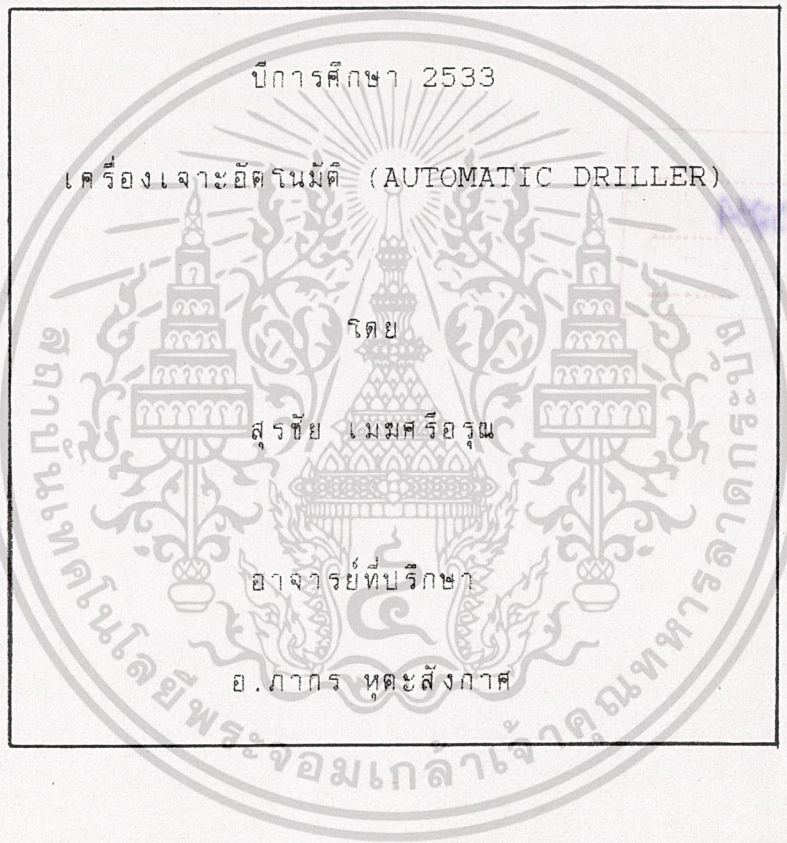


ปีการศึกษา 2533

เครื่องเจาะอัตโนมัติ (AUTOMATIC DRILLER)

โดย

สุรชัย เมฆศรีอรุณ

อาจารย์ที่ปรึกษา

อ.ภาคร ทุตะสังกาศ

ปริญญานิพนธ์ปีการศึกษา 2533

เรื่อง เครื่องเจาะอัตโนมัติ (AUTOMATIC DRILLER)

ผู้จัดทำ _____

1. นาย สุรชัย เมฆศรีอรุณ _____

2. _____

3. _____

_____อาจารย์ที่ปรึกษา

(_____)

_____อาจารย์ที่ปรึกษา

(_____)

_____อาจารย์ที่ปรึกษา

(_____)

เครื่องเจาะอัตโนมัติ (AUTOMATIC DRILLER)

โดย นาย สุรชัย เมฆศรีอรุณ
อาจารย์ที่ปรึกษา อ.ภาคร หุตะสังกาศ
ปีการศึกษา 2533

บทคัดย่อ

เครื่องเจาะอัตโนมัติเป็นการพัฒนางานทางด้านอุตสาหกรรมการผลิตอีกขั้นหนึ่ง เนื่องจากในปัจจุบัน งานด้านอุตสาหกรรมการผลิตต้องการความรวดเร็ว, ลดต้นทุนการผลิต (โดยเฉพาะแรงงาน) แต่ผลผลิตที่ได้ต้องมีคุณภาพตรงตามความต้องการไม่ผิดพลาด ดังนั้น อุปกรณ์ เครื่องมือเครื่องใช้ที่ใช้ในการผลิต จึงจำเป็นต้องมีความเที่ยงตรงแม่นยำ มีความผิดพลาดเกิดขึ้นน้อยมาก หรือไม่มีเลย เครื่องเจาะอัตโนมัตินี้ เป็นผลงานการพัฒนาของนักศึกษาภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ ซึ่งพัฒนาโดยประยุกต์เอาไมโครโปรเซสเซอร์ (MICROPROCESSOR) มาเป็นตัวควบคุมการทำงานให้เครื่องเจาะ ตัวควบคุมนี้เรียกอีกอย่างหนึ่งว่า ไมโครคอนโทรเลอร์ (MICROCONTROLLER) ซึ่งควบคุมเครื่องเจาะ ให้ทำงานในลักษณะอัตโนมัติ (AUTOMATION) ไมโครคอนโทรเลอร์นี้ทำงานด้วยความเที่ยงตรงแม่นยำและรวดเร็ว มีเสถียรภาพในการทำงาน จึงเหมาะที่จะนำไปควบคุมเครื่องเจาะที่ต้องทำงานเจาะติดต่อกันเป็นเวลานานๆ

AUTOMATIC DRILLER

By Surachai Meksriarun
Advisor: Phakorn Hutasungart
Year of Ed: 1990

ABSTRACT

The automatic driller is a product in industrial production development. Presently, industrial productions need to be done quickly and less costly (especially, the labour cost) while the quality of the productions has to be in accordance with the standards. Accordingly, the equipment to be used in the production process has to work accurately, with as little errors as possible or none. The automatic driller is the product of a student in the Section of Industrial Computer Technology, Department of Industrial Instrumentation Technology, the Faculty of Enginering. It was developed through an application of a microprocessor which controls the driller.The controller, called "microcontroller", automatically controls the drilling. It works accurately, Quickly and consistently. Thus, it is suitable tobe attached to a driller which would be used continously for a long period.

สารบัญ

บทที่ 1
บทนำ

เครื่องเจาะมีหลายประเภท เช่น เครื่องเจาะสำหรับเจาะโลหะ, เครื่องเจาะสำหรับเจาะไม้ แต่สำหรับโครงงานนี้ จะเป็นการพัฒนาเครื่องเจาะอัตโนมัติที่ใช้เจาะแผ่นวงจรพิมพ์ (PRINT CIRCUIT BOARD) ซึ่งเครื่องเจาะอัตโนมัตินี้ เหมาะกับงานเจาะแผ่นวงจรพิมพ์ ที่ไม่ใหญ่มากนัก คือมีขนาดใหญ่สุดที่ที่สามารถเจาะได้ประมาณ 1 ตารางฟุต และการทำงานของเครื่องเจาะอัตโนมัตินี้ สามารถทำงาน แบบเป็นเครื่องเจาะธรรมดาทั่วไป (MANUAL MODE) หรือทำงานในแบบอัตโนมัติ (PROGRAM MODE) ก็ได้ แต่การทำงานในแบบอัตโนมัตินี้ จำเป็นต้องทำการโปรแกรมให้แก่เครื่องเจาะอัตโนมัติรับรู้ในครั้งแรกเสียก่อน ต่อจากนั้นเมื่อต้องการจะทำการเจาะแผ่นวงจรพิมพ์ที่มีตำแหน่งการเจาะเหมือนกับแผ่นวงจรพิมพ์ต้นแบบ ที่ได้ทำการโปรแกรมแล้วนั้น ก็สามารถทำการเจาะได้โดยอัตโนมัติ โดยให้เครื่องทำการเจาะด้วยตัวเอง ดังนั้นจะเห็นว่าการทำงานในแบบอัตโนมัตินี้จะมีประโยชน์มาก ในการนำไปเจาะแผ่นวงจรพิมพ์ที่มีตำแหน่งการเจาะที่เหมือนกันแต่มีจำนวนมากๆ ซึ่งจะทำให้ประหยัดแรงงานคนที่จะต้องไปทำการเจาะด้วยตนเอง ในส่วนของการทำงานแบบเป็นเครื่องเจาะธรรมดาทั่วไป (MANUALMODE) นั้น ก็สามารถทำการเจาะได้สะดวกปลอดภัย โดยเพียงแต่นำแผ่นวงจรพิมพ์ไปวางยึดไว้บนแท่นเจาะ จากนั้นเมื่อต้องการเจาะรูบนแผ่นวงจรพิมพ์ก็เคลื่อนตัวสว่านไปยังตำแหน่งที่ต้องการจะเจาะ โดยใช้แป้นพิมพ์บนชุดควบคุมในการเคลื่อนตัวสว่าน เมื่อถึงตำแหน่งที่ต้องการจะเจาะ ก็เพียงแต่กดแป้นพิมพ์บนชุดควบคุมเบาๆ ตัวดอกสว่านก็จะเคลื่อนตัวลงเจาะรูบนแผ่นวงจรพิมพ์ทันที อย่างนุ่มนวล จะเห็นว่าสะดวกสบายและปลอดภัย ในการที่ตัวผู้เจาะไม่ต้องจับชิ้นงานด้วยมือตัวเอง อันอาจทำให้เกิดอันตรายได้จากการจับชิ้นงาน

ในโครงงานชุดเครื่องเจาะอัตโนมัตินี้ จะแบ่งการทำงานออกเป็นส่วนใหญ่ๆได้ 4 ส่วน ดังนี้คือ

1.1 ชุดควบคุม ซึ่งประกอบด้วยคอนโทรเลอร์บอร์ด (CONTROLLER BOARD) ที่ใช้ไมโครคอนโทรเลอร์เบอร์ 8052เอเอช เบสิก (8052AH BASIC) เป็น CPU ซึ่งคอนโทรเลอร์บอร์ดที่ใช้นี้มีพอร์ต (PORT) ที่สามารถต่อออกไปใช้งาน (USER PORT) อยู่ 3 พอร์ต หลักๆนอกเหนือจากพอร์ตที่ใช้ต่อกับส่วนแสดงผล, พอร์ตที่ใช้ต่อกับแป้นรับข้อมูลที่มีอยู่แล้ว พอร์ตใช้งาน

(USER PORT) 3 พอร์ตที่ว่านี้ จะนำไปต่อกับชุดวงจรขับสเตปปิ้งมอเตอร์ (STEPPING MOTOR DRIVER) ที่ใช้ในการเลื่อนแกนเจาะ ซึ่งประกอบด้วย แกนเอกซ์ (X-AXIS), แกนวาย (Y-AXIS), แกนแซด (Z-AXIS)

1.2 ชุดแท่นเจาะซึ่งมีขนาดประมาณครึ่งตารางเมตร ซึ่งชุดแท่นเจาะนี้ประกอบด้วย ลีดสกลู (LEAD SCREW) ยาวครึ่งเมตร 2 ตัว ประกอบกันเป็นตัวเลื่อนแกนเอ็กซ์และแกนวาย และใช้ลีดสกลูยาว 6 นิ้ว 1 ตัว เป็นตัวเลื่อนตัวสว่านเจาะทางแกนแซด และใช้สเต็ปปิ้งมอเตอร์ แบบ 4 เฟส มีมุมในการหมุน 1.8 องศาต่อสเตป ขนาด 5โวลท์ 3 แอมป์ จำนวน 3 ตัว เป็นตัวขับลีดสกลูให้เลื่อนไปตามจุดต่างๆได้ ส่วนตัวสว่านเจาะจะเป็นขนาดเล็ก แต่มีความเร็วรอบสูง (HIGH SPEED) มีน้ำหนักเบา

1.3 ชุดวงจรขับ (DRIVER) ประกอบด้วยชุดทรานซิสเตอร์ ที่นำกระแสสูงที่ใช้ในการขับสเตปปิ้งมอเตอร์ และชุดของทรานซิสเตอร์ที่ใช้เป็นสวิทช์ รีเลย์ (RELAY) ในการตัดต่อไฟที่ป้อนให้แก่ตัวสว่านเจาะ

1.4 ชุดโปรแกรมควบคุมการทำงาน (SOFT WARE) จะเขียนด้วยภาษาเบสิกของ 8052เอเอช เบสิก ซึ่งมีตัวแปลภาษาเบสิกอยู่ในตัว โดยการเขียนโปรแกรมควบคุมการทำงานนี้ต้องอาศัยคอมพิวเตอร์พีซี(PC-XT/AT) ที่มีพอร์ตสื่อสารแบบอนุกรมใช้ในการเขียนโปรแกรมบน 8052 คอนโทรเลอร์ (8052 CONTROLLER) โดยการต่อสายตามมาตรฐาน อาเอส-232ซี (RS-232C) ระหว่างพอร์ตสื่อสารอนุกรมของคอมพิวเตอร์พีซีกับพอร์ตสื่อสารอนุกรมของบอร์ด 8052 คอนโทรเลอร์

ดังนั้นจะเห็นว่าการทำงานของชุดเครื่องเจาะอัตโนมัตินี้ จะถูกควบคุมการทำงานโดยโปรแกรม ดังนั้นก็จะเป็นการง่ายและสะดวกในการพัฒนาระบบการทำงานของชุดเครื่องเจาะอัตโนมัตินี้ โดยเราเพียงแต่พัฒนาการเขียนโปรแกรมควบคุมการทำงานของเครื่องให้เป็นไปตามที่เราต้องการ โดยอาจจะไม่ต้องมีการเปลี่ยนแปลงทางด้านฮาร์ดแวร์ (HARD WARE) เลยก็ได้

หวังว่ารายงานฉบับนี้ จะสามารถใช้เป็นทั้งคู่มือในการอ้างอิงทางด้านเทคนิค และเป็นคู่มือการใช้งาน ของโครงงานเครื่องเจาะอัตโนมัตินี้ได้เป็นอย่างดี ทั้งนี้เพราะรายงานฉบับนี้จะได้กล่าวถึง ตั้งแต่ทฤษฎีและหลักการในการออกแบบและสร้างเครื่องเจาะอัตโนมัติ ตลอดจนโปรแกรมควบคุมระบบการทำงาน เพื่อเป็นแนวทางในการพัฒนาระบบการทำงานของเครื่องที่มีความซับซ้อนยิ่งขึ้นต่อไป รวมทั้งได้กล่าวถึง วิธีการใช้งานเครื่องเจาะอัตโนมัตินี้อย่างมีประสิทธิภาพอีกด้วย

บทที่ 2
ทฤษฎี

เกี่ยวกับสเต็ปปิ้งมอเตอร์

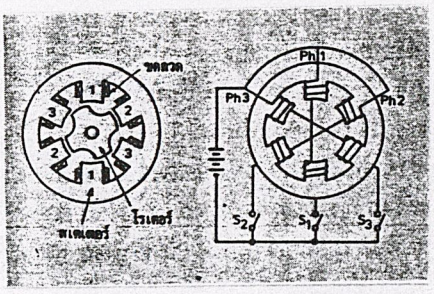
ในปัจจุบันนี้จะเห็นว่า สเต็ปปิ้งมอเตอร์มีใช้อยู่ในหลายๆงาน เช่น ในเครื่องพิมพ์,ใน X-Y พล็อตเตอร์,ในแขนกล, หรือในเครื่องถ่ายเอกสาร เพราะสเต็ปปิ้งมอเตอร์มีข้อได้เปรียบมอเตอร์ไฟตรงแบบธรรมดาคือ สเต็ปปิ้งมอเตอร์มีการควบคุมแบบหลูบเปิด ทำให้ง่ายต่อการควบคุม และเรายังสามารถที่จะรู้ตำแหน่งของมอเตอร์ได้ตลอดเวลาอย่างแน่นอนแม่นยำ ดังนั้นงานที่ต้องการควบคุมตำแหน่งที่แน่นอนจึงมักเลือกใช้สเต็ปปิ้งมอเตอร์

การควบคุมสเต็ปปิ้งมอเตอร์ส่วนใหญ่ใช้ 2 วิธี คือ วิธีแรก ใช้คอมพิวเตอร์หรือซิงเกิ้ลบอร์ดควบคุม ส่วนอีกแบบหนึ่ง ใช้วงจรดิจิตอลควบคุม ซึ่งวิธีแรกนั้น จะต้องทำชุดอินเตอร์เฟสระหว่างคอมพิวเตอร์กับสเต็ปปิ้งมอเตอร์ และจะต้องมีซอฟต์แวร์ เพื่อควบคุมสเต็ปปิ้งมอเตอร์ให้ทำงาน วิธีนี้ มีข้อเสียคือ ความยุ่งยากในการเขียนโปรแกรม ซึ่งต้องอาศัยความรู้และเทคนิคในการเขียนโปรแกรมอย่างมาก แต่ก็มีข้อดีคือ สามารถที่จะควบคุมตำแหน่งของสเต็ปปิ้งได้แน่นอนถูกต้องและมีความละเอียดดีมาก

ก่อนที่จะกล่าวถึงการทำงานของวงจรควบคุมสเต็ปปิ้งมอเตอร์ เราก็มาดูพื้นฐานหลักการทำงานของสเต็ปปิ้งมอเตอร์ และวิธีการควบคุมสเต็ปปิ้งมอเตอร์กันก่อน

โครงสร้างและการทำงานของสเต็ปปิ้งมอเตอร์

ภายในสเต็ปปิ้งมอเตอร์ประกอบด้วย สเตเตอร์, โรเตอร์ และขดลวดประกอบเข้าด้วยกันดังรูป 1 (สมมติเป็นมอเตอร์แบบ 3 เฟส)



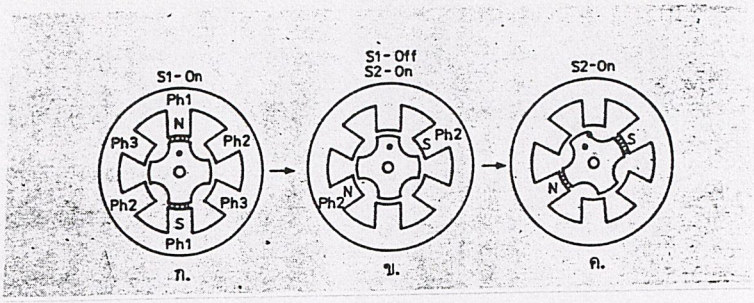
รูปที่ 1 ภาพหน้าตัดของสเตปปิ้งมอเตอร์แบบ 3 เฟส

เนื่องจากสเต็ปปิ้งมอเตอร์นี้ โรเตอร์เป็นเหล็กอ่อน ซึ่งมีคุณสมบัติพยายามปรับตัวเอง ให้อยู่ในแนวที่เส้นแรงแม่เหล็กผ่านมากที่สุด ดังในรูปที่ 2 เมื่อเกิดเส้นแรงแม่เหล็กขึ้นที่สเตเตอร์ตัดผ่านโรเตอร์ ตัวโรเตอร์ ก็จะพยายามปรับตัวเองให้เส้นแรงแม่เหล็กตัดผ่านตัวเองมากที่สุด โดยการหมุนตัวเอง ทำให้เกิดมุมของการหมุนขึ้น และมอเตอร์จะหยุดหมุนเมื่อเส้นแรงแม่เหล็กที่ตัดผ่านตัวมันถึงจุดที่มากที่สุด

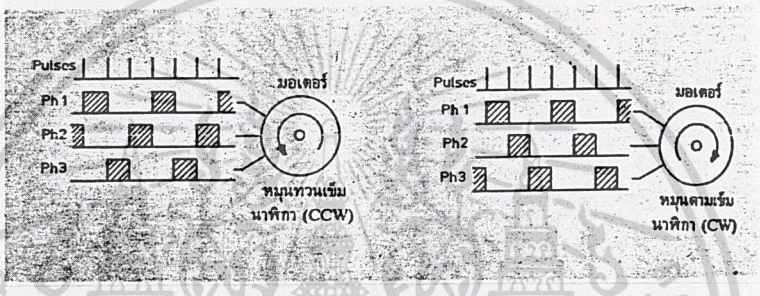
รูปที่ 2 เส้นแรงแม่เหล็กที่ทำให้เกิดแรงบิด

การทำให้สเต็ปปิ้งมอเตอร์หมุน ก็ทำได้โดยอาศัยหลักการนี้ แต่ต้องให้เส้นแรงแม่เหล็กเกิดขึ้น โดยรับช่วงต่อกันไปเรื่อยๆ ดังรูปที่ 3 ก., ข. และค. ซึ่งแสดงถึงการหมุนของมอเตอร์ โดยทิศทางขึ้นอยู่กับการขับกระแสเข้าขดลวด ว่าจะให้ไปทางไหน และเมื่อต้องการให้มอเตอร์หยุดก็หยุดการขับโรเตอร์ มอเตอร์ก็จะหยุด ณ ตำแหน่งสุดท้ายที่มีการขับที่สเตเตอร์ ดังนั้นเราจึงสามารถรู้ตำแหน่งของมอเตอร์ได้ โดยการนับจำนวนพัลส์ที่ป้อนให้มอเตอร์โดยใช้สูตร

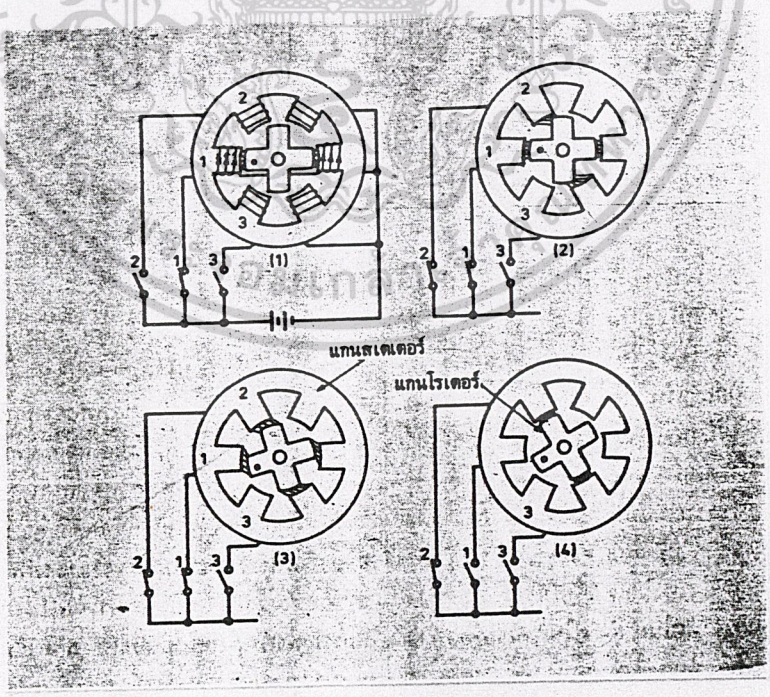
มุมที่เปลี่ยนไป = ค่ามุมต่อสเต็ป * จำนวนพัลส์ที่ป้อนให้



รูปที่ 3 แสดงการเคลื่อนที่ทีละสเตป เมื่อกระตุ้นเฟส 1 เฟส 2



รูปที่ 4 แสดงการกระตุ้น แบบเดินหน้าและถอยหลัง



รูปที่ 5 แสดงการทำงานของสเตปปิ้งมอเตอร์

การกระตุ้นเฟสของขดลวดสเตเตอร์

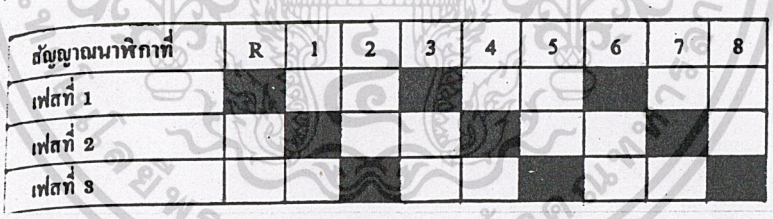
ดังที่รู้กันแล้วว่าในการทำให้สเต็ปปิ้งมอเตอร์หมุนได้นั้น จะต้องกระตุ้นเฟสของขดลวดสเตเตอร์ให้เรียงกันไปเรื่อยๆทางใดทางหนึ่ง ถ้าต้องการให้หมุนกลับ ก็กระตุ้นเฟสในทิศทางกลับกัน ซึ่งการกระตุ้นเฟสของสเตเตอร์มีอยู่ 3 แบบ คือ

1.การกระตุ้นเฟสเดียว เรียกว่า แบบ single phase excitation

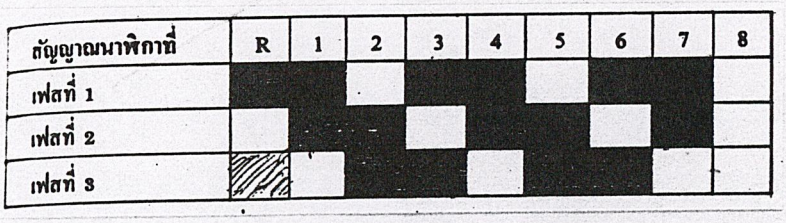
2.การกระตุ้นสองเฟสเรียกว่า แบบ two phase excitation

3.การกระตุ้นโดยใช้ แบบ 1 และ 2 สลับกันเรียกว่า แบบ one-two-phase excitation หรือแบบ half step operation

ในการขับแบบกระตุ้น 2 เฟส เส้นแรงแม่เหล็ก จะไม่ผ่านแกนเหล็กเป็นเส้นตรงเลยทีเดียวเหมือนแบบกระตุ้นเฟสเดียว แต่จะวกกลับมาเข้าสู่แกนทางด้านข้างๆ ดังรูปที่ 9 และเส้นแรงแม่เหล็กส่วนหนึ่งจะมาจากแกนตรงข้าม ดังรูปที่ 10



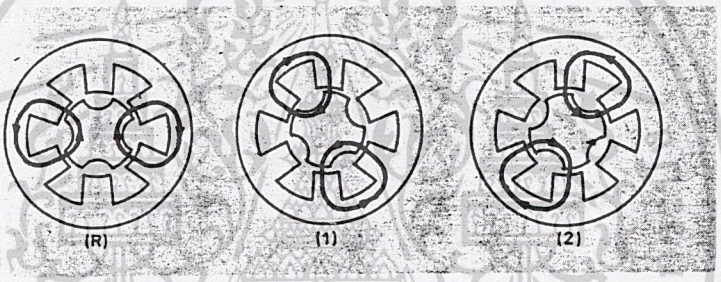
รูปที่ 6 แสดงการกระตุ้นแบบเฟสเดียว



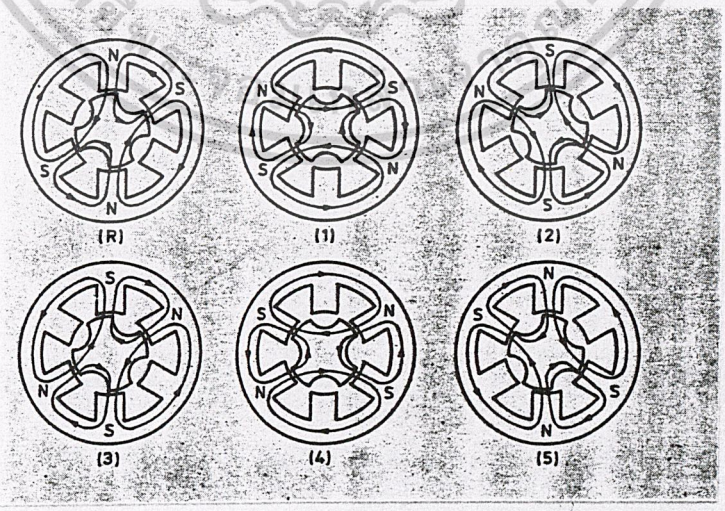
รูปที่ 7 แสดงการกระตุ้นแบบ 2 เฟส

สัญญาณนาฬิกาที่	R	1	2	3	4	5	6	7	8
เฟสที่ 1	■	■					■	■	■
เฟสที่ 2		■	■	■					■
เฟสที่ 3				■	■	■			

รูปที่ 8 แสดงการกระตุ้นแบบครึ่งสเตป



รูปที่ 9 เส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส



รูปที่ 10 ลักษณะเส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส

การขับสเต็ปปิ้งมอเตอร์ แบบกระตุ้น 2 เฟสนี้ จะมีลักษณะเดียวกันกับการขับแบบกระตุ้นเฟสเดียว แต่ในการกระตุ้นแต่ละครั้งนั้นจะกระตุ้นทีเดียวพร้อมกันทั้ง 2 เฟส จะต่างกันก็ตรงที่การขับแบบ 2 เฟสนั้น เข้าตำแหน่งแต่ละสเตปได้เร็วกว่าแบบเฟสเดียวและแรงบิดมากกว่าการขับแบบเฟสเดียวด้วย ดังนั้นในโครงงานนี้จึงใช้การขับแบบกระตุ้นสองเฟส

การเกิดออสซิลเลตของสเต็ปปิ้งมอเตอร์

ความเร็วของการหมุนของสเต็ปปิ้งมอเตอร์เป็นลิเนียร์กับความถี่ ที่ป้อนให้สเต็ปปิ้งมอเตอร์ แต่เมื่อเราป้อนความถี่เพิ่มขึ้นเรื่อยๆ จนถึงความถี่ค่าหนึ่ง สเต็ปปิ้งมอเตอร์ก็จะหยุดหมุน เนื่องจากการที่โรเตอร์หมุนตามพลังซ์แม่เหล็กไม่ทัน เราเรียกปรากฏการณ์นี้ว่า มอเตอร์เกิดการออสซิลเลต ซึ่งมอเตอร์แต่ละตัวจะออสซิลเลตที่ความถี่แตกต่างกันไป โดยทั่วไปจะออสซิลเลตที่ความถี่ประมาณ 500 Hz ในการขับแบบกระตุ้น 1 หรือ 2 เฟส และจะออสซิลเลตที่ความถี่ประมาณ 1 kHz เมื่อขับแบบครึ่งสเต็ป

แต่เมื่อลดความถี่ให้ต่ำลงมอเตอร์จะไม่หมุนทันที และเมื่อความถี่ลดลงจนถึงความถี่หนึ่ง มอเตอร์จึงจะเริ่มหมุนอีกครั้ง นั่นก็คือ มอเตอร์มีฮิสเทอ-รีซิส ซึ่งมอเตอร์จะเริ่มหมุน ที่ความถี่ประมาณ 200 Hz ในการขับแบบ 1 หรือ 2 เฟส และที่ความถี่ประมาณ 150 Hz เมื่อขับแบบครึ่งสเต็ป

เกี่ยวกับ 8052 เบสิก ไมโครคอนโทรลเลอร์

8052เอเอช เบสิก (8052AH BASIC) เป็นไมโครคอนโทรล-เลอร์แบบชิพเดี่ยว อยู่ในตระกูล เอ็มซีเอส-51 (MCS-51) ของอินเทล โดยมีตัวแปลภาษาเบสิก (BASIC interpreter) ขนาด 8 กิโลไบต์ อยู่ในหน่วยความจำรอมภายใน ทำให้สามารถโปรแกรมการทำงานด้วยภาษาเบสิกได้ โดยใช้เครื่องคอมพิวเตอร์ใดๆ ที่มีพอร์ตสื่อสารอนุกรม เป็นเทอมินอล และเมื่อพัฒนาโปรแกรมเรียบร้อยแล้ว ขณะใช้ในงานควบคุมระบบจริงชุดฮาร์ดแวร์ 8052เอเอช สามารถที่จะต่อ หรือไม่ต่อกับเทอมินอลก็ได้ ขึ้นอยู่กับความต้องการของผู้ใช้

รู้จักกับ เอ็มซีเอส เบสิก-52 (MCS BASIC-52)

เอ็มซีเอส เบสิก-52 คือตัวแปลภาษาเบสิกใน 8052เอเอช เบสิก มีระบบการจัดการติดต่อกับเทอร์มินอลในตัว เพื่อสะดวกในการเขียน แก้ไข และทดสอบโปรแกรมภาษาเบสิก โปรแกรมที่เขียนสามารถเก็บลงใน อีพรอม (EPROM)ที่ต่อภายนอก และนำโปรแกรมที่เก็บไว้ใน อีพรอม ภายนอก มาทำงานได้ ระบบรีลไทม์คล็อก (real time clock) และการอินเตอร์รัพต์ สามารถรับรู้และควบคุมด้วยโปรแกรมเบสิก มีคำสั่งสำหรับการติดต่อกับรีจิส-เตอร์, หน่วยความจำภายใน, ภายนอกและพอร์ต

เอ็มซีเอส เบสิก-52 สร้างขึ้นมาสำหรับ การเขียนโปรแกรมงานควบคุมระบบอัตโนมัติที่ใช้ไมโครคอนโทรลเลอร์ สามารถพัฒนาโปรแกรมได้ง่ายขึ้นและใช้เวลาน้อยลง เพราะการพัฒนาโปรแกรมด้วยภาษาแอสเซมบลี้บางครั้งยาก และน่าเบื่อ เช่น โปรกรมที่ต้องมีการคำนวณทางคณิตศาสตร์ การสร้างฐานข้อมูลของเครื่องบันทึกข้อมูล การเขียนโปรแกรมเกี่ยวกับการติดต่อกับจอภาพ และคีย์บอร์ดของเครื่องคอมพิวเตอร์ เพื่อใช้ในการแสดงผลและรับคำสั่งให้เข้ากับชุดฮาร์ดแวร์ของระบบไมโครคอนโทรลเลอร์ ซึ่งโปรแกรมเหล่านี้จะง่ายขึ้นทันที ถ้าเขียนด้วยภาษาเบสิก

เป็นที่แน่นอนว่า โปรแกรมภาษาแอสเซมบลี้ย่อมทำงานได้เร็วกว่าโปรแกรมภาษาเบสิก ดังนั้น เอ็มซีเอส เบสิก-52 จึงอนุญาตให้ผู้ใช้ ติดต่อกับภาษาแอสเซมบลี้ได้ ทำให้งานบางอย่างที่โปรแกรมเบสิกทำไม่ทัน เช่นการสแกนภาคแสดงผล 7 เซกเมนต์แบบมัลติเพล็กซ์ ซึ่งต้องใช้ความเร็วในการสแกนสูง ก็สามารถเขียนเป็นโปรแกรมย่อยภาษาแอสเซมบลี้ และเรียกใช้จากโปรแกรมเบสิค ด้วยคำสั่ง CALL

คุณสมบัติพิเศษของ เอ็มซีเอส เบสิก-52

เอ็มซีเอส เบสิก-52 เป็นตัวแปลภาษาเบสิก ที่มีคำสั่งและฟังก์ชันพื้นฐานคล้ายภาษาเบสิกทั่วไป แต่เอ็มซีเอส เบสิก-52 ได้เพิ่มคำสั่งเพื่อความสะดวกในการพัฒนาโปรแกรมควบคุมระบบ เช่น

-สามารถโปรแกรม อีพรอม ได้ โดยคำสั่ง "PROG" จะนำโปรแกรมเบสิกที่ใช้อยู่ขณะนั้น บันทึกลงใน อีพรอม ในลักษณะของไฟล์ (FILE)

-สามารถเรียกโปรแกรมเบสิก ที่เก็บไว้ใน อีพรอม ออกมาทำงานได้ โดยคำสั่ง "ROM"

-สามารถย้ายโปรแกรมเบสิกที่อยู่ใน อีพรอม มาไว้ใน RAM ด้วยคำสั่ง "XFER"

-สามารถเรียกโปรแกรมย่อย ภาษาแอสเซมบลี้ โดยใช้คำสั่ง "CALL"

-โปรแกรมย่อยภาษาแอสเซมบลี้สามารถเรียกใช้ฟังก์ชันภาษาเบสิกได้

-มีคำสั่งควบคุม ไทม์เมอร์/เคาน์เตอร์ ที่อยู่ในตัว ด้วยคำสั่ง "TIMER0, TIMER1, TIMER2, TCON, TMOD" เป็นต้น

-มีระบบรีลไทม์คล้อก ที่เที่ยงตรง ควบคุมด้วยคำสั่ง "CLOCK0, CLOCK1, TIMER"

-สามารถรับรู้และควบคุมการอินเตอร์รัพต์ ด้วยคำสั่ง "ONEX1, RETI, CLEARI, IDLE" เป็นต้น

-สามารถอ่านและเขียนค่าพอร์ตในตัวได้ ด้วยคำสั่ง "PORT1"

-มีคำสั่งควบคุมพอร์ตอนุกรมในตัว เช่น คำสั่ง "BAUD,RCAP2"

-สามารถติดต่อกับหน่วยความจำภายในและภายนอก ด้วยคำสั่ง "CBY(), DBY(), XBY()"

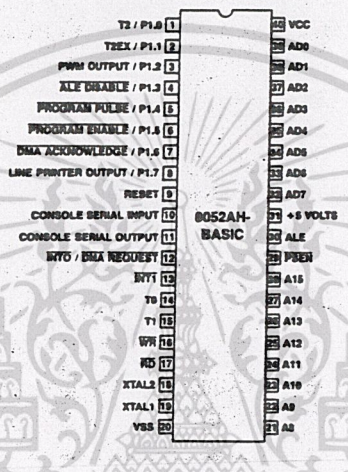
-มีคำสั่งควบคุมการลูป "DO WHILE, DO UNTIL"

-มีฟังก์ชันทางคณิตศาสตร์สมบูรณ์

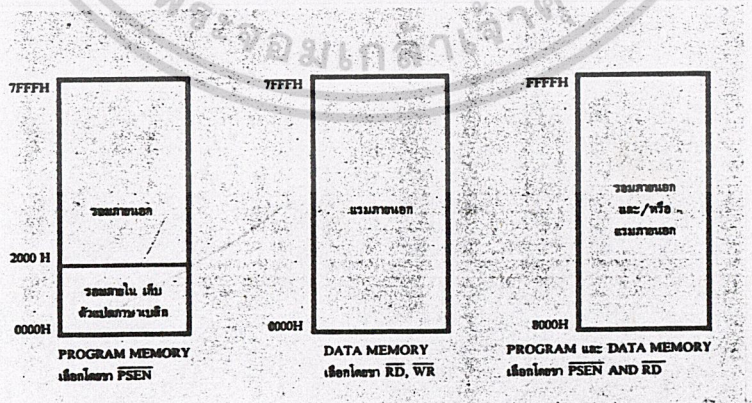
การจัดการหน่วยความจำ

เอ็มซีเอส เบสิก-52 เป็นตัวแปลภาษาเบสิก บรรจุอยู่ใน ROM ภายในชิพ 8052เอเอช เบสิก มีขนาด 8 กิโลไบต์ อยู่ระหว่างแอดเดรส 0000h-1FFFh (program memory) ซึ่งในการทำงาน ต้องการหน่วยความจำเป็นแรม (RAM)ภายนอกอย่างน้อย 1 กิโลไบต์ หลังจากการรีเซ็ต เอ็มซีเอส เบสิก-52 จะหาขนาดของแรมภายนอก (EXTERNAL RAM) โดยเริ่มตั้งแต่แอดเดรส 0000h-0DFFh (data memory) ข้อมูลในแรมที่ผ่านการทด

สอบเพื่อหาขนาด จะมีค่าเป็น 00h ถ้าขนาดของแรมภายนอก ไม่ถึง 1 กิ-ไลไบต์ เอ็มซีเอส เบสิก-52 จะไม่สามารถทำงานได้ ส่วนแอดเดรส 0E00h 0FFFFh (data memory) สงวนไว้ให้ผู้ใช้ สำหรับ อินพุต เอาต์พุต และโปรแกรมภาษาแอสเซมบลี้ ข้อมูลในแอดเดรสระหว่างนี้จะไม่ถูกเคลียร์เป็น 00h



รูปที่ 1 แสดงการจัดวางขาต่างๆของ 8052เอเอช เบสิก



รูปที่ 2 โครงสร้างการจัดหน่วยความจำของ 8052เอเอช เบสิก ใน แรม/อีพรอม โหมด

8052เอเอช เบสิก แบ่งการจัดการหน่วยความจำ ตามโครงสร้างทางฮาร์ดแวร์ของระบบที่ผู้ใช้สร้างขึ้นเป็น 2 โหมด

1.แรมออนลี่โหมด (RAM ONLY MODE) การทำงานในโหมดนี้สามารถต่อแรมภายนอก (external RAM) ได้ถึง 64 กิโลไบต์ เริ่มต้นที่แอดเดรส 0000h ถึง 0FFFFh ในโหมดนี้จะใช้สัญญาณติดต่อกับแรมภายนอกดังต่อไปนี้

-ดีโค้ดแอดเดรส(decode address) จาก 8052เอเอช ใช้เป็นสัญญาณ CS สำหรับแรมภายนอก

-ขา $\overline{RD}$ ของ 8052เอเอช ใช้เป็นสัญญาณ $\overline{OE}$ สำหรับแรมภายนอก

-ขา $\overline{WR}$ ของ 8052เอเอช ใช้เป็นสัญญาณ $\overline{WE}$ หรือ $\overline{WR}$ สำหรับแรมภายนอก

-ขา $\overline{PSEN}$ ของ 8052เอเอช ไม่ถูกใช้ในโหมดนี้ การทำงานในโหมดนี้ จึงไม่สามารถโปรแกรม อีพรอม ได้

โดยทั่วไปแล้ว โหมดนี้ใช้สำหรับ ทดสอบความถูกต้องของฮาร์ดแวร์ และทำความเข้าใจกับระบบการทำงาน ที่ง่ายที่สุดของ 8052เอเอช เพราะใช้อุปกรณ์ประกอบเป็นระบบเพียงไม่กี่ตัว ระบบฮาร์ดแวร์ในโหมดนี้ แสดงในรูปที่ 3

2.แรม/อีพรอม โหมด (RAM/EPROM MODE) การทำงานในโหมดนี้เป็นการทำงานที่สมบูรณ์ของระบบ 8052เอเอช เบสิก โหมดนี้ต้องการ การจัดหน่วยความจำภายนอกให้อยู่ในแบบแผนที่แน่นอน ในโหมดนี้จะใช้สัญญาณติดต่อกับหน่วยความจำภายนอกทั้ง แรม และ อีพรอม ดังต่อไปนี้

-ขา $\overline{RD}$ และ $\overline{WR}$ ของ 8052เอเอช ใช้เป็นสัญญาณ $\overline{OE}$ และ $\overline{WR}$ สำหรับแรมภายนอกตามลำดับ โดยมีแอดเดรสตั้งแต่ 0000h-7FFFh และมีสถานะเป็น ดาต้าเมมโมรี่ (data memory) ส่วนดีโค้ดแอดเดรสจาก 8052เอเอช ใช้เป็นสัญญาณ $\overline{CS}$ สำหรับแรมภายนอก

-ขา $\overline{PSEN}$ ของ 8052เอเอช ใช้เป็นสัญญาณ $\overline{OE}$ สำหรับ อีพรอม โดยมีแอดเดรสตั้งแต่ 2000h-7FFFh และมีสถานะเป็น โปรแกรมเมมโมรี่ (program memory) ดีโค้ดแอดเดรสจาก 8052เอเอช ใช้เป็นสัญญาณ $\overline{CS}$ สำหรับ อีพรอม ภายนอก

-ขา $\overline{RD}$ และ $\overline{PSEN}$ ของ 8052เอเอช นำมา AND กัน ใช้เป็นสัญญาณ $\overline{OE}$ สำหรับ แรม และ/หรือ อีพรอม โดยมีแอดเดรสตั้งแต่ 8000h-0FFFFh โดยมีสถานะเป็นได้ทั้ง ดาต้า และ โปรแกรมเมมโมรี่ ดีโค้ดแอด-

เดรสจาก 8053เอเอช ใช้เป็นสัญญาณ $\overline{CS}$ สำหรับ แรม และ/หรือ อีพรอม ภายนอก และขา $\overline{WR}$ ของ 8052เอเอช ใช้เป็นสัญญาณ $\overline{WR}$ หรือ $\overline{WE}$ สำหรับแรมภายนอก

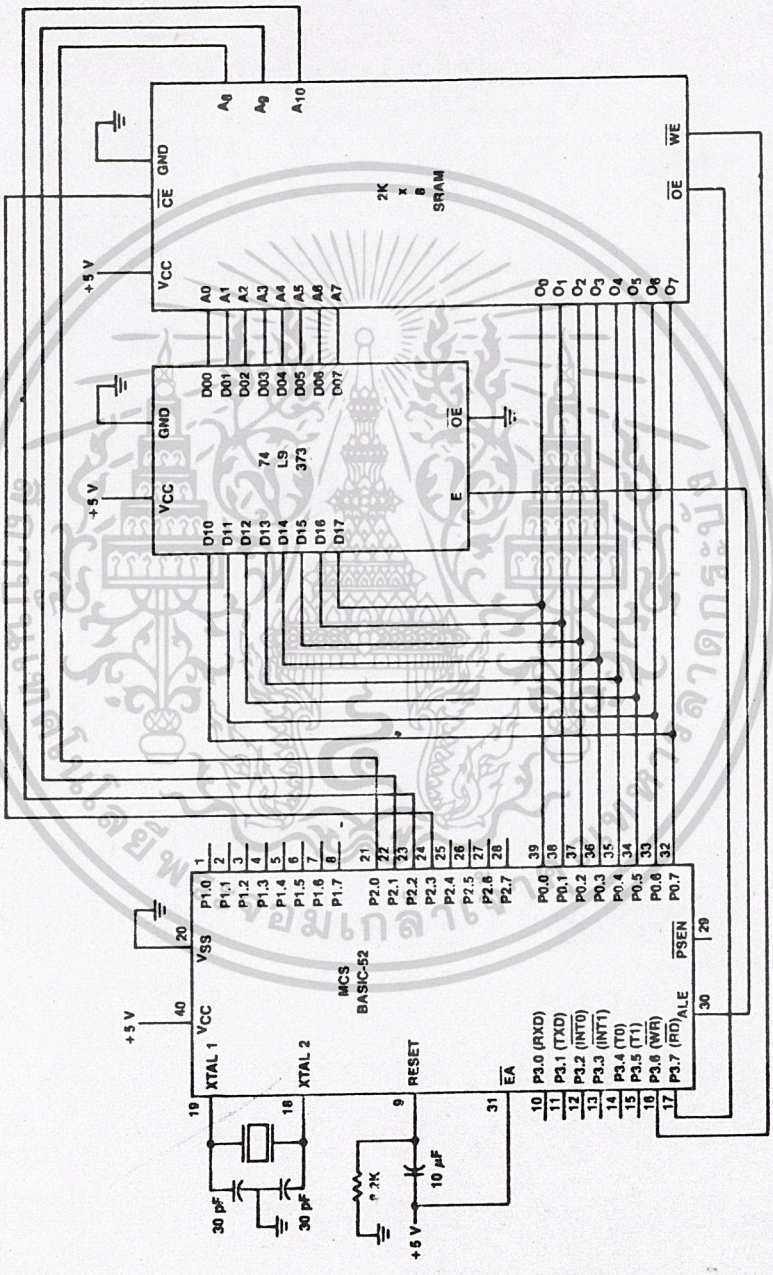
จากแบบแผนการจัดการหน่วยความจำในโหมดนี้ 8052เอเอช เบสิก จะสามารถอ้างหน่วยความจำได้ถึง 96 กิโลไบต์ ดังต่อไปนี้

32 กิโลไบต์ สำหรับ แรมภายนอก โดยมีแอดเดรสตั้งแต่ 0000h-7FFFh

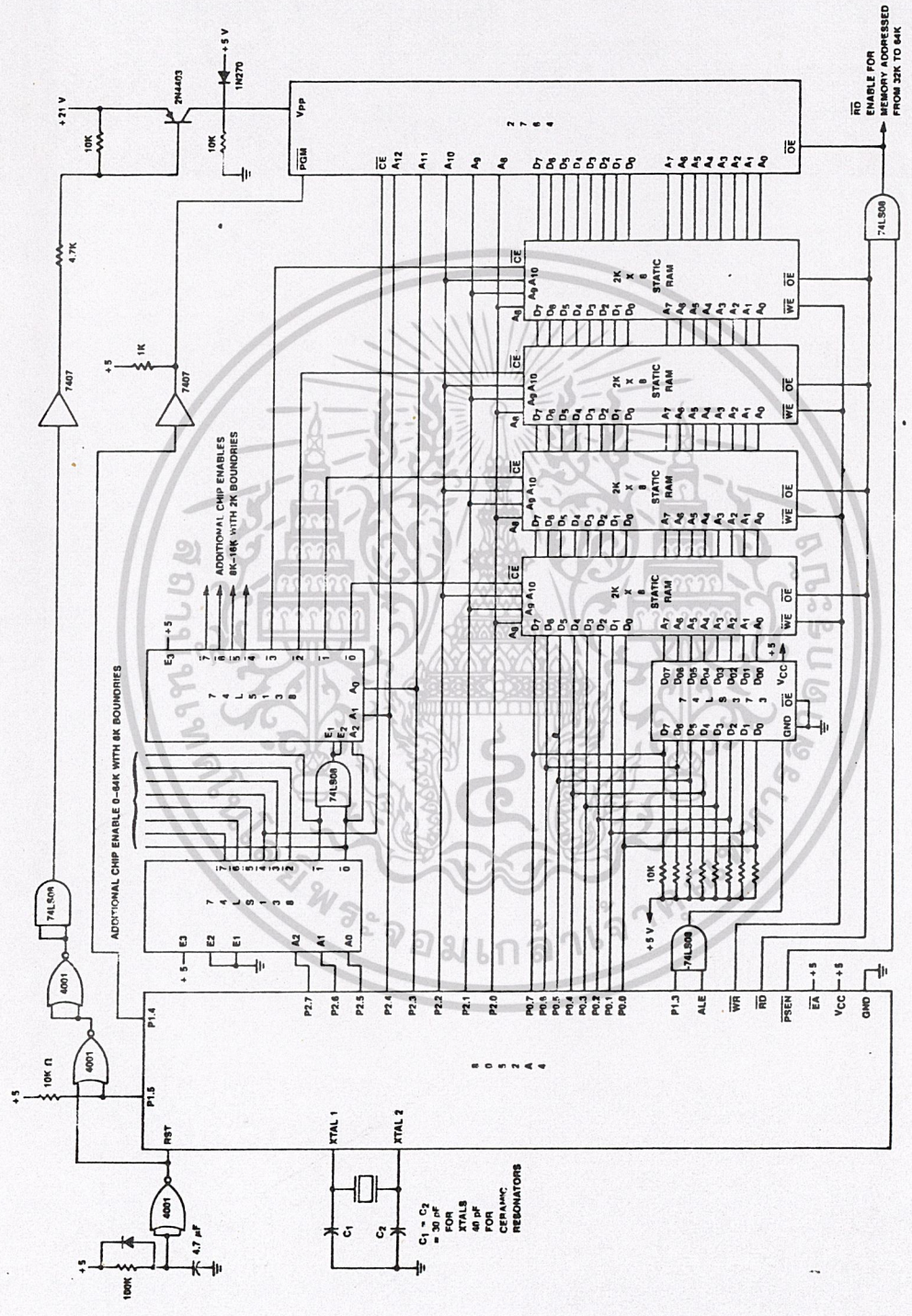
32 กิโลไบต์ สำหรับ รอมภายในและภายนอก มีแอดเดรสตั้งแต่ 0000h-7FFFh โดยรอมภายในอยู่ระหว่างแอดเดรส 0000h-1FFFh เป็นที่เก็บตัวแปลภาษาเบสิก นอกนั้นเป็นแอดเดรสของรอมภายนอก

32 กิโลไบต์สำหรับ แรม และ/หรือ อีพรอม ภายนอก มีแอดเดรสตั้งแต่ 8000h-0FFFFh

ในโหมดนี้สามารถโปรแกรมอีพรอมได้ อีพรอม ที่จะถูกโปรแกรมต้องอยู่ระหว่างแอดเดรส 8000h-0FFFFh โดยมีแอดเดรสเริ่มต้นที่ 8000h เมื่อคำสั่ง"PROG" ถูกใช้ สำหรับ อีพรอม ที่ว่าง (ข้อมูลเป็น 0FFh ทั้งหมด) โปรแกรมภาษาเบสิกจะถูกอัดลง โดยมีแอดเดรสเริ่มต้นที่ 8010h ส่วนแอดเดรส 8000h-800Fh ถูกสงวนไว้สำหรับเก็บค่าบ็อดเรต (baud rate) และค่าสำหรับเซ็ตอัพ (setup)ระบบตามที่ผู้ใช้ต้องการ ระบบฮาร์ดแวร์ที่ทำงานได้ในโหมดนี้แสดงในรูปที่ 4



รูปที่ 3 แสดงการจัดฮาร์ดแวร์ใน แรม ออนลี่ โหมด



รูปที่ 4 แสดงการจัดฮาร์ดแวร์ใน แรม/อีพรอม โหมด

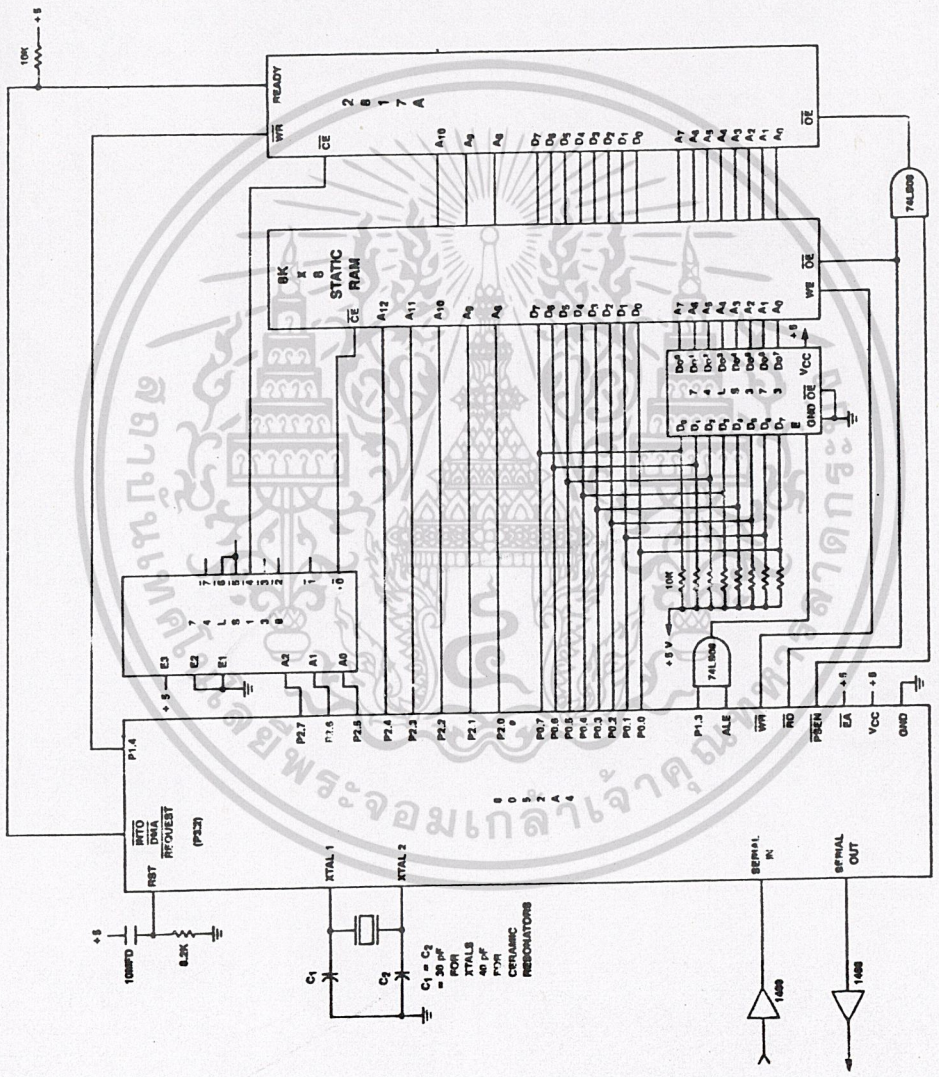


Figure 2B. Programming 2817A's with Version 1.1 of MCS BASIC-52

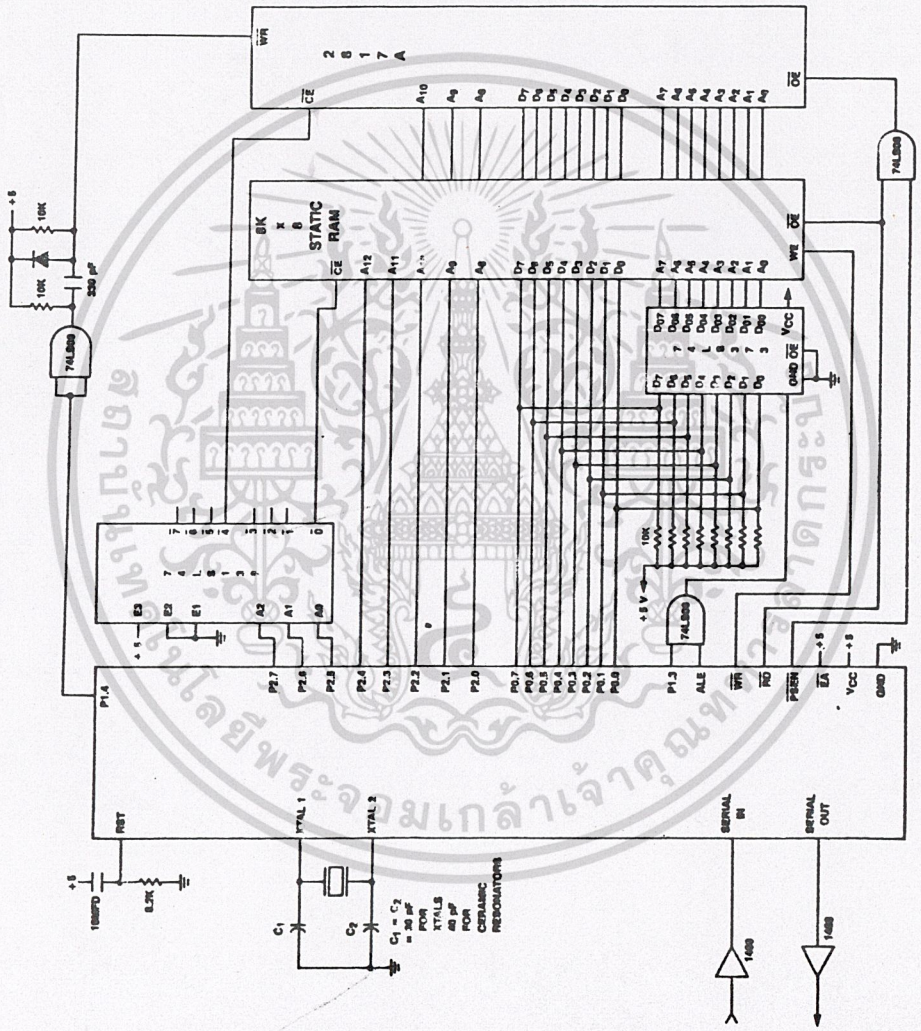


Figure 2C. Programming 2817A's with Version 1.0 of MCS BASIC-52

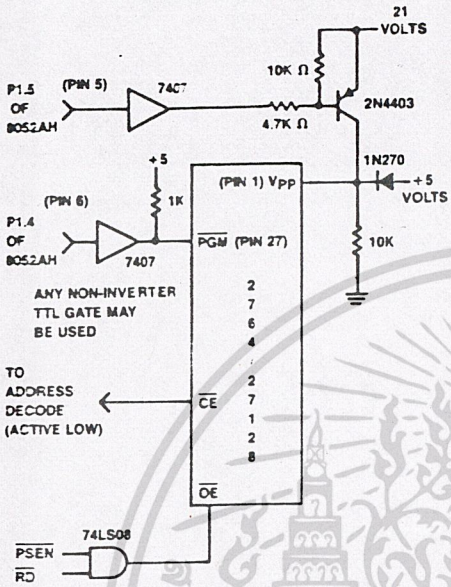


Figure 4A. Programming 2764's 27128's

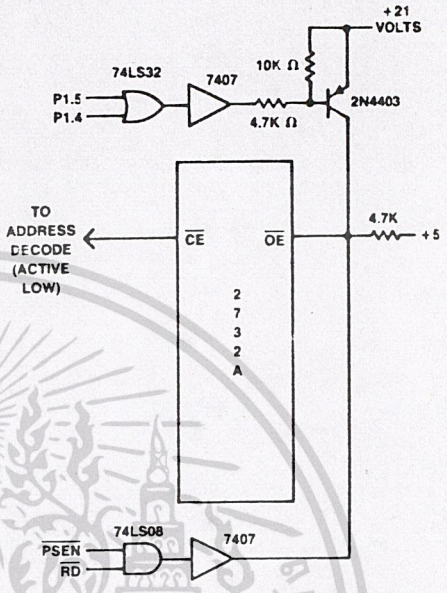


Figure 4B. Programming 2732A's

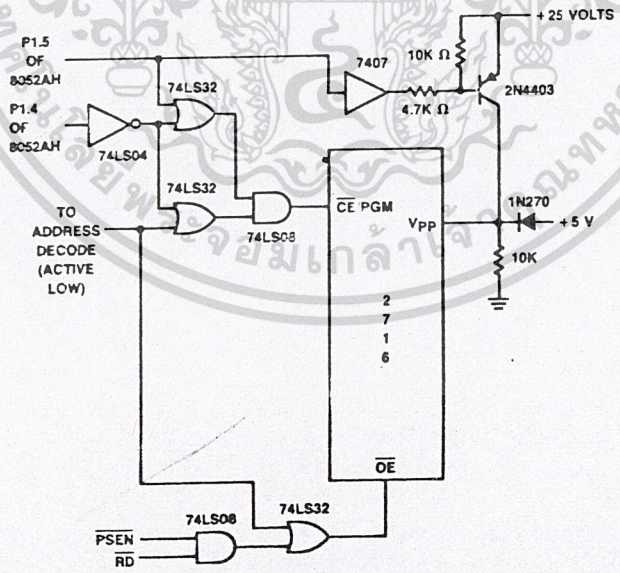


Figure 4C. Programming 2716's

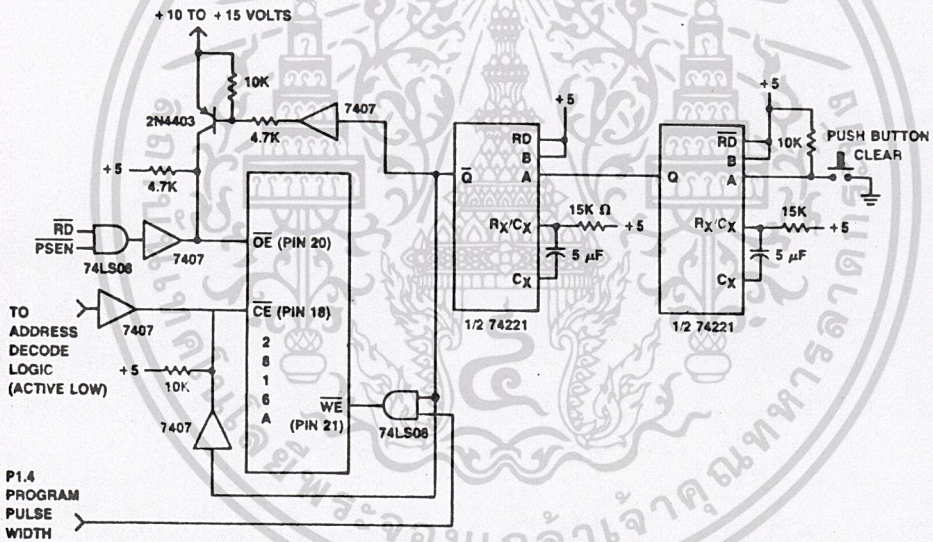


Figure 5. 2816A Circuit with Push Button Erase.

(Basic-52 should be "Idle" in the command mode when the Erase Button is pushed.)

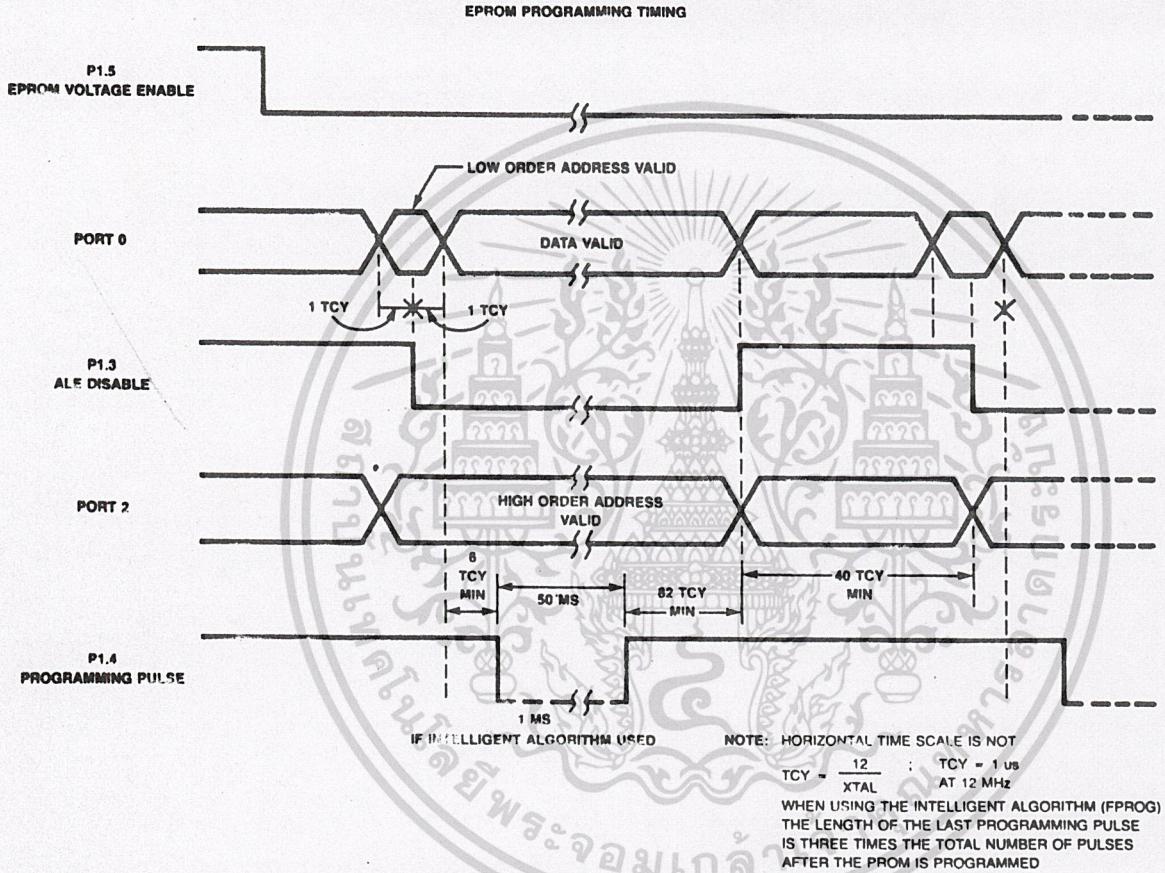
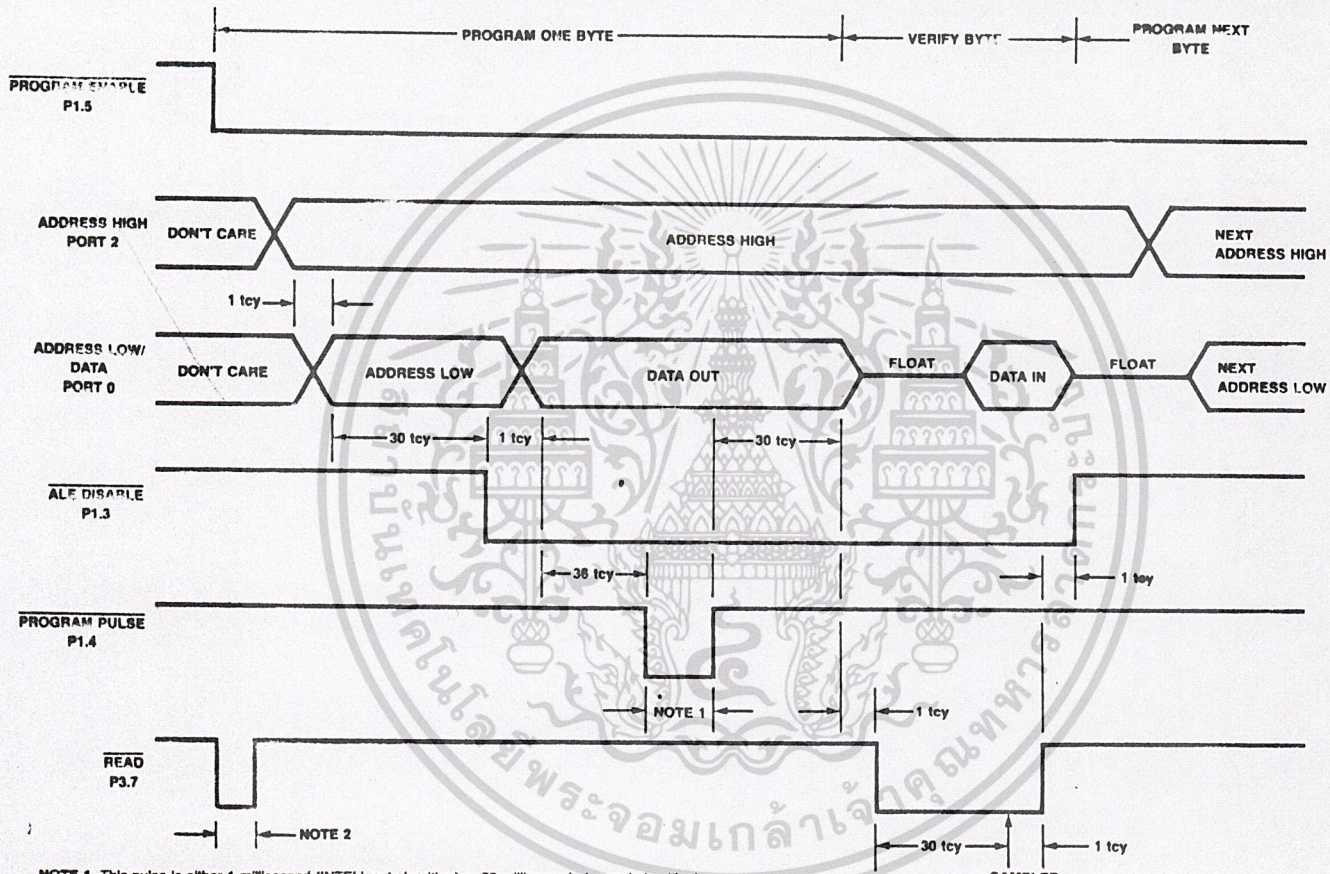


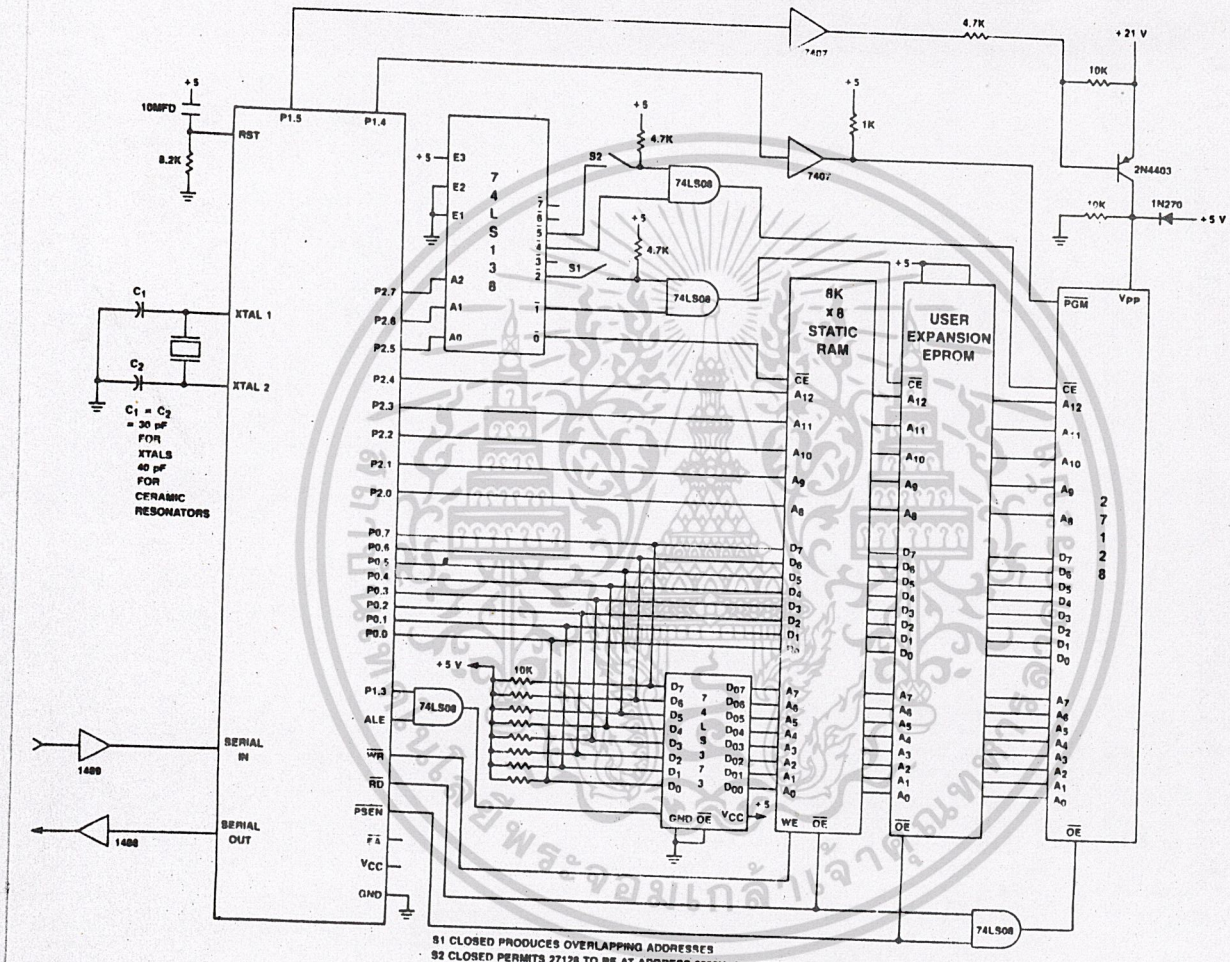
Figure 3A. EPROM Programming Timing Version 1.0



NOTE 1. This pulse is either 1 millisecond (INTELligent algorithm) or 50 milliseconds (normal algorithm).

NOTE 2. When PROG command is executed, P1.5 goes low, and then the EPROM is read to see where to place the BASIC program.

Figure 3B. EPROM programming timing for Version 1.1



Overlapping user EPROM address space

รูปแบบการรับส่งข้อมูล

รูปแบบของข้อมูลที่ เอ็มซีเอส เบสิก-52 ใช้ในการติดต่อรับส่งข้อมูลกับเทอมินอล อยู่ในโหมด 8-บิต ยูอ๊าท (8-BIT UART) ซึ่งมี 8 ดาต้าบิต และ 1 สต็อบบิต ไม่มีพาริตี้บิต

พอร์ตอนุกรมของ 8052เอเอช เบสิก เป็นสัญญาณแบบ ทีทีแอล (TTL) จึงไม่สามารถ ต่อเข้าโดยตรงกับ พอร์ตอนุกรมมาตรฐาน RS232C ของเทอมินอล จึงต้องมีวงจรอินเตอร์เฟส (INTERFACE) เพื่อแปลงสัญญาณทีทีแอล เป็นสัญญาณ อีไอเอ (EIA) ตามมาตรฐาน อาเอส232ซี (RS232C) ดังแสดงในรูปที่ 5

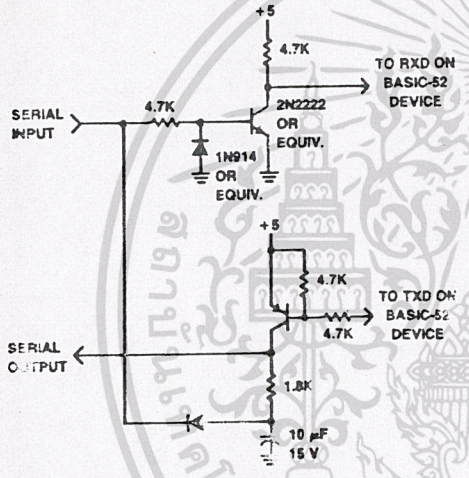


Figure 6A.
TWO TRANSISTORS TO IMPLEMENT RS-232 THE "NEGATIVE" SUPPLY FOR THE SERIAL OUTPUT LINE IS TAKEN FROM THE SERIAL INPUT LINE. NO ±12 VOLT SUPPLY IS REQUIRED.

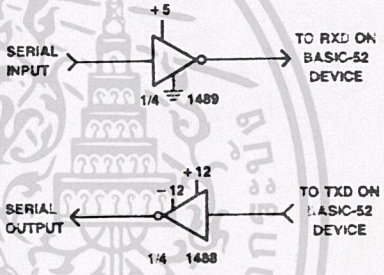


Figure 6B.
USING THE STANDARD 1489 AND 1488 LINE RECEIVERS AND DRIVERS, ±12 VOLTS IS NEEDED WITH THIS IMPLEMENTATION.

รูปที่ 5 แสดงวงจรอินเตอร์เฟสสำหรับแปลงสัญญาณ ทีทีแอล เป็น อีไอเอ และอีไอเอ เป็น ทีทีแอล

อะไรเกิดขึ้นหลังการรีเซ็ต

เมื่อทำการรีเซ็ตระบบ 8052เอเอช เบสิก จะเกิดผลดังต่อไปนี้

1. เคลียร์หน่วยความจำแรมภายใน
2. ให้ค่าต่างๆ แก่รีจิสเตอร์ และพอยต์เตอร์
3. ทดสอบ เคลียร์ และหาขนาดของแรมภายนอก

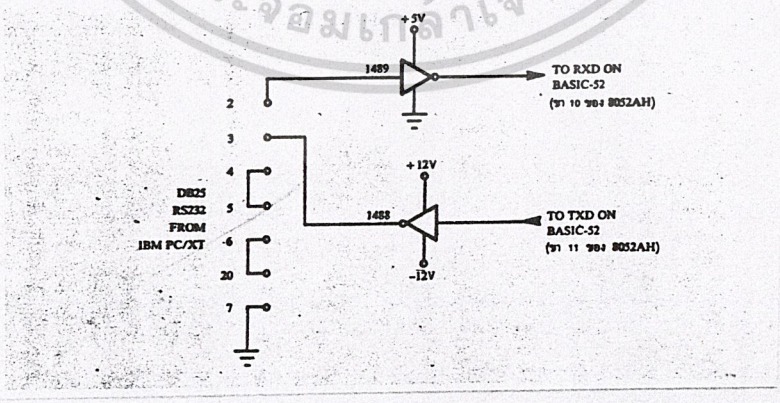
หลังจากนั้นจะเข้าสู่การทำงานของโปรแกรม ออโต้ บ็อด (AUTO BAUD) ซึ่งจะหาบ็อดเรตของข้อมูลที่ส่งมาจากเทอร์มินอลโดยอัตโนมัติ เมื่อหา

ได้แล้วจะใส่ค่าควบคุมบ็อดเรตให้แก่ RCAP2 ซึ่งเป็นรีจิสเตอร์ควบคุมบ็อดเรต ของ 8052เอเอช เบสิก แล้วจึงส่งสัญญาณแสดงความเรียบร้อย (SIGN ON) ไปที่จอภาพของเทอร์มินอล เป็นข้อความดังนี้

```
*MCS-51(tm)BASIC V1.1*
READY
```

การเชื่อมต่อกับเทอร์มินอล

การเขียนโปรแกรมภาษาเบสิกจำเป็นต้องมีจอภาพ และ คีย์บอร์ด เพื่อใช้ในการเขียน แก้ไข และทดสอบโปรแกรม ในการทดลองใช้งานระบบ 8052เอเอช เบสิก โดยใช้เครื่องคอมพิวเตอร์ ไอบีเอ็ม พีซี/เอ็กซที (IBM PC/XT) เป็นเทอร์มินอลให้กับระบบ การสื่อสารข้อมูลโดยผ่านพอร์ตสื่อสารอนุกรม อาเอส232ซี ซึ่งอยู่ในการ์ดมัลติไอโอ (MULTI I/O) ติดตั้งอยู่บนสล็อตของเครื่อง ไอบีเอ็ม พีซี/เอ็กซที และใช้โปแกรม ครอสทอล์ค (CROSS TALK) เป็นโปรแกรมควบคุมการสื่อสารข้อมูล โดยสามารถนำข้อมูลที่ถูกส่งมาจากชุดฮาร์ดแวร์ระบบ 8052เอเอช มาแสดงที่จอภาพของเครื่อง ไอบีเอ็ม พีซี/เอ็กซที และสามารถส่งข้อมูล จากคีย์บอร์ดของเครื่อง ไอบีเอ็ม พีซี/เอ็กซที ให้กับชุดฮาร์ดแวร์ของระบบ 8052เอเอช เบสิก โดยข้อมูลที่รับและส่งนั้นอยู่ในรูปของรหัสแอสกี้ (ASCII) การเชื่อมต่อระหว่างพอร์ต อาเอส232ซี ของเครื่อง ไอบีเอ็ม พีซี/เอ็กซที เข้ากับชุดฮาร์ดแวร์ ของระบบ 8052เอเอช เบสิก แสดงในรูปที่ 6



รูปที่ 6 แสดงการเชื่อมต่อระหว่างพอร์ต อาเอส232ซี กับ 8052เอเอช เบสิก

เมื่อเชื่อมต่อกันเรียบร้อยดีแล้วนี้ก็เปิดเครื่อง ไอบีเอ็ม พีซี/เอ็กซที แล้วเริ่มใช้โปรแกรม ครอสทอล์ค โดยคีย์ตามคำสั่งนี้

```
A>XTALK                <ENTER>
```

เมื่อเข้าสู่โปรแกรมครอสทอล์ค แล้วให้เซ็ตอัพโปรแกรมครอสทอล์ค ทางการรับส่งข้อมูล โดยมีอัตราการส่งข้อมูล (baud rate) = 9600 บิตต่อนาที, 8 ดาต้าบิต, 1 สตาร์ทบิต, 1 สต็อปบิต, ไม่มีพาริตี้บิต และในโหมดฟูลดูเพลกซ์ (full duplex)

หลังจากนั้นเครื่อง ไอบีเอ็ม พีซี/เอ็กซที ก็พร้อมแล้วสำหรับทำหน้าที่เป็นเทอร์มินอล ให้กับชุดฮาร์ดแวร์ของระบบ 8052เอเอช เบสิก

เมื่อจ่ายไฟเข้าชุดฮาร์ดแวร์ 8052เอเอช (หรือทำการรีเซ็ต) แล้วกดคีย์ สเปซบาร์ (spacebar) บนคีย์บอร์ดของเครื่อง ไอบีเอ็ม พีซี/เอ็กซที ข้อความต่อไปนี้จะปรากฏขึ้นที่มุมซ้ายบนสุดของจอภาพทันที

```
*MCS-51(tm)BASIC V1.1*
READY
```

จากนั้นทดลองเขียนโปรแกรมสั้นๆ เพื่อทดสอบการทำงานของชุดฮาร์ดแวร์ 8052เอเอช

```
>10 FOR I=1 TO 5
>20 PRINT I
>30 NEXT I
```

ระหว่างการเขียนโปรแกรม ถ้าเขียนผิด จะต้องเขียนบรรทัดนั้นใหม่ทั้งบรรทัด เขียนเสร็จแล้วทดลองใช้คำสั่ง "LIST"

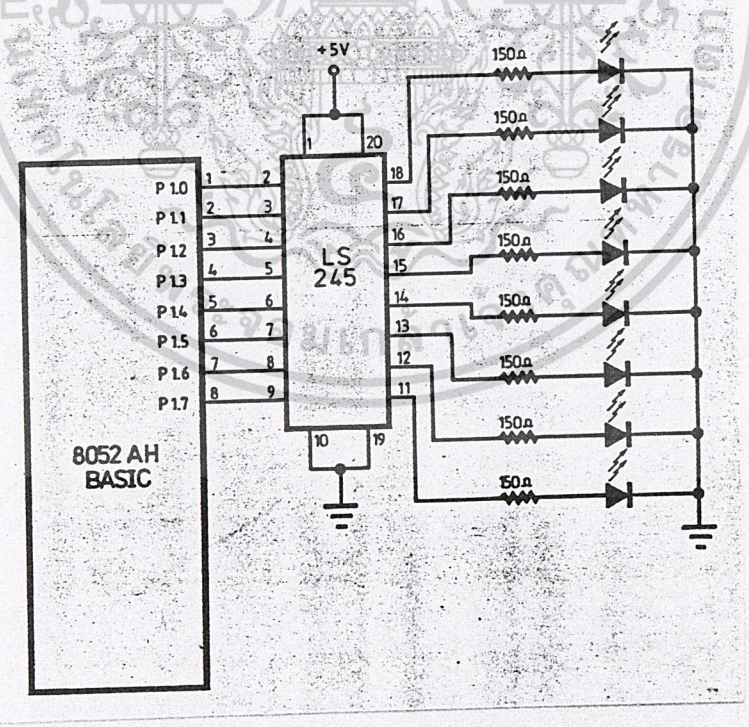
```
>LIST                  <ENTER>
>10 FOR I=1 TO 5
>20 PRINT I
>30 NEXT I
READY
```

```
>
```

ถ้าเรียบร้อยดีทดลองให้โปรแกรมทำงานด้วยคำสั่ง "RUN"

```
>RUN
 1
 2
 3
 4
 5
READY
>
```

ถ้าผลออกมาเป็นดังที่เห็นแสดงว่าการเชื่อมต่อระหว่างชุดฮาร์ดแวร์ 8052เอเอช เบสิก กับ ไอบีเอ็ม พีซี/เอ็กซ์ที เรียบร้อยดี และ เอ็มซีเอส เบสิก-52 พร้อมแล้วที่จะรับใช้



รูปที่ 7 แสดงการต่อภาคขับหลอด แอลอีดี ให้กับพอร์ต 1 ของ 8052เอเอช

การทดลองใช้งานพอร์ต

เอ็มซีเอส เบสิก-52 มีคำสั่ง "PORT1" ไว้สำหรับให้ผู้ใช้อ่านและเขียนค่าในพอร์ต 1 ซึ่งเป็นพอร์ตแบบขนาน 8 บิต ในตัว 8052เอเอช ถ้าอยากทดลองเกี่ยวกับโปรแกรมใช้งานพอร์ต ให้ต่อฮาร์ดแวร์เพิ่มเติมตามรูปที่แสดงในรูปที่ 7 แล้วทดลองคีย์ตามนี้

```
>PORT1 = 00 (หรือ 00H ก็ได้)          <ENTER>
```

เป็นการเขียนค่า 00H (00000000B) ให้แก่พอร์ต 1 ผลที่ได้คือหลอด แอลอีดี (LED) จะดับหมดทุกดวง

```
>PORT1 = 170 (หรือ 0AAH ก็ได้)        <ENTER>
```

เป็นการเขียนค่า AAH (10101010B) ให้แก่พอร์ต 1 ผลที่ได้คือหลอด แอลอีดี จะติดดวงเว้นดวง

```
>PRINT PORT1                          <ENTER>
 170
```

เป็นการอ่านค่าจากพอร์ต 1 แสดงค่าเป็นเลขฐานสิบ

```
>PHO.PORT1                            <ENTER>
 AAH
```

เป็นการอ่านค่าจากพอร์ต 1 แสดงค่าเป็นเลขฐานสิบหก

ตัวอย่าง การโปรแกรมให้หลอด แอลอีดี ติดทีละดวงวิ่งไล่กัน

```
>10 FOR I=0 TO 7
>20 A = 2**1
>30 PORT1 = A
>35 FOR J=1 TO 30 : NEXTJ
>40 NEXTI
```

```
>50 GOTO 10
>RUN                    <ENTER>
```

บทที่ 3
การออกแบบและการสร้าง

ดังที่ได้กล่าวมาแล้วในบทนำว่า สามารถแบ่งการทำงานของเครื่องเจาะอัตโนมัตินี้ออกเป็น 4 ส่วนใหญ่ๆ ดังนั้นเพื่อให้เป็นการง่ายต่อความเข้าใจจะขอกล่าวถึงการออกแบบและการสร้างในแต่ละส่วนดังต่อไปนี้

ชุดควบคุม

ในโครงงานเครื่องเจาะอัตโนมัตินี้ จะใช้ 8052 คอนโทรเลอร์ (8052 CONTROLLER) เป็นชุดควบคุมทำงานทั้งหมด ซึ่ง 8052 คอนโทร-เลอร์นี้ได้ออกแบบโครงสร้างในลักษณะเป็นโมดูล (MODULE) สามารถขยายการติดต่อกับคอนโทรเลอร์ โมดูลอื่นด้วยบัส (BUS) ขนาด40เส้น โดยมีคุณสมบัติเฉพาะตัว (SPECIFICATION) ของ 8052 คอนโทรเลอร์ดังนี้

CPU	8031,8051,8032,8052,8052AH-BASIC
CPU CLOCK	11.0592MHz
MEMORY	ROM1(U2) 8-32K SELECT 2764,27128, 27256
	ROM2(U4) 8-16K SELECT 2764,2864 (EEPROM),27128
	RAM(U3) 8-32KSELECT 6264,62256 (BACKUP)
BACKUP TIME	48HOUR
CHARGE TIME	48 HOUR
PORT	8255(U14)PRINTER,DIP-SWITCH
	8255(U19)USER PORT 24 BIT I/O
	8255(U15)KEYBOARD,DISPLAY
	8255(U18) D/A SOUND,USER LED,WATCH DOG
RTC	MM58167 (CLOCK 32.768 KHz)
SOUND	D/A R-LADDER & LM386 AMPLIFIER
CONNECTOR	40P SYSTEM BUS (Z80 COMPATABLE)
	20P PRINTER PORT (DB25 ON PANEL)
	26P USER PORT

26P KEY & DISPLAY
10P PORT1 OF CPU
2P POWER-LED (ON PANEL)
2P TXD-LED (ON PANEL)
2P PROG-EN. (SWITCH ON PANEL)
2P RESET SWITCH (SWITCH ON PANEL)
2P SPEAKER
3P RS232 (DB9 ON PANEL)
3P 2USER LED (ON PANEL)
4P POWER SUPPLY

JUMPER
2P X 2 SELECT U3
3P SELECT U2
2P WATCH DOG ENABLE
2P RTC INTERRUPT ENABLE

จะเห็นว่า 8052 คอนโทรเลอร์ ได้ออกแบบให้ใช้งานได้กับ ซีพียู หลายเบอร์คือ 8031,8051,8032,8052 และ 8052เอเอช เบสิก โดยผู้ใช้จะเลือกใช้งานได้ตามต้องการ ทั้งนี้กรณี 8032,8052 ยังสามารถใช้งานร่วมกับ 8052 คอนโทรเลอร์ ได้ดีด้วย คือโปรแกรม เบสิก มอนิเตอร์ (BASIC MONITOR) ที่จะช่วยให้การพัฒนาง่ายและสะดวกขึ้น ดังต่อไปนี้

8031,8051,8032,8052

ภาษาที่ใช้ แอสแซมบลี (ASSEMBLY)

การพัฒนา โดยผ่านการโปรแกรม อีพรอม หรืออุปกรณ์ประเภทแรม 2 ทาง

หมายเหตุ กรณีนี้ผู้ใช้ยังสามารถใช้ภาษา เบสิก-52 ได้เช่นกัน แต่จะต้องผ่านตัวแปร บีเอ็กซ์ซี51 (BXC51) ซึ่งเป็นโปรแกรม COMPILER บนเครื่อง PC โดยสามารถแปลภาษาเบสิก-52 เป็นภาษา แอสแซมบลี ของ 8031,8051,8032,8052 ได้ (กรณี 31,51 จะต้องไม่ใช้คำสั่งที่เกี่ยวกับ T2) แต่อย่างไรก็ตาม วิธีนี้ เหมาะสำหรับกรณีที่ จะนำโปรแกรมไปใช้งานจริงเท่านั้น ไม่เหมาะในการพัฒนาซึ่งมีขั้นตอนยุ่งยากเกินไป

8052AH-BASIC

ภาษาที่ใช้ เบสิก-52 และ แอสแซมบลี

การพัฒนา ผู้ใช้สามารถใช้งานด้วยภาษา เบสิก-52 ได้ทันที ตามคุณสมบัติของ ซีพียู (CPU) เบอร์นี้ โดยผ่านจอภาพและคีย์บอร์ดของเครื่อง พีซี ซึ่งจะต้องใช้โปรแกรมที่ทำให้ พีซี เป็นเสมือน เทอร์มินอล เช่น โปรแกรม ครอสทอล์ค (XTALK) และต่อกับ 8052 คอนโทรเลอร์ โดยสาย RS232

8032,8052 พร้อมกับ 8052 คอนโทรเลอร์

ภาษาที่ใช้ เบสิก-52 และ แอสแซมบลี

การพัฒนา ผู้ใช้จะใช้งานในทำนองเดียวกับ 8052เอเอช เบสิก แต่ตัวแปลภาษา เบสิก-52 จะอยู่ที่ อีพรอม ภายนอก ซึ่งกรณีนี้จะทำให้ไม่สามารถใช้คำสั่งเกี่ยวกับการโปรแกรมตัว อีพรอม ได้ อนึ่ง เบสิก-52 ที่อยู่ใน 8052 คอนโทรเลอร์นี้ จะมีคุณสมบัติเพิ่มเติมจากปกติ ซึ่งจะอธิบายในรายละเอียดต่อไป

8052เอเอช เบสิก พร้อมกับ 8052 คอนโทรเลอร์

ภาษาที่ใช้ เบสิก-52 และ แอสแซมบลี

การพัฒนา ผู้ใช้จะสามารถใช้งานได้สมบูรณ์ทุกอย่าง ทั้งคุณสมบัติของ เบสิก-52 และคุณสมบัติเพิ่มเติมจาก เบสิก มอนิเตอร์ ด้วย ซึ่งเหมาะอย่างยิ่งสำหรับการพัฒนาด้านไมโคร ด้วยภาษา เบสิก-52

8052 เบสิก มอนิเตอร์

8052 เบสิก มอนิเตอร์ คือโปรแกรมมอนนิเตอร์ในรูปของ อีพรอม โดยมี เบสิก อินเตอร์พิทเตอร์ (BASIC INTERPRETER) อยู่ในตัว และขณะเดียวกันได้เพิ่มคำสั่งพิเศษต่างๆ เข้าไป เพื่อให้ความสะดวกสูงสุดต่อผู้ใช้ ในการพัฒนา 8052 เบสิก มอนิเตอร์ มีคำสั่งเพิ่มอีก 7 คำสั่งโดยมีรายละเอียดดังนี้

XDOWN สำหรับการดาวน์โหลด (DOWN LOAD) ข้อมูลในลักษณะ INTEL-HEX FILE ลงใน แรม ทั้งนี้เพื่อให้ผู้ใช้โหลดส่วนที่เป็นข้อมูล หรือส่วนที่แปลมาจาก COMPILER ตัวอื่นๆ ได้ XDOWN นี้จะโหลดตามค่า แอดเดรส ที่อยู่ใน FILE เมื่อใช้คำสั่งนี้เครื่องจะรอ FILE ทางสาย RS232 ให้ผู้ใช้โหลด FILE ได้ โดยใช้คำสั่ง SE ในโปรแกรม ครอสทอล์ค ได้ทันที

SYNTAX: >XDOWN

XADOWN สำหรับการโหลดเช่นเดียวกับคำสั่ง XDOWN แต่จะมีการปรับค่าแอดเดรส (AUTO OFFSET) โดยอัตโนมัติ ทั้งนี้จะโหลดลงในหน่วยความจำที่ 10XXH เสมอ ค่า XX คือค่าแอสเดรส 2 หลักท้ายที่อยู่ใน INTEL-HEX FILE คำสั่งนี้สำหรับการโหลดข้อมูลลง แรม เพื่อใช้โปรแกรมลงในตัว อีพรอม

SYNTAX: >XADOWN

XUP สำหรับการ อัพ โหลด (UP LOAD) ข้อมูลขึ้นบนเครื่อง พีซี ในรูปแบบ INTEL-HEX FILE ซึ่งเป็นการทำงานตรงข้ามกับคำสั่ง XDOWN ขบวนการ อัพ โหลด นี้ ผู้ใช้จำเป็นต้องใช้คำสั่ง CA ในโปรแกรม XTALK เข้าร่วมด้วย

SYNTAX: >XUP START,END
START = ค่า ADDRESS เริ่มต้น (HEX)
END = ค่า ADDRESS สุดท้าย (HEX)

EXAMPLE: >XUP 1000,10FF

BUP สำหรับการ อัพ โหลด โปรแกรมภาษา เบสิก-52 ที่อยู่ในหน่วยความจำขึ้นบนเครื่อง พีซี โดยมีหลักการในทำนองเดียวกับคำสั่ง XUP เพียงแต่จะทำการคำนวณค่า แอดเดรส ให้ โดยอัตโนมัติ

SYNTAX: >BUP

BLANK สำหรับการล้างตัว อีสแคว์พรอม (EEPROM) ที่อยู่ตำแหน่ง U4 โดยจะทำการใส่ค่า 0FFH ลงใน EEPROM ทั้งหมด ทั้งนี้เพื่อความสะดวกกับการทำงานของคำสั่ง PROG ของ เบสิก-52 ซึ่งจะช่วยให้มีสภาพเหมือนกับเริ่มงานใหม่ โดยจะทำให้การเก็บโปรแกรมมีเนื้อที่สูงสุดตามขนาดของตัว EEPROM

XPROG สำหรับการโปรแกรมตัว อีพรอมหรืออีสแคว์พรอม โดยสามารถโปรแกรมที่ U4 ได้ในบอร์ด กรณีเป็น อีพรอม ผู้ใช้ต้องกำหนด VPP เองจากแหล่งจ่ายไฟภายนอก โดยเข้าที่ขา VP จากขั้วต่อสาย การโปรแกรมจะโปรแกรมจาก U3 (แรม) ไปที่ U4 หรือจาก U4 ไป U4 เท่านั้น

SYNTAX: >XPROG SOURCE,CEST,BYTE
SOURCE = ค่า ADDRESS เริ่มต้นของข้อมูล (HEX)
DEST = ค่า ADDRESS เริ่มต้นของตัว EPROM (HEX)
BYTE = ค่าจำนวนความยาวของข้อมูล (HEX)

INIT สำหรับการเริ่มต้นระบบ ไอ/โอ (I/O) บนบอร์ด 8052 คอนโทรเลอร์ โดยจะทำให้การใช้คำสั่งเพิ่มเติมจาก 8052 เบสิก มอนิเตอร์ เป็นไปอย่างสมบูรณ์ ทั้งนี้ควรจะทำคำสั่งนี้ก่อนทุกครั้งที่ใช้งานบอร์ดโดยผ่าน 8052 เบสิก มอนิเตอร์ หรือใส่ไว้ที่ต้นโปรแกรมของผู้ใช้ในกรณีที่โปรแกรมนี้เป็นโปรแกรมที่ทำงานทันทีที่เปิดเครื่อง และต้องการใช้คำสั่งเพิ่มเติมของ 8052 เบสิก มอนิเตอร์ คำสั่งนี้จะกระทำการ เซ็ตอัพ ค่า คอนโทรโคด (CONTROL CODE) ให้กับพอร์ต 8255 ทั้งหมดในบอร์ด ยกเว้น

8255 ที่เป็น USER PORT

SYNTAX: >INIT

PRINT@ คำสั่งนี้เป็นคำสั่งปกติของ เบสิก-52 อยู่แล้ว แต่สำหรับ 8052 เบสิก มอนิเตอร์ จะกำหนดให้เป็นการสั่งพิมพ์เพื่อออกเครื่องพิมพ์โดยเฉพาะ ทั้งนี้จากโครงสร้างของ เบสิก-52 เอง ที่อนุญาตให้มีการกำหนดโปรแกรมย่อย เพื่อใช้เป็นส่วนส่งข้อมูลออกทาง ไอ/โอ ตามลักษณะของ ฮาร์ดแวร์ นั้นๆ สำหรับ 8052 เบสิก มอนิเตอร์ จะส่งข้อมูลออกทาง พาราแลล์ พอร์ต (PARALLEL PORT) ซึ่งอยู่ในบอร์ดเรียบร้อยแล้ว ทั้งนี้เพื่อความสะดวกของผู้ใช้ในการพิมพ์โปรแกรมออกทางเครื่องพิมพ์ รวมทั้งในการพัฒนาโปรแกรมที่เกี่ยวข้องกับการใช้เครื่องพิมพ์ด้วย ซึ่งช่วยให้ง่ายและทำได้รวดเร็ว

EXAMPLE: >PRINT @ "Automatic driller....."

วิธีการใช้งานโปรแกรมภาษา เบสิก-52

3.1 ต่อสาย RS232 ระหว่างเครื่องคอมพิวเตอร์กับ 8052 คอนโทรเลอร์ให้เรียบร้อย

3.2 บูท (BOOT) เครื่องคอมพิวเตอร์จากแผ่น ดิสก์ (DISK) XTALK เครื่องจะเข้าโปรแกรม XTALK ให้ โดยจะอยู่ในสภาวะพร้อมที่จะทำการสื่อสาร

3.3 จากนั้นเปิดเครื่อง 8052 คอนโทรเลอร์ และกด SPACE ที่เครื่องคอมพิวเตอร์ เบสิก-52 จะทำการคำนวณค่าบอดเลต (BAUD RATE) ให้อัตโนมัติ และจะแสดงที่จอดังนี้

```
*MCS-51(TM) BASIC V1.1*
READY
>INIT
>
```

ก่อนอื่นใดให้ผู้ใช้ ทำคำสั่ง INIT เสียก่อน เพื่อเป็นการกำหนด CONTROL CODE ให้กับ 8255 ที่อยู่ในบอร์ด ซึ่งจะทำให้การใช้งานของ 8052 คอนโทรเลอร์ สมบูรณ์ที่สุด

3.4 เมื่อต้องการ ดาวน์ โหลด โปรแกรมจากแผ่น ดิสก์ ให้คีย์คำสั่ง XDOWN และกด ENTER โดยเครื่องจะแสดงข้อความให้ผู้ใช้ทราบว่ากำลังรอการส่งข้อมูลจากเครื่องคอมพิวเตอร์ดังนี้

```
>XDOWN
Wait for DOWNLOAD Intel Hex file.....
```

3.5 ที่จุดนี้ให้ผู้ใช้ทำขบวนการส่งข้อมูลจากโปรแกรม XTALK โดยกด ^A เพื่อเข้าสู่โหมด คอมมานด์ (COMMAND) และใช้คำสั่ง SE และตามด้วยชื่อ FILE ดังนี้

```
COMMAND: SE ONTIME.BHX
         SEnding file A:ONTIME.BHX
```

เครื่องจะส่ง FILE ไปยัง 8052 คอนโทรเลอร์ ตามต้องการ และเมื่อเรียบร้อยแล้วก็จะกลับมาที่ พร็อมพ์ (PROMPT) ของ เบสิก-52 ตามเดิม ผู้ใช้สามารถ ลิสท์ (LIST) โปรแกรมดูได้ ซึ่งจะเห็นว่าโปรแกรมได้อยู่ใน 8052 คอนโทรเลอร์ เรียบร้อยแล้ว

การใช้งานโปรแกรม ครอสทอร์ค (XTALK)

โปรมแกรม XTALK คือโปรแกรมสำหรับการสื่อสารทั่วไป โดยในที่นี้จะใช้สื่อสารในแบบ โลคอล (LOCAL) ซึ่งเป็นการสื่อสารโดยผ่านสาย อาเอส232 โปรแกรมนี้มีรายละเอียดค่อนข้างมาก แต่ในที่นี้จะสรุปการใช้งานพอสังเขป และจะเน้นที่การใช้งานกับ 8052 คอนโทรเลอร์ เป็นแนวทางหลัก โดยพอจะสรุปเป็นข้อๆ ได้ดังนี้

โปรแกรมในแผ่นดิสก์ที่ให้มาจะทำ ออโต้เอ็กซิคิว (AUTOEXEC) ให้เรียบร้อยแล้ว โดยเป็นการใช้งานตามการกำหนดคือ BAUD RATE=9600, STOP BIT=1,PARITY=NONE,DATA=8,PORT=COM1, CWAIT=DELAY 2, LOCAL COMMUNICATION

การใช้งานมีโหมดต่างๆ คือ

-โหมดการสื่อสาร โดยจะอยู่ในขบวนการรับและส่งข้อมูลทางสาย และที่บรรทัดล่างสุดจะแสดงสถานะบางอย่าง พร้อมทั้งแสดงคำสั่ง (^A) ในการเปลี่ยนโหมดด้วย

-โหมด คอมมานด์ (COMMAND) จะเป็นการรับคำสั่งต่าง ๆ ของ XTALK ในจุดนี้ผู้ใช้จะขอดูตัวแปรต่างๆของการสื่อสารได้ด้วยคำสั่ง ^F คำสั่งทั้งหมดของ XTALK จะดูได้ด้วยคำสั่ง HE (HELP) ในโหมดนี้สิ่งต่างๆ ที่ปรากฏบนจอจะเกิดขึ้นจาก XTALK เอง ไม่ใช่สิ่งที่มาจากการสื่อสาร ผู้ใช้จะกลับไปโหมดการสื่อสารได้ด้วยการกด ENTER หรือคำสั่ง GO LOCAL

การใช้คำสั่ง CA (CATURE) เพื่อการเก็บสิ่งต่างๆ ที่ปรากฏบนจอลงใน FILE จะทำได้ โดยใช้ CA และตามด้วยชื่อ FILE หลังจากนั้นสิ่งที่เกิดขึ้นบนจอทุกอย่าง (ที่มาจากการสื่อสาร) จะถูกเก็บลงใน FILE และผู้ใช้จะสิ้นสุดขบวนการได้ด้วยคำสั่ง CA OFF กรณีนี้จะใช้สำหรับการเก็บโปรแกรมภาษา เบสิก-52 ในแบบ TEXT FILE ได้ หรือใช้ร่วมกับคำสั่งของ 8052 เบสิก มอนิเตอร์ เพื่อการ อัพ โหลด ข้อมูลต่างๆจาก 8052 คอนโทรเลอร์ ขึ้นเครื่อง พีซี

คำสั่ง SE (SEND) ใช้สำหรับการส่งข้อมูลจาก FILE ออกทางสาย ซึ่งเป็นขบวนการ ดาวน์ โหลด จากเครื่อง พีซี ไปยัง 8052 คอนโทรเลอร์ โดยใช้งานร่วมกับคำสั่งใน 8052 เบสิก มอนิเตอร์ การใช้คำสั่ง SE และ CA จะช่วยให้การใช้งานของผู้ใช้สะดวกมากยิ่งขึ้น ซึ่งสามารถจะทำการ อัพ โหลด และ ดาวน์ โหลด ข้อมูลไปมา ทำให้การเก็บโปรแกรม หรือข้อมูลต่างๆ กระทำอยู่บนเครื่อง พีซี ได้

ชุดวงจรขับ

เนื่องจากสเต็ปปิ้งมอเตอร์ที่ใช้ เป็นสเต็ปปิ้งมอเตอร์ แบบ 4 เฟส ดังนั้นเราจะใช้ทรานซิสเตอร์ (TRANSISTER) 8 ตัวต่อสเต็ปปิ้งมอเตอร์ 1 ตัวสำหรับแกนเอ็กซ์และแกนวาย และ 4 ตัวต่อมอเตอร์ 1 ตัว สำหรับแกนแซด และมีทรานซิสเตอร์ ทำหน้าที่เป็นสวิทช์ควบคุมรีเลย์ในการปิด-เปิดไฟ 15 โวลท์ จ่ายให้แก่ตัวสว่าน

จากวงจรขับที่ต่อเข้ากับพอร์ตของชุดควบคุมจะประกอบด้วย Q1-Q4 เบอร์ 2N2222 ซึ่งรับสัญญาณจากพอร์ตของชุดควบคุม เพื่อนำไปขับให้แก่ Q1-Q12 เบอร์ 2N3055 สำหรับแกนเอ็กซ์ และ Q5-Q8, Q13-Q16 สำหรับแกนวาย ในลักษณะเดียวกัน ส่วนวงจรขับของแกนแซด จะใช้ทรานซิสเตอร์ Q17-Q20 เบอร์ 2N6043 และ Q21 เบอร์ 2N222 เป็นสวิทช์ไฟให้คอยล์

ของรีเลย์ ส่วนไดโอด (DIODE) ใช้สำหรับเป็นตัวกันแบ็คอีเอ็มเอส (BACK EMS) ของมอเตอร์แต่ละตัว แหล่งจ่ายไฟ Vcc และ +V มีขนาด 5 โวลท์

ชุดแท่นเครื่องเจาะ

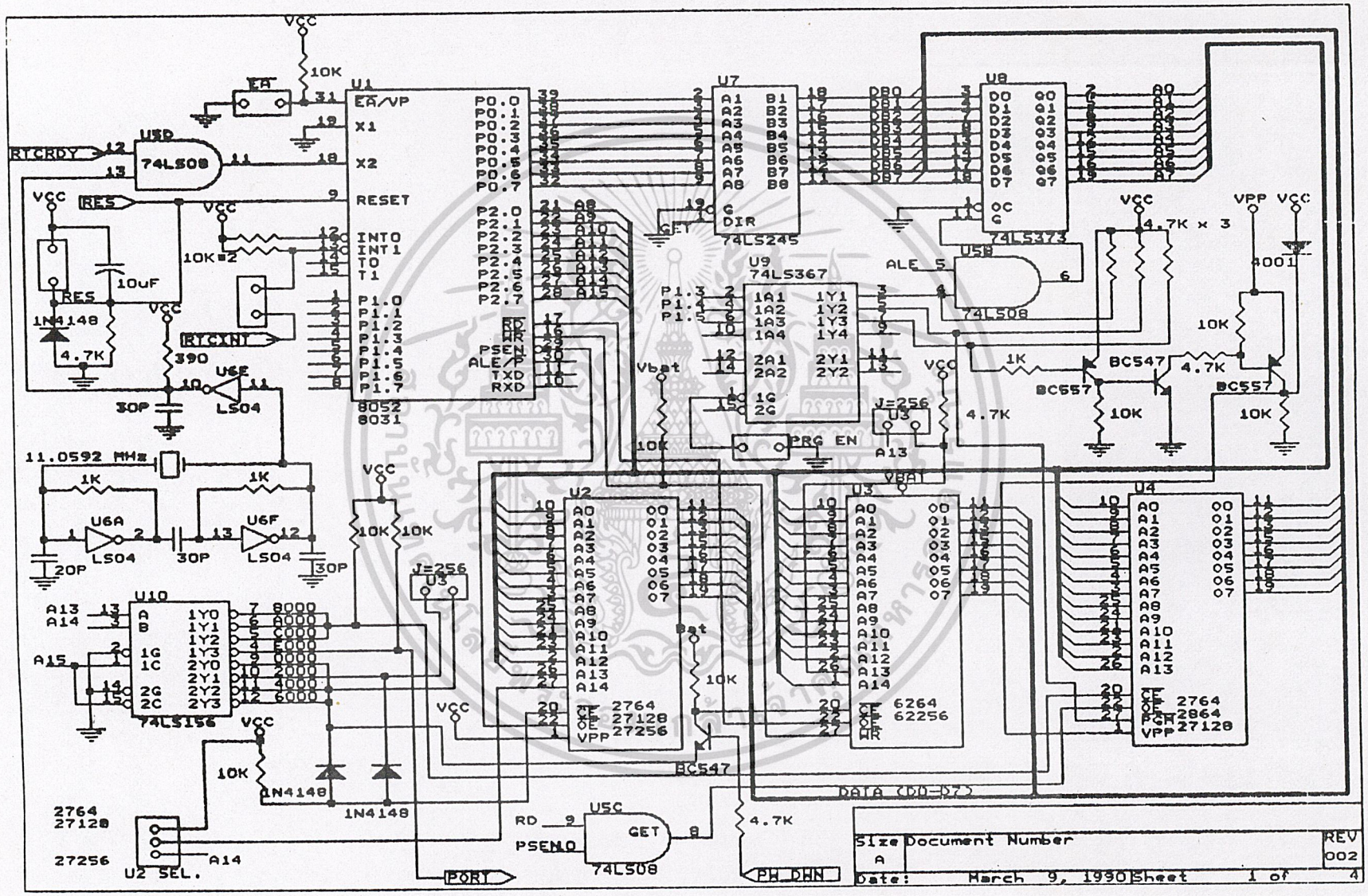
ตัวแท่นเจาะจะประกอบจากอลูมิเนียม ใช้ลีดสกลูยาว ครึ่งเมตรจำนวน 2 ท่อน สำหรับเป็นตัวเลื่อนตัวสว่านไปมา บนแกนเอ็กซ์ และ แกนวาย โดยมีเพลาสำหรับรับน้ำหนักอยู่บนแกนทั้งสอง ส่วนแกนแซดจะใช้ลีดสกลูขนาดเล็ก 6 นิ้ว เป็นตัวเลื่อนตัวสว่านในแกนแซด โดยมีเพลารับน้ำหนักอยู่ 2 ข้างเช่นกัน

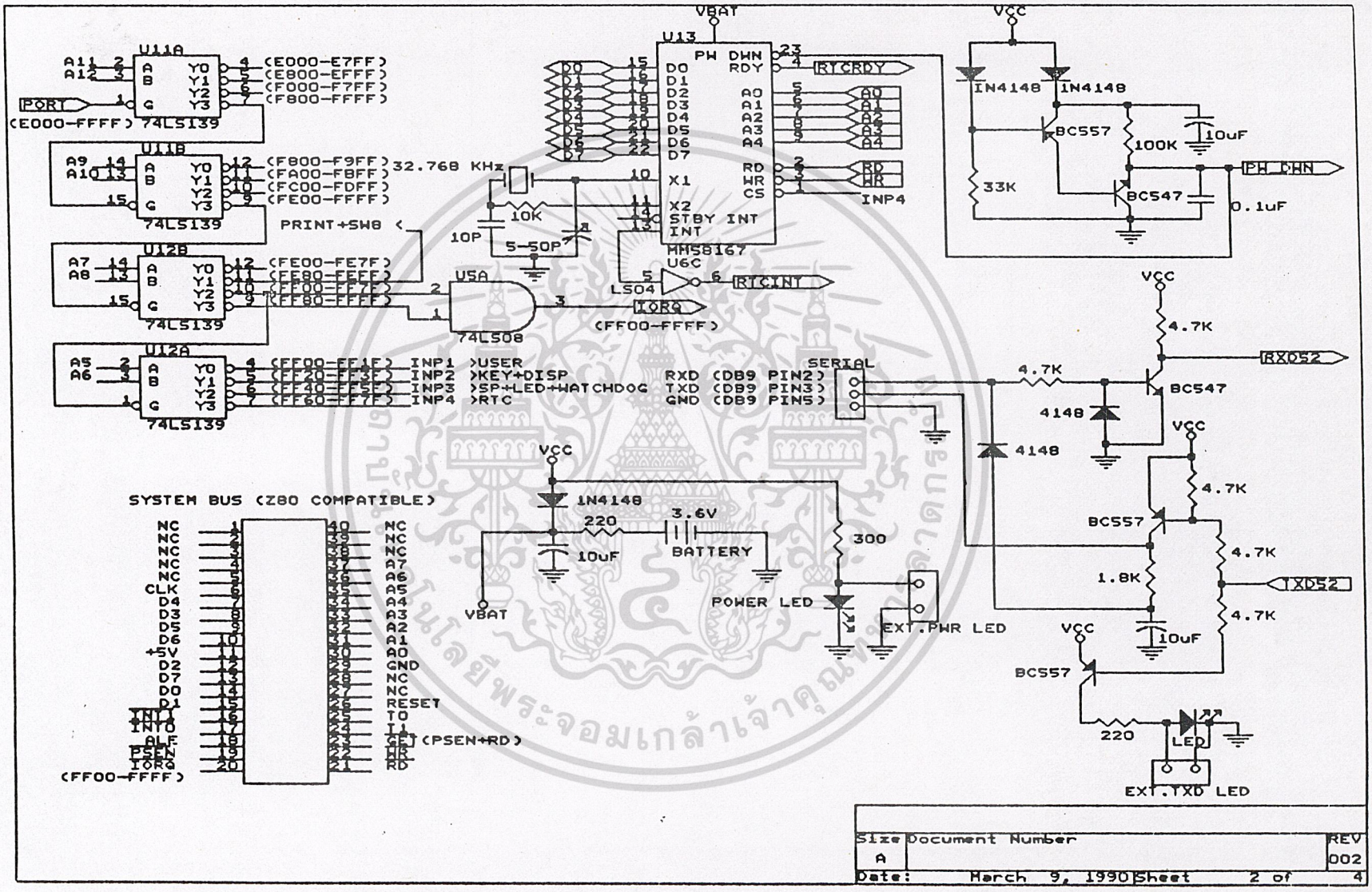
ส่วนมอเตอร์ที่จะนำไปขับลีดสกลูให้หมุนนั้น สำหรับแกนเอ็กซ์และแกนวายจะใช้วิธีขับผ่านสายพาน จากแกนของมอเตอร์ไปยังลีดสกลู เพื่อเป็นการลดแรงให้แก่ตัวมอเตอร์ สำหรับแกนแซดนั้น จะใช้วิธีขับลีดสกลูโดยตรง เพราะความยาวของแกนไม่มากนัก

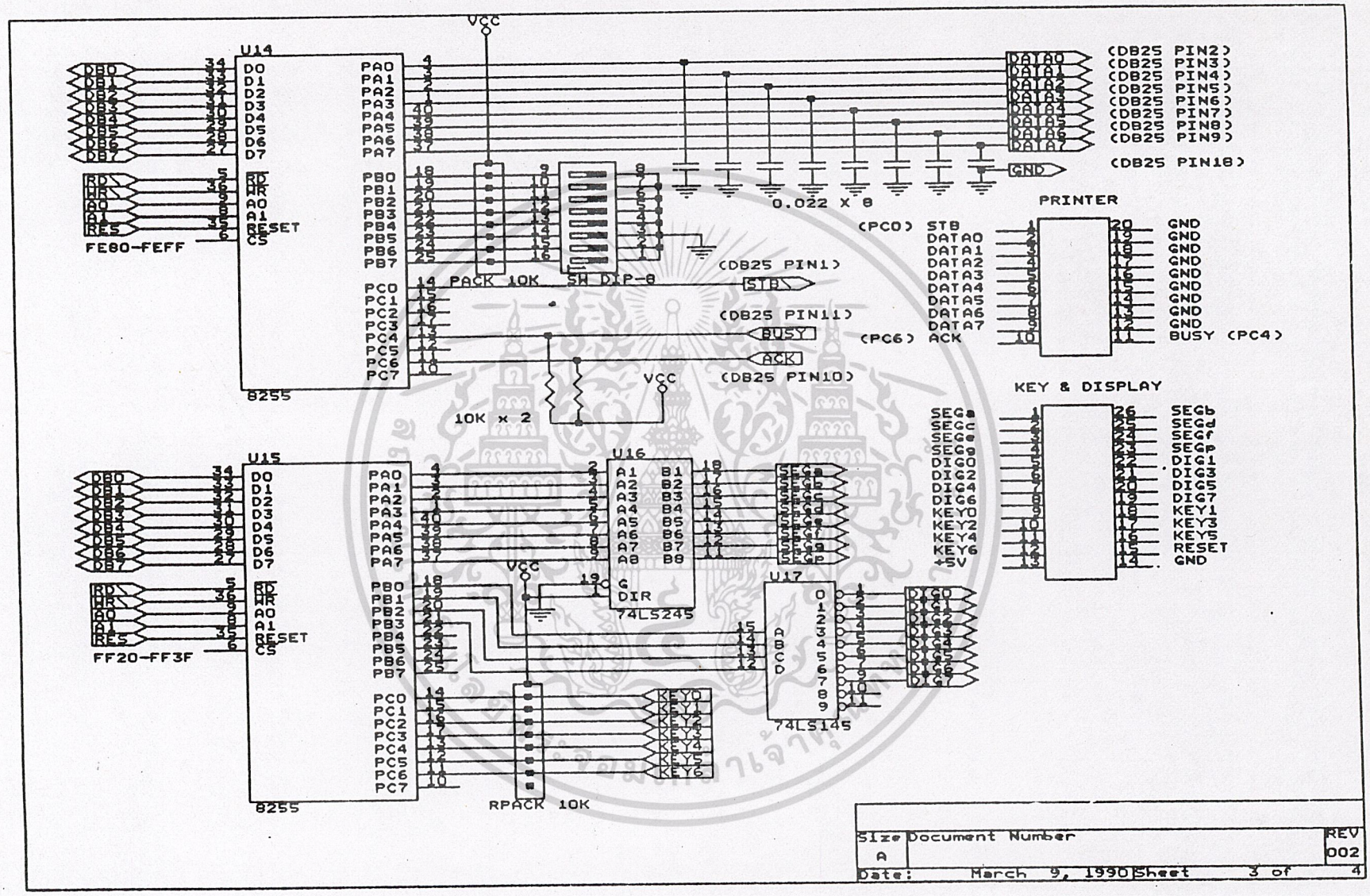
ลักษณะของการเคลื่อนที่ของแท่นเจาะที่ใช้กันทั่วไปจะมีอยู่ 2 ลักษณะคือ การเคลื่อนที่โดยการเลื่อนไปทั้งตัวแท่น ในลักษณะนี้ตัวสว่านจะอยู่กับที่แต่ชิ้นงานจะเป็นตัวเคลื่อนที่เอง ส่วนลักษณะที่ 2 จะเป็นการเคลื่อนที่โดยการเลื่อนแกนซึ่งมีตัวสว่านติดอยู่ไปมา แต่แท่นที่วางชิ้นงานจะอยู่กับที่ สำหรับโครงงานนี้จะเลือกใช้วิธีที่ 2 เนื่องจากไม่ยากเกินไปสำหรับการประกอบและนำไปใช้ในงานไม่หนักมากนัก ตัวสว่านเองก็มีน้ำหนักเบา สามารถเคลื่อนตัวอยู่บนแกนได้ ดังนั้นจึงไม่มีความจำเป็นที่จะต้องใช้วิธีแรก ซึ่งประกอบยากกว่า และเสียค่าใช้จ่ายมากกว่าด้วย

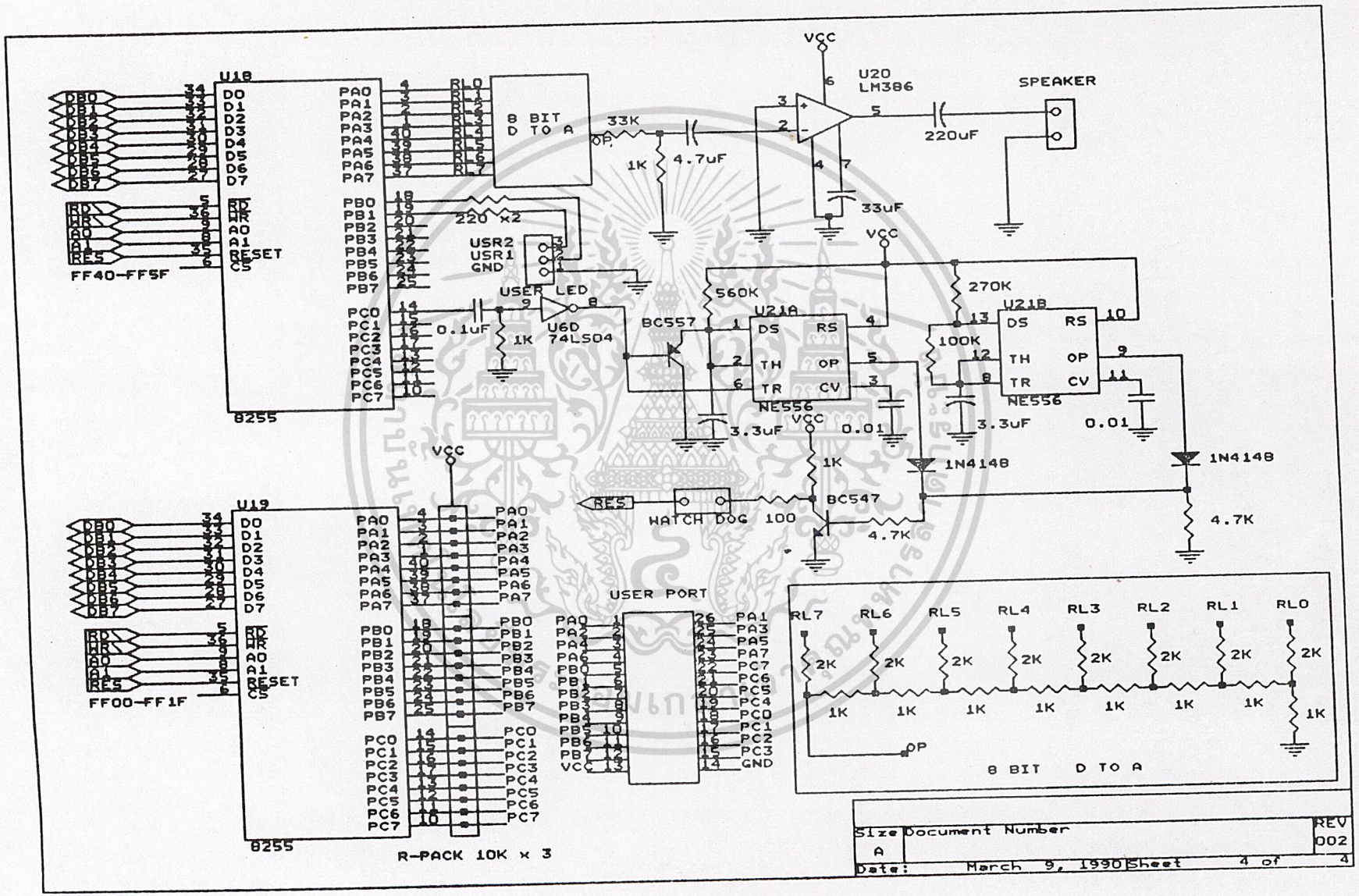
ชุดโปรแกรมควบคุมระบบการทำงานของเครื่องเจาะอัตโนมัติ

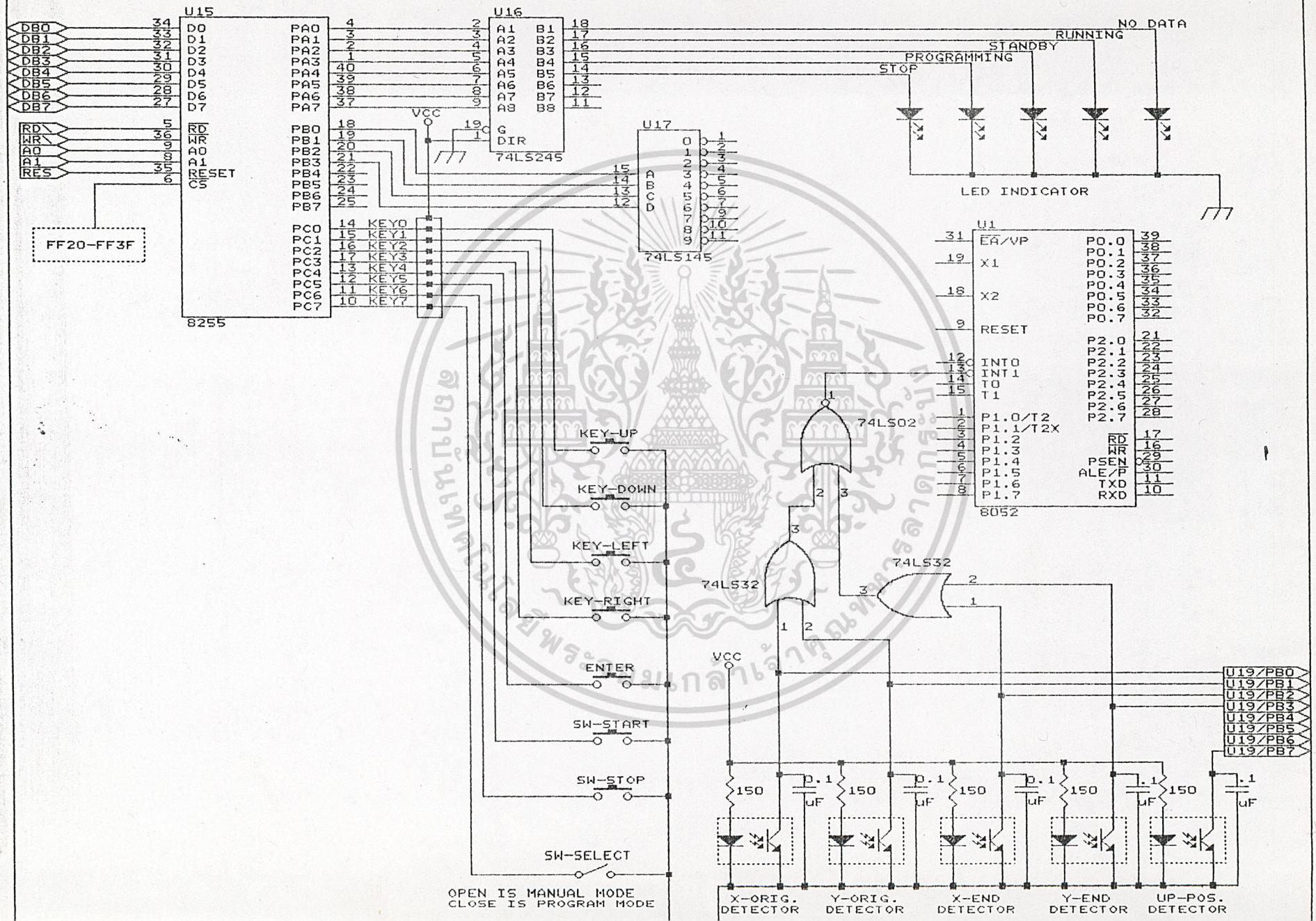
เนื่องจากโครงงานนี้ใช้ ซีพียู เป็นตัวควบคุมการทำงาน ดังนั้นจึงจำเป็นต้องมีโปรแกรมคอยสั่งการเพื่อควบคุมการทำงานตามที่ต้องการได้ ซึ่งโปรแกรมที่เขียนขึ้นใช้กับเครื่องเจาะอัตโนมัตินี้ จะใช้ภาษา เบสิก-52 เป็นหลัก ซึ่งลิสท์ของโปรแกรมควบคุมเครื่องเจาะนี้ดูได้จากบทที่ 4

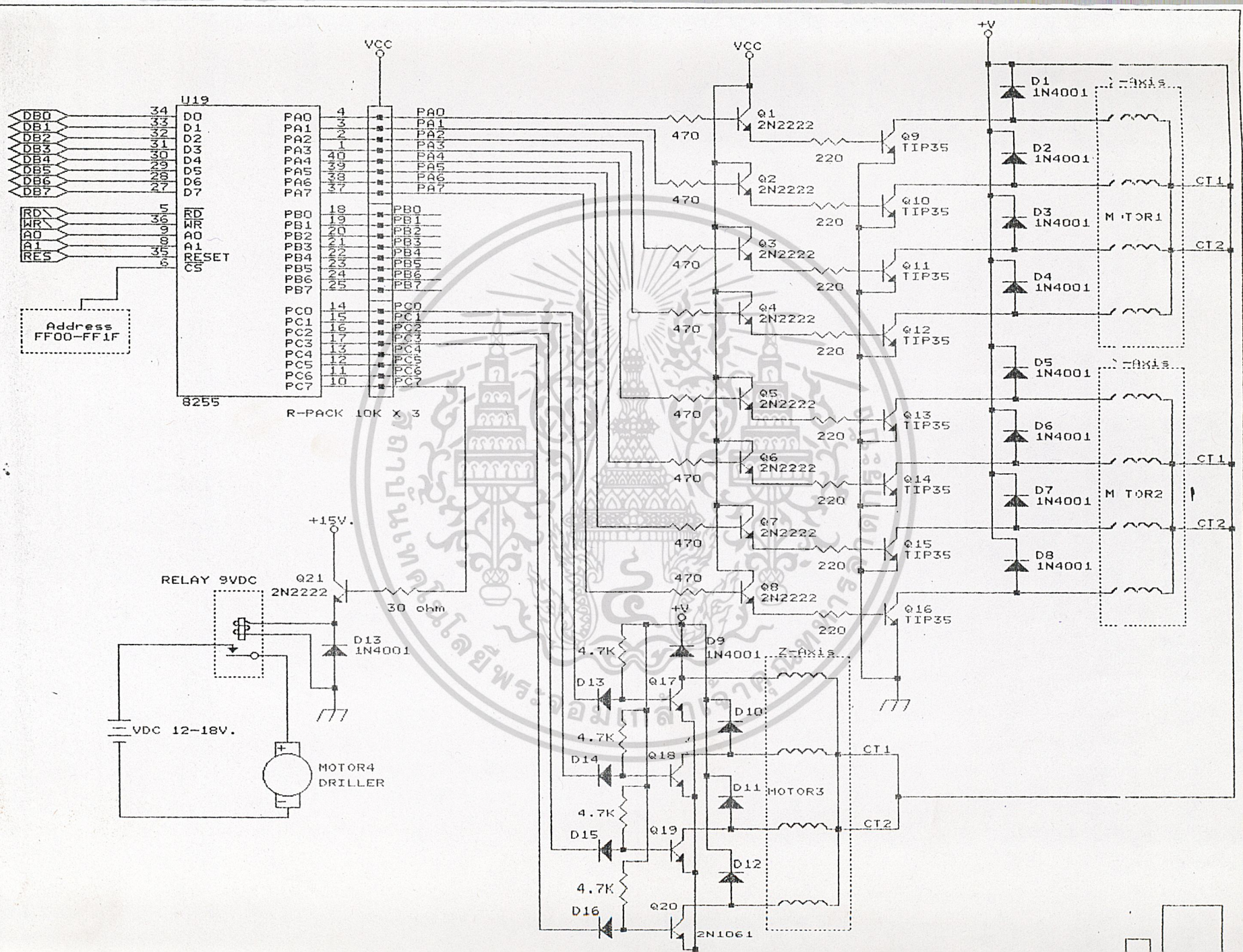

VCC
10K
EA
U1
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
8052
8031
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
RD
WR
PSEN
ALE/P
TXD
RXD
U5D
74LS08
RTCRDY
RES
10uF
1N4148
4.7K
390
RTCXNT
U6E
LS04
30P
11.0592 MHz
1K
U6A
U6F
20P
U10
74LS156
A13
A14
A15
8000
A000
C000
E000
0000
2000
4000
6000
10K
1N4148
2764
27128
27256
U2 SEL.
PORT
U7
74LS245
A1
A2
A3
A4
A5
A6
A7
A8
B1
B2
B3
B4
B5
B6
B7
B8
G
DIR
GET
U9
74LS367
P1.3
P1.4
P1.5
1A1
1A2
1A3
1A4
2A1
2A2
1G
2G
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
PRG EN
Vbat
U8
74LS373
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
OC
G
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A0
A1
A2
A3
A4
A5
A6
A7
U5B
74LS08
ALE
4.7K x 3
VPP
4001
BC547
BC557
1K
U2
U3
U4
2764
27128
27256
6264
62256
J=256
A13
Bat
DATA (D0-D7)
U5C
RD
PSEN
PH_DWN
Size
A
Document Number
REV
002
Date:
March 9, 1990
Sheet
1 of
4

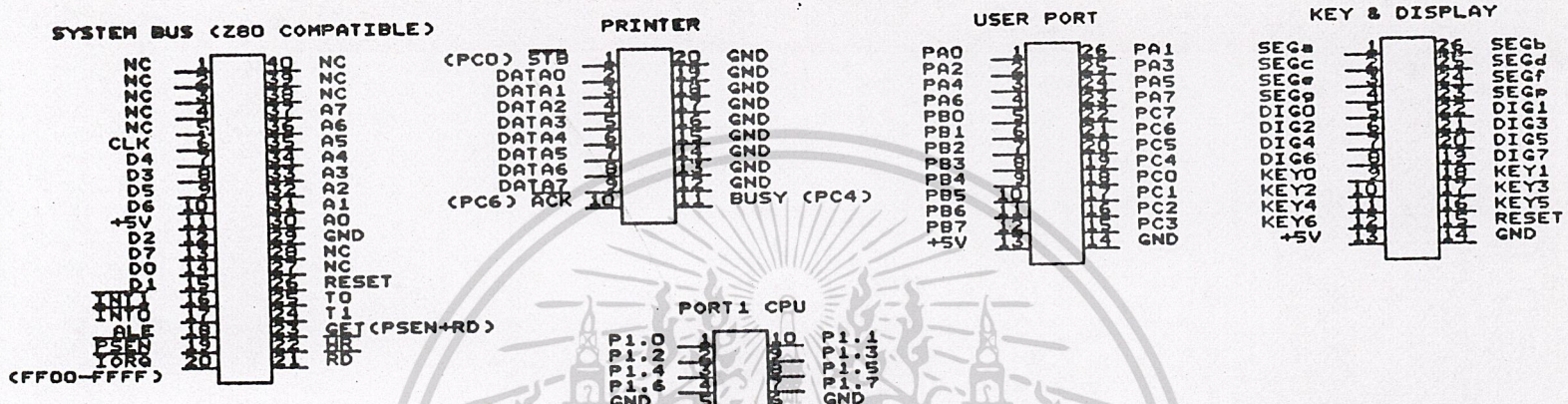



U14
8255
FE80-FEFF
PACK 10K
SW DIP-8
0.022 X 8
(DB25 PIN1)
(DB25 PIN11)
(DB25 PIN10)
10K x 2
PRINTER
(PC0) STB
(PC6) ACK
BUSY (PC4)
KEY & DISPLAY
U15
8255
FF20-FF3F
RPACK 10K
U16
74LS245
U17
74LS145
Size A
Document Number
REV 002
Date: March 9, 1990
Sheet 3 of 4


U18
8255
FF40-FF5F
8 BIT
D TO A
33K
1K
4.7uF
VCC
U20
LM386
SPEAKER
220uF
33uF
220 x2
USR2
USR1
GND
USER LED
U6D
74LS04
0.1uF
1K
560K
BC557
U21A
NE556
3.3uF
0.01
270K
U21B
100K
1N4148
4.7K
RES
WATCH DOG
100
BC547
U19
FF00-FF1F
R-PACK 10K x 3
USER PORT
RL7
RL6
RL5
RL4
RL3
RL2
RL1
RL0
2K
8 BIT D TO A
Size
A
Document Number
REV
002
Date: March 9, 1990 Sheet 4 of 4


U15
8255
U16
74LS245
U17
74LS145
U1
8052
74LS02
74LS32
74LS32
FF20-FF3F
LED INDICATOR
NO DATA
RUNNING
STANDBY
PROGRAMMING
STOP
KEY-UP
KEY-DOWN
KEY-LEFT
KEY-RIGHT
ENTER
SW-START
SW-STOP
SW-SELECT
OPEN IS MANUAL MODE
CLOSE IS PROGRAM MODE
VCC
X-ORIG. DETECTOR
Y-ORIG. DETECTOR
X-END DETECTOR
Y-END DETECTOR
UP-POS. DETECTOR

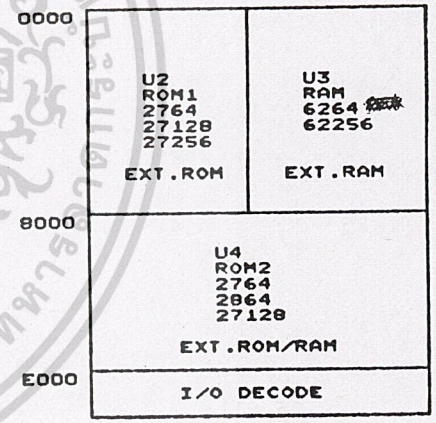

U19
8255
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
RD
WR
A0
A1
RES
CS
RESET
Address
FF00-FF1F
VCC
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
R-PACK 10K X 3
470
220
Q1 2N2222
Q2 2N2222
Q3 2N2222
Q4 2N2222
Q5 2N2222
Q6 2N2222
Q7 2N2222
Q8 2N2222
Q9 TIP35
Q10 TIP35
Q11 TIP35
Q12 TIP35
Q13 TIP35
Q14 TIP35
Q15 TIP35
Q16 TIP35
+V
D1 1N4001
D2 1N4001
D3 1N4001
D4 1N4001
D5 1N4001
D6 1N4001
D7 1N4001
D8 1N4001
X-Axis
Y-Axis
CT1
CT2
MOTOR1
MOTOR2
+15V.
RELAY 9VDC
Q21 2N2222
30 ohm
D13 1N4001
VDC 12-18V.
MOTOR4
DRILLER
4.7K
D9 1N4001
Z-Axis
Q17
Q18
Q19
Q20
D10
D11
D12
MOTOR3
2N1061



INPUT/OUTPUT DECODE TABLE

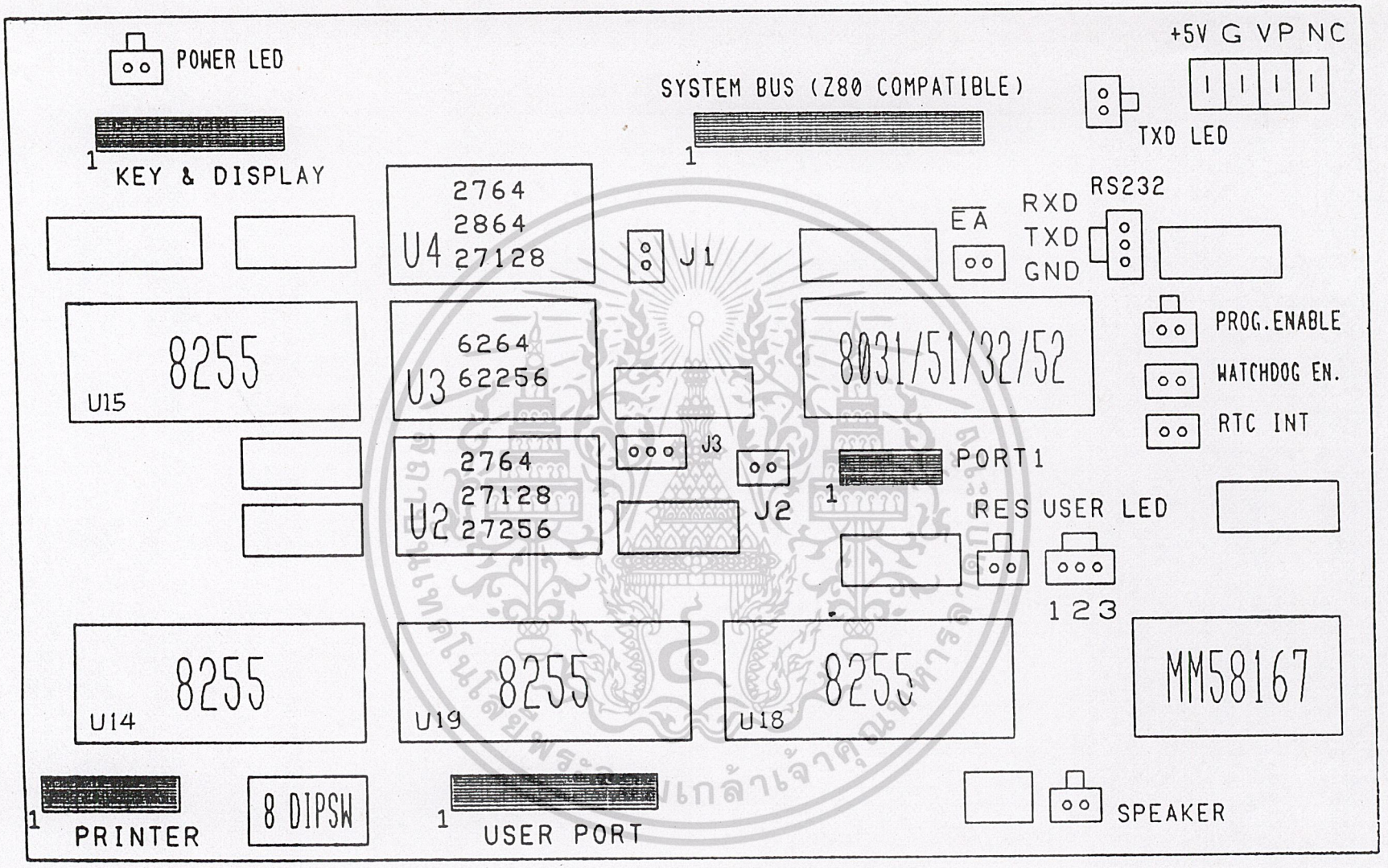
ADDRESS DECODE	ADD. USE	PORT	TYPE	DESCRIPTION
FE80-FEFF	FE80	A	O	PRINTER(DATA)
	FE81	B	I	DIP-SWITCH
	FE82	C	I/O	PRINTER(STB,BUSY,ACK)
	FE83	-	O	8255 CONTROL CODE
FF00-FF1F	FF00	A	ANY	USER
	FF01	B	ANY	USER
	FF02	C	ANY	USER
	FF03	-	O	8255 CONTROL CODE
FF20-FF3F	FF20	A	O	SEGMENT DRIVE
	FF21	B	O	DIGIT SELECT
	FF22	C	I	KEY BOARD
	FF23	-	O	8255 CONTROL CODE
FF40-FF5F	FF40	A	O	D/A SOUND
	FF41	B	O	USER LED
	FF42	C	O	WATCH DOG
	FF43	-	O	8255 CONTROL CODE
FF60-FF7F	FF60-FF77	-	I/O	REAL TIME CLOCK

MEMORY MAP



Size	Document Number	REV
A		KB
Date: March 8, 1990	Sheet of	

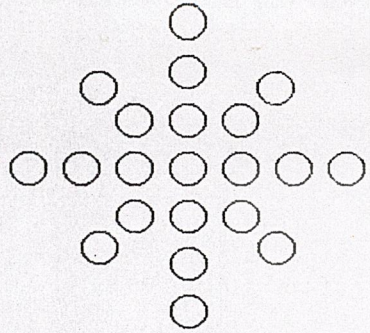
CONNECTOR PINOUT และ DECODE MAP



SELECT U2 2764/128 J3=LEFT
J3=RIGHT

SELECT U3 6264 J1=J2=OPEN
62256 J1=J2=CLOSE

FRONT PANEL LAOUT



ON

OFF

AUTOMATIC DRILLER.....

INDICATOR

- RUNNING
- STOP
- PROGRAMMING
- NO DATA
- STANDBY

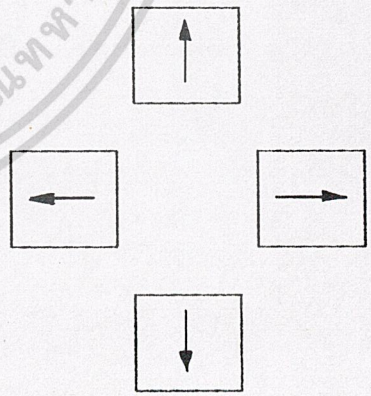
MANUAL MODE

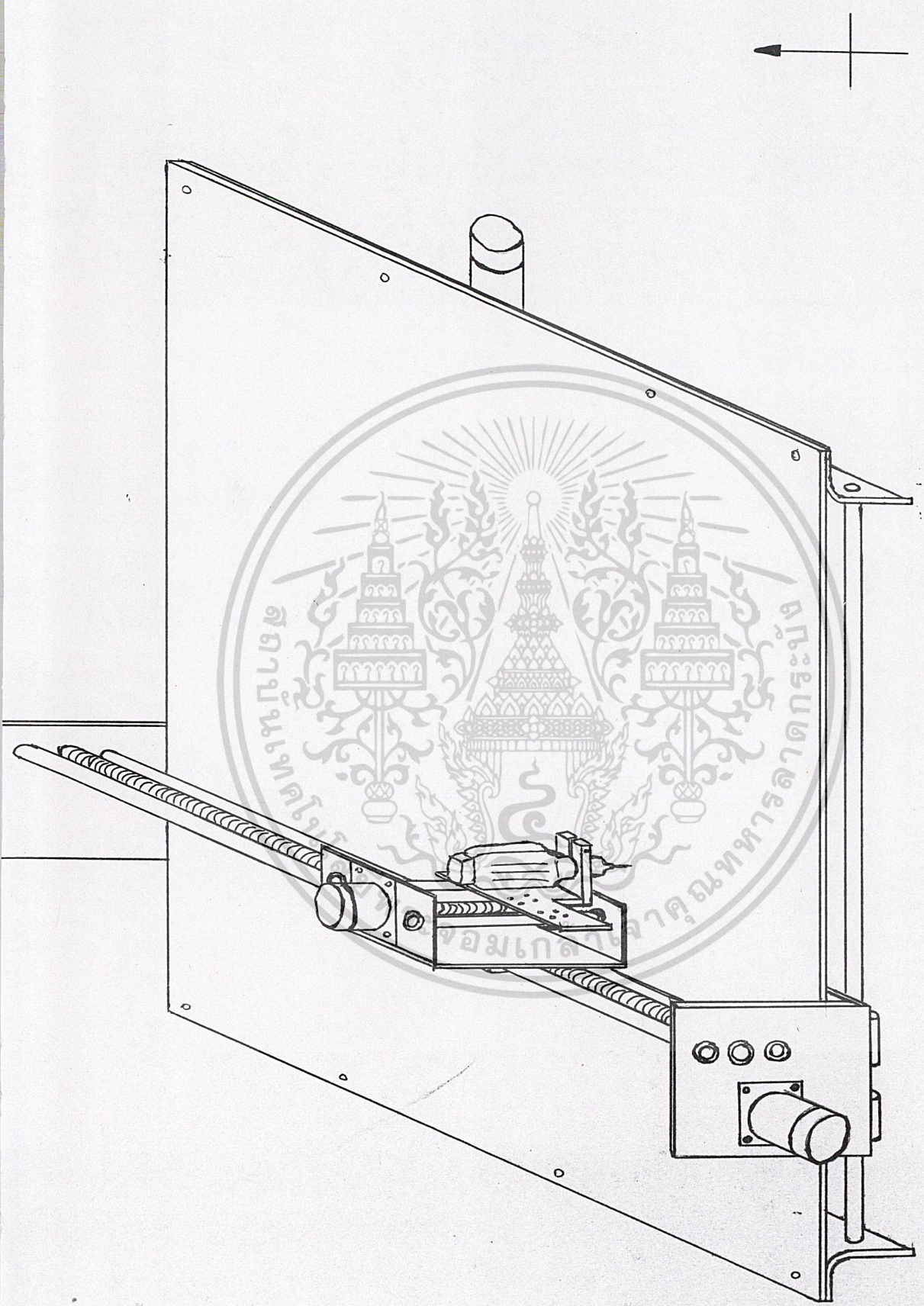
PROGRAM MODE

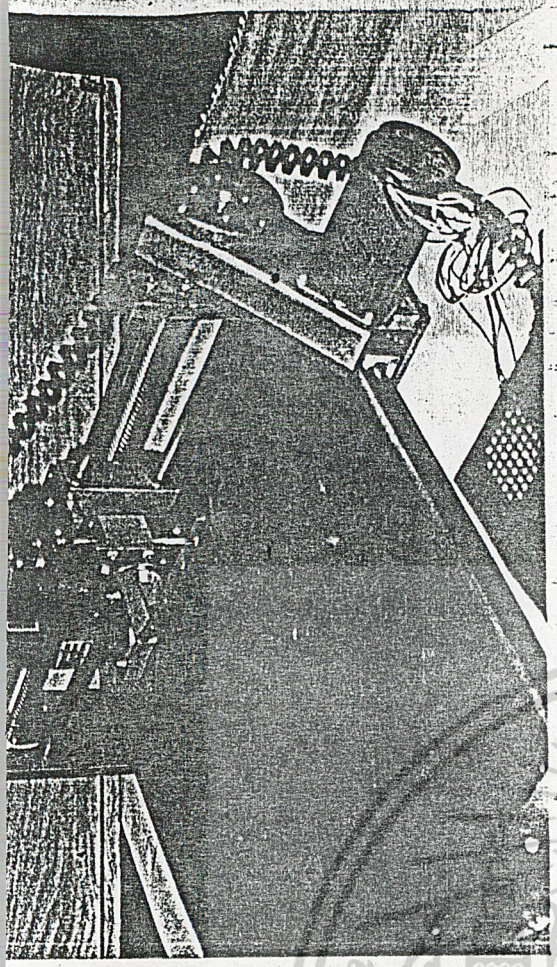
SRT

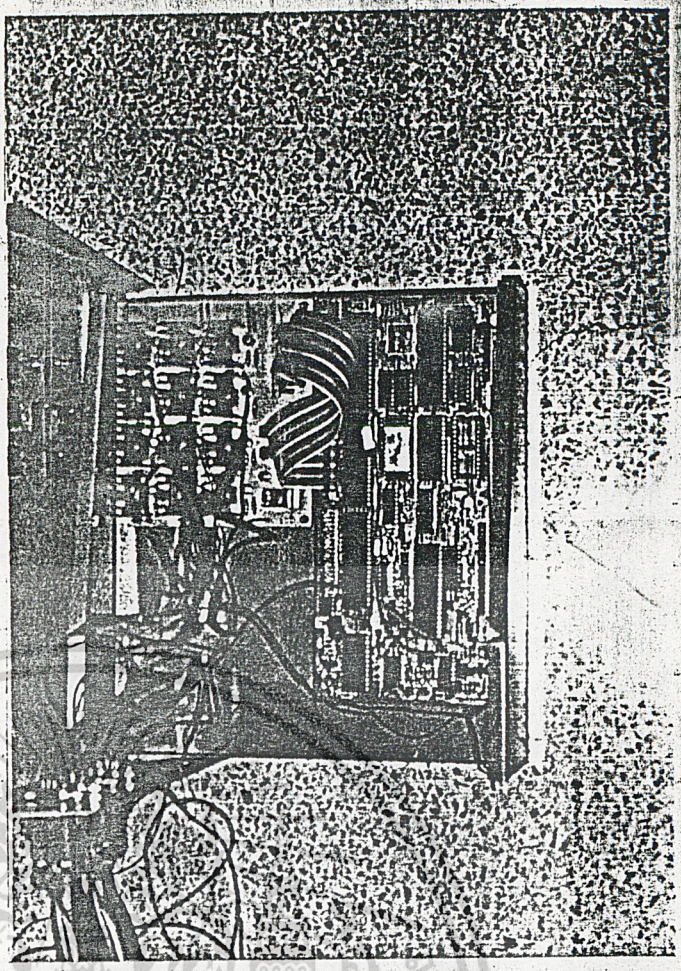
STP

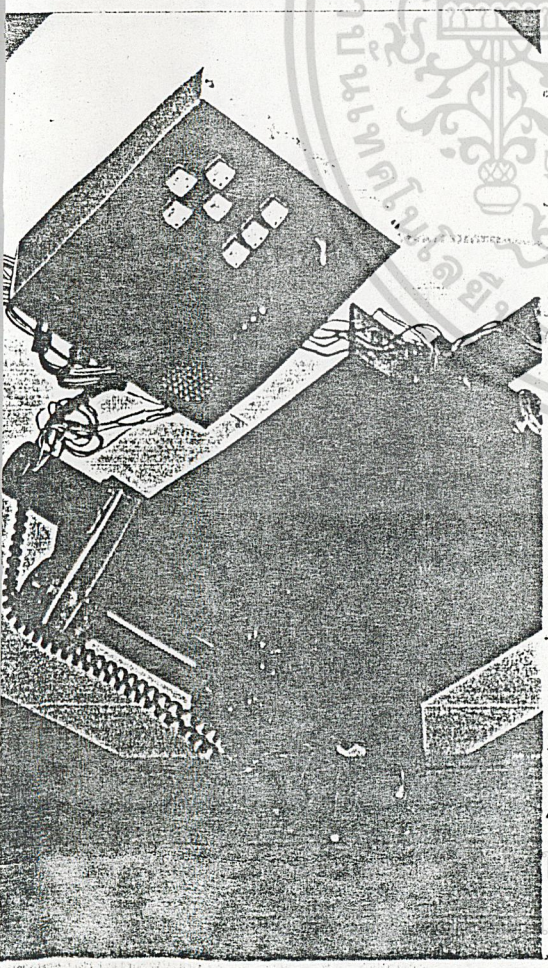
ENTER

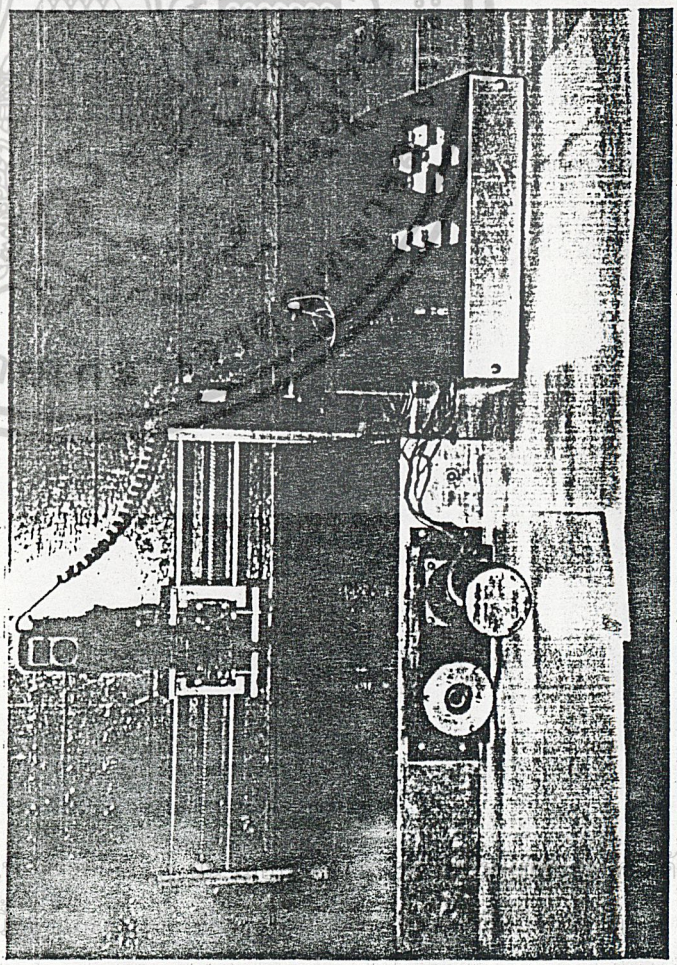












บทที่ 4
การเขียนโปรแกรมควบคุมระบบให้เครื่องเจาะอัตโนมัติ

ขั้นตอนในการเขียนโปรแกรมควบคุมระบบให้กับเครื่องเจาะอัตโนมัติ

ก่อนที่เราจะทำการเขียนโปรแกรม เพื่อไปควบคุม การทำงานให้เครื่องเจาะอัตโนมัตินั้น เราจำเป็นจะต้อง มีการกำหนดขั้นตอนการทำงาน (SEQUENCE) ของเครื่องเจาะอัตโนมัติเสียก่อน ดังนั้นเราจะต้องมีความเข้าใจเกี่ยวกับทางด้านฮาร์ดแวร์ที่ได้ออกแบบไว้ เป็นอย่างดี เพื่อที่จะได้เขียนโปรแกรมสั่งการควบคุมไปตามจุดต่างๆ ได้อย่างถูกต้อง

รายละเอียดต่างๆ เกี่ยวกับทางด้านฮาร์ดแวร์ เราได้กล่าวถึงไปแล้วในบทที่ 3 เรื่องการออกแบบและการสร้าง ต่อไปจะได้กล่าวถึงขั้นตอนการทำงานของเครื่องเจาะอัตโนมัติว่า มีขั้นตอนในการทำงาน และจะใช้งานอย่างไร เพื่อเป็นแนวทางในขั้นของการเขียนโปรแกรมควบคุมต่อไป

การทำงานของเครื่องเจาะอัตโนมัติ

การทำงานของเครื่องเจาะอัตโนมัตินี้ แบ่งการทำงานออกเป็น 2 โหมดดังนี้คือ

- โหมดการทำงานเป็นเครื่องเจาะปกติทั่วไป (MANUAL MODE)
- โหมดการทำงานแบบอัตโนมัติ (AUTOMIC MODE)

4.1โหมดการทำงานเป็นเครื่องเจาะปกติทั่วไป

ในโหมดการทำงานนี้จะทำงานในลักษณะเป็นเครื่องเจาะทั่วๆไป คือผู้ใช้จะเป็นผู้ทำการเจาะแผ่นวงจรพิมพ์ด้วยตัวเอง โดยอาศัยคีย์บนหน้าบัทย์ควบคุมเป็นตัวสั่งงาน ซึ่งคีย์ต่างๆ ที่ใช้ในโหมดนี้มีดังนี้

ใช้ในการเลื่อนตัวสว่านไปตามแกนวายขึ้นข้างบน

ใช้ในการเลื่อนตัวสว่านไปตามแกนวายลงข้างล่าง

ใช้ในการเลื่อนตัวสว่านไปตามแกนเอ็กซ์ไปทางซ้าย

ใช้ในการเลื่อนตัวสว่านไปตามแกนเอ็กซ์ไปทางขวา

START ใช้ในการสั่งให้ตัวสว่านทำงานและเลื่อนสว่านลงเจาะชิ้นงานตามแนวแกนแซด

อนึ่งสวิทช์โหมด (SWITCH MODE) จะต้องอยู่ในตำแหน่ง "MANUAL MODE" และไฟแสดนด์บาย (STANDBY) สว่างอยู่

ดังนั้น เมื่อเราต้องการจะ เจาะชิ้นงาน (แผ่นวงจรพิมพ์) เราก็เพียงแต่นำชิ้นงานนั้นไปวางยึดไว้กับตัวแท่นของเครื่องเจาะอัตโนมัตินี้ จากนั้นเมื่อต้องการเจาะตำแหน่งที่ใด ก็ใช้คีย์ลูกศรในการเลื่อนตัวสว่านไปยังตำแหน่งนั้น และเมื่อเลื่อนไปจนถึงตำแหน่งที่ต้องการจะ เจาะ ก็เพียงแต่กดคีย์ ENTER ชุดควบคุมก็จะสั่งให้ตัวสว่านทำงาน (หมุน) แล้วเลื่อนตัวสว่าน ลงเจาะบนชิ้นงานจนกระทั่งทะลุเป็นรู จากนั้นชุดควบคุมก็จะสั่งให้ตัวสว่านเลื่อนขึ้นกลับตำแหน่งบน (ที่เดิม) เพื่อรอการสั่งงานต่อไป เป็นเช่นนี้ไปเรื่อยๆ

ในกรณีที่เราเลื่อนตัวสว่านไปสุดแกนด้านใดด้านหนึ่ง จะมีตัวตรวจจับทางแสง (OPTICAL LIMIT SWITCH) คอยตรวจจับว่าขณะนี้ ตัวสว่านได้ถูกเลื่อนมาจนสุดแกนแล้ว ก็จะมีการ อินเตอร์รัพท์ (INTERRUPT) เกิดขึ้น ซึ่งชุดควบคุมจะตอบสนองการอินเตอร์รัพท์ โดยสั่งให้สเต็ปปิ้งมอเตอร์หยุดการหมุนต่อไปข้างหน้า ทั้งนี้เพื่อเป็นการป้องกันการเกิดความเสียหายแก่ ชุดแท่นเครื่องเจาะ

4.2โหมดการทำงานแบบอัตโนมัติ

ในโหมดนี้จะต้องผลักสวิทช์โหมดไปยังตำแหน่ง "AUTO MODE" (ออโตเมติกโหมด) ซึ่งในโหมดนี้มีหลักการทำงานคือ เมื่อเราต้องการจะเจาะแผ่นวงจรพิมพ์จำนวนมากๆ ที่มีตำแหน่งรูที่ต้องการเจาะอยู่ในตำแหน่งเหมือนกันหมดทุกแผ่น ขั้นแรกเราต้องทำการสอนเครื่องเจาะให้จำ รูต่างๆ บนแผ่นวงจรพิมพ์ก่อน จากนั้นในแผ่นต่อๆไป เราก็สามารถสั่งให้เครื่องเจาะทำงานเอง โดยอัตโนมัติ

คีย์ที่ใช้ในโหมดการทำงานแบบอัตโนมัติ มีดังนี้

ใช้ในการเลื่อนตัวสว่านไปตามแกนวายขึ้นข้างบน

ใช้ในการเลื่อนตัวสว่านไปตามแกนวายลงข้างล่าง

ใช้ในการเลื่อนตัวสว่านไปตามแกนเอ็กซ์ไปทางซ้าย

ใช้ในการเลื่อนตัวสว่านไปตามแกนเอ็กซ์ไปทางขวา

ENTER ใช้ในการสั่งให้เครื่องเจาะจำตำแหน่งที่ตัวสว่านอยู่ในขณะนั้น หรือใช้ในกรณีที่ต้องการจะลบข้อมูล (หมายถึง ตำแหน่งที่ได้บันทึกไว้) ออกจากหน่วยความจำ เพื่อทำการโปรแกรมการเจาะตำแหน่งใหม่

START ใช้ในการสั่งให้เครื่องเจาะเริ่มทำการเจาะโดยอัตโนมัติตามตำแหน่งต่างๆที่ได้ทำการบันทึกไว้ หรือใช้ในการยืนยันการลบข้อมูล หรือใช้ในการโหลดข้อมูลจากพีซี ลงสู่เครื่องเจาะ

STOP ใช้ในการสั่งให้เครื่องซึ่งกำลังทำการเจาะอยู่ หยุด ทำการเจาะชั่วคราว แล้วรอการกดคีย์ START เพื่อทำการเจาะต่อไป หรือใช้ในการยกเลิกการลบข้อมูล หรือใช้ในการเซฟข้อมูลจากเครื่องเจาะไปยังพีซี

การทำงาน เมื่อเราจะทำการโปรแกรมหรือสอนเครื่องเจาะให้จำตำแหน่งต่างๆ ในการเจาะ (ไพโปรแกรมมิ่ง (PROGRAMMING) ต้องสว่าง) ขั้นแรกเราก็นำแผ่นวงจรพิมพ์ดันแบบไปวางยึดบนแท่นเครื่องเจาะ (ตำแหน่งการวางแผ่นวงจรพิมพ์บนแท่นเครื่องเจาะนี้ จะเปลี่ยนแปลงไม่ได้ จนกว่าจะมีการเปลี่ยนแปลงข้อมูลของตำแหน่งในการเจาะใหม่) จากนั้นเมื่อเราต้องการ

ให้เครื่องจำตำแหน่งใด ก็ทำการเลื่อนตัวสว่านไปตามแกนเอ็กซ์ และแกนวาย โดยใช้คีย์ลูกศรเป็นตัวเลื่อน เมื่อถึงตำแหน่งที่ต้องการจะให้เครื่องจำ ก็กดคีย์ ENTER 1 ครั้ง พร้อมทั้งจะได้ยินเสียงดังบี๊บ ออกมาจากเครื่อง 1 ครั้ง เพื่อบอกให้รู้ว่าขณะนี้เครื่องได้จำตำแหน่งนั้นเก็บเข้าไปในหน่วยความจำแล้ว แต่ถ้าเรากดคีย์ ENTER 2 ครั้งติดต่อกัน จะหมายความว่า เราต้องการจะลบข้อมูลของตำแหน่งทั้งหมดที่ได้สอนเครื่องไว้ ออกจากหน่วยความจำ เมื่อถึงขั้นตอนนี้เครื่องจะส่งเสียงดังติดต่อกันออกมาทางลำโพง พร้อมทั้งจะหยุดรอรับคีย์ START ซึ่งหมายถึงการยืนยันการลบข้อมูลเดิมทิ้งไป ณ จุดนี้เครื่องจะหยุดการทำงานชั่วคราว (ไฟ สต๊อป(STOP) และ โนดาต้า(NO DATA) จะสว่าง) เพื่อให้ทำการโหลดข้อมูลจากแผ่นจานแม่เหล็กที่มีข้อมูลเก็บไว้ โดยใช้คำสั่งบนพีซี และเมื่อทำการโหลดข้อมูลเสร็จเรียบร้อยแล้วก็ใช้คำสั่ง >CONT เพื่อให้เครื่องเจาะทำงานต่อไป แต่ถ้าเราต้องการยกเลิกคำสั่งในการลบข้อมูล ก็จะต้องกดคีย์ STOP เพื่อบอกให้เครื่องยกเลิกคำสั่งการลบข้อมูล และ ณ จุดนี้เครื่องจะหยุดการทำงานชั่วคราว (ไฟ สต๊อป(STOP) จะสว่าง) เพื่อให้ทำการเก็บข้อมูลลงแผ่นจานแม่เหล็ก โดยใช้คำสั่งบนพีซี แล้วรอคำสั่ง >CONT เพื่อสั่งให้เครื่องเจาะทำงานต่อไป

เมื่อเราทำการสอนเครื่องให้จำตำแหน่งต่างๆที่จะเจาะจนหมดแล้ว ก็ให้กดคีย์ START เครื่องก็จะจำตำแหน่งต่างๆไว้ทั้งหมด (ตอนนี้ไฟ สต๊อป (STOP) จะสว่าง) และรอการกดคีย์ START อีกครั้งหนึ่งเพื่อทำการเจาะตามตำแหน่งต่างๆ ที่ได้สอนไว้โดยอัตโนมัติต่อไป

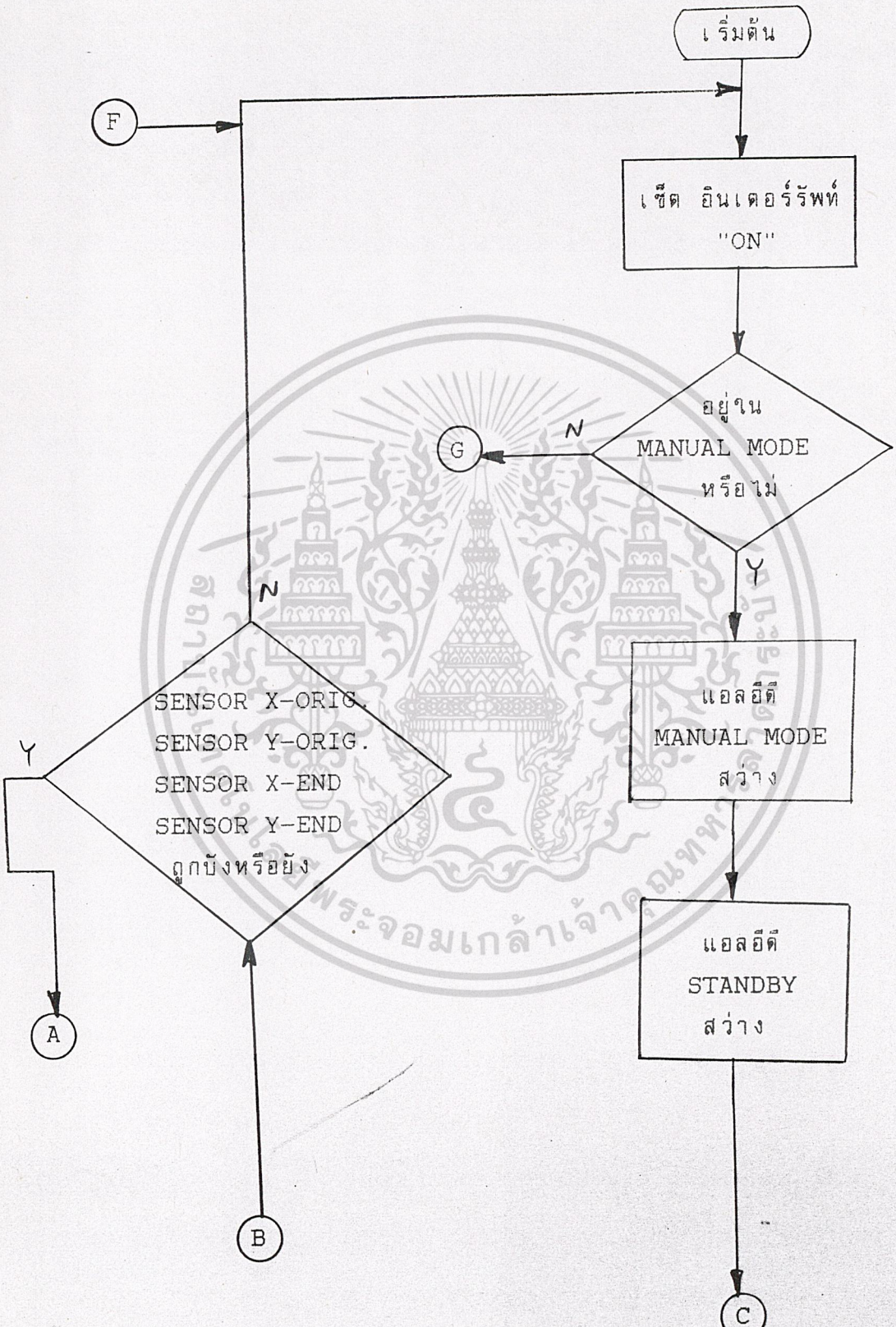
ในกรณีที่ไม่มีข้อมูลของตำแหน่งถูกจำไว้ในเครื่องเลย เมื่อมีการกดคีย์ START ให้เครื่องทำงานโดยอัตโนมัติ เครื่องจะเลื่อนตัวสว่านออกจากจุดออริจิน (ORIGIN) เล็กน้อย พร้อมกับไฟ รันนิ่ง(RUNNING) จะสว่างแล้วเลื่อนตัวสว่านกลับเข้าไปยังจุดออริจินเช่นเดิม ซึ่งนั่นก็เป็นการบอกให้เรารู้ว่าขณะนั้น เครื่องไม่ได้จำข้อมูลอยู่เลย ให้เราทำการสอนเครื่องจำตำแหน่งหรือทำการโหลดข้อมูลเข้าเครื่องเจาะเสียก่อน

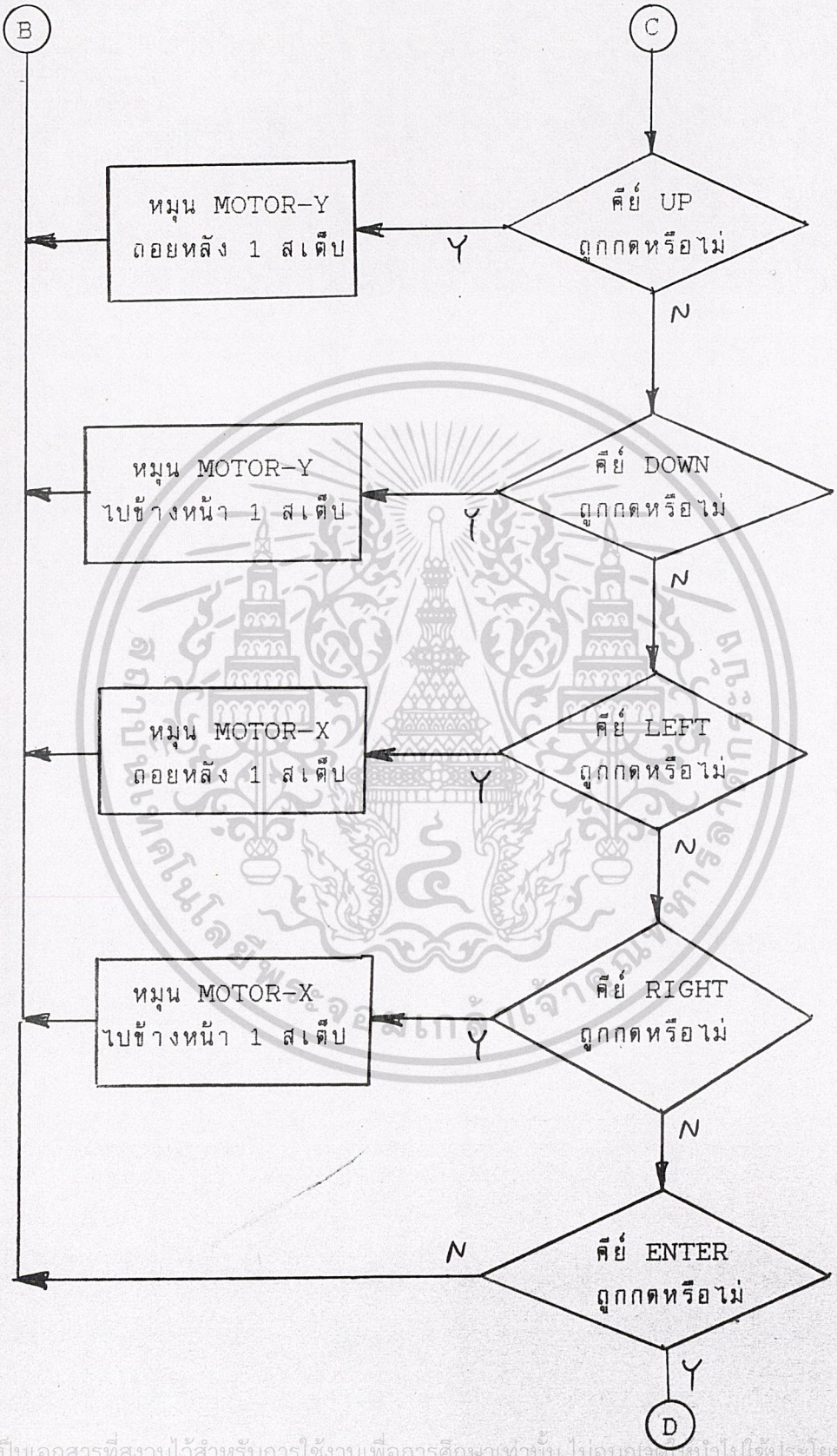
ในขณะที่เครื่องกำลังทำการเจาะโดยอัตโนมัติอยู่นั้น ไฟ รันนิ่ง (RUNNING) จะสว่าง และเมื่อมีการกดคีย์ STOP ไฟ สต๊อป(STOP) ก็จะสว่าง เครื่องจะหยุดการทำงานไว้ชั่วคราว รอจนกระทั่งมีการกดคีย์ START เครื่องจึงจะทำงานต่อไป

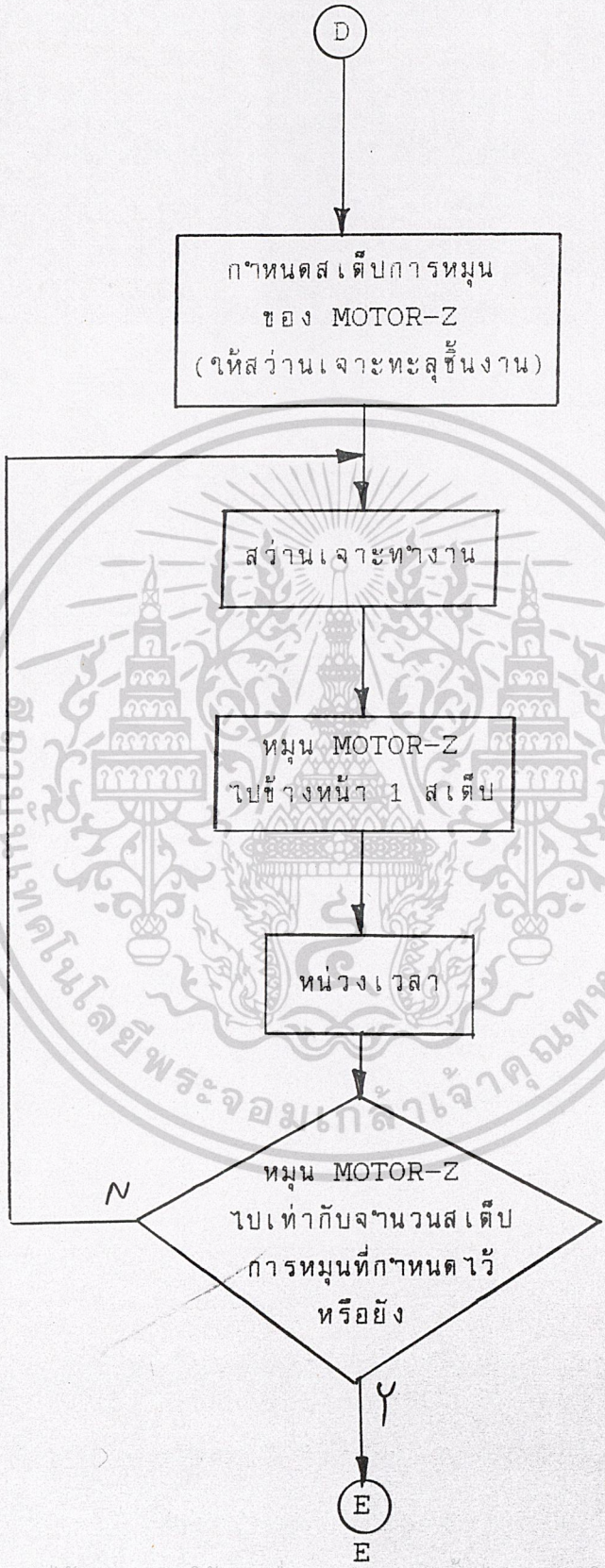
จากที่กล่าวมาแล้วจะเห็นว่า ตราบใดที่ยังมีการจ่ายไฟให้แก่เครื่องเจาะอยู่เครื่องเจาะก็จะยังคงทำงานอยู่ใน 2 โหมดที่กล่าวมาแล้ว จนกว่าจะปิดการจ่ายไฟให้เครื่อง เครื่องเจาะจึงจะหยุดทำงาน

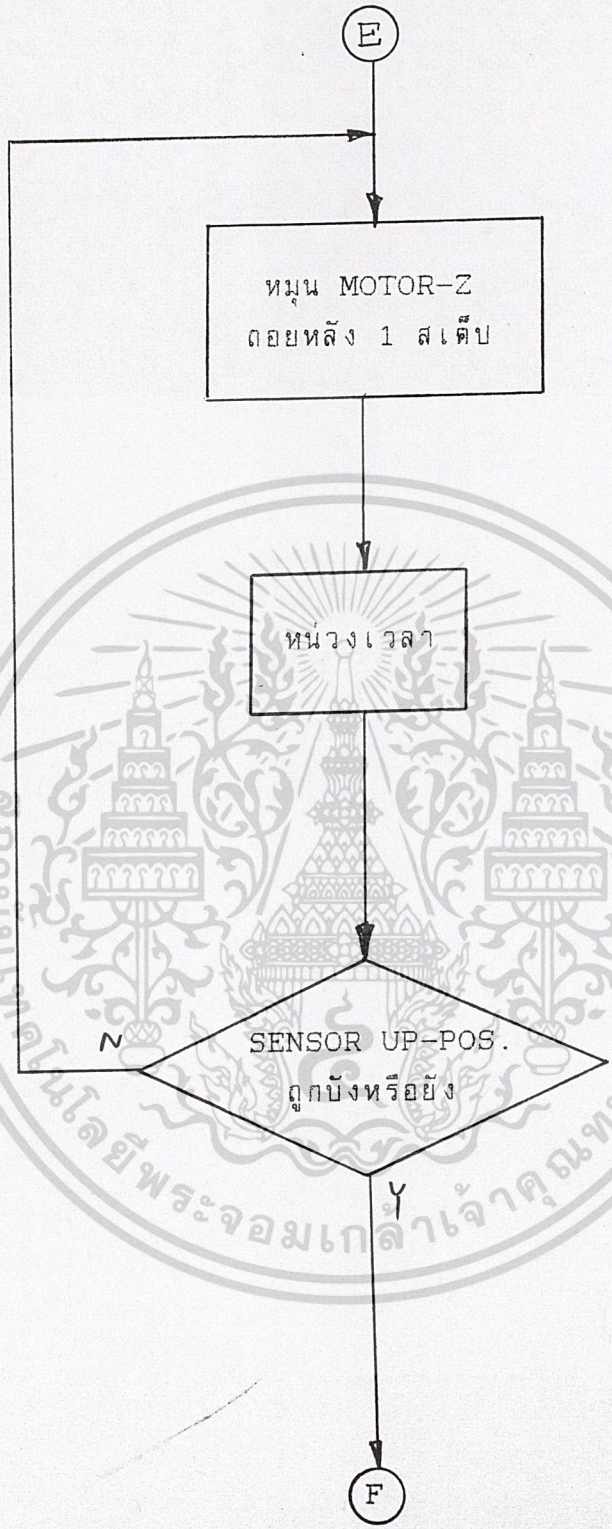
อธิบายการทำงานของโปรแกรมควบคุมระบบของเครื่องเจาะอัตโนมัติ

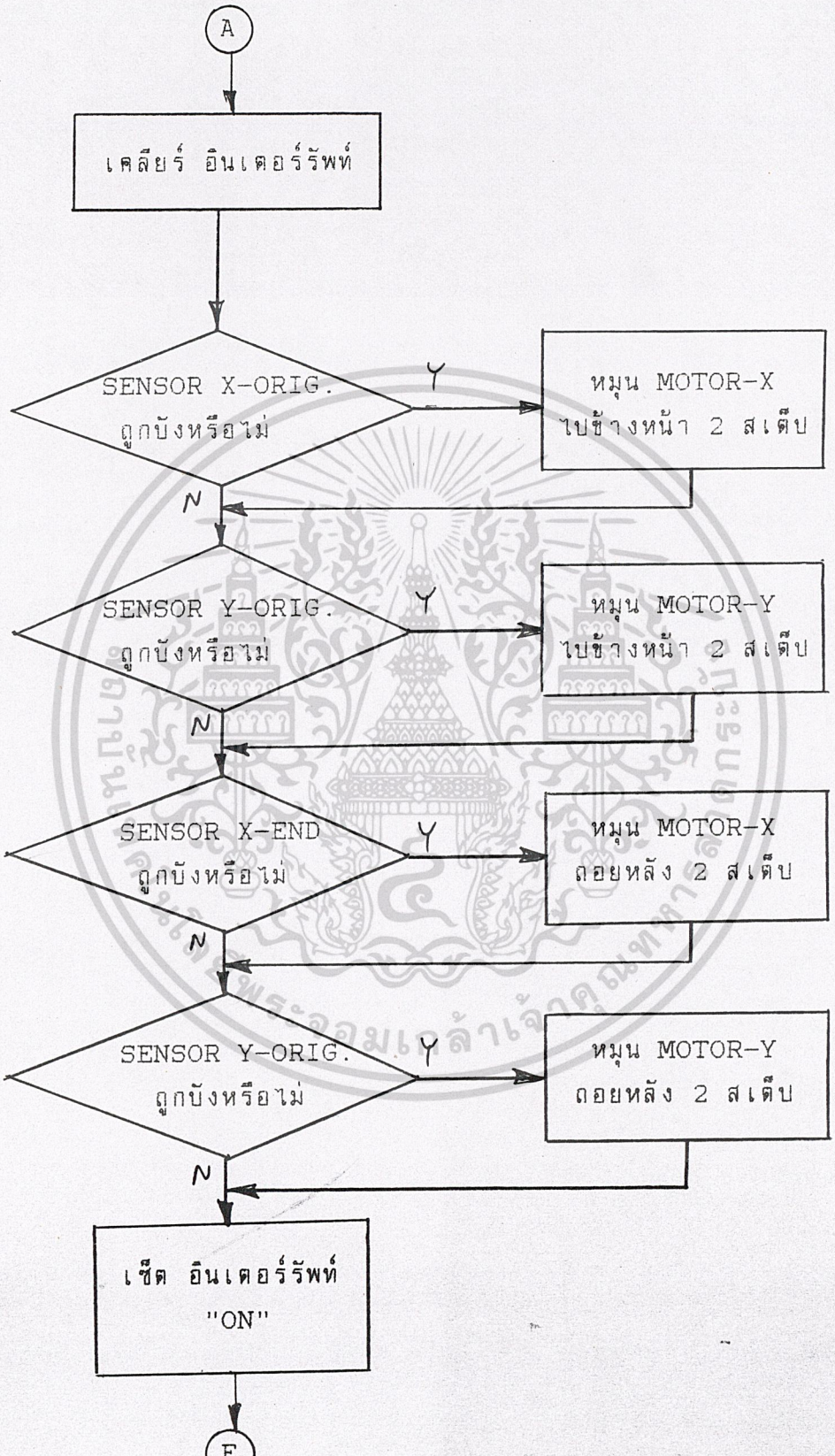
ลำดับการทำงานของโปรแกรมจะขออธิบาย ในผังการทำงาน (FLOWCHART) ต่อไปนี้

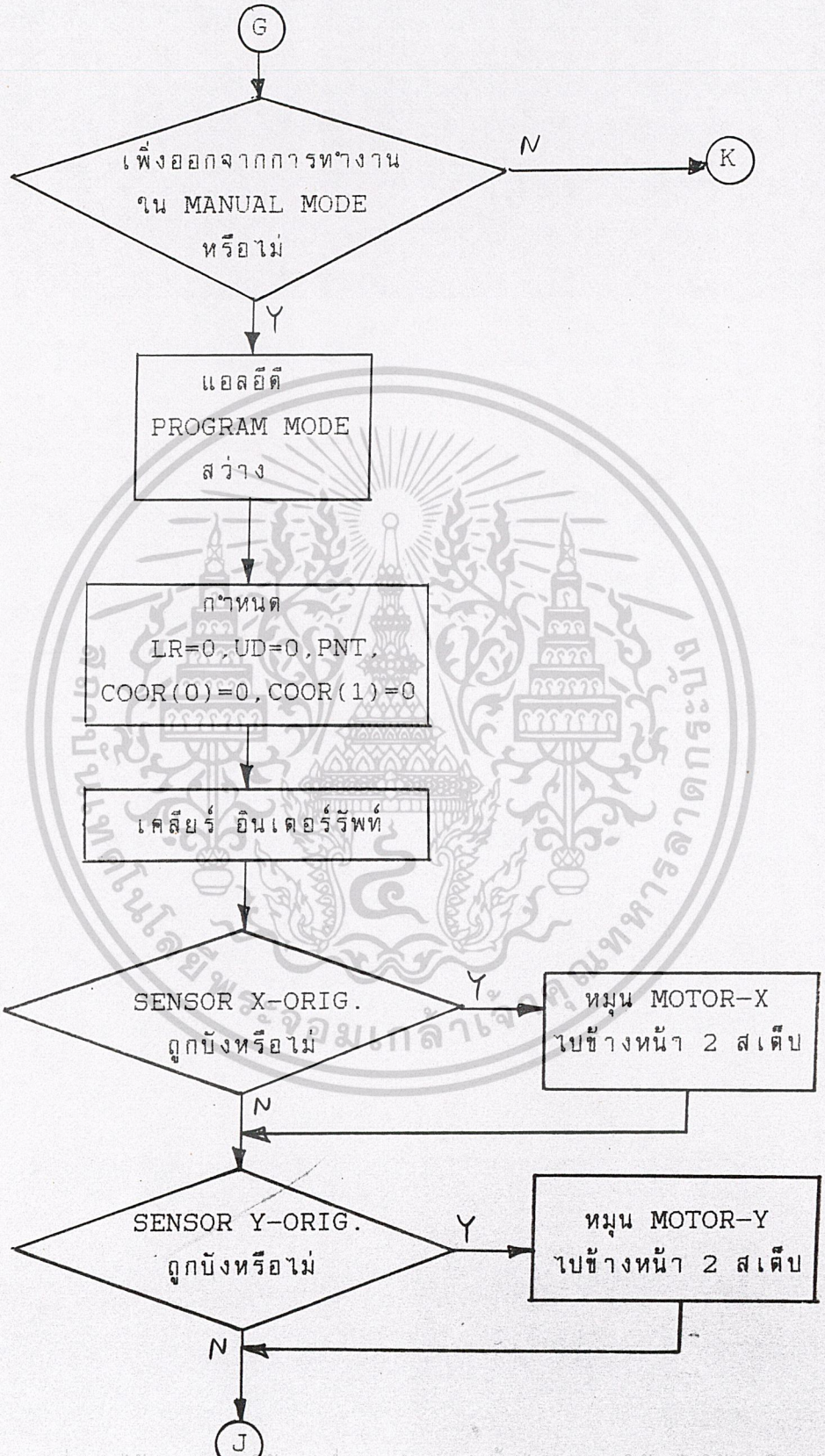


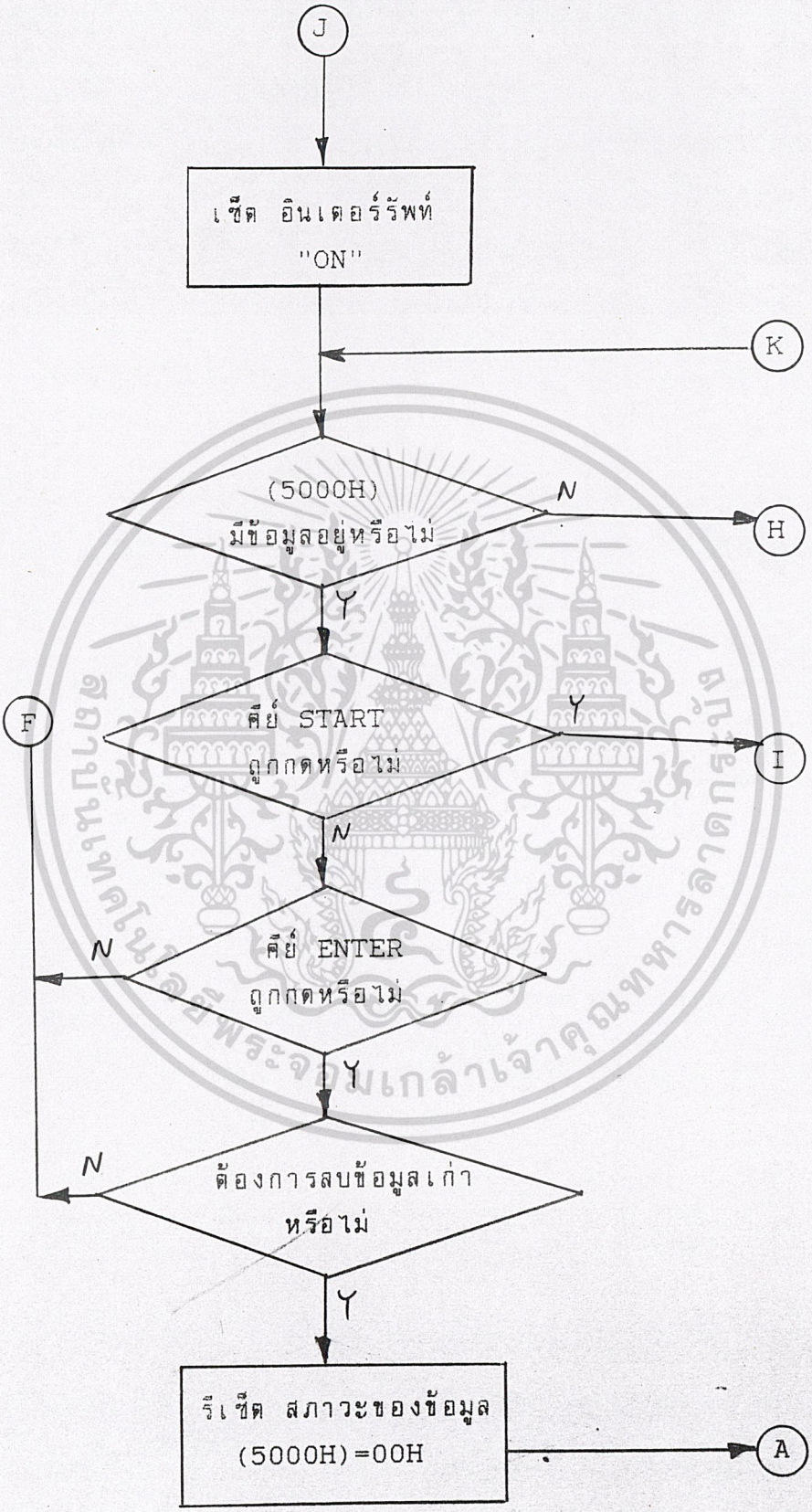


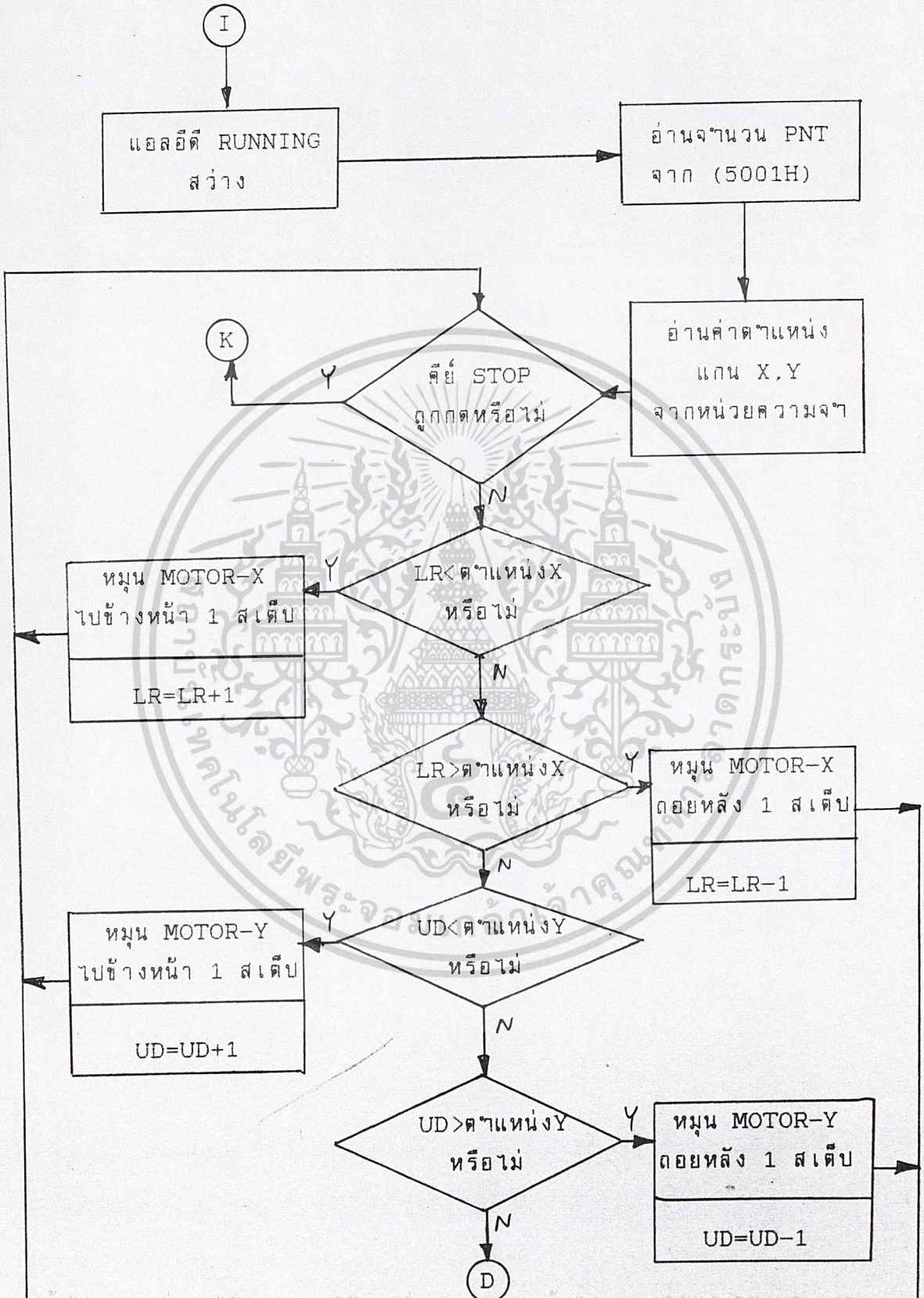


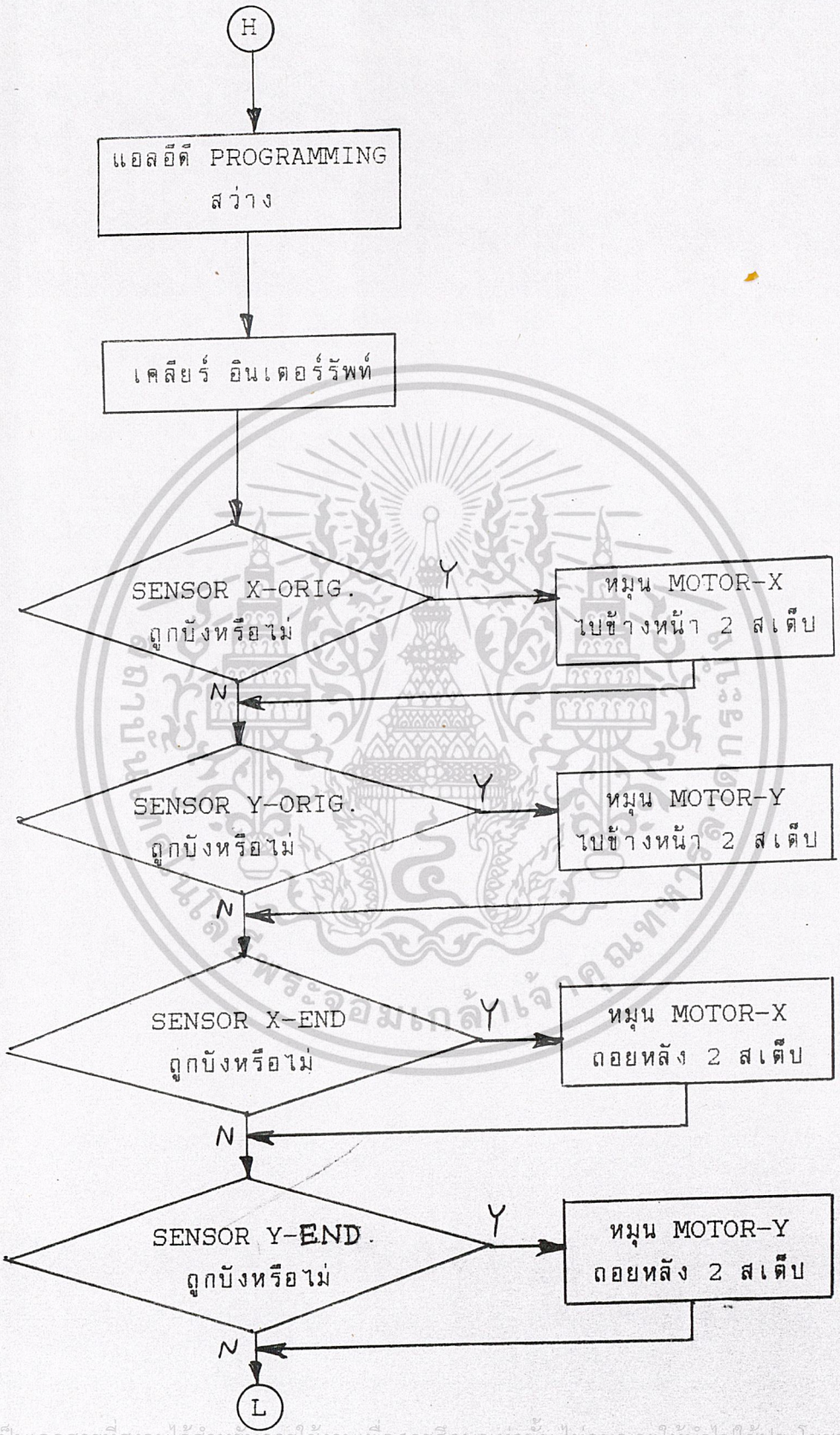


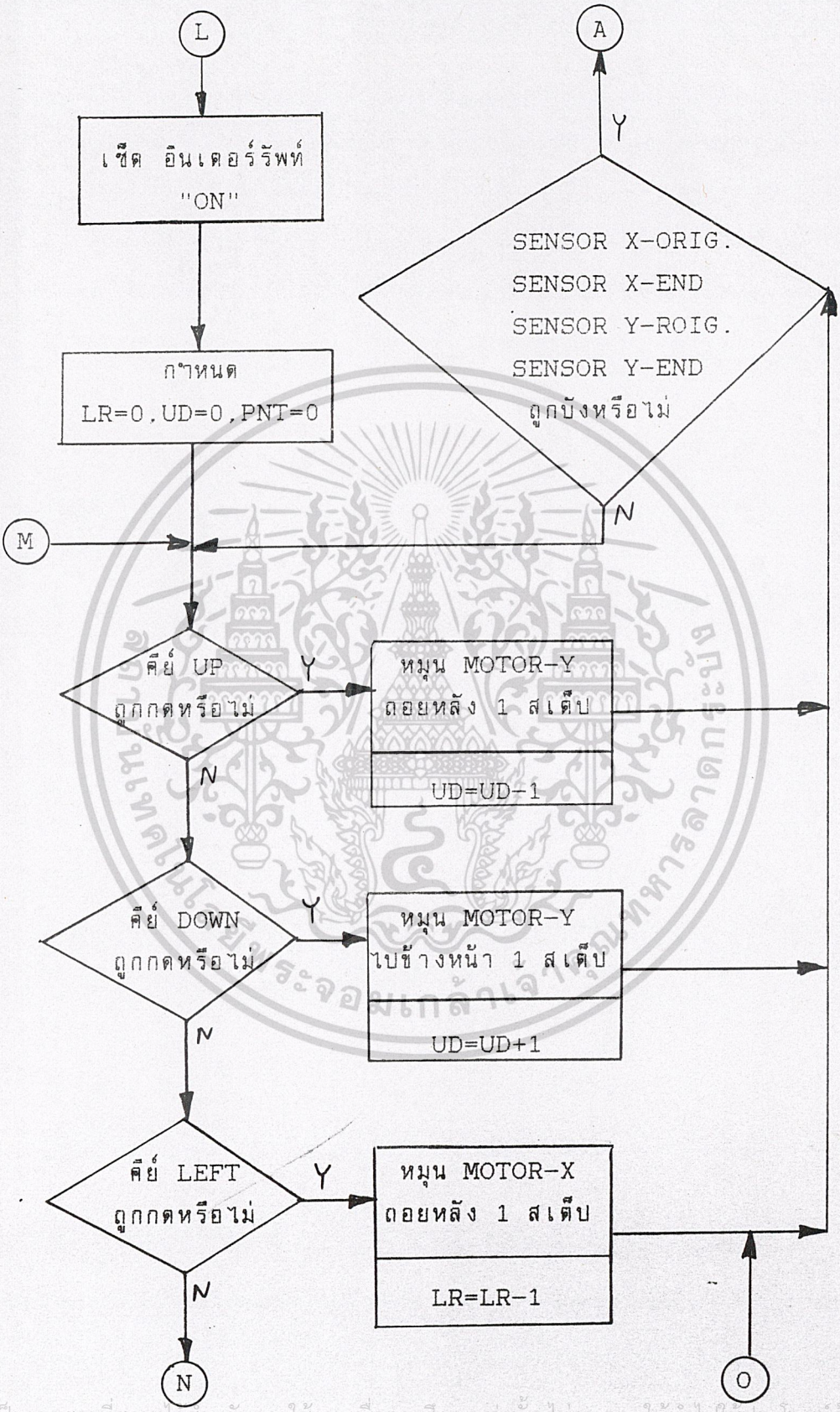


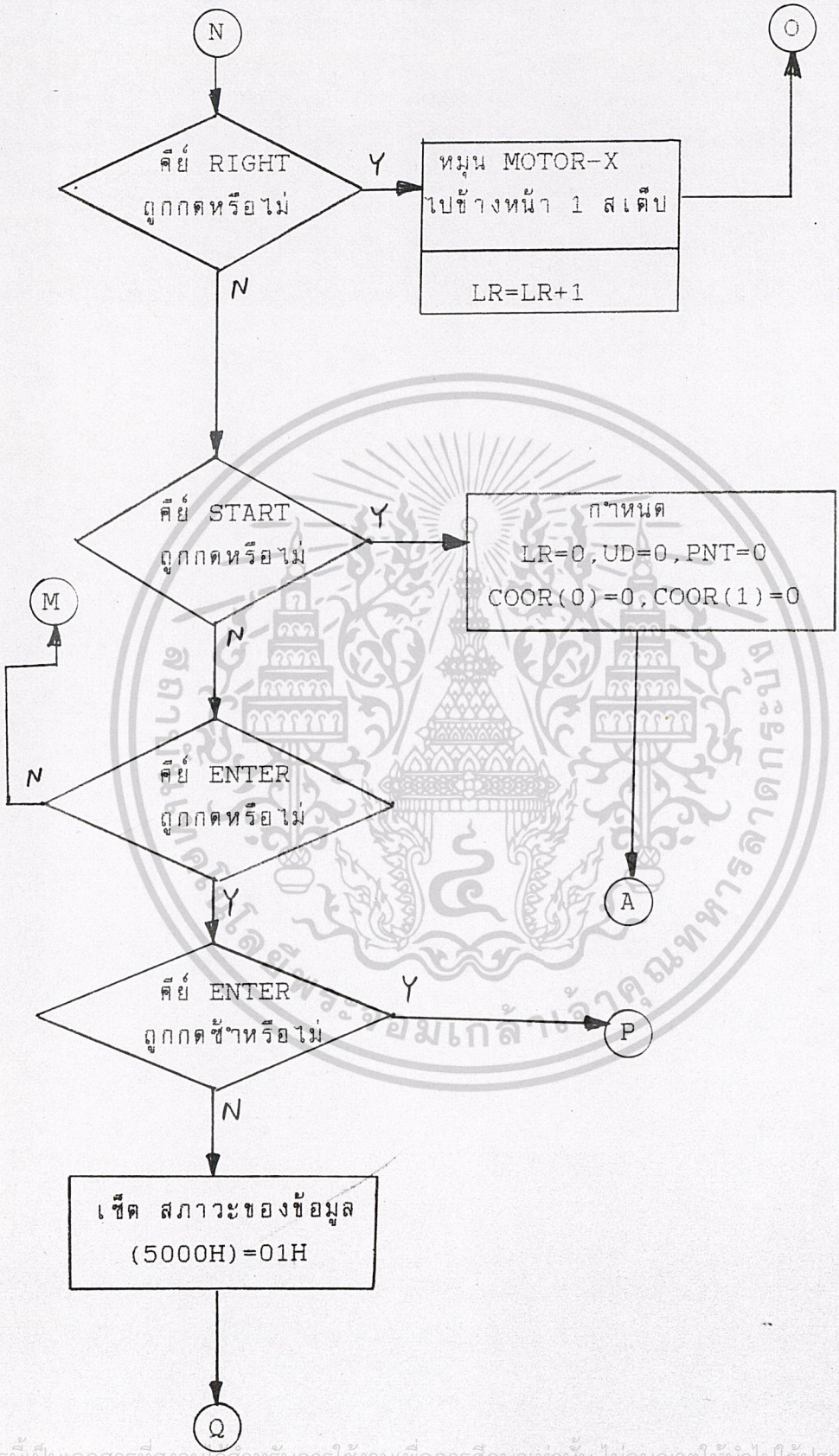


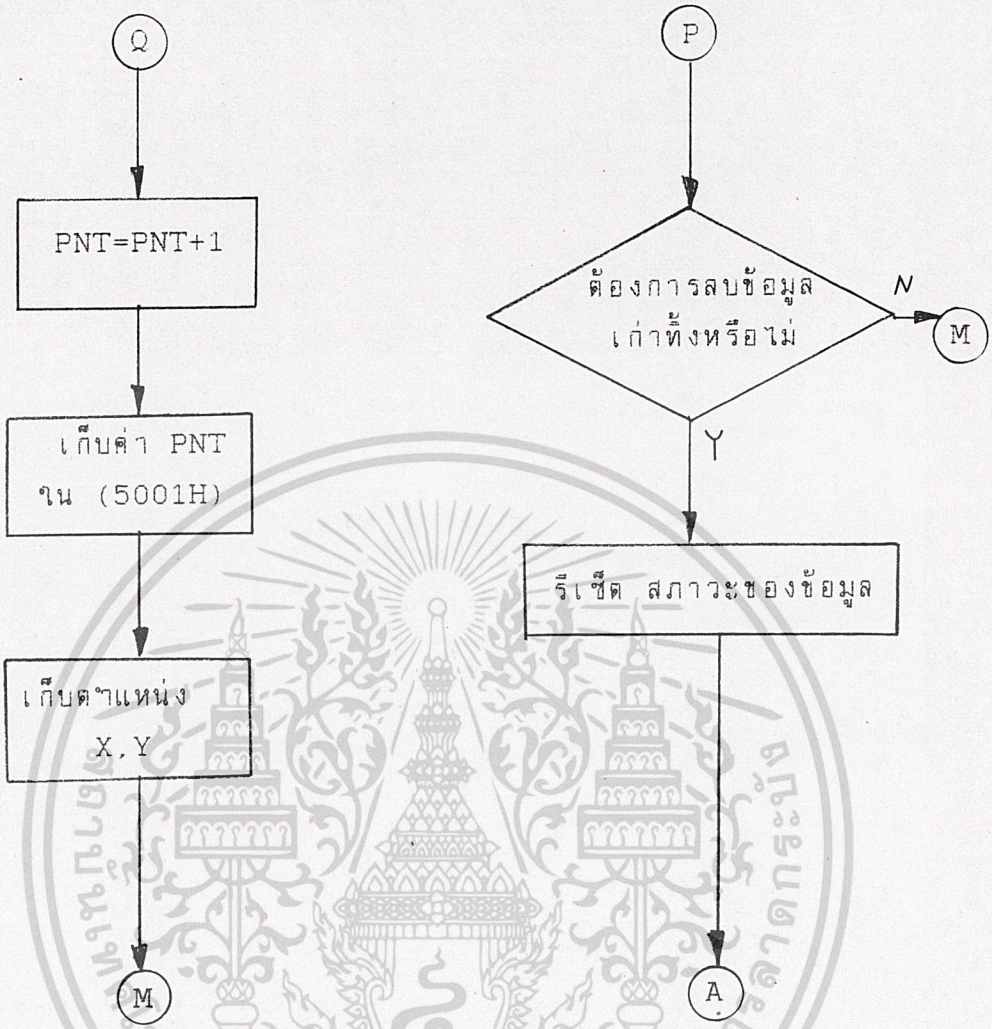












```
**********************************************
*      AUTOMATIC DRILLER PROGRAM LISTING      *
*                      BY                     *
*           MR.SURACHAI MEKSRIARUN            *
*        INDUSTRIAL COMPUTER TECHNOLOGY       *
*     KING MONGKUT INSTITUTE OF TECHNOLOGY    *
*              LADKRABARNG   1991             *
**********************************************
```

```
10      INIT
11     XBY(0FF03H)=82H
25      DIM COOR(2)
30      DIM A(4)
40      DIM B(4)
45      DIM C(4)
50     A(0)=03H
60     A(1)=06H
70     A(2)=0CH
80     A(3)=09H
90     B(0)=30H
100    B(1)=60H
110    B(2)=0C0H
120    B(3)=90H
122    C(0)=03H
124    C(1)=06H
126    C(2)=0CH
128    C(3)=09H
130    UD=0
131    LR=0
133    ADDR=0
134    PB=0
135    PNT=0
136    I=0
```

```
137   DWN=0
138   LE=0
140   X=0
141   Y=0
142   Z=0
143   XBY(0FF02H)=03H
144    IF 00H<>XBY(5501H) THEN XBY(5500H)=01H
145    GOSUB 900
146    GOSUB 1500
147    ONEX1 2500
148   MODE=0
149    REM ***** manual mode *****
150    IF 0FFH<>XBY(0FF22H).AND.MODE=0 THEN 256
155    DO
156   XBY(0FF20H)=06H
160   MODE=1
162   M=A(X).OR.B(Y)
165   XBY(0FF00H)=M
170    IF 0F7H=XBY(0FF22H) THEN  GOSUB 500
180    IF 0FBH=XBY(0FF22H) THEN  GOSUB 510
190    IF 0FDH=XBY(0FF22H) THEN  GOSUB 520
200    IF 0FEH=XBY(0FF22H) THEN  GOSUB 530
210    IF 0EFH=XBY(0FF22H) THEN  GOSUB 540
250    UNTIL 07FH=XBY(0FH22H)
255    REM ***** program mode *****
256   XBY(0FF20H)=08H
261    IF MODE=1 THEN  GOSUB 900
280   M=A(X).OR.B(Y)
285   XBY(0FF00H)=M
290    IF 07DH=XBY(0FF22H) THEN  GOSUB 1000
295    IF 07EH=XBY(0FF22H) THEN  GOSUB 1010
300    IF 077H=XBY(0FF22H) THEN  GOSUB 1020
305    IF 07BH=XBY(0FF22H) THEN  GOSUB 1030
```

```
310     IF 06FH=XBY(0FF22H) THEN  GOSUB 1040
315     IF 05FH=XBY(0FF22H) THEN  GOSUB 1200
320     GOTO 148
330     CLEAR S
331    XBY(0FF20H)=21H
334     GOSUB 900
335     DO
337    XBY(0FF20H)=10H
340     IF 05FH=XBY(0FF22H) THEN  GOSUB 1080
345     UNTIL 0FFH=XBY(0FF22H)
350     GOTO 148
400     END
499     REM ***** subroutine *****
500     IF X=3 THEN X=-1
502    X=X+1
504     RETURN
510     IF X=0 THEN X=4
512    X=X-1
514     RETURN
520     IF Y=3 THEN Y=-1
522    Y=Y+1
524     RETURN
530     IF Y=0 THEN Y=4
532    Y=Y-1
534     RETURN
540    LE=50
542     GOSUB 1500
543     REM **** move down ****
545     DO
546     IF Z=3 THEN Z=-1
547    Z=Z+1
550    DWN=C(Z).OR.80H
560    XBY(0FF02H)=DWN
```

```
570     FOR I=0 TO 10
580     NEXT I
600    LE=LE-1
610     UNTIL LE=0
615     REM **** move up ****
620     FOR I=0 TO 20
625     IF Z=0 THEN Z=4
627    Z=Z-1
630    DWN=C(Z).OR.80H
640    XBY(0FF02H)=DWN
650     NEXT I
660     GOSUB 1500
680     RETURN
700     DO
701     IF X=0 THEN X=4
702    X=X-1
703     GOSUB 800
704    PB=XBY(0FF01H).AND.01H
706     UNTIL 01H=PB
707     RETURN
710     DO
711     IF Y=0 THEN Y=4
712    Y=Y-1
715     GOSUB 800
716    PB=XBY(0FF01H).AND.02H
717     UNTIL 02H=PB
720     RETURN
800    M=A(X).OR.B(Y)
805    XBY(0FF00H)=M
810     RETURN
899     REM ***** move driller to origin point *****
900     CLEAR I
901    XBY(0FF20H)=02H
```

```
905   PB=XBY(0FF01H).AND.01H ←
906    IF 01H<>PB THEN  GOSUB 700
910   PB=XBY(0FF01H).AND.02H
911    IF 02H<>PB THEN  GOSUB 710
912    DO
914    GOSUB 500
916    GOSUB 800
918    UNTIL 72H=XBY(0FF01H).OR.0F2H=XBY(0FF01H)
920    DO
922    GOSUB 520
924    GOSUB 800
926    UNTIL 70H=XBY(0FF01H).OR.0F0H=XBY(0FF01H)
928    ONEX1 2500
930   LR=0
932   UD=0
934   COOR(0)=0
936   COOR(1)=0
938    RETURN
1000   IF Y=3 THEN Y=-1
1002  Y=Y+1
1004  UD=UD+1
1006   RETURN
1010   IF Y=0 THEN Y=4
1012  Y=Y-1
1014  UD=UD-1
1016   RETURN
1020   IF X=3 THEN X=-1
1022  X=X+1
1024  LR=LR+1
1026   RETURN
1030   IF X=0 THEN X=4
1032  X=X-1
1034  LR=LR-1
```

```
1036   RETURN
1039   REM ***** save coordinate value *****
1040   IF LR=COOR(0).AND.UD=COOR(1) THEN 1300
1041  ADDR=PNT*4
1042  XBY(5501H)=PNT+1
1045  COOR(0)=LR
1050  COOR(1)=UD
1055  XBY(5501H+ADDR+1)=INT(LR/100)
1060  XBY(5501H+ADDR+2)=LR-INT(LR/100)*100
1065  XBY(5501H+ADDR+3)=INT(UD/100)
1070  XBY(5501H+ADDR+4)=UD-INT(UD/100)*100
1071  PNT=PNT+1
1072   IF 00H<>XBY(5501H) THEN XBY(5500H)=01H
1074   FOR I=0 TO 10
1075  XBY(0FF40H)=0FFH
1076  XBY(0FF40H)=00H
1077   NEXT I
1078   RETURN
1079   REM ***** read coordinate value *****
1080  PNT=0
1082  ADDR=0
1085   DO
1087  ADDR=PNT*4
1090  COOR(0)=XBY(5501H+ADDR+1)*100+XBY(5501H+ADDR+2)
1095  COOR(1)=XBY(5501H+ADDR+3)*100+XBY(5501H+ADDR+4)
1100  PNT=PNT+1
1110   GOSUB 1800
1115   UNTIL 0FFH=XBY(0FF22H)
1120   RETURN
1200   IF 01H=XBY(5500H) THEN 330
1201  XBY(0FF20H)=03H
1202   FOR I=0 TO 25
1203  XBY(0FF40H)=0FFH
```

```
1204   GOSUB 500
1205  XBY(0FF40H)=00H
1206   GOSUB 520
1207  M=A(X).OR.B(Y)
1208  XBY(0FF00H)=M
1209   NEXT I
1210   GOSUB 900
1211   STOP
1212   RETURN
1300   DO
1301  XBY(0FF40H)=0FFH
1302   IF 05FH=XBY(0FF22H) THEN 1400
1304   IF 03FH=XBY(0FF22H) THEN 1550
1305  XBY(0FF40H)=00H
1306   UNTIL 0FFH=XBY(0FF22H)
1308   GOTO 148
1400  XBY(5500H)=00H
1401  XBY(5501H)=00H
1410  ADDR=0
1415  PNT=0
1450   GOTO 148
1500   DO
1503   IF Z=0 THEN Z=4
1505  Z=Z-1
1510  DWN=C(Z)
1515  XBY(0FF02H)=DWN
1530   UNTIL 0F0H=XBY(0FF01H)
1531   DO
1532   IF Z=3 THEN Z=-1
1533  Z=Z+1
1535  DWN=C(Z)
1537  XBY(0FF02H)=DWN
1539   UNTIL 70H=XBY(0FF01H)
```

```
1540   RETURN
1550   REM ************ save data to disk ***************
1560   DO
1570   IF 06FH=XBY(0FF22H) THEN 1650
1575   IF 05FH=XBY(0FF22H) THEN 148
1580   UNTIL 0FFH=XBY(0FF22H)
1650   STOP
1655   GOTO 148
1799   REM ***** move driller to drill *****
1800  XBY(0FF20H)=02H
1801   DO
1805   IF LR<COOR(0) THEN  GOSUB 1020
1810   IF UD<COOR(1) THEN  GOSUB 1000
1815   IF LR>COOR(0) THEN  GOSUB 1030
1820   IF UD>COOR(1) THEN  GOSUB 1010
1821  M=A(X).OR.B(Y)
1822  XBY(0FF00H)=M
1825   UNTIL LR=COOR(0).AND.UD=COOR(1).OR.0FFH=XBY(0FF22H)
       .OR.03FH=XBY(0FF22H)
1826   IF 03FH=XBY(0FF22H) THEN 2000
1830   IF 0FFH=XBY(0FF22H) THEN 148
1835   GOSUB 540
1840   IF PNT=XBY(5501H) THEN 330
1845   RETURN
2000  XBY(0FF20H)=10H
2001   DO
2005   IF 05FH=XBY(0FF22H) THEN 1800
2010   UNTIL 0FFH=XBY(0FF22H)
2015   GOTO 148
2499   REM ***** routine for interrupt *****
2500   CLEAR I
2503  PB=XBY(0FF01H).AND.01H
2505   IF 01H=PB THEN  GOSUB 2600
```

```
2509  PB=XBY(0FF01H).AND.02H←
2510   IF 02H=PB THEN  GOSUB 2610
2513  PB=XBY(0FF01H).AND.04H
2515   IF 04H=PB THEN  GOSUB 2620
2517  PB=XBY(0FF01H).AND.08H
2520   IF 08H=PB THEN  GOSUB 2630
2525   ONEX1 2500
2530   RETI
2600   DO
2602   GOSUB 500
2603   GOSUB 800
2604  PB=XBY(0FF01H).AND.01H
2605  LR=LR+1
2606   UNTIL 00H=PB
2607   ONEX1 2500
2608   RETURN
2610   DO
2612   GOSUB 520
2613   GOSUB 800
2614  PB=XBY(0FF01H).AND.02H
2615  UD=UD+1
2616   UNTIL 00H=PB
2617   ONEX1 2500
2618   RETURN
2620   DO
2622   GOSUB 510
2623   GOSUB 800
2624  PB=XBY(0FF01H).AND.04H
2625  LR=LR-1
2626   UNTIL 00H=PB
2627   ONEX1 2500
2628   RETURN
2630   DO
```

```
2632    GOSUB 530
2633    GOSUB 800
2634  PB=XBY(0FF01H).AND.08H
2635  UD=UD-1
2636    UNTIL 00H=PB
2637    ONEX1 2500
2638    RETURN
```

บทที่ 5
บทวิจารณ์และสรุป

จากการออกแบบและสร้างเครื่องเจาะอัตโนมัติ เมื่อประกอบชุดควบคุมและวงจรขับสเต็ปปิ้งมอเตอร์ รวมทั้งติดตั้งอินพุทเซนเซอร์ (INPUT SENSOR) ตามจุดต่างๆเรียบร้อยแล้ว ก็ทำการเขียนโปรแกรมเพื่อมาควบคุม ระบบการทำงาน ซึ่งรายละเอียดของโปรแกรมในส่วนของการจำตำแหน่งบนแท่นเจาะของเครื่องเจาะอัตโนมัตินั้น จะอาศัยการนับจำนวนสเต็ปการหมุน ของสเต็ปปิ้งมอเตอร์ เพื่อนำไปเก็บเป็นค่าของตำแหน่งต่างๆบนแท่นเจาะ ซึ่งในการเก็บจำนวนสเต็ปในการหมุนของมอเตอร์นั้น จะเก็บทั้งค่าในแกนเอ็กซ์ และแกนวายโดยการเก็บค่าละ 2 ไบท์ เริ่มจากตำแหน่ง 5002H จำนวนสเต็ปที่นับได้นี้จะเป็นจำนวนเลขฐาน 10 เพราะฉะนั้นจำนวนสเต็ปในการหมุนของแต่ละแกนจะต้องไม่เกิน 9999 ตำแหน่ง (เก็บค่าแกนละ 2 ไบท์เรียงกันไป) ดังนั้น จะเห็นว่าใน 1 ตำแหน่งของการเจาะ จะต้องอ่านค่าสเต็ปของแกนเอ็กซ์ 2 ไบท์และแกนวาย 2 ไบท์ ซึ่งหน่วยความจำที่นำค่าในแกนเอ็กซ์และแกนวายไปเก็บนี้ จะอยู่ในช่วง 5002H ถึง 6000H ซึ่งเป็นข้อจำกัดทางด้านฮาร์ดแวร์ จึงทำให้จำนวนตำแหน่งที่ต้องการให้เครื่องจำ ถูกจำกัดตามไปด้วย ดังนั้น เมื่อเราต้องการให้เครื่องจดจำตำแหน่งในการเจาะได้มากขึ้น จะต้องมีการเพิ่มหน่วยความจำให้มากขึ้นเข้าไป

ในส่วนของแท่นเจาะ (MACHANIC) ซึ่งต้องอาศัยความเที่ยงตรงแม่นยำของชิ้นส่วนต่างๆ ที่นำมาประกอบ เช่น ลีดสกรู จะต้องมีค่าผิดพลาดน้อยมาก ตลอดจนการประกอบชุดเพลากับกับแท่นตัวเลื่อนตัวสว่าน จะต้องได้ศูนย์ตรง จึงจะทำให้การเคลื่อนตัวของแท่นตัวสว่านไม่ติดขัด

สรุปท้ายนี้ ผู้จัดทำโครงงานหวังว่าวิธีการในการออกแบบและสร้างเครื่องเจาะอัตโนมัตินี้จะเป็นแนวทางในการพัฒนาเครื่องเจาะอัตโนมัติ ให้มีประสิทธิภาพสูงขึ้นต่อไป

ภาคผนวก

1.4 QUICK REFERENCE

COMMANDS:

COMMAND	FUNCTION	EXAMPLE(S)
RUN	Execute a program	RUN
CONT	CONTinue after a STOP or control-C	CONT
LIST	LIST program to the console device	LIST LIST 10-50
LIST#	LIST program to serial printer	LIST# LIST# 50
LIST@	LIST program to user driver (version 1.1 only)	LIST@ LIST@ 50
NEW	erase the program stored in RAM	NEW
NULL	set NULL count after carriage return-line feed	NULL NULL 4
RAM	evoke RAM mode, current program in READ/WRITE memory	RAM
ROM	evoke ROM mode, current program in ROM/EPROM memory	ROM ROM 3
XFER	transfer a program from ROM/EPROM to RAM	XFER
PROG	save the current program in EPROM	PROG
PROG1	save baud rate information in EPROM	PROG1
PROG2	save baud rate information in EPROM and execute program after RESET	PROG2
PROG3	save baud rate and MTOP information in EPROM (version 1.1 only)	PROG3
PROG4	save baud rate and MTOP information in EPROM and execute program after RESET (version 1.1 only)	PROG4

1.4 QUICK REFERENCE

COMMANDS:

COMMAND	FUNCTION	EXAMPLE(S)
PROG5	same as PROG4 except that external RAM is not cleared on RESET or power up if external RAM contains a 0A5H in location 5EH (version 1.1 only)	PROG5
PROG6	same as PROG6 except that external code location 4039H is CALLED after RESET (version 1.1 only)	PROG6
FPROG	save the current program in EPROM using the INTELligent algorithm	FPROG
FPROG1	save baud rate information in EPROM using the INTELligent algorithm	FPROG1
FPROG2	save baud rate information in EPROM and execute program after RESET, use INTELligent algorithm	FPROG2
FPROG3	same as PROG3, except INTELligent programming algorithm is used (version 1.1 only)	FPROG3
FPROG4	same as PROG4, except INTELligent programming algorithm is used (version 1.1 only)	FPROG4
FPROG5	same as PROG5, except INTELligent programming algorithm is used (version 1.1 only)	FPROG5
FPROG6	same as PROG6, except INTELligent programming algorithm is used (version 1.1 only)	FPROG6

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
BAUD	set baud rate for line printer port	BAUD 1200
CALL	CALL assembly language program	CALL 9000H
CLEAR	CLEAR variables, interrupts and Strings	CLEAR
CLEARS	CLEAR Stacks	CLEARS
CLEARI	CLEAR Interrupts	CLEARI
CLOCK1	enable REAL TIME CLOCK	CLOCK1
CLOCK0	disable REAL TIME CLOCK	CLOCK0
DATA	DATA to be read by READ statement	DATA 100
READ	READ data in DATA statement	READ A
RESTORE	RESTORE READ pointer	RESTORE
DIM	allocate memory for arrayed variables	DIM A(20)
DO	set up loop for WHILE or UNTIL	DO
UNTIL	test DO loop condition (loop if false)	UNTIL A=10
WHILE	test DO loop condition (loop if true)	WHILE A=B
END	terminate program execution	END
FOR-TO-{STEP}	set up FOR-NEXT loop	FOR A=1 TO 5
NEXT	test FOR-NEXT loop condition	NEXT A

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
GOSUB	execute subroutine	GOSUB 1000
RETURN	RETURN from subroutine	RETURN
GOTO	GOTO program line number	GOTO 500
ON GOTO	conditional GOTO	ON A GOTO 5, 20
ON GOSUB	conditional GOSUB	ON A GOSUB 2, 6
IF-THEN-{ELSE}	conditional test	IF A<B THEN A=0
INPUT	INPUT a string or variable	INPUT A
LET	assign a variable or string a value (LET is optional)	LET A=10
ONERR	ONERRor GOTO line number	ONERR 1000
ONTIME	generate an interrupt when TIME is equal to or greater than ONTIME argument-line number is after comma	ONTIME 10, 1000
ONEX1	GOSUB to line number following ONEX1 when INT1 pin is pulled low	ONEX1 1000
PRINT	PRINT variables, strings or literals P. is shorthand for PRINT	PRINT A
PRINT#	PRINT to software serial port	PRINT# A
PH0.	PRINT HEX mode with zero suppression	PH0. A
PH1.	PRINT HEX mode with no zero suppression	PH1. A
PH0.#	PH0. to line printer	PH0.# A
PH1.#	PH1.# to line printer	PH1.# A

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
PRINT@	PRINT to user defined driver (version 1.1 only)	PRINT@ 5*5
PH0.@	PH0. to user defined driver (version 1.1 only)	PH0. @ XBY(5EH)
PH1.@	PH1. to user defined driver (version 1.1 only)	PH1.@ A
PGM	Program an EPROM (version 1.1 only)	PGM
PUSH	PUSH expressions on argument stack	PUSH 10, A
POP	POP argument stack to variables	POP A, B, C
PWM	PULSE WIDTH MODULATION	PWM 50, 50, 100
REM	REMark	REM DONE
RETI	RETurn from Interrupt	RETI
STOP	break program execution	STOP
STRING	allocate memory for STRINGs	STRING 50, 10
UI1	evoke User console Input routine	UI1
UI0	evoke BASIC console Input routine	UI0
UO1	evoke User console Output routine	UO1
UO0	evoke BASIC console Output routine	UO0
ST@	store top of stack at user specified location (version 1.1 only)	ST@ 1000H ST@ A
LD@	load top of stack from user specified location (version 1.1 only)	LD@ 1000H LD@ A
IDLE	wait for interrupt (version 1.1 only)	IDLE
RROM	run a program in EP(ROM) (version 1.1 only)	RROM 3

1.4 QUICK REFERENCE

OPERATORS — DUAL OPERAND:

OPERATOR	FUNCTION	EXAMPLE(S)
+	ADDITION	1+1
/	DIVISION	10/2
**	EXPONENTATION	2**4
*	MULTIPLICATION	4*4
–	SUBTRACTION	8–4
.AND.	LOGICAL AND	10.AND.5
.OR.	LOGICAL OR	2.OR.1
.XOR.	LOGICAL EXCLUSIVE OR	3.XOR.2

OPERATORS — SINGLE OPERAND:

ABS( )	ABSOLUTE VALUE	ABS(–3)
NOT( )	ONES COMPLEMENT	NOT(0)
INT( )	INTEGER	INT(3.2)
SGN( )	SIGN	SGN(–5)
SQR( )	SQUARE ROOT	SQR(100)
RND	RANDOM NUMBER	RND
LOG( )	NATURAL LOG	LOG(10)
EXP( )	"e" (2.7182818) TO THE X	EXP(10)
SIN( )	RETURNS THE SINE OF ARGUMENT	SIN(3.14)
COS( )	RETURNS THE COSINE OF ARGUMENT	COS(0)
TAN( )	RETURNS THE TANGENT OF ARGUMENT	TAN(.707)
ATN( )	RETURNS ARCTANGENT OF ARGUMENT	ATN(1)

1.4 QUICK REFERENCE

OPERATORS — SPECIAL FUNCTION:

CBY( )	READ PROGRAM MEMORY	P. CBY(4000)
DBY( )	READ/ASSIGN INTERNAL DATA MEMORY	DBY(99) = 10
XBY( )	READ/ASSIGN EXTERNAL DATA MEMORY	P. XBY(10)
GET	READ CONSOLE	P. GET
IE	READ/ASSIGN IE REGISTER	IE = 82H
IP	READ/ASSIGN IP REGISTER	IP = 0
PORT1	READ/ASSIGN I/O PORT 1 (P1)	PORT1 = 0FFH
PCON	READ/ASSIGN PCON REGISTER	PCON = 0
RCAP2	READ/ASSIGN RCAP2 (RCAP2H:RCAP2L)	RCAP2 = 100
T2CON	READ/ASSIGN T2CON REGISTER	P. T2CON
TCON	READ/ASSIGN TCON REGISTER	TCON = 10H
TMOD	READ/ASSIGN TMOD REGISTER	P. TMOD
TIME	READ/ASSIGN THE REAL TIME CLOCK	P. TIME
TIMER0	READ/ASSIGN TIMER0 (TH0: TL0)	TIMER0 = 0
TIMER1	READ/ASSIGN TIMER1 (TH1: TL1)	P. TIMER1
TIMER2	READ/ASSIGN TIMER2 (TH2: TL2)	TIMER2 = 0FFH

STORED CONSTANT:

PI	PI – 3.1415926	PI

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in Instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri, # data	Move immediate data to indirect RAM	2	12
MOV	DPTR, # data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A + DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A + PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Table 11. Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, # data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A, # data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr, # data
44	2	ORL	A, # data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr, # data
54	2	ANL	A, # data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr, # data
64	2	XRL	A, # data
65	2	XRL	A,data addr

Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

8255 MODE Ø SUMMARY

CONTROL CODE IN MODE Ø

PORT A	PORT B	PORT CØ-C3	PORT C4-C7	CODE(HEX)
O	O	O	O	8Ø
O	O	O	I	88
O	O	I	O	81
O	O	I	I	89
O	I	O	O	82
O	I	O	I	8A
O	I	I	O	83
O	I	I	I	8B
I	O	O	O	9Ø
I	O	O	I	98
I	O	I	O	91
I	O	I	I	99
I	I	O	O	92
I	I	O	I	9A
I	I	I	O	93
I	I	I	I	9B

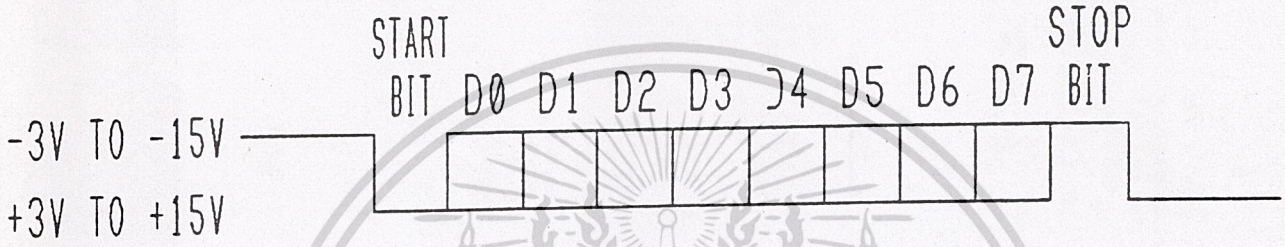
MM58167 SUMMARY

ADDRESS	DATA 7 6 5 4 3 2 1 0	R/W	FUNCTION
FF60	X X X X - - - -	R/W	COUNTER 1/10000 OF SECONDS
FF61	X X X X X X X X	R/W	COUNTER 1/100 OF SECONDS
FF62	- X X X X X X X	R/W	COUNTER SECONDS
FF63	- X X X X X X X	R/W	COUNTER MINUTES
FF64	- - X X X X X X	R/W	COUNTER HOURS
FF65	- - - - X X X X	R/W	COUNTER DAY OF WEEKS
FF66	- - X X X X X X	R/W	COUNTER DAY OF MONTH
FF67	- - - X X X X X	R/W	COUNTER MONTH
FF68	X X X X X X X X	R/W	RAM 1/10000 OF SECONDS
FF69	X X X X X X X X	R/W	RAM 1/100 OF SECONDS
FF6A	X X X X X X X X	R/W	RAM SECONDS
FF6B	X X X X X X X X	R/W	RAM MINUTES
FF6C	X X X X X X X X	R/W	RAM HOURS
FF6D	X X X X X X X X	R/W	RAM DAY OF WEEKS
FF6E	X X X X X X X X	R/W	RAM DAY OF MONTH
FF6F	X X X X X X X X	R/W	RAM MONTH
FF70	X X X X X X X X	R	INTERRUPT STATUS REGISTER
FF71	X X X X X X X X	W	INTERRUPT CONTROL REGISTER
FF72	1 1 1 1 1 1 1 1	W	COUNTER RESET
FF73	1 1 1 1 1 1 1 1	W	RAM RESET
FF74	- - - - - - - X	R	STATUS BIT (1=INVALID DATA)
FF75	- - - - - - - -	W	GO COMMAND
FF76	- - - - - - - X	W	STANDBY INT (1=ENABLE)
FF77	- - - - - - - -	W	TEST MODE

INTERRUPT REGISTER BIT-FORMAT

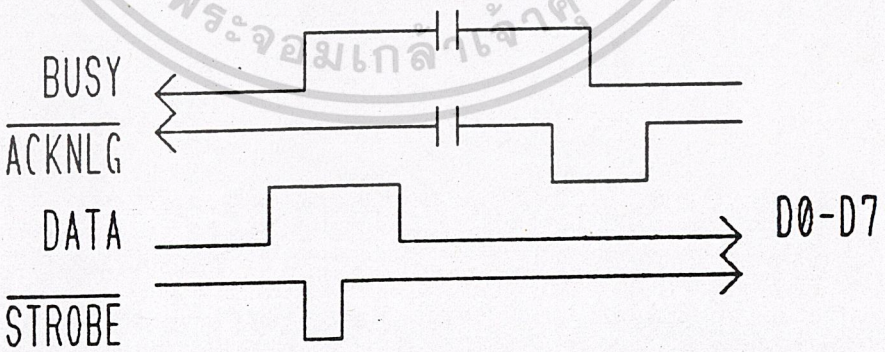
7=MONTH 6=WEEK 5=DAY 4=HOUR 3=MINUTE 2=SEC. 1=1/10SEC. 0=COMPARE

RS232 SIGNAL

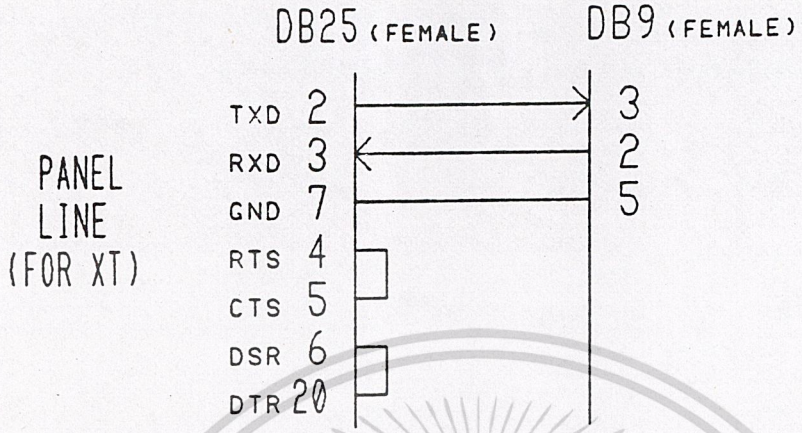


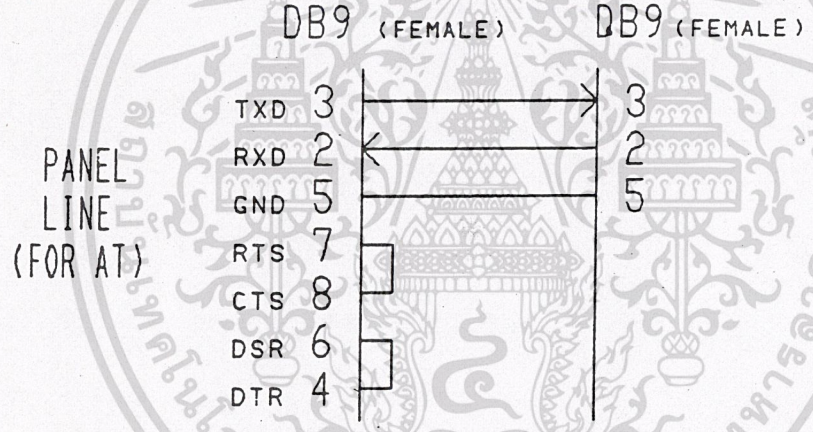
BAUD RATE = 1200, 2400, 4800, 9600
DATA = 8 BIT
STOP BIT = 1 BIT
PARITY BIT = NONE

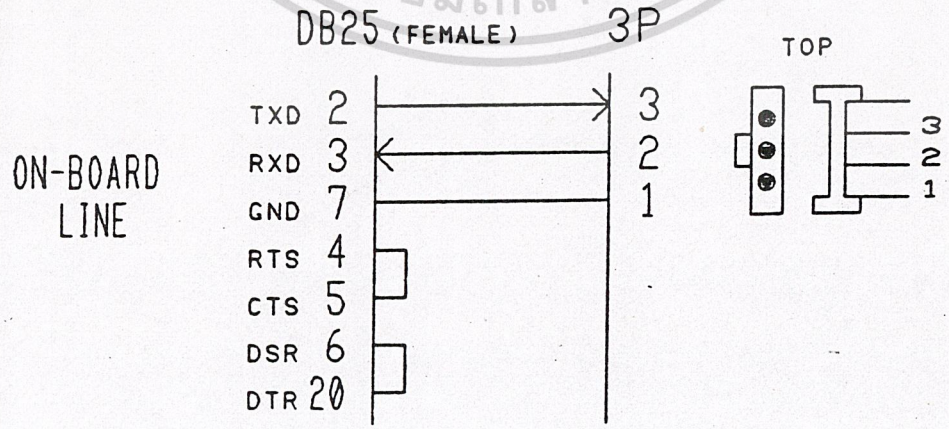
PARALLEL SIGNAL



การต่อสาย RS232

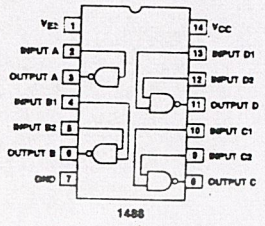


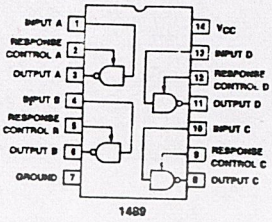


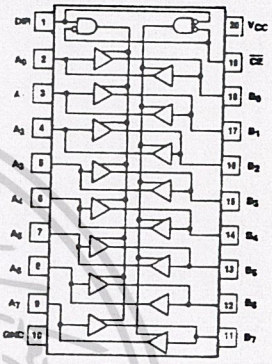


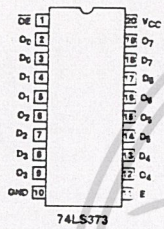
1.10 PIN-OUT LIST

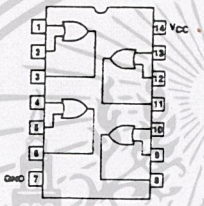
The following is a pin-out list of the most common devices found in an MCS BASIC-52 system:

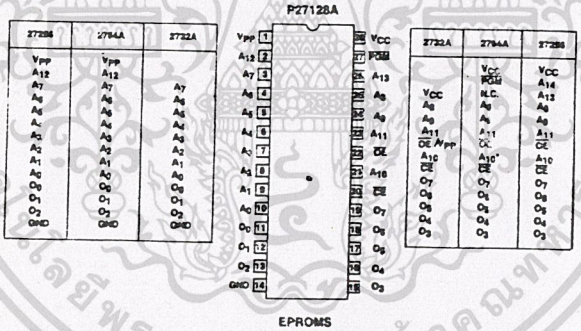


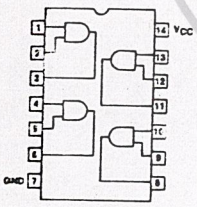


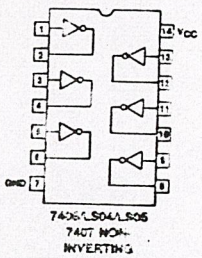


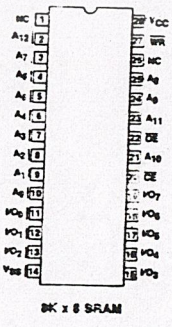


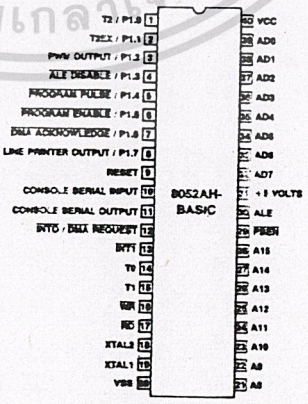


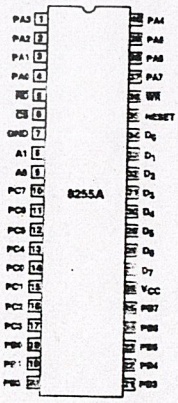












กิตติกรรมประกาศ

ขอขอบคุณท่าน อ.ภาคร หุตะสังกาศ และ อ.ภารไดย ลดาวัลย์ ที่ให้คำปรึกษาเป็นอย่างดี

หนังสืออ้างอิง

1. สมยศ ภูงามแปลง, "MCS-51 ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว", เซมิคอนดัคเตอร์อิเล็คทรอนิคส์, ฉบับที่ 93, หน้า264-269, 2532

2. เกรียงไกร จันทรา, "เครื่องควบคุมสเต็ปปิ้งมอเตอร์", เซ-มิคคอนดัคเตอร์อิเล็คทรอนิคส์, ฉบับที่ 92, หน้า 201-206, 2532

3. สมิธ วรภัทราทร, "8052 เบสิกไมโครคอนโทรลเลอร์",เซ-มิคอนดัคเตอร์อิเล็คทรอนิคส์, ฉบับที่ 96, หน้า 218-225, 2532

4. INTEL, "MCS BASIC-52 User'Mannal", INTEL,1987

5. INTEL, "MICROPROCESSOR DATA BOOK MCS-51 MI-CROCONTROLLERS", INTEL, 1980

6. HYUNDAI, "16 BIT PERSONAL COMPUTER USER'S GUIDE SUPER-286C", KOREA, 1988

7. IBM, "VLSI 12 MHZ ZERO-WAIT 286 TURBO MAIN-BOARD",IBM, 1987