ปีการศึกษา 2531

ELECTRONIC BANNER

(Programmable Computer Disply)

โดย

นายไพฑูรย์ หวังเพ็ชรงาม

สาขาเทคโนโลยีอิเล็กทรอนิกส์

อาจารย์ที่ปรึกษา

ผศ.นิกร สุขุตมตันติ

ELECTRONIC BANNER

(Programmable Computer Display)

ไพฑูรย์ หวังเพ็ชรงาม
ผศ.นิกร สุขุตมตันติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2531

บทคัดย่อ

"ELECTRONIC BANNER" เป็นแผงตัวอักษรคอมพิวเตอร์ ที่ใช้แผงดิสเพลย์แบบ DOT MATRIX LED ขนาด 10 ตัวอักษร (8 * 80 จุด) มีลักษณะโครงสร้างโดยทั่วไปคล้ายกับเครื่องคอมพิวเตอร์ คือมีคีย์บอร์ดเป็นอุปกรณ์อินพุท มีไมโครโปรเซสเซอร์เป็นตัวควบคุมการทำงานทั้งหมด มีหน่วยความจำไว้เก็บข้อมูลที่ทำการโปรแกรม และมีแผงดิสเพลย์เป็นอุปกรณ์เอาท์พุท สามารถแสดงผลได้ทั้งข้อความตัวอักษร(TEXT) และรูปภาพกราฟฟิค(GRAPHIC) นอกจากนั้นยังสามารถทำการโปรแกรมรูปแบบการแสดงผลได้อีกด้วย ซึ่งลักษณะการโปรแกรมคล้ายกับการโปรแกรมภาษาคอมพิวเตอร์

สารบัญ

บทที่ 1
บทนำ

ในปัจจุบันนี้ตามห้างสรรพสินค้าและร้านค้าต่าง ๆ ได้ปรากฏมีสื่อโฆษณาชนิดใหม่ขึ้นมา ซึ่งสามารถดึงดูดผู้คนที่เดินผ่านไปมาได้ดีสื่อโฆษณาที่กล่าวถึงนี้คือ"แผงตัวอักษรคอมพิวเตอร์" แผงตัวอักษรทั่วไปมีรูปแบบของตัวอักษรและลักษณะการแสดงผลที่ถูกกำหนดไว้ตายตัว ทำการแสดงผลวนไปวนมา หากต้องการเปลี่ยนแปลงตัวอักษรหรือรูปแบบการแสดงผลใหม่ก็ต้องทำการเปลี่ยนข้อมูลในไอซีหน่วยความจำถาวรที่เรียกกันว่า ROM (Read Only Memory) ซึ่งทำให้ยุ่งยากและไม่สดวก รวมทั้งยังต้องสิ้นเปลืองค่าใช้จ่ายในการโปรแกรม ROM ใหม่อีก

ด้วยเหตุนี้เอง จึงก่อให้เกิดแนวความคิดที่จะสร้างแผงตัวอักษรคอมพิวเตอร์ที่สามารถกำหนดรูปแบบของตัวอักษรและลักษณะการแสดงผลได้ง่าย มีรูปแบบการแสดงผลมากขึ้นโดยไม่ต้องทำการแก้ไขข้อมูลใน ROM และไม่ต้องเสียค่าใช้จ่ายใด ๆ อีกทั้งยังสามารถแสดงผลในลักษณะรูปภาพกราฟฟิคได้อีกด้วย แผงตัวอักษรคอมพิวเตอร์ที่กล่าวถึงนี้เราเรียกมันว่า " ELECTRONIC BANNER "

ELECTRONIC BANNER เป็นแผงตัวอักษรคอมพิวเตอร์ ใช้ไมโครโปรเซสเซอร์ขนาด 8 บิทเบอร์ Z80A ของบริษัท Zilog เป็นตัวควบคุมการทำงานทั้งหมด ส่วนประกอบโดยทั่วไปทางฮาร์ดแวร์ประกอบไปด้วย แผงแสดงผล DOT MATRIX LED ขนาด 8 * 80 จุด, แผงควบคุม (CONTROL BOARD), แผงคีย์บอร์ด และแผงวงจรภาคจ่ายไฟ

ส่วนทางด้านโปรแกรมควบคุมหรือซอฟท์แวร์นั้นกล่าวคร่าว ๆ ได้ดังนี้

- สามารถแสดงผลได้ทั้งตัวอักษร (TEXT) และรูปภาพกราฟฟิค (GRAPHIC)
- การโปรแกรมทำในลักษณะเดียวกับการโปรแกรมภาษาคอมพิวเตอร์ทั่วไป โดยในแบนเนอร์นี้จะมีชุดคำสั่งที่ใช้กำหนดรูปแบบของตัวอักษรและลักษณะการแสดงผล ที่ออกแบบขึ้นมาโดยเฉพาะ
- สามารถโปรแกรมตัวอักษรและรูปภาพได้ง่าย สดวก โดยไม่ต้องเปลี่ยนแปลงข้อมูลใน ROM เลย แต่จะใช้การโปรแกรมข้อมูลลงใน RAM (Random Access Memory) โดย RAM ส่วนนี้ใช้เก็บโปรแกรมแบนเนอร์ที่ทำการโปรแกรมเข้าไปทางคีย์บอร์ด และมีแบตเตอรี่สำรอง (back up) ต่อไว้ในกรณีไฟดับเพื่อป้องกันข้อมูลที่ทำการโปรแกรมไว้แล้วสูญหาย
- มีรูปแบบการแสดงผลได้หลายแบบ ซึ่งสามารถใช้ได้กับตัวอักษร (TEXT) และรูปภาพกราฟฟิค (GRAPHIC)

บทที่2

ดิสเพลย์และวงจรขับ (Display and Driver)

ในงานด้านไฟฟ้า อิเล็กทรอนิกส์และคอมพิวเตอร์นั้นจะต้องมีสิ่งที่บ่งบอกถึงการทำงานต่าง ๆ เกี่ยวกับระบบให้ผู้ใช้ทราบถึงขั้นตอนในการทำงานและแสดงผลที่ได้ ซึ่งเราเรียกส่วนนี้ว่า ดิสเพลย์ (Display) หรือส่วนแสดงผล ดิสเพลย์ที่ใช้อยู่ทั่วไปในเครื่องใช้ไฟฟ้าทั่วไปนั้นมีหลายอย่างหลายประเภทเช่น หลอดไฟแบบไส้,หลอดแอลอีดี (LED),7-SEGMENT LED, DOT MATRIX LED, หลอดฟลูออเรสเซนต์ เป็นต้น

ดิสเพลย์ต่าง ๆ ที่กล่าวมานี้มีดิสเพลย์อยู่ชนิดหนึ่งที่สามารถแสดงตัวอักษร,อักขระและรูปแบบต่าง ๆ ได้ นั่นคือ DOT MATRIX DISPLAY ในที่นี้จะกล่าวถึงเฉพาะ DOT MATRIX LED DISPLAY ซึ่งดิสเพลย์แบบนี้ประกอบขึ้นจากหลอดแอลอีดีหลายตัวประกอบกันในลักษณะแมทริกซ์ทำให้สามารถแสดงรูปแบบต่างกันได้ ดิสเพลย์แบบนี้มีประโยชน์มากคือ สามารถให้รายละเอียดต่าง ๆ ได้มากกว่าดิสเพลย์แบบอื่น ซึ่งความละเอียดของดิสเพลย์แบบนี้ขึ้นอยู่กับจำนวนของหลอดแอลอีดีที่นำมาประกอบกันเป็นแมทริกซ์ และระยะห่างของหลอดแอลอีดีแต่ละหลอด ดังนั้นดิสเพลย์แบบนี้จึงถูกนำมาใช้เป็นหน่วยแสดงผลที่ต้องการความละเอียดและแผงไฟโฆษณาต่าง ๆ ที่มีปรากฏให้เห็นทั่วไป

วงจรที่นำมาใช้งานกับแผงดิสเพลย์แบบ DOT MATRIX LED กล่าวคร่าว ๆ แบ่งออกได้ 2 ลักษณะคือ

2.1 วงจรแลทช์ (LATCH)

ซึ่งก็คือวงจรฟลิปฟลอบนั่นเอง กล่าวคือ ใช้ไอซีแลทช์ทำหน้าที่ขับหลอดแอลอีดีของ DOT MATRIX LED โดยตรง และจะคงค้าง (ติดค้าง) อยู่เรื่อยไปจนกว่าจะมีข้อมูลใหม่เข้ามา วิธีนี้มีข้อดีคือ ความสว่างของหลอดแอลอีดีจะสว่างสูงสุด แต่มีข้อเสียคือ หากต่อเพื่อใช้กับหลอดแอลอีดีจำนวนมากแล้วจะต้องใช้ไอซีแลทช์เพิ่มขึ้นตามจำนวนของหลอดแอลอีดี ตัวอย่างเช่น ถ้าต้องการใช้ไอซีแลทช์ขับดิสเพลย์แบบ DOT MATRIX LED ขนาด 8 * 8 ต้องใช้ไอซีแลทช์ขนาด 8 บิทถึง 8 ตัว และถ้าต่อใช้กับดิสเพลย์แบบนี้ถึง 10 หลัก ก็ต้องใช้ไอซีแลทช์ถึง 80 ตัว ทำให้สิ้นเปลืองแหล่งจ่ายไฟสิ้น เปลืองเนื้อที่ และที่สำคัญคือ สิ้นเปลืองค่าใช้จ่าย

2.2 วิธีมัลติเพล็กซ์ (MULTIPLEX)

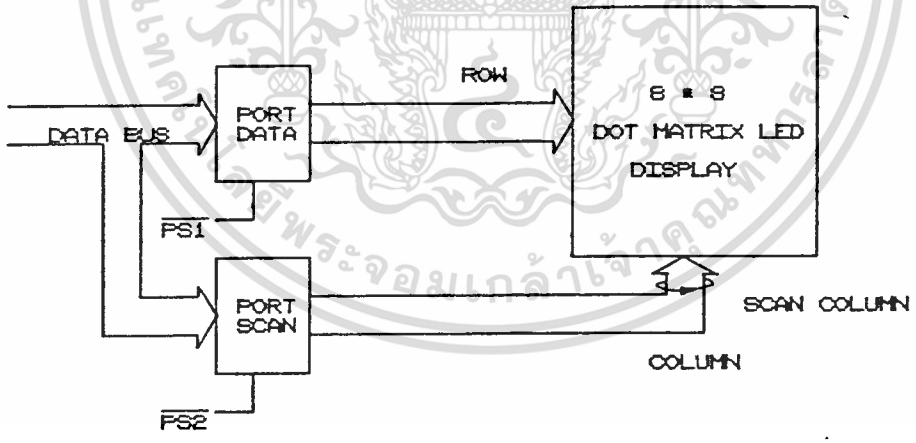
กล่าวคือใช้เทคนิคการสแกน (SCAN) โดยให้หลอดแอลอีดีของ DOT MATRIX LED แต่ละแถว (ROW) หรือแต่ละคอลัมท์ (COLUMN) ติดพร้อมกันเพียงแถวหรือคอลัมท์เดียวเท่านั้น ให้คงค้างอยู่ชั่วขณะหนึ่ง ต่อไปจึงให้หลอดแอลอีดีแถวหรือคอลัมท์ถัดไปติดพร้อมกัน ทำ

ในลักษณะเดียวกันนี้จนครบทุกแถวหรือทุกคอลัมท์ แล้วกลับไปทำการสแกนใหม่เรื่อยไปด้วยความเร็วสูงมากจนทำให้ตาเรามองเห็นว่า DOT MATRIX LEDติดพร้อมกันทุกแถวทุกคอลัมท์ ซึ่งด้วยวิธีการนี้ได้ถูกนำมาใช้ในโครงงานนี้

วิธีการสแกนมีข้อดีคือ ประหยัดวงจร ค่าใช้จ่าย และกินกระแสน้อยกว่าวิธีใช้วงจรแลทช์ วิธีนี้เหมาะสำหรับใช้กับหลอดแอลอีดีจำนวนมาก แต่ก็มีข้อเสียคือ ความสว่างรวมของ DOT MARIX LED จะลดลงแน่นอนต้องสว่างน้อยกว่าใช้วงจรแลทช์ แต่สามารถแก้ไขได้โดยใช้ทรานซิสเตอร์เป็น DRIVER ช่วยขยายกระแสให้สูงขึ้นเพื่อขับให้หลอดแอลอีดีสว่างขึ้นและใช้เทคนิคในการสแกน (SCAN TECHNIQUE) ช่วยได้

วิธีการสแกนนี้แบ่งออกได้ตามทิศทางของการสแกนคือ การสแกนคอลัมท์ (COLUMN SCAN) และการสแกนแถว (ROW SCAN)

2.2.1 การสแกนคอลัมท์ การสแกนแบบนี้จะประกอบไปด้วยเอาท์พุทพอร์ท 2 พอร์ท โดยพอร์ทหนึ่งจะเป็นพอร์ทข้อมูลป้อนเข้าทางแถวของ DOT MATRIX LED ส่วนอีกพอร์ทหนึ่งจะต่อเข้าทางคอลัมท์เป็นพอร์ทที่ทำการเลือกคอลัมท์ใด ๆ ของ DOT MATRIX LED ให้ติดสว่าง ดังรูปที่ 1 ซึ่งทุกครั้งที่มีการสแกน 1 คอลัมท์ ข้อมูลที่พอร์ทข้อมูลก็จะเปลี่ยนไปตามรูปแบบของตัวอักษรที่ต้องการแสดง

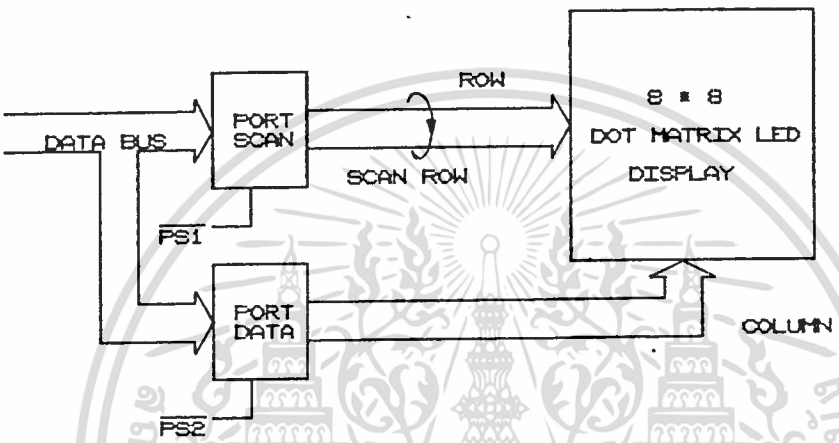


รูปที่ 1 แผนผังการทำงานของการสแกนดิสเพลย์โดยการสแกนคอลัมท์

จากการทดลองในครั้งแรกได้ใช้การสแกนแบบนี้ เมื่อทดลองต่อดูแล้วจึงได้รู้ว่าการสแกนคอลัมท์เหมาะที่จะใช้กับ DOT MATRIX LED เพียงหลักเดียวที่เป็นเช่นนี้เพราะว่าเมื่อทำการต่อ DOT MATRIX LED เพิ่มขึ้นจนถึง 10 หลัก จะมีผลทำให้เห็นรูปตัวอักษรที่ปรากฏบนดิสเพลย์นั้นโย้ไปเย้มา ลักษณะของตัวอักษรไม่นิ่งมีลักษณะคล้ายคลื่น แต่ถ้าต่อเพียงหลัก

เดียวแล้วตาคนเราจะไม่สามารถจับการเคลื่อนไหวโย้ไปเย้มาของตัวอักษรที่ปรากฏได้ จึงแลดูแล้วเห็นตัวอักษรนิ่งเป็นปกติ ดังนั้นจึงเปลี่ยนไปใช้การสแกนแถว

2.2.2 การสแกนแถว การสแกนแบบนี้มีลักษณะคล้ายกับการสแกนคอลัมท์ เพียงแต่กลับกันตรงแทนที่จะให้พอร์ทข้อมูลป้อนเข้าที่แถวก็มาป้อนเข้าที่คอลัมท์ของDOT MATRIX LED ส่วนพอร์ทสแกนก็เปลี่ยนจากคอลัมท์มาเป็นแถว เพื่อทำการสแกนแถวดังรูปที่ 2



รูปที่ 2 แผนผังการทำงานของการสแกนดิสเพลย์โดยการสแกนแถว

เมื่อได้ต่อทดลองดูแล้ว วิธีการสแกนแบบนี้ให้ผลเป็นที่น่าพอใจ ตัวอักษรที่ปรากฏบนดิสเพลย์นิ่งไม่สั่นพริ้ว มีลักษณะตรง ไม่เป็นคลื่นเหมือนในการสแกนคอลัมท์ ถึงแม้จะทำการต่อดิสเพลย์หลายหลักก็ตาม ดังนั้นวิธีการสแกนดิสเพลย์แบบการสแกนแถวจึงถูกนำมาใช้ในโครงงานนี้

บทที่ 3
โครงสร้างทางฮาร์ดแวร์ของอิเล็กทรอนิกส์แบนเนอร์

ในการออกแบบระบบที่ใช้ไมโครโปรเซสเซอร์เป็นตัวควบคุมการทำงาน ไม่ว่าจะใช้ CPU เบอร์ใดก็ตาม สิ่งที่จะต้องพิจารณาเป็นอันดับแรกคือ ต้องกำหนดขอบเขตและขนาดของ โดยจะต้องวางแนวทางของส่วนฮาร์ดแวร์ให้เหมาะสม และสอดคล้องกับซอฟท์แวร์ที่จะเขียน ด้วย เพราะในบางส่วนนั้นฮาร์ดแวร์และซอฟท์แวร์ต้องไปด้วยกัน

เราสามารถแบ่งโครงสร้างการทำงานของอิเล็กทรอนิกส์แบนเนอร์ได้เป็นโครงสร้าง ทางฮาร์ดแวร์ ซึ่งเป็นส่วนของวงจรทั้งหมด และโครงสร้างทางซอฟท์แวร์ ก็คือตัวโปรแกรม ที่ใช้ควบคุมการทำงานทั้งหมด ซึ่งเมื่อเริ่มออกแบบระบบนั้นส่วนฮาร์ดแวร์และซอฟท์แวร์ของ ระบบต้องสัมพันธ์กันอย่างมาก ฮาร์ดแวร์ที่ทำขึ้นนั้นผู้ออกแบบซอฟท์แวร์จะต้องสามารถใช้และ เขียนโปรแกรมได้โดยง่ายและการใช้อุปกรณ์ทางฮาร์ดแวร์เพิ่มขึ้นก็ทำให้การเขียนซอฟท์แวร์ ง่ายขึ้น และในทางกลับกันการใช้เทคนิคในการเขียนซอฟท์แวร์ก็สามารถลดอุปกรณ์ทางฮาร์ด แวร์ได้เช่นกัน

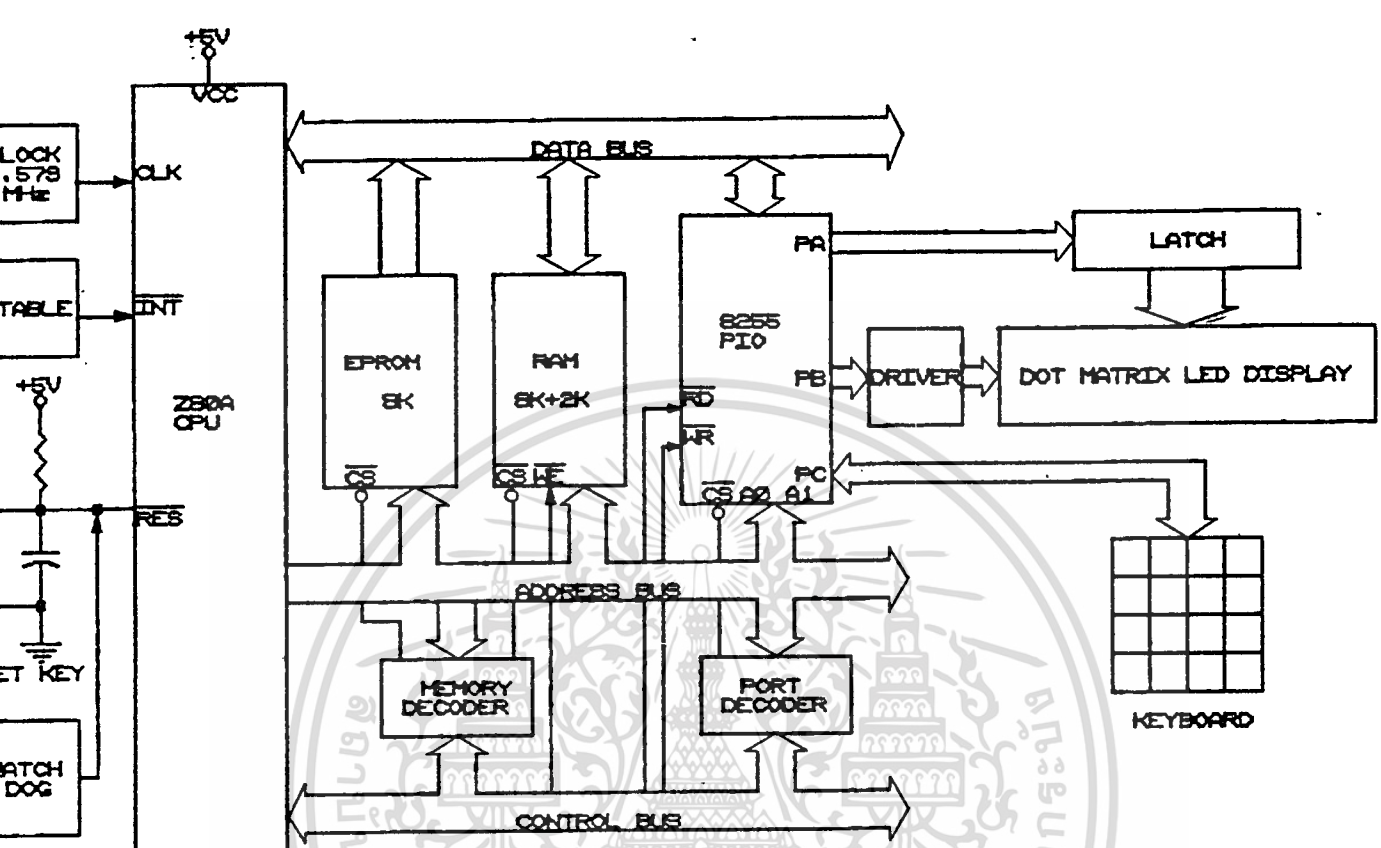
3.1 แผนผังฮาร์ดแวร์

เพื่อให้ดูวงจรฮาร์ดแวร์ได้ง่าย จึงได้แสดงแผนผังทางฮาร์ดแวร์ไว้ในรูปที่ 3 ส่วน ประกอบที่สำคัญของอิเล็กทรอนิกส์แบนเนอร์ ได้แก่ CPU, หน่วยความจำ (ROM,RAM) , พอร์ท, DRIVER , Astable Multivibrator, Watch Dog , DOT MATRIX LED DISPLAY และส่วนจ่ายไฟเลี้ยง

3.2 การทำงานของวงจร

วงจรในรูปที่ 4 แสดงถึงส่วนของ CPU, หน่วยความจำ, ส่วนถอดรหัสแอดเดรส ของหน่วยความจำและพอร์ท, ส่วนกำเนิดสัญญาณนาฬิกา, ส่วนกำเนิดสัญญาณมาสเคเบิ้ลอิน เตอร์รัพท์ (Maskable Interrupt) และส่วนของวงจร Watch Dog โดยที่ Z80A CPU (U1) ติดต่อกับหน่วยความจำ 2 ส่วน ส่วนแรกคือ U2 ซึ่งเป็น EPROM เบอร์ 2764 ขนาด 8 กิโลไบท์ ภายใน EPROM นี้บรรจุด้วยโปรแกรมมอนิเตอร์เพื่อควบคุมการทำงานของอิเล็ก ทรอนิกส์แบนเนอร์และเป็นส่วนของ CHARACTER GEN. ที่เก็บรูปแบบ (pattern) ของตัว อักษรตามรหัสแอสกี้ถึง 128 ตัวอักษร (ขนาด 1 กิโลไบท์) ส่วนที่สองเป็น RAM คือ U3 เบอร์ 6264 (8 กิโลไบท์) ทำหน้าที่เก็บโปรแกรมของแบนเนอร์ที่ผู้ใช้เขียนขึ้นมา และ U4 เบอร์ 6116 (2 กิโลไบท์) ใช้ในระบบเพื่อเป็นสแตค (stack) และส่วนใช้งานของ โปรแกรมมอนิเตอร์

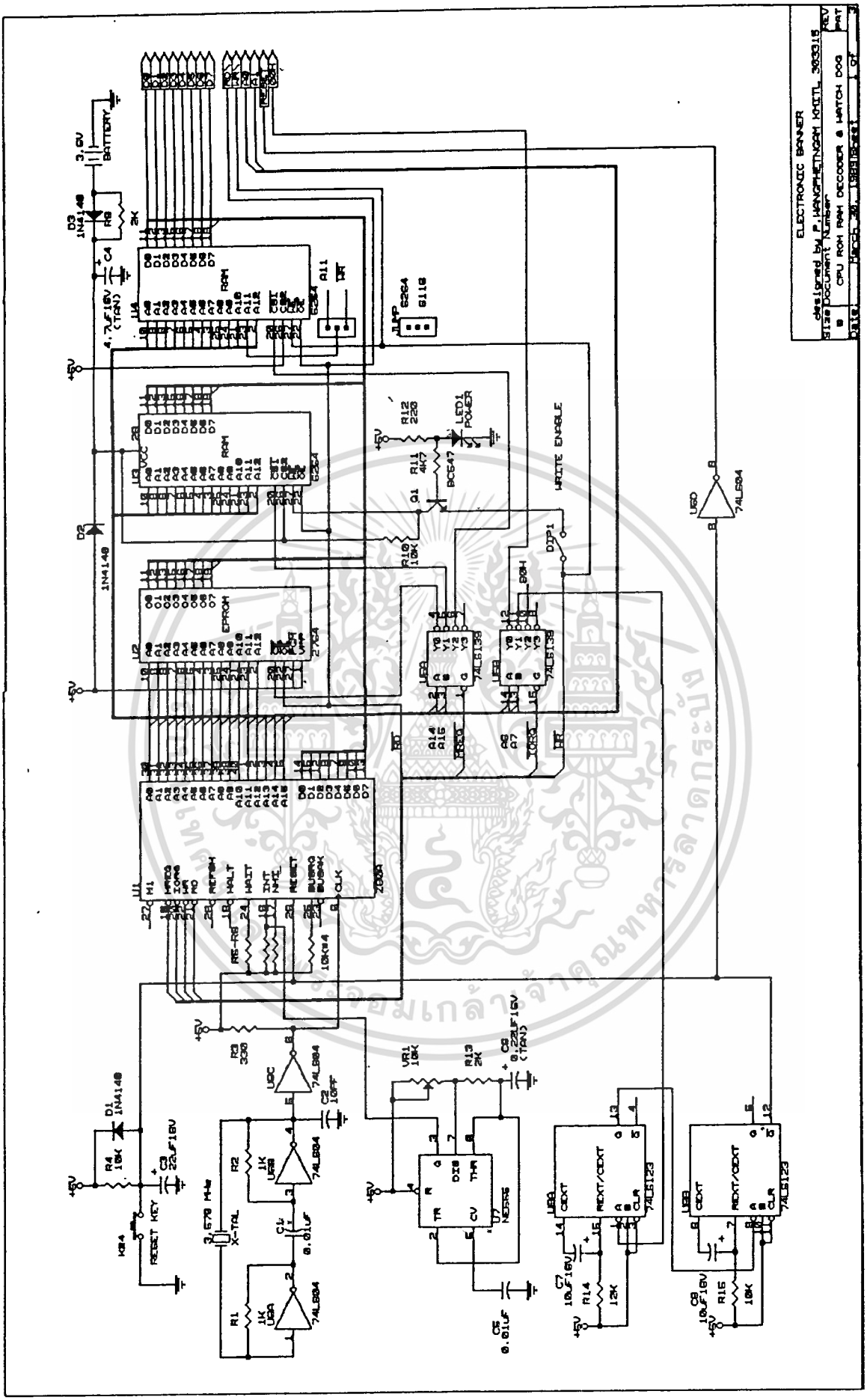
U5 เบอร์ 74LS139 แบ่งเป็น U5A ทำหน้าที่ถอดรหัสแอดเดรสของหน่วยความจำ



รูปที่ 3 แผนผังทางฮาร์ดแวร์ของอิเล็กทรอนิกส์แบนเนอร์

เพื่อจัดให้ U2 (2764) อยู่ที่แอดเดรส 0000H - 1FFFH, U3 (6264) อยู่ที่แอดเดรส 4000H - 5FFFH และ U4 (6116) อยู่ที่แอดเดรส 8000H - 87FFH (U4 นี้สามารถเลือกใช้ RAM เบอร์ 6116 หรือ 6264 ก็ได้) ส่วน U5B ทำหน้าที่ถอดรหัสแอดเดรสของพอร์ทโดยจัดให้ U9 (8255) อยู่ในตำแหน่งหมายเลขพอร์ทที่ 00H - 3FH และ U8 (74LS123) อยู่ในตำแหน่งหมายเลขพอร์ทที่ 40H - 7FH และที่ U3 ได้ต่อขาไฟเลี้ยงเข้ากับแบตเตอรี่สำรองเอาไว้ด้วยเพื่อให้โปรแกรมแบนเนอร์ที่ป้อนเข้ามาเก็บไว้ไม่สูญหายขณะที่ไฟดับหรือปิดเครื่อง ซึ่งสามารถเก็บโปรแกรมไว้ได้นานเป็นเดือน นอกจากนี้ที่ขา 27(WE) ของ U3 ยังได้ต่อสวิทช์ DIP1 ปิดเปิดสัญญาณ WR จาก CPU ด้วย เพื่อป้องกันโปรแกรมสูญหายอันเกิดจากการเขียนโปรแกรมแบนเนอร์ผิดพลาด

U6A และU6B ทำงานร่วมกับ X-TAL ความถี่ 3.579 MHz เป็นวงจรกำเนิดสัญญาณ



รูปที่ 4 วงจรในส่วนของ CPU,หน่วยความจำ,Decoder,Astable และ Watch Dog

นาฬิกาป้อนเข้าขาคล๊อกของ Z80A CPU โดยตรงโดยมี U6C เป็นบัฟเฟอร์ ส่วนสัญญาณรีเซทได้จาก R4, C3 และ D1 ซึ่งจะรีเซท CPU ขณะเปิดเครื่อง และมีสวิทซ์ K24 เป็นสวิทซ์รีเซทต่างหากอีกด้วย ส่วน U6D นั้นใช้สร้างสัญญาณรีเซทป้อนให้กับ U9 (8255)และ U10-U19 (74LS374)

U7 ใช้ไอซีไทม์เมอร์ยอดฮิตเบอร์ 555 ต่อเป็นวงจรอะสเตเบิ้ล มัลติไวเบรเตอร์ (Astable Multivibrator) เพื่อสร้างสัญญาณมาสเคเบิ้ลอินเตอร์รัพท์ (Maskable Interrupt) ไปต่อเข้าที่ขา 16 ($\overline{INT}$) ซึ่งเป็นขาอินเตอร์รัพท์ของ CPU โดยที่ CPU จะกระโดดไปทำงานใน โปรแกรมตอบสนองการอินเตอร์รัพท์ (interrupt service routine) ที่แอดเดรส 0038H ซึ่งเป็นโปรแกรมที่ใช้ในการสแกนดิสเพลย์

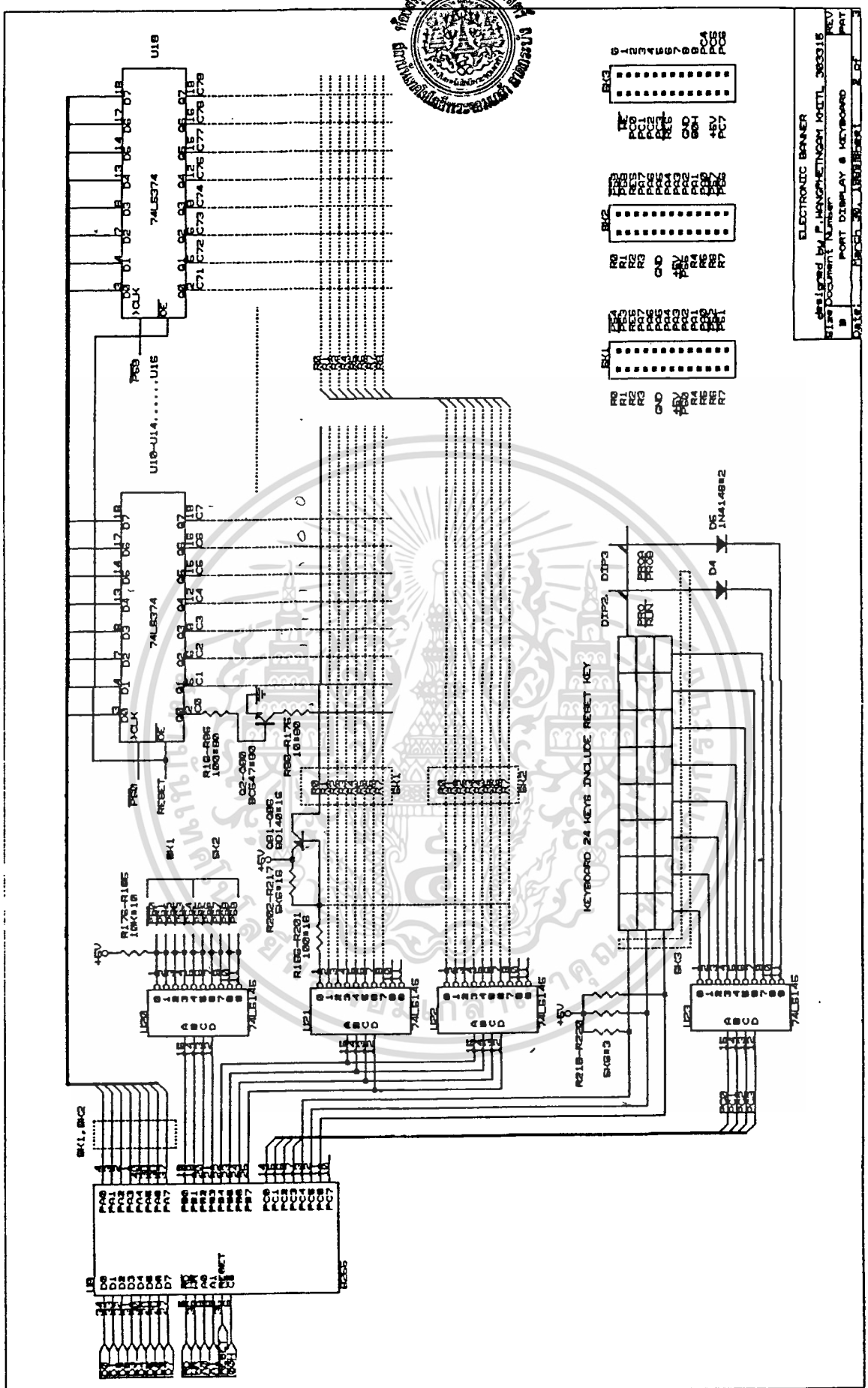
U8 เบอร์ 74LS123 เป็นไอซี Retriggerable Monostable Multivibrator แบ่งเป็น U8A และ U8B ต่อร่วมกันเป็นระบบ Watch Dog ขาอินพุทของ U8A ต่อมาจาก U5B ที่ตำแหน่งหมายเลขพอร์ทที่ 40H - 7FH และเอาท์พุทของ U8B ต่อเข้ากับขา 26 (RESET) ของ CPU ระบบ Watch Dog นี้มีความสำคัญมากในระบบไมโครโปรเซสเซอร์ ซึ่งจะกล่าวถึงรายละเอียดของระบบนี้ในหัวข้อถัดไป

วงจรในรูปที่ 5 เป็นส่วนของพอร์ททั้งหมดคือ เป็นส่วนที่ติดต่อกับผู้ใช้ ได้แก่คีย์บอร์ด และแผงดิสเพลย์ DOT MATRIX LED วงจรในส่วนนี้ใช้พอร์ทยอดนิยมเบอร์ 8255 เพียงตัวเดียวทำงานทั้งหมด ซึ่งถูกถอดรหัสโดย U5B ให้ 8255 อยู่ในตำแหน่งหมายเลขพอร์ทที่ 00H - 3FH

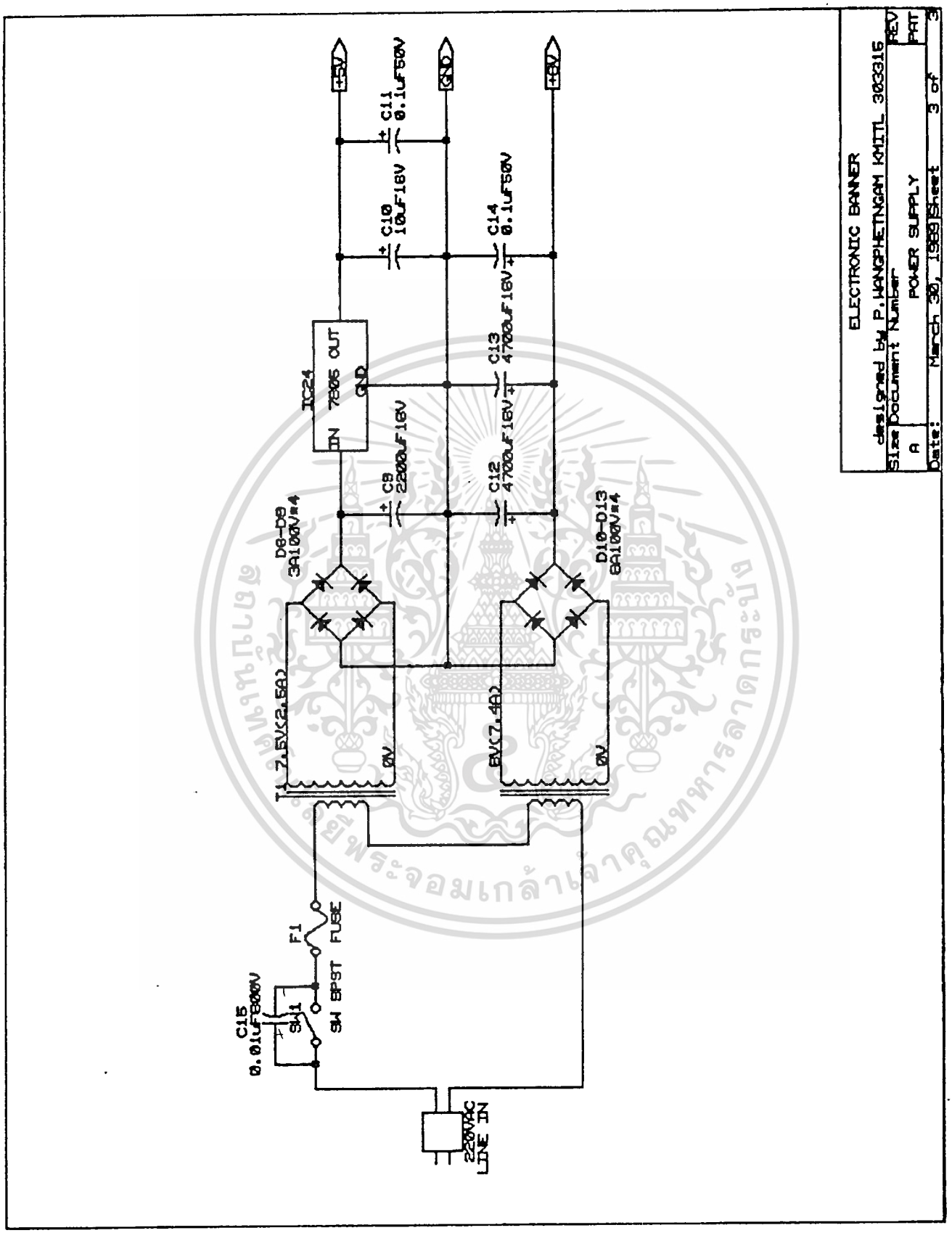
U9 (8255) ประกอบด้วยพอร์ท 3 พอร์ทคือ พอร์ท A, พอร์ท B และพอร์ท C พอร์ทละ 8 บิท โดยพอร์ท C แบ่งเป็น 2 พอร์ทย่อยคือ พอร์ท C ล่างเป็นพอร์ทเอาท์พุท ส่วนพอร์ท C บนเป็นพอร์ทอินพุท

พอร์ท A คือขา PA0-PA7 ต่อเข้ากับ U10-U19 โดยผ่านคอนเนคเตอร์ SK1 และ SK2 พอร์ท A นี้ทำหน้าที่เป็นบัสข้อมูลให้กับ U10-U19 U10-U19 (74LS374) เป็นพอร์ทเอาท์พุททำหน้าที่เป็นพอร์ทข้อมูลให้กับแผงดิสเพลย์ทั้ง 10 หลัก โดยต่อผ่าน Q2-Q80 ทำหน้าที่ขยายกระแสให้สูงขึ้นเพื่อจ่ายให้กับ DOT MATRIX LED DISPLAY ทางแนวคอลัมท์ทั้ง 80 คอลัมท์

พอร์ท B แบ่งออกเป็น พอร์ท B ล่างกับพอร์ท B บน พอร์ท B ล่าง (PB0-PB3) ต่อเข้ากับ U20 (74LS145) ทำหน้าที่ถอดรหัสเพื่อสร้างสัญญาณสำหรับเลือกสำหรับเลือกพอร์ท U10-U19 การแสดงผลของดิสเพลย์นี้เป็นแบบมัลติเพล็กซ์ทำการสแกนแถว คือให้หลอดแอลอีดีของแผงดิสเพลย์ทั้งหมดติดทีละแถวเรียงกันไปเรื่อย ๆ ด้วยความเร็วสูงมาก สัญญาณสแกนให้หลอดแอลอิดีติดสว่างทีละแถวได้มาจากพอร์ท B บนคือ ขา PB4-PB7 ป้อนเข้า U21 และ U22 (74LS145) ทำการถอดรหัสเลือกแถวเพื่อขับ DOT MATRIX LED ทางแนว ROW ทั้ง 8 ROW โดยมี Q81-Q96 ทำหน้าที่ Driver ขยายกระแสให้สูงขึ้นเพื่อ



รูปที่ 5 วงจรในส่วนของพอร์ททั้งหมด



รูปที่ 6 วงจรในส่วนภาคจ่ายไฟ

ขับให้ DOT MATRIX LED สว่าง

พอร์ท C โดยพอร์ท C ล่าง (PC0-PC3) ต่อเข้ากับ U23 ทำการสแกนคีย์บอร์ดและดิพสวิทช์ โดยมีพอร์ท C บน (PC4-PC6) เป็นอินพุทรับข้อมูลของคีย์บอร์ดซึ่งต่อเป็นแบบแมทริกซ์ขนาด 8 * 3 คีย์ แต่ว่างไว้ 1 คีย์ จึงเหลือเพียง 23 คีย์ เป็นคีย์คำสั่งของแบนเนอร์ 12 คีย์และคีย์ควบคุมการโปรแกรมอีก 11 คีย์ นอกจากขา PC4-PC6 จะเป็นอินพุทรับข้อมูลของคีย์บอร์ดแล้วยังเป็นอินพุทรับข้อมูลของดิพสวิทช์ด้วย ซึ่งมีดิพสวิทช์ต่ออยู่ 2 ตัวคือ DIP2 และ DIP3 โดย DIP2 ใช้เลือกโหมดการทำงานของแบนเนอร์ว่าจำทำการโปรแกรมหรือทำการรัน (PRO/RUN) ส่วน DIP3 ใช้เลือกโปรแกรมแบนเนอร์ว่าจะทำการโปรแกรมหรือรัน โปรแกรม A หรือโปรแกรม B (PROA/PROB)

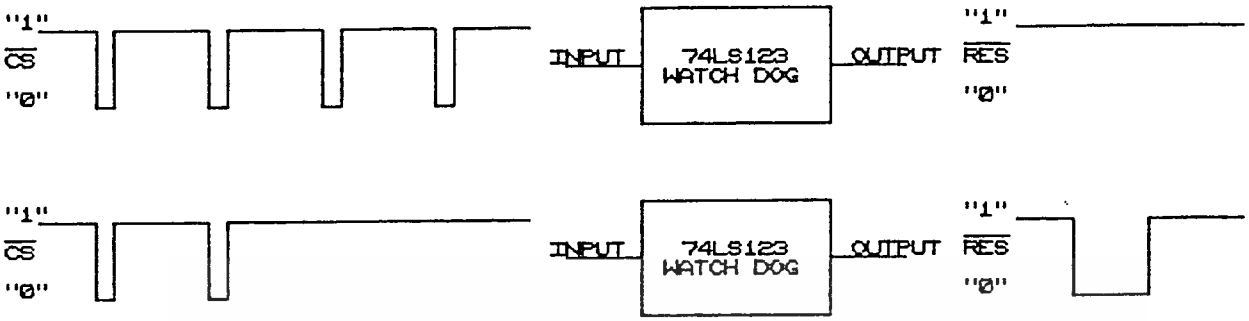
วงจรในรูปที่ 6 เป็นส่วนของภาคจ่ายไฟ ไฟเลี้ยงของวงจรทั้งหมดยกเว้น Q81-Q96 ได้มาจากหม้อแปลง T1 ขดแรกแปลงไฟให้เหลือ 7.5 โวลท์ TAP 7.5V - 0 AC. โดยผ่านไดโอดบริดจ์เร็คติไฟเออร์ D6-D9 และกรองกระแสด้วย C9 ก็จะได้แรงดันออกมาประมาณ 9 โวลท์ แล้วต่อผ่าน IC24 (7805) ให้เหลือไฟตรง 5 โวลท์เพื่อเอาไปเลี้ยงวงจร

ส่วนขดของหม้อแปลง T1 อีกขดหนึ่งแปลงไฟให้เหลือ 6 โวลท์ TAP 6V - 0 AC. ต่อผ่านไดโอดบริดจ์เร็คติไฟเออร์ D10-D13 และกรองกระแสด้วย C12 และ C13 ได้แรงดันประมาณ 8 โวลท์ เพื่อต่อเอาไปเลี้ยงวงจรในส่วนของการสแกนดิสเพลย์ที่ Q81-Q96 ใช้ในการขับดิสเพลย์ DOT MATRIX LED

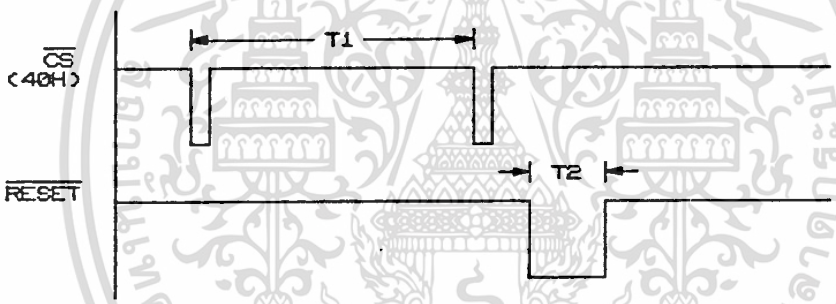
3.3 <u>หลักการของระบบ Watch Dog</u>

ปกติการใช้งานระบบไมโครโปรเซสเซอร์นั้นมักจะมีปัญหาเรื่อง การแฮงก์ (HANG) ของโปรแกรมบ่อย ๆ ทั้งนี้อาจเกิดจากสัญญาณรบกวนทางไฟฟ้าซึ่งส่วนใหญ่แล้วจะเข้ามาทางเอซีไลน์หรืออาจเกิดจากโปรแกรมที่เขียนขึ้นเอง เราสามารถใช้ระบบ Watch Dog แก้ปัญหานี้ได้ โดยมีหลักการคือ ให้โปรแกรมที่เขียนขึ้นทำการส่งสัญญาณมากระตุ้น (trig) ระบบ Watch Dog อยู่ตลอดเวลาซึ่งถ้าเกิดการแฮงก์จะทำให้โปรแกรมทำงานไม่เป็นระบบสัญญาณที่มากระตุ้นระบบ Watch Dog ก็จะหายไป และที่จุดนี้ระบบ Watch Dog ก็จะทำการสร้างสัญญาณเอาท์พุท 1 พัลส์เพื่อใช้เป็นสัญญาณรีเซทให้กับ CPU โดยอัตโนมัติ การทำงานของระบบ Watch Dog นี้แสดงได้ดังรูปที่ 7

หัวใจของระบบนี้ก็คือไอซี Retriggerable Monostable Multivibrator ในโครงงานนี้ใช้ U8 เบอร์ 74LS123 ซึ่งเป็น Monostable แบบคู่ในตัวเดียวกัน ส่วนสัญญาณที่ใช้กระตุ้นระบบ Watch Dog ได้มาจากสัญญาณเลือกพอร์ทหมายเลข 40-7FH ของ U5B ซึ่งให้พัลส์ลบแคบ ๆ 1 พัลส์เมื่อเราใช้คำสั่ง OUT (WATCHDOG),A แผนผังเวลา (Timing diagram) ของระบบ Watch Dog แสดงไว้ในรูปที่ 8



รูปที่ 7 การทำงานของระบบ Watch Dog



รูปที่ 8 แผนผังเวลาของระบบ Watch Dog

สัญญาณ $\overline{CS}$ (40H) ได้มาจาก Port Decoder U5B ช่วงเวลา T1 ของสัญญาณ $\overline{CS}$ เป็นช่วงเวลาที่ทำให้ระบบ Watch Dogเริ่มทำงานนั่นคือหากระบบไมโครโปรเซสเซอร์ทำงานปกติแล้ว ต้องให้สัญญาณ $\overline{CS}$ ที่มีคาบเวลาของพัลส์น้อยกว่า T1 เสมอระบบ Watch Dog จะไม่ทำงานให้เอาท์พุทเป็น HIGH (ลอจิก "1") ตลอด แต่ถ้าระบบเกิดการ HANG ขึ้นมาจะทำให้สัญญาณ CS มีคาบเวลามากกว่าช่วงเวลา T1 หรือขาดหายไปเลย (เป็น HIGH ตลอด) ระบบ Watch Dog จะทำงานและให้พัลส์ลบออกทางเอาท์พุทหนึ่งพัลส์ ซึ่งมีความกว้างของพัลส์มีค่าเท่ากับ T2 ใช้เป็นสัญญาณรีเซท CPU โดยความกว้างของสัญญาณรีเซทของ Z80 ต้องไม่น้อยกว่า 3 PERIOD ของ SYSTEM CLOCK ค่าของ T1 และ T2 ถูกกำหนดจาก R14, C7 และ R15, C8 สามารถคำนวณได้จากสูตรดังนี้

$t_w = K * R_T * C_{ext} * (1 + 0.7/R_T)$; K = 0.33 (74LS123)

โดยค่า t_w มีหน่วยเป็น nS (nanosecond)
R_T มีหน่วยเป็น กิโลโอห์ม (K)
C_{ext} มีหน่วยเป็น PF (picofarad) ต้องมีค่ามากกว่า 1000 PF

การใช้งานระบบ Watch Dog นี้ จะช่วยให้การทำงานของระบบไมโครโปรเซอร์มีเสถียรภาพที่ดีมาก แต่ทั้งนี้เทคนิคการเขียนโปรแกรมก็ค่อนข้างยุ่งยากยิ่งขึ้นด้วย กล่าวคือ เมื่อเกิดการรีเซทระบบ โปรแกรมก็ควรจะสามารถกลับไปทำงานล่าสุดได้ถูกต้อง เพราะฉะนั้นในการทำงานของโปรแกรมมอนิเตอร์จะต้องมีการเก็บสถานะต่าง ๆ ไว้ในหน่วยความจำเสมอ และในการเขียนโปรแกรมจำเป็นจะต้องแยกส่วนของโปรแกรมให้ชัดเจน และข้อสำคัญในการใช้งานะบบ Watch Dog คือระบบ Watch Dog จะเริ่มทำงานได้ก็ต่อเมื่อเริ่มมีการกระตุ้น (TRIG) และเมื่อทำการกระตุ้นแล้วจะหยุดไม่ได้ จะต้องกระตุ้นไปตลอดการทำงานของระบบ

ในโครงงานอิเล็กทรอนิกส์แบนเนอร์นี้เราจึงได้นำระบบ Watch Dog มาใช้ด้วย เพราะหากเครื่อง HANG แล้วที่แผงดิสเพลย์จะดับมืดไม่มีอะไรปรากฏเลย ต้องทำการรีเซทระบบตลอด ดังนั้นระบบ Watch Dog ที่กล่าวมานี้จึงสามารถช่วยได้โดยเมื่อเครื่อง HANG แล้วระบบ Watch Dog จะทำงานทันที เครื่องก็จะเริ่มทำงานใหม่โดยอัตโนมัติ

บทที่ 4
โครงสร้างทางซอฟท์แวร์ของอิเล็กทรอนิกส์แบนเนอร์

ซอฟท์แวร์ของระบบไมโครโปรเซอร์นับว่ามีส่วนสำคัญมาก เพราะหากขาดซอฟท์แวร์แล้วเครื่องก็ไม่สามารถทำงานได้ ซอฟท์แวร์ที่กล่าวถึงนี้ก็คือ โปรแกรมมอนิเตอร์ นั่นเอง

4.1 โปรแกรมมอนิเตอร์

โปรแกรมมอนิเตอร์ คือ โปรแกรมที่ควบคุมการทำงานทั้งหมดของอิเล็กทรอนิกส์แบนเนอร์ บรรจุอยู่ใน EPROM ขนาด 8 กิโลไบท์ (เบอร์ 2764) และในส่วนท้ายของ EPROM เก็บ CHARACTER GEN. ซึ่งเป็นส่วนของรูปแบบของตัวอักษร กำหนดไว้ 128 ตัวอักษรตามมาตราฐานรหัส ASCII โปรแกรมมอนิเตอร์นี้ทำการเขียนด้วยภาษาแอสเซมบลี้ของ CPU เบอร์ Z80

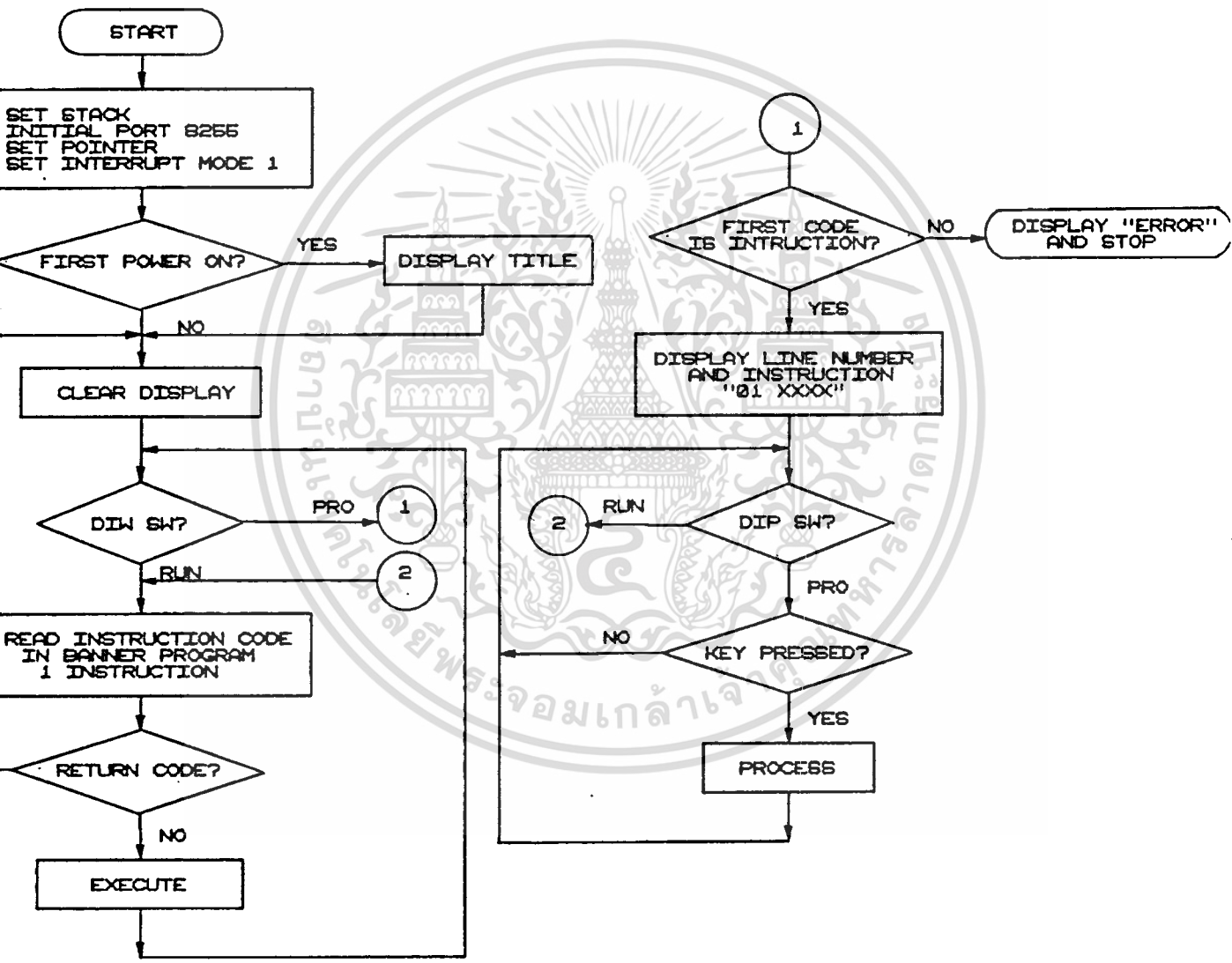
ลักษณะโครงสร้างของโปรแกรมมอนิเตอร์แบ่งออกเป็นส่วน ๆ ดังนี้

4.1.1 โปรแกรมหลัก (MAIN PROGRAM)

ในส่วนนี้โปรแกรมหลัก ๆ ของอิเล็กทรอนิกส์แบนเนอร์เขียนเป็น FLOW CHART ได้ดังรูปที่ 9 ประกอบไปด้วย 3 โปรแกรมหลักคือ

- SYSTEM RESET เป็นโปรแกรมเริ่มแรกของระบบ ทุกครั้งที่เริ่มเปิดเครื่องหรือกดคีย์รีเซท เครื่องจะมาทำงานที่โปรแกรมนี้ก่อนโดยโปรแกรมนี้จะทำการเซทค่าต่าง ๆ ที่จำเป็นเช่น เซทค่าสแตค , โปรแกรมพอร์ทและเซทโหมดการอินเตอร์รัพท์ นอกจากนั้นจะมีการเช็คดูว่า เป็นการเปิดเครื่องครั้งแรกหรือไม่ เมื่อโปรแกรมทำงานต่าง ๆ เรียบร้อยแล้วจะไปทำโปรแกรมย่อยเพื่อเช็คดูว่า ดิพสวิทช์ PRO/RUN อยู่ในตำแหน่งใด แล้วกระโดดไปทำงานในโหมดนั้น ๆ
- PROGRAMMING MODE เป็นโปรแกรมที่ใช้สำหรับการโปรแกรมคำสั่งที่ใช้กำหนดตัวอักษรและรูปภาพกราฟฟิค และรูปแบบการแสดงผลที่ปรากฏที่แผงดิสเพลย์ โดยมีลักษณะการโปรแกรมที่เป็นลำดับคล้ายกับการโปรแกรมภาษาคอมพิวเตอร์ซึ่งจะประกอบไปด้วย เลขบรรทัดและคำสั่ง (คำสั่งที่ใช้ในอิเลคทรอนิคส์แบนเนอร์นี้ถูกกำหนดขึ้นมาโดยเฉพาะ)
- RUN MODE เป็นโปรแกรมที่ทำหน้าที่นำคำสั่งต่าง ๆ ในแบนเนอร์โปรแกรมที่ได้ทำการโปรแกรมไว้แล้วไปตีความหมาย แล้วนำไปกระทำตามคำสั่งนั้น ๆ การทำงานจะเริ่มรันตั้งแต่คำ

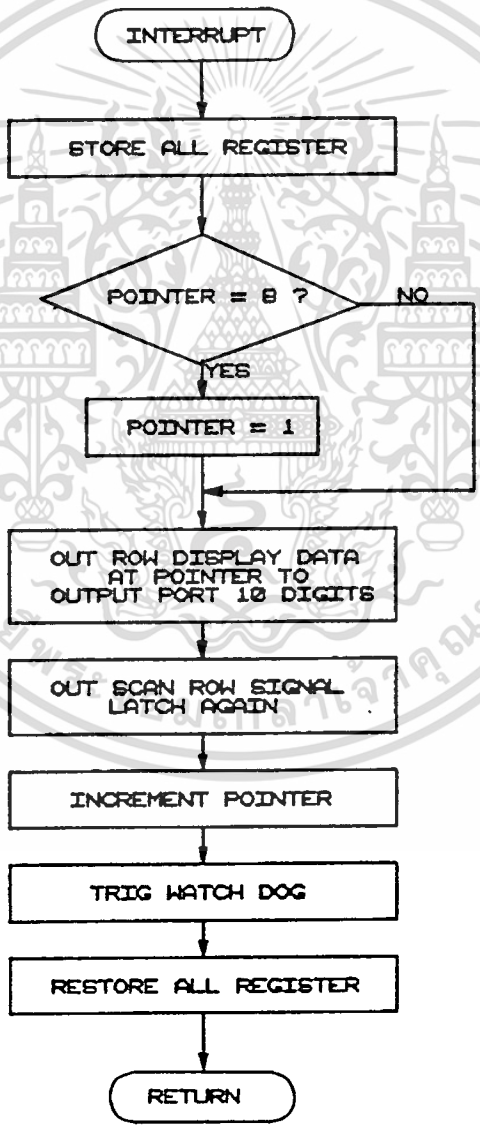
สั่งแรกไปจนถึงคำสั่งสุดท้าย แล้ววนกลับไปทำโปรแกรม
เดียวกันนี้ใหม่วนอยู่อย่างนี้เรื่อยไปจนกว่าจะเปลี่ยนโหมด



รูปที่ 9 FLOW CHART ของ MAIN PROGRAM

4.1.2 โปรแกรมตอบสนองการอินเตอร์รัพท์ (INTERRUPT SERVICE ROUTINE PROGRAM)

ใน MAIN PROGRAM กำหนดให้ Z80 ทำการอินเตอร์รัพท์ในโหมด 1 ซึ่งโปรแกรมนี้จะถูกใช้งานก็ต่อเมื่อ CPU ได้รับสัญญาณอินเตอร์รัพท์ CPU จะหยุดการทำงานจาก MAIN PROGRAM ชั่วขณะแล้วกระโดดไปที่แอดเดรส 0038Hเพื่อทำงานตามโปรแกรมตอบสนองการอินเตอร์รัพท์ทำการสแกนดิสเพลย์ และเมื่อ CPU ทำงานในโปรแกรมนี้จนจบแล้ว CPU ก็จะกลับไปทำคำสั่งใน MAIN PROGRAM ตามเดิม เขียน FLOW CHART แสดงได้ดังรูปที่ 10



รูปที่ 10 FLOW CHART ของ INTERRUPT SERVICE ROUTINE PROGRAM

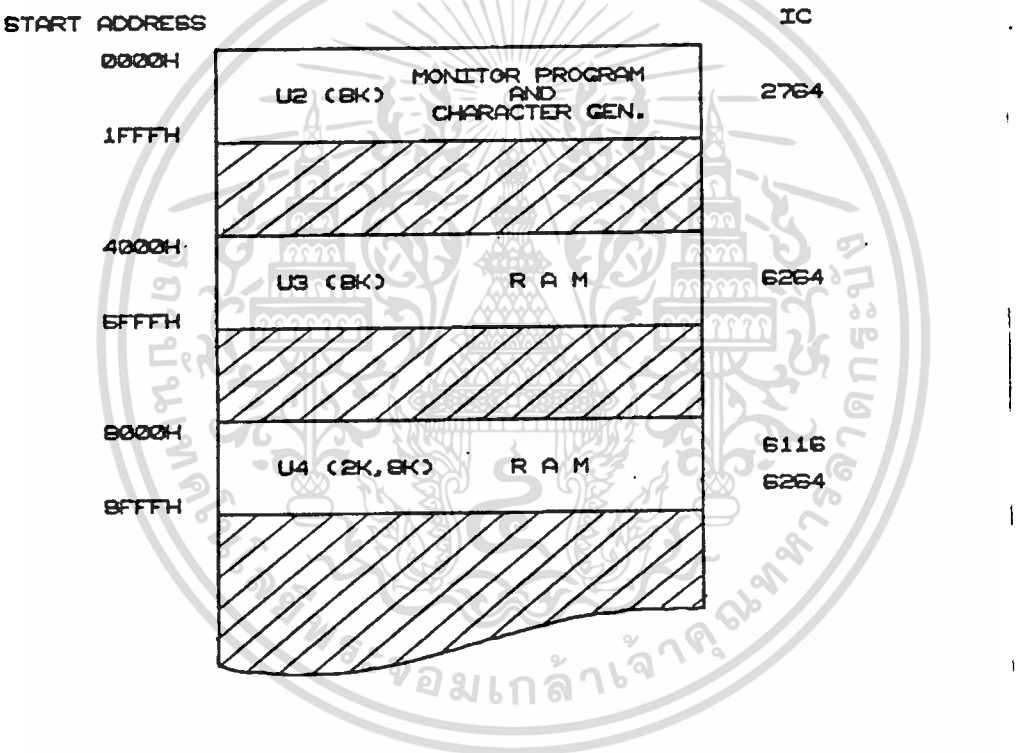
4.1.3 โปรแกรมซับรูทีน (SUBROUTINE PROGRAM)

เป็นโปรแกรมย่อยหลายโปรแกรมที่มีการทำงานเฉพาะเจาะจง และถูกเรียกใช้งานบ่อย ๆ โดยผ่านคำสั่ง CALL ที่ถูกเรียกจากโปรแกรมหลักและโปรแกรมตอบสนองการอินเตอร์รัพท์ โปรแกรมซับรูทีนที่เขียนขึ้นนี้แบ่งออกได้เป็น 2 กลุ่มคือ

- กลุ่มทั่วไป เป็นกลุ่มที่มีการทำงานทั่ว ๆ ไป เช่น การหน่วงเวลา, สแกนคีย์บอร์ดที่ใช้ในการโปรแกรมแบนเนอร์, สแกนคีย์บอร์ดที่ใช้ในการ EDIT ตัวอักษรและรูปภาพกราฟฟิค, การเช็คดิพสวิทช์, การป้อนตัวอักษรและรูปภาพกราฟฟิค, การนำเอาคำสั่งแบนเนอร์ไปรัน, การอ่านค่า CODEของคำสั่งจากแบนเนอร์โปรแกรม
- กลุ่มโยกย้ายข้อมูล เป็นกลุ่มที่ทำหน้าที่เคลื่อนย้าย แปลงค่า และค้นหาข้อมูลต่าง ๆ เช่นการแปลง CODE ของคำสั่งเป็นรหัสASCII, การแปลงเลขบรรทัดเป็นรหัส ASCII, การดิสเพลย์โปรแกรม, การกำหนดจำนวนข้อมูลของคำสั่ง,การแทรกและการลบคำสั่ง, การหาตำแหน่งสุดท้ายของโปรแกรม และการเคลียร์โปรแกรมแบนเนอร์
- กลุ่มคำสั่งแบนเนอร์ เป็นกลุ่มของคำสั่งแบนเนอร์ (BANNER INSTRUCTION SET) ที่กำหนดขึ้นมาเองมีทั้งหมด 13 คำสั่งเช่นCLEAR, TEXT SCREEN, TEXT LINE, GRAPHIC, DISPLAY, LOAD, DOOR, SHIFT, DELAY, FLASH, INVERSE, ROTATE และ RETURN

4.2 การจัดหน่วยความจำ

การจัดหน่วยความจำ โดยดูจากวงจรในรูปที่ 4 จะเห็นได้ว่า เราได้ทำการถอดรหัสแอดเดรสของหน่วยความจำไว้ 4 บล๊อก ๆ ละ 8 กิโลไบท์ ซึ่งในการจัดหน่วยความจำทั้ง ROM และ RAM ในบางกรณีไม่จำเป็นจะต้องให้แอดเดรสของหน่วยความจำแต่ละตัวต่อเนื่องกันก็ได้ เพียงแต่ว่าในการเขียนโปรแกรมเราสามารถอ้างตำแหน่งแอดเดรสของหน่วยความจำนั้นได้ถูกต้องก็เพียงพอแล้ว ในรูปที่ 11 แสดงถึงการจัดหน่วยความจำทั้ง ROM และ RAM ของอิเล็กทรอนิกส์แบนเนอร์ เราสามารถเลือก RAM U4 ได้ 2 เบอร์คือ 6116 และ6264 ดูได้จากวงจรในรูปที่ 4

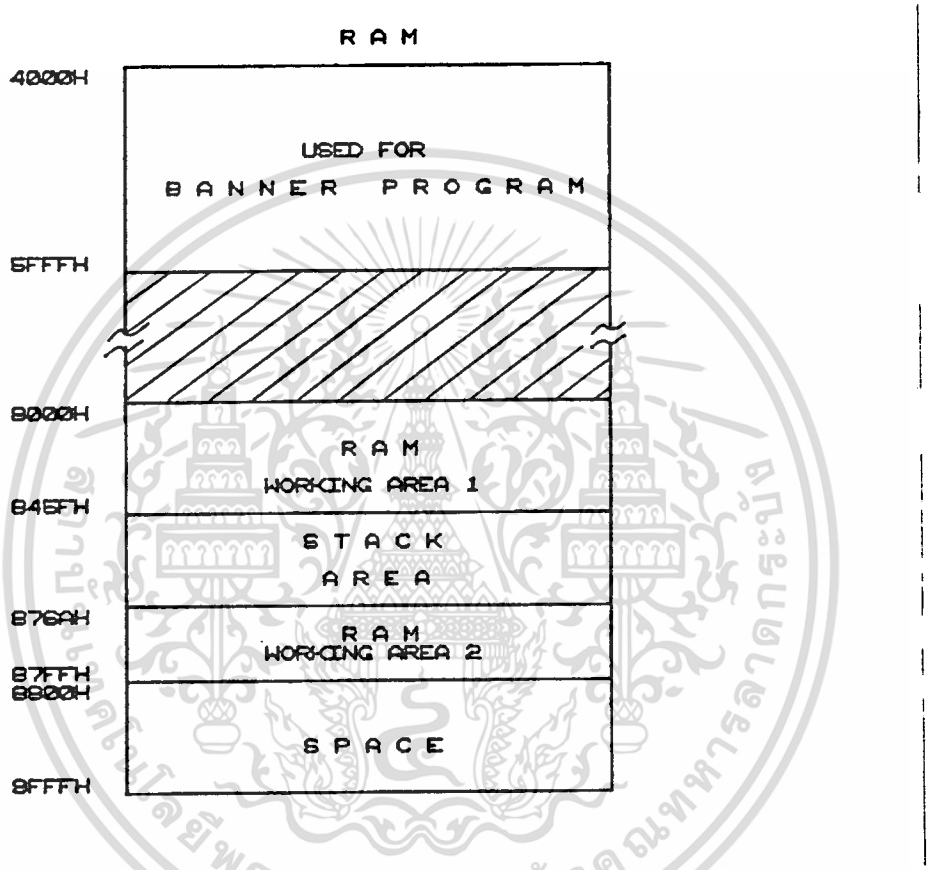


รูปที่ 11 การจัดหน่วยความจำทั้งหมด

ในการทำงานของโปรแกรมมอนิเตอร์นั้น จำเป็นที่จะต้องใช้ RAM บางส่วนสำหรับเก็บค่าข้อมูลต่าง ๆ รวมทั้งเป็นใช้เป็นบัฟเฟอร์สำหรับถ่ายทอดข้อมูลระหว่างอุปกรณ์ INPUT และ OUTPUT ซึ่ง RAM ในส่วนนี้เราเรียกรวมว่า RAM WORKING AREA สำหรับอิเล็กทรอนิกส์แบนเนอร์ได้ทำการจัด RAM เพื่อใช้งานในส่วนนี้ไว้ 2 ช่วง ช่วงแรกอยู่ที่แอดเดรส 8000H-845FH และช่วงหลังอยู่ที่แอดเดรส 876AH-87FFH

และ RAM อีกส่วนหนึ่งซึ่งจะขาดไม่ได้ก็คือ ส่วนของสแตค (STACK) ส่วนของสแตคนี้ใช้สำหรับเก็บตำแหน่งแอดเดรสของ PC (PROGRAM COUNTER)ซึ่งจะถูกใช้โดยผ่านคำสั่ง

CALL และใช้เก็บค่าของรีจิสเตอร์ (REGISTER) อื่น ๆ โดยผ่านคำสั่ง PUSH ในอิเล็กทรอนิกส์แบนเนอร์กำหนดส่วนของสแตคไว้ที่ตำแหน่งแอดเดรส 8460H-8769H ค่าของสแตคพอยเตอร์ (STACK POINTER : SP) ถูกกำหนดไว้ที่แอดเดรส 876AH การจัด RAM U3 และ U4 แสดงดังรูปที่ 12



รูปที่ 12 การจัด RAM ของ U3 และ U4

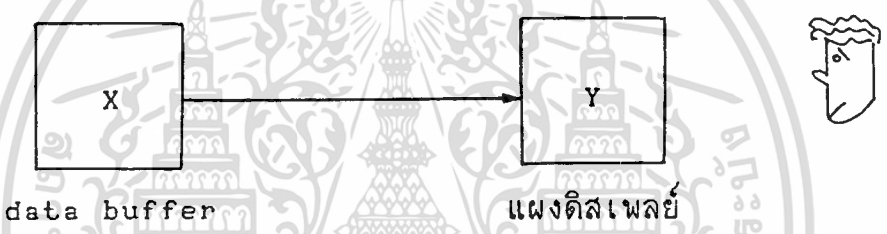
ส่วนของ RAM U4 สามารถเลือกใช้ RAM เบอร์ 6116 หรือ 6264 ได้ ซึ่งหากใช้ RAM 6264 ในตำแหน่งแอดเดรสส่วนที่เกินจากคือตำแหน่งแอดเดรสที่ 8800-9FFFH จะไม่ถูกใช้ ที่ทำเช่นนี้ไว้เพราะต้องการสำรองไว้สำหรับการขยายระบบในอนาคตและยังสามารถเลือกใช้ RAM ได้หลายเบอร์ด้วย

บทที่ 5
หลักในการโปรแกรมและชุดคำสั่งแบนเนอร์

เพื่อให้ง่ายต่อการกำหนดรูปแบบ และลักษณะการแสดงผลของอิเล็กทรอนิกส์แบนเนอร์ เราจึงได้กำหนดชุดคำสั่งแบนเนอร์ขึ้นมา เพื่อให้ง่ายต่อการโปรแกรมและแก้ไข ก่อนที่จะกล่าวถึงชุดคำสั่งแบนเนอร์ เรามาทำความเข้าใจกับหลักในการโปรแกรมของอิเล็กทรอนิกส์กันก่อน

5.1 หลักในการโปรแกรม (CONCEPT OF PROGRAMMING)

ในอิเล็กทรอนิกส์แบนเนอร์มีหน่วยความจำอยู่ 2 ส่วนที่ผู้ใช้ควรทราบไว้ ส่วนแรกคือ DATA BUFFER (X) และส่วนที่สองคือ DISPLAY BUFFER (Y)ซึ่งก็คือแผงดิสเพลย์นั่นเอง



รูปที่ 13 หลักในการโปรแกรมของอิเล็กทรอนิกส์แบนเนอร์

หลักในการโปรแกรมมีอยู่ว่า ให้ทำการเตรียมข้อมูลตัวอักษรหรือรูปภาพกราฟฟิคไว้ใน X ให้เรียบร้อยเสียก่อน และเมื่อต้องการแสดงผลออกทางแผงดิสเพลย์ก็ให้ทำการเคลื่อนย้ายข้อมูลจาก X ไปไว้ใน Y โดยที่ค่าใน X ไม่เปลี่ยนแปลง ซึ่งคำสั่งแบนเนอร์ที่กำหนดขึ้นมานี้มีคำสั่งทั้งที่กระทำกับ X, การนำค่าจาก X ไปไว้ใน Y (X -> Y) และกระทำกับ Y

5.2 ชุดคำสั่ง BANNER (BANNER INSTRUCTION SET)

ชุดคำสั่งแบนเนอร์ เป็นคำสั่งที่ถูกกำหนดขึ้นมาเพื่อใช้กับอิเล็กทรอนิกส์แบนเนอร์โดยเฉพาะ ซึ่งประกอบไปด้วยคำสั่งทั้งหมด 13 คำสั่งดังนี้

คำสั่ง	ความหมาย	การทำงาน
CL	CLear screen	X
TS-	Text Screen	X
TL-	Text Line	X -> Y
GR-	GRaphic	X

DI	DIsplay	X -> Y
LD,d	LoaD	X -> Y
DR,d	DooR	X -> Y
SH,d	SHift	X -> Y
DL	DeLay	-
FS	FlaSh	Y
IN	INverse	Y
RT,d	RoTate	Y
RE	REturn	loop back

หมายเหตุ d : direction UP ^ , DOWN v , LEFT < , RIGHT >

<u>คำอธิบายคำสั่ง</u>

คำสั่ง CL เป็นคำสั่งที่ทำหน้าที่เคลียร์ X

คำสั่ง TS- เป็นคำสั่งที่ใช้กำหนดตัวอักษรได้ 10 ตัวอักษร โดยทำการเก็บไว้ที่ X

คำสั่ง TL- เป็นคำสั่งที่ใช้กำหนดตัวอักษรได้ถึง 100 ตัวอักษรโดยทำการเก็บไว้ที่ X และทำการแสดงผลที่แผงดิสเพลย์ (Y) โดยวิ่งจากขวาไปซ้ายจนครบทุกตัวอักษรที่ทำการป้อนเข้าไป

คำสั่ง GR- เป็นคำสั่งที่ใช้กำหนดรูปภาพกราฟฟิคซึ่งสามารถสร้างภาพหรือตัวอักษรได้ตามต้องการ มีความละเอียดของภาพขนาด 8 * 80 จุด รูปภาพที่ป้อนเข้าไปนี้จะเก็บไว้ที่ X

คำสั่ง DI เป็นคำสั่งที่นำเอารูปแบบตัวอักษรหรือรูปภาพกราฟฟิคที่กำหนดไว้ใน X ไปแสดงที่แผงดิสเพลย์ (Y) โดยที่ค่าใน X ไม่เปลี่ยนแปลง

คำสั่ง LD,d เป็นคำสั่งที่นำเอารูปแบบตัวอักษรหรือรูปภาพกราฟฟิคที่กำหนดไว้ใน X ไปแสดงที่แผงดิสเพลย์ (Y) ทีละแถวหรือทีละคอลัมท์ โดยสามารถกำหนดทิศทางการแสดงผลได้ 4 ทิศทางคือ up ^, down v, left <- และ right ->

คำสั่ง DR,d เป็นคำสั่งที่นำเอารูปแบบตัวอักษรหรือรูปภาพกราฟฟิคที่กำหนดไว้ใน X ไปแสดงที่แผงดิสเพลย์ (Y) ในลักษณะคล้ายกับเป็นการเปิดปิดประตู ซึ่งสามารถกำหนดทิศทางได้เช่นเดียวกับคำสั่ง LD,d

คำสั่ง SH,d เป็นคำสั่งที่นำเอารูปแบบตัวอักษรหรือรูปภาพกราฟฟิคที่กำหนดไว้ใน X ไปแสดงที่แผงดิสเพลย์ (Y) ในลักษณะคล้ายกับเป็นการเลื่อนตัวอักษรหรือเลื่อนภาพ ซึ่งสามารถกำหนดทิศทางได้เช่นกัน

คำสั่ง DL เป็นคำสั่งใช้ในการหน่วงเวลา โดยจะทำการหน่วงเวลาประมาณหนึ่ง

วินาที คำสั่งนี้จะไม่มีผลต่อค่าของ X และ Y

คำสั่ง FS เป็นคำสั่งที่ใช้ในการกระพริบสิ่งที่แสดงอยู่ที่แผงดิสเพลย์ ซึ่งคล้ายกับการกระพริบของแฟลช คำสั่งนี้ไม่มีผลต่อค่าของ X

คำสั่ง IN เป็นคำสั่งที่ทำการ COMPLEMENT จุดทุกจุดของภาพที่แสดงอยู่ที่แผงดิสเพลย์ (Y) กล่าวคือ คล้ายกับการทำภาพ NEGATIVE คำสั่งนี้ไม่มีผลต่อค่าของ X

คำสั่ง RT,d เป็นคำสั่งที่ใช้ในการหมุนสิ่งที่แสดงอยู่ที่แผงดิสเพลย์ (Y) ซึ่งสามารถกำหนดทิศทางได้เช่นเดียวกับคำสั่ง LD,d คำสั่งนี้ไม่มีผลต่อค่าของ X

คำสั่ง RE เป็นคำสั่งที่ให้เครื่องกลับรันโปรแกรมแบนเนอร์ใหม่ โดยจะทำการรันในคำสั่งแรกต้นโปรแกรม และเมื่อเครื่องทำการรันจนถึงคำสั่งสุดท้ายของโปรแกรมซึ่งก็คือ คำสั่ง RE เครื่องก็จะกลับไปรันโปรแกรมใหม่ คำสั่งนี้ไม่ต้องทำการคีย์เข้าเครื่อง เพราะเครื่องจะทำการใส่คำสั่งนี้ให้โดยอัตโนมัติ

บทที่ 6
การใช้งานเบื้องต้น

เมื่อเริ่มใช้อิเล็กทรอนิกส์แบนเนอร์ โดยเสียบปลั๊กเปิดไฟเข้าเครื่องที่แผงดิสเพลย์จะปรากฏข้อความไตเติ้ล " Electronic Banner by Pat 88. KMITL 30.3315 " วิ่งจากขวาไปซ้าย หลังจากนั้นการทำงานของอิเล็กทรอนิกส์แบนเนอร์จะเข้าไปอยู่ในโหมดการทำงาน

6.1 โหมดการทำงาน

โหมดการทำงานของอิเล็กทรอนิกส์แบนเนอร์แบ่งออกได้ 2 โหมดคือ

6.1.1 โหมดโปรแกรม (PROGRAMMING MODE) อิเล็กทรอนิกส์แบนเนอร์จะเข้ามาทำงานในโหมดนี้ก็ต่อเมื่อ ดิพสวิทช์ตัวที่สอง(PRO/RUN) ที่แผงคีย์บอร์ดอยู่ที่ตำแหน่ง "PRO" (ดิพสวิทช์ ON) โหมดนี้ใช้สำหรับการโปรแกรมคำสั่งแบนเนอร์ มีลักษณะการโปรแกรมเป็นลำดับคล้ายกับการโปรแกรมคอมพิวเตอร์ภาษาเบสิค ซึ่งประกอบไปด้วย เลขบรรทัด (LINE NUMBER) และคำสั่ง (INSTRUCTION)

- ในกรณีที่ยังไม่ได้ทำการโปรแกรม ที่แผงดิสเพลย์จะแสดงเลขบรรทัดพร้อมกับคำสั่งแรกของโปรแกรมซึ่งก็คือ คำสั่ง RE ดังนี้

แผงดิสเพลย์

01 RE

แต่ถ้าที่แผงดิสเพลย์แสดงตัวอักษรคำว่า "Error" แสดงว่าเราทำการเปิดเครื่องโดยไม่ได้ ON ดิพสวิทช์ตัวแรก ($\overline{WE}$) ที่แผงคีย์บอร์ด (ในการเปิดเครื่องครั้งแรก เครื่องจะทำการเคลียร์หน่วยความจำที่ใช้เก็บโปรแกรมแบนเนอร์ เพื่อให้เครื่องพร้อมที่จะทำการโปรแกรมได้) ดังนั้นต้องถอดปลั๊กไฟเพื่อปิดไฟเข้าเครื่อง แล้วเลื่อนดิพสวิทช์ตัวแรกนี้ให้อยู่ในตำแหน่ง "ON" รอสักครู่ ค่อยเสียบปลั๊กเพื่อจ่ายไฟเข้าเครื่องใหม่

ต่อจากนี้ไปก็สามารถทำการโปรแกรมโดยป้อนคำสั่งต่าง ๆ เข้าไปเก็บเป็นโปรแกรมแบนเนอร์ได้ ซึ่งแบ่งออกเป็น 2 โปรแกรมคือ โปรแกรม A และโปรแกรม B โปรแกรมละ 99 บรรทัดโดยเลือกจากดิพสวิทช์ที่อยู่บนแผงควบคุม (CONTROL BOARD) และที่สำคัญในการป้อนโปรแกรมคำสั่งเข้าไปใหม่นั้นจะต้องทำการเลื่อนดิพสวิทช์ตัวแรก($\overline{WE}$)ที่แผงคีย์บอร์ดไว้ที่ตำแหน่ง "ON" เสมอ และเมื่อทำการโปรแกรมเรียบร้อยแล้วก็ควรเลื่อนดิพสวิทช์ตัวเดียวกันนี้ไว้ที่ตำแหน่ง "OFF" เพื่อป้องกันการสูญหายอันเกิดจากการโปรแกรมผิดพลาดและใช้ในกรณีที่ต้องการเก็บโปรแกรมไว้ (BACK UP)

- หากได้ทำการโปรแกรมไว้แล้ว ที่แผงดิสเพลย์จะแสดงเลขบรรทัดพร้อมกับคำสั่งแรกของโปรแกรมดังนี้

แผงดิสเพลย์

01 XXXX

และถ้าต้องการแก้ไขหรือทำโปรแกรมใหม่ให้ทำการปรับดิพสวิทช์ตัวที่สอง ($\overline{WE}$) ให้อยู่ที่ตำแหน่ง "ON"

6.1.2 <u>โหมดรัน</u> (RUN MODE) อิเล็กทรอนิกส์แบนเนอร์จะทำงานในโหมดนี้เมื่อดิพสวิทช์ตัวที่สองที่แผงคีย์บอร์ดอยู่ที่ตำแหน่ง "RUN"(ดิพสวิทช์ OFF) โหมดนี้จะนำเอาคำสั่งในโปรแกรมแบนเนอร์ที่ได้โปรแกรมไว้แล้วไปทำงานตามคำสั่งนั้น ๆ โดยผลของโปรแกรมจะแสดงที่แผงดิสเพลย์

- ในกรณีที่ยังไม่ได้ทำการโปรแกรม ที่แผงดิสเพลย์จะมืด ไม่มีอะไรปรากฏ ต้องกลับไปโหมดโปรแกรมเพื่อทำการโปรแกรมคำสั่งเสียก่อน

- หากได้ทำการโปรแกรมไว้แล้ว อิเล็กทรอนิกส์แบนเนอร์ก็จะทำการอ่านคำสั่งที่ได้ทำการโปรแกรมไว้ในแบนเนอร์โปรแกรมแล้วนำไปปฏิบัติตามคำสั่งนั้น ๆ ตั้งแต่คำสั่งแรกจนถึงคำสั่งสุดท้ายแล้ววนกลับไปทำคำสั่งแรกที่ต้นโปรแกรม เรื่อยไป ผลที่ได้จากการรันโปรแกรมจะแสดงผลที่แผงดิสเพลย์และสามารถรันโปรแกรมได้ทั้งโปรแกรม A และโปรแกรม B เช่นเดียวกับโหมดโปรแกรม

6.2 <u>ส่วนของคีย์บอร์ดและดิพสวิทช์</u>

อิเล็กทรอนิกส์แบนเนอร์มีการจัดคีย์บอร์ดและดิพสวิทช์เป็นไปตามรูปที่ 14 ซึ่งส่วนของคีย์บอร์ดทั้ง 24 คีย์จะถูกใช้ในโหมดโปรแกรมเท่านั้น สามารถแยกออกเป็นกลุ่ม ๆ ได้ดังนี้

- คีย์คำสั่งแบนเนอร์ 12 คีย์
- คีย์ควบคุม 12 คีย์
- คีย์ตัวอักษรและตัวอักษรกราฟฟิค 96 ตัวอักษร โดยคีย์ตัวอักษรเหล่านี้จะอยู่บนคีย์คำสั่งแบนเนอร์

6.2.1 <u>คีย์คำสั่งแบนเนอร์</u>

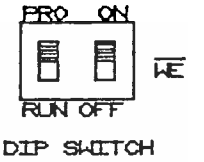
เป็นคีย์ที่ใช้ป้อนคำสั่งแบนเนอร์มีทั้งหมด 12 คีย์คือ

CL	TS-	TL-	GR-

DI	LD,	DR,	SH,	DL	FS	IN	RT,

ABCD CL abcd	EFGH DI efgh	IJKL DL ijkl	EDIT	INS	RESET
MNOP TS- mnop	QRST LD, qrst	UVWX FS uvwx	CLEAR ALL	DEL	LINE INC
YZ01 TL- yz()	2345 DR, +-*/	6789 IN .,<>	CHN	^	LINE DEC
!:; GR- =\?@	♥♦♣♠ SH, □{}	↑↓←→ RT, #$%&	<	V	>

KEYBOARD 24 KEYS (USED FOR PROGRAM MODE ONLY)



DIP SWITCH

รูปที่ 14 การจัดคีย์บอร์ดและดิพสวิทช์

6.2.2 คีย์ควบคุม

1. คีย์ RESET

ใช้สำหรับรีเซทระบบทั้งหมด ลักษณะเป็นฮาร์ดแวร์รีเซทต่อตรงเข้าระบบ โดยเมื่อกดคีย์รีเซทแล้วอิเล็กทรอนิกส์แบนเนอร์จะกลับไปเริ่มต้นทำงานใหม่ เช่น ถ้าอยู่ในโหมดโปรแกรมเครื่องก็จะแสดงคำสั่งแรกของโปรแกรมเพื่อทำการโปรแกรมต่อไป และถ้าอยู่ในโหมดรันเครื่องก็จะเริ่มรันโปรแกรมนั้นใหม่

2. คีย์ EDIT

ใช้สำหรับการป้อนและแก้ไขตัวอักษรในลักษณะข้อความ (TEXT) และรูปภาพกราฟฟิค คีย์นี้จะใช้กับคำสั่ง TS- TL- และ GR- เท่านั้น และเมื่อแก้ไขเรียบร้อยแล้วให้กดคีย์นี้อีกครั้งเพื่อทำการป้อนคำสั่งต่อไป รายละเอียดการใช้งานของคีย์นี้จะกล่าวในบทที่ 7

3. คีย์ CLEAR ALL

เป็นคีย์ที่ใช้สำหรับเคลียร์โปรแกรมแบนเนอร์ในขณะที่ทำโปรแกรมอยู่ โดยทำการเคลียร์ทั้งโปรแกรม เมื่อกดคีย์นี้ที่แผงดิสเพลย์จะแสดงข้อความว่า "Confirm" เพื่อให้ผู้ใช้ยืนยันว่าจะทำการเคลียร์โปรแกรม หากไม่ต้องการเคลียร์ก็ให้กดคีย์อื่นเพื่อทำการโปรแกรมต่อไป แต่ถ้าแน่ใจแล้วว่าต้องการเคลียร์โปรแกรมก็ให้กดคีย์นี้อีกครั้ง เครื่องก็จะทำการเคลียร์โปรแกรมให้แล้วกลับไปเริ่มต้นเพื่อทำการป้อนโปรแกรมใหม่

นอกจากคีย์นี้จะใช้ในการเคลียร์โปรแกรมแล้ว ยังถูกใช้ในการเคลียร์ตัวอักษรและรูปภาพกราฟฟิคในขณะที่ทำการ EDIT คำสั่ง TS- TL- และ GR- อีกด้วย

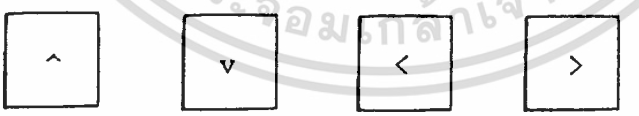
4. คีย์ CHN

CHN ย่อมาจากคำว่า CHANGE คีย์นี้ใช้ในการ EDIT ข้อความตัวอักษร(TEXT)และรูปภาพกราฟฟิค (GRAPHIC) โดยใช้ร่วมกับ ARROW KEYS แยกการทำงานของคีย์นี้ได้ 2 ส่วนดังนี้

- ใช้ในการ EDIT คำสั่ง TS- และ TL- ตัวอักษรที่ป้อนเข้าไปในคำสั่งทั้งสองนี้ใช้คีย์ตัวอักษรที่อยู่บนคีย์คำสั่งแบนเนอร์ มีทั้งหมด 96 ตัวอักษร โดยคีย์นี้จะทำหน้าที่คล้ายกับคีย์ CAP LOCKS ในคีย์บอร์ดคอมพิวเตอร์คือ ใช้เลือกตัวอักษรตัวเล็กและตัวใหญ่

- ใช้ในการ EDIT คำสั่ง GR- คีย์นี้ใช้กำหนดให้จุดใด ๆบนภาพกราฟฟิคสว่างหรือมืดได้ ซึ่งจุดสว่างและจุดมืดเหล่านี้จะก่อให้เกิดภาพต่าง ๆ กันได้ตามต้องการ การทำงานของคีย์นี้เป็นแบบสลับไปมา (TOGGLE) คือกดติด (สว่าง) และกดดับ (มืด)

5. กลุ่ม ARROW KEYS มีอยู่ด้วยกัน 4 คีย์ คือ UP, DOWN, LEFT และ RIGHT



แยกการทำงานของคีย์นี้ได้เป็น 2 ส่วนดังนี้

- ใช้กำหนดทิศทางการแสดงผลของคำสั่ง LD,d DR,d SH,d และ RT,d ซึ่งสามารถกำหนดได้ 4 ทิศทาง

- ใช้ในการ EDIT ข้อความตัวอักษรและรูปภาพกราฟฟิคเป็นคีย์ที่ใช้ย้ายตำแหน่งของ CURSOR ไปในตำแหน่งที่ต้องการ การ EDIT ตัวอักษรในคำสั่ง TS- และ TL- จะใช้อยู่ 2 คีย์คือ LEFT และ RIGHT ส่วนการ EDIT รูปภาพกราฟฟิคในคำสั่ง GR- จะใช้ทั้งหมด 4 คีย์คือ UP, DOWN, LEFT และ RIGHT

6. คีย์ [LINE INC] และ [LINE DEC]

เป็นคีย์ที่ใช้ในการเพิ่มหรือลดบรรทัดของคำสั่งในโปรแกรมแบนเนอร์

7. คีย์ [INS] และ [DEL]

INS ย่อมาจาก INSERT และ DEL ย่อมาจาก DELETE คีย์ทั้งสองนี้ใช้ในการสอดแทรกและตัดทิ้งคำสั่งในโปรแกรมแบนเนอร์ โดยจะมีประโยชน์มากในระหว่างการทำโปรแกรม เช่น ถ้าต้องการ INSERT คำสั่งเข้าไปในระหว่างช่วงของโปรแกรมหรือ DELETE คำสั่งบางส่วนออกในช่วงของโปรแกรม

ในการใช้งานเมื่อต้องการ INSERT ให้ทำการเลื่อนเลขบรรทัดไปยังบรรทัดของคำสั่งที่ต้องการแทรกก่อนหน้าคำสั่งนั้นโดยใช้คีย์ [LINE INC] และ [LINE DEC] ซึ่งจะแทรกโดยกดคีย์ [INS] ที่แผงดิสเพลย์จะแสดงเลขบรรทัดที่จะทำการแทรกคำสั่งเข้าไป หลังจากนั้นให้กดคีย์คำสั่งที่ต้องการแทรก ที่แผงดิสเพลย์ก็จะแสดงเลขบรรทัดพร้อมกับคำสั่งที่ได้แทรกเข้าไปแล้วและจะเลื่อนคำสั่งที่เหลือออกไป 1 บรรทัด การ INSERT จะกระทำได้ทีละ 1 คำสั่งเท่านั้น

ตัวอย่าง ถ้าเรามีโปรแกรมแบนเนอร์เดิมอยู่และต้องการแทรกคำสั่งเข้าไป 1 คำสั่ง โดยให้คำสั่งที่อยู่ถัดไปเลื่อนออกไป 1 บรรทัด

เลขบรรทัด	คำสั่ง
01	TS-
02	DL
03	RE

คำสั่งที่ต้องการแทรกเข้าไปคือ คำสั่ง DI โดยจะแทรกก่อนคำสั่ง DL ให้อยู่หลังคำสั่ง TS- ให้ทำตามขั้นตอนดังต่อไปนี้

กดปุ่ม	แผงดิสเพลย์
	02 DL
[INS]	02
[DI]	02 DI
	03 RE

ดังนั้นจะได้โปรแกรมที่ทำการ INSERT แล้วเป็นดังนี้

เลขบรรทัด	คำสั่ง
01	TS-
02	DI
03	DL
04	RE

และเมื่อต้องการ DELETEให้ทำการเลื่อนเลขบรรทัดไปยังบรรทัดของคำสั่งที่ต้องการตัดทิ้งทำการกดคีย์ DEL ที่แผงดิสเพลย์จะแสดงเลขบรรทัดเดิมและคำสั่งของบรรทัดถัดไปที่ขึ้นมาแทนบรรทัดเดิม การกดคีย์นี้ 1 ครั้งเท่ากับเป็นการ DELETE คำสั่ง 1 คำสั่ง

ตัวอย่าง จากโปรแกรมแบนเนอร์ที่ได้ทำการ INSERT ในตัวอย่างที่แล้ว ถ้าต้องการตัดคำสั่งทิ้ง 1 คำสั่ง โดยให้คำสั่งถัดไปขึ้นมาแทนที่เดิมที่ลบออกไป

เลขบรรทัด	คำสั่ง
01	TS-
02	DI
03	DL
04	RE

คำสั่งที่ต้องการตัดทิ้งคือ คำสั่ง DL ให้ทำตามขั้นตอนดังต่อไปนี้

กดปุ่ม	แผงดิสเพลย์
	03 DL
DEL	03 RE

ดังนั้นจะได้โปรแกรมที่ทำการ DELETE แล้วเป็นดังนี้

เลขบรรทัด	คำสั่ง
01	TS-
02	DI
03	RE

6.2.3 คีย์ตัวอักษรและตัวอักษรกราฟฟิค

คีย์ตัวอักษรและตัวอักษรกราฟฟิคอยู่บนคีย์คำสั่งแบนเนอร์ ซึ่งมีทั้งหมด 12 คีย์ ๆ ละ 8 ตัวอักษร รวมทั้งหมด 96 ตัวอักษร สามารถเลือกตัวอักษรตัวเล็กหรือตัวใหญ่โดยใช้คีย์ CHN และเมื่อต้องการตัวอักษรถัดไปที่อยู่บนคีย์เดียวกันก็ให้กดคีย์เดิมอีก หากกดคีย์

เดิมนี้ต่อไปจนสุดตัวอักษรตัวที่ 4 ที่อยู่บนคีย์นี้แล้วก็จะวนกลับมาที่ตัวอักษรตัวเดิมอีก เช่นคีย์ ABCD ถ้าต้องการตัวอักษร "C" ให้กดคีย์นี้ 2 ครั้ง และเมื่อกดคีย์นี้ต่อไปจนถึงตัวอักษร "D" แล้วจะวนกลับไปเป็นตัวอักษรตัวเดิมคือ ตัวอักษร "A" อีก (A->B->C->D->A->B-> .....) และถ้าต้องการตัวอักษร "b" โดยต่อจากตัวอักษร "A" ก็ให้กดคีย์ CHN ก่อนแล้วตามด้วยคีย์ abcd ก็จะได้ตัวอักษร "b" ออกมา

บทที่ 7
การโปรแกรมและแก้ไข

ในการป้อนโปรแกรมเข้าเครื่องคอมพิวเตอร์นั้นก็คือ การเขียนข้อมูล(คำสั่ง)ที่ได้จัดเตรียมขึ้นแล้วทำการป้อนลงไปในหน่วยความจำ โดยข้อมูลนั้นจะต้องเก็บในตำแหน่งที่เรากำหนดขึ้นด้วย เพื่อให้โปรแกรมทำงานได้ถูกต้องตามที่เรากำหนดไว้

7.1 การเขียนคำสั่งแบนเนอร์ในหน่วยความจำ

สำหรับในอิเล็กทรอนิกส์แบนเนอร์ เราป้อนคำสั่งได้โดยปรับให้เครื่องทำงานในโหมดโปรแกรม แล้วกดคีย์คำสั่ง เครื่องก็จะนำคำสั่งนั้น ๆ ไปเก็บลงในหน่วยความจำและเครื่องจะทำการเลื่อนบรรทัดให้เพื่อทำการป้อนคำสั่งถัดไปโดยอัตโนมัติยกเว้นคำสั่ง LD,d DR,d SH,d RT,d TS- TL- และ GR- ซึ่งคำสั่ง LD,d DR,d SH,d และ RT,d ต้องกำหนดทิศทางการแสดงผลโดยใช้ ARROW KEYS ก่อนแล้วเครื่องจะเลื่อนบรรทัดให้ ส่วนคำสั่ง TS- TL- และ GR- ต้องทำการ EDIT ตัวอักษรและรูปภาพก่อน เมื่อ EDITเรียบร้อยแล้ว เครื่องจึงจะเลื่อนบรรทัดให้ แต่ถ้ายังไม่ต้องการ EDIT แต่ต้องการป้อนคำสั่งถัดไปให้กดคีย์ [LINE INC] เพื่อเลื่อนบรรทัด

ตัวอย่าง เราต้องการป้อนคำสั่งดังโปรแกรมข้างล่างนี้ลงในโปรแกรม A (PRO A)

เลขบรรทัด	คำสั่ง
01	TL
02	DL
03	TS-
04	SH,<-
05	DL
06	RE

ก่อนที่จะเริ่มป้อนคำสั่งให้เลื่อนดิพสวิทซ์(PROA/PROB)ที่แผงควบคุมไว้ที่ตำแหน่ง"ON" เพื่อเลือกส่วนที่จะโปรแกรมให้ทำการโปรแกรมไว้ใน PROGRAM A หลังจากนั้นให้ทำตามขั้นตอนดังต่อไปนี้

กดปุ่ม	แผงดิสเพลย์
	01 RE
TL-	01 TL-

กดปุ่ม		แผงดิสเพลย์
LINE INC		02 RE
DL		02 DL
TS-		03 TS-
LINE INC		04 RE
SH,	<-	05 SH,<-
DL		06 DL
		07 RE

ในการป้อนคำสั่ง ถ้าหากกดคีย์คำสั่งแล้วเครื่องไม่รับ กล่าวคือ ที่แผงดิสเพลย์ไม่เปลี่ยนแปลงยังคงแสดงคำสั่งเดิม ให้ตรวจดูดิพสวิทช์ตัวแรก ($\overline{WE}$) ที่แผงคีย์บอร์ดว่าอยู่ในตำแหน่ง "ON" หรือไม่

และถ้าต้องการเปลี่ยนแปลงคำสั่งบางคำสั่งในโปรแกรม เช่น จากโปรแกรมในตัวอย่างที่แล้ว ให้เปลี่ยนคำสั่ง SH,<- เป็น LD,-> เราทำได้โดยเลื่อนบรรทัดโดยใช้คีย์ LINE INC หรือ LINE DEC ให้ปรากฏคำสั่งที่ต้องการเปลี่ยนแสดงที่แผงดิสเพลย์ และกดคีย์คำสั่งใหม่ที่ต้องการเปลี่ยนแทนคำสั่งเดิมลงไป

กดปุ่ม		แผงดิสเพลย์
		04 SH,<-
LD,	->	04 LD,->
		05 DL

7.2 การ EDIT ตัวอักษร (TEXT)

การป้อนและแก้ไขข้อความตัวอักษรจะใช้กับคำสั่ง TS- และ TL- คำสั่งทั้งสองนี้มีความแตกต่างกันคือ คำสั่ง TL- เป็นคำสั่งที่เก็บตัวอักษรได้ถึง 100 ตัวอักษร และแสดงผลโดยวิ่งจากซ้ายไปขวาจนครบตามจำนวนตัวอักษรที่ป้อนเข้าไปโดยใช้คำสั่งนี้เพียงคำสั่งเดียว ส่วนคำสั่ง TS- เป็นคำสั่งที่เก็บตัวอักษรได้ 10 ตัวอักษร คำสั่งนี้ต้องใช้ร่วมกับคำสั่งอื่นเพื่อนำตัวอักษรที่อยู่ในคำสั่งไปแสดงผล สามารถกำหนดรูปแบบการแสดงได้หลายแบบ ซึ่งต่างกับคำสั่ง TL- ที่แสดงผลได้เพียงแบบเดียว

ในการ EDIT ตัวอักษรในคำสั่งทั้งสองนี้ นอกจากจะใช้คีย์ตัวอักษรแล้วยังต้องใช้คีย์ [<-] และ [->] เป็นคีย์แบบ AUTO REPEAT ทำการเลื่อน CURSOR ให้ไปอยู่ในตำแหน่งที่ต้องการ

7.2.1 การ EDIT คำสั่ง TS-

ตัวอย่าง เราต้องการป้อนตัวอักษร "Banner" เข้าไปในคำสั่ง TS- ดังโปรแกรมข้างล่างนี้

เลขบรรทัด	คำสั่ง
01	TS-"Banner"
02	DI
03	RE

กดปุ่ม		แผงดิสเพลย์	
		01 TS-	
EDIT		_	เริ่มต้นการ EDIT
ABCD	2 ครั้ง	B	
->		B_	
CHN		B	
abcd		Ba	
->		Ba_	

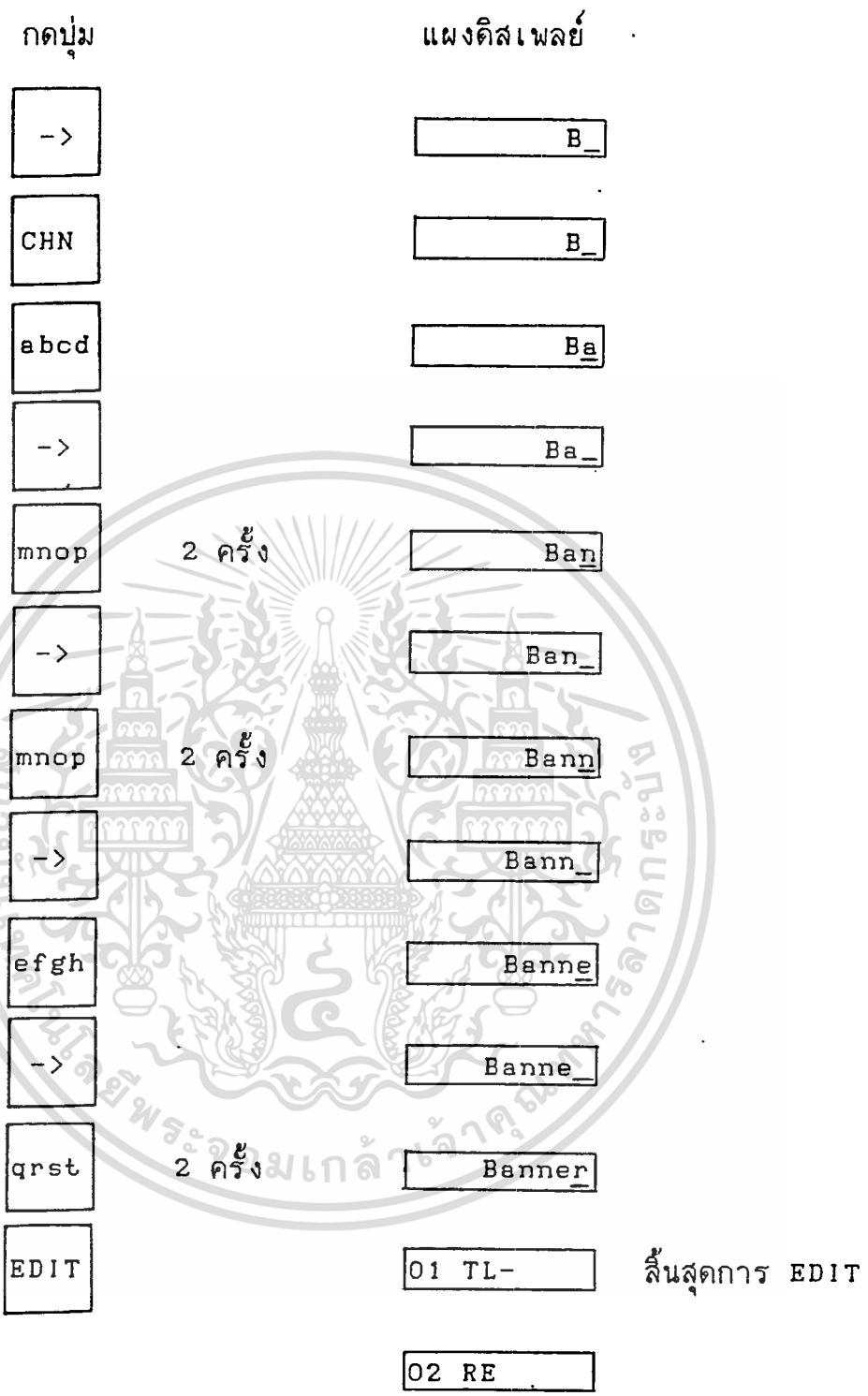
กดปุ่ม		แผงดิสเพลย์	
mnop	2 ครั้ง	Ban	
->		Ban_	
mnop	2 ครั้ง	Bann	
->		Bann_	
efgh		Banne	
->		Banne_	
qrst	2 ครั้ง	Banner	
EDIT		01 TS-	สิ้นสุดการ EDIT
		02 DI	

7.2.2 การ EDIT คำสั่ง TL-

ตัวอย่าง เราต้องการป้อนตัวอักษร "Banner" เข้าไปในคำสั่ง TL- ดังโปรแกรมข้างล่างนี้

เลขบรรทัด	คำสั่ง
01	TL-
02	RE

กดปุ่ม		แผงดิสเพลย์	
		01 TL-	
EDIT		. _	เริ่มต้นการ EDIT
ABCD	2 ครั้ง	. B	



7.3 <u>การ EDIT รูปภาพกราฟฟิค</u> (GRAPHIC)

การป้อนและแก้ไขรูปภาพกราฟฟิคใช้กับคำสั่ง GR- เท่านั้น คำสั่งนี้เก็บรูปภาพกราฟฟิคได้ 1 ภาพซึ่งภาพกราฟฟิคนี้มีความละเอียด 8 * 80 จุด การสร้างภาพกราฟฟิคในคำสั่งนี้ใช้การกำหนดจุดสว่างและจุดดับของ DOT MATRIX LED โดยใช้คีย์ CHN ซึ่งลักษณะการทำงานของคีย์นี้เป็นแบบ Toggle (สลับไปมา) คือ กดติด (จุดสว่าง) และกดอีกทีดับ

(จุดดับ)

ในการ EDIT รูปภาพกราฟฟิคนี้จะใช้ ARROW KEYS ทั้ง 4 คีย์คือ [^] [v] [<-] และ [->] ซึ่งเป็นคีย์แบบ AUTO REPEAT ใช้ย้าย CURSOR ให้ไปอยู่ในตำแหน่งของจุดภาพที่ต้องการได้

ในการเคลียร์ตัวอักษรในคำสั่ง TS- TL- และรูปภาพกราฟฟิคในคำสั่ง GR- กระทำได้ในระหว่างการ EDIT คำสั่ง เมื่อต้องการเคลียร์ก็ให้กดคีย์ [CLEAR ALL] เครื่องจะทำการเคลียร์ให้เป็น BLANK (ว่าง)

บทที่ 8

โปรแกรมตัวอย่าง (Example Program)

8.1 นาฬิกาจำลอง (Pseudo Clock)

เลขบรรทัด	คำสั่ง
01	TS-" 15 : 00"
02	DI
03	DL
04	TS-" 15 00"
05	DI
06	DL
07	RE

8.2 แสดงข้อความตัวอักษร (Text Show)

เลขบรรทัด	คำสั่ง
01	TS-"text show"
02	DI
03	DL
04	RT,<-
05	DL
06	RT,->
07	DL
08	RT,^
09	DL
10	RT,v
11	DL
12	FS
13	FS
14	DL
15	TS-"TEXT SHOW"
16	LD,->
17	DL
18	RE

8.3 <u>แสดงชุดคำสั่งแบนเนอร์</u> (Banner Instruction Set Show)

<u>เลขบรรทัด</u>	<u>คำสั่ง</u>
01	TL-"Banner Instruction Set--->"
02	DL
03	TS-" 01 CL"
04	LD,^
05	DL
06	TS-" 02 TS-"
07	LD,v
08	DL
09	TS-" 03 TL-"
10	LD,<-
11	DL
12	TS-" 04 GR-"
13	LD,->
14	DL
15	TS-" 05 DI"
16	DR,<-
17	DL
18	TS-" 06 LD,d"
19	DR,^
20	DL
21	TS-" 07 DR,d"
22	SH,^
23	DL
24	TS-" 08 SH,d"
25	SH,v
26	DL
27	TS-" 09 DL"
28	SH,<-
29	DL
30	TS-" 10 FS"
31	SH,->

เลขบรรทัด	คำสั่ง
32	DL
33	TS-" 11 IN"
34	DI
35	RT,^
36	DL
37	TS-" 12 RT,d"
38	DI
39	RT,<-
40	DL
41	TS-" 13 RE"
42	DI
43	FS
44	DL
45	TL-"end----"
46	DL
47	RE

บทที่ 9
สรุปผลโครงงานและข้อเสนอแนะ

หลังจากที่ได้สร้างอิเล็กทรอนิกส์แบนเนอร์เสร็จแล้ว มีส่วนที่ต้องปรับแต่งก่อนที่จะใช้งานคือตัวต้านทานเกือกม้า VR1 ซึ่งอยู่บนแผงควบคุม เพื่อปรับความถี่ของสัญญาณมาสเคเบิ้ลอินเตอร์รัพท์ซึ่งใช้ในการสแกนดิสเพลย์ โดยปรับ VR1 พร้อมกับสังเกตตัวอักษรที่ปรากฏที่แผงดิสเพลย์ ตัวอักษรต้องนิ่ง ไม่พริ้วหรือกระตุก การวิ่งของตัวอักษรนิ่งเรียบ สม่ำเสมอ และต้องให้ได้ความสว่างของ DOT MATRIX LED สูงสุดด้วย ซึ่งหลังจากได้ทำการทดลองปรับแต่งเรียบร้อยแล้วให้ผลเป็นไปตามที่ต้องการ

เมื่อปรับแต่งเรียบร้อยแล้ว ได้ทดลองใช้งานพบว่าผลของโครงงานเป็นไปตามจุดมุ่งหมายที่ได้กำหนดไว้ คือสามารถทำการโปรแกรมรูปแบบของตัวอักษร รูปภาพกราฟฟิค และสามารถโปรแกรมรูปแบบการแสดงผลได้ การใช้งานง่ายและสดวก นอกจากนั้นยังสามารถเก็บโปรแกรมที่ได้ป้อนเข้าเครื่องไว้แล้วได้อีกด้วย ในเครื่องต้นแบบนี้ใช้แบตเตอรี่ธรรมดา แต่ในการใช้งานจริง ๆ ควรใช้แบตเตอรี่นิเกิ้ลแคดเมี่ยม ทำให้ไม่ต้องเปลี่ยนแบตเตอรี่บ่อย เนื่องจากเครื่องจะทำการชาร์ทไฟให้โดยอัตโนมัติ แต่ก็มีปัญหา ส่วนใหญ่เกิดจากแผ่นวงจรที่ใช้เนื่องจากแผงวงจรบางส่วนจำเป็นต้องใช้แผ่นวงจรชนิด 2หน้าโดยใช้ลวดทองแดงเชื่อมระหว่างจุดบัดกรีบนและล่าง ทำให้มีปัญหาเครื่องรวนบ่อย ๆ โดยเฉพาะที่แผงดิสเพลย์จะปรากฏว่ามี DOT MATRIX LED บางจุดสว่างค้างและบางจุดดับ

ถ้าจะเปรียบเทียบอิเล็กทรอนิกส์แบนเนอร์กับแผงตัวอักษรวิ่งทั่วไปแล้ว ซึ่งส่วนใหญ่จะเป็นของที่สั่งเข้ามาจากต่างประเทศ อิเล็กทรอนิกส์แบนเนอร์มีรูปแบบการแสดงผลมากกว่าและสามารถแสดงผลเป็นภาพกราฟฟิคได้ ซึ่งในแผงตัวอักษรวิ่งทั่วไปไม่มี นอกจากนั้นผู้ใช้ยังสามารถทำโปรแกรมเพื่อเปลี่ยนแปลงรูปแบบตัวอักษรและรูปภาพกราฟฟิค รวมทั้งลักษณะการแสดงผลได้ด้วยตนเอง โดยไม่ต้องเสียค่าใช้จ่ายใด ๆ

ข้อเสนอแนะสำหรับโครงงานนี้ มีดังนี้

1. พัฒนาโครงงานนี้ให้สามารถแสดงผลเป็นตัวอักษรภาษาไทยได้
2. เพิ่มคำสั่งแบนเนอร์เพื่อเพิ่มรูปแบบการแสดงผล
3. เพิ่มคำสั่งที่ช่วยในการโปรแกรมและการแสดงผลรูปภาพกราฟฟิคเคลื่อนไหว

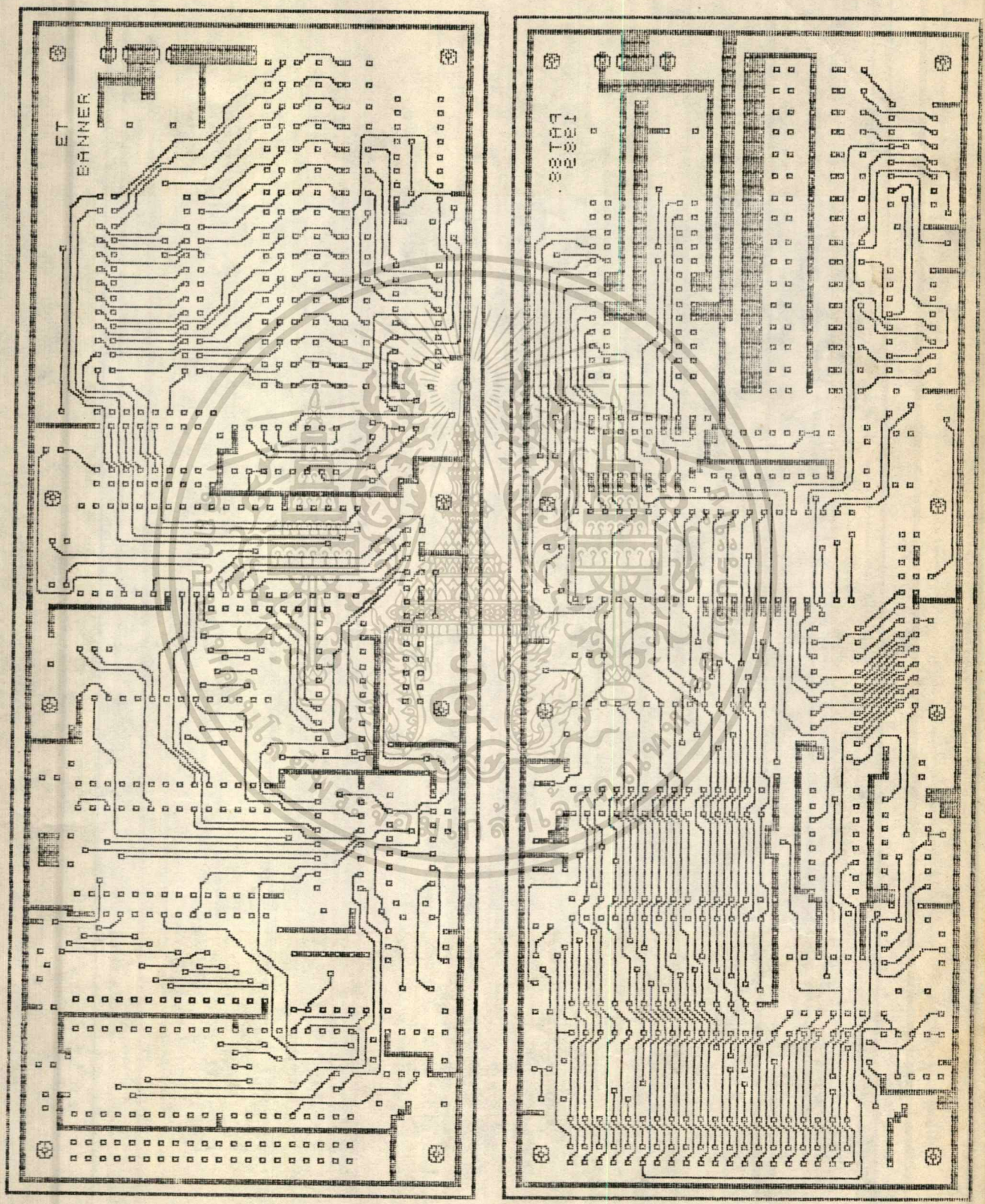
ภาคผนวก ก.

SPECIFICATION OF ELECTRONIC BANNER

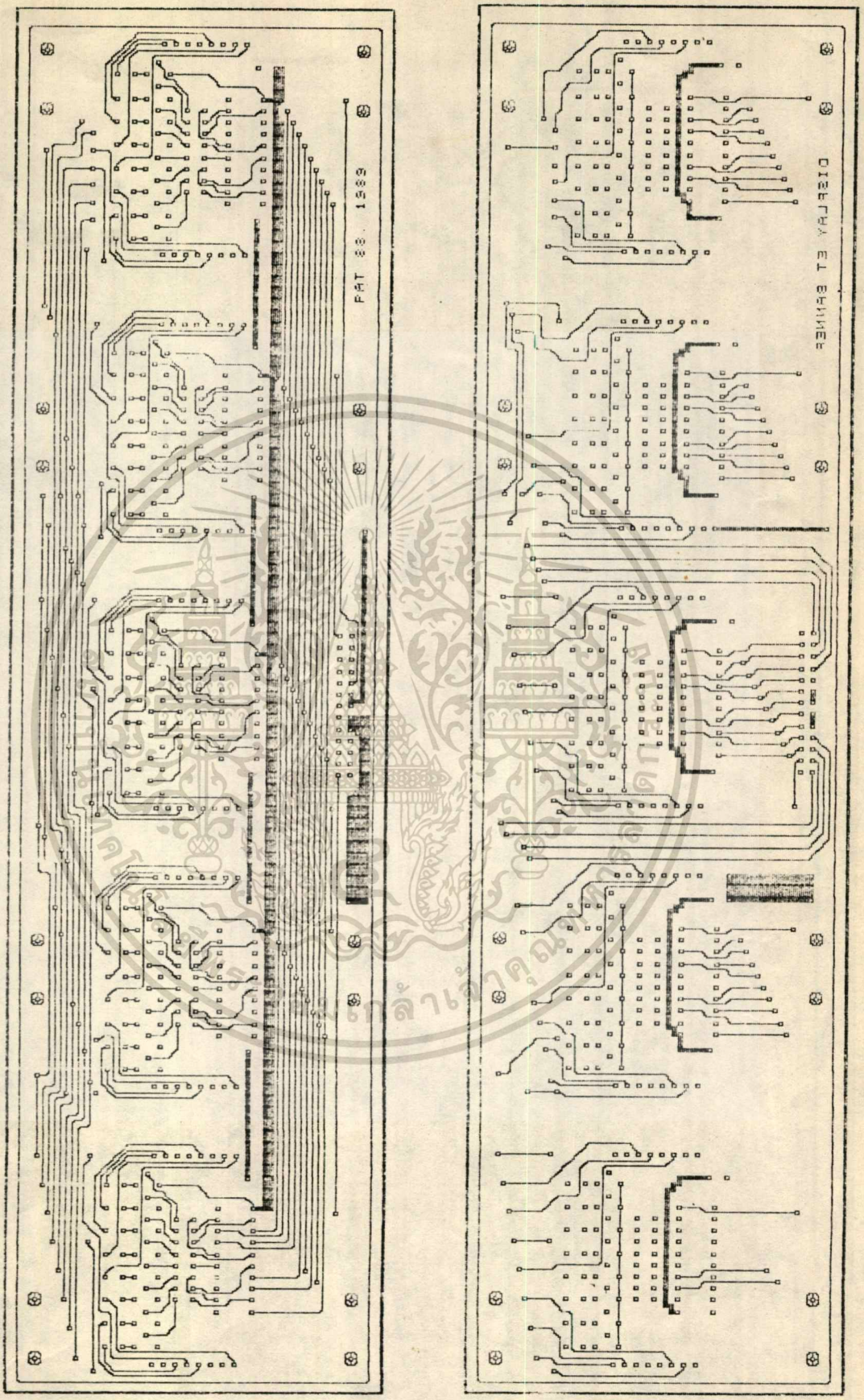
CPU	Z80A
ROM	2764 8kbyte
RAM	6264, 6116 10kbyte
PORT	8255 3 port
KEYBOARD	24 keys (12 banner instruction keys, 10 programming keys, reset key)
DISPLAY	10 digits 8 * 8 dot matrix LED
LED	1 power red LED
TRANSISTOR	97
TTL IC	17
LINEAR IC	1
REGULATOR IC	1
CONNECTOR	1 40pins-header strip for system expansion
	6 26pins-header strip on display board and keyboard board
	2 4pins-connector for DC supply
DIP SWITCH	2 selected programming or run mode and RAM write protect (on keyboard board)
	1 selected program A or B (on control board)
CLOCK RATE	3.579545 MHz
POWER SUPPLY	5VDC for system
	8VDC for transister driver
BACKUP RAM	8kbyte

ภาคผนวก ข.

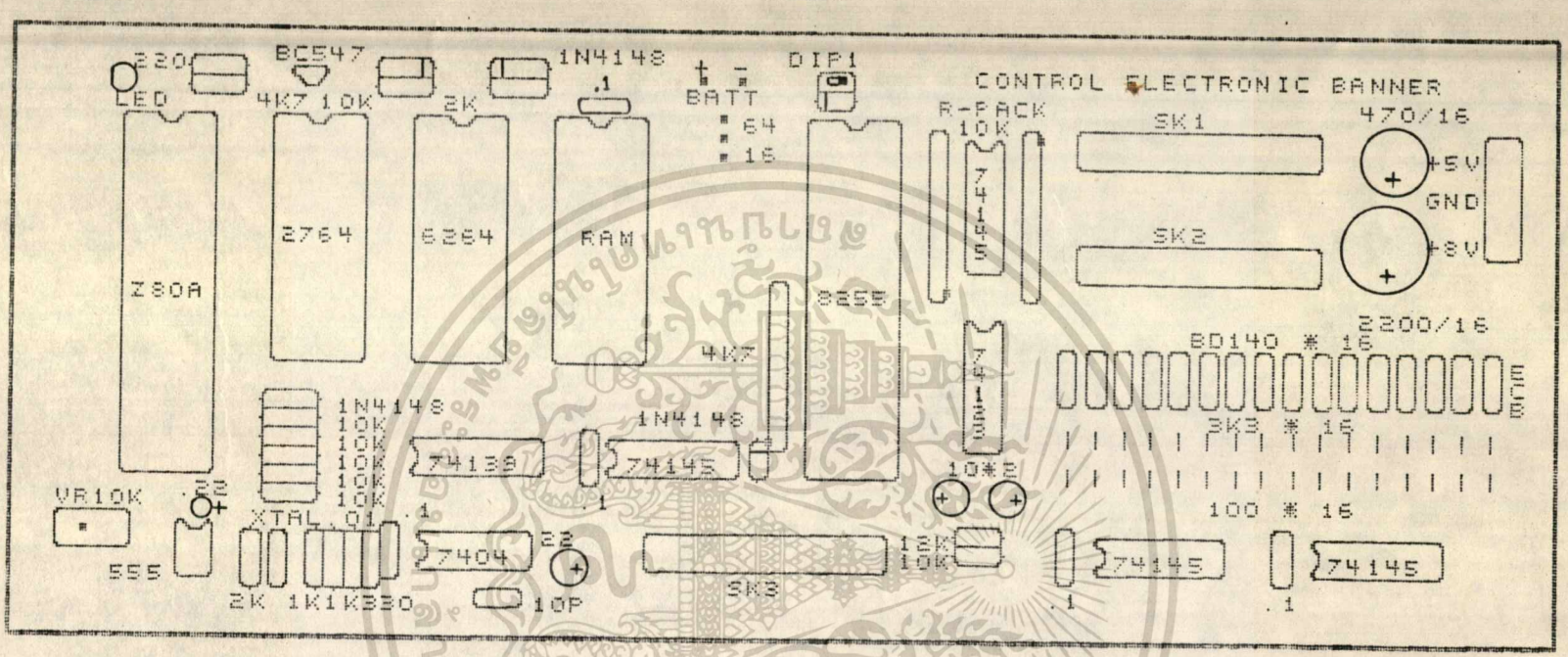
PRINT CIRCUIT BOARD

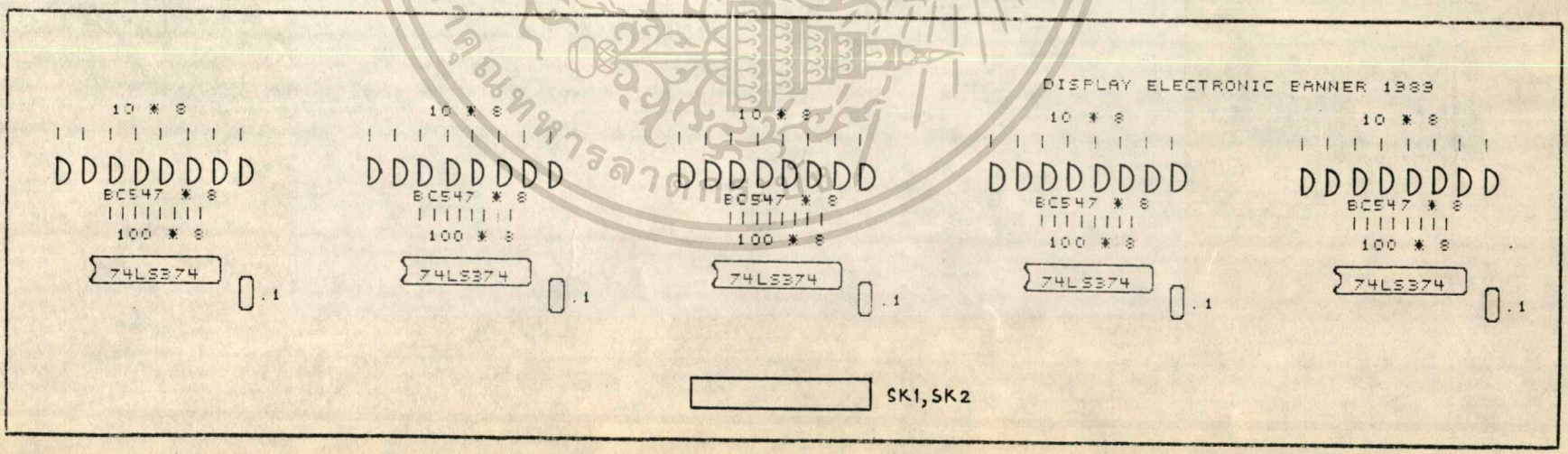


แสดงลายทองแดงด้าน component side และ solder side ของ Control Board

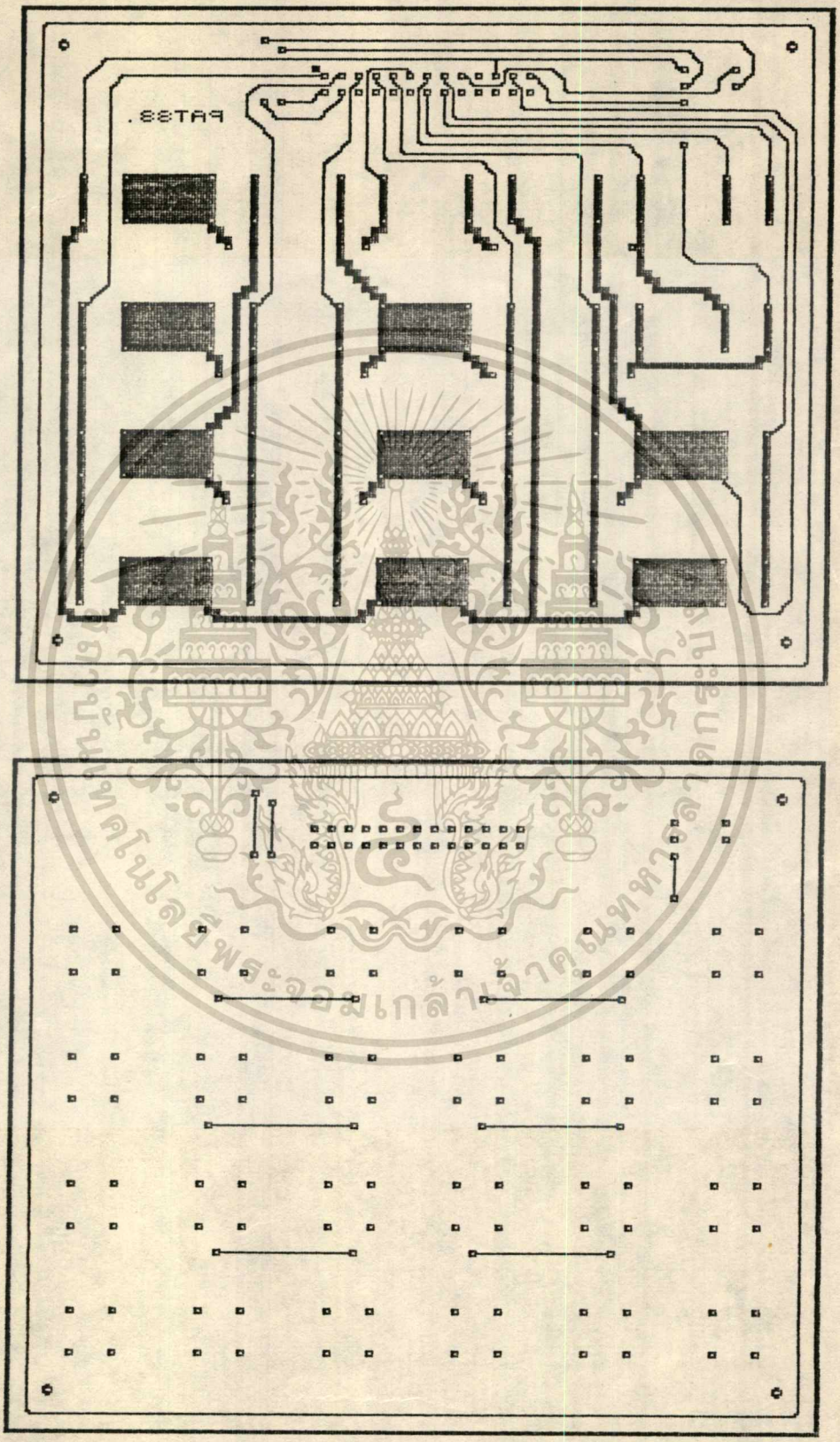


แสดงลายทองแดงด้าน component side และ solder side ของ Display Board

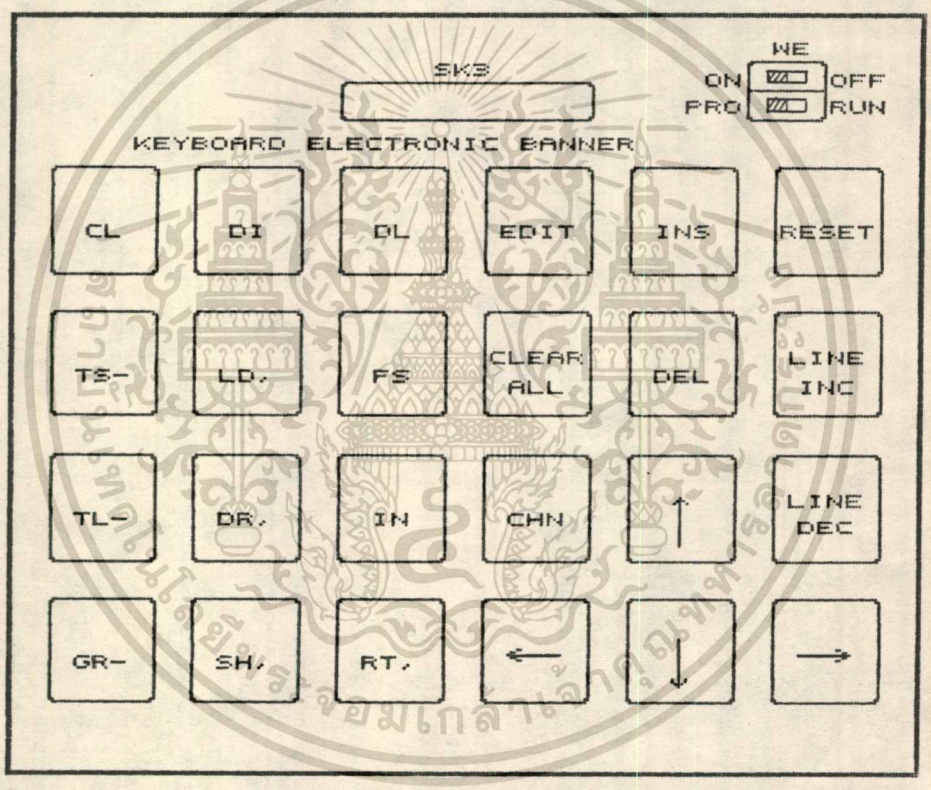




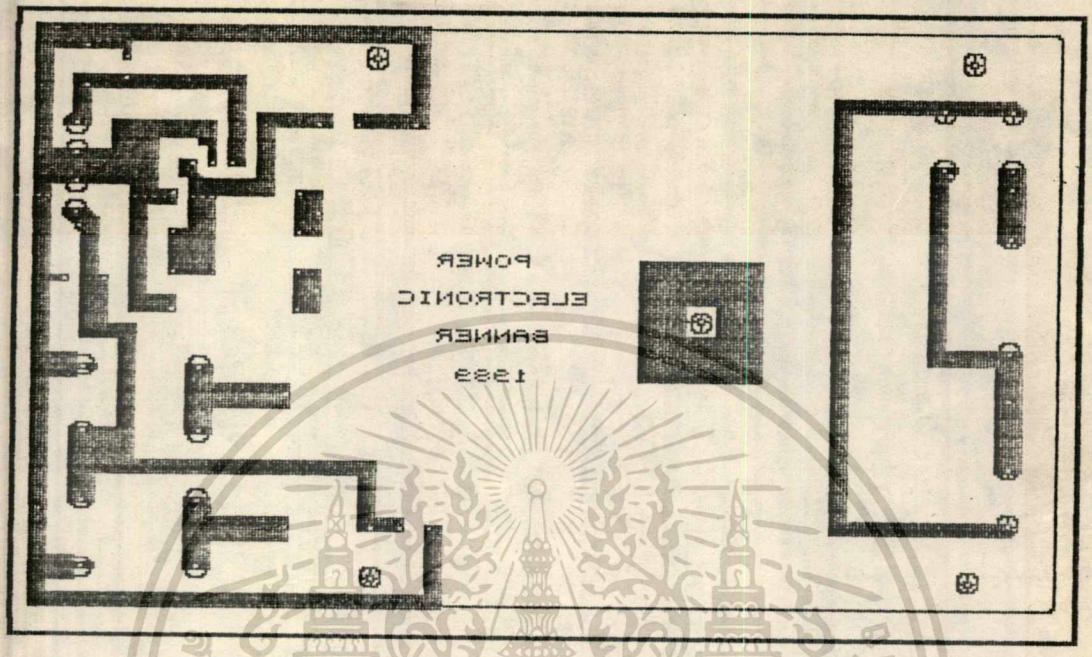
แสดงการวางอุปกรณ์ของ Control Board และ Display Board

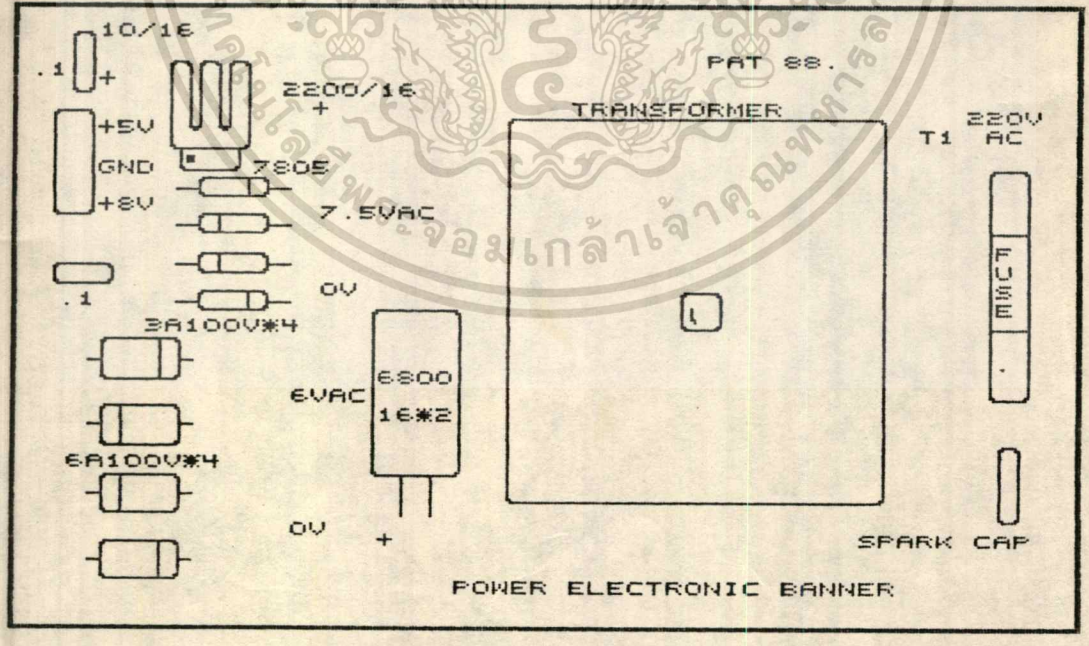


แสดงลายทองแดงด้าน solder และสายจั้มด้าน componentของ Keyboard Board



แสดงการวางอุปกรณ์ของ Keyboard Board

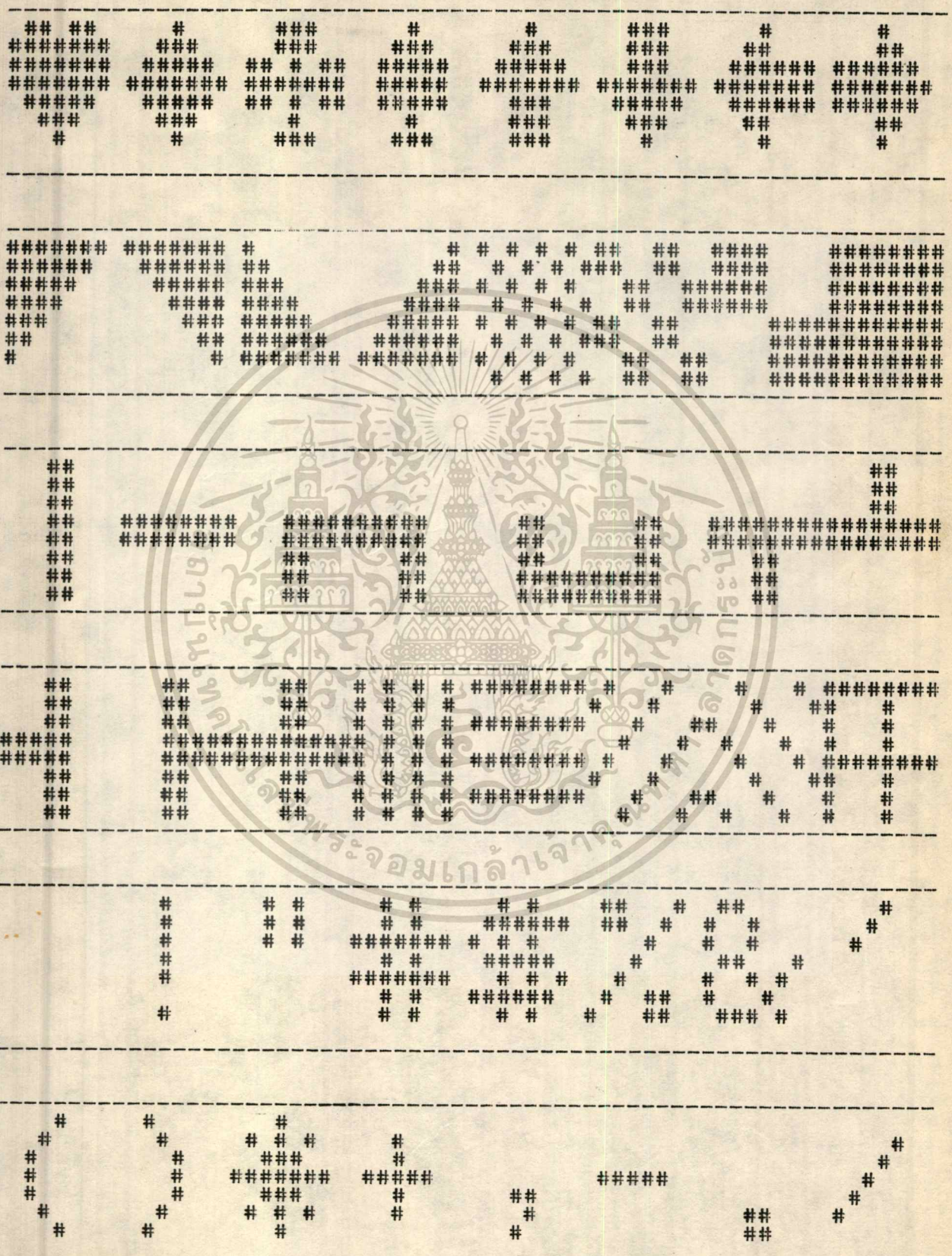


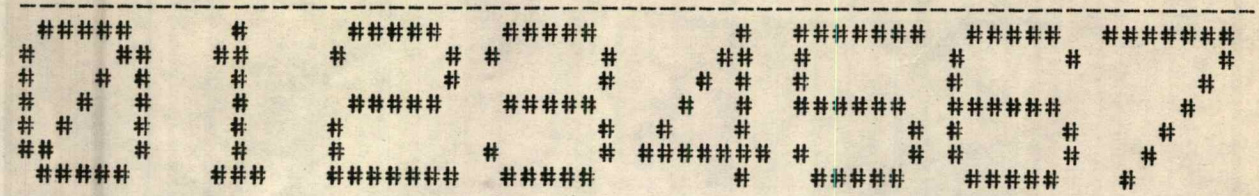


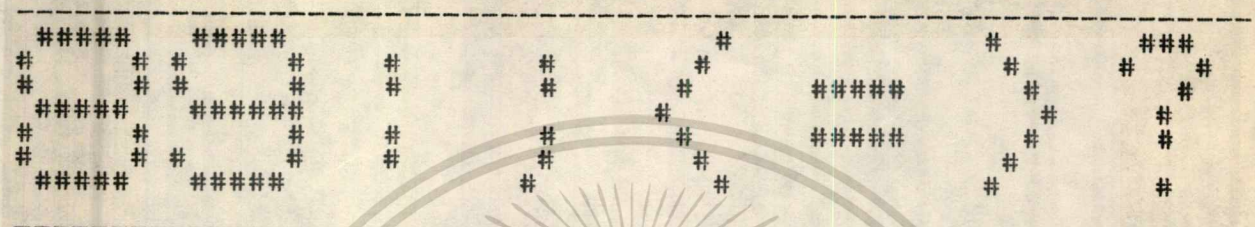
แสดงลายทองแดงด้าน solder และการวางอุปกรณ์ของ Power Supply Board

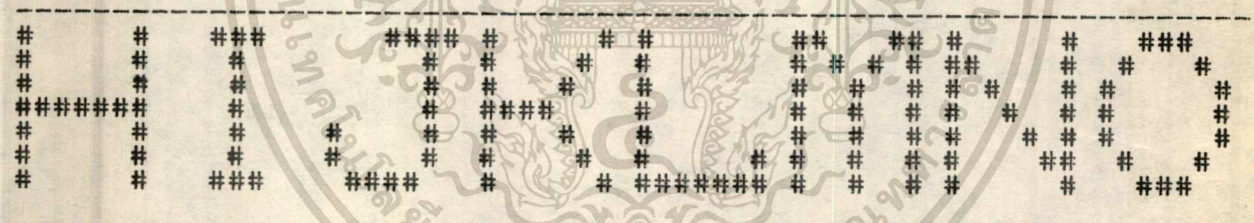
ภาคผนวก ค.

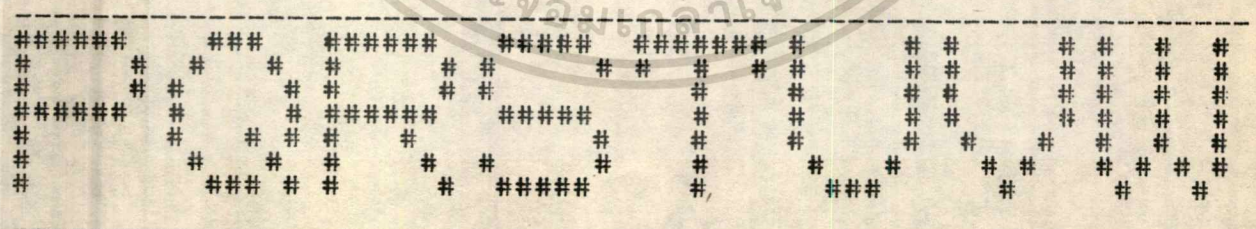
DISPLAY CHARACTER

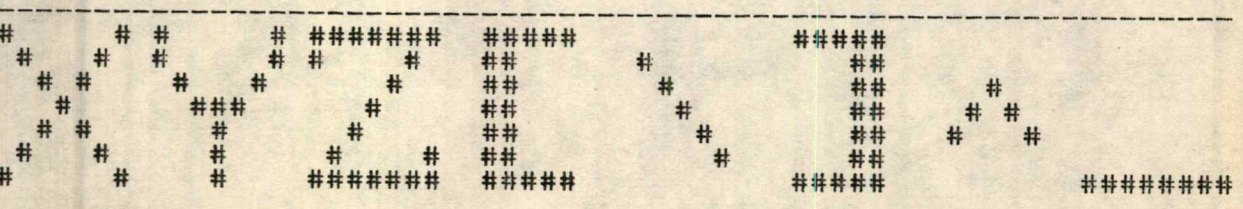


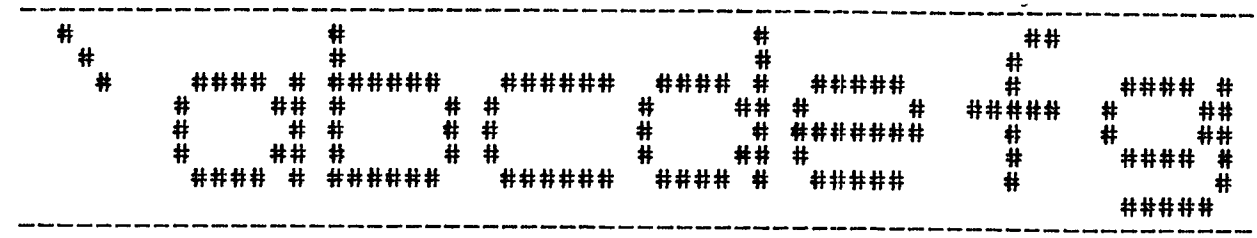


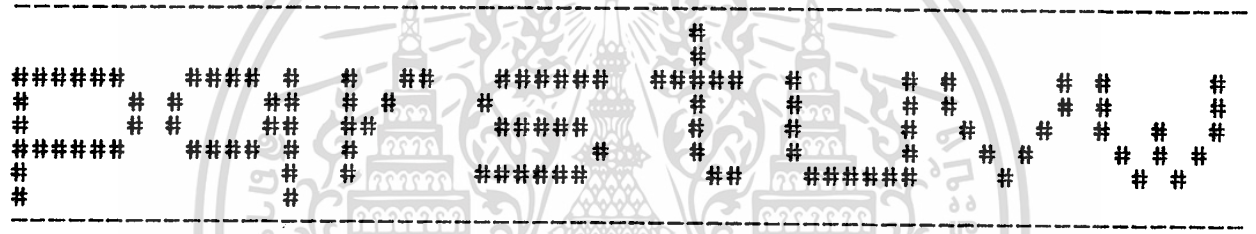












ภาคผนวก ง.

MONITOR PROGRAM

```
2500 A.D. Z80 CROSS ASSEMBLER   -   VERSION 3.00b
--------------------------------------------------

        INPUT  FILENAME :  BANNER.Z80
        OUTPUT FILENAME :  BANNER.OBJ


0000                    ORG  0000H
                    ;*********************************
                    ;*                               *
                    ; * ELECTRONIC BANNER             *
                    ; * 10 digit model                *
                    ; * programmable computer display *
                    ;*                               *
                    ; * Developed by Pat 88.          *
                    ;*              KMITL            *
                    ;*              ET 2N (303315)   *
                    ; * Mar 1989   Bangkok Thailand   *
                    ;*                               *
                    ;*********************************
                    ;
                    ; * Variable declarations
                    ; * --------------------
                    ;
00 00               PDATA    EQU  0
01 00               PCHAR    EQU  1
02 00               PKEYB    EQU  2
03 00               PCONT    EQU  3
40 00               WATDOG    EQU  40H
01 00               SPEED1   EQU  1
04 00               SPEED2    EQU  4
50 00               BYTE     EQU  80
A3 00               CODE     EQU  0A3H
00 1C               CHRGEN   EQU  1C00H
                    ;
                    ;
                    ;
                    ;************************
                    ;*                      *
                    ;*    SYSTEM RESET 1    *
                    ;*                      *
                    ;************************
                    ;
                    ; * Power up delay
                    ;
0000   AF           RESET0   XOR  A
0001   3D           RESET1   DEC  A
0002   20 FD                 JR   NZ,RESET1
```

```
                        *
                        * Set stack and 8255
                        *
0004   31 6A 87                LD    SP,STACK
0007   3E 88                   LD    A,88H
0009   D3 03                   OUT   (PCONT),A
000B   3E FF                   LD    A,0FFH        ;off display
000D   D3 01                   OUT   (PCHAR),A
                        *
                        * cold start check
                        *
000F   3A FE 87                LD    A,(PWCODE)
0012   FE A3                   CP    CODE
0014   CA B2 00                JP    Z,RESET2
                        *
                        * clear working area
                        *
0017   21 38 5F        INIT    LD    HL,LBYTEA
001A   01 C8 00                LD    BC,200
001D   16 00                   LD    D,0
001F   72               INIT1   LD    (HL),D
0020   23                      INC   HL
0021   0B                      DEC   BC
0022   78                      LD    A,B
0023   B1                      OR    C
0024   20 F9                   JR    NZ,INIT1
                        *
0026   21 00 80                LD    HL,RAM2
0029   01 00 08                LD    BC,2048
002C   16 00                   LD    D,0
002E   72               INIT2   LD    (HL),D
002F   23                      INC   HL
0030   0B                      DEC   BC
0031   78                      LD    A,B
0032   B1                      OR    C
0033   20 F9                   JR    NZ,INIT2
0035   C3 A3 00                JP    INIT3
                        *
                        *
                        *
                        ************************
                        *                      *
                        * SCAND(INT) SUBROUTINE *
                        *                      *
                        ************************
                        *
0038   F5               SCAND   PUSH AF
0039   C5                       PUSH BC
```

```
003A   D5                        PUSH DE
003B   E5                        PUSH HL
003C   DD E5                     PUSH IX
003E   FD E5                     PUSH IY
                          *
0040   3A 6A 87                  LD   A,(ROW)
0043   FE 7F                     CP   7FH          ;first row
0045   20 0B                     JR   NZ,SCAND1
0047   21 00 80                  LD   HL,DIS
004A   11 10 84                  LD   DE,DIS1
004D   01 50 00                  LD   BC,BYTE
0050   ED B0                     LDIR
                          *
0052   21 10 84        SCAND1    LD   HL,DIS1
0055   DD 21 6C 87               LD   IX,SHOUT
0059   06 0A                     LD   B,10         ;10 char
005B   0E F0                     LD   C,0F0H       ;first char
005D   16 08           SCAND2    LD   D,8
005F   CB 0E           SCAND3    RRC  (HL)
0061   DD CB 00 1E               RR   (IX+0)
0065   23                        INC  HL
0066   15                        DEC  D
0067   20 F6                     JR   NZ,SCAND3
0069   DD 7E 00                  LD   A,(IX+0)
006C   D3 00                     OUT  (PDATA),A
006E   79                        LD   A,C
006F   D3 01                     OUT  (PCHAR),A
0071   3E FF                     LD   A,0FFH
0073   D3 01                     OUT  (PCHAR),A
0075   0C                        INC  C
0076   10 E5                     DJNZ SCAND2
                          *
0078   3A 6A 87                  LD   A,(ROW)
007B   D3 01                     OUT  (PCHAR),A
007D   D6 10                     SUB  10H
007F   32 6A 87                  LD   (ROW),A
0082   FE FF                     CP   0FFH
0084   20 10                     JR   NZ,SCAND4
0086   3E 7F                     LD   A,7FH
0088   32 6A 87                  LD   (ROW),A
                          *
008B   3A 71 87                  LD   A,(COUNT)
008E   FE 00                     CP   0
0090   28 04                     JR   Z,SCAND4
0092   3D                        DEC  A
0093   32 71 87                  LD   (COUNT),A
                          *
0096   D3 40           SCAND4    OUT  (WATDOG),A   ;trig watch dog
```

```
0098   FD E1              POP  IY
009A   DD E1              POP  IX
009C   E1                 POP  HL
009D   D1                 POP  DE
009E   C1                 POP  BC
009F   F1                 POP  AF
00A0   FB                 EI
00A1   ED 4D              RETI
                *
                * fill 0CH (return) to pro a-b
                *
00A3   21 00 40   INIT3   LD   HL,PROA
00A6   01 38 1F           LD   BC,7992
00A9   16 0C              LD   D,0CH
00AB   72         INIT4   LD   (HL),D
00AC   23                 INC  HL
00AD   0B                 DEC  BC
00AE   78                 LD   A,B
00AF   B1                 OR   C
00B0   20 F9              JR   NZ,INIT4
                *
                *
                *
                ************************
                *                      *
                *    SYSTEM RESET 2    *
                *                      *
                ************************
                *
00B2   3E 7F      RESET2  LD   A,7FH        ;first row
00B4   32 6A 87           LD   (ROW),A
                *
                * Set interupt mode 1
                *
00B7   ED 56              IM   1
00B9   FB                 EI
                *
00BA   3A FE 87           LD   A,(PWCODE)
00BD   FE A3              CP   CODE
00BF   28 0E              JR   Z,RESET3
00C1   3E A3              LD   A,CODE
00C3   32 FE 87           LD   (PWCODE),A
                *
00C6   21 01 0E           LD   HL,HELTAB    ;display title
00C9   22 75 87           LD   (ADDR),HL
00CC   CD A1 0A           CALL TL
                *
                * check dip sw.
```

```
                        *
00CF   21 FF 87     RESET3  LD   HL,SYSFAG
00D2   36 00                LD   (HL),0        ;clear sysfag
00D4   16 FF                LD   D,0FFH
00D6   CD 6B 05             CALL CKDIP1
                        *
                        *
                        *
                        ************************
                        *                      *
                        *   PROGRAMMING MAIN   *
                        *                      *
                        ************************
                        *
                        * Selected working area
                        * for program a/b
                        *
00D9   21 FF 87     PGM     LD   HL,SYSFAG
00DC   CB 76                BIT  6,(HL)
00DE   28 0A                JR   Z,PGM1
00E0   DD 21 00 40          LD   IX,PROA       ;b6=1
00E4   FD 21 38 5F          LD   IY,LBYTEA
00E8   18 08                JR   PGM2
00EA   DD 21 9C 4F  PGM1    LD   IX,PROB       ;b6=0
00EE   FD 21 9C 5F          LD   IY,LBYTEB
                        *
                        * display first line num and instruction
                        * "01 XXX"
                        *
00F2   3E 01        PGM2    LD   A,1           ;first line
00F4   32 77 87             LD   (LINE),A
00F7   CD 12 09             CALL DISPRO
                        *
                        * Check data first byte
                        *
00FA   DD 7E 00             LD   A,(IX+0)
00FD   FE 0C                CP   0CH
00FF   28 1C                JR   Z,PGM3        ;0CH(RE)
0101   DA E4 01             JP   C,PGM6
                        *
                        * display "Error"
                        *
0104   CD 70 0A             CALL CL
0107   21 33 0E             LD   HL,ERRTAB
010A   11 50 80             LD   DE,BUF
010D   06 05                LD   B,5
010F   7E           PGM2A   LD   A,(HL)
0110   CD 89 0A             CALL TSS
```

```
0113   23                      INC  HL
0114   10 F9                   DJNZ PGM2A
0116   CD FF 0A                CALL DX
0119   CD 93 0C                CALL DL
011C   76                      HALT
                        *
                        * Check intruction keys only
                        *
011D   CD 1C 05        PGM3     CALL SCANK
0120   3A 74 87                 LD   A,(KEYIN)
0123   FE 0C                    CP   0CH
0125   30 F6                    JR   NC,PGM3
                        *
                        * display line num and instruction
                        *
0127   3A 74 87        PGM4     LD   A,(KEYIN)
012A   DD 77 00                 LD   (IX+0),A
012D   CD 12 09        PGM4A    CALL DISPRD
                        *
                        * Instruction keys
                        *
0130   3A 74 87                 LD   A,(KEYIN)
0133   FE 01                    CP   01H            ;TS-
0135   28 20                    JR   Z,PGM5A
0137   FE 02                    CP   02H            ;TL-
0139   28 2F                    JR   Z,PGM5C
013B   FE 03                    CP   03H            ;GR-
013D   28 3C                    JR   Z,PGM5D
013F   FE 05                    CP   05H            ;LD,
0141   28 49                    JR   Z,PGM5F
0143   FE 06                    CP   06H            ;DR,
0145   28 45                    JR   Z,PGM5F
0147   FE 07                    CP   07H            ;SH,
0149   28 41                    JR   Z,PGM5F
014B   FE 0B                    CP   0BH            ;RT,
014D   28 3D                    JR   Z,PGM5F
                        *
                        * fill 1 byte op-code
                        *
014F   3E 01           PGM5     LD   A,1            ;1byte
0151   FD 77 00                 LD   (IY+0),A
0154   C3 DE 01                 JP   PGM5L
                        *
                        * fill TS- op-code and data
                        *
0157   DD E5          PGM5A    PUSH IX
0159   E1                      POP  HL
015A   06 0A                    LD   B,BYTE/8
```

```
015C   3E 20            LD   A,20H          ;blank
015E   23        PGM5B   INC  HL
015F   77                LD   (HL),A
0160   10 FC             DJNZ PGM5B
0162   3E 0B             LD   A,BYTE/8+1    ;11byte
0164   FD 77 00          LD   (IY+0),A
0167   C3 E4 01          JP   PGM6
                 *
                 * fill TL- op-code,num and data
                 *
016A   3E 01      PGM5C  LD   A,1
016C   DD 77 01          LD   (IX+1),A      ;num=1
016F   3E 20             LD   A,20H         ;blank
0171   DD 77 02          LD   (IX+2),A
0174   3E 03             LD   A,3           ;3byte
0176   FD 77 00          LD   (IY+0),A
0179   18 69             JR   PGM6
                 *
                 * fill GR- op-code and data
                 *
017B   DD E5      PGM5D  PUSH IX
017D   E1                POP  HL
017E   06 50             LD   B,BYTE
0180   AF                XOR  A             ;space
0181   23        PGM5E  INC  HL
0182   77                LD   (HL),A
0183   10 FC             DJNZ PGM5E
0185   3E 51             LD   A,BYTE+1      ;81byte
0187   FD 77 00          LD   (IY+0),A
018A   18 58             JR   PGM6
                 *
                 * Check direction keys (U,D,L,R)
                 *
018C   CD 1C 05  PGM5F   CALL SCANK
018F   3A 74 87          LD   A,(KEYIN)
0192   FE 12             CP   12H           ;^
0194   28 0E             JR   Z,PGM5G
0196   FE 13             CP   13H           ;v
0198   28 14             JR   Z,PGM5H
019A   FE 0F             CP   0FH           ;<-
019C   28 1A             JR   Z,PGM5I
019E   FE 16             CP   16H           ;->
01A0   28 20             JR   Z,PGM5J
01A2   18 E8             JR   PGM5F
                 *
01A4   3E 00      PGM5G  LD   A,0
01A6   32 72 87          LD   (DIRBUF),A
01A9   3A 9D 0D          LD   A,(ISCTAB+39);U
```

```
01AC   18 1C              JR    PGM5K
01AE   3E 01      PGM5H   LD    A,1
01B0   32 72 87           LD    (DIRBUF),A
01B3   3A 9E 0D           LD    A,(ISCTAB+40);D
01B6   18 12              JR    PGM5K
01B8   3E 02      PGM5I   LD    A,2
01BA   32 72 87           LD    (DIRBUF),A
01BD   3A 9F 0D           LD    A,(ISCTAB+41);L
01C0   18 08              JR    PGM5K
01C2   3E 03      PGM5J   LD    A,3
01C4   32 72 87           LD    (DIRBUF),A
01C7   3A A0 0D           LD    A,(ISCTAB+42);R
                  *
01CA   11 80 80   PGM5K   LD    DE,BUF+48     ;char 4
01CD   CD 89 0A           CALL  TSS
01D0   CD FF 0A           CALL  DX
01D3   3A 72 87           LD    A,(DIRBUF)
01D6   DD 77 01           LD    (IX+1),A      ;fill dir code
01D9   3E 02              LD    A,2           ;2byte
01DB   FD 77 00           LD    (IY+0),A
                  *
01DE   CD 93 0C   PGM5L   CALL  DL            ;auto line inc
01E1   C3 8B 02           JP          PGM8
                  *
                  * Check programming keys
                  *
01E4   21 FF 87   PGM6    LD    HL,SYSFAG
01E7   3A 74 87           LD    A,(KEYIN)
01EA   FE 0D              CP    0DH           ;clear all
01EC   20 04              JR    NZ,PGM6A
01EE   CB E6              SET   4,(HL)
01F0   18 02              JR    PGM6B
01F2   CB A6      PGM6A   RES   4,(HL)
01F4   CD 1C 05   PGM6B   CALL  SCANK
01F7   3A 74 87           LD    A,(KEYIN)
01FA   FE 0C              CP    0CH
01FC   DA 1F 02           JP    C,PGM7        ;instruction
01FF   FE 14              CP    14H
0201   CA 8B 02           JP    Z,PGM8        ;line inc
0204   FE 15              CP    15H
0206   CA B9 02           JP    Z,PGM9        ;line dec
0209   FE 0C              CP    0CH
020B   CA E1 02           JP    Z,PGM10       ;edit
020E   FE 10              CP    10H
0210   CA F3 03           JP    Z,PGM11       ;ins
0213   FE 11              CP    11H
0215   CA 3F 04           JP    Z,PGM12       ;del
0218   FE 0D              CP    0DH
```

```
021A   CA A0 04              JP   Z,PGM13       ;clear all
021D   18 D5                 JR   PGM6B
                      *
                      * instruction keys (new press)
                      *
021F   DD 7E 00     PGM7      LD   A,(IX+0)
0222   FE 0C                 CP   0CH           ;RE
0224   CA 27 01              JP   Z,PGM4
0227   3A 74 87               LD   A,(KEYIN)
022A   CD 6F 09               CALL PROBYT
022D   FD 7E 00               LD   A,(IY+0)
0230   4F                     LD   C,A
0231   32 7D 87               LD   (OLDNUM),A   ;old lbyte
0234   3E 00                  LD   A,0
0236   FD 77 00               LD   (IY+0),A
0239   3A 7E 87               LD   A,(NEWNUM)   ;new lbyte
023C   B9                     CP   C
023D   CA 27 01               JP   Z,PGM4
0240   38 1D                  JR   C,OPDEL
                      *
0242   DD E5        OPINS     PUSH IX            ;new>old
0244   DD 22 7F 87            LD   (START),IX    ;start addr
0248   CD E8 09               CALL FNPRO
024B   22 81 87               LD   (FINAL),HL    ;final addr
024E   3A 7D 87               LD   A,(OLDNUM)
0251   4F                     LD   C,A
0252   3A 7E 87               LD   A,(NEWNUM)
0255   91                     SUB  C
0256   47                     LD   B,A           ;ins byte
0257   CD A1 09               CALL INSERT
025A   DD E1                  POP  IX
025C   C3 27 01               JP   PGM4
                      *
025F   DD E5        OPDEL     PUSH IX            ;new<old
0261   3A 7D 87               LD   A,(OLDNUM)
0264   4F                     LD   C,A
0265   06 00                  LD   B,0
0267   DD 09                  ADD  IX,BC
0269   DD 2B                  DEC  IX
026B   DD 22 7F 87            LD   (START),IX
026F   CD E8 09               CALL FNPRO
0272   22 81 87               LD   (FINAL),HL
0275   3A 7E 87               LD   A,(NEWNUM)
0278   4F                     LD   C,A
0279   3A 7D 87               LD   A,(OLDNUM)
027C   91                     SUB  C
027D   47                     LD   B,A           ;del byte
027E   3E 0C                  LD   A,0CH         ;RE
```

```
0280   32 7C 87            LD   (STOREY),A
0283   CD C2 09            CALL DELETE
0286   DD E1               POP  IX
0288   C3 27 01            JP   PGM4
                    *
                    * line inc key
                    *
028B   3A 77 87    PGM8     LD   A,(LINE)
028E   FE 99                CP   99H          ;final line
0290   CA E4 01             JP   Z,PGM6
0293   DD 7E 00             LD   A,(IX+0)
0296   FE 0C                CP   0CH          ;0Ch (RE)
0298   CA E4 01             JP   Z,PGM6
029B   DD E5                PUSH IX
029D   E1                   POP  HL
029E   FD 7E 00             LD   A,(IY+0)
02A1   5F                   LD   E,A
02A2   16 00                LD   D,0
02A4   19                   ADD  HL,DE
02A5   E5                   PUSH HL
02A6   DD E1                POP  IX
02A8   3A 77 87             LD   A,(LINE)
02AB   C6 01                ADD  A,1
02AD   27                   DAA
02AE   32 77 87             LD   (LINE),A
02B1   FD 23                INC  IY
02B3   CD 12 09             CALL DISPRO
02B6   C3 E4 01             JP   PGM6
                    *
                    * line dec key
                    *
02B9   3A 77 87    PGM9     LD   A,(LINE)
02BC   FE 01                CP   1            ;first line
02BE   CA E4 01             JP   Z,PGM6
02C1   DD E5                PUSH IX
02C3   E1                   POP  HL
02C4   FD 2B                DEC  IY
02C6   FD 7E 00             LD   A,(IY+0)
02C9   5F                   LD   E,A
02CA   16 00                LD   D,0
02CC   AF                   XOR  A
02CD   ED 52                SBC  HL,DE
02CF   E5                   PUSH HL
02D0   DD E1                POP  IX
02D2   3A 77 87             LD   A,(LINE)
02D5   D6 01                SUB  1
02D7   27                   DAA
02D8   32 77 87             LD   (LINE),A
```

```
02DB   CD 12 09            CALL DISPRO
02DE   C3 E4 01            JP   PGM6
                    *
                    * edit key
                    *
02E1   DD 7E 00    PGM10    LD   A,(IX+0)
02E4   FE 01                CP   01H
02E6   28 0C                JR   Z,PGMTS
02E8   FE 02                CP   02H
02EA   28 2C                JR   Z,PGMTL
02EC   FE 03                CP   03H
02EE   CA CF 03             JP   Z,PGMGR
02F1   C3 E4 01             JP   PGM6
                    *
02F4   DD E5       PGMTS    PUSH IX
02F6   E1                   POP  HL
02F7   23                   INC  HL
02F8   11 8C 87             LD   DE,EDBUF
02FB   01 0A 00             LD   BC,10
02FE   ED B0                LDIR
0300   CD 9A 06             CALL TSEDIT
0303   DD E5                PUSH IX
0305   D1                   POP  DE
0306   13                   INC  DE
0307   21 8C 87             LD   HL,EDBUF
030A   01 0A 00             LD   BC,10
030D   ED B0                LDIR
030F   CD 12 09             CALL DISPRO
0312   CD 93 0C             CALL DL            ;auto line inc
0315   C3 8B 02             JP   PGM8
                    *
0318   06 64       PGMTL    LD   B,100
031A   21 8C 87             LD   HL,EDBUF
031D   3E 20                LD   A,20H
031F   77           PGMTL1   LD   (HL),A
0320   23                   INC  HL
0321   10 FC                DJNZ PGMTL1
                    *
0323   DD E5                PUSH IX
0325   E1                   POP  HL
0326   23                   INC  HL
0327   7E                   LD   A,(HL)
0328   4F                   LD   C,A
0329   06.00                LD   B,0
032B   23                   INC  HL
032C   11 8C 87             LD   DE,EDBUF
032F   ED B0                LDIR
0331   CD 66 07             CALL TLEDIT
```

```
                        *
0334   21 EF 87                 LD    HL,EDBUF+99
0337   06 64                    LD    B,100
0339   7E              FNEW     LD    A,(HL)
033A   FE 20                    CP    20H             ;blank
033C   20 05                    JR    NZ,FNEW1
033E   2B                       DEC   HL
033F   10 F8                    DJNZ  FNEW
0341   18 08                    JR    FNEW2
0343   11 8C 87         FNEW1   LD    DE,EDBUF
0346   AF                       XOR   A
0347   ED 52                    SBC   HL,DE
0349   23                       INC   HL
034A   7D                       LD    A,L
034B   32 7E 87         FNEW2   LD    (NEWNUM),A
                        *
034E   DD 7E 01                 LD    A,(IX+1)
0351   4F                       LD    C,A             ;old num char
0352   32 7D 87                 LD    (OLDNUM),A
0355   3A 7E 87                 LD    A,(NEWNUM)
0358   DD 77 01                 LD    (IX+1),A
035B   C6 02                    ADD   A,2
035D   FD 77 00                 LD    (IY+0),A
0360   3A 7E 87                 LD    A,(NEWNUM)
0363   B9                       CP    C
0364   28 4D                    JR    Z,TLEQA
0366   38 21                    JR    C,TLDEL
                        *
0368   DD E5            TLINS    PUSH  IX
036A   DD E5                    PUSH  IX
036C   01 02 00                 LD    BC,2
036F   DD 09                    ADD   IX,BC
0371   DD 22 7F 87              LD    (START),IX
0375   CD E8 09                 CALL  FNPRO
0378   22 81 87                 LD    (FINAL),HL
037B   3A 7D 87                 LD    A,(OLDNUM)
037E   4F                       LD    C,A
037F   3A 7E 87                 LD    A,(NEWNUM)
0382   91                       SUB   C
0383   47                       LD    B,A
0384   CD A1 09                 CALL  INSERT
0387   18 2E                    JR    TLMOV
                        *
0389   DD E5            TLDEL    PUSH  IX
038B   DD E5                    PUSH  IX
038D   3A 7D 87                 LD    A,(OLDNUM)
0390   4F                       LD    C,A
0391   0C                       INC   C
```

```
0392   06 00              LD    B,0
0394   DD 09              ADD   IX,BC
0396   DD 22 7F 87        LD    (START),IX
039A   CD E8 09           CALL  FNPRO
039D   22 81 87           LD    (FINAL),HL
03A0   3A 7E 87           LD    A,(NEWNUM)
03A3   4F                 LD    C,A
03A4   3A 7D 87           LD    A,(OLDNUM)
03A7   91                 SUB   C
03A8   47                 LD    B,A
03A9   3E 0C              LD    A,0CH          ;RE
03AB   32 7C 87           LD    (STOREY),A
03AE   CD C2 09           CALL  DELETE
03B1   18 04              JR    TLMOV
                  *
03B3   DD E5      TLEQA   PUSH  IX
03B5   DD E5              PUSH  IX
03B7   D1         TLMOV   POP   DE
03B8   13                 INC   DE
03B9   13                 INC   DE
03BA   21 8C 87           LD    HL,EDBUF
03BD   DD E1              POP   IX
03BF   DD 4E 01           LD    C,(IX+1)
03C2   06 00              LD    B,0
03C4   ED B0              LDIR
03C6   CD 12 09           CALL  DISPRO
03C9   CD 93 0C           CALL  DL             ;auto line inc
03CC   C3 8B 02           JP    PGM8
                  *
03CF   DD E5      PGMGR   PUSH  IX
03D1   E1                 POP   HL
03D2   23                 INC   HL
03D3   11 8C 87           LD    DE,EDBUF
03D6   01 50 00           LD    BC,BYTE
03D9   ED B0              LDIR
03DB   CD 4E 08           CALL  GREDIT
03DE   DD E5              PUSH  IX
03E0   D1                 POP   DE
03E1   13                 INC   DE
03E2   21 8C 87           LD    HL,EDBUF
03E5   01 50 00           LD    BC,BYTE
03E8   ED B0              LDIR
03EA   CD 12 09           CALL  DISPRO
03ED   CD 93 0C           CALL  DL             ;auto line inc
03F0   C3 8B 02           JP    PGM8
                  *
                  * ins key
                  *
```

```
03F3   DD 7E 00     PGM11    LD   A,(IX+0)
03F6   FE 0C                 CP   0CH          ;RE
03F8   CA E4 01              JP   Z,PGM6
03FB   CD 70 0A              CALL CL
03FE   CD FD 08              CALL NUM
0401   11 50 80              LD   DE,BUF
0404   78                    LD   A,B
0405   CD 89 0A              CALL TSS
0408   79                    LD   A,C
0409   CD 89 0A              CALL TSS
040C   CD FF 0A              CALL DX
040F   CD 1C 05     PGM11A   CALL SCANK
0412   3A 74 87              LD   A,(KEYIN)
0415   FE 0C                 CP   0CH          ;instruction
0417   30 F6                 JR   NC,PGM11A
0419   CD 6F 09              CALL PROBYT
                     *
041C   DD 22 7F 87            LD   (START),IX
0420   CD E8 09              CALL FNPRO
0423   22 81 87              LD   (FINAL),HL
0426   3A 7E 87              LD   A,(NEWNUM)
0429   47                    LD   B,A
042A   CD A1 09              CALL INSERT
042D   CD 05 0A              CALL FLBYTE
0430   FD 22 7F 87            LD   (START),IY
0434   22 81 87              LD   (FINAL),HL
0437   06 01                 LD   B,1
0439   CD A1 09              CALL INSERT
043C   C3 27 01              JP   PGM4
                     *
                     * del key
                     *
043F   DD 7E 00     PGM12    LD   A,(IX+0)
0442   FE 0C                 CP   0CH          ;RE
0444   CA E4 01              JP   Z,PGM6
0447   FD 7E 01              LD   A,(IY+1)
044A   FE 00                 CP   0
044C   20 15                 JR   NZ,PGM12B
044E   FD 46 00              LD   B,(IY+0)
0451   DD E5                 PUSH IX
0453   E1                    POP  HL
0454   3E 0C                 LD   A,0CH
0456   77           PGM12A   LD   (HL),A
0457   23                    INC  HL
0458   10 FC                 DJNZ PGM12A
045A   FD 70 00              LD   (IY+0),B     ;fill 1byte=0
045D   CD 12 09              CALL DISPRO
0460   C3 E4 01              JP   PGM6
```

```
0463   DD E5          PGM12B   PUSH IX
0465   FD 4E 00                LD   C,(IY+0)
0468   06 00                   LD   B,0
046A   DD 09                   ADD  IX,BC
046C   DD 22 7F 87             LD   (START),IX
0470   CD E8 09                CALL FNPRO
0473   22 81 87                LD   (FINAL),HL
0476   FD 46 00                LD   B,(IY+0)
0479   3E 0C                   LD   A,0CH          ;RE
047B   32 7C 87                LD   (STOREY),A
047E   CD C2 09                CALL DELETE
                       *
0481   FD 23                   INC  IY
0483   FD 22 7F 87             LD   (START),IY
0487   CD 05 0A                CALL FLBYTE
048A   22 81 87                LD   (FINAL),HL
048D   06 01                   LD   B,1
048F   AF                      XOR  A
0490   32 7C 87                LD   (STOREY),A
0493   CD C2 09                CALL DELETE
0496   FD 2B                   DEC  IY
0498   DD E1                   POP  IX
049A   CD 12 09                CALL DISPRO
049D   C3 E4 01                JP   PGM6
                       *
                       * clear all key
                       *
04A0   21 FF 87        PGM13    LD   HL,SYSFAG
04A3   CB 66                    BIT  4,(HL)
04A5   20 27                    JR   NZ,PGM13C
04A7   DD 7E 00                 LD   A,(IX+0)
04AA   FE 0C                    CP   0CH            ;01 RE
04AC   20 08                    JR   NZ,PGM13A
04AE   3E FF                    LD   A,0FFH
04B0   32 74 87                 LD   (KEYIN),A
04B3   C3 E4 01                 JP   PGM6
04B6   CD 70 0A        PGM13A   CALL CL
04B9   21 2C 0E                 LD   HL,CLRTAB
04BC   11 50 80                 LD   DE,BUF
04BF   06 07                    LD   B,7
04C1   7E              PGM13B   LD   A,(HL)
04C2   CD 89 0A                 CALL TSS
04C5   23                       INC  HL
04C6   10 F9                    DJNZ PGM13B
04C8   CD FF 0A                 CALL DX
04CB   C3 E4 01                 JP   PGM6
                       *
04CE   CB 76           PGM13C   BIT  6,(HL)
```

```
04D0   28 0E              JR    Z,PGM13D
04D2   21 00 40           LD    HL,PROA
04D5   CD 1E 0A           CALL  CLPRO
04D8   11 9B 0F           LD    DE,3995
04DB   CD 2B 0A           CALL  CLLBYT
04DE   18 0C              JR    PGM13E
04E0   21 9C 4F    PGM13D LD    HL,PROB
04E3   CD 1E 0A           CALL  CLPRO
04E6   11 63 00           LD    DE,99
04E9   CD 2B 0A           CALL  CLLBYT
                   *
04EC   21 FF 87    PGM13E LD    HL,SYSFAG
04EF   CB A6               RES   4,(HL)
04F1   C3 D9 00            JP    PGM
                   *
                   *
                   *
                   ***************************
                   *                         *
                   *        RUN MAIN         *
                   *                         *
                   ***************************
                   *
04F4   CD 70 0A    RUN     CALL  CL
04F7   CD FF 0A            CALL  DX
                   *
04FA   21 FF 87    RUN1    LD    HL,SYSFAG
04FD   CB 76               BIT   6,(HL)
04FF   28 05               JR    Z,RUN2
0501   21 00 40           LD    HL,PROA       ;b6=1
0504   18 03              JR    RUN3
0506   21 9C 4F    RUN2    LD    HL,PROB       ;b6=0
0509   22 75 87    RUN3    LD    (ADDR),HL
050C   CD 34 0A           CALL  RUNS
050F   18 E9              JR    RUN1
                   *
                   *
                   *
                   ***************************
                   *                         *
                   *   DLSCAN SUBROUTINE     *
                   *                         *
                   ***************************
                   *
0511   32 71 87    DLS     LD    (COUNT),A
0514   3A 71 87    DLS1    LD    A,(COUNT)
0517   FE 00              CP    0
0519   20 F9              JR    NZ,DLS1
```

```
051B   C9                        RET
                          *
                          *
                          *
                          ************************
                          *                      *
                          *   SCANK SUBROUTINE   *
                          *                      *
                          ************************
                          *
051C   01 00 08    SCANK     LD   BC,800H
051F   50                    LD   D,B
0520   79          SCANK1    LD   A,C
0521   D3 02                 OUT  (PKEYB),A
0523   3E 20                 LD   A,20H
0525   3D          SCANK2    DEC  A
0526   20 FD                 JR   NZ,SCANK2
0528   DB 02                 IN   A,(PKEYB)
052A   E6 70                 AND  70H
052C   FE 70                 CP   70H
052E   20 1A                 JR   NZ,SCANK4
0530   15                    DEC  D
0531   20 12                 JR   NZ,SCANK3
0533   21 FF 87              LD   HL,SYSFAG    ;rlease key
0536   CB 86                 RES  0,(HL)
                          *
0538   E5                    PUSH HL
0539   21 45 05              LD   HL,SCANK3
053C   E3                    EX   (SP),HL
053D   21 FF 87              LD   HL,SYSFAG
0540   CB EE                 SET  5,(HL)
0542   C3 67 05              JP   CKDIF
                          *
0545   0C          SCANK3    INC  C
0546   10 D8                 DJNZ SCANK1
0548   18 D2                 JR   SCANK
                          *
                          * look table
                          *
054A   21 FF 87    SCANK4    LD   HL,SYSFAG
054D   CB 46                 BIT  0,(HL)
054F   20 F4                 JR   NZ,SCANK3    ;no rlease
0551   B1                    OR   C
0552   21 6B 0D              LD   HL,KEYTAB+22
0555   06 16                 LD   B,16H
0557   BE          SCANK5    CP   (HL)
0558   28 03                 JR   Z,SCANK6
055A   2B                    DEC  HL
```

```
055B   10 FA                    DJNZ SCANK5
                        *
                        * found
                        *
055D   78               SCANK6  LD   A,B
055E   32 74 87                 LD   (KEYIN),A
0561   21 FF 87                 LD   HL,SYSFAG
0564   CB C6                    SET  0,(HL)
0566   C9                       RET
                        *
                        *
                        *
                        ***************************
                        *                         *
                        *   CHECK DIP SW SUB.     *
                        *                         *
                        ***************************
                        *
0567   21 FF 87         CKDIP   LD   HL,SYSFAG
056A   56                       LD   D,(HL)        ;save sysfag
                        *
                        * dip sw.-pro a/b
                        *
056B   3E 09            CKDIP1  LD   A,9
056D   D3 02                    OUT  (PKEYB),A
056F   3E 10                    LD   A,10H
0571   3D               CKDIP2  DEC  A
0572   20 FD                    JR   NZ,CKDIP2
0574   DB 02                    IN   A,(PKEYB)
0576   E6 70                    AND  70H
0578   FE 70                    CP   70H
057A   28 04                    JR   Z,CKDIP3
057C   CB F6                    SET  6,(HL)        ;proa,b6=1
057E   18 02                    JR   CKDIP4
0580   CB B6            CKDIP3  RES  6,(HL)        ;prob,b6=0
                        *
                        * dip sw.-pro/run
                        *
0582   3E 08            CKDIP4  LD   A,8
0584   D3 02                    OUT  (PKEYB),A
0586   3E 10                    LD   A,10H
0588   3D               CKDIP5  DEC  A
0589   20 FD                    JR   NZ,CKDIP5
058B   DB 02                    IN   A,(PKEYB)
058D   E6 70                    AND  70H
058F   FE 70                    CP   70H
0591   28 04                    JR   Z,CKDIP6
0593   CB FE                    SET  7,(HL)        ;pro,b7=1
```

```
0595   18 02              JR    CKDIP7
0597   CB BE     CKDIP6   RES   7,(HL)        ;run,b7=0
0599   7E        CKDIP7   LD    A,(HL)
059A   BA                 CP    D
059B   C8                 RET   Z
                  *
059C   CB 6E               BIT   5,(HL)
059E   28 06               JR    Z,CKDIPX      ;b5=0(call)
05A0   E1                  POP   HL            ;b5=1(jump)
05A1   21 FF 87            LD    HL,SYSFAG
05A4   CB AE               RES   5,(HL)
05A6   CB 7E     CKDIPX    BIT   7,(HL)
05A8   20 05               JR    NZ,CKDIP8
05AA   21 F4 04            LD    HL,RUN        ;b7=0->run
05AD   18 03               JR    CKDIP9
05AF   21 D9 00  CKDIP8    LD    HL,PGM        ;b7=1->pro
05B2   E3        CKDIP9    EX    (SP),HL
05B3   C9                  RET
                  *
                  *
                  *
                  **************************
                  *                        *
                  *   SCANKE SUBROUTINE    *
                  *                        *
                  **************************
                  *
05B4   01 00 08   SCANKE    LD    BC,800H
05B7   50                  LD    D,B
05B8   79         SCANKE1   LD    A,C
05B9   D3 02               OUT   (PKEYB),A
                  *
05BB   1E 08               LD    E,8
05BD   3A F5 87            LD    A,(DOTFAG)
05C0   FE 01               CP    1
05C2   20 02               JR    NZ,SCANKE2
05C4   1E 02               LD    E,2
05C6   1D         SCANKE2  DEC   E
05C7   20 FD               JR    NZ,SCANKE2
                  *
05C9   DB 02               IN    A,(PKEYB)
05CB   E6 70               AND   70H
05CD   FE 70               CP    70H
05CF   20 15               JR    NZ,SCANKE4
05D1   15                  DEC   D
05D2   20 0A               JR    NZ,SCANKE3
05D4   21 FF 87            LD    HL,SYSFAG     ;release
05D7   CB 86               RES   0,(HL)
```

```
05D9   3E 00            LD   A,0          ;speed1
05DB   32 F4 87         LD   (AUTORE),A
05DE   0C         SCANKE3  INC  C
05DF   10 D7            DJNZ SCANKE1
05E1   CD 27 06         CALL CURSOR
05E4   18 CE            JR   SCANKE
                  *
                  * look table
                  *
05E6   21 FF 87   SCANKE4  LD   HL,SYSFAG
05E9   CB 46            BIT  0,(HL)
05EB   20 16            JR   NZ,SCANKE7   ;no release
05ED   B1               OR   C
05EE   21 6B 0D         LD   HL,KEYTAB+22
05F1   06 16            LD   B,22
05F3   BE         SCANKE5  CP   (HL)
05F4   28 03            JR   Z,SCANKE6
05F6   2B               DEC  HL
05F7   10 FA            DJNZ SCANKE5
                  *
05F9   78         SCANKE6  LD   A,B          ;found
05FA   32 74 87         LD   (KEYIN),A
05FD   21 FF 87         LD   HL,SYSFAG
0600   CB C6            SET  0,(HL)
0602   C9               RET                ;exit
                  *
                  * check auto-repeat
                  *
0603   3A 74 87   SCANKE7  LD   A,(KEYIN)
0606   FE 0F            CP   0FH
0608   28 0E            JR   Z,SCANKE8
060A   FE 12            CP   12H
060C   28 0A            JR   Z,SCANKE8
060E   FE 13            CP   13H
0610   28 06            JR   Z,SCANKE8
0612   FE 16            CP   16H
0614   28 02            JR   Z,SCANKE8
0616   18 C6            JR   SCANKE3
0618   21 F4 87   SCANKE8  LD   HL,AUTORE
061B   35               DEC  (HL)
061C   20 C0            JR   NZ,SCANKE3
061E   3E 60            LD   A,60H        ;speed2
0620   32 F4 87         LD   (AUTORE),A
0623   3A 74 87         LD   A,(KEYIN)
0626   C9               RET               ;exit auto
                  *
                  * cursor flash
                  *
```

```
0627   3A F1 87     CURSOR   LD    A,(CURCNT)
062A   3D                    DEC   A
062B   32 F1 87              LD    (CURCNT),A
062E   C0                    RET   NZ
                    *
062F   3E 40                 LD    A,40H          ;flash speed
0631   32 F1 87              LD    (CURCNT),A
0634   3A F5 87              LD    A,(DOTFAG)
0637   FE 01                 CP    1
0639   28 2F                 JR    Z,CURDOT
                    *
063B   3A F0 87              LD    A,(CURNUM)
063E   6F                    LD    L,A
063F   26 00                 LD    H,0
0641   29                    ADD   HL,HL
0642   29                    ADD   HL,HL
0643   29                    ADD   HL,HL
0644   11 00 80              LD    DE,DIS
0647   19                    ADD   HL,DE
                    *
0648   06 07                 LD    B,7
064A   3A F2 87              LD    A,(CURFAG)
064D   FE 00                 CP    0
064F   28 0C                 JR    Z,CURSOR2
0651   7E           CURSOR1  LD    A,(HL)         ;set line
0652   F6 01                 OR    01H
0654   77                    LD    (HL),A
0655   23                    INC   HL
0656   10 F9                 DJNZ  CURSOR1
0658   AF                    XOR   A
0659   32 F2 87              LD    (CURFAG),A
065C   C9                    RET
                    *
065D   7E           CURSOR2  LD    A,(HL)
065E   E6 FE                 AND   0FEH
0660   77                    LD    (HL),A
0661   23                    INC   HL
0662   10 F9                 DJNZ  CURSOR2
0664   3E 01                 LD    A,1
0666   32 F2 87              LD    (CURFAG),A
0669   C9                    RET
                    *
066A   16 01        CURDOT   LD    D,1
066C   3A F7 87              LD    A,(DOTY)
066F   3C                    INC   A
0670   3D           CURDOT1  DEC   A
0671   28 04                 JR    Z,CURDOT2
0673   CB 02                 RLC   D
```

```
0675   18 F9              JR    CURDOT1      ;D
                    *
0677   3A F6 87    CURDOT2  LD    A,(DOTX)
067A   6F                  LD    L,A
067B   26 00               LD    H,0
067D   01 00 80            LD    BC,DIS
0680   09                  ADD   HL,BC        ;HL
                    *
0681   3A F2 87             LD    A,(CURFAG)
0684   FE 00                CP    0
0686   28 08                JR    Z,CURDOT3
0688   7E                  LD    A,(HL)       ;set dot
0689   B2                  OR    D
068A   77                  LD    (HL),A
068B   AF                  XOR   A
068C   32 F2 87            LD    (CURFAG),A
068F   C9                  RET
                    *
0690   7A          CURDOT3  LD    A,D          ;clr dot
0691   2F                  CPL
0692   A6                  AND   (HL)
0693   77                  LD    (HL),A
0694   3E 01               LD    A,1
0696   32 F2 87            LD    (CURFAG),A
0699   C9                  RET
                    *
                    *
                    *
                    **************************
                    *                        *
                    *     TEXT SCREEN EDIT   *
                    *                        *
                    **************************
                    *
069A   AF          TSEDIT   XOR   A
069B   32 F0 87             LD    (CURNUM),A
069E   32 F5 87             LD    (DOTFAG),A
                    *
06A1   32 FC 87             LD    (CHRKEY),A
06A4   3E FF                LD          A,0FFH
06A6   32 FD 87             LD    (OLDKEY),A
06A9   21 A1 0D             LD    HL,UPCTAB    ;default uppercase
06AC   22 F8 87             LD    (CASTAB),HL
                    *
06AF   06 0A       TSE1     LD    B,10
06B1   21 8C 87             LD    HL,EDBUF
06B4   11 00 80             LD    DE,DIS
06B7   7E          TSE2     LD    A,(HL)
```

```
06B8   CD 89 0A            CALL TSS
06BB   23                  INC  HL
06BC   10 F9               DJNZ TSE2
                      *
06BE   CD B4 05     TSE3   CALL SCANKE
06C1   FE 0C               CP   0CH
06C3   38 5D               JR   C,TSE8
06C5   FE 16               CP   16H          ;->
06C7   28 39               JR   Z,TSE6
06C9   FE 0F               CP   0FH          ;<-
06CB   28 45               JR   Z,TSE7
06CD   FE 0E               CP   0EH          ;change
06CF   28 19               JR   Z,TSE5
06D1   FE 0D               CP   0DH          ;clear all
06D3   28 05               JR   Z,TSE4
06D5   FE 0C               CP   0CH          ;edit
06D7   C8                  RET  Z
06D8   18 E4               JR   TSE3
                      *
06DA   32 FD 87     TSE4   LD   (OLDKEY),A
06DD   21 8C 87            LD   HL,EDBUF
06E0   06 0A               LD   B,10
06E2   3E 20               LD   A,20H        ;blank
06E4   77           TSE41  LD   (HL),A
06E5   23                  INC  HL
06E6   10 FC               DJNZ TSE41
06E8   18 C5               JR   TSE1
                      *
06EA   2A F8 87     TSE5   LD   HL,(CASTAB)
06ED   11 D1 0D            LD   DE,LWCTAB
06F0   AF                  XOR  A
06F1   ED 52               SBC  HL,DE
06F3   28 05               JR   Z,TSE51
06F5   21 D1 0D            LD   HL,LWCTAB
06F8   18 03               JR   TSE52
06FA   21 A1 0D     TSE51  LD   HL,UPCTAB
06FD   22 F8 87     TSE52  LD   (CASTAB),HL
0700   18 AD               JR   TSE1
                      *
0702   32 FD 87     TSE6   LD   (OLDKEY),A
0705   3A F0 87           LD   A,(CURNUM)
0708   FE 09              CP   9
070A   28 B2              JR   Z,TSE3
070C   3C                 INC  A
070D   32 F0 87           LD   (CURNUM),A
0710   18 9D              JR   TSE1
                      *
0712   32 FD 87     TSE7  LD   (OLDKEY),A
```

```
0715   3A F0 87              LD    A,(CURNUM)
0718   FE 00                 CP    0
071A   28 A2                 JR    Z,TSE3
071C   3D                   DEC   A
071D   32 F0 87              LD    (CURNUM),A
0720   18 8D                 JR    TSE1
                     *
0722   6F            TSE8    LD    L,A
0723   26 00                 LD    H,0
0725   29                    ADD   HL,HL          ;*4
0726   29                    ADD   HL,HL          ;
0727   E5                    PUSH  HL
0728   D1                    POP   DE
0729   2A F8 87              LD    HL,(CASTAB)
072C   19                    ADD   HL,DE
072D   22 FA 87              LD    (CHRTAB),HL    ;set chrtab
                     *
0730   21 8C 87              LD    HL,EDBUF
0733   3A F0 87              LD    A,(CURNUM)
0736   5F                    LD    E,A
0737   16 00                 LD    D,0
0739   19                    ADD   HL,DE
073A   E5                    PUSH  HL             ;char position in edbuf
                     *
073B   3A FD 87              LD    A,(OLDKEY)     ;last key
073E   4F                    LD    C,A
073F   3A 74 87              LD    A,(KEYIN)      ;current key
0742   B9                    CP    C
0743   20 0A                 JR    NZ,TSE81
0745   3A FC 87              LD    A,(CHRKEY)
0748   FE 03                 CP    3
074A   28 03                 JR    Z,TSE81        ;4th position
074C   3C                    INC   A              ;inc position
074D   18 01                 JR    TSE82
074F   AF            TSE81   XOR   A
0750   32 FC 87      TSE82   LD          (CHRKEY),A   ;1st position
0753   5F                    LD    E,A
0754   16 00                 LD    D,0
0756   2A FA 87              LD    HL,(CHRTAB)
0759   19                    ADD   HL,DE
075A   7E                    LD    A,(HL)         ;real ascii code
075B   E1                    POP   HL
075C   77                    LD    (HL),A
075D   3A 74 87              LD    A,(KEYIN)      ;update oldkey
0760   32 FD 87              LD    (OLDKEY),A
0763   C3 AF 06              JP    TSE1
                     *
                     *
```

```
                        *
                        ************************
                        *                      *
                        *     TEXT LINE EDIT   *
                        *                      *
                        ************************
                        *
0766   06 08            TLEDIT  LD   B,8
0768   3E 2E                    LD   A,2EH
076A   21 83 87                 LD   HL,SPACE
076D   77                       LD   (HL),A
076E   23                       INC  HL
076F   3E 20                    LD   A,20H
0771   77               TLE1    LD   (HL),A
0772   23                       INC  HL
0773   10 FC                    DJNZ TLE1
0775   AF                       XOR  A
0776   32 F3 87                 LD   (CHRCNT),A
0779   32 F5 87                 LD   (DOTFAG),A
077C   3E 09                    LD   A,9
077E   32 F0 87                 LD   (CURNUM),A   ;cursor
                        *
0781   AF                       XOR  A
0782   32 FC 87                 LD   (CHRKEY),A
0785   3E FF                    LD   A,0FFH
0787   32 FD 87                 LD   (OLDKEY),A
078A   21 A1 0D                 LD   HL,UPCTAB    ;default uppercase
078D   22 F8 87                 LD   (CASTAB),HL
                        *
0790   3A F3 87        TLE2     LD   A,(CHRCNT)
0793   6F                       LD   L,A
0794   26 00                    LD   H,0
0796   11 83 87                 LD   DE,SPACE
0799   19                       ADD  HL,DE        ;start data
079A   06 0A                    LD   B,10
079C   11 00 80                 LD   DE,DIS
079F   7E               TLE21    LD   A,(HL)
07A0   CD B9 0A                 CALL TSS
07A3   23                       INC  HL
07A4   10 F9                    DJNZ TLE21
                        *
07A6   CD B4 05         TLE3     CALL SCANKE
07A9   FE 0C                    CP   0CH          ;instr
07AB   38 5D                    JR   C,TLE8
07AD   FE 16                    CP   16H          ;->
07AF   28 39                    JR   Z,TLE6
07B1   FE 0F                    CP   0FH          ;<-
07B3   28 45                    JR   Z,TLE7
```

```
07B5   FE 0E                    CP    0EH          ;change
07B7   28 19                    JR    Z,TLE5
07B9   FE 0D                    CP    0DH          ;clear all
07BB   28 05                    JR    Z,TLE4
07BD   FE 0C                    CP    0CH          ;edit
07BF   C8                       RET   Z
07C0   18 E4                    JR    TLE3
                        *
07C2   32 FD 87         TLE4    LD    (OLDKEY),A
07C5   21 8C 87                 LD    HL,EDBUF
07C8   06 64                    LD    B,100
07CA   3E 20                    LD    A,20H
07CC   77               TLE41   LD    (HL),A
07CD   23                       INC   HL
07CE   10 FC                    DJNZ  TLE41
07D0   18 BE                    JR    TLE2
                        *
07D2   2A F8 87         TLE5    LD    HL,(CASTAB)
07D5   11 D1 0D                 LD    DE,LWCTAB
07D8   AF                       XOR   A
07D9   ED 52                    SBC   HL,DE
07DB   28 05                    JR    Z,TLE51
07DD   21 D1 0D                 LD    HL,LWCTAB
07E0   18 03                    JR    TLE52
07E2   21 A1 0D         TLE51    LD    HL,UPCTAB
07E5   22 F8 87         TLE52   LD    (CASTAB),HL
07E8   18 A6                    JR    TLE2
                        *
07EA   32 FD 87         TLE6    LD    (OLDKEY),A
07ED   3A F3 87                 LD    A,(CHRCNT)
07F0   FE 63                    CP    99
07F2   28 B2                    JR    Z,TLE3
07F4   3C                       INC   A
07F5   32 F3 87                 LD    (CHRCNT),A
07F8   18 96                    JR    TLE2
                        *
07FA   32 FD 87         TLE7    LD    (OLDKEY),A
07FD   3A F3 87                 LD    A,(CHRCNT)
0800   FE 00                    CP    0
0802   28 A2                    JR    Z,TLE3
0804   3D                       DEC   A
0805   32 F3 87                 LD    (CHRCNT),A
0808   18 86                    JR    TLE2
                        *
080A   6F               TLE8    LD    L,A
080B   26 00                    LD    H,0
080D   29                       ADD   HL,HL        ;*4
080E   29                       ADD   HL,HL        ;
```

```
080F   E5                  PUSH HL
0810   D1                  POP  DE
0811   2A F8 87            LD   HL,(CASTAB)
0814   19                  ADD  HL,DE
0815   22 FA 87            LD   (CHRTAB),HL  ;set chrtab
                    *
0818   21 8C 87            LD   HL,EDBUF
081B   3A F3 87            LD   A,(CHRCNT)
081E   5F                  LD   E,A
081F   16 00               LD   D,0
0821   19                  ADD  HL,DE
0822   E5                  PUSH HL           ;char position in edbuf
                    *
0823   3A FD 87            LD   A,(OLDKEY)   ;last key
0826   4F                  LD   C,A
0827   3A 74 87            LD   A,(KEYIN)    ;current key
082A   B9                  CP   C
082B   20 0A               JR   NZ,TLE81
082D   3A FC 87            LD   A,(CHRKEY)
0830   FE 03               CP   3
0832   28 03               JR   Z,TLE81      ;4th position
0834   3C                  INC  A            ;inc position
0835   18 01               JR   TLE82
0837   AF            TLE81 XOR  A
0838   32 FC 87      TLE82 LD         (CHRKEY),A   ;1st position
083B   5F                  LD   E,A
083C   16 00               LD   D,0
083E   2A FA 87            LD   HL,(CHRTAB)
0841   19                  ADD  HL,DE
0842   7E                  LD   A,(HL)       ;real ascii code
0843   E1                  POP  HL
0844   77                  LD   (HL),A
0845   3A 74 87            LD   A,(KEYIN)    ;update oldkey
0848   32 FD 87            LD   (OLDKEY),A
084B   C3 90 07            JP   TLE2
                    *
                    *
                    *
                    ************************
                    *                      *
                    *      GRAPHIC EDIT    *
                    *                      *
                    ************************
                    *
084E   3E 01        GREDIT  LD   A,1
0850   32 F5 87            LD   (DOTFAG),A
0853   AF                  XOR  A
0854   32 F6 87            LD   (DOTX),A
```

```
0857   32 F7 87            LD    (DOTY),A
                     *
085A   21 8C 87     GRE1    LD    HL,EDBUF
085D   11 00 80             LD    DE,DIS
0860   01 50 00             LD    BC,80
0863   ED B0                LDIR
                     *
0865   CD B4 05     GRE2    CALL  SCANKE
0868   FE 16                CP    16H          ;->
086A   28 4B                JR    Z,GRE5
086C   FE 0F                CP    0FH          ;<-
086E   28 54                JR    Z,GRE6
0870   FE 12                CP    12H          ;^
0872   28 5D                JR    Z,GRE7
0874   FE 13                CP    13H          ;v
0876   28 67                JR    Z,GRE8
0878   FE 0E                CP    0EH          ;change
087A   28 18                JR    Z,GRE4
087C   FE 0D                CP    0DH          ;clear all
087E   28 05                JR    Z,GRE3
0880   FE 0C                CP    0CH          ;edit
0882   C8                   RET   Z
0883   18 E0                JR    GRE2
                     *
0885   21 8C 87     GRE3    LD    HL,EDBUF
0888   11 8D 87             LD    DE,EDBUF+1
088B   01 4F 00             LD    BC,79
088E   AF                   XOR   A
088F   77                   LD    (HL),A
0890   ED B0                LDIR
0892   18 C6                JR    GRE1
                     *
0894   16 01        GRE4    LD    D,1
0896   3A F7 87             LD    A,(DOTY)
0899   3C                   INC   A
089A   3D           GRE41   DEC   A
089B   28 04                JR    Z,GRE42
089D   CB 02                RLC   D
089F   18 F9                JR    GRE41        ;D
08A1   3A F6 87     GRE42   LD    A,(DOTX)
08A4   6F                   LD    L,A
08A5   26 00                LD    H,0
08A7   01 8C 87             LD    BC,EDBUF
08AA   09                   ADD   HL,BC        ;HL
                     *
08AB   7A                   LD    A,D
08AC   A6                   AND   (HL)         ;1 bit
08AD   2F                   CPL
```

```
08AE   A2                        AND  D            ;1 bit
08AF   5F                        LD   E,A
08B0   7A                        LD   A,D
08B1   2F                        CPL
08B2   A6                        AND  (HL)         ;clear bit
08B3   B3                        OR   E            ;new bit
08B4   77                        LD   (HL),A
08B5   18 A3                     JR   GRE1
                    *
08B7   3A F6 87     GRE5         LD   A,(DOTX)
08BA   FE 4F                     CP   79
08BC   28 A7                     JR   Z,GRE2
08BE   3C                        INC  A
08BF   32 F6 87                  LD   (DOTX),A
08C2   18 96                     JR   GRE1
                    *
08C4   3A F6 87     GRE6         LD   A,(DOTX)
08C7   FE 00                     CP   0
08C9   28 9A                     JR   Z,GRE2
08CB   3D                        DEC  A
08CC   32 F6 87                  LD   (DOTX),A
08CF   18 89                     JR   GRE1
                    *
08D1   3A F7 87     GRE7         LD   A,(DOTY)
08D4   FE 07                     CP   7
08D6   28 8D                     JR   Z,GRE2
08D8   3C                        INC  A
08D9   32 F7 87                  LD   (DOTY),A
08DC   C3 5A 08                  JP   GRE1
                    *
08DF   3A F7 87     GRE8         LD   A,(DOTY)
08E2   FE 00                     CP   0
08E4   CA 65 08                  JP   Z,GRE2
08E7   3D                        DEC  A
08E8   32 F7 87                  LD   (DOTY),A
08EB   C3 5A 08                  JP   GRE1
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *     ISC SUBROUTINE      *
                    *                         *
                    ***************************
                    *
                    * change op-code to ascii
                    * 3 character
                    *
```

```
08EE   6F           ISC      LD   L,A
08EF   26 00                 LD   H,0
08F1   29                    ADD  HL,HL        ;*3
08F2   5F                    LD   E,A          ;
08F3   16 00                 LD   D,0          ;
08F5   19                    ADD  HL,DE        ;
08F6   E5                    PUSH HL
08F7   D1                    POP  DE
08F8   21 76 0D              LD   HL,ISCTAB
08FB   19                    ADD  HL,DE
08FC   C9                    RET
                     *
                     *
                     *
                     ************************
                     *                      *
                     *     NUM SUBROUTINE   *
                     *                      *
                     ************************
                     *
                     * change line num. to ascii code
                     *
08FD   3A 77 87      NUM     LD   A,(LINE)
0900   E6 F0                AND  0F0H
0902   0F                    RRCA
0903   0F                    RRCA
0904   0F                    RRCA
0905   0F                    RRCA
0906   F6 30                 OR   30H
0908   47                    LD   B,A
0909   3A 77 87              LD   A,(LINE)
090C   E6 0F                AND  0FH
090E   F6 30                 OR   30H
0910   4F                    LD   C,A
0911   C9                    RET
                     *
                     *
                     *
                     ************************
                     *                      *
                     *   DISPRO SUBROUTINE  *
                     *                      *
                     ************************
                     *
                     * display program 1 line
                     * and instruction
                     *
0912   CD 70 0A      DISPRO  CALL CL           ;clear buf
```

```
0915   3A 77 87            LD    A,(LINE)
0918   CD FD 08            CALL  NUM
091B   11 50 80            LD    DE,BUF
091E   78                  LD    A,B
091F   CD 89 0A            CALL  TSS
0922   79                  LD    A,C
0923   CD 89 0A            CALL  TSS
                   *
0926   DD 7E 00            LD    A,(IX+0)
0929   CD EE 08            CALL  ISC
092C   11 68 80            LD    DE,BUF+24
092F   06 03               LD    B,3
0931   7E          DISPR1  LD    A,(HL)
0932   CD 89 0A            CALL  TSS
0935   23                  INC   HL
0936   10 F9               DJNZ  DISPR1
                   *
0938   FD 7E 00            LD    A,(IY+0)
093B   FE 02               CP    02H             ;2 byte
093D   20 13               JR    NZ,DISPR2
093F   DD 7E 01            LD    A,(IX+1)
0942   FE 00               CP    0               ;U
0944   28 10               JR    Z,DISPR3
0946   FE 01               CP    1               ;D
0948   28 11               JR    Z,DISPR4
094A   FE 02               CP    2               ;L
094C   28 12               JR    Z,DISPR5
094E   FE 03               CP    3               ;R
0950   28 13               JR    Z,DISPR6
0952   CD FF 0A    DISPR2  CALL  DX              ;display
0955   C9                  RET
                   *
0956   3A 9D 0D    DISPR3  LD    A,(ISCTAB+39);up
0959   18 0D               JR    DISPR7
095B   3A 9E 0D    DISPR4  LD    A,(ISCTAB+40);down
095E   18 08               JR    DISPR7
0960   3A 9F 0D    DISPR5  LD    A,(ISCTAB+41);left
0963   18 03               JR    DISPR7
0965   3A A0 0D    DISPR6  LD    A,(ISCTAB+42);right
                   *
0968   CD 89 0A    DISPR7  CALL  TSS
096B   CD FF 0A            CALL  DX              ;display
096E   C9                  RET
                   *
                   *
                   *
                   ************************
                   *                      *
```

```
                        *   PROBYTE SUBROUTINE  *
                        *                       *
                        *************************
                        *
096F   FE 01            PROBYT  CP   01H        ;TS-
0971   28 1C                    JR   Z,PROBY1
0973   FE 02                    CP   02H        ;TL-
0975   28 1C                    JR   Z,PROBY2
0977   FE 03                    CP   03H        ;GR-
0979   28 1C                    JR   Z,PROBY3
097B   FE 05                    CP   05H        ;LD,
097D   28 1C                    JR   Z,PROBY4
097F   FE 06                    CP   06H        ;DR,
0981   28 18                    JR   Z,PROBY4
0983   FE 07                    CP   07H        ;SH,
0985   28 14                    JR   Z,PROBY4
0987   FE 0B                    CP   0BH        ;RT,
0989   28 10                    JR   Z,PROBY4
                        *
                        * defind lbyte
                        *
098B   3E 01                    LD   A,1
098D   18 0E                    JR   PROBY5
098F   3E 0B            PROBY1  LD   A,11
0991   18 0A                    JR   PROBY5
0993   3E 03            PROBY2  LD   A,3
0995   18 06                    JR   PROBY5
0997   3E 51            PROBY3  LD   A,81
0999   18 02                    JR   PROBY5
099B   3E 02            PROBY4  LD   A,2
099D   32 7E 87          PROBY5  LD   (NEWNUM),A
09A0   C9                       RET
                        *
                        *
                        *
                        *************************
                        *                       *
                        *   INSERT SUBROUTINE   *
                        *                       *
                        *************************
                        *
                        * B = byte insert
                        * (START) = start addr
                        * (FINAL) = final addr
                        *
09A1   C5               INSERT  PUSH BC
09A2   48                       LD   C,B
09A3   06 00                    LD   B,0
```

```
09A5   2A 81 87            LD   HL,(FINAL)
09A8   ED 5B 7F 87         LD   DE,(START)
09AC   AF                  XOR  A
09AD   ED 52               SBC  HL,DE
09AF   23                  INC  HL
09B0   E5                  PUSH HL
09B1   2A 81 87            LD   HL,(FINAL)
09B4   09                  ADD  HL,BC
09B5   EB                  EX   DE,HL
09B6   2A 81 87            LD   HL,(FINAL)
09B9   C1                  POP  BC
09BA   ED B8               LDDR
                    *
09BC   C1                  POP  BC
09BD   12           INSER1 LD   (DE),A
09BE   1B                  DEC  DE
09BF   10 FC               DJNZ INSER1
09C1   C9                  RET
                    *
                    *
                    *
                    ************************
                    *                      *
                    *   DELETE SUBROUTINE  *
                    *                      *
                    ************************
                    *
                    * B = byte delete
                    * (START) = start addr
                    * (FINAL) = final addr
                    *
09C2   C5           DELETE PUSH BC
09C3   48                  LD   C,B
09C4   06 00               LD   B,0
09C6   2A 81 87            LD   HL,(FINAL)
09C9   ED 5B 7F 87         LD   DE,(START)
09CD   AF                  XOR  A
09CE   ED 52               SBC  HL,DE
09D0   23                  INC  HL
09D1   E5                  PUSH HL
09D2   2A 7F 87            LD   HL,(START)
09D5   AF                  XOR  A
09D6   ED 42               SBC  HL,BC
09D8   EB                  EX   DE,HL
09D9   2A 7F 87            LD   HL,(START)
09DC   C1                  POP  BC
09DD   ED B0               LDIR
                    *
```

```
09DF   C1              POP  BC
09E0   3A 7C 87        LD   A,(STOREY)
09E3   12       DELET1 LD   (DE),A
09E4   13              INC  DE
09E5   10 FC           DJNZ DELET1
09E7   C9              RET
                *
                *
                *
                ************************
                *                      *
                *   FNPRO SUBROUTINE   *
                *                      *
                ************************
                *
                * find final addr in pro a/b
                *
09E8   21 FF 87 FNPRO  LD   HL,SYSFAG
09EB   CB 76           BIT  6,(HL)
09ED   28 05           JR   Z,FNPRO1
09EF   21 9B 4F        LD   HL,PROA+3995 ;b6=1
09F2   18 03           JR   FNPRO2
09F4   21 37 5F FNPRO1 LD   HL,PROB+3995 ;b6=0
09F7   01 9C 0F FNPRO2 LD   BC,3996
09FA   7E       FNPRO3 LD   A,(HL)
09FB   FE 0C           CP   0CH          ;RE
09FD   C0              RET  NZ
09FE   2B              DEC  HL
09FF   0B              DEC  BC
0A00   78              LD   A,B
0A01   B1              OR   C
0A02   20 F6           JR   NZ,FNPRO3
0A04   C9              RET
                *
                *
                *
                ************************
                *                      *
                *   FLBYTE SUBROUTINE  *
                *                      *
                ************************
                *
                * find final addr in lbyte a/b
                *
0A05   21 FF 87 FLBYTE LD   HL,SYSFAG
0A08   CB 76           BIT  6,(HL)
0A0A   28 05           JR   Z,FLBYT1
0A0C   21 9B 5F        LD   HL,LBYTEA+99 ;b6=1
```

```
0A0F   18 03              JR    FLBYT2
0A11   21 FF 5F  FLBYT1   LD    HL,LBYTEB+99 ;b6=0
0A14   06 64     FLBYT2   LD    B,100
0A16   7E        FLBYT3   LD    A,(HL)
0A17   FE 00              CP    0
0A19   C0                 RET   NZ
0A1A   2B                 DEC   HL
0A1B   10 F9              DJNZ  FLBYT3
0A1D   C9                 RET
                *
                *
                *
                ************************
                *                      *
                *   CLPRO SUBROUTINE   *
                *                      *
                ************************
                *
                * clear working area of pro
                *
0A1E   01 9C 0F  CLPRO    LD    BC,3996
0A21   16 0C              LD    D,0CH
0A23   72        CLPRO1   LD    (HL),D
0A24   23                 INC   HL
0A25   0B                 DEC   BC
0A26   78                 LD    A,B
0A27   B1                 OR    C
0A28   20 F9              JR    NZ,CLPRO1
0A2A   C9                 RET
                *
                *
                *
                ************************
                *                      *
                *  CLLBYT SUBROUTINE   *
                *                      *
                ************************
                *
                * clear working area of lbyte
                *
0A2B   19        CLLBYT   ADD   HL,DE
0A2C   06 64              LD    B,100
0A2E   AF                 XOR   A
0A2F   77        CLLBY1   LD    (HL),A
0A30   23                 INC   HL
0A31   10 FC              DJNZ  CLLBY1
0A33   C9                 RET
                *
```

```
                        *
                        *
                        ************************
                        *                      *
                        *    RUNS SUBROUTINE   *
                        *                      *
                        ************************
                        *
                        * run banner program
                        * start at (ADDR)
                        * and end if command is RE (OCH)
                        * or error command
                        *
0A34   CD 67 05         RUNS    CALL CKDIP
0A37   CD 4F 0A                 CALL NEXT
0A3A   FE 0C                    CP   0CH
0A3C   D0                       RET  NC             ;exit
0A3D   21 3D 0D                 LD   HL,RUNTAB
0A40   87                       ADD  A,A ;*2
0A41   16 00                    LD   D,0
0A43   5F                       LD   E,A
0A44   19                       ADD  HL,DE
0A45   5E                       LD   E,(HL)
0A46   23                       INC  HL
0A47   56                       LD   D,(HL)
0A48   D5                       PUSH DE
0A49   E1                       POP  HL
0A4A   11 34 0A                 LD   DE,RUNS
0A4D   D5                       PUSH DE
0A4E   E9                       JP   (HL)
                        *
                        *
                        *
                        ************************
                        *                      *
                        *    NEXT SUBROUTINE   *
                        *                      *
                        ************************
                        *
                        * read command to A
                        * and next address
                        *
0A4F   2A 75 87         NEXT    LD   HL,(ADDR)
0A52   7E                       LD   A,(HL)
0A53   23                       INC  HL
0A54   22 75 87                 LD   (ADDR),HL
0A57   C9                       RET
                        *
```

```
                        *
                        *
                        ****************************
                        *                           *
                        *   DISBAK SUBROUTINE       *
                        *                           *
                        ****************************
                        *
0A58    21 00 80        DISBAK  LD    HL,DIS
0A5B    11 A0 80                LD    DE,BAK
0A5E    01 50 00                LD    BC,BYTE
0A61    ED B0                   LDIR
0A63    C9                      RET
                        *
                        *
                        *
                        ****************************
                        *                           *
                        *   BAKDIS SUBROUTINE       *
                        *                           *
                        ****************************
                        *
0A64    21 A0 80        BAKDIS  LD    HL,BAK
0A67    11 00 80                LD    DE,DIS
0A6A    01 50 00                LD    BC,BYTE
0A6D    ED B0                   LDIR
0A6F    C9                      RET
                        *
                        *
                        *
                        ****************************
                        *                           *
                        *   CLEAR SUBROUTINE        *
                        *                           *
                        ****************************
                        *
0A70    21 50 80        CL      LD    HL,BUF
0A73    06 50                   LD    B,BYTE
0A75    AF                      XOR   A
0A76    77              CL1     LD    (HL),A
0A77    23                      INC   HL
0A78    10 FC                   DJNZ  CL1
0A7A    C9                      RET
                        *
                        *
                        *
                        ****************************
                        *                           *
```

```
                        *   TEXT SCREEN SUB.    *
                        *                       *
                        *************************
                        *
                        * IN = ((ADDR))..((ADDR)+10)
                        *      10 byte of ascii char.
                        *      transfer to BUF
                        *
0A7B   11 50 80         TS       LD   DE,BUF
0A7E   06 0A                     LD   B,10
                        *
0A80   CD 4F 0A         TS1      CALL NEXT
0A83   CD 89 0A                  CALL TSS
0A86   10 F8                     DJNZ TS1
0A88   C9                        RET
                        *
0A89   C5               TSS      PUSH BC
0A8A   E5                        PUSH HL
0A8B   D5                        PUSH DE
0A8C   6F                        LD   L,A
0A8D   26 00                     LD   H,0
0A8F   29                        ADD  HL,HL          ;*8
0A90   29                        ADD  HL,HL          ;
0A91   29                        ADD  HL,HL          ;
0A92   E5                        PUSH HL
0A93   D1                        POP  DE
0A94   21 00 1C                  LD   HL,CHRGEN
0A97   19                        ADD  HL,DE
0A98   D1                        POP  DE
0A99   01 08 00                  LD   BC,8
0A9C   ED B0                     LDIR
0A9E   E1                        POP  HL
0A9F   C1                        POP  BC
0AA0   C9                        RET
                        *
                        *
                        *
                        *************************
                        *                       *
                        *    TEXT LINE SUB.     *
                        *                       *
                        *************************
                        *
                        * IN = ((ADDR)) = n
                        *      ((ADDR)+1)..((ADDR)+n+1)
                        *      n byte of ascii char.
                        *
                        *      transfer to TLBUF
```

```
                        *       and shift left display
                        *
0AA1    CD 4F 0A        TL      CALL NEXT
0AA4    FE 65                   CP   100+1 \       ;max char
0AA6    D0                      RET  NC
0AA7    FE 00                   CP   0
0AA9    C8                      RET  Z
                        *
0AAA    47                      LD   B,A
0AAB    32 7A 87                LD   (STOREX),A
0AAE    11 F0 80                LD   DE,TLBUF
0AB1    CD 4F 0A        TL1     CALL NEXT
0AB4    CD 89 0A                CALL TSS
0AB7    10 F8                   DJNZ TL1
                        *
0AB9    3A 7A 87                LD   A,(STOREX)
0ABC    6F                      LD   L,A
0ABD    26 00                   LD   H,0
0ABF    29                      ADD  HL,HL         ;*8
0AC0    29                      ADD  HL,HL         ;
0AC1    29                      ADD  HL,HL         ;
0AC2    11 50 00                LD   DE,BYTE
0AC5    19                      ADD  HL,DE
0AC6    E5                      PUSH HL
0AC7    C1                      POP  BC            ;lenght
0AC8    DD 21 F0 80             LD   IX,TLBUF
                        *
0ACC    16 00           TL2     LD   D,0
0ACE    21 50 00                LD   HL,BYTE
0AD1    ED 42                   SBC  HL,BC
0AD3    30 03                   JR   NC,TL3        ;bc<=byte
0AD5    DD 56 00                LD   D,(IX+0)      ;bc> byte
                        *
0AD8    21 4F 80        TL3     LD   HL,DIS+79
0ADB    C5                      PUSH BC
0ADC    06 50                   LD   B,BYTE
0ADE    7E              TL4     LD   A,(HL)
0ADF    72                      LD   (HL),D
0AE0    57                      LD   D,A
0AE1    2B                      DEC  HL
0AE2    10 FA                   DJNZ TL4
                        *
0AE4    3E 01                   LD   A,SPEED1
0AE6    CD 11 05                CALL DLS
0AE9    DD 23                   INC  IX
0AEB    C1                      POP  BC
0AEC    0B                      DEC  BC
0AED    78                      LD   A,B
```

```
0AEE   B1                  OR   C
0AEF   20 DB               JR   NZ,TL2
0AF1   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *  GRAPHIC SUBROUTINE     *
                    *                         *
                    ***************************
                    *
                    * IN = ((ADDR))..((ADDR)+80)
                    *       80 byte bit image
                    *       transfer to BUF
                    *
0AF2   11 50 80     GR     LD   DE,BUF
0AF5   06 50               LD   B,BYTE
0AF7   CD 4F 0A     GR1    CALL NEXT
0AFA   12                  LD   (DE),A
0AFB   13                  INC  DE
0AFC   10 F9               DJNZ GR1
0AFE   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *  DISPLAY SUBROUTINE     *
                    *                         *
                    ***************************
                    *
0AFF   21 50 80     DX     LD   HL,BUF
0B02   11 00 80            LD   DE,DIS
0B05   01 50 00            LD   BC,BYTE
0B08   ED B0               LDIR
0B0A   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *    LOAD SUBROUTINE      *
                    *                         *
                    ***************************
                    *
                    * IN = ((ADDR))
                    *       0-3 :up down left right
```

```
                        *
0B0B   CD 4F 0A     LX      CALL  NEXT
0B0E   FE 01                CP    1
0B10   28 31                JR    Z,LXD
0B12   FE 02                CP    2
0B14   28 56                JR    Z,LXL
0B16   FE 03                CP    3
0B18   28 66                JR    Z,LXR
                        *
0B1A   11 01 FE     LXU     LD    DE,0FE01H
0B1D   0E 08                LD    C,8
0B1F   21 00 80     LXU1    LD    HL,DIS
0B22   DD 21 50 80          LD    IX,BUF
0B26   06 50                LD    B,BYTE
0B28   7E           LXU2    LD    A,(HL)
0B29   A2                   AND   D
0B2A   77                   LD    (HL),A
0B2B   DD 7E 00             LD    A,(IX+0)
0B2E   A3                   AND   E
0B2F   B6                   OR    (HL)
0B30   77                   LD    (HL),A
0B31   23                   INC   HL
0B32   DD 23                INC   IX
0B34   10 F2                DJNZ  LXU2
0B36   3E 04                LD    A,SPEED2
0B38   CD 11 05             CALL  DLS
0B3B   CB 02                RLC   D
0B3D   CB 03                RLC   E
0B3F   0D                   DEC   C
0B40   20 DD                JR    NZ,LXU1
0B42   C9                   RET
                        *
0B43   11 80 7F     LXD     LD    DE,7F80H
0B46   0E 08                LD    C,8
0B48   21 00 80     LXD1    LD    HL,DIS
0B4B   DD 21 50 80          LD    IX,BUF
0B4F   06 50                LD    B,BYTE
0B51   7E           LXD2    LD    A,(HL)
0B52   A2                   AND   D
0B53   77                   LD    (HL),A
0B54   DD 7E 00             LD    A,(IX+0)
0B57   A3                   AND   E
0B58   B6                   OR    (HL)
0B59   77                   LD    (HL),A
0B5A   23                   INC   HL
0B5B   DD 23                INC   IX
0B5D   10 F2                DJNZ  LXD2
0B5F   3E 04                LD    A,SPEED2
```

```
0B61   CD 11 05              CALL DLS
0B64   CB 0A                 RRC  D
0B66   CB 0B                 RRC  E
0B68   0D                    DEC  C
0B69   20 DD                 JR   NZ,LXD1
0B6B   C9                    RET
                      *
0B6C   21 9F 80       LXL    LD   HL,BUF+79
0B6F   11 4F 80              LD   DE,DIS+79
0B72   06 50                 LD   B,BYTE
0B74   7E             LXL1   LD   A,(HL)
0B75   12                    LD   (DE),A
0B76   3E 01                 LD   A,SPEED1
0B78   CD 11 05              CALL DLS
0B7B   2B                    DEC  HL
0B7C   1B                    DEC  DE
0B7D   10 F5                 DJNZ LXL1
0B7F   C9                    RET
                      *
0B80   21 50 80       LXR    LD   HL,BUF
0B83   11 00 80              LD   DE,DIS
0B86   06 50                 LD   B,BYTE
0B88   7E             LXR1   LD   A,(HL)
0B89   12                    LD   (DE),A
0B8A   3E 01                 LD   A,SPEED1
0B8C   CD 11 05              CALL DLS
0B8F   23                    INC  HL
0B90   13                    INC  DE
0B91   10 F5                 DJNZ LXR1
0B93   C9                    RET
                      *
                      *
                      *
                      ************************
                      *                      *
                      *    DOOR SUBROUTINE   *
                      *                      *
                      ************************
                      *
                      * IN = ((ADDR))
                      *      0-3 :up down left right
                      *
0B94   CD 4F 0A       DR     CALL NEXT
0B97   FE 01                 CP   1
0B99   28 08                 JR   Z,DRUD
0B9B   FE 02                 CP   2
0B9D   28 3F                 JR   Z,DRLR
0B9F   FE 03                 CP   3
```

```
0BA1   28 3B              JR    Z,DRLR
                    *
0BA3   0E 04        DRUD  LD    C,4
0BA5   11 18 E7           LD    DE,0E718H
0BA8   21 00 80     DRUD1 LD    HL,DIS
0BAB   DD 21 50 80        LD    IX,BUF
0BAF   06 50              LD    B,BYTE
0BB1   7E           DRUD2 LD    A,(HL)
0BB2   A2                 AND   D
0BB3   77                 LD    (HL),A
0BB4   DD 7E 00           LD    A,(IX+0)
0BB7   A3                 AND   E
0BB8   B6                 OR    (HL)
0BB9   77                 LD    (HL),A
0BBA   23                 INC   HL
0BBB   DD 23              INC   IX
0BBD   10 F2              DJNZ  DRUD2
0BBF   3E 04              LD    A,SPEED2
0BC1   CD 11 05           CALL  DLS
0BC4   1E 00              LD    E,0
0BC6   7A                 LD    A,D
0BC7   E6 F0              AND   0F0H
0BC9   CB 27              SLA   A
0BCB   57                 LD    D,A
0BCC   07           DRUD3 RLCA
0BCD   30 04              JR    NC,DRUD4
0BCF   CB 13              RL    E
0BD1   18 F9              JR    DRUD3
0BD3   7A           DRUD4 LD    A,D
0BD4   B3                 OR    E
0BD5   57                 LD    D,A
0BD6   3E FF              LD    A,0FFH
0BD8   92                 SUB   D
0BD9   5F                 LD    E,A
0BDA   0D                 DEC   C
0BDB   20 CB              JR    NZ,DRUD1
0BDD   C9                 RET
                    *
0BDE   06 28        DRLR  LD    B,40
0BE0   21 27 80           LD    HL,DIS+39
0BE3   11 28 80           LD    DE,DIS+40
0BE6   DD 21 77 80        LD    IX,BUF+39
0BEA   FD 21 78 80        LD    IY,BUF+40
0BEE   DD 7E 00     DRLR1 LD    A,(IX+0)
0BF1   77                 LD    (HL),A
0BF2   FD 7E 00           LD    A,(IY+0)
0BF5   12                 LD    (DE),A
0BF6   3E 01              LD    A,SPEED1
```

```
0BF8   CD 11 05            CALL DLS
0BFB   2B                  DEC  HL
0BFC   13                  INC  DE
0BFD   DD 2B               DEC  IX
0BFF   FD 23               INC  IY
0C01   10 EB               DJNZ DRLR1
0C03   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *   SHIFT SUBROUTINE      *
                    *                         *
                    ***************************
                    *
                    * IN = ((ADDR))
                    *      0-3 :up down left right
                    *
0C04   CD 4F 0A     SH     CALL NEXT
0C07   FE 01               CP   1
0C09   28 2B               JR   Z,SHD
0C0B   FE 02               CP   2
0C0D   28 4A               JR   Z,SHL
0C0F   FE 03               CP   3
0C11   28 63               JR   Z,SHR
                    *
0C13   0E 08        SHU    LD   C,8          ;8 bit
0C15   21 00 80     SHU1   LD   HL,DIS
0C18   DD 21 50 80         LD   IX,BUF
0C1C   06 50               LD   B,BYTE
0C1E   AF           SHU2   XOR  A
0C1F   CB 26               SLA  (HL)
0C21   DD CB 00 06         RLC  (IX+0)
0C25   17                  RLA
0C26   B6                  OR   (HL)
0C27   77                  LD   (HL),A
0C28   23                  INC  HL
0C29   DD 23               INC  IX
0C2B   10 F1               DJNZ SHU2
0C2D   3E 04               LD   A,SPEED2
0C2F   CD 11 05            CALL DLS
0C32   0D                  DEC  C
0C33   20 E0               JR   NZ,SHU1
0C35   C9                  RET
                    *
0C36   0E 08        SHD    LD   C,8
0C38   21 00 80     SHD1   LD   HL,DIS
```

```
0C3B   DD 21 50 80            LD    IX,BUF
0C3F   06 50                  LD    B,BYTE
0C41   AF             SHD2    XOR   A
0C42   CB 3E                  SRL   (HL)
0C44   DD CB 00 0E            RRC   (IX+0)
0C48   1F                     RRA
0C49   B6                     OR    (HL)
0C4A   77                     LD    (HL),A
0C4B   23                     INC   HL
0C4C   DD 23                  INC   IX
0C4E   10 F1                  DJNZ  SHD2
0C50   3E 04                  LD    A,SPEED2
0C52   CD 11 05               CALL  DLS
0C55   0D                     DEC   C
0C56   20 E0                  JR    NZ,SHD1
0C58   C9                     RET
                       *
0C59   21 50 80        SHL    LD    HL,BUF
0C5C   0E 50                  LD    C,BYTE
0C5E   56              SHL1   LD    D,(HL)
0C5F   E5                     PUSH  HL
0C60   21 4F 80               LD    HL,DIS+79
0C63   06 50                  LD    B,BYTE
0C65   7E              SHL2   LD    A,(HL)
0C66   72                     LD    (HL),D
0C67   57                     LD    D,A
0C68   2B                     DEC   HL
0C69   10 FA                  DJNZ  SHL2
0C6B   3E 01                  LD    A,SPEED1
0C6D   CD 11 05               CALL  DLS
0C70   E1                     POP   HL
0C71   23                     INC   HL
0C72   0D                     DEC   C
0C73   20 E9                  JR    NZ,SHL1
0C75   C9                     RET
                       *
0C76   21 9F 80        SHR    LD    HL,BUF+79
0C79   0E 50                  LD    C,BYTE
0C7B   56              SHR1   LD    D,(HL)
0C7C   E5                     PUSH  HL
0C7D   21 00 80               LD    HL,DIS
0C80   06 50                  LD    B,BYTE
0C82   7E              SHR2   LD    A,(HL)
0C83   72                     LD    (HL),D
0C84   57                     LD    D,A
0C85   23                     INC   HL
0C86   10 FA                  DJNZ  SHR2
0C88   3E 01                  LD    A,SPEED1
```

```
0C8A   CD 11 05            CALL DLS
0C8D   E1                  POP  HL
0C8E   2B                  DEC  HL
0C8F   0D                  DEC  C
0C90   20 E9               JR   NZ,SHR1
0C92   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *  DELAY (1 SEC.) SUB.    *
                    *                         *
                    ***************************
                    *
0C93   3E 28        DL     LD   A,40
0C95   CD 11 05            CALL DLS
0C98   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *   FLASH SUBROUTINE      *
                    *                         *
                    ***************************
                    *
0C99   CD 58 0A     FS     CALL DISBAK
0C9C   21 00 80            LD   HL,DIS
0C9F   06 50               LD   B,BYTE
0CA1   AF                  XOR  A
0CA2   F3                  DI
0CA3   77           FS1    LD   (HL),A
0CA4   23                  INC  HL
0CA5   10 FC               DJNZ FS1
0CA7   FB                  EI
0CA8   3E 14               LD   A,20
0CAA   CD 11 05            CALL DLS
0CAD   CD 64 0A            CALL BAKDIS
0CB0   3E 14               LD   A,20
0CB2   CD 11 05            CALL DLS
0CB5   C9                  RET
                    *
                    *
                    *
                    ***************************
                    *                         *
                    *  INVERSE SUBROUTINE     *
```

```
                        *                           *
                        ****************************
                        *
0CB6   21 00 80         IV          LD    HL,DIS
0CB9   06 50                        LD    B,BYTE
0CBB   F3                           DI
0CBC   7E               IV1         LD    A,(HL)
0CBD   2F                           CPL
0CBE   77                           LD    (HL),A
0CBF   23                           INC   HL
0CC0   10 FA                        DJNZ  IV1
0CC2   FB                           EI
0CC3   C9                           RET
                        *
                        *
                        *
                        ****************************
                        *                           *
                        *   ROTATE SUBROUTINE       *
                        *                           *
                        ****************************
                        *
                        * IN = ((ADDR))
                        *      0-3 :up down left right
                        *
0CC4   CD 4F 0A         RT          CALL  NEXT
0CC7   FE 01                        CP    1
0CC9   28 1D                        JR    Z,RTD
0CCB   FE 02                        CP    2
0CCD   28 2E                        JR    Z,RTL
0CCF   FE 03                        CP    3
0CD1   28 4A                        JR    Z,RTR
                        *
0CD3   0E 08            RTU         LD    C,8
0CD5   21 00 80         RTU1        LD    HL,DIS
0CD8   06 50                        LD    B,BYTE
0CDA   CB 06            RTU2        RLC   (HL)
0CDC   23                           INC   HL
0CDD   10 FB                        DJNZ  RTU2
0CDF   3E 04                        LD    A,SPEED2
0CE1   CD 11 05                     CALL  DLS
0CE4   0D                           DEC   C
0CE5   20 EE                        JR    NZ,RTU1
0CE7   C9                           RET
                        *
0CE8   0E 08            RTD         LD    C,8
0CEA   21 00 80         RTD1        LD    HL,DIS
0CED   06 50                        LD    B,BYTE
```

```
0CEF   CB 0E       RTD2     RRC  (HL)
0CF1   23                   INC  HL
0CF2   10 FB                DJNZ RTD2
0CF4   3E 04                LD   A,SPEED2
0CF6   CD 11 05             CALL DLS
0CF9   0D                   DEC  C
0CFA   20 EE                JR   NZ,RTD1
0CFC   C9                   RET
                    *
0CFD   CD 58 0A    RTL      CALL DISBAK
0D00   21 A0 80             LD   HL,BAK
0D03   0E 50                LD   C,BYTE
0D05   56          RTL1     LD   D,(HL)
0D06   E5                   PUSH HL
0D07   21 4F 80             LD   HL,DIS+79
0D0A   06 50                LD   B,BYTE
0D0C   7E          RTL2     LD   A,(HL)
0D0D   72                   LD   (HL),D
0D0E   57                   LD   D,A
0D0F   2B                   DEC  HL
0D10   10 FA                DJNZ RTL2
0D12   3E 01                LD   A,SPEED1
0D14   CD 11 05             CALL DLS
0D17   E1                   POP  HL
0D18   23                   INC  HL
0D19   0D                   DEC  C
0D1A   20 E9                JR   NZ,RTL1
0D1C   C9                   RET
                    *
0D1D   CD 58 0A    RTR      CALL DISBAK
0D20   21 EF 80             LD   HL,BAK+79
0D23   0E 50                LD   C,BYTE
0D25   56          RTR1     LD   D,(HL)
0D26   E5                   PUSH HL
0D27   21 00 80             LD   HL,DIS
0D2A   06 50                LD   B,BYTE
0D2C   7E          RTR2     LD   A,(HL)
0D2D   72                   LD   (HL),D
0D2E   57                   LD   D,A
0D2F   23                   INC  HL
0D30   10 FA                DJNZ RTR2
0D32   3E 01                LD   A,SPEED1
0D34   CD 11 05             CALL DLS
0D37   E1                   POP  HL
0D38   2B                   DEC  HL
0D39   0D                   DEC  C
0D3A   20 E9                JR   NZ,RTR1
0D3C   C9                   RET
```

```
                        *
                        *
                        *
                        ************************
                        *                      *
                        *  ROM DATA AND TABLE  *
                        *                      *
                        ************************
                        *
0D3D  70 0A 7B 0A RUNTAB  DW   CL,TS
0D41  A1 0A F2 0A         DW   TL,GR
0D45  FF 0A 0B 0B         DW   DX,LX
0D49  94 0B 04 0C         DW   DR,SH
0D4D  93 0C 99 0C         DW   DL,FS
0D51  B6 0C C4 0C         DW   IV,RT
                        *
0D55  65 30 50 60 KEYTAB  DB   65H,30H,50H,60H     ;CL,TS,TL,GR
0D59  55 31 51 61         DB   55H,31H,51H,61H     ;DI,LD,DR,SH
0D5D  35 32 52 62         DB   35H,32H,52H,62H     ;DL,FS,IN,RT
0D61  66 33 53 63         DB   66H,33H,53H,63H     ;edit,clear,change,<-
0D65  56 34 54 64         DB   56H,34H,54H,64H     ;ins,del,^,v
0D69  36 57 67            DB   36H,57H,67H         ;linc,ldec,->
                        *
0D6C  30 31 32 33 NUMTAB  DB   '0123456789'
0D70  34 35 36 37
0D74  38 39
                        *
0D76  43 4C 20    ISCTAB  DB   'CL '
0D79  54 53 2D            DB   'TS-'
0D7C  54 4C 2D            DB   'TL-'
0D7F  47 52 2D            DB   'GR-'
0D82  44 49 20            DB   'DI '
0D85  4C 44 2C            DB   'LD,'
0D88  44 52 2C            DB   'DR,'
0D8B  53 48 2C            DB   'SH,'
0D8E  44 4C 20            DB   'DL '
0D91  46 53 20            DB   'FS '
0D94  49 4E 20            DB   'IN '
0D97  52 54 2C            DB   'RT,'
0D9A  52 45 20            DB   'RE '
0D9D  04 05 06 07         DB   04H,05H,06H,07H     ;^,v,<,>
                        *
0DA1  41 42 43 44 UPCTAB    DB   'ABCD'
0DA5  4D 4E 4F 50           DB   'MNOP'
0DA9  59 5A 30 31           DB   'YZ01'
0DAD  20 21 3A 3B           DB   ' !:;'
0DB1  45 46 47 48           DB   'EFGH'
0DB5  51 52 53 54           DB   'QRST'
```

```
0DB9   32 33 34 35          DB    '2345'
0DBD   00 01 02 03          DB    00H,01H,02H,03H
0DC1   49 4A 4B 4C          DB    'IJKL'
0DC5   55 56 57 58            DB    'UVWX'
0DC9   36 37 38 39          DB    '6789'
0DCD   04 05 06 07          DB    04H,05H,06H,07H
                       *
0DD1   61 62 63 64 LWCTAB   DB    'abcd'
0DD5   6D 6E 6F 70          DB    'mnop'
0DD9   79 7A 28 29          DB    'yz()'
0DDD   3D 5C 3F 40          DB    '=\?@'
0DE1   65 66 67 68          DB    'efgh'
0DE5   67 72 73 74          DB    'grst'
0DE9   2B 2D 2A 2F          DB    '+-*/'
0DED   5B 5D 7B 7D          DB    '[]{}'
0DF1   69 6A 6B 6C            DB    'ijkl'
0DF5   75 76 77 78          DB    'uvwx'
0DF9   2E 2C 3C 3E          DB    '.,<>'
0DFD   23 24 25 26          DB    '#$%&'
                       *
0E01   2A                   HELTAB    DB    42                    ;lenght
0E02   45 6C 65 63                    DB    'Electronic Banner by Pat 88. KMITL 30.3315'
0E06   74 72 6F 6E
0E0A   69 63 20 42
0E0E   61 6E 6E 65
0E12   72 20 62 79
0E16   20 50 61 74
0E1A   20 38 38 2E
0E1E   20 4B 4D 49
0E22   54 4C 20 33
0E26   30 2E 33 33
0E2A   31 35
                       *
0E2C   43 6F 6E 66 CLRTAB   DB    'Confirm'
0E30   69 72 6D
0E33   45 72 72 6F ERRTAB     DB    'Error'
0E37   72
                       *
                       *
                       *
0E38                   ENDPRO
                       ****************************
                       *                          *
                       *         RAM1 8K          *
                       *                          *
                       ****************************
                       *
       00 40           RAM1      EQU  4000H
```

```
00 40       PROA    *EQU  RAM1
9C 4F       PROB     EQU  PROA+3996
38 5F       LBYTEA   EQU  PROB+3996
9C 5F       LBYTEB   EQU  LBYTEA+100
            *
            *
            *
            ************************
            *                      *
            *       RAM2 2K        *
            *                      *
            ************************
            *
00 80       RAM2     EQU  8000H
00 80       DIS      EQU  RAM2
50 80       BUF      EQU  DIS+BYTE
A0 80       BAK      EQU  BUF+BYTE
F0 80       TLBUF    EQU  BAK+BYTE
10 84       DIS1     EQU  TLBUF+(100*8);100 char max
            *
6A 87       STACK    EQU  DIS1+BYTE+778
6A 87       ROW      EQU  STACK
6B 87       POINT    EQU  ROW+1
6C 87       SHOUT    EQU  POINT+1
6D 87       SHIFT    EQU  SHOUT+1
6F 87       LENGHT   EQU  SHIFT+2
71 87       COUNT    EQU  LENGHT+2
            *
72 87       DIRBUF   EQU  COUNT+1
73 87       OPCODE   EQU  DIRBUF+1
74 87       KEYIN    EQU  OPCODE+1
75 87       ADDR     EQU  KEYIN+1
77 87       LINE     EQU  ADDR+2
78 87       INDEX    EQU  LINE+1
7A 87       STOREX   EQU  INDEX+2
7C 87       STOREY   EQU  STOREX+2
7D 87       OLDNUM   EQU  STOREY+1
7E 87       NEWNUM   EQU  OLDNUM+1
7F 87       START    EQU  NEWNUM+1
81 87       FINAL    EQU  START+2
            *
83 87       SPACE    EQU  FINAL+2
8C 87       EDBUF    EQU  SPACE+9
F0 87       CURNUM   EQU  EDBUF+100    ;0-9
F1 87       CURCNT   EQU  CURNUM+1
F2 87       CURFAG   EQU  CURCNT+1     ;0-1
F3 87       CHRCNT   EQU  CURFAG+1     ;0-99 in hex
F4 87       AUTORE   EQU  CHRCNT+1     ;auto-repeat
```

```
F5 87       DOTFAG   EQU  AUTORE+1      ;0-1
F6 87       DOTX     EQU  DOTFAG+1     ;0-79 in hex
F7 87       DOTY     EQU  DOTX+1       ;0-7
F8 87       CASTAB   EQU  DOTY+1       ;upc/lwc
FA 87       CHRTAB   EQU  CASTAB+2
FC 87       CHRKEY   EQU  CHRTAB+2     ;0-3
FD 87       OLDKEY   EQU  CHRKEY+1
            *
FE 87       PWCODE   EQU  OLDKEY+1
FF 87       SYSFAG   EQU  PWCODE+1
            * b0 = key press flag
            *
0E38        ENDRAM
            *
0E38              END
```

********** S Y M B O L I C R E F E R E N C E T A B L E **********

ADDR	= 8775	AUTORE	= 87F4	BAK	= 80A0	BAKDIS	0A64
BUF	= 8050	BYTE	= 0050	CASTAB	= 87F8	CHRCNT	= 87F3
CHRGEN	= 1C00	CHRKEY	= 87FC	CHRTAB	= 87FA	CKDIP	0567
CKDIP1	056B	CKDIP2	0571	CKDIP3	0580	CKDIP4	0582
CKDIP5	0588	CKDIP6	0597	CKDIP7	0599	CKDIP8	05AF
CKDIP9	05B2	CKDIPX	05A6	CL	0A70	CL1	0A76
CLLBY1	0A2F	CLLBYT	0A2B	CLPRO	0A1E	CLPRO1	0A23
CLRTAB	0E2C	CODE	= 00A3	COUNT	= 8771	CURCNT	= 87F1
CURDOT	066A	CURDOT1	0670	CURDOT2	0677	CURDOT3	0690
CURFAG	= 87F2	CURNUM	= 87F0	CURSOR	0627	CURSOR1	0651
CURSOR2	065D	DELET1	09E3	DELETE	09C2	DIRBUF	= 8772
DIS	= 8000	DIS1	= 8410	DISBAK	0A58	DISPR1	0931
DISPR2	0952	DISPR3	0956	DISPR4	095B	DISPR5	0960
DISPR6	0965	DISPR7	0968	DISPRO	0912	DL	0C93
DLS	0511	DLS1	0514	DOTFAG	= 87F5	DOTX	= 87F6
DOTY	= 87F7	DR	0B94	DRLR	0BDE	DRLR1	0BEE
DRUD	0BA3	DRUD1	0BA8	DRUD2	0BB1	DRUD3	0BCC
DRUD4	0BD3	DX	0AFF	EDBUF	= 878C	ENDPRO	0E38
ENDRAM	0E38	ERRTAB	0E33	FINAL	= 8781	FLBYT1	0A11
FLBYT2	0A14	FLBYT3	0A16	FLBYTE	0A05	FNEW	0339
FNEW1	0343	FNEW2	034B	FNPRO	09E8	FNPRO1	09F4
FNPRO2	09F7	FNPRO3	09FA	FS	0C99	FS1	0CA3
GR	0AF2	GR1	0AF7	GRE1	085A	GRE2	0865
GRE3	0885	GRE4	0894	GRE41	089A	GRE42	08A1
GRE5	08B7	GRE6	08C4	GRE7	08D1	GRE8	08DF
GREDIT	084E	HELTAB	0E01	INDEX	= 8778	INIT	0017
INIT1	001F	INIT2	002E	INIT3	00A3	INIT4	00AB
INSER1	09BD	INSERT	09A1	ISC	08EE	ISCTAB	0D76
IV	0CB6	IV1	0CBC	KEYIN	= 8774	KEYTAB	0D55
LBYTEA	= 5F38	LBYTEB	= 5F9C	LENGHT	= 876F	LINE	= 8777
LWCTAB	0DD1	LX	0B0B	LXD	0B43	LXD1	0B48
LXD2	0B51	LXL	0B6C	LXL1	0B74	LXR	0B80
LXR1	0B88	LXU	0B1A	LXU1	0B1F	LXU2	0B28
NEWNUM	= 877E	NEXT	0A4F	NUM	08FD	NUMTAB	0D6C
OLDKEY	= 87FD	OLDNUM	= 877D	OPCODE	= 8773	OPDEL	025F
OPINS	0242	PCHAR	= 0001	PCONT	= 0003	PDATA	= 0000
PGM	00D9	PGM1	00EA	PGM10	02E1	PGM11	03F3
PGM11A	040F	PGM12	043F	PGM12A	0456	PGM12B	0463
PGM13	04A0	PGM13A	04B6	PGM13B	04C1	PGM13C	04CE
PGM13D	04E0	PGM13E	04EC	PGM2	00F2	PGM2A	010F
PGM3	011D	PGM4	0127	PGM4A	012D	PGM5	014F
PGM5A	0157	PGM5B	015E	PGM5C	016A	PGM5D	017B
PGM5E	0181	PGM5F	018C	PGM5G	01A4	PGM5H	01AE
PGM5I	01B8	PGM5J	01C2	PGM5K	01CA	PGM5L	01DE
PGM6	01E4	PGM6A	01F2	PGM6B	01F4	PGM7	021F

PGM8	028B	PGM9	02B9	PGMGR	03CF	PGMTL	0318
PGMTL1	031F	PGMTS	02F4	PKEYB =	0002	POINT =	876B
PROA =	4000	PROB =	4F9C	PROBY1	098F	PROBY2	0993
PROBY3	0997	PROBY4	099B	PROBY5	099D	PROBYT	096F
PWCODE =	87FE	RAM1 =	4000	RAM2 =	8000	RESET0	0000
RESET1	0001	RESET2	00B2	RESET3	00CF	ROW =	876A
RT	0CC4	RTD	0CE8	RTD1	0CEA	RTD2	0CEF
RTL	0CFD	RTL1	0D05	RTL2	0D0C	RTR	0D1D
RTR1	0D25	RTR2	0D2C	RTU	0CD3	RTU1	0CD5
RTU2	0CDA	RUN	04F4	RUN1	04FA	RUN2	0506
RUN3	0509	RUNS	0A34	RUNTAB	0D3D	SCAND	0038
SCAND1	0052	SCAND2	005D	SCAND3	005F	SCAND4	0096
SCANK	051C	SCANK1	0520	SCANK2	0525	SCANK3	0545
SCANK4	054A	SCANK5	0557	SCANK6	055D	SCANKE	05B4
SCANKE1	05B8	SCANKE2	05C6	SCANKE3	05DE	SCANKE4	05E6
SCANKE5	05F3	SCANKE6	05F9	SCANKE7	0603	SCANKE8	0618
SH	0C04	SHD	0C36	SHD1	0C38	SHD2	0C41
SHIFT =	876D	SHL	0C59	SHL1	0C5E	SHL2	0C65
SHOUT =	876C	SHR	0C76	SHR1	0C7B	SHR2	0C82
SHU	0C13	SHU1	0C15	SHU2	0C1E	SPACE =	8783
SPEED1 =	0001	SPEED2 =	0004	STACK =	876A	START =	877F
STOREX =	877A	STOREY =	877C	SYSFAG =	87FF	TL	0AA1
TL1	0AB1	TL2	0ACC	TL3	0AD8	TL4	0ADE
TLBUF =	80F0	TLDEL	0389	TLE1	0771	TLE2	0790
TLE21	079F	TLE3	07A6	TLE4	07C2	TLE41	07CC
TLE5	07D2	TLE51	07E2	TLE52	07E5	TLE6	07EA
TLE7	07FA	TLE8	080A	TLE81	0837	TLE82	0838
TLEDIT	0766	TLEQA	03B3	TLINS	0368	TLMOV	03B7
TS	0A7B	TS1	0A80	TSE1	06AF	TSE2	06B7
TSE3	06BE	TSE4	06DA	TSE41	06E4	TSE5	06EA
TSE51	06FA	TSE52	06FD	TSE6	0702	TSE7	0712
TSE8	0722	TSE81	074F	TSE82	0750	TSEDIT	069A
TSS	0A89	UPCTAB	0DA1	WATDOG =	0040		

0000 ASSEMBLY ERRORS

DATA OF CHARACTER GEN.

```
1C00  70 F8 FC 7E FC F8 70 00--10 38 7C FE 7C 38 10 00   p..~..p..8|.|8..
1C10  38 38 D2 FE D2 38 38 00--00 38 7A FE 7A 38 00 00   88...88..8z.z8..
1C20  10 30 7E FE 7E 30 10 00--10 18 FC FE FC 18 10 00   .0~.~0..........
1C30  10 38 7C FE 38 38 38 00--38 38 38 FE 7C 38 10 00   .8|.888.888.|8..
1C40  FE FC F8 F0 E0 C0 80 00--80 C0 E0 F0 F8 FC FE 00   ................
1C50  FE 7E 3E 1E 0E 06 02 00--02 06 0E 1E 3E 7E FE 00   .~>.........>~..
1C60  AA 55 AA 55 AA 55 AA 55--CC CC 33 33 CC CC 33 33   .U.U.U.U..33..33
1C70  F0 F0 F0 F0 0F 0F 0F 0F--FF FF FF FF FF FF FF FF   ................
1C80  00 00 00 FF FF 00 00 00--18 18 18 18 18 18 18 18   ................
1C90  00 00 00 1F 1F 18 18 18--18 18 18 1F 1F 00 00 00   ................
1CA0  00 00 00 1F 1F 03 03 03--03 03 03 1F 1F 00 00 00   ................
1CB0  18 18 18 1F 1F 18 18 18--18 18 18 F8 F8 18 18 18   ................
1CC0  18 18 18 FF FF 00 00 00--00 00 00 FF FF 18 18 18   ................
1CD0  18 18 18 FF FF 18 18 18--FF 00 FF 00 FF 00 FF 00   ................
1CE0  AA AA AA AA AA AA AA AA--44 88 11 22 44 88 11 22   ........D.."D.."
1CF0  22 11 88 44 22 11 88 44--FF 88 88 88 FF 88 88 88   "..D"..D........
1D00  00 00 00 00 00 00 00 00--00 00 00 FA 00 00 00 00   ................
1D10  00 00 E0 00 E0 00 00 00--28 28 FE 28 FE 28 28 00   ........((.(.((.
1D20  24 54 FE 54 FE 54 48 00--02 C4 C8 10 26 46 80 00   $T.T.TH.....&F..
1D30  6C 92 92 6A 04 0A 10 00--00 00 20 40 80 00 00 00   l..j...... @....
1D40  00 00 38 44 82 00 00 00--00 00 82 44 38 00 00 00   ..8D.......D8...
1D50  10 54 38 FE 38 54 10 00--00 10 10 7C 10 10 00 00   .T8.8T.....|....
1D60  00 00 00 0A 0C 00 00 00--00 10 10 10 10 00 00 00   ................
1D70  00 00 00 06 06 00 00 00--00 04 08 10 20 40 00 00   ............ @..
1D80  7C 86 8A 92 A2 C2 7C 00--00 00 42 FE 02 00 00 00   |.....|...B.....
1D90  4E 92 92 92 92 92 62 00--44 92 92 92 92 92 6C 00   N.....b.D.....l.
1DA0  04 0C 14 24 44 FE 04 00--F4 92 92 92 92 92 8C 00   ...$D...........
1DB0  7C 92 92 92 92 92 4C 00--80 82 84 88 90 A0 C0 00   |.....L.........
1DC0  6C 92 92 92 92 92 6C 00--64 92 92 92 92 92 7C 00   l.....l.d.....|.
1DD0  00 00 00 6C 00 00 00 00--00 00 02 6C 00 00 00 00   ...l.......l....
1DE0  00 10 28 44 82 00 00 00--00 28 28 28 28 28 00 00   ..(D.....(((((..
1DF0  00 00 82 44 28 10 00 00--00 40 80 9A A0 40 00 00   ...D(....@...@..
1E00  7C 82 9A BA BA 8A 7A 00--3E 50 90 90 90 50 3E 00   |.....z.>P...P>.
1E10  FE 92 92 92 92 92 6C 00--38 44 82 82 82 82 82 00   ......l.8D......
1E20  FE 82 82 82 82 44 38 00--FE 92 92 92 92 92 82 00   .....D8.........
1E30  FE 90 90 90 90 90 80 00--7C 82 82 92 92 92 5E 00   ........|.....^.
1E40  FE 10 10 10 10 10 FE 00--00 00 82 FE 82 00 00 00   ................
1E50  0C 02 02 82 82 FC 80 00--FE 10 10 10 28 44 82 00   ............(D..
1E60  FE 02 02 02 02 02 06 00--FE 80 40 3E 40 80 FE 00   ..........@>@...
1E70  FE 40 20 10 08 04 FE 00--38 44 82 82 82 44 38 00   .@ .....8D...D8.
1E80  FE 90 90 90 90 90 60 00--38 44 82 82 8A 44 3A 00   ......'.8D...D:.
1E90  FE 90 90 90 98 94 62 00--64 92 92 92 92 92 4C 00   ......b.d.....L.
1EA0  C0 80 80 FE 80 80 C0 00--F8 04 02 02 02 04 F8 00   ................
1EB0  F0 08 04 02 04 08 F0 00--FC 02 04 F8 04 02 FC 00   ................
1EC0  82 44 28 10 28 44 82 00--C0 20 10 1E 10 20 C0 00   .D(.(D... ... ..
1ED0  C2 86 8A 92 A2 C2 86 00--00 FE FE 82 82 82 00 00   ................
1EE0  00 40 20 10 08 04 00 00--00 82 82 82 FE FE 00 00   .@ .............
1EF0  00 08 10 20 10 08 00 00--02 02 02 02 02 02 02 02   ... ............
1F00  00 00 80 40 20 00 00 00--1C 22 22 22 22 14 3E 00   ...@ .... """".>.
1F10  FE 22 22 22 22 22 1C 00--1C 22 22 22 22 22 22 00   .""""".... """""".
1F20  1C 22 22 22 22 14 FE 00--1C 2A 2A 2A 2A 2A 18 00   .""""....*****..
1F30  00 10 10 7E 90 90 00 00--18 25 25 25 25 19 3E 00   ...~.....%%%%.>.
1F40  FE 10 10 10 10 10 0E 00--00 00 12 5E 02 00 00 00   ...........^....
1F50  00 01 01 01 01 5E 00 00--00 FE 08 08 14 22 00 00   .....^.......".."
1F60  00 00 82 FE 02 00 00 00--3E 20 20 1E 20 20 1E 00   ........> .. .. ..
1F70  3E 20 20 20 20 20 1E 00--1C 22 22 22 22 22 1C 00   > ... """"".....
1F80  3F 24 24 24 24 24 18 00--18 24 24 24 24 24 18 3F   ?$$$$$...$$$$.?.
1F90  00 3E 08 10 20 20 00 00--12 2A 2A 2A 2A 2A 24 00   .>.. ....*****$.
1FA0  00 20 20 FC 22 22 00 00--3C 02 02 02 02 02 3E 00   . .."".."<.....>.
1FB0  30 08 04 02 04 08 30 00--38 04 02 0C 02 04 38 00   0.....0.8.....8.
1FC0  22 22 14 08 14 22 22 00--38 05 05 05 05 05 3E 00   ""...""".8.....>.
1FD0  22 22 26 2A 32 22 22 00--00 00 10 6C 82 82 00 00   ""&*2""....l....
1FE0  00 00 00 EE 00 00 00 00--00 82 82 6C 10 00 00 00   ...........l....
1FF0  20 40 80 40 20 40 80 00--08 04 02 FE 80 80 80 00    @.@ @..........
```

ภาคผนวก จ.

DATA SHEET

Z80™-CPU
Z80A™-CPU

Zilog

Product Specification

The Zilog Z80 product line is a complete set of microcomputer components, development systems and support software. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

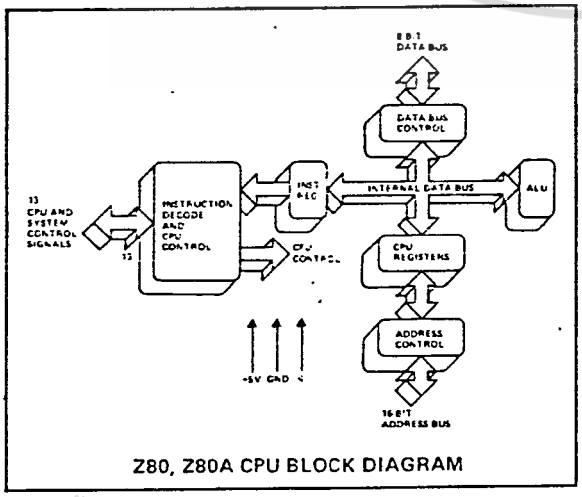
The Z80 and Z80A CPU's are third generation single chip microprocessors with unrivaled computational power. This increased computational power results in higher system through-put and more efficient memory utilization when compared to second generation microprocessors. In addition, the Z80 and Z80A CPU's are very easy to implement into a system because of their single voltage requirement plus all output signals are fully decoded and timed to control standard memory or peripheral circuits. The circuit is implemented using an N-channel, ion implanted, silicon gate MOS process.

Figure 1 is a block diagram of the CPU. Figure 2 details the internal register configuration which contains 208 bits of Read/Write memory that are accessible to the programmer. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or as 16-bit register pairs. There are also two sets of accumulator and flag registers. The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast Interrupt response. Each CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

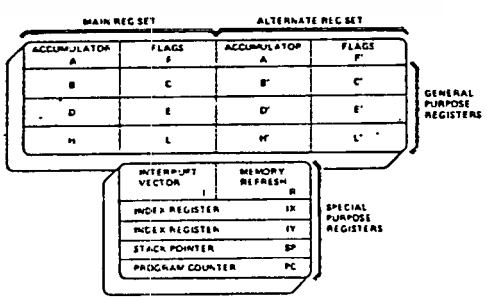
The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The Refresh register provides for automatic, totally transparent refresh of external dynamic memories. The I register is used in a powerful interrupt response mode to form the upper 8 bits of a pointer to a interrupt service address table, while the interrupting device supplies the lower 8 bits of the pointer. An indirect call is then made to this service address.

FEATURES

- Single chip, N-channel Silicon Gate CPU.
- 158 instructions–includes all 78 of the 8080A instructions with total software compatibility. New instructions include 4-, 8- and 16-bit operations with more useful addressing modes such as indexed, bit and relative.
- 17 internal registers.
- Three modes of fast interrupt response plus a non-maskable interrupt.
- Directly interfaces standard speed static or dynamic memories with virtually no external logic.
- 1.0 μs instruction execution speed.
- Single 5 VDC supply and single-phase 5 volt Clock.
- Out-performs any other single chip microcomputer in 4-, 8-, or 16-bit applications.
- All pins TTL Compatible
- Built-in dynamic RAM refresh circuitry.

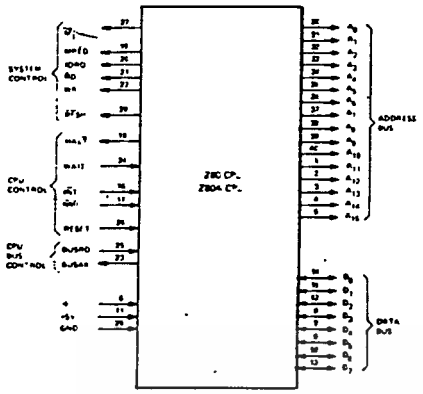


Z80, Z80A CPU BLOCK DIAGRAM



Z80, Z80A CPU REGISTERS

Z80, Z80A-CPU Pin Description



Z80, Z80A CPU PIN CONFIGURATION

A_0-A_{15} (Address Bus) Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges.

D_0-D_7 (Data Bus) Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one) Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution.

$\overline{MREQ}$ (Memory Request) Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$ (Input/ Output Request) Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus.

$\overline{RD}$ (Memory Read) Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$ (Memory Write) Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$ (Refresh) Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{HALT}$ (Halt state) Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{WAIT}$ (Wait) Input, active low. $\overline{WAIT}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active.

$\overline{INT}$ (Interrupt Request) Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled.

$\overline{NMI}$ (Non Maskable Interrupt) Input, active low. The non-maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H.

$\overline{RESET}$ Input, active low. $\overline{RESET}$ initializes the CPU as follows: reset interrupt enable flip-flop, clear PC and registers I and R and set interrupt to 8080A mode. During reset time, the address and data bus go to a high impedance state and all control output signals go to the inactive state.

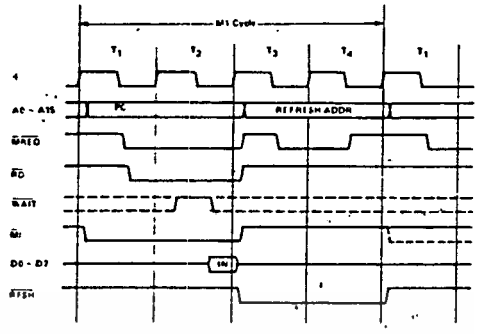
$\overline{BUSRQ}$ (Bus Request) Input, active low. The bus request signal has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle and is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these busses.

$\overline{BUSAK}$ (Bus Acknowledge) Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Timing Waveforms

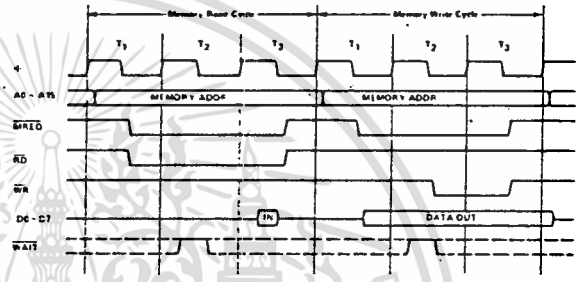
INSTRUCTION OP CODE FETCH

The program counter content (PC) is placed on the address bus immediately at the start of the cycle. One half clock time later $\overline{MREQ}$ goes active. The falling edge of $\overline{MREQ}$ can be used directly as a chip enable to dynamic memories. $\overline{RD}$ when active indicates that the memory data should be enabled onto the CPU data bus. The CPU samples data with the rising edge of the clock state T_3. Clock states T_3 and T_4 of a fetch cycle are used to refresh dynamic memories while the CPU is internally decoding and executing the instruction. The refresh control signal $\overline{RFSH}$ indicates that a refresh read of all dynamic memories should be accomplished.



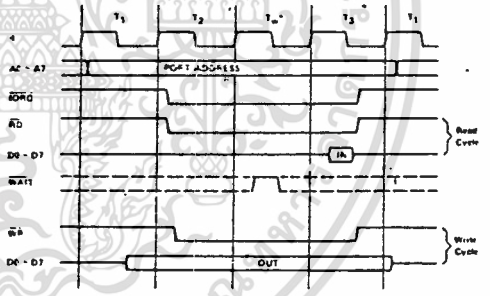
MEMORY READ OR WRITE CYCLES

Illustrated here is the timing of memory read or write cycles other than an OP code fetch (M_1 cycle). The $\overline{MREQ}$ and $\overline{RD}$ signals are used exactly as in the fetch cycle. In the case of a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory.



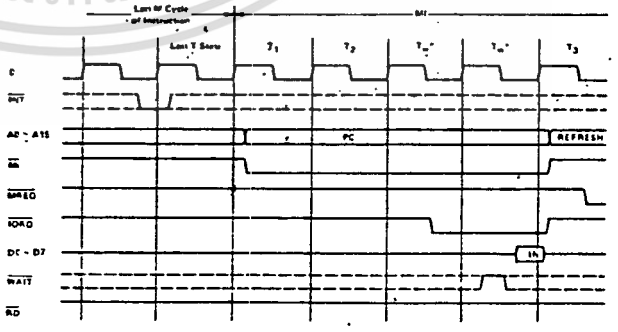
INPUT OR OUTPUT CYCLES

Illustrated here is the timing for an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted (Tw*). The reason for this is that during I/O operations this extra state allows sufficient time for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

The interrupt signal is sampled by the CPU with the rising edge of the last clock at the end of any instruction. When an interrupt is accepted, a special M_1 cycle is generated. During this M_1 cycle, the $\overline{IORQ}$ signal becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states (Tw*) are automatically added to this cycle so that a ripple priority interrupt scheme, such as the one used in the Z80 peripheral controllers, can be easily implemented.



Z80, Z80A Instruction Set

The following is a summary of the Z80, Z80A instruction set showing the assembly language mnemonic and the symbolic operation performed by the instruction. A more detailed listing appears in the Z80-CPU technical manual, and assembly language programming manual. The instructions are divided into the following categories:

8-bit loads	Miscellaneous Group
16-bit loads	Rotates and Shifts
Exchanges	Bit Set, Reset and Test
Memory Block Moves	Input and Output
Memory Block Searches	Jumps
8-bit arithmetic and logic	Calls
16-bit arithmetic	Restarts
General purpose Accumulator & Flag Operations	Returns

In the table the following terminology is used.

b ≡ a bit number in any 8-bit register or memory location

cc ≡ flag condition code
- NZ ≡ non zero
- Z ≡ zero
- NC ≡ non carry
- C ≡ carry
- PO ≡ Parity odd or no over flow
- PE ≡ Parity even or over flow
- P ≡ Positive
- M ≡ Negative (minus)

d ≡ any 8-bit destination register or memory location

dd ≡ any 16-bit destination register or memory location

e ≡ 8-bit signed 2's complement displacement used in relative jumps and indexed addressing

L ≡ 8 special call locations in page zero. In decimal notation these are 0, 8, 16, 24, 32, 40, 48 and 56

n ≡ any 8-bit binary number

nn ≡ any 16-bit binary number

r ≡ any 8-bit general purpose register (A, B, C, D, E, H, or L)

s ≡ any 8-bit source register or memory location

s_b ≡ a bit in a specific 8-bit register or memory location

ss ≡ any 16-bit source register or memory location

subscript "L" ≡ the low order 8 bits of a 16-bit register

subscript "H" ≡ the high order 8 bits of a 16-bit register

( ) ≡ the contents within the ( ) are to be used as a pointer to a memory location or I/O port number

8-bit registers are A, B, C, D, E, H, L, I and R

16-bit register pairs are AF, BC, DE and HL

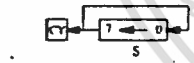
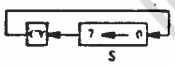
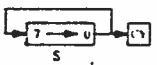
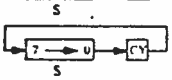
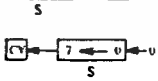
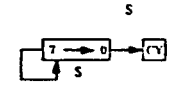
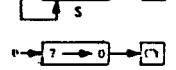
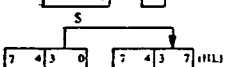
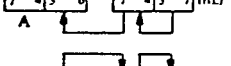
16-bit registers are SP, PC, IX and IY

Addressing Modes implemented include combinations of the following:

Immediate	Indexed
Immediate extended	Register
Modified Page Zero	Implied
Relative	Register Indirect
Extended	Bit

	Mnemonic	Symbolic Operation	Comments
8-BIT LOADS	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16-BIT LOADS	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H$; $(SP-2) \leftarrow ss_L$	ss = BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP)$; $dd_H \leftarrow (SP+1)$	dd = BC, DE, HL, AF, IX, IY
EXCHANGES	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	
	EX (SP), ss	$(SP) \leftrightarrow ss_L$, $(SP+1) \leftrightarrow ss_H$	ss ≡ HL, IX, IY

	Mnemonic	Symbolic Operation	Comments
MEMORY BLOCK MOVES	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
MEMORY BLOCK SEARCHES	CPI	A-(HL), HL ← HL+1 BC ← BC-1	A-(HL) sets the flags only. A is not affected
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, Repeat until BC= 0 or A = (HL)	
8-BIT ALU	ADD s	A ← A + s	CY is the carry flag
	ADC s	A ← A + s + CY	
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ∈ s	

	Mnemonic	Symbolic Operation	Comments
8-BIT ALU	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d - 1	
16-BIT ARITHMETIC	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
GP ACC. & FLAG	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	A ← 00 - A	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
MISCELLANEOUS	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
ROTATES AND SHIFTS	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemonic	Symbolic Operation	Comments
BIT S, R, & T	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag
	SET b, s	$s_b \leftarrow 1$	s ≡ r, (HL) (IX+e), (IY+e)
	RES b, s	$s_b \leftarrow 0$	
INPUT AND OUTPUT	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
JUMPS	JP nn	PC ← nn	cc { NZ PO; Z PE; NC P; C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	kk { NZ NC; Z C
	JR kk, e	If condition kk is true PC ← PC + e, else continue	
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
CALLS	CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, PC ← nn	cc { NZ PO; Z PE; NC P; C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
RESTARTS	RST L	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$ $PC_L \leftarrow L$	
RETURNS	RET	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$	cc { NZ PO; Z PE; NC P; C M
	RET cc	If condition cc is false continue, else same as RET	
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	4	1121	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	180	[1]	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{f(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	[3]		nsec	
	t_{caf}	Address Stable From RD or WR During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF
	$t_{f(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{su(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{su(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From WR	[7]		nsec	
	t_H	Any Hold Time for Setup Time	0		nsec	
MRQ	$t_{DL\downarrow(MR)}$	MRQ Delay From Falling Edge of Clock, MRQ Low		100	nsec	C_L = 50pF
	$t_{DH\uparrow(MR)}$	MRQ Delay From Rising Edge of Clock, MRQ High		100	nsec	
	$t_{DH\downarrow(MR)}$	MRQ Delay From Falling Edge of Clock, MRQ High		100	nsec	
	$t_{w(MRL)}$	Pulse Width, MRQ Low	[8]		nsec	
	$t_{w(MRH)}$	Pulse Width, MRQ High	[9]		nsec	
IORQ	$t_{DL\uparrow(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ Low		90	nsec	C_L = 50pF
	$t_{DL\downarrow(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ Low		110	nsec	
	$t_{DH\uparrow(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ High		100	nsec	
	$t_{DH\downarrow(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ High		110	nsec	
RD	$t_{DL\uparrow(RD)}$	RD Delay From Rising Edge of Clock, RD Low		100	nsec	C_L = 50pF
	$t_{DL\downarrow(RD)}$	RD Delay From Falling Edge of Clock, RD Low		130	nsec	
	$t_{DH\uparrow(RD)}$	RD Delay From Rising Edge of Clock, RD High		100	nsec	
	$t_{DH\downarrow(RD)}$	RD Delay From Falling Edge of Clock, RD High		110	nsec	
WR	$t_{DL\uparrow(WR)}$	WR Delay From Rising Edge of Clock, WR Low		80	nsec	C_L = 50pF
	$t_{DL\downarrow(WR)}$	WR Delay From Falling Edge of Clock, WR Low		90	nsec	
	$t_{DH\uparrow(WR)}$	WR Delay From Falling Edge of Clock, WR High		100	nsec	
	$t_{w(WRL)}$	Pulse Width, WR Low	[10]		nsec	
M1	$t_{DL(M1)}$	M1 Delay From Rising Edge of Clock, M1 Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	M1 Delay From Rising Edge of Clock, M1 High		130	nsec	
RFSH	$t_{DL(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH Low		180	nsec	C_L = 50pF
	$t_{DH(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH High		150	nsec	
WAIT	$t_{s(WT)}$	WAIT Setup Time to Falling Edge of Clock	70		nsec	
HALT	$t_{D(HT)}$	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
INT	$t_{s(IT)}$	INT Setup Time to Rising Edge of Clock	80		nsec	
NMI	$t_{w(NML)}$	Pulse Width, NMI Low	80		nsec	
BUSRQ	$t_{s(BQ)}$	BUSRQ Setup Time to Rising Edge of Clock	80		nsec	
BUSAK	$t_{DL(BA)}$	BUSAK Delay From Rising Edge of Clock, BUSAK Low		120	nsec	C_L = 50pF
	$t_{DH(BA)}$	BUSAK Delay From Falling Edge of Clock, BUSAK High		110	nsec	
RESET	$t_{s(RS)}$	RESET Setup Time to Rising Edge of Clock	90		nsec	
	$t_{f(C)}$	Delay to Float (MREQ, IORQ, RD and WR)		100	nsec	
	t_{mr}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r + 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_f - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_f - 80$

[8] $t_{w(MRL)} = t_c - 40$

[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WR}L)} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

NOTES

A Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

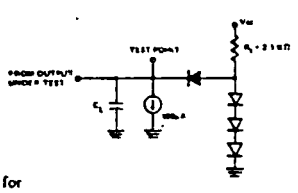
C The RESET signal must be active for a minimum of 3 clock cycles.

D Output Delay vs. Loaded Capacitance

TA = 70°C Vcc = +5V ±5%

Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines

E Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum

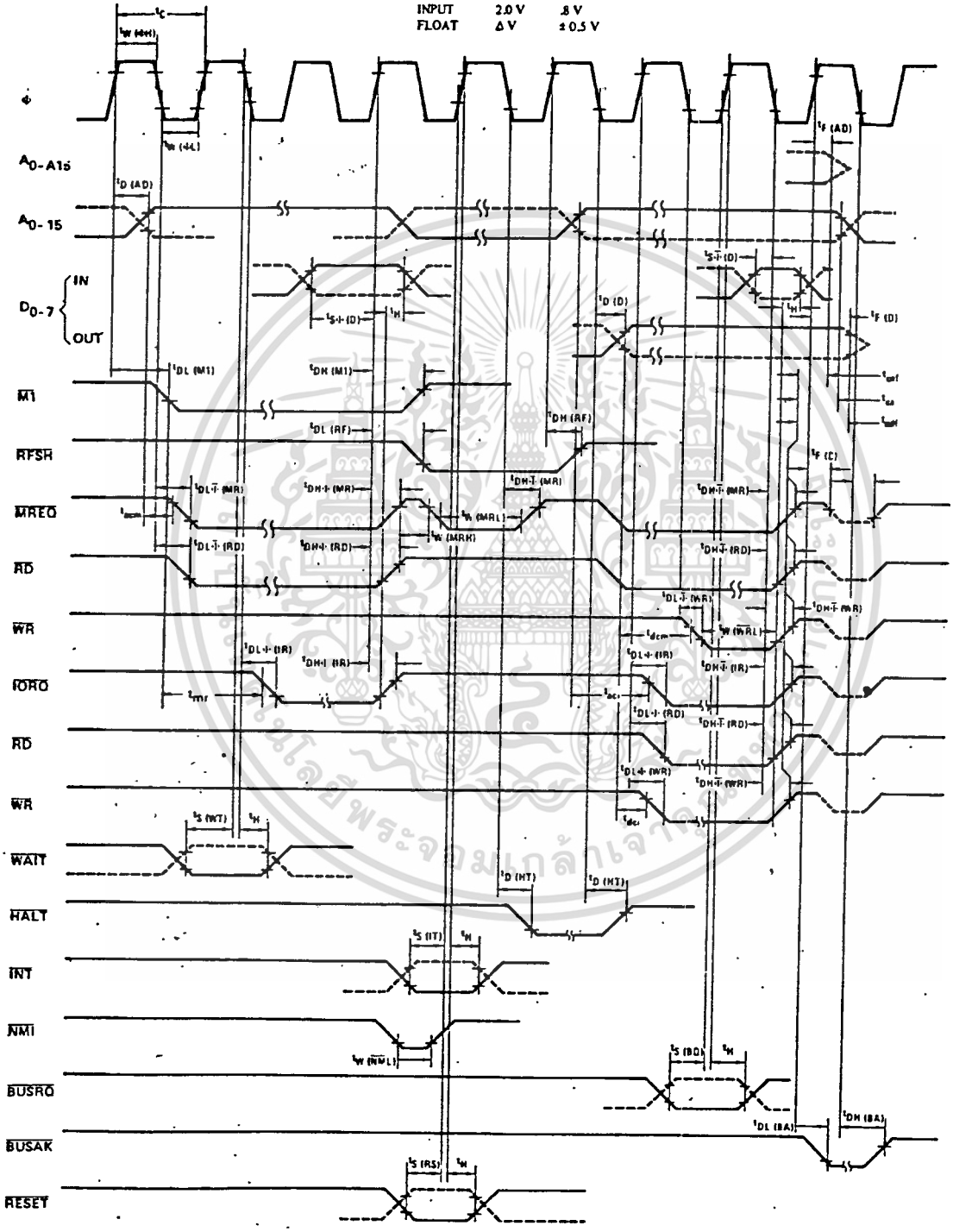


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{cc} -.6V	.45V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	Δ V	± 0.5 V



Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note. For Z80-CPU all AC and DC characteristics remain the same for the military grade parts except I_{cc}.

I_{cc} = 200 mA

Z80-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C
E – Extended 5V ±5% -40° to 85°C
M – Military 5V ±10% -55° to 125°C

Z80A-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.25	[12]	μsec	
	t_w (ΦH)	Clock Pulse Width, Clock High	110	[1]	nsec	
	t_w (ΦL)	Clock Pulse Width, Clock Low	110	2000	nsec	
	t_r, t_f	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	[3]		nsec	
	t_{caf}	Address Stable From RD or WR During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\bar{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From WR	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
MREQ	$t_{DL\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	MREQ Delay From Rising Edge of Clock, MREQ High		85	nsec	
	$t_{DH\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ High		85	nsec	
	t_w (MRL)	Pulse Width, MREQ Low	[8]		nsec	
	t_w (MRH)	Pulse Width, MREQ High	[9]		nsec	
IORQ	$t_{DL\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ Low		75	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ Low		85	nsec	
	$t_{DH\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ High		85	nsec	
	$t_{DH\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ High		85	nsec	
RD	$t_{DL\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD Low		85	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD Low		95	nsec	
	$t_{DH\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD High		85	nsec	
	$t_{DH\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD High		85	nsec	
WR	$t_{DL\Phi(WR)}$	WR Delay From Rising Edge of Clock, WR Low		65	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(WR)}$	WR Delay From Falling Edge of Clock, WR Low		80	nsec	
	$t_{DH\bar{\Phi}(WR)}$	WR Delay From Falling Edge of Clock, WR High		80	nsec	
	t_w (WRL)	Pulse Width, WR Low	[10]		nsec	
M1	$t_{DL(M1)}$	M1 Delay From Rising Edge of Clock, M1 Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	M1 Delay From Rising Edge of Clock, M1 High		100	nsec	
RFSH	$t_{DL(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH High		120	nsec	
WAIT	t_s (WT)	WAIT Setup Time to Falling Edge of Clock	70		nsec	
HALT	$t_{D(HT)}$	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
INT	t_s (IT)	INT Setup Time to Rising Edge of Clock	80		nsec	
NMI	t_w (NML)	Pulse Width, NMI Low	80		nsec	
BUSRQ	t_s (BQ)	BUSRQ Setup Time to Rising Edge of Clock	50		nsec	
BUSAK	$t_{DL(BA)}$	BUSAK Delay From Rising Edge of Clock, BUSAK Low		100	nsec	C_L = 50pF
	$t_{DH(BA)}$	BUSAK Delay From Falling Edge of Clock, BUSAK High		100	nsec	
RESET	t_s (RS)	RESET Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to Float (MREQ, IORQ, RD and WR)		80	nsec	
	t_{mi}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 70$

[8] $t_{w(\overline{MRL})} = t_c - 30$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 20$

[10] $t_{w(\overline{WRL})} = t_c - 30$

[11] $t_{mi} = 2t_c + t_{w(\Phi H)} + t_f - 65$

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

NOTES:

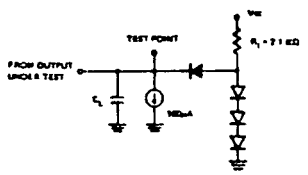
A. Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The RESET signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%
Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.



Load circuit for Output

intel Silicon Gate MOS 8255

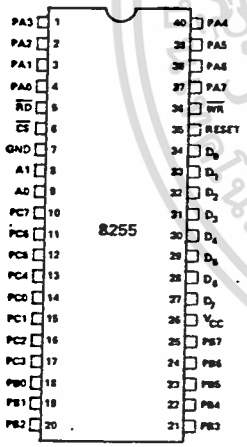
PROGRAMMABLE PERIPHERAL INTERFACE

- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Fully Compatible with MCS™-8 and MCS™-80 Microprocessor Families**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**
- **40 Pin Dual In-Line Package**
- **Reduces System Package Count**

The 8255 is a general purpose programmable I/O device designed for use with both the 8008 and 8080 microprocessors. It has 24 I/O pins which may be individually programmed in two groups of twelve and used in three major modes of operation. In the first mode (Mode 0), each group of twelve I/O pins may be programmed in sets of 4 to be input or output. In Mode 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining four pins three are used for handshaking and interrupt control signals. The third mode of operation (Mode 2) is a Bidirectional Bus mode which uses 8 lines for a bidirectional bus, and five lines, borrowing one from the other group, for handshaking.

Other features of the 8255 include bit set and reset capability and the ability to source 1mA of current at 1.5 volts. This allows darlington transistors to be directly driven for applications such as printers and high voltage displays.

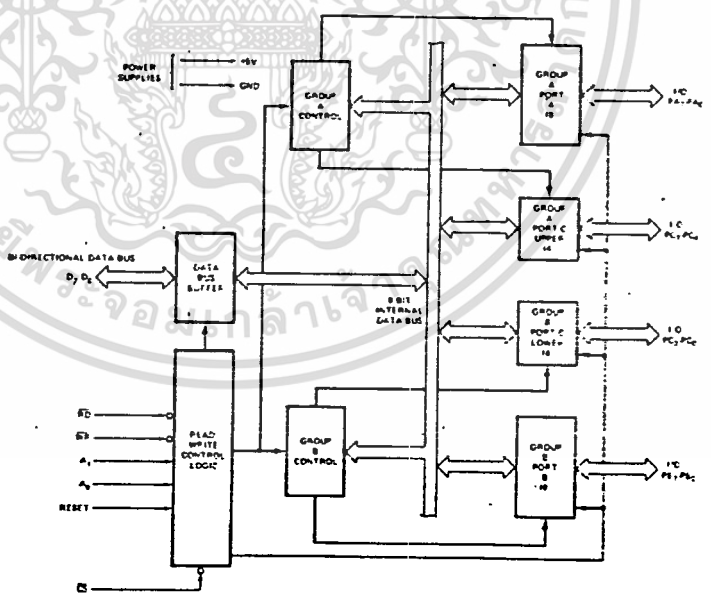
PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255 BLOCK DIAGRAM



8255 BASIC FUNCTIONAL DESCRIPTION

General

The 8255 is a Programmable Peripheral Interface (PPI) device designed for use in 8080 Microcomputer Systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the 8080 system bus. The functional configuration of the 8255 is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state, bi-directional, eight bit buffer is used to interface the 8255 to the 8080 system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions by the 8080 CPU. Control Words and Status information are also transferred through the Data Bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the 8080 CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select: A "low" on this input pin enables the communication between the 8255 and the 8080 CPU.

($\overline{RD}$)

Read: A "low" on this input pin enables the 8255 to send the Data or Status information to the 8080 CPU on the Data Bus. In essence, it allows the 8080 CPU to "read from" the 8255.

($\overline{WR}$)

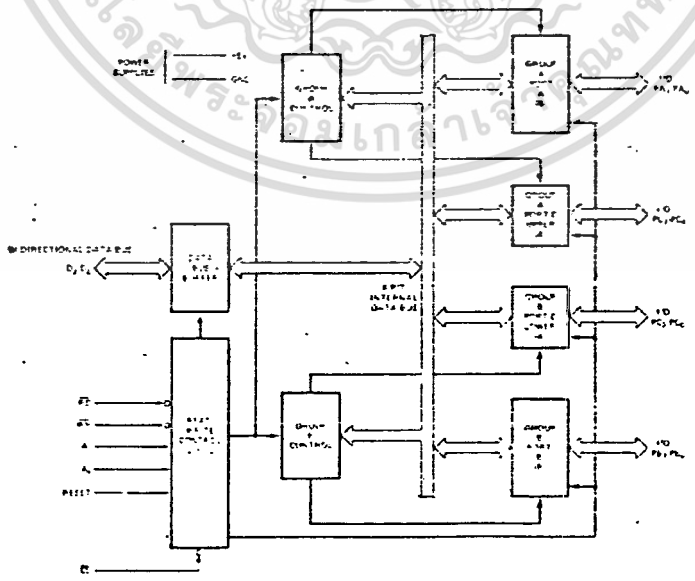
Write: A "low" on this input pin enables the 8080 CPU to write Data or Control words into the 8255.

(A_0 and A_1)

Port Select 0 and Port Select 1: These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the Control Word Register. They are normally connected to the least significant bits of the Address Bus (A_0 and A_1).

8255 BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS → 3-STATE
1	1	0	1	0	ILLEGAL CONDITION



8255 Block Diagram

(RESET)

Reset: A "high" on this input clears all internal registers including the Control Register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the 8080 CPU "outputs" a control word to the 8255. The control word contains information such as "mode", "bit set", "bit reset" etc. that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

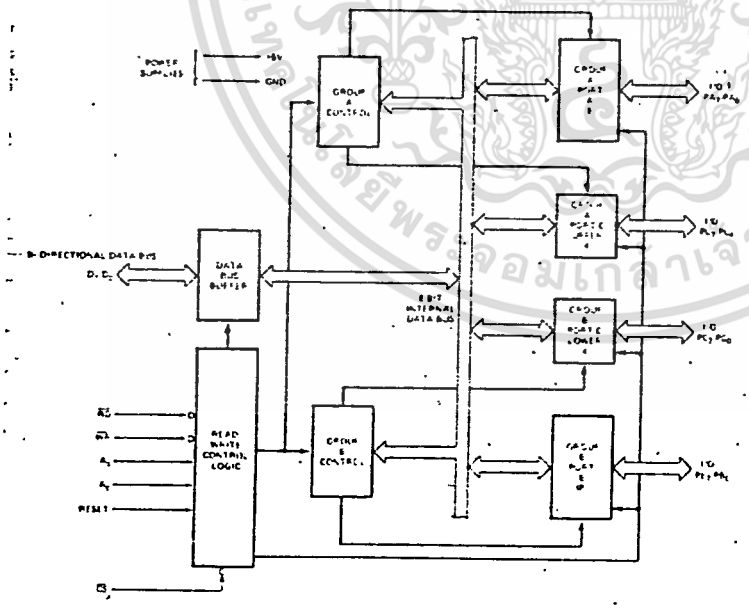
The 8255 contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.

Port A: One 8-bit data output latch/buffer and one 8-bit data input latch.

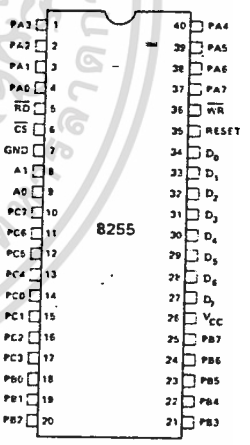
Port B: One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with Ports A and B.

8255 BLOCK DIAGRAM



PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255 DETAILED OPERATIONAL DESCRIPTION

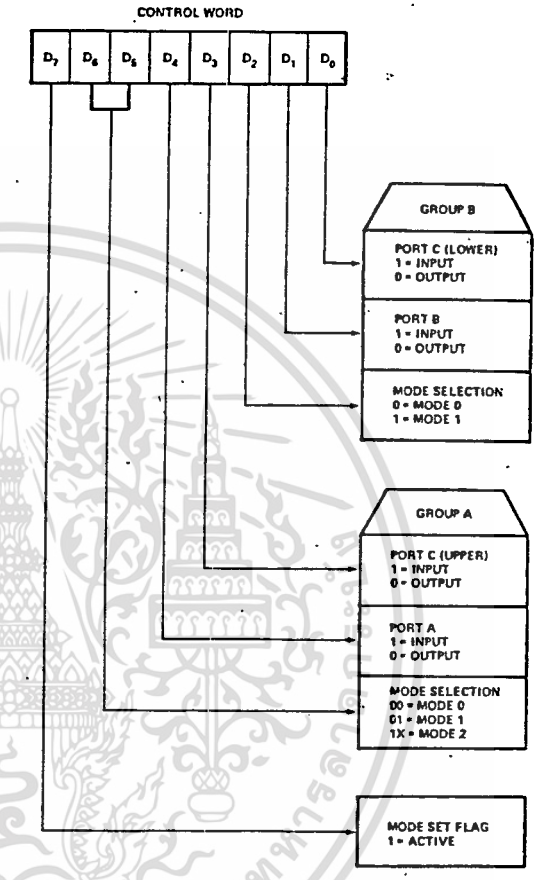
Mode Selection

There are three basic modes of operation that can be selected by the system software:

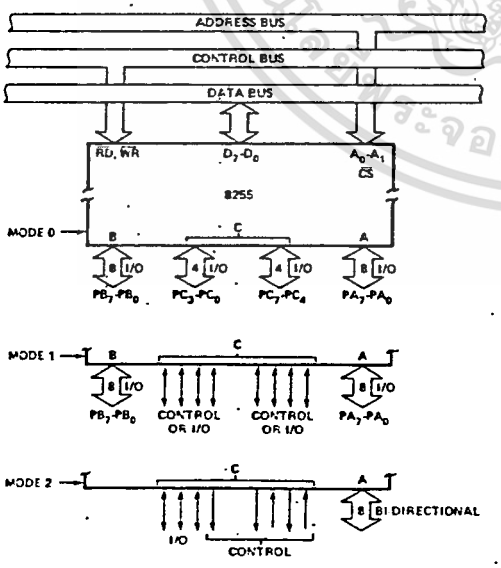
Mode 0 – Basic Input/Output
Mode 1 – Strobed Input/Output
Mode 2 – Bi-Directional Bus

When the RESET input goes "high" all ports will be set to the Input mode (i.e., all 24 lines will be in the high impedance state). After the RESET is removed the 8255 can remain in the Input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single OUTput instruction. This allows a single 8255 to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.



Mode Definition Format

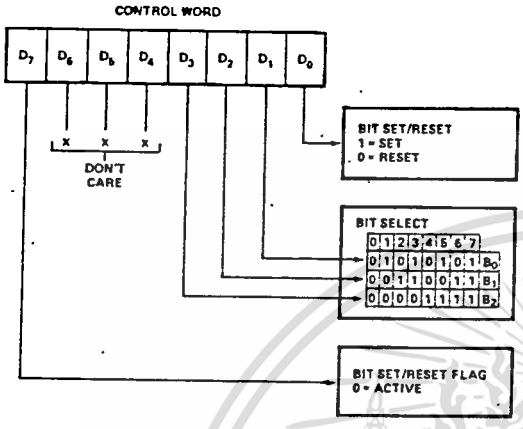


Basic Mode Definitions and Bus Interface

The Mode definitions and possible Mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255 has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.



Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255 is programmed to operate in Mode 1 or Mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from Port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the Bit set/reset function of Port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without effecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) — INTE is SET — Interrupt enable
(BIT-RESET) — INTE is RESET — Interrupt disable

Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

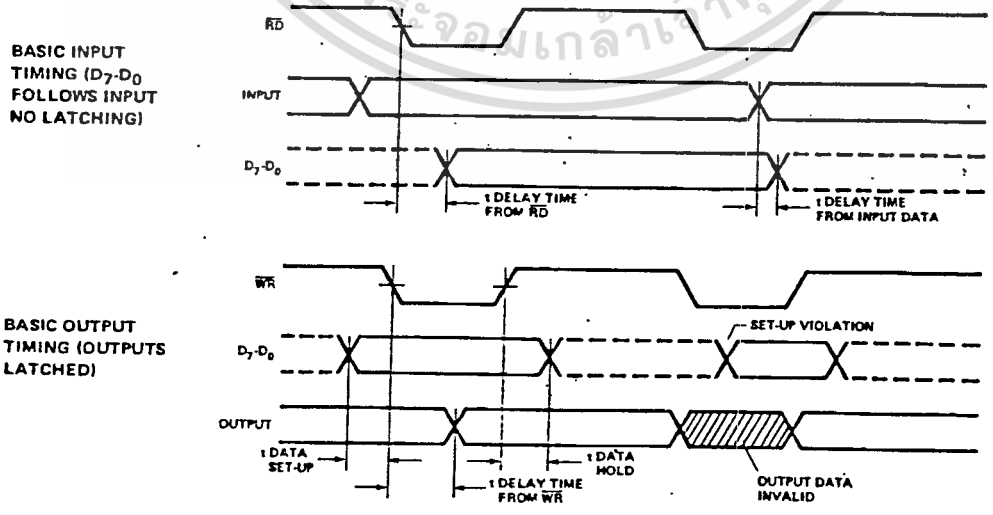
Operating Modes

Mode 0 (Basic Input/Output)

This functional configuration provides simple Input and Output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

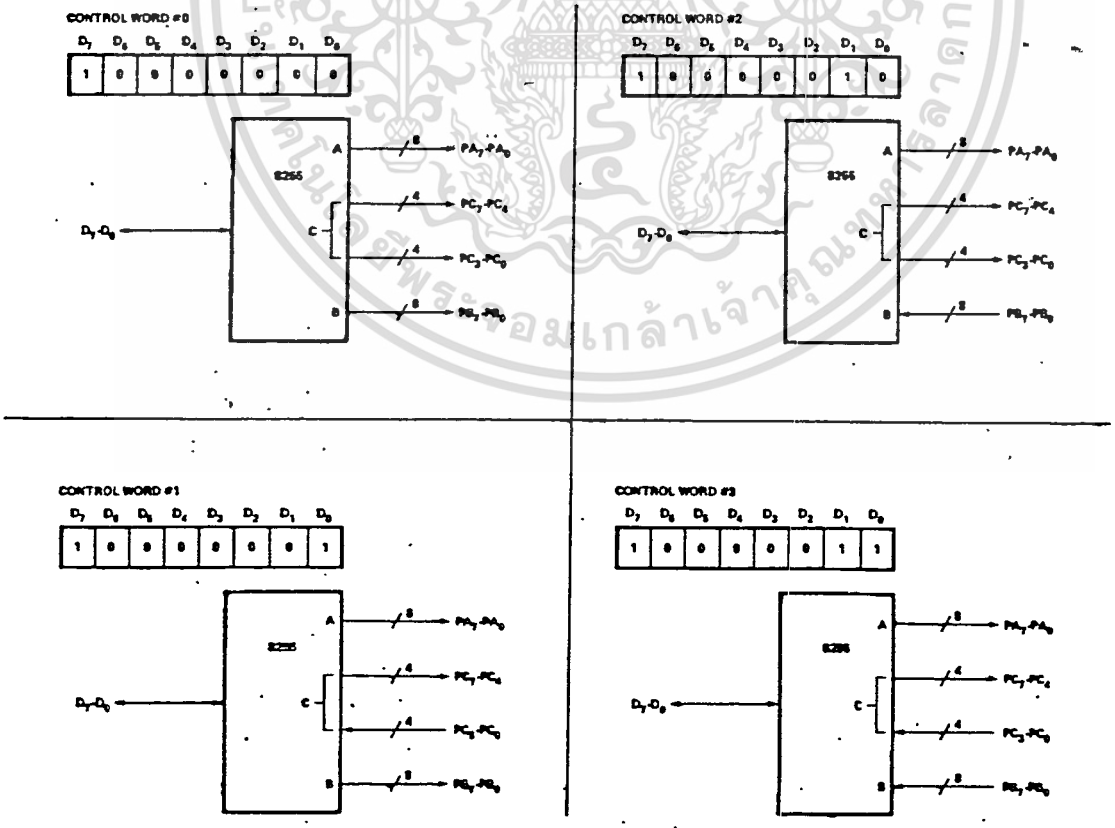


Mode 0 Timing

MODE 0 PORT DEFINITION CHART

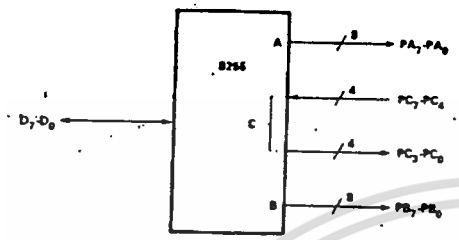
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 CONFIGURATIONS



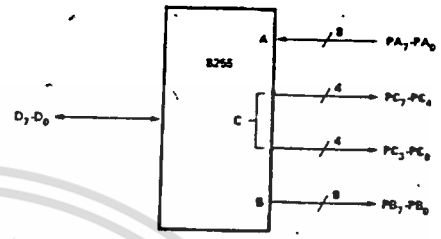
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



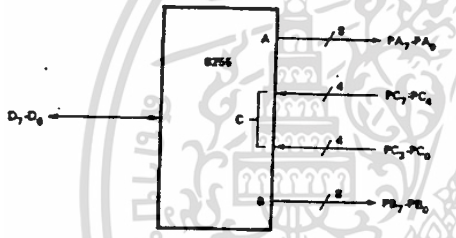
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



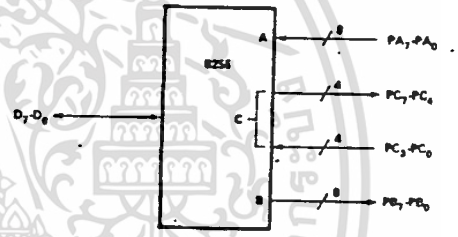
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



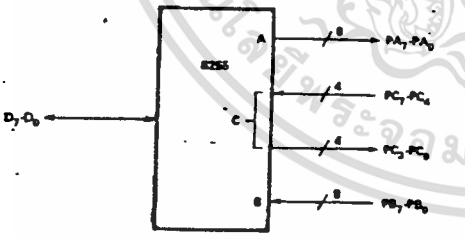
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



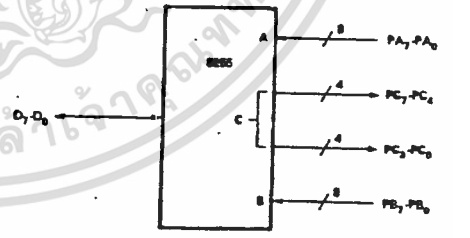
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



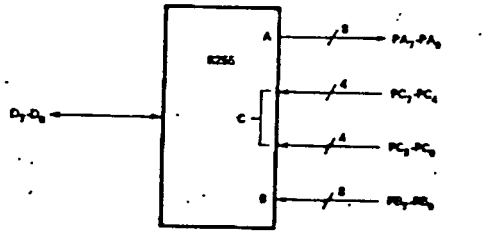
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



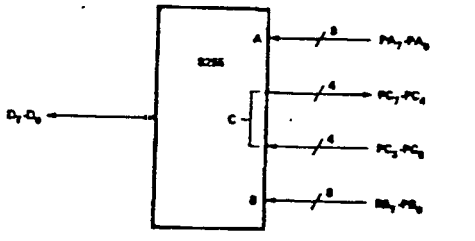
CONTROL WORD #7

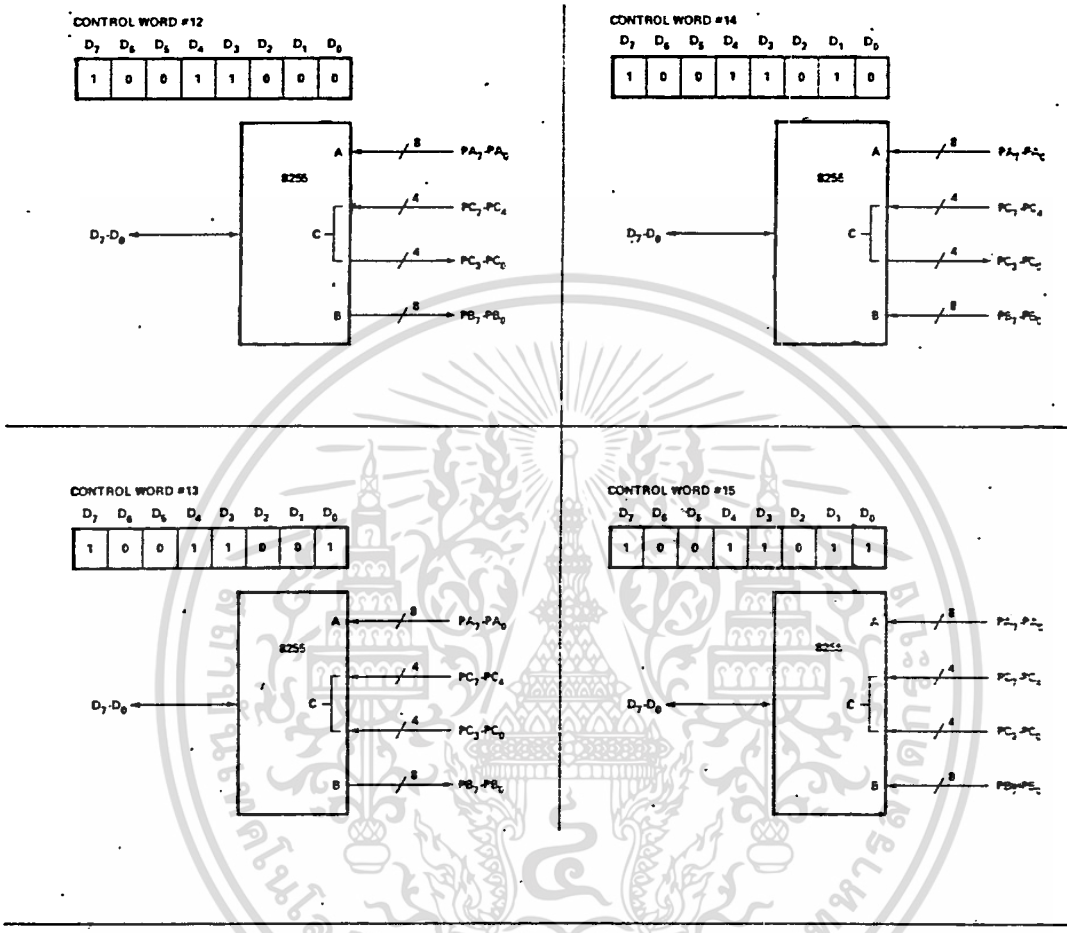
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1





Operating Modes

Mode 1 (Strobed Input/Output)

This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In Mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input)

A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by the falling edge of the STB input and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

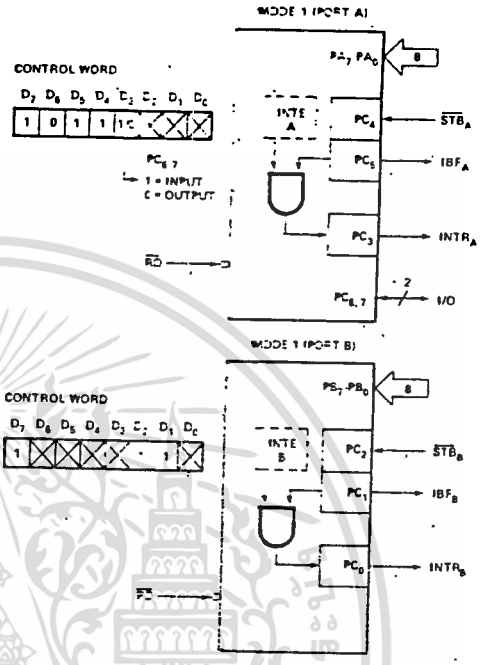
A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the rising edge of $\overline{STB}$ if IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

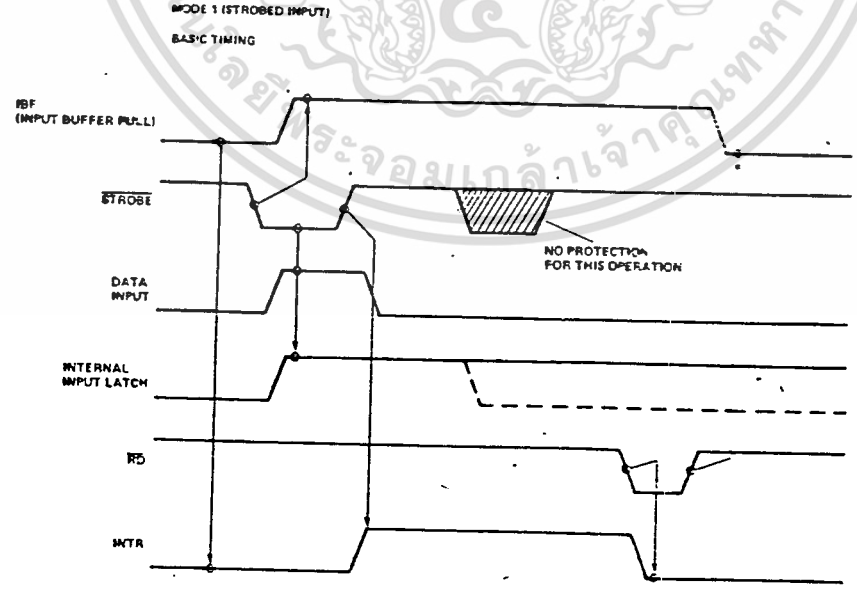
Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.



Mode 1 Input



Basic Timing Input

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F)

The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the WR input and reset by the falling edge of the $\overline{ACK}$ input signal.

$\overline{ACK}$ (Acknowledge Input)

A "low" on this input informs the 8255 that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request)

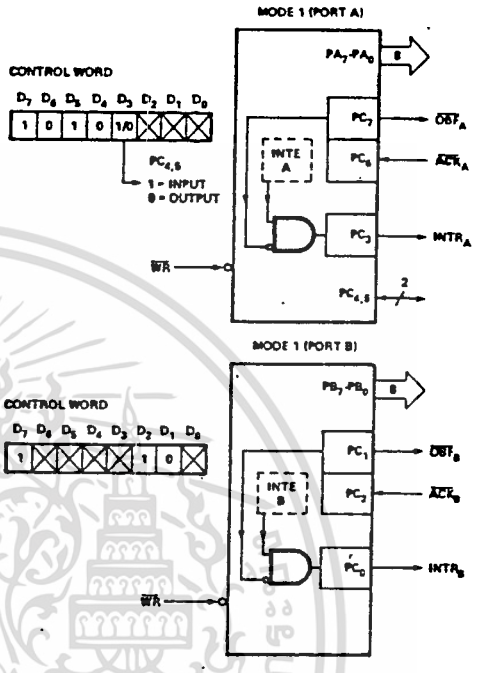
A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set by the rising edge of $\overline{ACK}$ if $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

INTE A

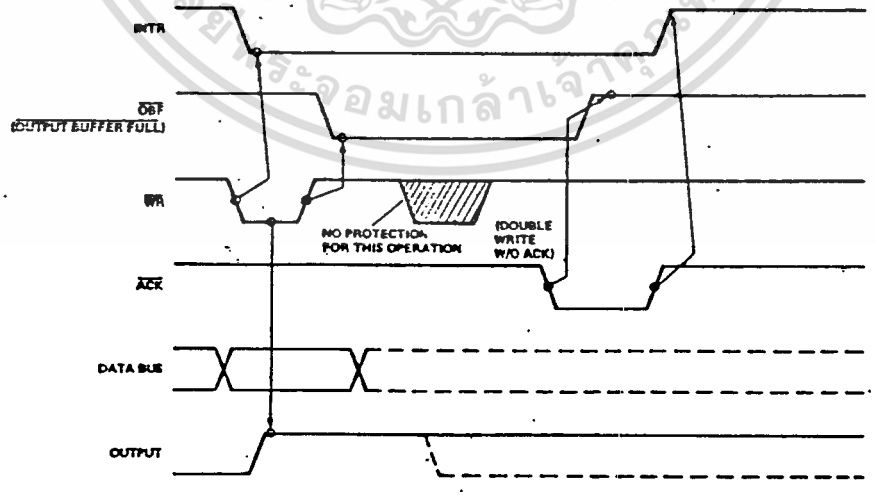
Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.



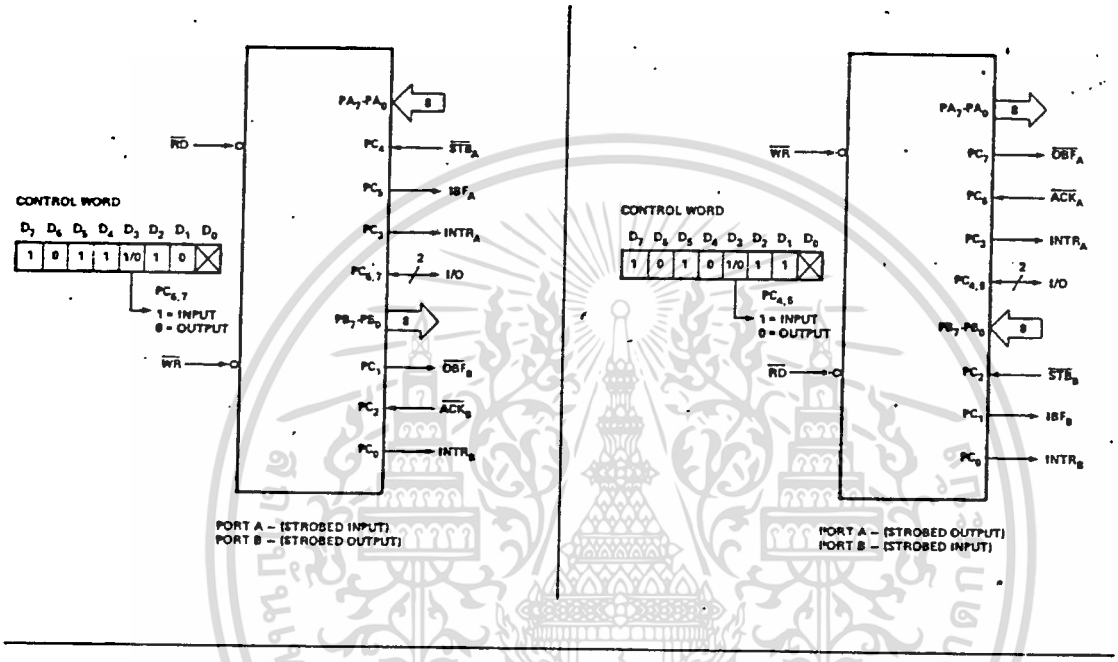
Mode 1 Output



Basic Timing Output

Combinations of Mode 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.



Operating Modes

Mode 2 (Strobed Bi-Directional Bus I/O)

This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bi-directional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to Mode 1. Interrupt generation and enable/disable functions are also available.

Mode 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bi-Directional Bus I/O Control Signal Definition

INTR (Interrupt Request)

A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full)

The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to Port A.

$\overline{ACK}$ (Acknowledge)

A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high-impedance state.

INTE 1 (The INTE Flip-Flop associated with $\overline{OBF}$)

Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input)

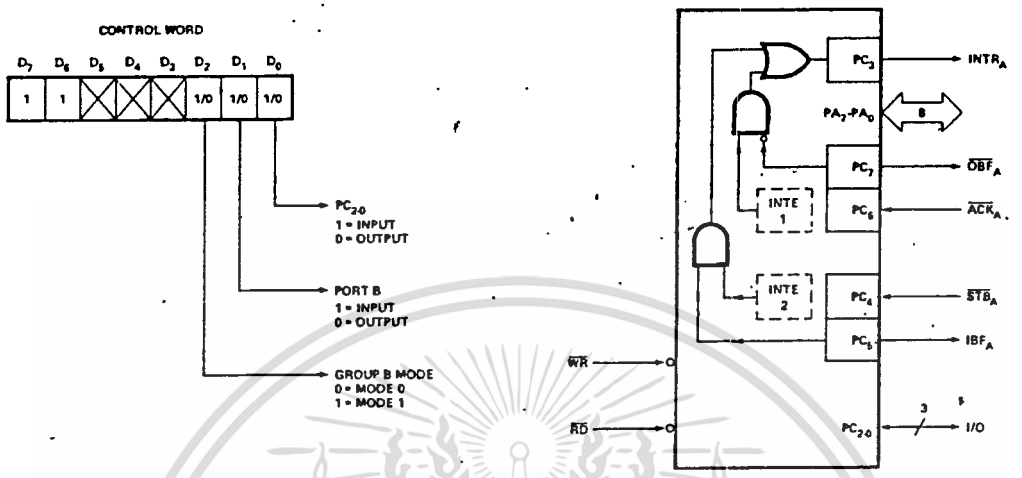
A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that data has been loaded into the input latch.

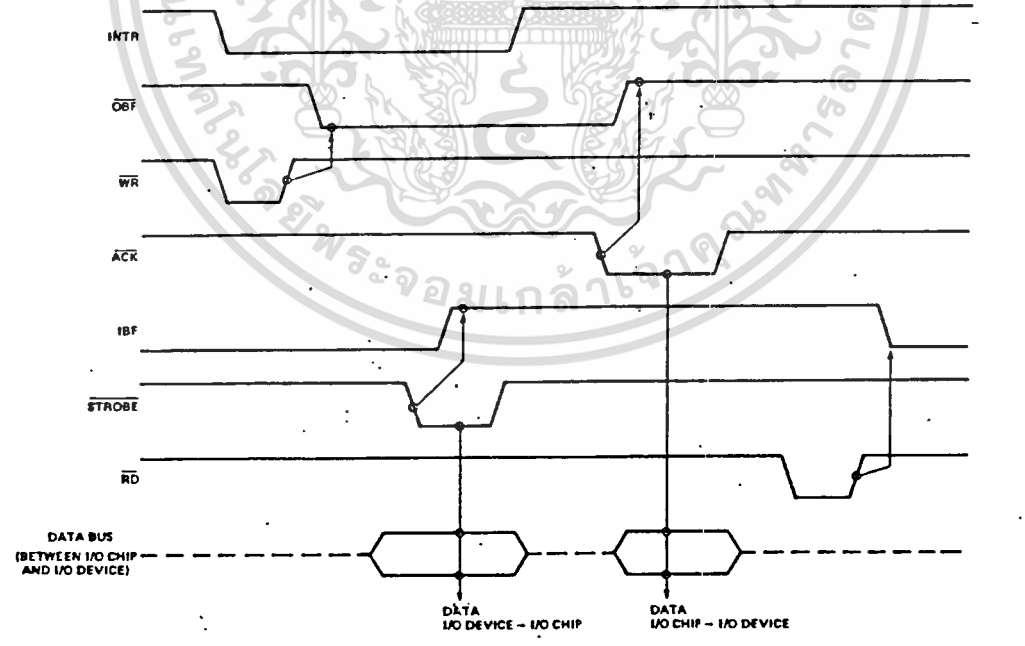
INTE 2 (The INTE Flip-Flop associated with IBF)

Controlled by bit set/reset of PC_4.



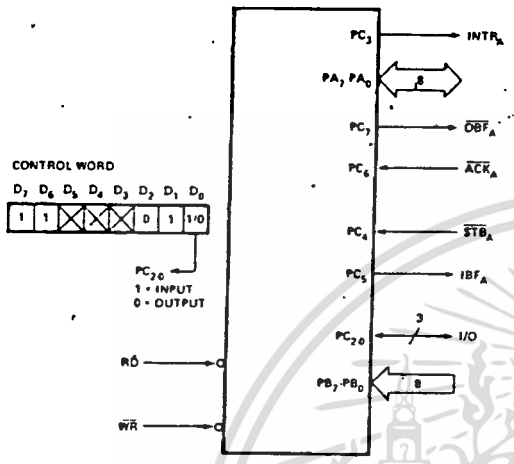
Mode 2 Control Word

Mode 2

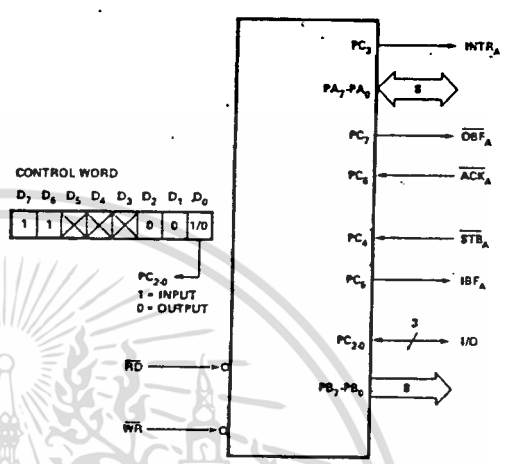


Mode 2 (Bi-directional) Timing

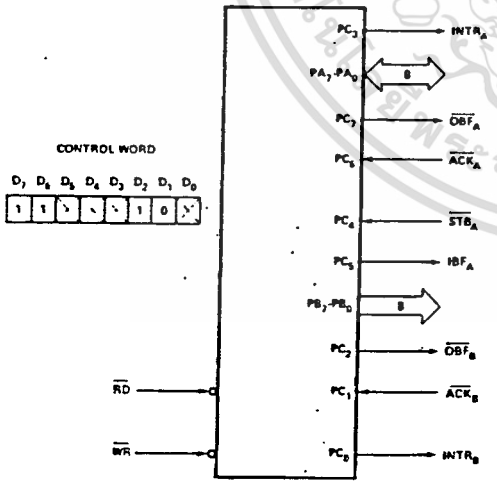
MODE 2 AND MODE 0 (INPUT)



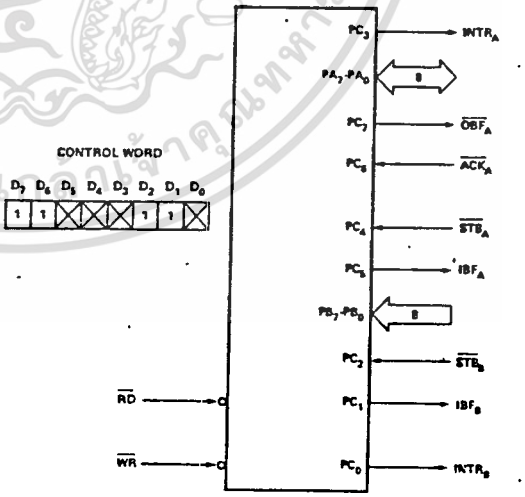
MODE 2 AND MODE 0 (OUTPUT)



MODE 2 AND MODE 1 (OUTPUT)



MODE 2 AND MODE 1 (INPUT)



Mode 2 Combinations

MODE DEFINITION SUMMARY TABLE

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	⟷	
PA_1	IN	OUT	IN	OUT	⟷	
PA_2	IN	OUT	IN	OUT	⟷	
PA_3	IN	OUT	IN	OUT	⟷	
PA_4	IN	OUT	IN	OUT	⟷	
PA_5	IN	OUT	IN	OUT	⟷	
PA_6	IN	OUT	IN	OUT	⟷	
PA_7	IN	OUT	IN	OUT	⟷	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs –
All input lines can be accessed during a normal Port C read.

If Programmed as Outputs –
Bits in C upper (PC_7-PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3-PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

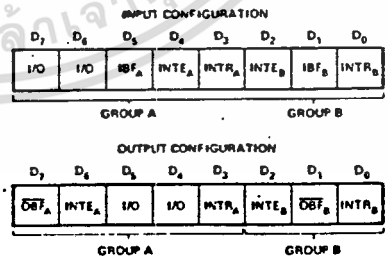
Source Current Capability on Port B and Port C

Any set of eight output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

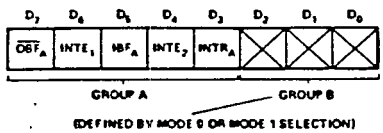
Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.



Mode 1 Status Word Format

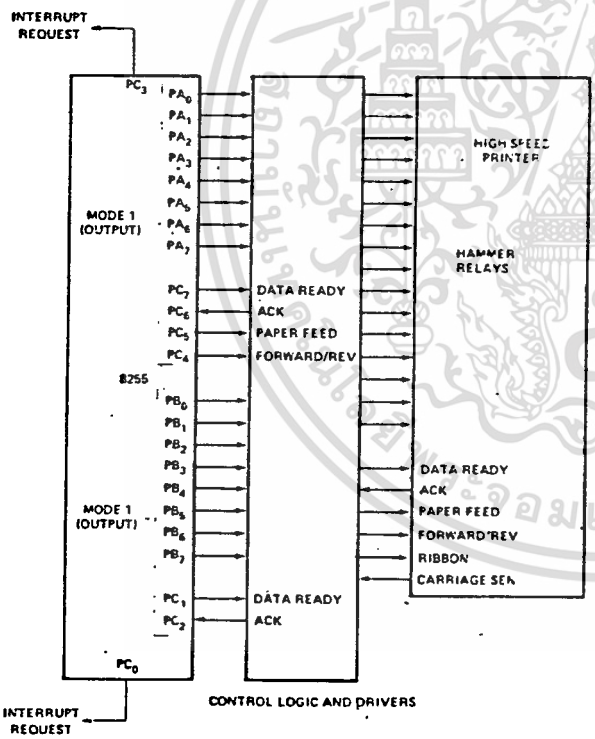


Mode 2 Status Word Format

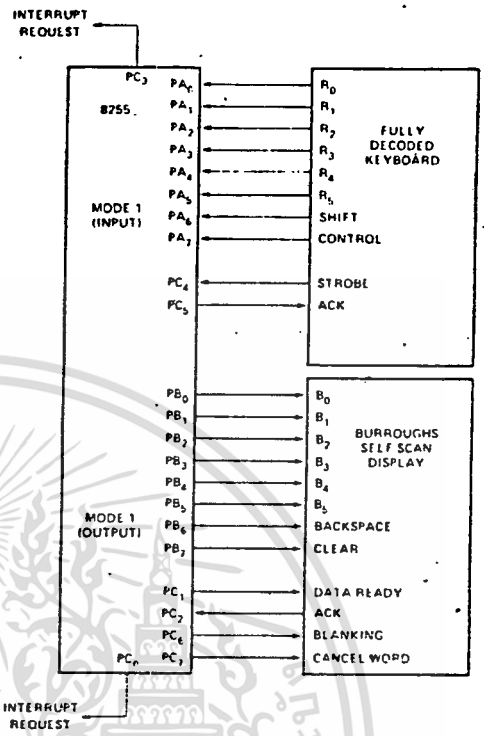
APPLICATIONS OF THE 8255

The 8255 is a very powerful tool for interfacing peripheral equipment to the 8080 microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

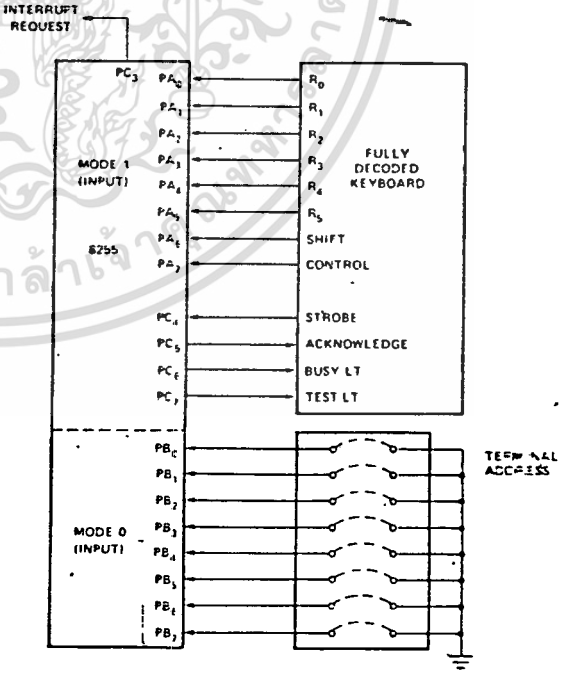
Each peripheral device in a Microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255 is programmed by the I/O service routine and becomes an extension of the systems software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the Detailed Operational Description, a control word can easily be developed to initialize the 8255 to exactly "fit" the application. Here are a few examples of typical applications of the 8255.



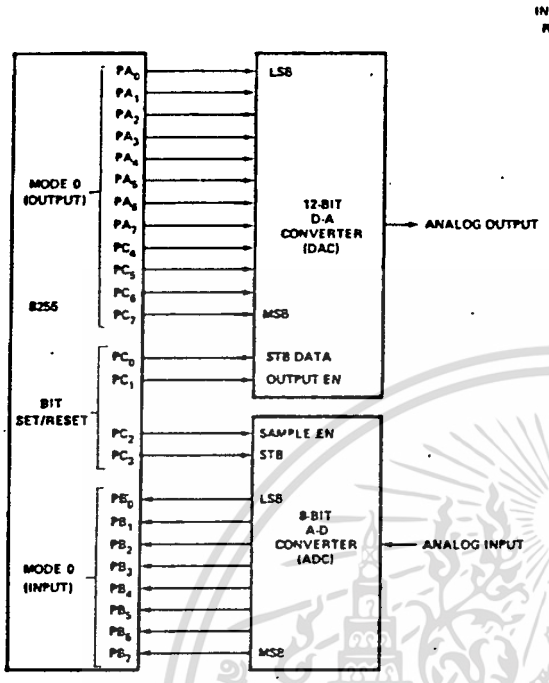
Printer Interface



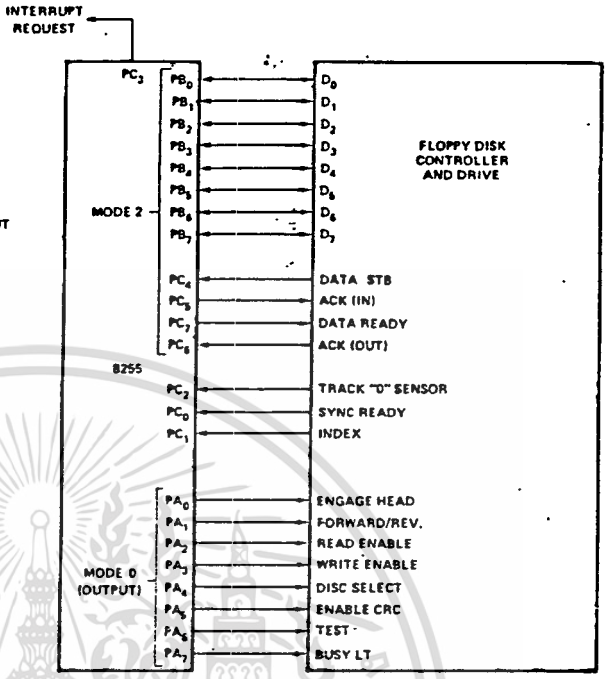
Keyboard and Display Interface



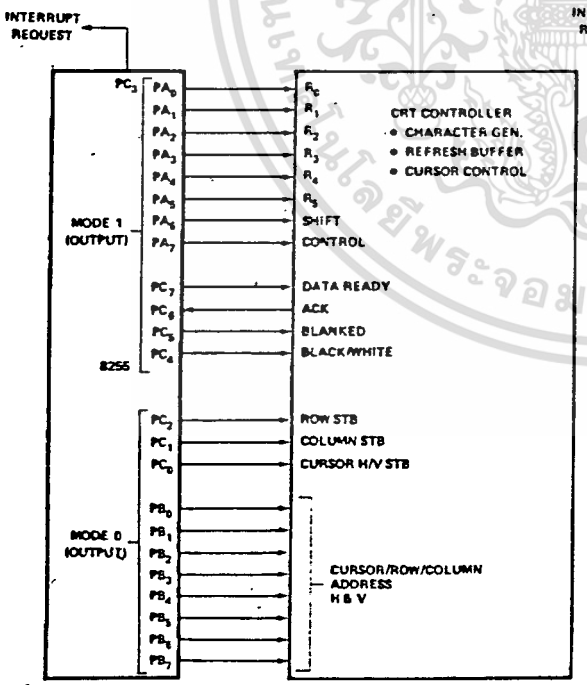
Keyboard and Terminal Address Interface



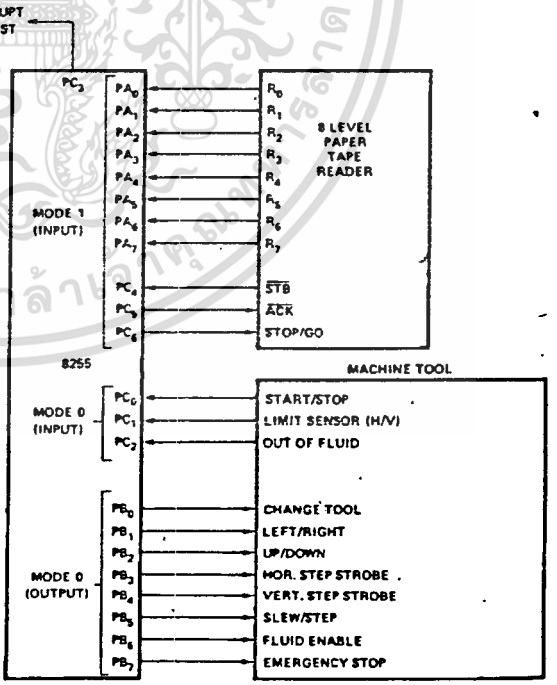
Digital to Analog, Analog to Digital



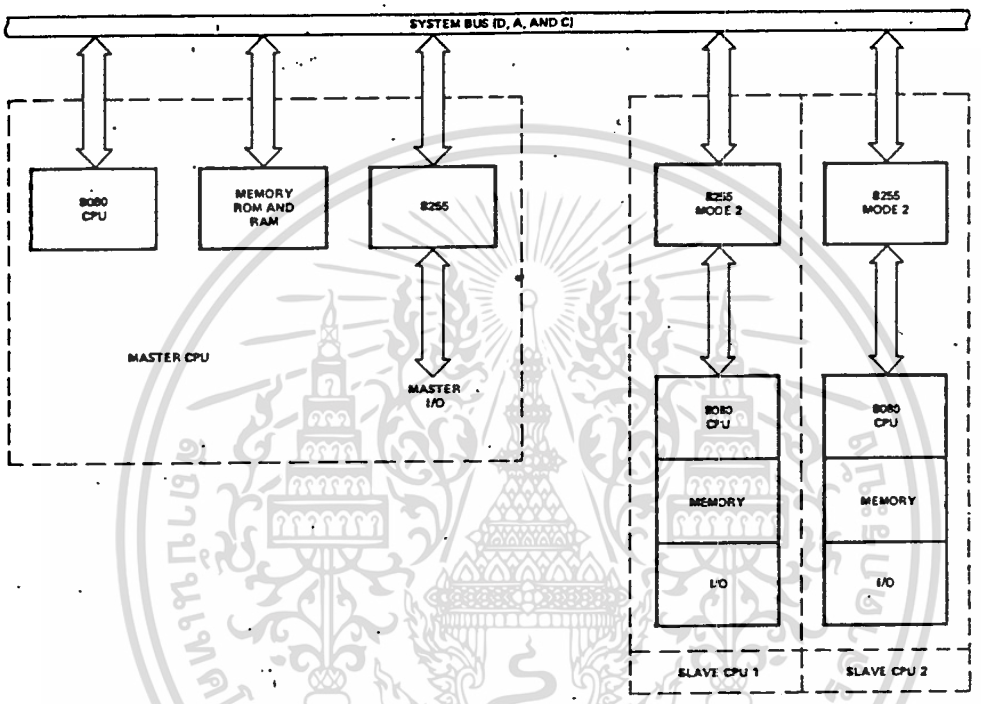
Basic Floppy Disc Interface



Basic CRT Controller Interface



Machine Tool Controller Interface



Distributed Intelligence Multi-Processor Interface

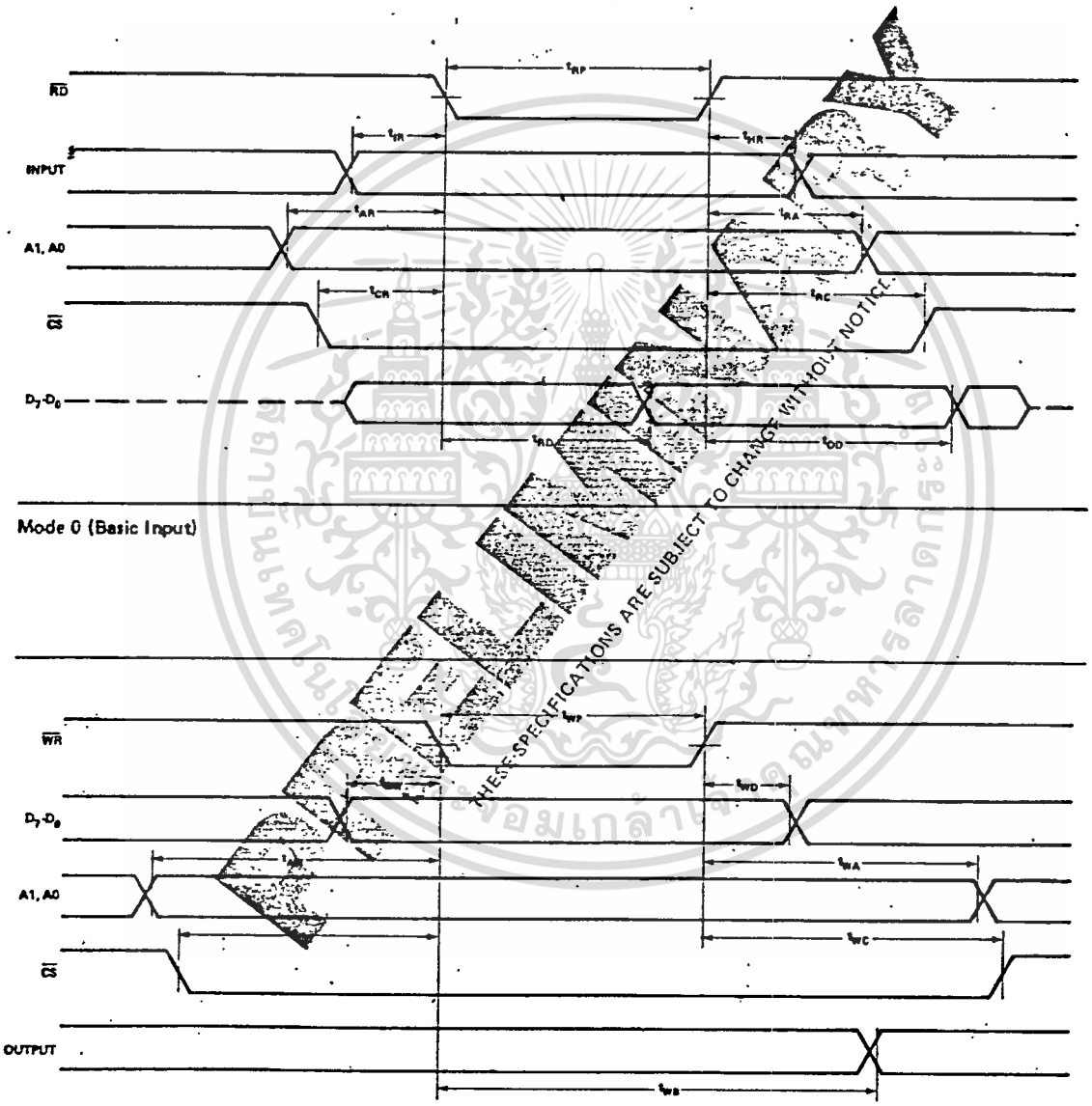
D.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$; $V_{CC} = +5V \pm 5\%$; $V_{SS} = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage			.8	V	
V_{IH}	Input High Voltage	2.0			V	
V_{OL}	Output Low Voltage			.4	V	$I_{OL} = 1.6mA$
V_{OH}	Output High Voltage	2.4			V	$I_{OH} = -50\mu A$ ($-100\mu A$ for D.B. Port)
I_{OH}[1]	Darlington Drive Current		2.0		mA	$V_{OH} = 1.5V$, $R_{EXT} = 390\Omega$
I_{CC}	Power Supply Current		40		mA	

NOTE:
1. Available on 8 pins only.

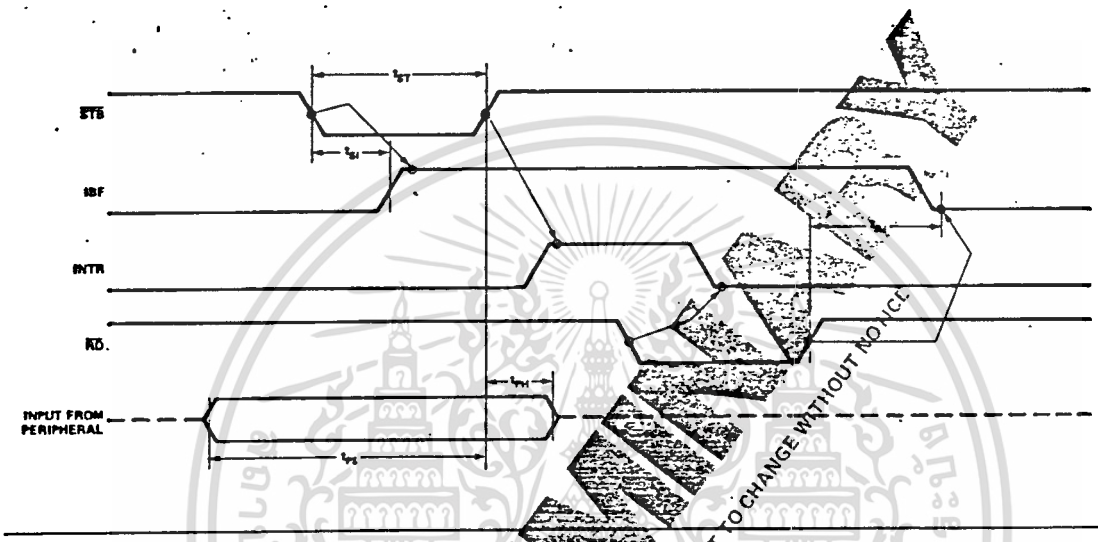
A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$; $V_{CC} = +5V \pm 5\%$; $V_{SS} = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
t_{WP}	Pulse Width of $\overline{WR}$		250		ns	
t_{DW}	Time D.B. Stable Before $\overline{WR}$		10		ns	
t_{WD}	Time D.B. Stable After $\overline{WR}$		10		ns	
t_{AW}	Time Address Stable Before $\overline{WR}$		25		ns	
t_{WA}	Time Address Stable After $\overline{WR}$		10		ns	
t_{CW}	Time $\overline{CS}$ Stable Before $\overline{WR}$		25		ns	
t_{WC}	Time $\overline{CS}$ Stable After $\overline{WR}$		10		ns	
t_{WB}	Delay From $\overline{WR}$ To Output		200		ns	
t_{RP}	Pulse Width of $\overline{RD}$		300		ns	
t_{IR}	$\overline{RD}$ Set-Up Time		50		ns	
t_{HR}	Input Hold Time		10		ns	
t_{RD}	Delay From $\overline{RD} = 0$ To System Bus		200		ns	
t_{OD}	Delay From $\overline{RD} = 1$ To System Bus		100		ns	
t_{AR}	Time Address Stable Before $\overline{RD}$		25		ns	
t_{CR}	Time $\overline{CS}$ Stable Before $\overline{RD}$		25		ns	
t_{AK}	Width Of $\overline{ACK}$ Pulse		100		ns	
t_{ST}	Width Of $\overline{STB}$ Pulse		100		ns	
t_{PS}	Set-Up Time For Peripheral		200		ns	
t_{PH}	Hold Time For Peripheral		10		ns	
t_{RA}	Hold Time for A_1, A_0 After $\overline{RD} = 1$		10		ns	
t_{RC}	Hold Time For $\overline{CS}$ After $\overline{RD} = 1$		10		ns	
t_{AD}	Time From $\overline{ACK} = 0$ To Output (Mode 2)		200		ns	
t_{KD}	Time From $\overline{ACK} = 1$ To Output Floating		250		ns	
t_{WO}	Time From $\overline{WR} = 1$ To $\overline{OBF} = 0$		50		ns	
t_{AO}	Time From $\overline{ACK} = 0$ To $\overline{OBF} = 1$		200		ns	
t_{SI}	Time From $\overline{STB} = 0$ To IBF		200		ns	
t_{RI}	Time From $\overline{RD} = 1$ To IBF = 0		200		ns	

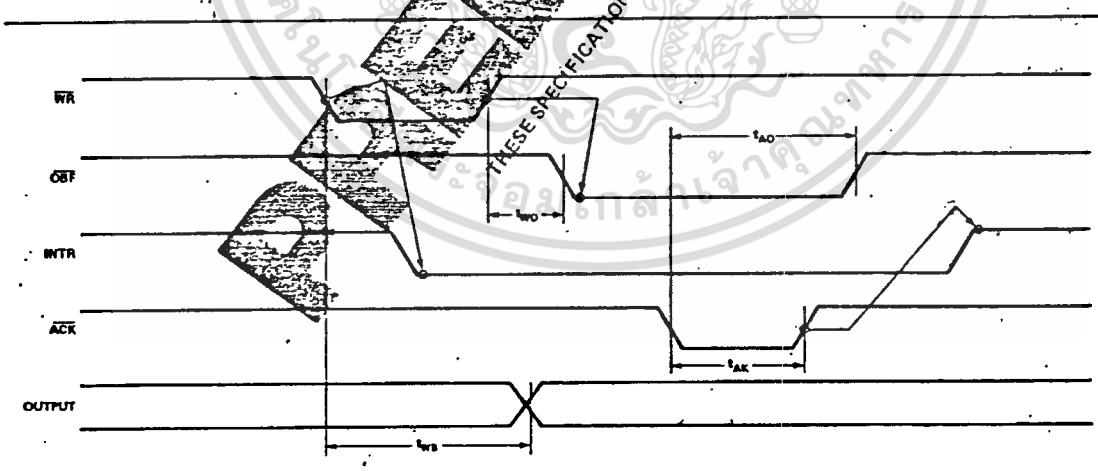


Mode 0 (Basic Input)

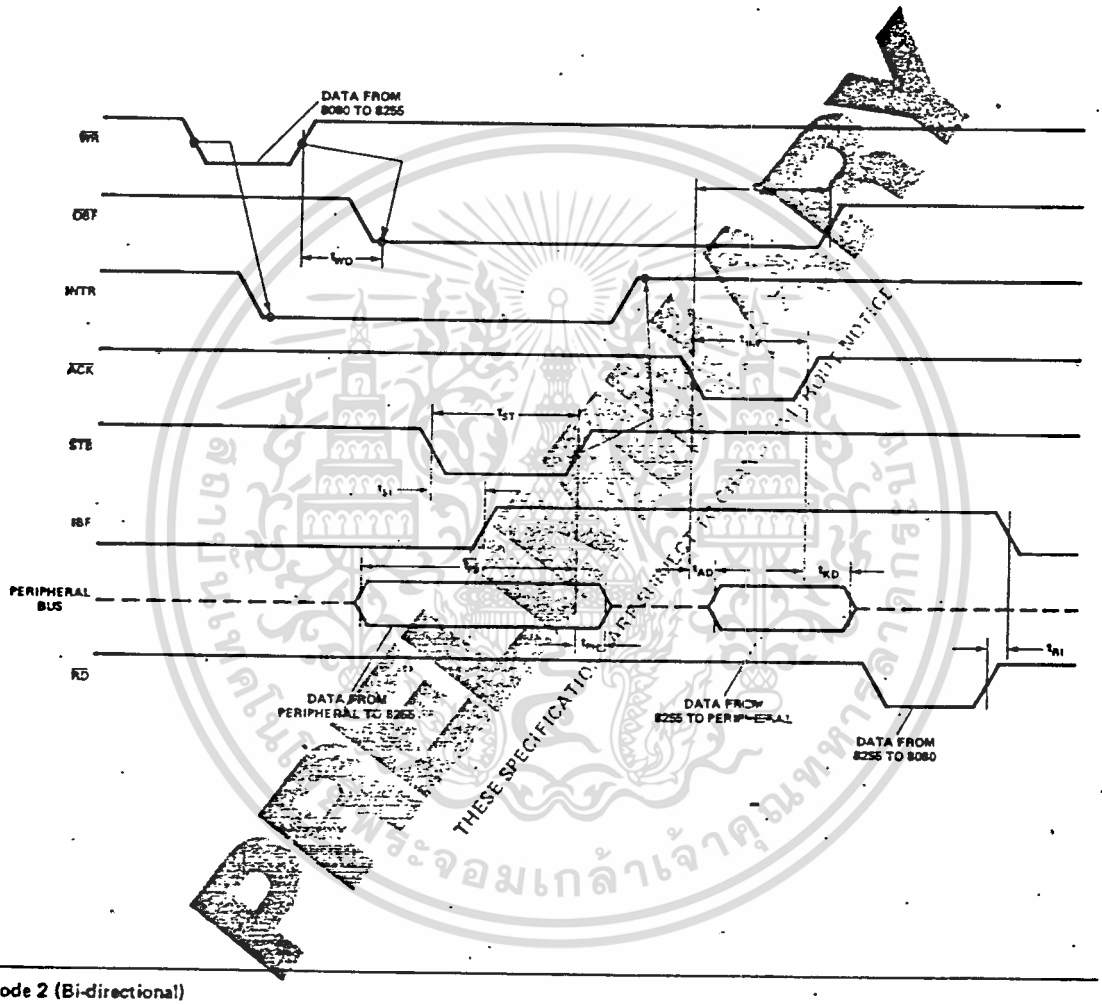
Mode 0 (Basic Output)



Mode 1 (Strobed Input)



Mode 1 (Strobed Output)



Mode 2 (Bi-directional)

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ อาจารย์ที่ปรึกษา ผศ.นิกร สุขุตมตันติ ที่ให้คำปรึกษาและแนะนำ และขอขอบคุณ คุณเกรียงศักดิ์ บุญเสริมสุวงศ์ คุณกอบกิจ เติมผาติ ที่ให้คำแนะนำตลอดจนเอื้อเฟื้อเครื่องไมโครคอมพิวเตอร์ในการทำโครงงานนี้

หนังสืออ้างอิง

1. กฤษดา วิศวธีรานนท์, ยืน ภู่วรวรรณ, "ไมโครโปรเซสเซอร์" สมาคมส่งเสริม-เทคโนโลยี(ไทย-ญี่ปุ่น), 2526
2. ยืน ภู่วรวรรณ, วัฒนา เชียงกูล, "ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์" บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2527
3. บริษัท ซีเอ็ดยูเคชั่น จำกัด , "TTL DATA BOOK" บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2527
4. Hall Douglas V. , "MICROPROCESSORS AND INTERFACING", Fong & Sons, 309 p., 1987.
5. Jammes W.Coffron, "Z-80 Application", SYBEX INC., 1983
6. RODNAY ZARKS "PROGRAMMING THE Z-80", SYBEX INC., 1980
7. TEXAS INSTRUMENTS "The TTL Data Book", U.S.A, pp.134-140
8. National Semiconductor "LINEAR DATABOOK", California, pp.9-33 - 9.39