
 ubar endp
 ;*************************************************
 escm proc near
                call cursor_on
                mov  ax,3
                int  10h
                outport 11h,00h
                mov  ax,4c01h
                int  21h
 escm endp
 ;*************************************************
 extrn initial:near              ;trigger:near
 extrn reading:near,rearrange:near
 ent proc near
                cmp  hor1,4
                je  continue
                ret
 continue:        call write_begin
                mov  dx,0a08h
                call mov_cursor
                mov  bl,0f0h
                lea  si,begin2
                call write_text
                call initial
                call write_begin
                mov  dx,0c08h
                call mov_cursor
                mov  bl,0f0h
                lea  si,begin3
                call write_text
                call write_begin
                mov  dx,0e08h
                call mov_cursor
                mov  bl,0f0h
                lea  si,begin4
                call write_text
                call reading
                call write_begin
                mov  dx,1008h
                call mov_cursor
                mov  bl,0f0h
                lea  si,begin5
                call write_text
                call rearrange
                call write_begin
                mov  dx,1208h
                call mov_cursor
                mov  bl,0f0h
```

```
                lea si,begin6
                call write_text
                mov cx,10
movdx:            mov dx,0ffffh
decdx:            dec dx
                jnz decdx
                loop movdx
                call cursor_on
                outport 11h,80h
                mov cx,0
                mov dx,184fh
                mov bh,7
                call window
                mov dx,0
                call mov_cursor
                mov bl,7
                mov cx,2
                lea si,complete
okagain:          call write_text
                inc dh
                call mov_cursor
                loop okagain
                outport 11h,00h
                mov ax,4c00h
                int 21h
ent endp
;*************************************************
write_begin proc near
                call clear
                mov dx,0808h
                call mov_cursor
                lea si,begin1
                mov bl,17h
                call write_text
                inc dh
                inc dh
                call mov_cursor
                mov cx,8
                lea si,begin2
                call write_text
repeat:           inc dh
                call mov_cursor
                call write_text
                loop repeat
                ret
write_begin endp
;*************************************************
write_bar proc near
                call dialog
                cmp byte ptr ds:hor1,5
                je h5f
                cmp byte ptr ds:hor1,4
                je h4f
                cmp byte ptr ds:hor1,3
                je h3f
                cmp byte ptr ds:hor1,2
                je h2
                cmp byte ptr ds:hor1,1
                je h1
h0:               mov dx,305h
                mov bl,3fh
                mov bh,3
                mov cl,13
                call write_table
                mov dx,406h
                call mov_cursor
                lea si,oper
                call write_text
```

```
                inc  dh
                call mov_cursor
                lea  si,level
                call write_text
                ret
h1:              mov  dx,312h
                mov  bl,2fh
                mov  bh,3
                mov  cl,12
                call write_table
                mov  dx,413h
                call mov_cursor
                lea  si,time
                call write_text
                inc  dh
                call mov_cursor
                lea  si,sel
                call write_text
                ret
h5f:             jmp  h5
h4f:             jmp  h4
h3f:             jmp  h3
h2:              mov  dx,31eh
                mov  bl,2fh
                mov  bh,3
                mov  cl,13
                call write_table
                mov  dx,41fh
                call mov_cursor
                lea  si,thes
                call write_text
                inc  dh
                call mov_cursor
                lea  si,seln
                call write_text
                ret
h3:              mov  dx,32bh
                mov  bl,2fh
                mov  bh,3
                mov  cl,12
                call write_table
                mov  dx,42ch
                call mov_cursor
                lea  si,chan
                call write_text
                inc  dh
                call mov_cursor
                lea  si,grop
                call write_text
                ret
h4:              mov  dx,337h
                mov  bl,04fh
                mov  bh,3
                mov  cl,9
                call write_table
                mov  dx,438h
                call mov_cursor
                lea  si,start
                call write_text
                inc  dh
                call mov_cursor
                lea  si,cla
                call write_text
                ret
h5:              mov  dx,340h
                mov  bl,5fh
                mov  bh,3
                mov  cl,10
```

```
                call write_table
                mov  dx,441h
                call mov_cursor
                lea  si,help
                call write_text
                inc  dh
                call mov_cursor
                lea  si,menu
                call write_text
                ret
write_bar endp
;*************************************************
threshold_bar proc near
                call des_threshold
                mov  bl,70h
                cmp  byte ptr ds:ver2,7
                je   th7f
                cmp  byte ptr ds:ver2,6
                je   th6f
                cmp  byte ptr ds:ver2,5
                je   th5
                cmp  byte ptr ds:ver2,4
                je   th4
                cmp  byte ptr ds:ver2,3
                je   th3
                cmp  byte ptr ds:ver2,2
                je   th2
                cmp  byte ptr ds:ver2,1
                je   th1
th0:             mov  dx,0a08h
                call mov_cursor
                lea  si,thres2
                call write_text
                jmp  adj_threshold
th1:             mov  dx,0b08h
                call mov_cursor
                lea  si,thres3
                call write_text
                jmp  adj_threshold
th2:             mov  dx,0c08h
                call mov_cursor
                lea  si,thres4
                call write_text
                jmp  adj_threshold
th6f:            jmp  th6
th7f:            jmp  th7
th3:             mov  dx,0d08h
                call mov_cursor
                lea  si,thres5
                call write_text
                jmp  adj_threshold
th4:             mov  dx,0e08h
                call mov_cursor
                lea  si,thres6
                call write_text
                jmp  adj_threshold
th5:             mov  dx,0f08h
                call mov_cursor
                lea  si,thres7
                call write_text
                jmp  adj_threshold
th6:             mov  dx,1008h
                call mov_cursor
                lea  si,thres8
                call write_text
                jmp  adj_threshold
th7:             mov  dx,1108h
                call mov_cursor
```

```
                lea  si,thres9
                call write_text
adj_threshold:      push ax
                push bx
                push cx
                push dx
                mov  bx,0000h
                mov  byte ptr bl,ds:ver2
                lea  si,volt_table
                outport 10h,[si+bx]
                pop  dx
                pop  cx
                pop  bx
                pop  ax
                ret
threshold_bar endp
;*************************************************
channel_bar proc near
                call des_channel
                mov  bl,70h
                cmp  byte ptr ds:ver3,5
                je   ch5
                cmp  byte ptr ds:ver3,4
                je   ch4
                cmp  byte ptr ds:ver3,3
                je   ch3
                cmp  byte ptr ds:ver3,2
                je   ch2
                cmp  byte ptr ds:ver3,1
                je   ch1
ch0:              mov  dx,0a08h
                call mov_cursor
                lea  si,chan2
                call write_text
                ret
ch1:              mov  dx,0b08h
                call mov_cursor
                lea  si,chan3
                call write_text
                ret
ch2:              mov  dx,0c08h
                call mov_cursor
                lea  si,chan4
                call write_text
                ret
ch3:              mov  dx,0d08h
                call mov_cursor
                lea  si,chan5
                call write_text
                ret
ch4:              mov  dx,0e08h
                call mov_cursor
                lea  si,chan6
                call write_text
                ret
ch5:              mov  dx,0f08h
                call mov_cursor
                lea  si,chan7
                call write_text
                ret
channel_bar endp
;*************************************************
timebase_bar proc near
                call des_timebase
                mov  bl,70h
                cmp  byte ptr ds:ver1,0fh
                je   tiff
                cmp  byte ptr ds:ver1,0eh
```

```
                je   tief
                cmp  byte ptr ds:ver1,0dh
                je   tidf
                cmp  byte ptr ds:ver1,0ch
                je   ticf
                cmp  byte ptr ds:ver1,0bh
                je   tibf
                cmp  byte ptr ds:ver1,0ah
                je   tiaf
                cmp  byte ptr ds:ver1,09h
                je   ti9f
                cmp  byte ptr ds:ver1,08h
                je   ti8f
                cmp  byte ptr ds:ver1,07h
                je   ti7f
                cmp  byte ptr ds:ver1,06h
                je   ti6f
                cmp  byte ptr ds:ver1,05h
                je   ti5
                cmp  byte ptr ds:ver1,04h
                je   ti4
                cmp  byte ptr ds:ver1,03h
                je   ti3
                cmp  byte ptr ds:ver1,02h
                je   ti2
                cmp  byte ptr ds:ver1,01h
                je   ti1
ti0:            mov  dx,0a08h
                call mov_cursor
                lea  si,time2
                call write_text
                ret
ti1:            mov  dx,0b08h
                call mov_cursor
                lea  si,time3
                call write_text
                ret
tiff:           jmp  tif
tief:           jmp  tie
tidf:           jmp  tid
ticf:           jmp  tic
tibf:           jmp  tib
tiaf:           jmp  tia
ti9f:           jmp  ti9
ti8f:           jmp  ti8
ti7f:           jmp  ti7
ti6f:           jmp  ti6
ti2:            mov  dx,0c08h
                call mov_cursor
                lea  si,time4
                call write_text
                ret
ti3:            mov  dx,0d08h
                call mov_cursor
                lea  si,time5
                call write_text
                ret
ti4:            mov  dx,0e08h
                call mov_cursor
                lea  si,time6
                call write_text
                ret
ti5:            mov  dx,0f08h
                call mov_cursor
                lea  si,time7
                call write_text
                ret
ti6:            mov  dx,1008h
```

```
                call mov_cursor
                lea  si,time8
                call write_text
                ret
ti7:            mov  dx,1108h
                call mov_cursor
                lea  si,time9
                call write_text
                ret
ti8:            mov  dx,0a28h
                call mov_cursor
                lea  si,time10
                call write_text
                ret
ti9:            mov  dx,0b28h
                call mov_cursor
                lea  si,time11
                call write_text
                ret
tia:            mov  dx,0c28h
                call mov_cursor
                lea  si,time12
                call write_text
                ret
tib:            mov  dx,0d28h
                call mov_cursor
                lea  si,time13
                call write_text
                ret
tic:            mov  dx,0e28h
                call mov_cursor
                lea  si,time14
                call write_text
                ret
tid:            mov  dx,0f28h
                call mov_cursor
                lea  si,time15
                call write_text
                ret
tie:            mov  dx,1028h
                call mov_cursor
                lea  si,time16
                call write_text
                ret
tif:            mov  dx,1128h
                call mov_cursor
                lea  si,time17
                call write_text
                ret
timebase_bar endp
;*************************************************
start_bar proc near
                mov  dx,808h
                call mov_cursor
                mov  bl,17h
                lea  si,start1
                call write_text
                inc  dh
                inc  dh
                call mov_cursor
                lea  si,time1
                call write_text
                inc  dh
                call mov_cursor
                lea  si,time2
                mov  ax,32
                mov  cx,0
                mov  cl,ds:ver1
```

```
                mul  cl
                add  si,ax
                call write_text
                inc  dh
                inc  dh
                call mov_cursor
                lea  si,thres1
                call write_text
                inc  dh
                call mov_cursor
                lea  si,thres2
                mov  ax,29
                mov  cx,0
                mov  cl,ds:ver2
                mul  cl
                add  si,ax
                call write_text
                inc  dh
                inc  dh
                call mov_cursor
                lea  si,chan1
                call write_text
                inc  dh
                call mov_cursor
                lea  si,chan2
                mov  ax,42
                mov  cx,0
                mov  cl,ds:ver3
                mul  cl
                add  si,ax
                call write_text
                inc  dh
                inc  dh
                call mov_cursor
                lea  si,start2
                mov  bl,70h
                call write_text
                ret
start_bar endp
;*************************************************
help_menu proc near
                mov  dx,0808h
                call mov_cursor
                mov  cx,10
                mov  bl,17h
                lea  si,help1
                call write_text
nextphres:        inc  dh
                call mov_cursor
                call write_text
                loop nextphres
                ret
help_menu endp
;*************************************************
write_destination proc near
                call clear
                cmp  byte ptr ds:hor1,5
                je   no_5
                cmp  byte ptr ds:hor1,4
                je   no_4
                cmp  byte ptr ds:hor1,3
                je   no_3
                cmp  byte ptr ds:hor1,2
                je   no_2
                cmp  byte ptr ds:hor1,1
                je   no_1
no_0:             call des_operation
                ret
```

```
no_1:           call timebase_bar
                ret
no_2:           call threshold_bar
                ret
no_3:           call channel_bar
                ret
no_4:           call start_bar
                ret
no_5:           call help_menu
                ret

write_destination endp
;*************************************************
cursor_on proc near
                push ax
                push cx
                push dx
                mov  ah,1
                mov  cx,sizecur
                int  10h
                pop  dx
                pop  cx
                pop  ax
                ret
cursor_on endp
;*************************************************
cursor_off proc near
                push ax
                push bx
                push cx
                push dx
                mov  ah,3
                xor  bh,bh
                int  10h
                mov  word ptr sizecur,cx
                mov  ah,1
                mov  ch,1
                xor  cl,cl
                int  10h
                pop  dx
                pop  cx
                pop  bx
                pop  ax
                ret
cursor_off endp
cseg ends
;*************************************************
dseg segment public
extrn filename1:word,timebase1:word
logo      db " COMPUTER LOGIC ANALYZER V.1.0 ",0
oper      db "  ABOUT  ",0
level     db "  MENU  ",0
time      db " TIMEBASE ",0
sel       db "  SELECT  ",0
thes      db " THRESHOLD ",0
seln      db "  SELECT  ",0
chan      db " CHANNEL  ",0
grop      db " GROUPING ",0
start     db " START ",0
cla       db "  CLA  ",0
help      db "  HELP  ",0
menu      db "  MENU  ",0
sizecur   dw ?
mess1     db "OPERATION LEVEL:",0
          db "   LEVEL 0 - BASIC OPERATION",0
          db 0
          db "CONFIGURATION:",0
          db "   SLOT 0: -  4 CHANNEL",0
```

```
        db "          - 8 CHANNEL",0
        db "   SLOT 1: - 12 CHANNEL",0
        db "          - 16 CHANNEL",0
        db "   SLOT 2: - 20 CHANNEL",0
        db "          - 24 CHANNEL",0
time1   db "TIMEBASE SELECT:",0
        db 0
time2   db "    SELECT 00 -   0.5 mS    ",0
time3   db "    SELECT 01 -   1.0 mS    ",0
time4   db "    SELECT 02 -   2.0 mS    ",0
time5   db "    SELECT 03 -   4.0 mS    ",0
time6   db "    SELECT 04 -   8.0 mS    ",0
time7   db "    SELECT 05 -  16.0 mS    ",0
time8   db "    SELECT 06 -  32.0 mS    ",0
time9   db "    SELECT 07 -  64.0 mS    ",0
time10  db "    SELECT 08 -  128.0 mS    ",0
time11  db "    SELECT 09 -  256.0 mS    ",0
time12  db "    SELECT 10 -  512.0 mS    ",0
time13  db "    SELECT 11 -  1.024 S     ",0
time14  db "    SELECT 12 -  2.048 S     ",0
time15  db "    SELECT 13 -  4.096 S     ",0
time16  db "    SELECT 14 -  8.192 S     ",0
time17  db "    SELECT 15 - 16.384 S     ",0
thres1  db "THRESHOLD SELECT:",0
        db 0
thres2  db "   SELECT 0 -  1.5 VOLT    ",0
thres3  db "   SELECT 1 -  3.0 VOLT    ",0
thres4  db "   SELECT 2 -  4.5 VOLT    ",0
thres5  db "   SELECT 3 -  5.0 VOLT    ",0
thres6  db "   SELECT 4 -  6.5 VOLT    ",0
thres7  db "   SELECT 5 -  7.5 VOLT    ",0
thres8  db "   SELECT 6 -  9.0 VOLT    ",0
thres9  db "   SELECT 7 - 10.0 VOLT    ",0
chan1   db "CHANNEL SELECT:",0
        db 0
chan2   db "   SELECT 0 -  4 CHANNEL/SLOT 0    ",0
chan3   db "   SELECT 1 -  8 CHANNEL/SLOT 0    ",0
chan4   db "   SELECT 2 - 12 CHANNEL/SLOT 1    ",0
chan5   db "   SELECT 3 - 16 CHANNEL/SLOT 1    ",0
chan6   db "   SELECT 4 - 20 CHANNEL/SLOT 2    ",0
chan7   db "   SELECT 5 - 24 CHANNEL/SLOT 2    ",0
start1  db "START COMPUTER LOGIC ANALYZER BY USED:",0
start2  db "              PRESS ",11h,0d9h," TO CONFIRM START                ",0
begin1  db "Working:",0
begin2  db "          Initial Port          ",0
        db 0
begin3  db "       Send Trigger Pulse       ",0
        db 0
begin4  db "          Reading Data          ",0
        db 0
begin5  db "          Arrange Data          ",0
        db 0
begin6  db "          Completed!            ",0
reg     db "SOFTWARE VERSION 1.0 COPYRIGHT (C) 1995 BY KMITL       ^[^E^s^c^]^=^E^x^l^t",0
help1   db "Setting Timebase.",0
        db "  1.Select to TIMEBASE SELECT bar by ^",27,"^",26," at the top of the screen.",0
        db "  2.Used ^",24,"^",25," arrow key to select one value from 0-15.",0
        db 0
        db "Setting Input Thresholds.",0
        db "  1.Select to THRESHOLD SELECT bar by ^",27,"^",26," at the top of screen.",0
        db "  2.Used ^",24,"^",25," arrow key to select the value of threshold.",0
        db 0
        db "Setting Channel Grouping.",0
        db "  1.Select to CHANNEL GROUPING bar by ^",27,"^",26," at the top of the screen.",0
        db "  2.Used ^",24,"^",25,"arrow key to select one value from 0-5.",0
bye     db "Computer Logic Analyzer.",0
        db "Hardware not install or Device error...",0
        db "Copyright (C) 1995 by Kmitl.Engineer(Ind).2N.",0
```

```
        db "Press a key to continue...",0
complete    db "Completed!.",0
        db "Copyright (C) 1995 by Kmitl.Engineer(Ind).2N.",0
volt_table  db 026h,04dh,073h,080h,0a6h,0bfh,0e6h,0ffh
;           1.50,3.00,4.50,5.00,6.50,7.50,9.00,10.0
public ver1,ver2,ver3
ver1        db 07h
ver2        db 3
ver3        db 0
hor1        db 0
TABLE_KEY LABEL BYTE
        DW 4800H                ;UpArrow
        DW OFFSET CSEG:UBAR
        DW 5000H                ;DnArrow
        DW OFFSET CSEG:DBAR
        DW 4B00H                ;LeftArrow
        DW OFFSET CSEG:LBAR
        DW 4D00H                ;RightArrow
        DW OFFSET CSEG:RBAR
        DW 1C0DH                ;ENTER
        DW OFFSET CSEG:ENT
        DW 011BH                ;ESC
        DW OFFSET CSEG:ESCM
        DW 0F09H                ;TAB
        DW OFFSET CSEG:RBAR
        DW 0F00H                ;SHIFT+TAB
        DW OFFSET CSEG:LBAR
        DW 0E08H                ;BACK
        DW OFFSET CSEG:LBAR
        DW 2D00H                ;ALT+X
        DW OFFSET CSEG:ESCM
        DW 4400H                ;F10
        DW OFFSET CSEG:ESCM
        DW 1051H                ;Q
        DW OFFSET CSEG:ESCM
        DW 1071H                ;q
        DW OFFSET CSEG:ESCM
        DW 1000H                ;ALT+Q
        DW OFFSET CSEG:ESCM
        DW 0
dseg ends
stck segment public
db 1000 dup (0)
stck ends
end main
```

PULSE.C

```
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include <bios.h>
#include <ctype.h>
#include <conio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "nimage.h"

#define PMODE0  0   /* 60  point:inc (normal)              */
#define PMODE1  1   /* 120 point:inc (2 dencity)           */
#define PMODE2  2   /* 120 point:inc (2 dencity ,hi speed) */
#define PMODE3  3   /* 240 point:inc (4 dencity)           */
#define PMODE4  4   /* 80  point:inc (CRT Graphics)        */
#define PMODE5  5   /* 72  point:inc (Plotter Graphics)    */
#define PMODE6  6   /* 90  point:inc (CRT Graphics II)     */

#define MAXDATA  15000
#define MAXFUNC     26
#define PUSH       100

void start_gph();
void read_data();
void pulse(int x,int y,int endy,int length,int space,int grid);
void plot_line(int x,int y,int endy,int length,int space,int first_ch);
void clear_line(int x,int y,int endy,int length,int space,int first_ch);
void print_hex(int x,int y);
void disp_block(int x,int y);
void help(int x,int y);
void copybyte(int byte);
void copydot(int scan,int pmode);
int  hardcopy(int left,int ptimes);

FILE *fp1,*fp2,*fp3,*fp4,*fp5,*fp6,*fp7,*fp8,*fp9,*fp10,*fp11,*fp12,*fp13,
    *fp14,*fp15,*fp16,*fp17,*fp18,*fp19,*fp20,*fp21,*fp22,*fp23,*fp24,*fp25;

char *ch[] = { "","CH1","CH2","CH3","CH4","CH5","CH6","CH7","CH8","CH9","CH10",
               "CH11","CH12","CH13","CH14","CH15","CH16","CH17","CH18",
               "CH19","CH20","CH21","CH22","CH23","CH24" };

int pattern=SOLID_FILL;
struct record {
            unsigned char logic;
          };
struct record huge ptr[MAXFUNC][MAXDATA];
int count,total;
unsigned char length,grid,oldlength;
unsigned char savedata;
int flag0,flag1;
int flag_end,flag_start;
int step,startbyte,endbyte;
int byte_of_file;
int ret_prn;
int chanel;

main()
{
  ret_prn=0;
  start_gph();
  change_color();
  loadfont("fon.t");
  read_data();
```

```
}

  pulse(41,25,43,1,40,5);
}

/*-------------------------*/

void start_gph()

{
  int driver=DETECT;
  int mode=2;
  int errorcode;
  registerbgidriver(EGAVGA_driver);
  registerbgifont(small_font);
  registerbgifont(gothic_font);
  registerbgifont(triplex_font);
  initgraph(&driver,&mode,"");
  errorcode=graphresult();
  if(errorcode!=grOk)
    {
     printf("\nGraphics system error : %s\n",grapherrormsg(errorcode));
     exit(1);
    }
}

/*-------------------------*/

void read_data()

{
  FILE *fp_cfg,*fp_ch,*fp_tb;
  char *file_data[] = { "","LOGIC01.DAT","LOGIC02.DAT","LOGIC03.DAT","LOGIC04.DAT",
                        "LOGIC05.DAT","LOGIC06.DAT","LOGIC07.DAT","LOGIC08.DAT",
                        "LOGIC09.DAT","LOGIC10.DAT","LOGIC11.DAT","LOGIC12.DAT",
                        "LOGIC13.DAT","LOGIC14.DAT","LOGIC15.DAT","LOGIC16.DAT",
                        "LOGIC17.DAT","LOGIC18.DAT","LOGIC19.DAT","LOGIC20.DAT",
                        "LOGIC21.DAT","LOGIC22.DAT","LOGIC23.DAT","LOGIC24.DAT",
                       };
  int i;
  setcolor(YELLOW);
  outtextxy(230,220,"Loading data please wait...");
  if((fp_cfg=fopen("LOGIC.CFG","rb"))==NULL)
    {
     closegraph();
     printf("can't open LOGIC.CFG file..!\n");
     getch();
     exit(0);
    }
  chanel=getc(fp_cfg);
  fclose(fp_cfg);

  for(i=1;i<=chanel;i++)
    {
     if((fp_ch=fopen(file_data[i],"rb"))==NULL)
          {
           closegraph();
           printf("can't open %s file..!\n",file_data[i]);
           getch();
           exit(0);
          }
     count=0;
     while(!feof(fp_ch))
          {
           ptr[i][count].logic=getc(fp_ch);
           count++;
          }
     fclose(fp_ch);
  }
```

```
 if((fp_tb=fopen("TIMEBASE.DAT","rb"))==NULL)
   {
    closegraph();
    printf("can't open TIMEBASE.DAT file..!\n");
    getch();
    exit(0);
   }
  count=0;
  while(!feof(fp_tb))
   {
    ptr[25][count].logic=getc(fp_tb);
    count++;
   }
  fclose(fp_tb);

  cleardevice();
  outtextxy(340,220,"Ok.");
  byte_of_file=count-1;
  total=140;
  startbyte=0;
  endbyte=total+startbyte;
  step=byte_of_file/60;
  if(step<1)  { step=1; }
}

/*--------------------------------------*/

void pulse(int x,int y,int endy,int length,int space,int grid)

{
  char key1,key2;
  static int state;
  static int n;
  static int zoom_bar1,zoom_bar2;
  static int zoom_max,zoom_min;
  int a,b,c,d,e;
  int nx,ny;
  unsigned int size;
  void *bitimage;

  if(ret_prn) { goto Form_Printing; }

  n=1;
  state=0;
  zoom_bar1=7;
  zoom_bar2=8;
  flag_start=1;
  flag_end=0;
  zoom_min=1;
  zoom_max=0;

  Form_Printing:

  setfillstyle(SOLID_FILL,LIGHTGRAY);
  bar(0,0,639,479);
  box3d(41,11,627,379,DARKGRAY,BLACK,5,8);
  setcolor(5);
  line(41-8,11-8,41,11);
  setcolor(LIGHTGRAY);
  line(627,379,627+8,379+8);
  setcolor(BLACK);
  rectangle(32,2,636,388);

  box3d(217,450,265,474,WHITE,LIGHTGRAY,DARKGRAY,2);
  print_text(226,465,"More",BLACK,0,0,1);
  box3d(270,450,318,474,WHITE,LIGHTGRAY,DARKGRAY,2);
  print_text(276,465,"Print",BLACK,0,0,1);
  box3d(323,464,371,474,WHITE,LIGHTGRAY,DARKGRAY,2);
```

```
print_text(332,466,"Quit",BLACK,0,0,1);
box3d(323,450,371,460,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(332,452,"Help",BLACK,0,0,1);
box3d(5,432,53,444,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(9,435,"State",WHITE,0,0,1);
box3d(58,432,106,444,DARKGRAY,LIGHTGRAY,WHITE,2);
print_text(59,436,"Timing",BLACK,0,0,1);
print_text(58,435,"Timing",WHITE,0,0,1);

box3d(4,28,29,39,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,28,ch[n],LIGHTBLUE,2,0,4);
box3d(4,68,29,79,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,68,ch[n+1],LIGHTBLUE,2,0,4);
box3d(4,108,29,119,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,108,ch[n+2],LIGHTBLUE,2,0,4);
box3d(4,148,29,159,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,148,ch[n+3],LIGHTBLUE,2,0,4);
box3d(4,188,29,199,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,188,ch[n+4],LIGHTBLUE,2,0,4);
box3d(4,228,29,239,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,228,ch[n+5],LIGHTBLUE,2,0,4);
box3d(4,268,29,279,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,268,ch[n+6],LIGHTBLUE,2,0,4);
box3d(4,308,29,319,WHITE,LIGHTGRAY,DARKGRAY,2);
print_text(5,308,ch[n+7],LIGHTBLUE,2,0,4);
box3d(4,343,29,364,WHITE,LIGHTGRAY,DARKGRAY,2);
setusercharsize(10,110,10,30);
print_text(5,343,"Time",RED,SMALL_FONT,0,4);
print_text(5,353,"Base",RED,SMALL_FONT,0,4);
box3d(411,445,634,475,WHITE,LIGHTBLUE,DARKGRAY,3);
print_text(427,449,"KMIT 'L",BLACK,TRIPLEX_FONT,0,1);
print_text(502,447,"Logic Analyzer",BLACK,GOTHIC_FONT,0,1);
print_text(426,448,"KMIT 'L",LIGHTGRAY,TRIPLEX_FONT,0,1);
print_text(501,446,"Logic Analyzer",LIGHTGRAY,GOTHIC_FONT,0,1);
print_text(425,447,"KMIT 'L",YELLOW,TRIPLEX_FONT,0,1);
print_text(500,445,"Logic Analyzer",YELLOW,GOTHIC_FONT,0,1);

box3d(6,401,69,406,DARKGRAY,LIGHTGRAY,WHITE,2);
box3d(zoom_bar1,401,zoom_bar2,406,WHITE,LIGHTGRAY,DARKGRAY,2);
setusercharsize(10,110,10,30);
print_text(5,388,"min",BLACK,SMALL_FONT,0,4);
print_text(55,388,"max",BLACK,SMALL_FONT,0,4);
print_text(8,409,"zoom scale",BLACK,SMALL_FONT,0,4);

size=imagesize(627,0,639,400);
if((bitimage=malloc(size))==NULL)
  {
   closegraph();
   printf("\n   Not enough memory for fun PULSE.EXE !");
   printf("\nplease free memory before run again.");
   getch();
   exit(0);
  }
getimage(627,0,639,395,bitimage);

LBO:

disp_block(320,400);
if(state==0)
  {
   box3d(5,450,53,474,WHITE,LIGHTGRAY,DARKGRAY,2);
   print_text(9,455,"zoom",BLACK,0,0,1);
   print_text(9,465," In ",BLACK,0,0,1);
   if(zoom_max)
        {
         print_text(9,455,"zoom",DARKGRAY,0,0,1);
         print_text(9,465," In ",DARKGRAY,0,0,1);
         mark_text(5,450,53,474,LIGHTGRAY);
```

```
        }
    box3d(58,450,106,474,WHITE,LIGHTGRAY,DARKGRAY,2);
    print_text(62,455,"zoom",BLACK,0,0,1);
    print_text(62,465," Out",BLACK,0,0,1);
    if(zoom_min)
         {
          print_text(62,455,"zoom",DARKGRAY,0,0,1);
          print_text(62,465," Out",DARKGRAY,0,0,1);
          mark_text(58,450,106,474,LIGHTGRAY);
         }
   }

  box3d(111,450,159,474,WHITE,LIGHTGRAY,DARKGRAY,2);
  print_text(120,465,"Back",BLACK,0,0,1);
  if(flag_start)
   {
    print_text(120,465,"Back",DARKGRAY,0,0,1);
    mark_text(111,450,159,474,LIGHTGRAY);
   }
  box3d(164,450,212,474,WHITE,LIGHTGRAY,DARKGRAY,2);
  print_text(173,465,"Next",BLACK,0,0,1);
  if(flag_end)
   {
    print_text(173,465,"Next",DARKGRAY,0,0,1);
    mark_text(164,450,212,474,LIGHTGRAY);
   }

  if(state==1)
   {
    print_text(9,455,"zoom",DARKGRAY,0,0,1);
    print_text(9,465," In ",DARKGRAY,0,0,1);
    mark_text(7,456,52,473,LIGHTGRAY);
    print_text(62,455,"zoom",DARKGRAY,0,0,1);
    print_text(62,465," Out",DARKGRAY,0,0,1);
    mark_text(58,448,106,474,LIGHTGRAY);
    print_text(226,465,"More",DARKGRAY,0,0,1);
    mark_text(217,450,265,474,LIGHTGRAY);
    print_text(276,465,"Print",DARKGRAY,0,0,1);
    mark_text(270,450,318,474,LIGHTGRAY);
    print_hex(38,3);
    goto LB2;
   }

  LB1:

  savedata=1,flag0=1,flag1=1;
  plot_line(x,y,endy,length,40,n);
  setcolor(BLACK);
  line(x,y-14,x,379);
  putimage(627,0,bitimage,COPY_PUT);

  LB2:

  key1=toupper(getch());
  if(key1==0)
   {
    key2=getch();
    switch(key2)
         {
          case 71 :    // HOME
           {
            flag_start=1;
            flag_end=0;
            print_text(120,465,"Back",DARKGRAY,0,0,1);
            mark_text(111,450,159,474,LIGHTGRAY);
            if(state!=1)
             {
              clear_line(x,y,endy,length,space,n);
```

```
            endbyte=(total);
           }
          if(state==1) endbyte=4;
          startbyte=0;
          goto LBO;
         }

        case 79 :   // END
         {
          flag_end=1;
          flag_start=0;
          print_text(173,465,"Next",DARKGRAY,0,0,1);
          mark_text(164,450,212,474,LIGHTGRAY);
          if(state!=1) clear_line(x,y,endy,length,space,n);
          switch(total)
           {
            case 140 : { startbyte=byte_of_file-139; break; }
            case 24  : { startbyte=byte_of_file-13;  break; }
            case 13  : { startbyte=byte_of_file-12;  break; }
            case 9   : { startbyte=byte_of_file-8;   break; }
            case 6   : { startbyte=byte_of_file-5;   break; }
            case 5   : { startbyte=byte_of_file-4;   break; }
            case 4   : { startbyte=byte_of_file-3;   break; }

           }
          if(state==1) { startbyte=byte_of_file-5; }
          endbyte=byte_of_file;
          goto LBO;
         }
       }
  }

switch(key1)
  {
   case 'l' :
       {
        if(state==1)    { goto LBO; }
        if(zoom_max==1) { goto LBO; }
        zoom_min=0;
        print_text(62,455,"zoom",BLACK,0,0,1);
        print_text(62,465," Out",BLACK,0,0,1);
        box3d(5,450,53,474,DARKGRAY,LIGHTGRAY,WHITE,2);
        print_text(11,455,"zoom",BLACK,0,0,1);
        print_text(11,465," In ",BLACK,0,0,1);
        delay(PUSH);
        box3d(5,450,53,474,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(9,455,"zoom",BLACK,0,0,1);
        print_text(9,465," In ",BLACK,0,0,1);

        box3d(zoom_bar1,401,zoom_bar2,406,LIGHTGRAY,LIGHTGRAY,LIGHTGRAY,2);
        box3d(6,401,69,406,DARKGRAY,LIGHTGRAY,WHITE,2);
        box3d(zoom_bar1+=10,401,zoom_bar2+=10,406,WHITE,LIGHTGRAY,DARKGRAY,2);

        clear_line(x,y,endy,length,space,n);
        length+=grid;
        switch(length)
         {
          case 1  : { total=140; break; }
          case 6  : { total=24;  break; }
          case 11 : { total=13;  break; }
          case 16 : { total=9;   break; }
          case 21 : { total=6;   break; }
          case 26 : { total=5;   break; }
          case 31 : { total=4;   break; }
         }
        flag_end=0;
        endbyte=(total+startbyte);
        if(length>=31)
```

```
        {
         length=31;
         zoom_max=1;
        }
       goto LB0;
      }

  case 'O' :
      {
       if(state==1)    { goto LB0; }
       if(zoom_min==1)  { goto LB0; }
       zoom_max=0;
       print_text(9,455,"zoom",BLACK,0,0,1);
       print_text(9,465," In ",BLACK,0,0,1);
       box3d(58,450,106,474,DARKGRAY,LIGHTGRAY,WHITE,2);
       print_text(64,456,"zoom",BLACK,0,0,1);
       print_text(64,465," Out",BLACK,0,0,1);
       delay(PUSH);
       box3d(58,450,106,474,WHITE,LIGHTGRAY,DARKGRAY,2);
       print_text(62,455,"zoom",BLACK,0,0,1);
       print_text(62,465," Out",BLACK,0,0,1);

       clear_line(x,y,endy,length,space,n);

       box3d(zoom_bar1,401,zoom_bar2,406,LIGHTGRAY,LIGHTGRAY,LIGHTGRAY,2);
       box3d(6,401,69,406,DARKGRAY,LIGHTGRAY,WHITE,2);
       box3d(zoom_bar1-=10,401,zoom_bar2-=10,406,WHITE,LIGHTGRAY,DARKGRAY,2);

       if(length<grid)
        {
         length=1;
         zoom_min=1;
         goto LB1;
        }
       length-=grid;
       switch(length)
        {
         case 1  : { total=140;  zoom_min=1; break; }
         case 6  : { total=24;   break; }
         case 11 : { total=13;   break; }
         case 16 : { total=9;    break; }
         case 21 : { total=6;    break; }
         case 26 : { total=5;    break; }
         case 31 : { total=4;    break; }
        }
       endbyte=(total+startbyte);
       goto LB0;
      }

  case 'N' :
      {
       if(flag_end==1)  { goto LB0; }
       flag_start=0;
       print_text(120,465,"Back",BLACK,0,0,1);
       box3d(164,450,212,474,DARKGRAY,LIGHTGRAY,WHITE,2);
       print_text(175,465,"Next",BLACK,0,0,1);
       delay(PUSH);
       box3d(164,450,212,474,WHITE,LIGHTGRAY,DARKGRAY,2);
       print_text(173,465,"Next",BLACK,0,0,1);
       if(state!=1)  clear_line(x,y,endy,length,space,n);
       if(state==1)
        {
         startbyte+=5;
         endbyte=startbyte+4;
         goto LB3;
        }
       switch(total)
        {
```

```
            case 140 : { startbyte+=140; break; }
            case 24  : { startbyte+=24;  break; }
            case 13  : { startbyte+=13;  break; }
            case 9   : { startbyte+=9;   break; }
            case 6   : { startbyte+=6;   break; }
            case 5   : { startbyte+=5;   break; }
            case 4   : { startbyte+=4;   break; }
           }
          endbyte=(total+startbyte);

          LB3:

          if(endbyte>=byte_of_file)
           {
            endbyte=byte_of_file;
            flag_end=1;
            print_text(173,465,"Next",DARKGRAY,0,0,1);
            mark_text(164,450,212,474,LIGHTGRAY);
           }

          goto LB0;
         }

    case 'B' :
          {
          if(flag_start==1)  { goto LB0; }
          flag_end=0;
          print_text(173,465,"Next",BLACK,0,0,1);
          box3d(111,450,159,474,DARKGRAY,LIGHTGRAY,WHITE,2);
          print_text(122,465,"Back",BLACK,0,0,1);
          delay(PUSH);
          box3d(111,450,159,474,WHITE,LIGHTGRAY,DARKGRAY,2);
          print_text(120,465,"Back",BLACK,0,0,1);

          if(state!=1)  clear_line(x,y,endy,length,space,n);
          if(state==1)
           {
            startbyte-=5;
            endbyte=startbyte+4;
            goto LB4;
           }
          switch(total)
           {
            case 140 : { startbyte-=140; break; }
            case 24  : { startbyte-=24;  break; }
            case 13  : { startbyte-=13;  break; }
            case 9   : { startbyte-=9;   break; }
            case 6   : { startbyte-=6;   break; }
            case 5   : { startbyte-=5;   break; }
            case 4   : { startbyte-=4;   break; }
           }
          endbyte=(total+startbyte);

          LB4:

          if(startbyte<=0)
           {
            startbyte=0;
            endbyte=total;
            flag_start=1;
            print_text(120,465,"Back",DARKGRAY,0,0,1);
            mark_text(111,450,159,474,LIGHTGRAY);
           }
          goto LB0;
         }
    case 'S' :
          {
```

```
        setfillstyle(pattern,LIGHTGRAY);
        bar(2,26,31,397);
        bar(3,392,71,420);
        setcolor(BLACK);
        line(x,y-14,x,379);
        box3d(5,432,53,444,DARKGRAY,LIGHTGRAY,WHITE,2);
        print_text(10,436,"State",BLACK,0,0,1);
        print_text(9,435,"State",WHITE,0,0,1);
        box3d(58,432,106,444,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(58,435,"Timing",WHITE,0,0,1);
        endbyte=startbyte+4;
        if(state==1) { goto LB0; }
        clear_line(x,y,endy,length,space,n);
        putimage(627,0,bitimage,COPY_PUT);
        state=1;
        goto LB0;
       }

   case 'T' :
        {
        box3d(4,28,29,39,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,28,ch[n],LIGHTBLUE,2,0,4);
        box3d(4,68,29,79,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,68,ch[n+1],LIGHTBLUE,2,0,4);
        box3d(4,108,29,119,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,108,ch[n+2],LIGHTBLUE,2,0,4);
        box3d(4,148,29,159,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,148,ch[n+3],LIGHTBLUE,2,0,4);
        box3d(4,188,29,199,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,188,ch[n+4],LIGHTBLUE,2,0,4);
        box3d(4,228,29,239,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,228,ch[n+5],LIGHTBLUE,2,0,4);
        box3d(4,268,29,279,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,268,ch[n+6],LIGHTBLUE,2,0,4);
        box3d(4,308,29,319,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(05,308,ch[n+7],LIGHTBLUE,2,0,4);
        box3d(4,343,29,364,WHITE,LIGHTGRAY,DARKGRAY,2);
        setusercharsize(10,110,10,30);
        print_text(5,343,"Time",RED,SMALL_FONT,0,4);
        print_text(5,353,"Base",RED,SMALL_FONT,0,4);

        print_text(226,465,"More",BLACK,0,0,1);
        print_text(276,465,"Print",BLACK,0,0,1);

        box3d(6,401,69,406,DARKGRAY,LIGHTGRAY,WHITE,2);
        box3d(zoom_bar1,401,zoom_bar2,406,WHITE,LIGHTGRAY,DARKGRAY,2);
        setusercharsize(10,110,10,30);
        print_text(5,388,"min",BLACK,SMALL_FONT,0,4);
        print_text(55,388,"max",BLACK,SMALL_FONT,0,4);
        print_text(8,409,"zoom scale",BLACK,SMALL_FONT,0,4);

        box3d(58,432,106,444,DARKGRAY,LIGHTGRAY,WHITE,2);
        print_text(59,436,"Timing",BLACK,0,0,1);
        print_text(58,435,"Timing",WHITE,0,0,1);
        box3d(5,432,53,444,WHITE,LIGHTGRAY,DARKGRAY,2);
        print_text(9,435,"State",WHITE,0,0,1);
        if(state==0) { goto LB0; }
        for(a=38,b=3;b<=23;b++)
         {
          gotoxy(a,b);
          printf("        ");
         }
        endbyte=total+startbyte;
        state=0;
        goto LB0;
       }
   case 'Q' :
```

```
        {
         box3d(323,464,371,474,DARKGRAY,LIGHTGRAY,WHITE,2);
         print_text(334,466,"Quit",BLACK,0,0,1);
         delay(PUSH);
         box3d(323,464,371,474,WHITE,LIGHTGRAY,DARKGRAY,2);
         print_text(332,466,"Quit",BLACK,0,0,1);
         if(win_massage(180,80,295,"Quit program ?","ออกจาก  Logic Analyzer ?","","",1,0))
          {
           free(bitimage);
           closegraph();
           exit(0);
          }
         goto LB0;
        }

    case 'P' :
        {
         nx=380;
         ny=469;
         if (state==1)  { goto LB0; }
         box3d(270,450,318,474,DARKGRAY,LIGHTGRAY,WHITE,2);
         print_text(278,465,"Print",BLACK,0,0,1);
         delay(PUSH);
         box3d(270,450,318,474,WHITE,LIGHTGRAY,DARKGRAY,2);
         print_text(276,465,"Print",BLACK,0,0,1);
         ret_prn=1;
         free(bitimage);
         for(d=1;d==1;)
          {
           cleardevice();
           for(a=0,b=3,c=1;c<=8;b+=17,c++)
            print_text(a,b,ch[c],LIGHTBLUE,SMALL_FONT,0,4);
           plot_line(25,4,14,length,17,1);
           for(a=0,b=163,c=9;c<=16;b+=17,c++)
            print_text(a,b,ch[c],LIGHTBLUE,SMALL_FONT,0,4);
           plot_line(25,164,174,length,17,9);
           for(a=0,b=323,c=17;c<=24;b+=17,c++)
            print_text(a,b,ch[c],LIGHTBLUE,SMALL_FONT,0,4);
           plot_line(25,324,334,length,17,17);
           for(a=0,b=139,c=0;c<=2;b+=160,c++)
            print_text(a,b,"T.B.",RED,2,0,4);
           print_text(nx-100,ny,"Bit start is",WHITE,SMALL_FONT,0,4);
           settextstyle(SMALL_FONT,0,4);
           gprintf(&nx,&ny,"%u",startbyte*4);
           if(flag_end!=1)
            {
             setcolor(WHITE);
             setlinestyle(USERBIT_LINE,0x5555,1);
             switch(length)
                  {
                   case 1  : { line(585,0,585,479); break; }
                   case 6 :  { line(601,0,601,479); break; }
                   case 11 : { line(597,0,597,479); break; }
                   case 16 : { line(601,0,601,479); break; }
                   case 21 : { line(529,0,529,479); break; }
                   case 26 : { line(545,0,545,479); break; }
                   case 31 : { line(521,0,521,479); break; }
                  }
             setlinestyle(SOLID_LINE,0,1);
            }

           e=win_massage(180,80,310,"Printing","กดปุ่ม Ok เมื่อพร้อมพิมพ์...","","",1,0);
           if(!e) { break; }
           d=hardcopy(10,0);
          }
         pulse(41,25,43,length,40,5);
        }
```

```
    if(func==first_ch+8) { func=25,funct=9; }
    for(count=startbyte;count<=endbyte;count++)
        {
        for(data_no=0;data_no<=3;data_no++)
          {
          data=((ptr[func][count].logic >> (data_no*2)) & 3);
          setcolor(GREEN);
          if(savedata==2)
            {
              if(data==savedata) flag1=0;
              if(data!=savedata) flag1=1;
            }
          if(savedata==1)
            {
              if(data==savedata) flag0=0;
              if(data!=savedata) flag0=1;
            }
          if(data!=3) savedata=data;

          switch(data)
            {
              case 2 :
                {
                if(!flag1)
                  {
                   setcolor(BLACK);
                  }
                line(x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)),
                      x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)));
                setcolor(GREEN);
                line(x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)),
                      x+(length*((((count-startbyte)*4)+data_no)+1)),y+(space*(funct-1)));
                break;
                }
              case 1 :
                {
                if(!flag0)
                  {
                   setcolor(BLACK);
                  }
                line(x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)),
                      x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)));
                setcolor(GREEN);
                line(x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)),
                      x+(length*((((count-startbyte)*4)+data_no)+1)),endy+(space*(funct-1)));
                }
            }
          }
        }
    }
}

/*-------------------------------*/

void clear_line(int x,int y,int endy,int length,int space,int first_ch)

{
  int func,funct;
  int data_no,data;
  int f;

  setcolor(BLACK);
  switch(first_ch)
    {
     case 1  : { f=0; break; }
     case 9  : { f=1; break; }
     case 17 : { f=2; break; }
    }
```

```
}

for(func=first_ch;func<=first_ch+8;func++)
  {
   funct=(func-(8*f));
   if(func==first_ch+8)  { func=25,funct=9; }
   endbyte=(total+startbyte);
   for(count=startbyte;count<=endbyte;count++)
        {
         for(data_no=0;data_no<=3;data_no++)
          {
           data=((ptr[func][count].logic >> (data_no*2)) & 3);
           switch(data)
             {
              case 2 :
                   {
                   line(x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)),
                       x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)));
                   line(x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)),
                       x+(length*((((count-startbyte)*4)+data_no)+1)),y+(space*(funct-1)));
                   break;
                   }
              case 1 :
                   {
                   line(x+(length*(((count-startbyte)*4)+data_no)),y+(space*(funct-1)),
                       x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)));
                   line(x+(length*(((count-startbyte)*4)+data_no)),endy+(space*(funct-1)),
                       x+(length*((((count-startbyte)*4)+data_no)+1)),endy+(space*(funct-1)));
                   break;
                   }
             }
          }
        }
  }
}

/*-----------------------------*/

void print_hex(int x,int y)

{
  unsigned int dat_l[5][4];
  unsigned int dat_h[5][4];
  int b,c,hi_z;
  int i,j,k;
  static int data_no,data;
  int func,func_l;

  endbyte=startbyte+4;
  for(j=0;j<=4;j++)
    {
     for(k=0;k<=3;k++)
          {
          dat_l[j][k] = 0x0000;
          dat_h[j][k] = 0x0000;
          }
    }

  for(i=startbyte;i<=endbyte;i++)
    {
     for(data_no=0;data_no<=3;data_no++)
          {
          hi_z=0;
          b=0x00;
          func_l=chanel;
          if(chanel>16) func_l=16;
          for(func=func_l;func>=1;func--)
           { endbyte=startbyte+4;
```

```
            data=((ptr[func][i].logic >> (data_no*2)) & 3);
            dat_l[(i-startbyte)][data_no] |= 0x01;
            switch(data)
              {
               case 1 : { b=0xFFFE; break; }
               case 2 : { b=0xFFFF; break; }
               case 3 : { hi_z=1; break; }
              }
            dat_l[(i-startbyte)][data_no] &= b;
            if(func!=1) { dat_l[(i-startbyte)][data_no] <<= 1; }
            if(func==1 && hi_z==1)
              {
               dat_l[(i-startbyte)][data_no]=0xffff;
               if(chanel<=16) dat_h[(i-startbyte)][data_no]=0xffff;
              }
           }

          if(chanel>16)
           {
            for(func=chanel;func>=17;func--)
              {
               data=((ptr[func][i].logic >> (data_no*2)) & 3);
               dat_h[(i-startbyte)][data_no] |= 0x01;
               switch(data)
                    {
                     case 1 : { b=0x00FE; break; }
                     case 2 : { b=0x00FF; break; }
                     case 3 : { hi_z=1; break; }
                    }
               dat_h[(i-startbyte)][data_no] &= b;
               if(func!=17) { dat_h[(i-startbyte)][data_no] <<= 1; }
               if(func==17 && hi_z==1) { dat_h[(i-startbyte)][data_no]=0xFFFF; }
              }
           }
      }
   }

  for(i=startbyte;i<=endbyte;i++)
    {
     for(data_no=0;data_no<=3;data_no++,y+=1)
          {
           gotoxy(x,y);
           printf("%.4X",dat_h[(i-startbyte)][data_no]);
           gotoxy(x+4,y);
           printf("%.4X",dat_l[(i-startbyte)][data_no]);
          }
    }
}

/*-----------------*/

void disp_block(int x,int y)

{
  int a,b;
  int nx,ny,ty;
  static int block;

  nx=x+305;
  ny=y+17;
  ty=ny+15;
  box3d(x,y,x+310,y+8,DARKGRAY,LIGHTGRAY,WHITE,1);
  if(endbyte>=byte_of_file)
    {
     box3d((x-1)+(60*5),y+1,x+9+(60*5),y+7,WHITE,LIGHTGRAY,DARKGRAY,2);
     goto END;
    }
  for(a=1;a<=60;a++)
```

```
  {
   b=1;
   if(a==1) { b=0; }
   if(startbyte>=(step*(a-1)+b) && startbyte<=(step*a))
        {
        block=a-1;
        a=61;
        }
   if(a!=61) { block=59; }
  }
 box3d((x+1)+(block*5),y+1,(x+11)+(block*5),y+7,WHITE,LIGHTGRAY,DARKGRAY,2);

END:

 box3d(x+250,y+15,x+310,y+25,DARKGRAY,BLACK,WHITE,2);
 print_text(nx-133,ny-3,"Bit start is",BLACK,SMALL_FONT,0,4);
 setcolor(YELLOW);
 settextstyle(0,0,1);
 gprintf(&nx,&ny,"%u",startbyte*4);
 print_text(nx-133,ty-4,"Total bits : ",BLACK,SMALL_FONT,0,4);
 setcolor(BLACK);
 settextstyle(0,0,1);
 gprintf(&nx,&ty,"%u",byte_of_file*4);

}

/*-------------------------*/

void help(int x,int y)
{
 char *hlp_msg[] = { "          *** Help screen ***",
                     " ",
                     " การใช้งาน Logicanalyzer",
                     " ",
                     "     การสั่งงานโปรแกรม ทำได้โดยการกดปุ่มตัว",
                     " อักษรตัวใหญ่ของแต่ละปุ่ม (เช่น Help ก็จะใช้ปุ่ม H)",
                     " โดยรายละเอียด และหน้าที่การทำงานของแต่ละปุ่ม มีดัง",
                     " ต่อไปนี้",
                     " ",
                     " \"State (S)\"",
                     " ",
                     "     ใช้ในการเปลี่ยนโหมด ไปยังโหมดของ State ",
                     "  (คือจะนำสถานะโลจิกของแต่ละ channel มาแปลงเป็น",
                     " เลขฐาน 16) ",
                     "     ในกรณีที่ข้อมูลใด มีบิทใดบิทหนึ่งเป็น Hi Impe-",
                     " dance ะแสดงผลข้อมูลเป็น FFFFFFFF",
                     " ",
                     " \"Timing (T)\"",
                     " ",
                     "     ใช้ในการเปลี่ยนโหมด ไปยังโหมดของ Timing",
                     "  (คือจะแสดงสัญญาณเป็นรูป Timing diagram) โดยปุ่ม",
                     "  นี้จะใช้สลับการทำงานกับปุ่ม State",
                     "     ในกรณีที่สัญญาณเป็น Hi Impedance สัญญาณจะ",
                     " ถูกแสดงผลเป็นช่องว่าง ๆ แทนที่จะเป็นระดับโลจิก \"1\"",
                     " หรือ \"0\"",
                     " ",
                     " \"zoom in (I)\"",
                     " ",
```

```
"     เป็นปุ่มที่ใช้ในโหมด Timing ซึ่งใช้ในการขยาย",
" ความกว้างของสัญญาณ ให้มีขนาดกว้างขึ้นเพื่อให้สามารถ",
" ดูสัญญาณได้โดยละเอียด ในกรณีที่สัญญาณมีความถี่สูง ๆ",
"     โดยในการ zoom นั้นจะทำได้ทั้งหมด 7 ระดับ",
" ",
" \"zoom Out (O)\"",
" ",
"     ทำหน้าที่ตรงข้ามกับปุ่ม zoom In คือจะลดความ",
" กว้างของรูปสัญญาณลง เพื่อให้สามารถดูสัญญาณโดยรวม",
" ได้ และใช้ปรับเพื่อให้ไปดูสัญญาณที่จุดอื่น ๆ ได้เร็วขึ้น",
" ",
" \"Next (N)\"",
" ",
"     ใช้ในการเลื่อนหน้าจอไปดูสัญญาณที่อื่น ๆ โดยจะ",
" ทิศทางไปข้างหน้า และจะเลื่อนทีละหนึ่งหน้าจอ (ในการ",
" เลื่อนหน้าจอนั้นจะแสดงสัญญาณส่วนท้ายของจอภาพซ้ำเดิม",
" เล็กน้อย เพื่อให้สัญญาณที่แสดงออกมา มีความต่อเนื่องกัน",
" และถูกต้อง) ใน Timing Mode",
"     ส่วนใน State Mode นั้น จะใช้ในการเลื่อนดู",
" ข้อมูลที่เป็นเลขฐาน 16 ถัดไปข้างหน้าทีละ 20 ข้อมูล",
" ",
" \"Back (B)\"",
" ",
"     ทำหน้าที่กลับกับปุ่ม Next คือใน Timing mode",
" จะทำการเลื่อนหน้าจอถอยกลับทีละหนึ่งหน้าจอ",
"     ส่วนใน State Mode ก็จะเลื่อนดูข้อมูลถอยหลัง",
" ไปทีละ 20 ข้อมูล",
" ",
" \"More (M)\"",
" ",
"     ใช้เลือก channel ของสัญญาณใน Timing -",
" mode ว่าจะดูที่ channel ใด (หนึ่งหน้าจอจะสามารถ",
" แสดงผลได้ทีละ 8 channel ดังนั้น จะเป็นการเลือกว่า",
" จะให้แสดงที่ channel 1-8, channel 9-16, หรือ",
" channel 17-24)",
" ",
" \"Print (P)\"",
" ",
"     เป็นปุ่มสั่งพิมพ์รูปสัญญาณใน Timing Mode โดย",
" จะทำการพิมพ์ครั้งละหนึ่งหน้าจอทั้งหมด 24 channel",
" พร้อม ๆ กัน (จะทำการพิมพ์สัญญาณ ในหน้าจอที่กำลัง",
" แสดงผลอยู่ในปัจจุบัน และเมื่อพิมพ์เสร็จหนึ่งหน้าจอแล้ว",
" จะพิมพ์หน้าต่อไปหรือไม่ ก็ได้)",
"     เส้นประที่เห็นในการพิมพ์คือจุดสิ้นสุดของสัญญาณ",
" ในหนึ่งหน้าจอจริง ๆ เพราะจะมีสัญญาณบางส่วนซ้ำเดิม",
" ดังที่ได้กล่าวมาแล้ว",
" ",
" \"Help (H)\"",
" ",
"     เป็นปุ่มขอความช่วยเหลือ ในการใช้งาน โปร",
" แกรม Logic Analyzer โดยจะใช้ปุ่มลูกศรเลื่อนหน้า",
```

```
                    " จอขึ้นลงทีละบรรทัด หรือใช้ปุ่ม PageUp, PageDown",
                    " เพื่อเลื่อนขึ้นลงทีละหนึ่งหน้าจอก็ได้ และสามารถใช้ปุ่ม",
                    " Home, End เพื่อไปยังบรรทัดแรก, บรรทัดสุดท้าย ตาม",
                    " ลำดับ และปิด Help ด้วยปุ่ม ESC",
                    " ",
                    " \"Quit\"",
                    " ",
                    "      ออกจากโปรแกรม Logic Analyzer",
                    " ",
                    " ----------",
                    " Trip !",
                    " ",
                    "      - เราสามารถกระโดดไปยังจุดสิ้นสุดของสัญญาณ",
                    " และ จุดเริ่มต้นของสัญญาณ อย่างรวดเร็วได้โดยใช้ปุ่ม",
                    " Home และ End ตามลำดับ (ใช้ได้ทั้ง Timing Mode",
                    " และ State Mode",
                    " ",
                    "      - ระหว่างทำการพิมพ์สัญญาณอยู่นั้น เราสามารถ",
                    " หยุดการพิมพ์ชั่วคราว หรือยกเลิกการพิมพ์ได้ ด้วยการกด",
                    " ปุ่มใด ๆ",
                    " " };

 int row,line=0,end_line=11,key1,key2;
 unsigned int size;
 void *saveimage;
 settextstyle(0,0,1);
 size=imagesize(x-4,y-4,x+369,y+289);
 if((saveimage=malloc(size))==NULL)
   {
    closegraph();
    printf("\n   Not enough memory for fun PULSE.EXE !");
    printf("\nplease free memory before run again.");
    getch();
    exit(0);
   }
 getimage(x-4,y-4,x+369,y+289,saveimage);
 setcolor(BLACK);
 rectangle(x-4,y-4,x+369,y+289);
 box3d(x,y,x+365,y+285,LIGHTGRAY,WHITE,LIGHTGRAY,3);
 setcolor(BLACK);
 rectangle(x,y,x+365,y+285);
 box3d(x,y,x+365,y+20,LIGHTGRAY,BLUE,DARKGRAY,0);
 setcolor(WHITE);
 outtextxy(x+160,y+7,"Help");
 box3d(x+2,y+2,x+19,y+18,WHITE,LIGHTGRAY,DARKGRAY,1);
 touttextxy(x+7,y-1,BLACK,"~");

 OUT_MSG :

 for(row=0;line<=end_line;line++,row++)
   {
    touttextxy(x+5,y+20+(row*22),BLACK,hlp_msg[line]);
   }
 line-=12;

 GETKEY :

 key1=getch();
 if(key1==0)
   {
    key2=getch();
```

```
    switch(key2)
        {
         case 80 :   // DOWN
          {
           if(line==88) goto GETKEY;
           line++;
           end_line++;
           if(line>88) { line=88; end_line=99; }
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         case 72 :   // UP
          {
           if(line==0) goto GETKEY;
           line--;
           end_line--;
           if(line<0) { line=0; end_line=11; }
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         case 81 :  // PAGE DOWN
          {
           if(line==88) goto GETKEY;
           line+=12;
           end_line+=12;
           if(line>88) { line=88; end_line=99; }
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         case 73 :  // PAGE UP
          {
           if(line==0) goto GETKEY;
           line-=12;
           end_line-=12;
           if(line<11) { line=0; end_line=11; }
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         case 79 :  // END
          {
           if(line==88) goto GETKEY;
           line=88;
           end_line=99;
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         case 71 :   // HOME
          {
           if(line==0) goto GETKEY;
           line=0;
           end_line=11;
           box3d(x+1,y+21,x+364,y+284,WHITE,WHITE,WHITE,0);
           goto OUT_MSG;
          }
         default : break;
        }
    }

   if(key1==27)
    {
     putimage(x-4,y-4,saveimage,COPY_PUT);
     free(saveimage);
    }
   else goto GETKEY;
}
/*-------------------------*/
/*----- Routine for printing -----*/
```

```
void copybyte(int byte)
{
 int lpt=0;            /* print to LTP1 */
 biosprint(0,byte,lpt);
}

void copydot(int scan,int pmode)
{
 int bit[8]={128,64,32,16,8,4,2,1};
 int bitno,printbyte=1,maxy,piccode,n1,n2,quit=0;
 int bkcolor=getbkcolor();
 maxy=getmaxy()+1;
 if(printbyte!=0)
  {
   maxy=maxy++;
   n2=maxy/256;
   n1=maxy%256;
   copybyte(27);copybyte(51);copybyte(24);  /* line space = 24/216 */
   copybyte(27);copybyte(42);copybyte(pmode);copybyte(n1);copybyte(n2);
   for(piccode=maxy-1;piccode>=0;piccode--)
        {
         printbyte=0;
         for(bitno=0;bitno<=7;bitno++)
          {
           if(getpixel(scan+bitno,piccode)!=bkcolor)
            printbyte+=bit[bitno];
          }
         copybyte(printbyte);
        }
  }
}

/*-------------------------*/

int hardcopy(int left,int ptimes)
{
 int scanline,status;
 int con_print;
 int i;
 status=biosprint(2,0,0);
 if(status!=144)
  {
   win_massage(180,80,295,"Printer error !","เครื่องพิมพ์ไม่พร้อม","กดปุ่ม Ok
เพื่อยกเลิกการพิมพ์","",0,1);
   return(-1);
  }
 copybyte(27);copybyte(108);copybyte(left);      /* set left margin */
 scanline=0;
 do
  {
   if(kbhit())
        {
         i=win_massage(180,80,260,"Interupt printing !","แน่ใจหรือว่าต้องการหยุดพิมพ์ ?","","",1,1);
         if(i==1)
          {
           copybyte(12);              /* form feed */
           return(-1);
          }
        }
   for(i=0;i<=ptimes;i++)      /* print X times */
        {
         copybyte(13);             /* move printerhead to left margin */
         copydot(scanline,PMODE4);  /* defaulf PMODE4 */
        }
   scanline+=8;
```

```
    copybyte(10);              /* line feedup */
   }
  while(scanline<(getmaxx()+1));
  copybyte(27);copybyte(2);    /* restore line space (1/6 incs) */
  copybyte(12);              /* form feed */
  if(!flag_end)
    {
    con_print=win_massage(200,100,280,"Continue printing ?","ต้องการพิมพ์หน้าต่อไปหรือไม่ ?","","",1,0);
    if(con_print==1)
         {
         copybyte(10);              /* line feedup */
         switch(total)
           {
           case 140 : { startbyte+=140; break; }
           case 13  : { startbyte+=13;  break; }
           case 6   : { startbyte+=6;   break; }
           case 4   : { startbyte+=4;   break; }
           case 3   : { startbyte+=3;   break; }
           case 2   : { startbyte+=2;   break; }
           }
         endbyte=(total+startbyte);
         if(endbyte>byte_of_file)
           {
           endbyte=byte_of_file;
           flag_end=1;
           }
         }
    }
  return(con_print);
}
```

TTL.ASM

```
outport macro port,data
                mov  dx,300h
                mov  al,port
                out  dx,al
                mov  dx,30fh
                mov  al,data
                out  dx,al
                endm
inport macro port
                mov  dx,300h
                mov  al,port
                out  dx,al
                mov  dx,30fh
                in   al,dx
                endm
fn_colour equ 01111111b
cseg segment public
assume cs:cseg,ds:dseg
extrn mov_cursor:near,get_cursor:near,cursor_on:near
extrn window:near,write_char:near,write_text:near
extrn cursor_off:near,send_crlf:near,write_table:near
extrn write_table1:near,read_key:near
menu proc near
                mov  ax,dseg
                mov  ds,ax
                mov  ax,0003h
                int  10h
                outport 13h,88h
                outport 10h,80h
                outport 11h,80h
                inport  10h
                cmp  al,80h
                je   work
                jmp  nohard
work:             call cursor_off
                mov  bh,17h
                mov  ch,3
                mov  cl,5
                mov  dh,21
                mov  dl,74
                call window
                mov  dh,3
                mov  dl,5
                call mov_cursor
                mov  bl,17h
                mov  bh,19
                mov  cl,70
                call write_table
                mov  ch,4
                mov  cl,8
                mov  bh,70h
                mov  dh,4
                mov  dl,71
                call window
                mov  dh,4
                mov  dl,28
                call mov_cursor
                mov  bl,7eh
                lea  si,logo
                call write_text
                call write_fn_bar
                call write_des
                call read_key
end_ttl:          outport 11h,00h
```

```
                mov  ax,4c01h
                int  21h
nohard:           lea  dx,hard1
                mov  ah,9
                int  21h
                lea  dx,hard2
                int  21h
                lea  dx,hard3
                int  21h
                lea  dx,hard4
                int  21h
                mov  ah,7
                int  21h
                jmp  end_ttl
menu endp
;/*******************************************************
clear_fn proc near
                mov  bh,17h
                mov  ch,5
                mov  cl,8
                mov  dh,20
                mov  dl,23
                call window
                ret
clear_fn endp
;/*******************************************************
clear_des proc near
                mov  bh,17h
                mov  ch,5
                mov  cl,27
                mov  dh,20
                mov  dl,72
                call window
                ret
clear_des endp
;/*******************************************************
write_fn proc near
                mov  dh,6
                mov  dl,8
                call mov_cursor
                mov  bl,17h
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,7
                mov  dl,9
                call mov_cursor
                lea  si,fn1
                call write_text
                mov  dh,9
                mov  dl,8
                call mov_cursor
                mov  bl,17h
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,10
                mov  dl,9
                call mov_cursor
                lea  si,fn2
                call write_text
                mov  dh,0ch
                mov  dl,8
                call mov_cursor
                mov  bl,17h
                mov  bh,2
                mov  cl,15
                call write_table1
```

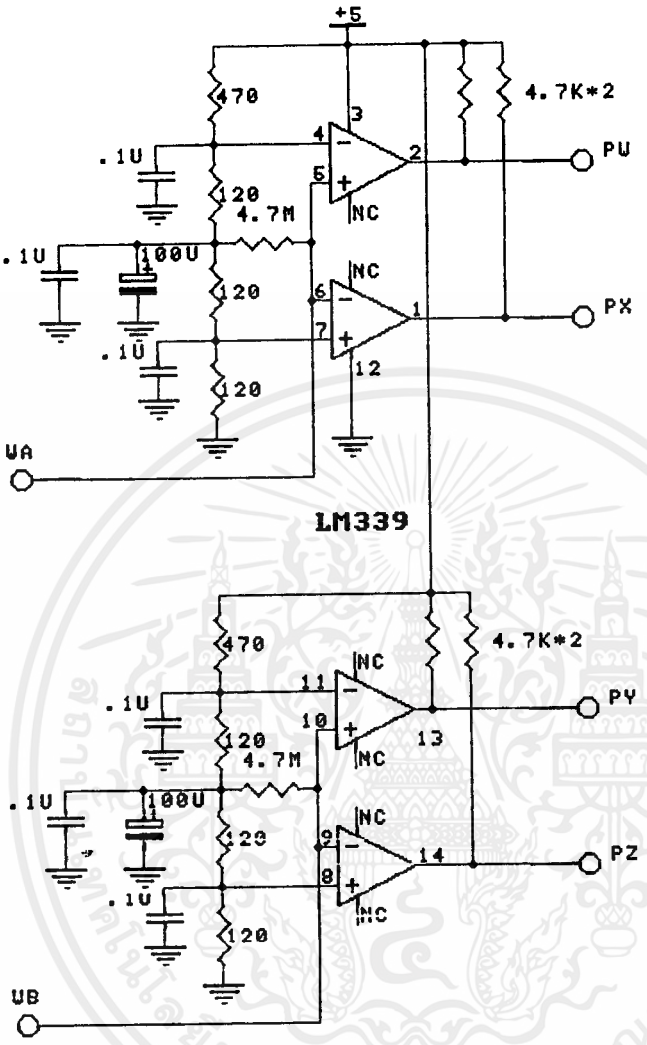
```
                mov  dh,0dh
                mov  dl,9
                call mov_cursor
                lea  si,fn3
                call write_text
                mov  dh,0fh
                mov  dl,8
                call mov_cursor
                mov  bl,17h
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,10h
                mov  dl,9
                call mov_cursor
                lea  si,fn4
                call write_text
                mov  dh,12h
                mov  dl,8
                call mov_cursor
                mov  bl,17h
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,13h
                mov  dl,9
                call mov_cursor
                lea  si,fn5
                call write_text
                ret
write_fn endp
;/*****************************************************
public write_des
write_des proc near
                call clear_des
                mov  dh,6
                mov  dl,27
                call mov_cursor
                mov  bl,17h
                mov  bh,14
                mov  cl,45
                call write_table1
                mov  dh,8
                mov  dl,30
                call mov_cursor
                cmp  ver,4
                je   des_4
                cmp  ver,3
                je   des_3
                cmp  ver,2
                je   des_2
                cmp  ver,1
                je   des_1
des_0:            mov  cx,6
                lea  si,des_num
des_01:           call write_text
                inc  dh
                call mov_cursor
                loop des_01
                jmp  ok_des
des_1:            mov  cx,6
                lea  si,des_serach
des_11:           call write_text
                inc  dh
                call mov_cursor
                loop des_11
                jmp  ok_des
des_2:            mov  cx,7
```

```
                lea  si,des_delay
des_21:           call write_text
                inc  dh
                call mov_cursor
                loop des_21
                jmp  ok_des
des_3:            mov  cx,5
                lea  si,des_start
des_31:           call write_text
                inc  dh
                call mov_cursor
                loop des_31
                jmp  ok_des
des_4:            mov  cx,5
                lea  si,des_exit
des_41:           call write_text
                inc  dh
                call mov_cursor
                loop des_41
ok_des:           mov  dh,19
                mov  dl,33
                call mov_cursor
                lea  dx,ttl
                mov  ah,9
                int  21h
                mov  cx,3
                lea  si,num_temp
                mov  bl,1eh
aa:              lodsb
                call write_char
                loop aa
                mov  dh,19
                mov  dl,35h
                call mov_cursor
                lea  dx,del_mes
                mov  ah,9
                int  21h
                mov  bl,1eh
                mov  al,del_buf
                call write_char
                ret
write_des endp
;/*****************************************************
public write_fn_bar
write_fn_bar proc near
                push ax
                call clear_fn
                call write_fn
                cmp  ver,4
                je   fbar4f
                cmp  ver,3
                je   fbar3
                cmp  ver,2
                je   fbar2
                cmp  ver,1
                je   fbar1
fbar0:            mov  dh,6
                mov  dl,8
                call mov_cursor
                mov  bl,fn_colour
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,7
                mov  dl,9
                call mov_cursor
                lea  si,fn1
                call write_text
```

```
                jmp  fbar_end
fbar1:            mov  dh,9
                mov  dl,8
                call mov_cursor
                mov  bl,fn_colour
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,10
                mov  dl,9
                call mov_cursor
                lea  si,fn2
                call write_text
                jmp  fbar_end
fbar4f:           jmp  fbar4
fbar2:            mov  dh,0ch
                mov  dl,8
                call mov_cursor
                mov  bl,fn_colour
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,0dh
                mov  dl,9
                call mov_cursor
                lea  si,fn3
                call write_text
                jmp  fbar_end
fbar3:            mov  dh,0fh
                mov  dl,8
                call mov_cursor
                mov  bl,fn_colour
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,10h
                mov  dl,9
                call mov_cursor
                lea  si,fn4
                call write_text
                jmp  fbar_end
fbar4:            mov  dh,12h
                mov  dl,8
                call mov_cursor
                mov  bl,fn_colour
                mov  bh,2
                mov  cl,15
                call write_table1
                mov  dh,13h
                mov  dl,9
                call mov_cursor
                lea  si,fn5
                call write_text
fbar_end:         pop  ax
                ret
write_fn_bar endp
;/*******************************************************
cseg ends
dseg segment public
EXTRN DEL_BUF:BYTE,DEL_MES:BYTE,NUM_TEMP:BYTE,TTL:BYTE
logo            db "TTL & CMOS TESTER V1.0",0
kmitl           db "      KMIT 'L",0
fn1             db "   ^Number   ",0
fn2             db "   Sea^rch   ",0
fn3             db "   ^Delay    ",0
fn4             db "   ^Start    ",0
fn5             db "   ^Exit     ",0
des_num           db "^S^e^l^e^c^t ^n^u^m^b^e^r ^e^q^u^i^p^m^e^n^t^.",0
```

```
                db 0
                db "   Before start the ttl/cmos tester you",0
                db "must be select number of ttl or cmos.",0
                db "   If you want to  choose number please",0
                db "press key ","^",11h,"^",0d9h," when you ready.",0
des_serach        db "^A^u^t^o ^d^e^t^e^c^t ^d^e^v^i^c^e",0
                db 0
                db "    If you don't know about number of",0
                db "TTL & CMOS device.  You can uses this",0
                db "menu for detect number of device by",0
                db " press ","^",11h,"^",0d9h," when you ready.",0
des_delay         db "^S^e^l^e^c^t ^D^e^l^a^y ^T^i^m^e",0
                db 0
                db "   Delay time,chosse when your computer",0
                db "run at hight speed.",0
                db "   Default is (5) for CPU 80486DX,if you",0
                db "want to change this value press  ","^",11h,"^",0d9h,0
                db "when you ready.",0
des_start       db "^D^o ^y^o^u ^w^a^n^t ^t^o ^s^t^a^r^t ^n^o^w^?",0
                db 0
                db "  If you want to run TTL & CMOS tester.",0
                db " Press key ","^",11h,"^",0d9h," when you ready.",0
                db "And press ^y for start testing.",0
des_exit        db "^D^o ^y^o^u ^w^a^n^t ^t^o ^e^x^i^t ^n^o^w^?",0
                db 0
                db "    You can leave this program by press",0
                db "key ","^",11h,"^",0d9h," when you ready.",0
                db "And press ^y for exit this program.",0
hard1           db 0dh,0ah,"TTL&CMOS Tester.$"
hard2           db 0dh,0ah,"Hardware not install or device error..!$"
hard3           db 0dh,0ah,"Copyright (c) 1995-1996 by KMIT'L Engineer.2N$"
hard4           db 0dh,0ah,"Press a key to continute.",0dh,0ah,"$"
public ver
ver             db 1 dup(00h)
dseg ends
end menu
```

ภาคผนวก ข.
วงจรของฮาร์ดแวร์



UA CONECT
U0,U2,U4,U6,U8
U10,U12,U14,U16
U18,U20,U22

UB CONECT
U1,U3,U5,U7,U9
U11,U13,U15,U17
U19,U21,U23

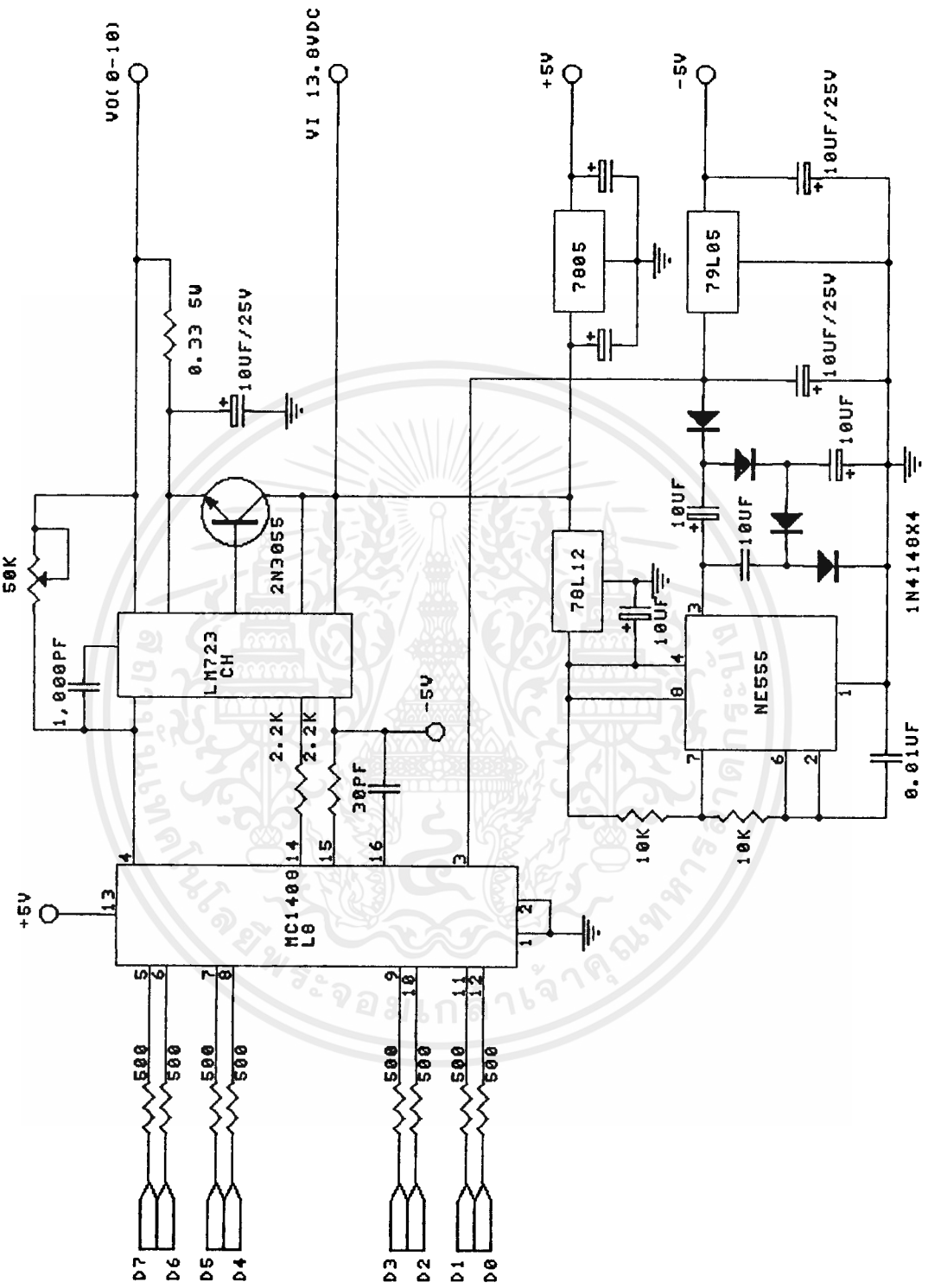
PW CONECT
PA0,PA4 IC12,IC13
PB0,PB4 IC12,IC13
PC0,PC4 IC12,IC13

PX CONECT
PA1,PA5 IC12,IC13
PB1,PB5 IC12,IC13
PC1,PC5 IC12,IC13

PY CONECT
PA2,PA6 IC12,IC13
PB2,PB6 IC12,IC13
PC2,PC6 IC12,IC13

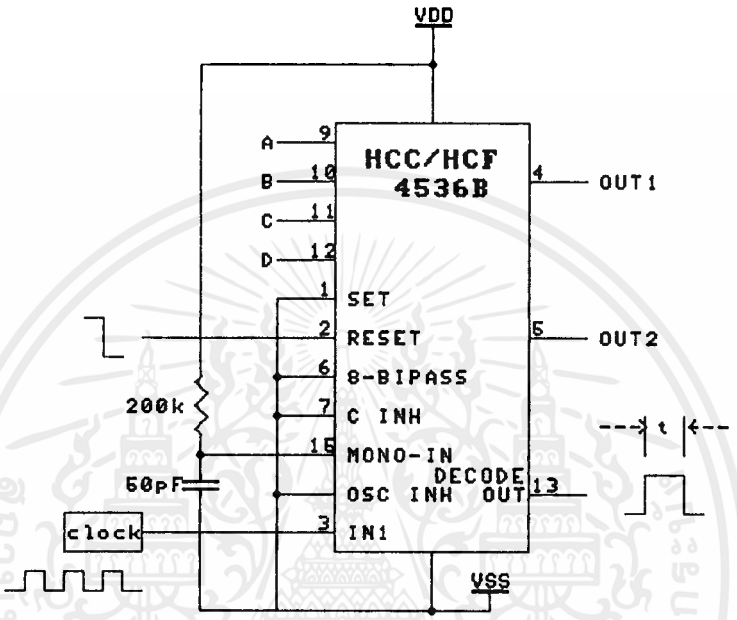
PZ CONECT
PA3,PA7 IC12,IC13
PB3,PB7 IC12,IC13
PC3,PC7 IC12,IC13

วงจร Comparator



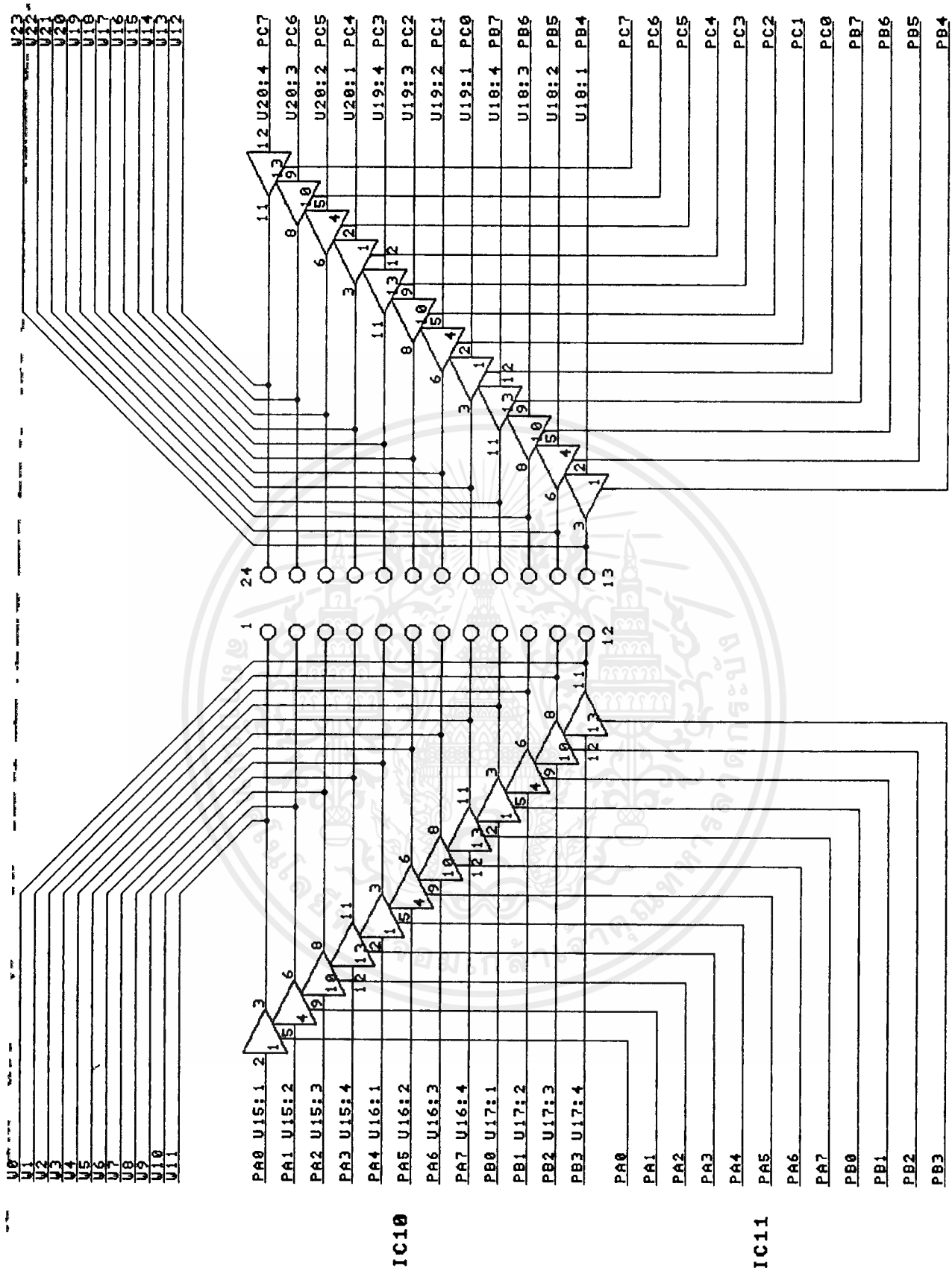
วงจรปรับแรงดันอ้างอิง หรือแรงดัน Thershold

(เป็นส่วนหนึ่งของวงจรหลัก)



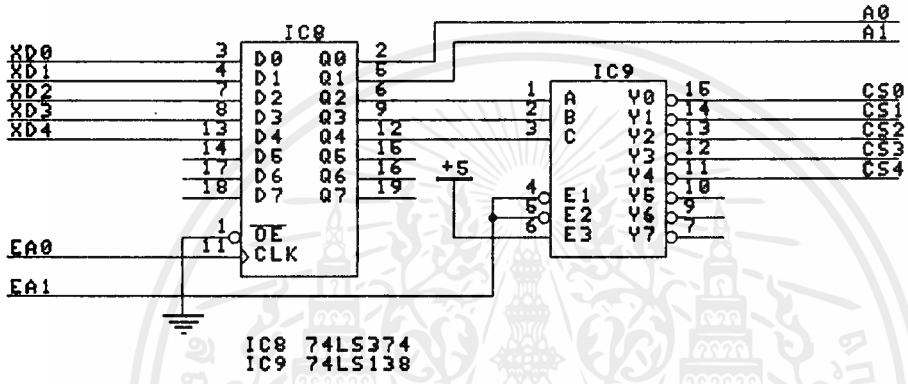
วงจรสร้างฐานเวลาหรือ Time base

(เป็นส่วนหนึ่งของวงจรหลัก)



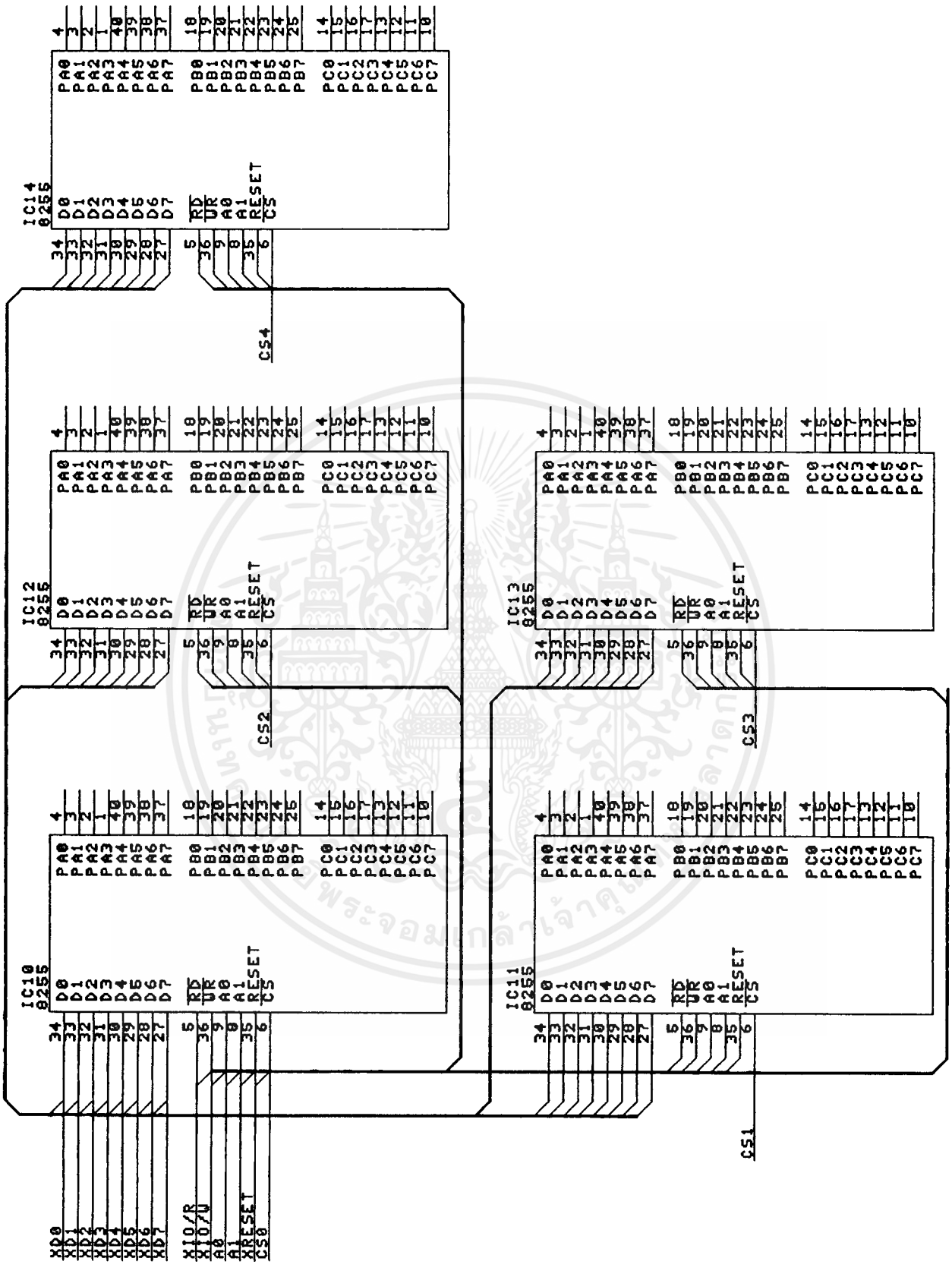
วงจรบัฟเฟอร์อินพุทเอาท์พุทของไอซีที่นำมาวิเคราะห์

(เป็นส่วนหนึ่งของวงจรหลัก)



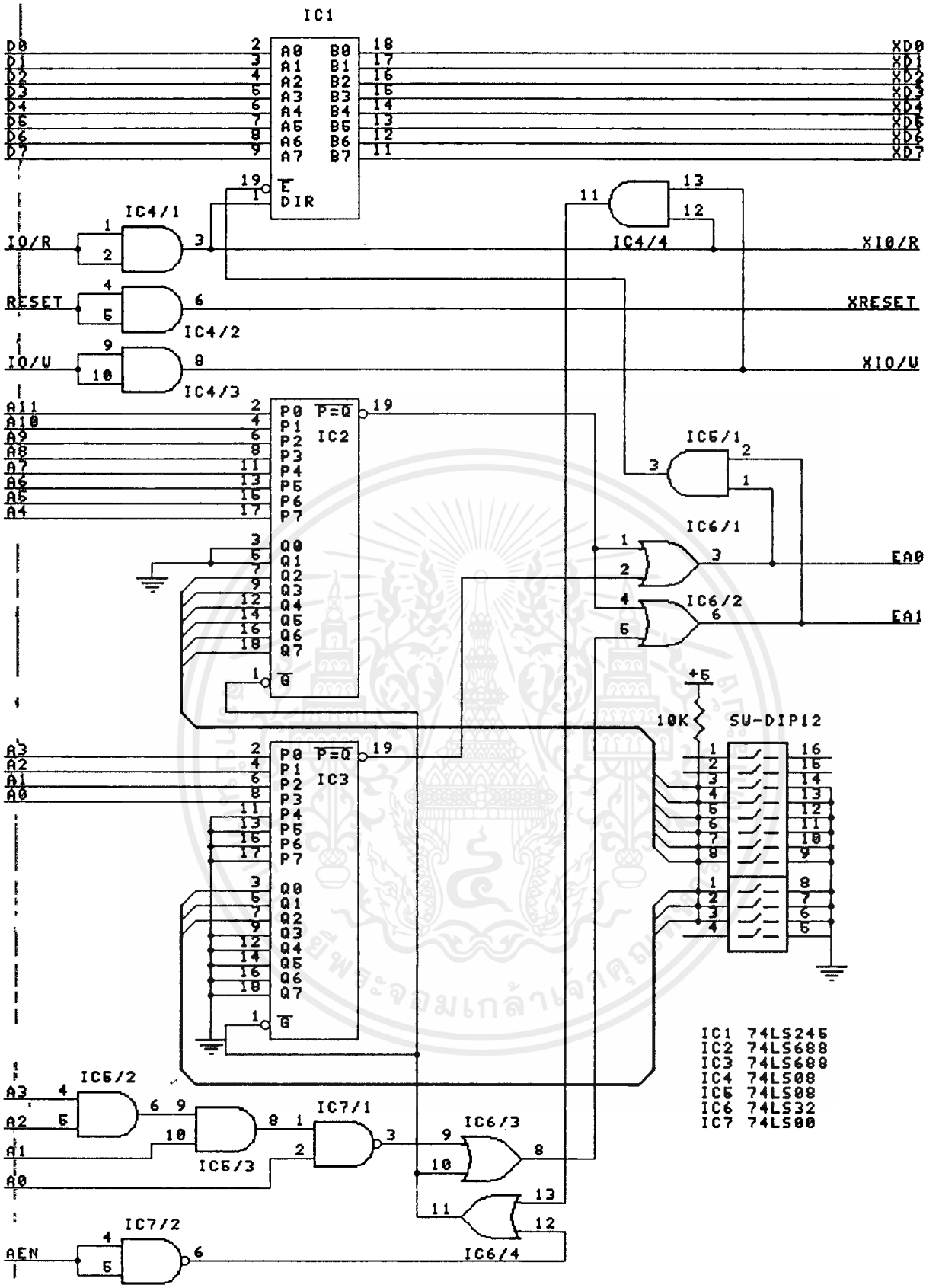
วงจรถอดรหัสเบอร์พอร์ท

(เป็นส่วนหนึ่งของวงจรหลัก)



วงจร 8255 I/O

(เป็นส่วนหนึ่งของวงจรหลัก)



วงจร Interface Card

Logic Diagram PAL16L8