เครื่องเทเล็กซ์บนไมโครคอมพิวเตอร์
ADVANCE TELEX ON PC

โดย

นาย เกรียงไกร มงคลศิริ

นาย ฤกษ์ชัย สมานสุขมาล

นาย สมประสงค์ พัฒนคุณเจริญกิจ

บริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาอุตสาหกรรมศาสตรบัณฑิต
สาขาเทคโนโลยีโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

ADVANCE TELEX ON PC

Mr. KREINGKRI MONGKOLSIRI
Mr. ROUKCHAI SMANSUKUMAL
Mr. SOMPRASONG PATTANAKUNJARERNKIJ

Project Report Submitted in Partial Fulfillment of The Requirements
For The Bachelor's Degree
Department of Industrial Technology
faculty of Engineering
King Mongkut's Institute of Technology
1993

หัวข้อปริญญานิพนธ์ เครื่องเทเล็กซ์บนไมโครคอมพิวเตอร์

โดย	นาย เกรียงไกร	มงคลศิริ
	นาย ฤกษ์ชัย	สมานสุขุมาล
	นายสมประสงค์	พัฒนคุณเจริญกิจ
ภาควิชา	เทคนิคอุตสาหกรรม	
อาจารย์ที่ปรึกษา	อาจารย์ ชวลิต	เบญจางคประเสริฐ
	อาจารย์ ไพศาล	สิทธิโยภาสกุล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อนุมัติให้นับปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของ การศึกษาตามหลักสูตรปริญญา
อุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

ADVANCE TELEX ON PC

นาย เกรียงไกร มงคลศิริ 34132201
นาย ฤกษ์ชัย สมานสุขุมาล 34132219
นาย สมประสงค์ พัฒนคุณเจริญกิจ 34132231

อาจารย์ที่ปรึกษา

อาจารย์ ชวลิต เบญจางคประเสริฐ
อาจารย์ ไพศาล สิทธิโยภาสกุล

ปีการศึกษา 2536

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เสนอการนำเอาเครื่องไมโครคอมพิวเตอร์ (PC) ไปประยุกต์ใช้งานร่วมกับระบบเทเล็กซ์ โดยใช้คอมพิวเตอร์ส่วนบุคคล ทำหน้าที่แทนเครื่องเทเล็กซ์เครื่องหนึ่ง โดยทำงานร่วมกับชุดอินเตอร์เฟสและชุดควบคุมที่สร้างขึ้น ทำหน้ารับส่งข้อความเทเล็กซ์ โดยเพิ่มความสามารถพิเศษให้สามารถเก็บข้อความเทเล็กซ์ในขณะที่เครื่องไมโครคอมพิวเตอร์ปิดอยู่หรือทำงานอื่นอยู่ โดยการทำงานจะเป็นไปอย่างอัตโนมัติ พร้อมกับเรียกอุปกรณ์ชุดนี้ว่า เครื่องเทเล็กซ์บนไมโครคอมพิวเตอร์ ซึ่งจากแนวความคิดดังกล่าวจะช่วยลดต้นทุนการจัดซื้อ และบำรุงรักษาเครื่องเทเล็กซ์ได้อย่างมากอีกทั้งทำให้ใช้งานระบบเทเล็กซ์ได้สะดวกขึ้น

สารบัญ

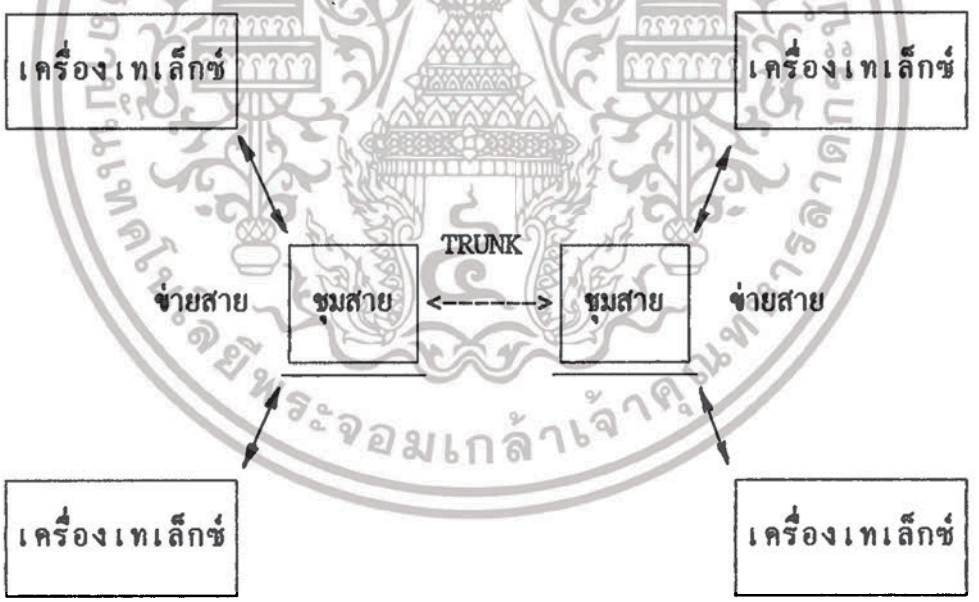
บทที่ 1
บทนำ

ความรู้ทั่วไปเกี่ยวกับระบบ TELEX

คำว่าเทเล็กซ์เป็นคำภาษาอังกฤษ "TELEX" ย่อมาจาก TELEPRINTER และ EXCHANGE เป็นบริการที่การสื่อสารแห่งประเทศไทย เปิดให้เช่าใช้บริการ ซึ่งผู้ใช้สามารถ รับ-ส่ง ข้อความเป็นอักษรผ่านเครื่องเทเล็กซ์ ไปยังผู้เช่ารายอื่นที่อยู่ภายในและภายนอกประเทศได้โดยตรง พร้อมทั้งสำเนาทั้งฝ่ายผู้ส่ง และฝ่ายผู้รับเป็นหลักฐานเป็นเอกสารที่ยอมรับกันในทางธุรกิจทั่วโลก ซึ่งผู้ใช้จะสามารถมั่นใจได้ว่าการติดต่อกับปลายทางเป็นไปได้อย่างถูกต้อง

อุปกรณ์สำหรับการให้บริการเทเล็กซ์

ในทางเทคนิคจะแบ่งอุปกรณ์เป็น 3 ส่วน คือ อุปกรณ์ด้านชุมสาย ตัวเครื่องและข่ายสาย



รูปที่ 1 แสดงให้เห็นอุปกรณ์ทั้ง 3 ส่วน

เครื่องเทเล็กซ์มี 2 ชนิด คือ

1. เครื่องชนิด 5 ยูนิตโค๊ต เป็นเครื่องที่มีอักขระภาษาอังกฤษ พร้อมตัวเลขและเครื่องหมายต่าง ๆ

2. เครื่องชนิด 6 ยูนิตโค๊ต เป็นเครื่อง 2 ภาษาคือ ไทย,อังกฤษ พร้อมตัวเลขและเครื่องหมายต่าง ๆ

เครื่องทั้ง 2 ชนิด (5,6 ยูนิต) สามารถใช้เรียกติดต่อกันได้ โดยผ่านอุปกรณ์แปลงสัญญาน (Code Convertor) ที่ชุมสาย เครื่องต่างชนิดกันนี้เมื่อใช้ติดต่อกันจะใช้ได้เฉพาะอักขระภาษาอังกฤษและเครื่องหมายต่างๆ เท่านั้น (อักขระภาษาไทยจะใช้ไม่ได้)

เครื่องเทเล็กซ์จะมีชื่อย่อประจำเครื่อง เรียกโดยทั่วไปว่า ANSWERBACK CODE (AAB) สามารถใส่ข้อความได้ 20 ตัวอักษร ANSWERBACK CODE (AAB) จะประกอบด้วยเลขหมายประจำเครื่อง ชื่อย่อของผู้เช่า และอักษรย่อประเทศ เช่น 72012 BNKCHAT TH เป็น ANSWERBACK CODE ของธนาคารแห่งประเทศไทย เป็นต้น

ANSWERBACK CODE มีประโยชน์สำหรับตรวจสอบการเรียกติดต่อของผู้เช่า เช่น เครื่อง ก เรียกไปยังเครื่อง ข ANSWERBACK CODE ของเครื่อง ข จะตอบกลับมาที่เครื่อง ก โดยอัตโนมัติ และเครื่อง ก ก็สามารถบอกเครื่อง ข ให้ทราบว่าใครเป็นผู้เรียกโดยการกดแป้น WRU (ⓦ) ก็จะทำให้ ANSWERBACK CODE ของฝ่ายตรงข้ามตอบกลับมา วิธีการนี้ทำให้ฝ่ายรับ(ผู้ถูกเรียก)ไม่จำเป็นต้องมีคนอยู่ประจำเครื่อง ผู้เรียกก็สามารถมั่นใจได้ว่า เรียกได้ถูกต้องก่อนส่งข้อความไป

<u>ความเร็วในการรับ-ส่ง</u>

หน่วยความเร็วในการรับส่ง เรียกว่า BAUD (บอด) ปัจจุบันในระบบเทเล็กซ์ใช้ความเร็ว 50 BAUD เป็นมาตรฐานทั่วโลก

<u>ที่ความเร็ว 50 BAUD</u> จะใช้เวลาในการรับส่ง แต่ละตัวอักษร สำหรับเครื่องแต่ละชนิดดังนี้

- เครื่อง 5 ยูนิต
  1 ตัวอักษร ใช้เวลารับ-ส่ง 150 ms หรือ
  1 นาที รับ-ส่ง กันได้ 400 ตัวอักษร
- เครื่อง 6 ยูนิต
  1 ตัวอักษร ใช้เวลารับ-ส่ง 160 ms หรือ
  1 นาที รับ-ส่งกันได้ 375 ตัวอักษร

จะเห็นว่าในเวลา 1 นาที เครื่อง 5 ยูนิต รับ-ส่ง ได้เร็วกว่าเครื่อง 6 ยูนิต (เร็วกว่า 25 ตัวอักษร) ดังนั้นหากมีการ รับ-ส่ง ระหว่างเครื่องต่างชนิดกัน

จะมีผลดังนี้

เครื่อง 5 ยูนิตส่งไปยังเครื่อง 6 ยูนิต แต่ละตัวอักษรของเครื่อง 6 ยูนิต จะรับช้าไป 10 ms หากส่งจากเครื่อง 5 ยูนิต 1,000 ตัวอักษร เครื่อง 6 ยูนิต จะรับช้าไป 10,000 ms หรือ 6 วินาที

ปกติการรับส่งถึงแม้จะเป็นเครื่องชนิดเดียวกันเวลาจะต่างกันอยู่ 1 ตัวอักษร ดังนั้นในกรณีข้างต้นตัวอักษรตัวสุดท้ายที่เครื่อง 6 ยูนิตรับได้ จะห่างจากเวลาที่เครื่อง 5 ยูนิตส่งตัวสุดท้ายเท่ากับ 10 วินาที บวก 150 ms และหากเมื่อเครื่อง 5 ยูนิตส่งตัวสุดท้าย แล้วกด WRU ถาม กว่าเครื่อง 5 ยูนิตจะได้รับ ANSWERBACK CODE ตอบกลับมาก็ต้องรอเกือบ 11 นาที หากทางด้านต่างประเทศเรียกส่งข่าวมายังเครื่องเทเล็กซ์ 6 ยูนิต และใช้เวลาในการส่งติดต่อกันเกิน 40 วินาที จะมีผลทำให้ส่วนที่เก็บข้อมูลในช่วงเวลาที่แตกต่างกันเต็ม (Buffer Full) และการเรียกติดต่อจะถูกตัดออกแต่กรณีตามที่กล่าวมา จะมีโอกาสเกิดขึ้นน้อยมากจึงไม่มีปัญหา ในการใช้งานแต่อย่างใด

กรณีส่งจากเครื่อง 6 ยูนิตไปยังเครื่อง 5 ยูนิตเวลาจะต่างกัน 1 ตัวอักษร เท่านั้น ซึ่งในทางปฏิบัติจะไม่รู้สึกว่าช้า

<u>ประโยชน์ของบริการเทเล็กซ์</u>

- เป็นบริการโทรคมนาคมที่ผู้เช่าสามารถควบคุมได้ด้วยตัวตนเอง
- ใช้บริการได้ทั้งเรียกเข้าและเรียกออกตลอด 24 ชั่วโมง
- มีความแน่นอนในการ รับ-ส่ง สูง
- มีสำเนาเป็นหลักฐานตรงกันทั้งผู้รับและผู้ส่ง
- ใช้ร่วมกับบริการอื่น เช่น

การจองพูดโทรศัพท์ระหว่างประเทศ
การ รับ-ส่ง โทรเลขทางเทเล็กซ์
บริการ TELEBOX
บริการ TELEX WORLD LETTER
บริการฐานข้อมูลทางเทเล็กซ์ (TELEX DATABASE)
บริการประกาศหรือโฆษณาทางเครื่องเทเล็กซ์ (TELEX BULLETIN BOARD)
บริการข้อมูลการซื้อขายหลักทรัพย์

ตัวอย่าง การเรียกติดต่อต่างประเทศระหว่างหมายเลข 80810 CATELEX TH เรียก SANWA RS 99235 (BANGKOK-SINGAPORE)

ข้อความที่ปรากฏบนกระดาษหน้าเครื่อง	คำอธิบายวิธีทำ
BKK GA	1. กดปุ่มเรียกออก หลอดไฟการเรียกติดสว่าง พร้อมกับ BKK GA จากชุมสาย CTE ปรากฏที่หน้าเครื่องนาน 10 วินาที หากไม่เรียกออก เครื่องจะดับไปเอง
08799235 +	2. พิมพ์ AREA CODE หมายเลขปลายทาง และเครื่องหมาย + (บวก)
80810 CATELEX TH	3. ชุมสายถามชื่อผู้เรียก เพื่อเตียมบันทึกเวลา
JULY 01 92 1125 018547	4. ชุมสายส่ง เดือน,วัน,ปี,เวลาและหมายเลขอ้างอิง
SANWA RS 99235	5. เรียกได้หมายเลขปลายทาง(AAB) ชุมสายเริ่มคิดเวลา หากเรียกไม่ได้ชุมสายจะแจ้งด้วย SERVICE SIGNAL
80810 CATELEX TH	6. ผู้เรียกแสดงตัว กดปุ่ม HERE IS ให้ปลายทางทราบ
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOS	7. เริ่มส่งข้อความที่ต้องการส่ง จะใช้วิธีการพิมพ์หรือส่งจาก MEMORY ก็ได้
SANWA RS 99235	8. จบข้อความแล้ว กดปุ่ม WRU เพื่อแสดงว่าถึงผู้รับปลายทางแล้ว
MMMMM	9. เลิกการติดต่อโดยพิมพ์ M 5 ตัวหรือ ..... เพื่อขอเวลาจากชุมสาย
1126 000.9	10. ชุมสายส่งเวลาที่จบ และเวลาที่ใช้มาให้
THANK YOU	11. ชุมสายส่งข้อความ "ขอบคุณที่ใช้บริการ"

SERVICE SIGNAL

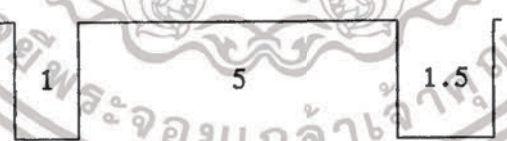
Service Signal คือสัญญาณที่ชุมสายตอบให้ผู้ใช้บริการทราบ เกี่ยวกับสาเหตุต่าง ๆ ที่ผู้ใช้บริการเรียกติดต่อปลายทางยังไม่ได้ มีดังนี้

1. NC (NO CIRCUIT) หมายถึง วงจรการติดต่อจากชุมสายถึงปลายทางไม่ว่าง
2. OCC (OCCUPY) หมายถึง เลขหมายปลายทางไม่ว่าง
3. DER (OUT OF ORDER) หมายถึง วงจรถูกตัดขาดทางสาย,หรืออีกฝ่ายหนึ่งปิดเครื่อง
4. NP (NO PARTY) หมายถึง ไม่มีการติดต่อการใช้บริการ
5. NA (NOT ADMITTED) หมายถึง การเรียกติดต่อที่ไม่ถูกต้อง
6. ABS (ABSENT SUBSCRIBER) หมายถึง เครื่องขัดข้องเนื่องจากไฟฟ้าดับ หรือเครื่องกำลังได้รับการแก้ไข
7. NCH (NUMBER CHANGER) หมายถึง หมายเลขได้เปลี่ยนใหม่แล้ว

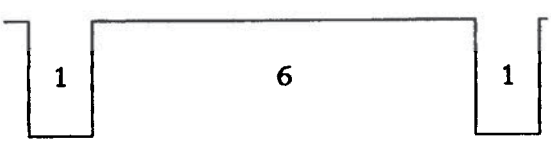
ระบบสัญญาณของเทเล็กซ์ (TELEX)

สัญญาณของระบบเทเล็กซ์ เป็นการส่งสัญญาณในลักษณะ Series จำนวนของ Bit ภายใน 1 ตัวอักษร จะมีลักษณะแตกต่างกันเนื่องจากเทเล็กซ์ ในประเทศไทยได้แบ่งเป็น 2 ชนิด คือ ชนิด 5 ยูนิต และชนิด 6 ยูนิต

- ระบบ 5 ยูนิต



- ระบบ 6 ยูนิต



รหัสบอโด (Baudot Code)

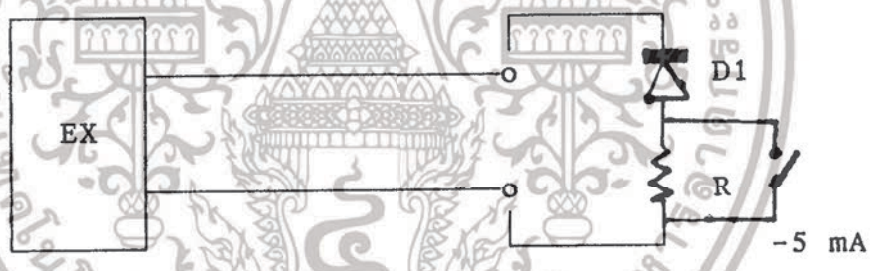
รหัส Baudot เป็นรหัสที่เป็นมาตรฐานของ CCITT No.2 และระบบเทเล็กซ์ทั่วโลกจะใช้กัน รหัสนี้จะประกอบด้วยรหัสขนาด 5 บิท จึงใช้แทนอักษรได้ 32 รูปแบบ ซึ่งจะเห็นได้ว่าไม่เพียงพอที่จะใช้ส่งตัวอักษรได้ทั้งหมด ดังนั้นจึงเพิ่มอักขระพิเศษขึ้นมา

อีก 2 ตัว เพื่อเป็นตัวบอกเครื่องรับว่ากำลังใช้กลุ่มตัวอักษรใด ตัวอักขระพิเศษที่กล่าวนี้คือ Letter Shift Character (LS) และ Figure Shift Character(FS) โดยที่ Letter Shift Character (LS) มีรหัสคือ 11111 ส่วน Figure Shift Character (FS) มีรหัสคือ 11011 และอาจแทน FS ด้วยรูปของลูกศรขึ้น ( ) ส่วน LS แทนด้วยรูปของลูกศรลง ( )

สำหรับรหัสบอโด ที่ใช้ในกิจการสื่อสารของประเทศไทย ได้มีการพัฒนาให้ให้เป็นรหัส 6 บิท เพื่อให้เข้ารหัสได้มากพอที่จะส่งข้อความได้ทั้งภาษาไทย และภาษาอังกฤษนั่นเอง

<u>ระบบ Current Loop ในเทเล็กซ์</u>

ในระบบเทเล็กซ์(อาศัยหลักการของ Current Loop) เมื่อเครื่องอยู่ในภาวะปกติ (Idle) ชุมสายจะจ่ายไฟใน Line 120 VDC (เป็นระบบ Two Wire) และกระแส Idle เท่ากับ -5 mA ตัวเครื่องเทเล็กซ์จะเปรียบเสมือนอุปกรณ์ 2 ตัวต่อกันอยู่ ดังรูป



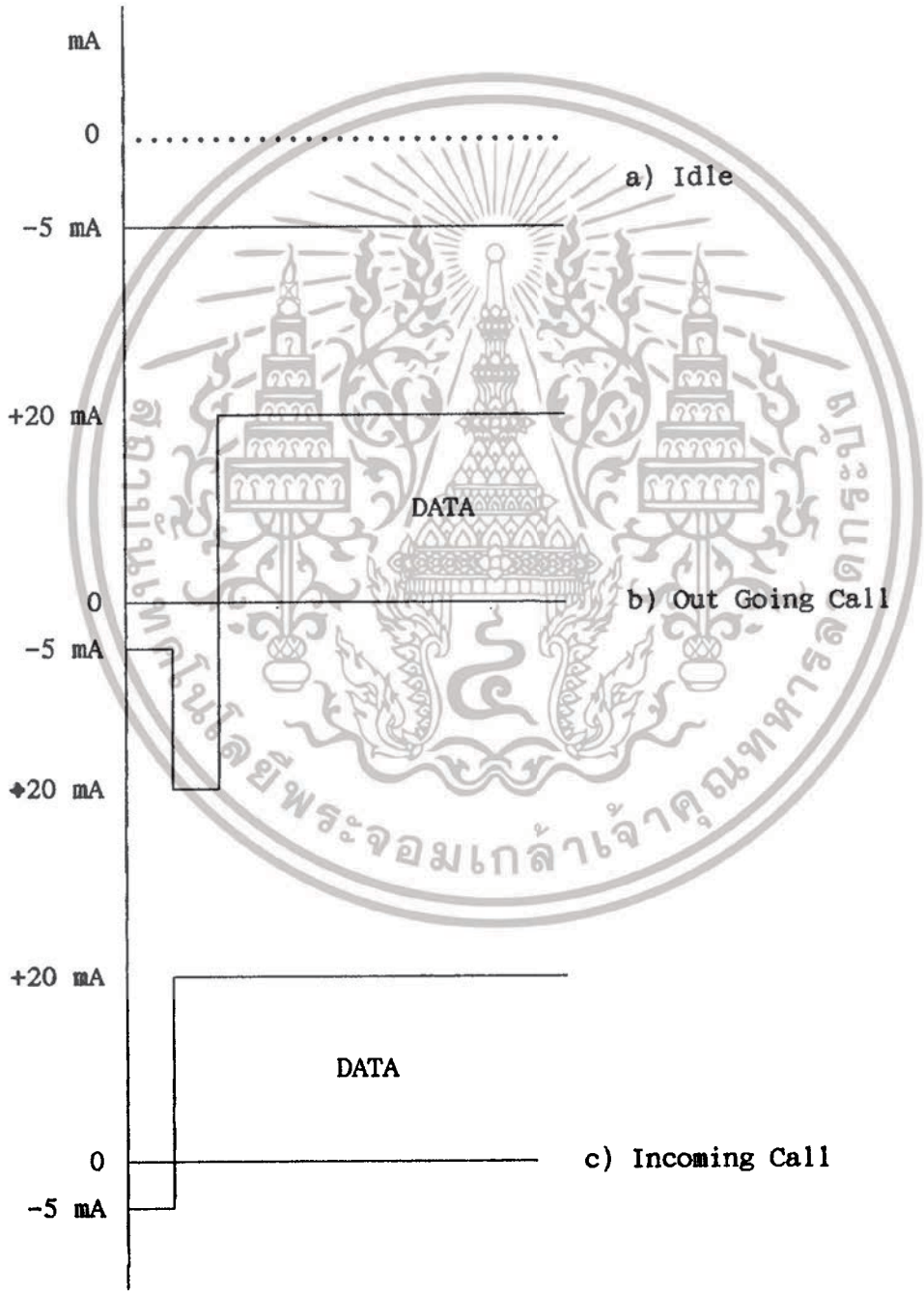
<u>รูปที่ 2</u> วงจรเสมือนของเครื่องเทเล็กซ์

ซึ่งกระแส -5 mA จะไหลผ่าน Diode D1 และ R ผ่านเข้ามายังชุมสาย เป็นลักษณะของ Current Loop ครบวงจร จากวงจรนี้จะเห็นว่าระบบถ้ามีการกลับ Line กันจะไม่ทำงานโดยดูอาการได้จากเครื่องเทเล็กซ์ จะส่งสัญญาณดังเป็นช่วง ๆ จาก Buzzer ภายในเครื่อง และจะไม่สามารถรับส่งกันได้ อาการนี้ที่ชุมสายก็จะรับรู้ได้เช่นเดียวกัน

เมื่อเกิดกรณีเรียกออก (Out Going Call) หมายถึง เรียกออกจากเครื่องเทเล็กซ์ผู้เช่า ไปยังชุมสายจากภาวะ Idle กระแสจะอยู่ที่ -5 mA เมื่อเครื่องถูกกดเรียกออก ค่าความต้านทานในวงจรของเครื่องส่วนที่ต่ออยู่กับ Line จะลดลง (ตามรูปที่ 2) กระแสที่ได้จะมีค่า -20 mA ซึ่งกระแสค่านี้จะเป็นตัวบอกให้ชุมสายได้รับ

รู้ว่าเครื่องเทเล็กซ์ต้องการเรียกออก ชุมสายก็จะส่งกระแส +20 mA มาให้(เป็นการกลับ Line) หลังจากนั้นก็เป็นการติดต่อส่งข้อมูลกันได้ ตามกระบวนการของเทเล็กซ์

ส่วนกรณีเรียกเข้า (Incoming Call) หมายถึงชุมสายหรือเครื่องผู้เช่ารายอื่นเรียกเข้ามายังเครื่องเทเล็กซ์ จากสภาวะ Idle เช่นกัน (-5 mA) ชุมสายจะเป็นตัวส่งสัญญาน +20 mA มาให้เป็นการบอกให้เครื่องรับรู้ว่ามีการเรียกเข้า หลังจากนั้นก็จะเป็นสัญญาน ตามกระบวนการเทเล็กซ์ต่อไป



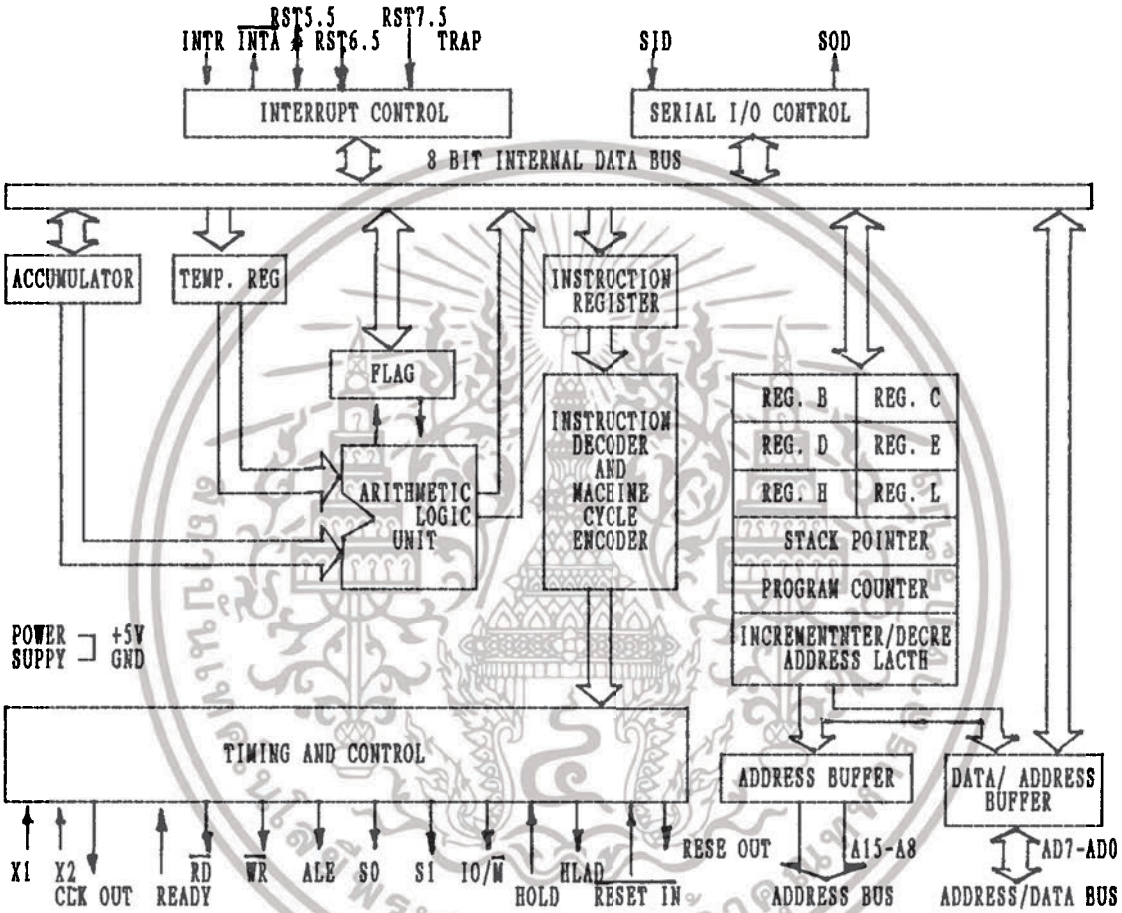
รูปที่ 3 เปรียบเทียบสัญญานของกระแสทั้ง 3 แบบ รูป a) สภาวะปกติ b) ภาวะเรียกออก c) สภาวะเรียกเข้า

บทที่ 2
CPU 8085
และ พอร์ตสื่อสารอนุกรม 8251

CPU 8085

ไมโครโปรเซสเซอร์ 8085 ผลิตโดยบริษัท INTEL มีขนาด 40 ขา มีการทำงานเป็นแบบ 8 บิท หมายความว่าส่วนที่ทำหน้าที่ในการคำนวณ (ARITHMETIC LOGIC UNIT : ALU) จะทำงานสูงสุดที่ละ 8 บิท

โครงสร้างภายในของ 8085



รูปที่ 4 แสดงโครงสร้างของ 8085

โครงสร้างภายใน 8085 จะมีส่วนที่สำคัญ 4 ส่วน คือ

1. กลุ่ม Registor
2. Arithemetic Logic Unit : ALU
3. Registor คำสั่ง (Instruction Registor) และวงจรควบคุม
4. บัฟเฟอร์ของบัสข้อมูล (Data Bus Buffer)

กลุ่ม Registor กลุ่มรีจีสเตอร์ภายใน 8085 ทำหน้าที่ในการเก็บข้อมูล 8 บิท หรือ 16 บิท ซึ่งคราวภายในกลุ่มรีจีสเตอร์แบ่งเป็น

- รีจีสเตอร์ชนิดใช้งานทั่วไป (8-Bit General Purpose Register) จำนวน 6 ชุด ได้แก่ B,C,D,E,H,L
- 16 บิทโปรแกรมเคาว์เตอร์ (PC)
- 16 บิทสแตคพอยเตอร์ (SP)

รีจีสเตอร์ชนิดใช้งานทั่วไป เป็นรีจีสเตอร์ขนาด 8 บิท สามารถใช้งานแบบคู่ได้ คือ BC,DE,HL การใช้งานแบบเป็นคู่นี้เป็นการใช้งานตามคำสั่งที่อยู่ในกลุ่มอ้างอิงหน่วยความจำ (Memory Reference) ในขณะนั้น B,D,H จะแสดงแอดเดรส 8 บิทบน และ C,E,L จะแสดงแอดเดรส 8 บิทล่าง

โปรแกรมเคาว์เตอร์(PC) เป็นรีจีสเตอร์ขนาด 16 บิท ที่แสดงแอดเดรสในหน่วยความจำบอกตำแหน่งของคำสั่งถัดไป PC จะเพิ่มหนึ่งให้ตัวเองอย่างอัตโนมัติทุกครั้งหลังการอ่านคำสั่ง สแตกพอยเตอร์ (SP) เป็นรีจีสเตอร์ขนาด 16 บิท ที่แสดงแอดเดรสในหน่วยความจำบอกตำแหน่งยอดของสแตก 8085 จะใช้เนื้อที่ในหน่วยความจำส่วนหนึ่งเป็นสแตกและใช้สแตกพอยเตอร์ช่วยชี้บอกตำแหน่งของสแตก สแตกพอยเตอร์จะถูกลดลงหนึ่งโดยอัตโนมัติเมื่อข้อมูลถูก PUSH เข้าไปในสแตก และจะถูกเพิ่มโดยอัตโนมัติเมื่อข้อมูลถูก POP ออกจากสแตก

<u>ARITHMETIC LOGIC UNIT : ALU</u> ประกอบด้วยส่วนที่สำคัญ คือ

- ALU
- 8 บิทแอคคิวมูเลเตอร์ (8 Bit Accumulator : ACC)
- 8 บิทแอคคิวมูเลเตอร์ชั่วคราว (8 Bit Temporary ACC : ACT)
- 8 บิทรีจีสเตอร์ชั่วคราว (8 Bit Temporary Registor : TMP)
- 5 บิทแฟลกรีจีสเตอร์

ALU เป็นแหล่งที่ทำการคำนวณทางคณิตศาสตร์ และทางลอจิก (Logic) และทำการเลื่อนหรือหมุนข้อมูล(Shift or Rotate) ข้อมูลจะถูกส่งเข้า ALU ได้ 2 ทาง จาก TMP และ ACT เมื่อทำการคำนวณเรียบร้อยแล้วจะส่งผลออกที่แอคคิวมูเลเตอร์หรือที่บัสข้อมูลภายในเพื่อส่งถ่ายออกไปเก็บที่กลุ่มรีจีสเตอร์อีกที่หนึ่ง TMP เป็นรีจีสเตอร์ที่ทำหน้าที่รับข้อมูลเข้าจากบัสข้อมูลภายในและป้อนเข้า ALU ดังนั้นจึงเป็นรีจีสเตอร์ซึ่งทำงานประสานระหว่างกลุ่มรีจีสเตอร์กับ ALU ส่วน ACT จะเป็นรีจีสเตอร์ซึ่งอยู่ระหว่าง ACC กับ ALU โดยจะรับข้อมูลจาก ACC ส่งเข้า ALU ACC สามารถส่งข้อมูลเข้าออกบัสข้อมูลภายใน และรับข้อมูลเข้า ALU ได้โดยตรง

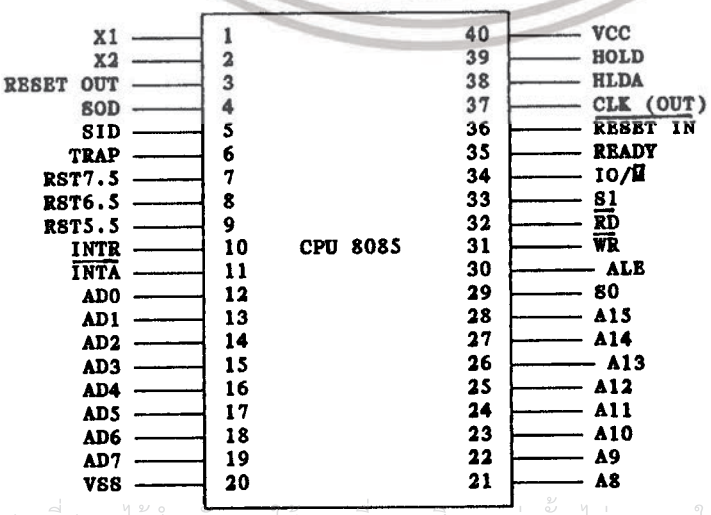
5 บิทแฟลกรีจีสเตอร์(Flag Registor) เป็นรีจีสเตอร์ที่แสดงภาวะการทำงานของ ALU ประกอบด้วย แฟลกศูนย์(Zero Flag) เป็นแฟลกที่แสดงผลของการคำนวณ แอคคิวมูเลเตอร์มีค่าเป็นศูนย์("0" ทุกบิท) หรือไม่ ตามปกติเท่ากับ "1" เมื่อแอคคิวมูเลเตอร์เท่ากับศูนย์ และเป็น "0" เมื่อแอคคิวมูเลเตอร์ไม่เท่ากับศูนย์ แฟลกศูนย์สามารถนำมาใช้ประโยชน์ได้มากในกรณีที่ต้องการตรวจสอบบิทใดบิทหนึ่งในแอคคิวมูเลเตอร์

ว่าเป็นศูนย์หรือหนึ่ง แฟลกตัวทด(Carry Flag) เป็นแฟลกที่ใช้ในการทดเลขที่เกิดจากการบวกในแอคคิวมูเลเตอร์ การใช้ประโยชน์อีกอย่างหนึ่งของแฟลกตัวทดคือ ใช้ในการเลื่อนและหมุนข้อมูลในแอคคิวมูเลเตอร์ แฟลกเครื่องหมาย(Sign Flag) เพื่อแสดงเครื่องหมายของตัวเลข โดยจะมีค่าเป็น "1" เมื่อตัวเลขในแอคคิวมูเลเตอร์มีค่าเป็นลบและจะมีค่า "0" เมื่อตัวเลขมีค่าเป็นบวก แฟลกพาริตี้(Parity Flag) จะเป็นหนึ่งขึ้นกับข้อมูลในแอคคิวมูเลเตอร์ หลังจากทำคำสั่งพวกกลุ่มคำสั่งคณิตศาสตร์และกลุ่มคำสั่งตรรกแล้วถ้าบิทที่เป็นหนึ่งในแอคคิวมูเลเตอร์ทั้งหมดนับได้เป็นเลขคู่จะทำให้แฟลกพาริตี้เป็นหนึ่ง ถ้านับได้เป็นเลขคี่จะทำให้แฟลกพาริตี้เป็นศูนย์ แฟลกตัวทดช่วย (Auxiliary Carry Flag) ถ้าเกิดการทดของข้อมูลที่บิท D_3 ไปยังบิท D_4 จะทำให้แฟลกตัวทดเป็นหนึ่ง ถ้าไม่มีการทดเกิดขึ้นแฟลกตัวทดจะเป็นศูนย์

<u>รีจีสเตอร์คำสั่งและวงจรควบคุม</u> (IR) จะทำหน้าที่รับข้อมูลที่เป็นคำสั่งจากหน่วยความจำผ่านเข้ามาทาง Buffer และบัสข้อมูลภายใน คำสั่งที่เก็บใน IR นี้จะถูกถอดรหัสโดยวงจรถอดรหัสคำสั่ง เพื่อให้วงจรควบคุมสามารถผลิตสัญญาณควบคุมที่เหมาะสมเพื่อควบคุมการทำงานภายใน 8085 ให้เป็นไปตามคำสั่งนั้น และสร้างสัญญาณควบคุมเพื่อส่งออกทางบัสควบคุม เพื่อควบคุมวงจรภายนอกอีกที่หนึ่ง

<u>บัฟเฟอร์ของบัสข้อมูล</u>(Data Bus Buffer) เป็นบัฟเฟอร์ 8 บิทคั่นกลางระหว่างบัสข้อมูลภายในกับบัสข้อมูลภายนอก ซีพียูมีไว้สำหรับควบคุมการส่งข้อมูลเข้าออก ในช่วงเวลาการส่งข้อมูลเข้าออกจากซีพียู ข้อมูลภายในจะถูกแลทช์ โดย Data Bus Latch เพื่อส่งต่อให้เอาท์พุทบัฟเฟอร์ส่งออกภายนอกอีกที เอาท์พุทบัฟเฟอร์จะหยุดการทำงานในช่วงเวลาที่รับข้อมูลเข้าจากภายนอก บัฟเฟอร์ของข้อมูลนี้จะถูกควบคุมการทำงานโดยวงจรควบคุม

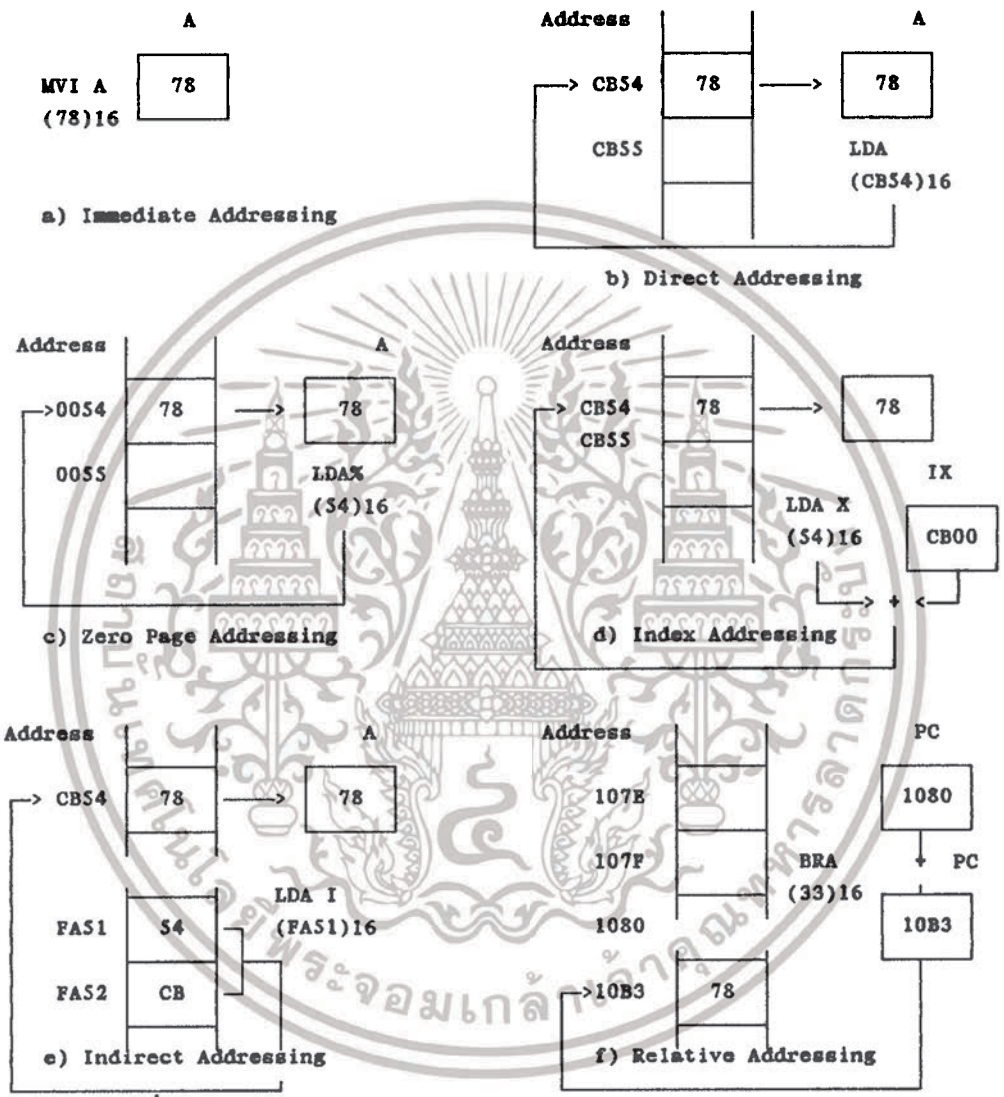
<u>รายละเอียดของขาสัญญาณต่าง ๆ</u>



<u>รูปที่ 5</u> แสดงขาต่าง ๆ ของ 8085

การอ้างแอดเดรสของ 8085

การนำข้อมูลจากหน่วยความจำเข้ามาในไมโครโปรเซสเซอร์ ต้องมีการชี้บอกตำแหน่งหรือแอดเดรสที่เก็บข้อมูลในหน่วยความจำเสมอ แอดเดรสที่จะส่งออกไปแจ้งที่หน่วยความจำ สามารถส่งออกทาง PC ,SP และรีจีสเตอร์ข้อมูลที่เป็นคู่ วิธีการอ้างแอดเดรสที่ใช้ในไมโครโปรเซสเซอร์มีหลายวิธีคือ



รูปที่ 6 แสดงการอ้าง Address ในแบบต่างๆ ของ 8085

1. อิมมิเดียทแอดเดรส (Immediate Addressing) เป็นวิธีนำข้อมูลที่เป็นตัวเลขเข้ารีจีสเตอร์โดยตรง ไม่จำเป็นต้องอ้างแอดเดรสในหน่วยความจำ วิธีการนำข้อมูลเข้าด้วยวิธีนี้สะดวกในการเซทค่าคงที่ในรีจีสเตอร์

2. แอดเดรสโดยตรง(Direct Addressing) เป็นวิธีการอ้างแอดเดรสแบบปกติที่ใช้งานทั่วไป ข้อมูลแอดเดรสขนาด 2 ไบท์จะถูกส่งไปหน่วยความจำโดยตรง การอ้างแอดเดรสแบบนี้สามารถทำได้โดยใช้คำสั่งเดียว

3. แอดเดรสเพจศูนย์(Zeropage Addressing) เป็นวิธีอ้างแอดเดรสในเพจศูนย์ ซึ่งข้อมูลแอดเดรส 8 บิทแรกจะเป็นศูนย์หมด การอ้างแอดเดรสชนิดนี้ใช้จำนวนไบท์ของคำสั่งน้อยกว่าแบบแอดเดรสโดยตรง

4. อินเดกส์แอดเดรส(Index Addressing) เป็นการอ้างแอดเดรสโดยอินเดกส์รีจีสเตอร์ แอดเดรสที่ส่งไปหน่วยความจำจะเท่ากับผลรวมของข้อมูลในอินเดกส์รีจีสเตอร์กับค่าดิสเพลสเมนต์(Displacement)ซึ่งเป็นค่าตัวเลข

5. แอดเดรสทางอ้อม(Indirect Addressing) คือแอดเดรสที่เก็บข้อมูลที่ต้องการ จะถูกเก็บในแอดเดรสที่อ้างอิง

6. แอดเดรสสัมพันธ์(Relative Addressing) เป็นการอ้างแอดเดรสที่นิยมใช้กับ PC เพื่อให้ไมโครโปรเซสเซอร์กระโดดไปทำงานตามแอดเดรสที่ต้องการได้ ค่าใน PC จะถูกบวกเข้ากับค่าดิสเพลสเมนต์ได้เป็นแอดเดรสใหม่

ชุดคำสั่งของ 8085 แบ่งออกเป็น 5 กลุ่มดังนี้

1. กลุ่มคำสั่งโอนย้ายข้อมูล(Data Transfer Group)เป็นคำสั่งที่ใช้ในการเคลื่อนย้ายข้อมูลเข้าออกระหว่างรีจีสเตอร์กับหน่วยความจำหรือระหว่างรีจีสเตอร์กับรีจีสเตอร์

2. กลุ่มคำสั่งเลขคณิต(Arithematic Group) เป็นคำสั่งการคำนวณข้อมูลที่อยู่ในรีจีสเตอร์หรือข้อมูลที่อยู่ในแอดดิวมูเลเตอร์กับข้อมูลในหน่วยความจำ

3. กลุ่มคำสั่งลอจิก(Logical Group) เป็นคำสั่งการทำงานแบบลอจิกด้วยข้อมูลที่อยู่ในแอดดิวมูเลเตอร์กับข้อมูลที่อยู่ในรีจีสเตอร์หรือหน่วยความจำ

4. กลุ่มคำสั่งย้ายการทำงาน(Branch Group) เป็นคำสั่งที่ทำให้เกิดการเปลี่ยนแปลงการทำงานตามลำดับของคำสั่งในโปรแกรม

5. กลุ่มคำสั่งเกี่ยวกับสแตก อินพุทเอาท์พุท และการควบคุม(Stack I/O And Machine Control Group) เป็นคำสั่งเกี่ยวกับ อินพุทและเอาท์พุท การทำงานของสแตก และควบคุมสภาพของแฟลกภายใน

การทำงาน-การติดต่อหน่วยความจำและหน่วยอินพุทเอาท์พุทของซีพียู

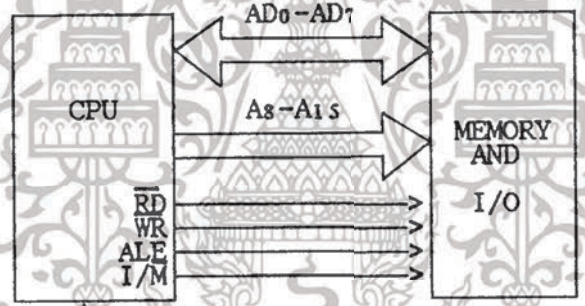
การทำงานของ CPU8085 ตามคำสั่งเริ่มต้นจากการอ่านคำสั่ง (Fetch) จากหน่วยความจำ แล้วจึงทำงานตามคำสั่งนั้น (Execute) ช่วงเวลาในการอ่านและทำตามคำสั่งหนึ่งคำสั่งใดเรียกว่าอินสตักชันไซเคิล (Instruction Cycle) ช่วงเวลาในอินสตักชันไซเคิลของแต่ละคำสั่งมีความยาวไม่เท่ากัน แล้วแต่คำสั่งนั้นจะให้ทำงานภายในซีพียูเอง หรือต้องมีการอ่านหรือเขียนจากหน่วยความจำ หรือ I/O

ในแต่ละอินสตักชันไซเคิล จะประกอบด้วยแมชีนไซเคิลจำนวนตั้งแต่ 1-5 ไซเคิล อินสตักชันไซเคิลเป็นหน่วยเวลาพื้นฐานของการทำงานของซีพียู คำสั่งหนึ่งจะมีแมชีนไซเคิลเท่ากับจำนวนที่ต้องมีการอ่านหรือเขียนหน่วยความจำ หรือ I/O ทุกคำสั่งจะต้องมีแมชีนไซเคิลพื้นฐานอยู่หนึ่งไซเคิลที่เรียกว่า Fecth Cycle (M1) ซึ่งคือการอ่านคำสั่งของมันเองนั่นเอง บางครั้งจะมีแมชีนไซเคิลเดียวคือ M1 แต่บางคำสั่งที่ต้องการการเขียนอ่านหน่วยความจำ หรือ I/O เพิ่มเติมต้องมีแมชีนไซเคิลจำนวน 2,3,4 หรือ 5 ไซเคิล

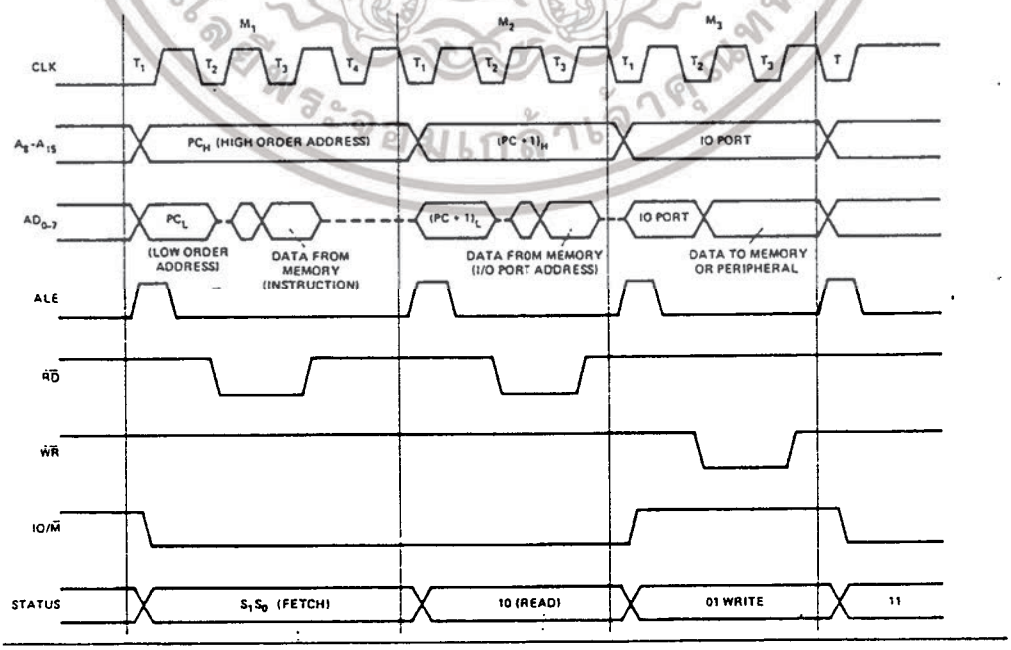
การทำงานของซีพียูนั้น ช่วงเวลาการทำงานที่ต้องการใน 1 แมชีนไซเคิล สามารถแบ่งง่าย ๆเป็น 6 ชนิด คือ

1. อ่านหน่วยความจำ
2. เขียนหน่วยความจำ
3. อ่านหน่วย I/O
4. เขียนหน่วย I/O
5. อ่านข้อมูลการอินเตอร์รัพท์
6. การคำนวณและการทำงานภายใน

การติดต่อกับหน่วยความจำและหน่วยอินพุทเอาท์พุท (I/O) ซีพียูจะทำการส่งสัญญาณที่สำคัญออกไป คือ สัญญาณ $\overline{RD}$,$\overline{WR}$,I/$\overline{M}$ และ ALE โดยสัญญาณ I/$\overline{M}$ จะเป็นตัวกำหนดว่าจะทำการติดต่อกับหน่วยความจำหรือหน่วยอินพุทเอาท์พุท เมื่อซีพียูทำการติดต่อกับหน่วยความจำก็จะส่งสัญญาณ I/$\overline{M}$ เท่ากับ "0" ออกไป และเมื่อซีพียูทำการติดต่อกับหน่วยอินพุทเอาท์พุท สัญญาณ I/$\overline{M}$ จะมีค่าเท่ากับ "1" ส่วนสัญญาณ ALE จะใช้ในการ Latch ข้อมูล 8 บิทล่าง (AD_0 ถึง AD_7) เพราะว่าข้อมูล 8 บิทล่างจะส่งค่าตำแหน่งหน่วยความจำออกมาเพียงชั่วขณะหนึ่งเท่านั้น หลังจากนั้นจะใช้ในการส่งข้อมูล



รูปที่ 7 แสดงการติดต่อ Memory และ I/O



8085A Basic System Timing

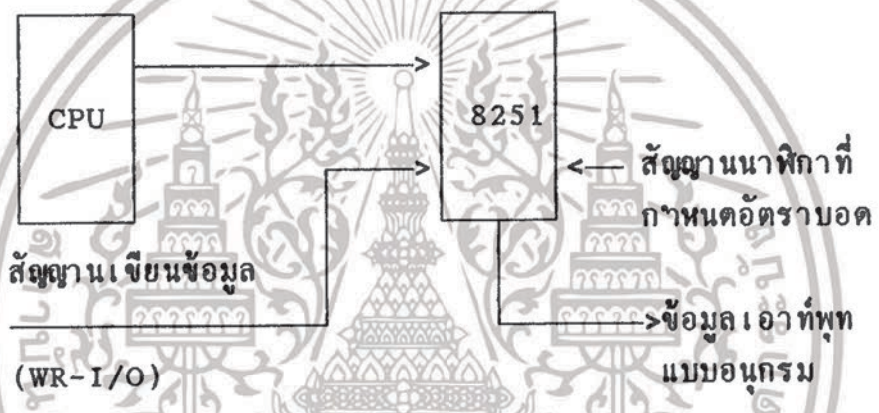
รูปที่ 8 แสดง Timing Diagram การติดต่อ

พอร์ตสื่อสารอนุกรม 8251

ในการสื่อสารข้อมูลแบบอนุกรมนั้น ต้องแปลงข้อมูลแบบขนานมาเป็นแบบอนุกรม โดยปกติจะใช้ไมโครโปรเซสเซอร์ส่งข้อมูลมายังรีจีสเตอร์แล้วแปลงข้อมูลเป็นอนุกรม ขบวนการแปลงข้อมูลเป็นดังนี้

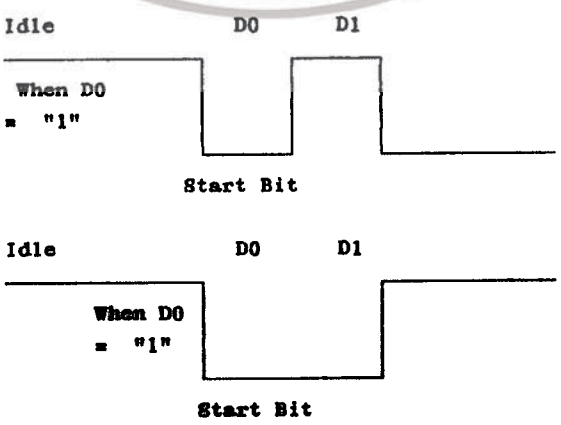
1. ทำการเก็บข้อมูลขนาด 8 บิท เข้ามาเก็บในชิฟรีจีสเตอร์
2. เลื่อนข้อมูลจากชิฟรีจีสเตอร์ ไปที่ละบิทตามการกำหนดด้วยช่วงเวลาตามการกำหนดของอัตราบอด

ลักษณะการทำงานจะเป็นไปตามรูปที่ 9 โดยให้ CPU ส่งข้อมูลเอาท์พุทออกมาเป็นแบบขนาน 8 บิท มายังชิฟรีจีสเตอร์ แล้วให้มีการเลื่อนบิทออกไป



รูปที่ 9 Block Diagram ของการส่งข้อมูลแบบอนุกรม

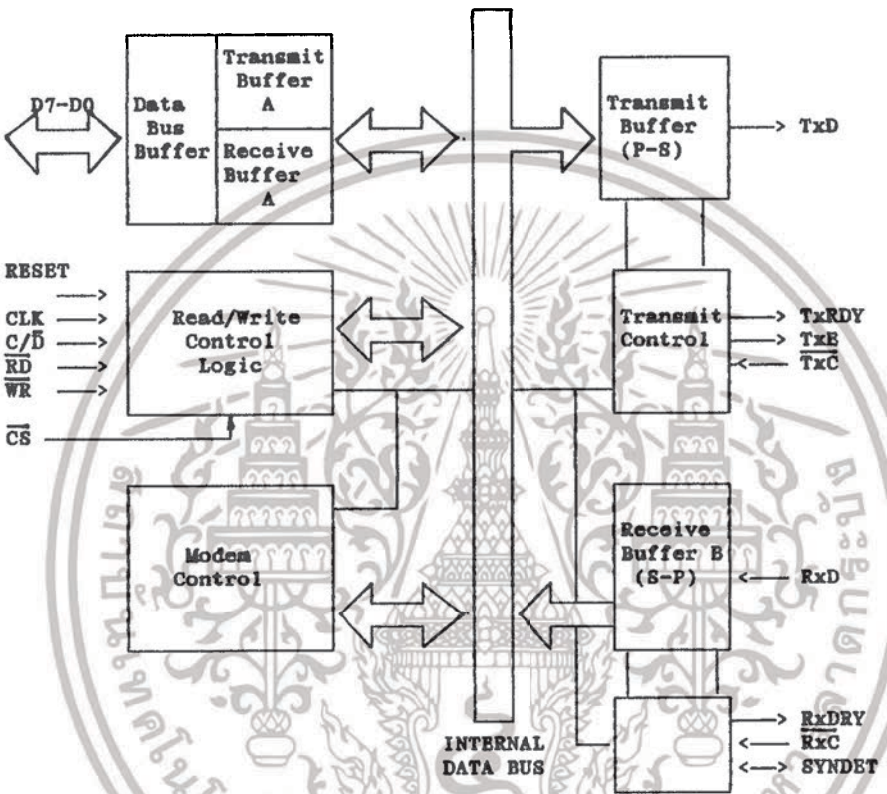
การรับส่งข้อมูลแบบอะซิงโครนัสจะต้องมีการบอกจุดเริ่มต้น และสิ้นสุดของเฟรมข้อมูล โดยปกติจะให้สภาวะ Idle เหมือนเช่นบิทสตอป ดังนั้นส่วนของบิทสตาร์ตจะตรงข้ามกับ Idle โดยทั่วไปของการส่งข้อมูลเราจะใช้ 1 บิท เป็นตัวบอกสตาร์ต และเราใช้ "0" เป็นตัวบอกบิตสตาร์ตดังรูปที่ 10



รูปที่ 10 แสดงบิตสตาร์ต

โครงสร้างของ 8251

ชิพ 8251 เป็นชิพที่ทำหน้าที่เป็นตัวรับและส่งข้อมูลระหว่างไมโครโปรเซสเซอร์และอุปกรณ์อินพุท-เอาท์พุทแบบอนุกรม ชิพนี้ทำงานได้ทั้งโหมดอะซิงโครนัสและซิงโครนัส แต่ที่ประยุกต์ใช้กันมากจะเป็นแบบอะซิงโครนัส ตัว 8251 เป็นไอซีขนาด 28 ขา โดยมีการจัดขาและโครงสร้างภายในดังรูปที่ 11

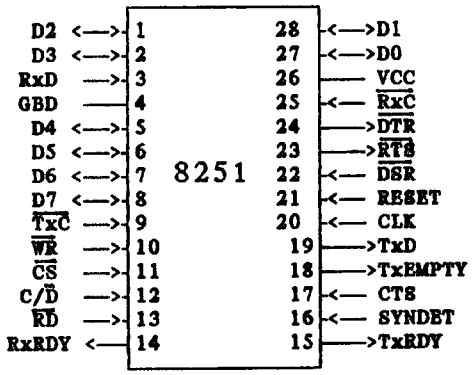


รูปที่ 11 แสดงโครงสร้างของ 8251

สัญญาณตามรูปที่ 11 มีรายละเอียดเพิ่มเติมดังนี้

4

1. Data Bus Buffer เป็นส่วนที่จะต้องเชื่อมโยงกับบัสข้อมูลของไมโครโปรเซสเซอร์ ข้อมูลที่ส่งมาจาก CPU จะถูกเก็บไว้ใน Transmit Buffer A ซึ่งจะถูกส่งต่อไปยังวงจร Buffer B เพื่อจะเลื่อนออกไปแบบอนุกรมทาง TxD ในทางตรงข้ามข้อมูลที่ 8251 รับมาแบบอนุกรมจะถูกเปลี่ยนเป็นแบบขนานโดย Receive Buffer B ก่อน แล้วส่งต่อมาเก็บไว้ใน Receive Buffer A เพื่อส่งต่อให้ CPU ต่อไป CPU ยังสามารถบังคับให้ 8251 ทำหน้าที่ต่าง ๆ ได้ และข้อมูลหรือสถานะของ 8251 จะได้รับการอ่านโดย CPU ผ่านเข้าทางบัสข้อมูลได้
2. Read/Write Control Logic ทำหน้าที่ควบคุมการอ่านหรือเขียนข้อมูลมายัง 8251
3. Modem Control ทำหน้าที่ควบคุม 8251 เมื่อ 8251 ต้องทำหน้าที่ติดต่อกับ Modem



รูปที่ 12 แสดงขาสัญญาณต่าง ๆ ของ 8251

โครงสร้างของสัญญาณต่าง ๆ สรุปได้ดังตารางที่ A

ชื่อสัญญาน	หน้าที่โดยย่อ	ชนิดของสัญญาน
D_0 -D_7	บัสข้อมูล	สองทิศทาง
RESET	สัญญาน Reset	INPUT
CLK	สัญญานนาฬิกา	INPUT
C/$\overline{D}$	สัญญานเลือก Control/Data	INPUT
$\overline{RD}$	สัญญานแสดงการอ่าน	INPUT
$\overline{WR}$	สัญญานแสดงการเขียน	INPUT
$\overline{CS}$	สัญญานเลือกชิพ 8251	INPUT
$\overline{DSR}$	Data Set Ready	INPUT
$\overline{DTR}$	Data Terminal Ready	OUTPUT
$\overline{CTS}$	Clear To Send	OUTPUT
$\overline{RTS}$	Request To Send	OUTPUT
TxD	ข้อมูลเอาท์พุทแบบอนุกรม	OUTPUT
TxRDY	พร้อมจะรับข้อมูลไปส่ง	OUTPUT
TxEMPTY	Buffer ส่งว่าง	OUTPUT
$\overline{TxC}$	สัญญานนาฬิกากำหนดการส่ง	INPUT
RxD	ข้อมูลอินพุทแบบอนุกรม	INPUT
TxRDY	ข้อมูลพร้อมที่จะส่งไปยังบัสข้อมูล	OUTPUT
$\overline{RxC}$	สัญญานนาฬิกากำหนดการรับ	INPUT
SYNDET/BD	Synchronous Detect/Break Det	สองทิศทาง
VCC,GND	ไฟเลี้ยงและ Ground	-

ตารางที่ 1 สรุปสัญญาณต่าง ๆ ของ 8251

4. Transmit Buffer B ทำหน้าที่รับข้อมูลจาก Buffer A แล้วเปลี่ยนเป็นแบบอนุกรมส่งไปยังอุปกรณ์ภายนอกทางขา TxD
5. Transmit Control ทำหน้าที่ควบคุมการส่งข้อมูลแบบอนุกรม
6. Receive Buffer B เป็นที่เก็บข้อมูลที่มาจากอุปกรณ์ภายนอกแบบอนุกรมทางขา RxD แล้วเปลี่ยนเป็นแบบขนานให้ Receive Buffer A
7. Receive Control ทำหน้าที่การรับข้อมูลแบบอนุกรม
8. Internal Data Bus เป็นเส้นทางติดต่อภายใน

<u>สัญญาณติดต่อกับ CPU</u>

1. D_0-D_7 เป็นสัญญาณที่จะต่อกับบัสของไมโครโปรเซสเซอร์
2. RESET สัญญาณ "1" ที่ขานี้จะบังคับให้ 8251 อยู่ในสภาวะรีเซตหรือวงจรภายในจะอยู่ในสภาพเหมือนเริ่มต้นใหม่หมด คำสั่งเก่า ๆ จาก CPU จะถูกลบไปหมด
3. CLK สัญญาณนาฬิกาสำหรับวงจรต่าง ๆ ภายใน 8251
4. $\overline{WR}$ แอคทีฟที่ลอจิก "0" เป็นการเขียนข้อมูลมายังพอร์ต
5. $\overline{RD}$ แอคทีฟที่ลอจิก "0" เมื่อ CPU ต้องการอ่านข้อมูล
6. C/$\overline{D}$ ถ้าเป็น "1" คือ สัญญาณควบคุม เป็นการเขียนอ่านรหัสควบคุมแต่ถ้าเป็น "0" หมายถึงการเขียนอ่านข้อมูล
7. $\overline{CS}$ เป็นสัญญาณเลือกชิพ 8251 แอคทีฟที่ลอจิก "0"

การใช้งานจะใช้ร่วมกันระหว่างสัญญาณ C/$\overline{D}$, $\overline{RD}$, $\overline{WR}$ และ $\overline{CS}$ ดังตาราง 2

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	ความหมาย
0	0	1	0	ข้อมูลจาก 8251-บัสข้อมูล
0	1	0	0	บัสข้อมูล-8251
1	0	1	0	ข้อมูลบอกสถานะ-บัสข้อมูล
1	1	0	0	บัสข้อมูล-รีจีสเตอร์ควบคุม

<u>ตารางที่ 2</u> การทำงานร่วมกันของสัญญาณควบคุม

สัญญาณติดต่อกับโมเด็ม

1. $\overline{\text{DSR}}$ (Data Set Ready) เป็นสัญญาณอินพุทเข้า 8251 สถานะของ $\overline{\text{DSR}}$ สามารถตรวจสอบได้โดยโปรแกรมของ CPU จึงสามารถใช้สัญญาน $\overline{\text{DSR}}$ เป็นสัญญาณพอร์ตอินพุทขนาด 1 บิทได้ แต่ในระบบโมเด็มจะใช้เป็นสัญญาณตอบสนองความพร้อมจากโมเด็ม
2. $\overline{\text{DTR}}$ (Data Terminal Ready) เป็นสัญญาณเอาท์พุทจาก 8251 สถานะของ $\overline{\text{DTR}}$ สามารถควบคุมโดยโปรแกรมจาก CPU ได้จึงใช้เป็นพอร์ตเอาท์พุทขนาด 1 บิทได้ แต่ในงานโมเด็มจะใช้เป็นสัญญาณแสดงบอกโมเด็มถึงความพร้อมของ 8251
3. $\overline{\text{RTS}}$ (Request To Send) เป็นสัญญาณเอาท์พุทจาก 8251 สถานะของ $\overline{\text{RTS}}$ สามารถควบคุมได้โดยโปรแกรมจาก CPU จึงใช้เป็นสัญญาณพอร์ตเอาท์พุทขนาด 1 บิทได้ แต่ถ้าใช้งานกับโมเด็ม จะเป็นสัญญาณบอกไปยังโมเด็มเพื่อเตรียมส่งข้อมูล
4. $\overline{\text{CTS}}$ (Clear To Send) เมื่อเป็น "0" ตัว 8251 จึงจะส่งข้อมูลออกไปได้

สัญญาณติดต่อกับภาครับส่ง

1. TxRDY (Transmitter Ready) มีความสัมพันธ์กับ Transmit Buffer A และ B ดังนี้ TxRDY จะเป็น "1" เมื่อข้อมูลใน Transmit Buffer A ถูกส่งต่อไปยัง Transmit Buffer B แล้ว นั่นคือ Transmit Buffer A ว่างพร้อมที่จะรับใหม่จาก CPU ได้ TxRDY จะเป็น "0" เมื่อ Transmit Buffer A รับข้อมูลมาจาก CPU แต่มีข้อต้องระวังบิทข้อมูล TxRDY ในไบต์แสดงสถานะจะไปออกยังสัญญาน TxRDY ขา 15 ก็ต่อเมื่อ 8251 ได้รับอีนาเบิลให้ส่งข้อมูลได้หรือ TxEN ในไบต์แสดงสถานะเป็น "1" และ $\overline{\text{CTS}}$ เป็น "0"
2. TxEMPTY จะเป็น "1" เมื่อข้อมูลใน Transmit Buffer B ถูกเลื่อนออกไปทางขา TxD หมดแล้วซึ่งก็คือ Transmit Buffer B ว่าง นั่นคือ TxEMPTY จะเป็น "0" เมื่อ Transmit Buffer B รับข้อมูลมาจาก Transmit Buffer A
3. $\overline{\text{TxC}}$ (Transmit Clock) เป็นสัญญาณนาฬิกาฐานเวลาของข้อมูลที่ส่งออกไป ในระบบซิงโครนัส ความถี่ของ $\overline{\text{TxC}}$ ก็คือ อัตราส่ง(Baud Rate) ของการส่ง(1X) แต่ในระบบอะซิงโครนัส ความถี่ของ TxC จะสูงเป็น 1,16 หรือ 64 เท่าของอัตราส่งจริง(baud Rate) ก็ได้ ซึ่งเรียกได้โดยโปรแกรม

ตัวอย่างเช่น

อัตราบอด เป็น 100 Baud

- ความถี่ $\overline{TxC}$ จะต้องเป็น 100 Hz ถ้าเลือก (1X)
- ความถี่ $\overline{TxC}$ จะต้องเป็น 1.6 KHz ถ้าเลือก (16X)
- ความถี่ $\overline{TxC}$ จะต้องเป็น 6.4 KHz ถ้าเลือก (64X)

4. RxRDY (Reveiver Ready) เป็นสัญญานเอาท์พุทไปยัง CPU แสดงบอกว่ารับข้อมูลจากภายนอกเข้ามาอยู่ใน Receive Buffer A แล้วพร้อมที่จะให้ CPU มาอ่านไปได้
5. $\overline{RxC}$ (Receiver Clock) เป็นสัญญานนาฬิกาฐานเวลาของการรับข้อมูลจากภายนอก มีกฎเกณฑ์เช่นเดียวกับ $\overline{TxC}$

<u>การโปรแกรม 8251</u>

การโปรแกรม 8251 คือ การสั่งให้ 8251 ทำงานตามรูปแบบที่ระบบเราต้องการก่อนที่จะเริ่มการรับการส่งทั้งหมด CPU ต้องออกคำสั่งมายัง 8251 โดยมีขั้นตอนและประเภทของคำสั่ง คือ

1. สัญญาน RESET เป็นการเริ่มต้นก่อนใด ๆ
2. ทำคำสั่งเลือกขบวนการทำงาน (Mode Select)

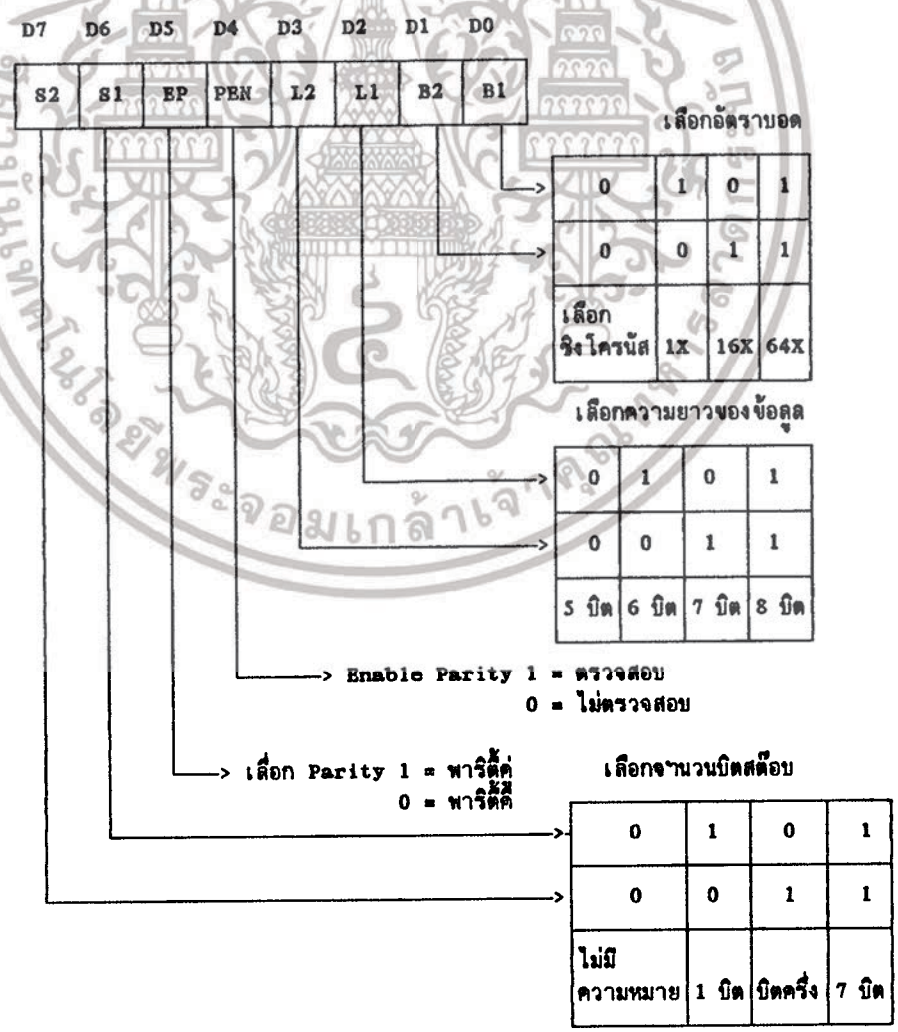
ขั้นที่	ขาสัญญาน	รหัสคำสั่งหรือข้อมูลที่ส่งมาให้ 8251
1	C/$\overline{D}$ = 1	Mode Instruction : เลือกโหมด
2	C/$\overline{D}$ = 1	SYNC Charactor 1 ทำถ้าเลือกโหมดซิงโครนัส
3	C/$\overline{D}$ = 1	SYNC Charactor 2 ทำถ้าเลือกโหมดซิงโครนัส
4	C/$\overline{D}$ = 1	Command Instruction : รหัสควบคุม
5	C/$\overline{D}$ = 0	Data...
6	C/$\overline{D}$ = 1	Command Instruction ทำเมื่อต้องการเปลี่ยนรหัส
7	C/$\overline{D}$ = 0	Data...
8	C/$\overline{D}$ = 1	Command Instruction ทำเมื่อต้องการเปลี่ยนรหัส

<u>ตาราง 3</u> ลำดับขั้นตอนที่ CPU ติดต่อกับ 8251

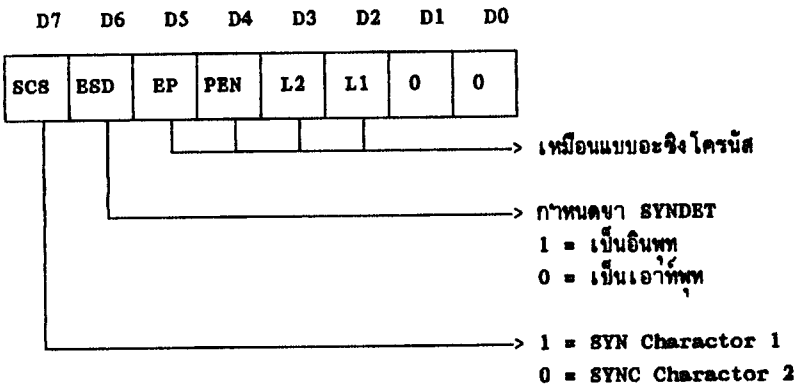
3. ทาคาสั่งควบคุม (Command) หลังจากสัญญาน Reset แล้ว CPU ต้องออกคาสั่งเลือกโหมด จบแล้วจะตามด้วยรหัส SYNC หรือรหัสคาสั่งควบคุม แล้วแต่ว่าจะเลือกการทางานในแบบซิงโครนัสหรืออะซิงโครนัส โดย CPU จะต้องส่งรหัสคาสั่งหรือข้อมูลเป็นลาดับขั้นตอน ดังตาราง 3
4. ขั้นตอนที่ 2 และ 3 ให้ข้ามไปได้เลยถ้าเลือกโหมดอะซิงโครนัส และขั้นตอนที่ 3 อาจไม่ต้องทาถ้าเลือกขบวนการซิงโครนัสแบบใช้รหัสซิงค์ตัวเดียว สาหรับขั้นที่ 5 และ 7 หมายถึงการรับหรือส่งข้อมูลต่อเนื่องกันไป จนกว่าจะเปลี่ยนโหมดหรือรหัสควบคุม ส่วนขั้นตอนที่ 6 หรือ 8 จะทาถ้าต้องการเปลี่ยนรหัสควบคุมแต่ไม่เปลี่ยนแปลงโหมด

<u>คาสั่งเลือกโหมดและรหัสควบคุม</u>

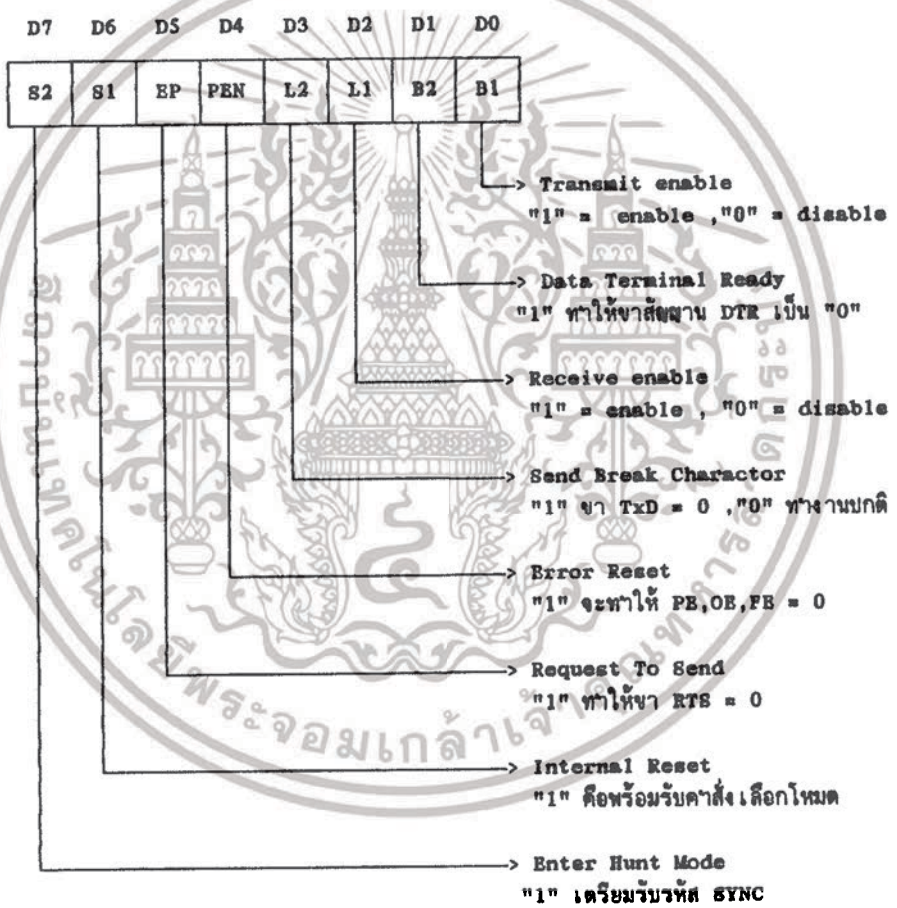
ลักษณะคาสั่งเลือกโหมดทั้ง 8 บิท ของคาสั่งที่ CPU ส่งมาให้ 8251 เพื่อเริ่มต้นการทางานจะมีความหมายดังรูปที่ 13 สาหรับโหมดแบบอะซิงโครนัส และเป็นดังรูปที่ 14 สาหรับโหมดแบบซิงโครนัส ต่อจากนั้นเป็นการส่งรหัสควบคุม ซึ่งมีความหมายดังรูปที่ 15



รูปที่ 13 รหัสเลือกโหมดแบบอะซิงโครนัส



รูปที่ 14 รหัสเลือกโหมดแบบซิงโครนัส

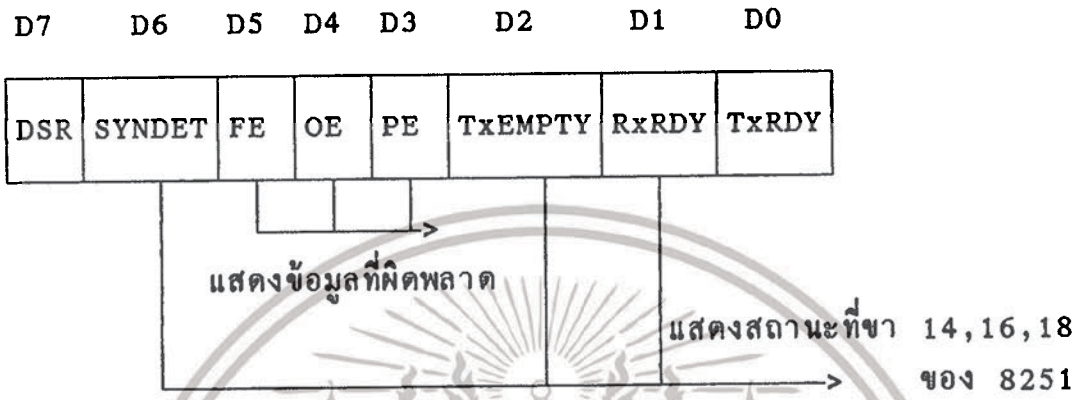


รูปที่ 15 แสดงรหัสควบคุมแต่ละบิท

สังเกตุที่การเลือกอัตราบอดของบิท D_0 และ D_1 ในรูปที่ 15 ความหมายของ 16X และ 64X คือการนำความถี่ของสัญญานฐานเวลา $\overline{TxC}$ หรือ $\overline{RxC}$ มาหารด้วย 16 และ 64 ตามลำดับ แล้วนำไปกำหนดอัตราบอด ซึ่งมีประโยชน์ในกรณีที่ต้องใช้วงจรกำเนิดสัญญานนาฬิกาความถี่สูง ๆ แล้วต้องนำมาหารให้เหลือความถี่เท่ากับอัตราบอด การเลือก 16X และ 64X จะช่วยลดวงจรหารให้น้อยลงได้

ไบต์แสดงสถานะ

ไบต์แสดงสถานะนี้มีขนาด 8 บิทแต่ละบิทจะแสดงสถานะของ 8251 โดยมีความหมายเป็น "ON" กับ "OFF" ความหมายของแต่ละบิทแสดงดังรูปที่ 16



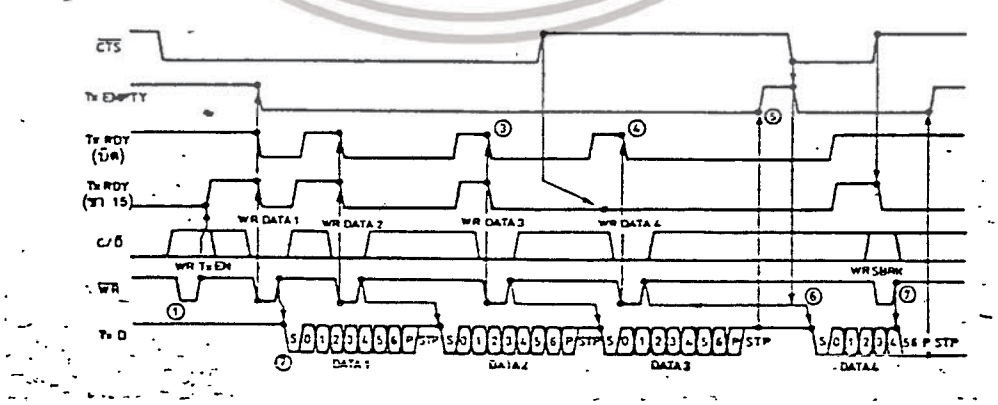
รูปที่ 16 ไบต์แสดงสถานะของ 8251

1. TxRDY (D_0) จะเป็น "1" ทันทีที่ Transmit Buffer A ว่าง แสดงถึงความพร้อมที่จะรับข้อมูลไปส่งได้แล้ว และยังแสดงถึงสัญญาน TxRDY ที่ขา 15 อีกด้วย
2. RxRDY แสดงถึงได้รับข้อมูลเข้ามาครบไบต์แล้วให้ CPU มาอ่านไปได้ และยังมีลอจิกเหมือนกับสัญญานที่ขา 14 อีกด้วย
3. Parity Error (PE) จะเป็น "1" เมื่อเกิดการผิดพลาดจากการตรวจสอบบิทพาริตี้ และจะถูกรีเซ็ตเป็น "0" โดย ER ในหัสควบคุมจาก CPU
4. Overrun Error (OE) จะเป็น "1" ถ้า CPU ไม่มาอ่านข้อมูลที่ 8251 ได้รับเข้ามาไว้ใน Receive Buffer A แล้วมีข้อมูลใหม่เข้ามาอีก และ OE จะถูกรีเซ็ตเป็น "0" โดย ER ในรหัสควบคุม
5. TxEMPTY แสดงลอจิกของสัญญาน TxEMPTY ที่ขา 18 โดยจะเป็น "1" เมื่อ 8251 ว่างหรือไม่มีข้อมูลจะส่ง แต่จะกลับเป็น "0" เมื่อได้รับข้อมูลจาก CPU ในขณะที่ภาคส่งได้รับการอีนาเบิ้ล
6. Framing Error (FE) ใช้ในขบวนการอะซิงโครนัสเท่านั้นโดย FE จะเป็น "1" ถ้าข้อมูลที่รับเข้ามาขาดบิทสตอบไปและ FE จะถูกรีเซ็ตเป็น "0" โดย ER ในรหัสควบคุม

ทั้ง PE,OE และ FE จะไม่ขัดขวางการทำงานของ 8251 แต่ประการใด เป็นเพียงการเตือนให้ทราบถึงการเกิดการผิดพลาดแล้วเท่านั้น

<u>การส่งข้อมูลแบบอะซิงโครนัส</u>

รูปที่ H เป็น Diagram การส่งข้อมูลขนาด 6 บิท 1 Parity 2 Stop Bit เริ่มด้วย CTS เป็น "0" อยู่ก่อนแล้วซีพียูส่งคำสั่งมายัง 8251 โดย C/$\overline{D}$ = "1" โดยส่งรหัสคำสั่ง $\overline{WR}$ TxEN มาให้ 8251(1) (สมมุติซีพียูได้เลือกแบบอะซิงโครนัสมาก่อนแล้ว) TxRDY เป็น "1" ซีพียูเริ่มส่งข้อมูลตัวแรก (DATA1) TxEMPTY และ TxRDY (ใน Status) และ TxRDY ขา 15 จะกลายเป็น "0" และข้อมูลแบบขนานจะถูกเลื่อนออกไปแบบอนุกรม(2) และ TxRDY ทั้งคู่ (ใน Status บิตและขา 15) จะกลับเป็น "1" แสดงว่า Buffer A ว่างพร้อมจะรับข้อมูลตัวใหม่จากซีพียู ซีพียูจะส่งข้อมูลตัวที่2 มาอีก(DATA2) ในขณะนั้น DATA1 กำลังถูกเลื่อนออกไปเมื่อ DATA(1) ถูกเลื่อนออกไปหมด TxRDY จะกลายเป็น "1" อีกครั้งซีพียูจะส่งข้อมูลตัวที่3 (DATA3) มาได้ ในขณะเดียวกัน DATA2 จะถูกส่งไปยัง Transmit Buffer B เพื่อเริ่มเลื่อนออกไปเมื่อหมด DATA2 DATA3 ก็จะถูกเลื่อนตามออกไป ถึงแม้ $\overline{CTS}$ จะกลายเป็น "1" แล้วก็ตาม แต่ DATA3 ถูกเขียนเข้าไปก่อนแล้ว จึงถูกส่งออกไป แต่ให้สังเกตุว่าเมื่อ $\overline{CTS}$ เป็น "1" และก่อนที่ซีพียูส่ง DATA4 เข้ามา TxRDY ใน Status บิทจะเป็น "1"แต่ TxRDY ขา 15 จะไม่เป็น "1" และ TXRDY ใน Status บิทถูกรีเซ็ตเป็น "0" เมื่อซีพียูส่ง DATA4(3) มาเมื่อ DATA3 ถูกส่งออกไปแล้ว TxEMPTY จะกลายเป็น "1" (4) ในลักษณะที่ TxEMPTY เป็น "1" แต่ TxRDY เป็น "0" แสดงให้เห็นว่ามี DATA อยู่ใน Transmit Buffer A แต่ยังไม่ถูกส่งให้ Transmit Buffer B นั่นคือมี DATA ค้างอยู่ใน 8251 อยู่ 1 ตัว จนกระทั่ง $\overline{CTS}$ กลับเป็น "0" อีก DATA4 จึงจะถูกส่งออกไป(5) แต่ยังส่งไปได้ไม่กี่บิท ซีพียูก็ส่งรหัสคำสั่งให้ 8251 หยุดหรือ Break(6) 8251 จะหยุดการส่งลงตามรูปที่ 17

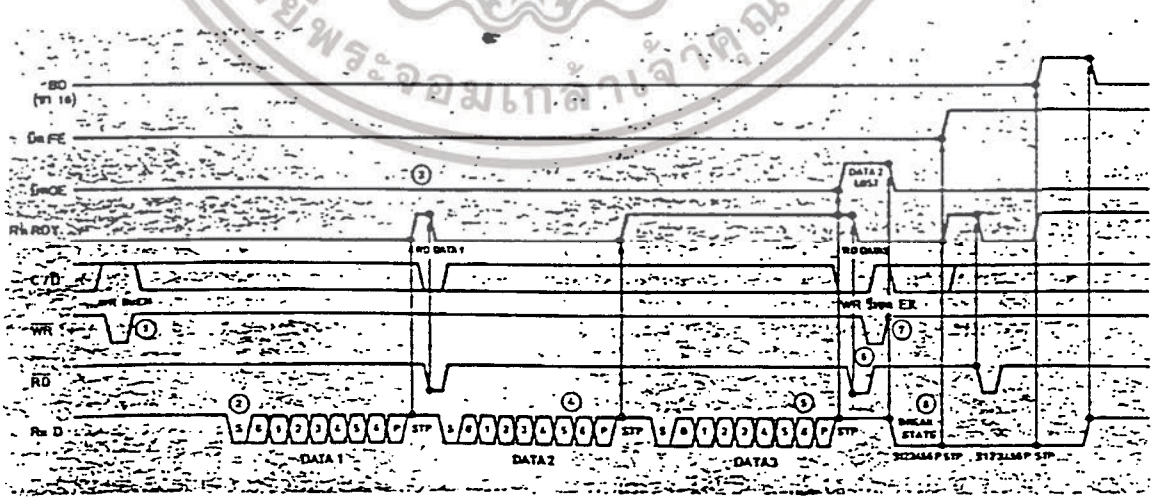


<u>รูปที่ 17</u> แผงผังเวลาแสดงการส่งข้อมูลแบบอะซิงโครนัส

การรับข้อมูลแบบอะซิงโครนัส

ตามรูป 18แสดงให้เห็นสัญญาณต่าง ๆ ซึ่งเกี่ยวข้องกันในการรับข้อมูลแบบอะซิงโครนัส ขั้นตอนต่าง ๆ จะเป็นดังนี้

1. ซีพียูส่งคำสั่ง RxEN มายัง 8251(โดย C/$\overline{D}$ = 0) มาให้ 8251 รับข้อมูลได้
2. ข้อมูลเริ่มมีเข้ามายัง 8251 ทางขา RxD เริ่มด้วย Start Bit ตามด้วยข้อมูลและพาริตี้และจบด้วย Stop Bit
3. ที่ Stop Bit นี้เองเป็นการบอกถึงจบ 1 ตัว RxRDY จะเป็น "1" แสดงว่ารับข้อมูลมา 1 ตัวแล้ว ซีพียูจะทำคำสั่ง Read มาที่ 8251 (C/$\overline{D}$ = 0) รับข้อมูลไปได้เลยแล้ว RxRDY จะกลับเป็น "0"
4. ข้อมูลตัวที่ 2 ส่งมาอีก RxRDY จะเป็น "1"
5. สมมุติว่า RxRDY เป็น "1" แล้ว ซีพียูยังไม่ยอมมาอ่านข้อมูลไป แล้ว CPU ยังไม่ยอมมาอ่านข้อมูลไป แล้วมีข้อมูลตัวที่3 เข้ามาอีก ทำให้บิท OE (Overrun Error) จะกลายเป็น "1" แสดงให้เห็นว่ามีข้อมูลที่สูญหายไปเพราะถูกส่งมาทับ (DATA2 หายไป)
6. ซีพียูทำ Read จะได้ DATA3 (DATA2 ถูกทับไปแล้ว) OE ไม่ได้ขัดขวางการทำงานของซีพียูแต่ประการใด
7. ซีพียูสามารถรีเซ็ต OE ใน Status Word ได้โดยส่งคำสั่งมารีเซ็ตได้
8. ในช่วงเวลา 8 นี้เป็นการแสดงการตรวจจับพบว่าข้อมูลที่รับเข้ามาเป็น Break Charactor BD จะเป็น "1"



รูปที่ 18 แผงผังเวลาแสดงการรับข้อมูลแบบอะซิงโครนัส

บทที่ 3
การทำงานของ ADVANCE TELEX

จากเดิมในระบบ SUPER TELEX ของกองเทเล็กซ์ การสื่อสารแห่งประเทศไทย เป็นระบบที่นำเอาเครื่องไมโครคอมพิวเตอร์ มาใช้แทนเครื่องเทเล็กซ์ ซึ่งจะเป็นการสะดวกแก่ผู้เช่าบริการเทเล็กซ์ ซึ่งส่วนใหญ่จะมีไมโครคอมพิวเตอร์ประจำอยู่ในสำนักงานหรือบริษัทของตนเองอยู่แล้ว เครื่องไมโครคอมพิวเตอร์นี้จะทำหน้าที่แทนเครื่องเทเล็กซ์ได้ทุกประการ โดยใช้งานร่วมกับโปรแกรม TOP (TELEX ON PC) ที่กองเทเล็กซ์การสื่อสารแห่งประเทศไทยได้พัฒนาขึ้น การใช้ประโยชน์ของบริการ SUPER TELEX นอกจากจะใช้งานได้เหมือนเครื่องเทเล็กซ์ทุกประการแล้ว ยังสามารถส่งข้อมูลที่เป็น Text File ซึ่งเก็บอยู่ในแผ่น Diskette ได้ทันทีโดยไม่ต้องนำมาเตรียมเพื่อส่งออก การแก้ไขข้อมูลสามารถทำได้สะดวก และรวดเร็ว ในการรับข้อมูลจะบันทึกลงบนแผ่น Diskette ทุกครั้งพร้อมทั้งตั้งชื่อ File ข้อมูลนั้นให้ด้วย

จากระบบ SUPER TELEX ที่กล่าวมายังมีจุดพกพร่องที่ต้องแก้ไข ตรงที่ถ้ามีการปิดเครื่องไมโครคอมพิวเตอร์ก็ไม่สามารถที่จะรับส่งข้อมูลเทเล็กซ์ได้ จากจุดนี้เอง จึงได้ออกแบบวงจรขึ้นมาอีกชุดหนึ่งเพื่อแก้ไขข้อบกพร่องดังกล่าว และเรียกวงจรชุดนี้ว่า ADVANCE TELEX ON PC

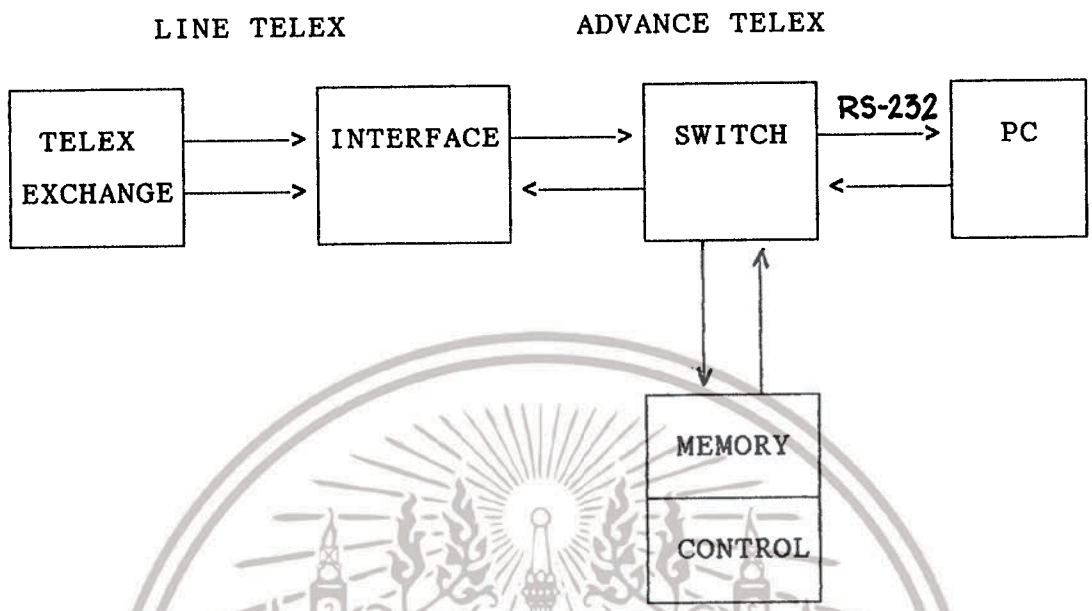
โครงสร้างของ SUPER TELEX (เดิม)

1. ชุด Interface
2. ส่วนที่เป็น Software (Program Top)
3. ไมโครคอมพิวเตอร์
4. Printer

โครงสร้างของ ADVANCE TELEX (ที่พัฒนาขึ้น)

1. ชุด Control และ Memory
2. ชุด Switch
3. ชุด Interface (จะใช้ร่วมกับ SUPER TELEX)
4. Software สำหรับใช้ในการติดต่อเพื่อ Load ข้อมูลจากหน่วยความจำ
5. ไมโครคอมพิวเตอร์ + Printer

Block Diagram แสดงการทำงานของระบบทั้งหมด



รูปที่ 19 แสดง Block Diagram การทำงานทั้งหมด

จากรูปที่ 19 แสดงส่วนประกอบของชุด ADVANCE TELEX จะแบ่งเป็น 3 ส่วนใหญ่ ๆ คือ

1. ชุด INTERFACE ทำหน้าที่แปลงสัญญาณเทเล็กซ์ที่อยู่ในรูปกระแส เป็น TTL ในลักษณะของ RS232
2. ชุด SWITCH จะมีลักษณะคล้ายกับ Switch 3 ทาง ทำหน้าที่เชื่อมต่อระหว่าง
   - INTERFACE กับ ชุด Control
   - INTERFACE กับ PC
   - ชุด Control กับ PC
3. ชุด Control + Memory ทำหน้าที่เป็นตัวควบคุมการทำงานของชุด ADVANCE TELEX และเก็บข้อความเทเล็กซ์

การทำงานสามารถแบ่งเป็น 2 ขั้นตอนได้ดังนี้

1. โปรแกรม TOP ทำงาน ชุด Control ของ ADVANCE TELEX ซึ่งจะทำหน้าที่ตรวจสอบสถานะของ PC ว่ามีการ Run Program Top หรือไม่ เมื่อทำการตรวจสอบสถานะแล้วพบว่า PC ได้ Run Program Top อยู่ ก็จะทำการต่อ Loop Current เข้ากับ PC ในช่วงนี้การรับส่งข้อความจะเป็นไปตามกระบวนการของระบบเทเล็กซ์

2. โปรแกรม TOP ไม่ทำงาน เมื่อชุด Control ตรวจสอบสถานะรู้ว่า PC ไม่ได้ Run Program Top ก็จะทำการตัด Line Telex ออกจากเครื่อง PC แล้วจะทำการต่อตัวมันเอง(ชุด Control) เข้ากับ Loop Current แทน PC ถ้ามีการเรียกเข้า ชุด ADVANCE TELEX จะทำหน้าที่ตอบรับข้อความและนำเอาข้อมูลไปเก็บในหน่วยความจำ ข้อมูลที่อยู่ในหน่วยความจำจะสามารถเรียกดูหรือลบออกได้จาก PC โดยใช้โปรแกรมที่พัฒนาโดยภาษา C

บทที่ 4
การทำงานของวงจร

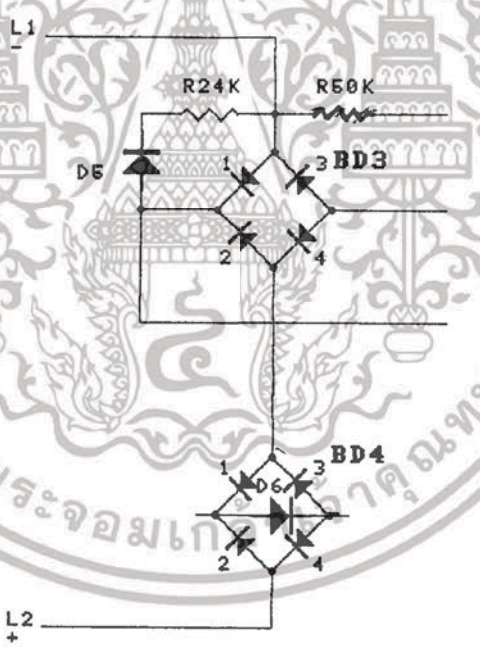
ในส่วนการทำงานของวงจรสามารถแยกการทำงานได้เป็น 2 ส่วนใหญ่ ๆ คือ

1. ชุด CURRENT LOOP ทำหน้าที่เปลี่ยนระบบ Current ของ Line Telex ไปเป็นสัญญาน TTL และต่อจากนั้นสัญญาน TTL จะถูกเพิ่มระดับสัญญานให้เป็นสัญญานในระบบ RS232 (ในกรณีที่ ชุด Loop Current ต่อกับ PC)

2. ชุด Control และ Memory ประกอบด้วย CPU #8085A ,ROM #2764 RAM #6264X2 ,PORT 8251 ,วงจรสร้างความถี่สำหรับสร้าง Baud Rate ให้ 8251 และวงจร Buffer (ทำหน้าที่เหมือน Switch 3 ทาง)

1. การทำงานของ LOOP CURRENT

- ในสภาะปกติ (Idle) การทำงานของวงจรจะเป็นดังนี้



รูปที่ 20 แสดงวงจรในภาวะ IDLE

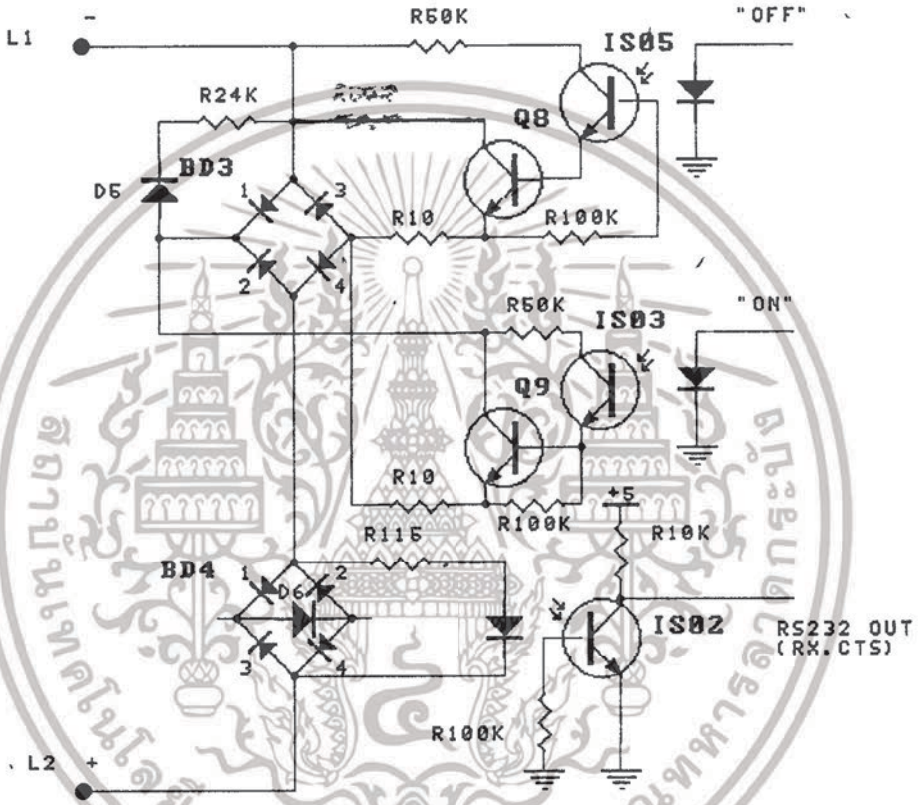
1. สถานะของ Current Loop ในวงจรเมื่อวัดด้วย Meter จะได้ -5 mA การทำงานของวงจรคือที่จุด L1 กระแสจะไหลผ่าน R24K ,D5 ,BD3 ชุด2, BD4 ชุด3 และครบ Loop ที่ L2

2. BD3 ชุด3 จะมีสภาพเหมือน Short เนื่องจาก IS05 ทำงานทำให้ทรานซิสเตอร์ Q8 ทำงานตามไปด้วย จึงเสมือน BD3 ชุด3 Short อยู่

3. จากจุด RS232 เข้ามา (TX,RTS) สัญญาน Logic ทั้งคู่จะมีสภาพเป็น "0" และจุด RS232 ออกไป (RX,CTS) สัญญาน Logic ของ RX จะเป็น "1" และ CTS จะเป็น "0"

4. LED ที่ติดสว่างจะมีหลอด Idle และหลอด DTR

- เมื่อมีการเรียกเข้า (Incoming Call) ลักษณะของวงจร Current Loop จะแสดงได้ดังนี้



รูปที่ 21 แสดงวงจรเมื่อมีการเรียกเข้า

1. จากกสภาพเดิมที่ IS05 "ON" อยู่ทำให้สภาพของ BD3 ชุด3 Short ตามไปด้วย

2. เมื่อชุมสายส่งสัญญาน +20 mA เข้ามา กระแสจะไหลผ่าน BD3 ชุด3 ที่ Short อยู่ ผ่านมายัง BD3 ชุด4 => BD4 ชุด1 => D6 => BD4 ชุด4 และครบวงจรที่จุด L2 ตามรูป

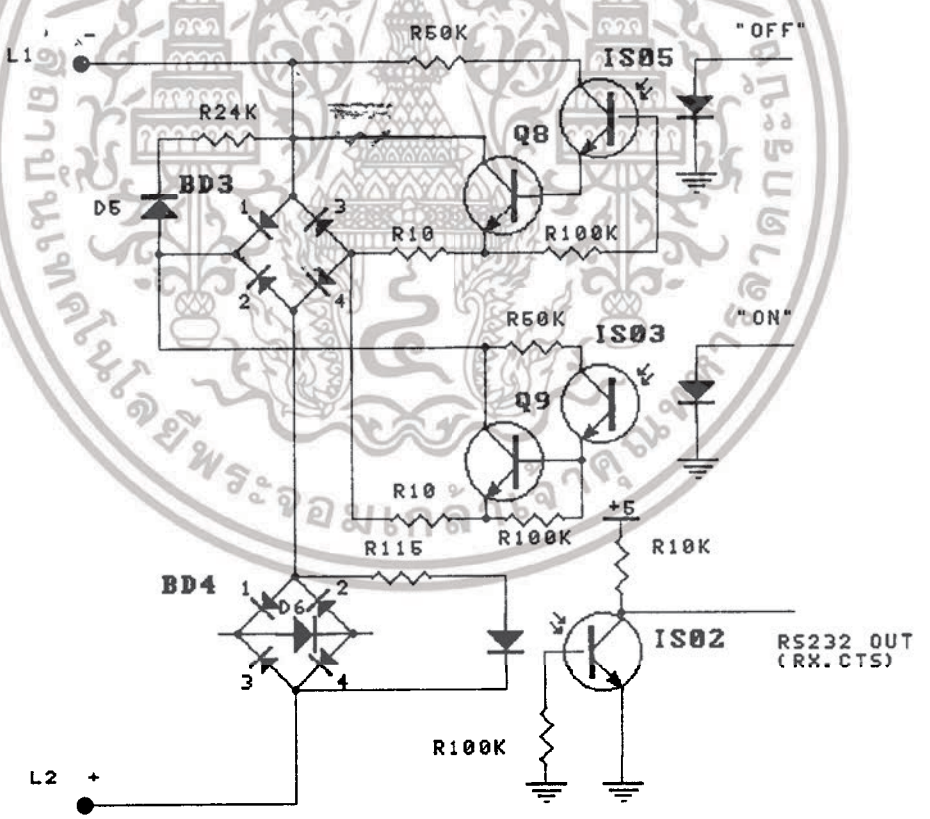
3. สังเกตุที่ IS02 ซึ่งต่อคร่อม BD4 ชุด1 และ BD4 ชุด4 อยู่จะทำหน้าที่ Couple สัญญานที่ส่งมาจากชุมสาย ออกไปยัง RS232 Output นั่นคือเป็นสัญญาน RX และ CTS ที่ต้องป้อนเข้าสู่เครื่อง PC ที่มีการ Load โปรแกรม SUPER TELEX ลงไปแล้ว เมื่อโปรแกรมรับทราบว่ามีสัญญาน Telex เข้ามา ก็จะส่งสัญญาน TX และ RTS

ตอบกลับออกไปนั่นคือสัญญาน Logic ที่ขา 2,4 จะเป็น "1" ทั้งคู่ จึงมีผลทำให้ IS05 มีสภาวะ "OFF" คือขา Anode ของ IS05 เป็น "0" นั่นหมายถึงการคืนสภาพของ BD3 ชุด3 กลับคืนมา

4. ทำให้กระแสเดิมไหลผ่าน BD3 ชุด3 ที่ Short อยู่เดิม หันเหเปลี่ยนทางไหลผ่าน BD3 ชุด1 ผ่านมาที่ IS03 ซึ่งขณะนี้มีสภาพ "ON" อยู่คือที่ Anode มีลอจิก "1" ทำให้สภาพของทรานซิสเตอร์ Q9 เสมือน Short เมื่อผ่าน IS03 แล้วจะผ่านมายัง BD3 ชุด4 => BD4 ชุด1 => D6 => BD4 ชุด4 และครบวงจรที่ L2

5. สภาพครบวงจรเช่นนี้ ทำให้ชุมสายเทเล็กซ์สามารถต่อผ่าน Line Telex เข้ากับเครื่อง SUPER TELEX ได้แล้ว และสามารถรับส่งข้อมูลกันได้โดยมี IS02 เป็นตัว Couple สัญญานข้อมูลจากสภาพของ Current Loop ให้อยู่ในรูปของ TTL และจาก TTL เป็น RS232 โดย Driver 1488

- เมื่อมีการเรียกออก (Out Going) สถานะของวงจร Current Loop จะเป็นดังนี้



รูปที่ 21 แสดงวงจรเมื่อมีการเรียกออก

1. เมื่อมีการเรียกออกจะทำให้

TX (ขา2 ของ RS232 Input) เป็น "1"

RTS (ขา4 ของ RS232 Input) เป็น "1"

2. IS05 ที่ขา Anode จะมีสภาพเป็น "0" นั่นคือ IS05 "OFF" คืนสภาพของ BD3 ชุด3 ให้คืนสภาพปกติ (จากเดิมเป็น Short)
3. IS03 ที่ขา Anode จะมีสภาพเป็น "1" นั่นคือ IS03 "ON" ทำให้ทรานซิสเตอร์ Q9 ทำงานมีสภาพเป็น Short จึงทำให้จุด N และ P เสมือน Short ถึงกันอยู่
4. จะเกิดกระแส -20 mA ขึ้นโดยเกิดจากการไหลของกระแสจาก L1 ผ่าน BD3 ชุด3 ผ่าน BD3 ชุด2 (เมื่อจุด N และ P Short กัน) ผ่าน BD4 ชุด2 ผ่าน D6 => BD4 ชุด3 และครบวงจรที่ L2
5. เมื่อชุมสายเห็นสัญญานจาก -5 mA เปลี่ยนเป็น -20 mA ก็จะทำการกลับ Current Line จาก -20 mA ให้เป็น +20 mA มาให้
6. สภาพจึงเปลี่ยนแปลงกลับกันนั่นคือ กระแสจะไหลผ่าน BD3 ชุด1 ผ่านจุด N และ P ที่ Short กันอยู่ ฝ่าย BD3 ชุด4 => BD4 ชุด1 => D6 => BD4 ชุด4 และครบวงจรที่ L2
7. เมื่อชุมสายกลับ Line Current (จาก -20 mA => +20 mA) มาให้แล้วนั้น ชุมสายก็จะส่งสัญญาน BKK GA มาให้ด้วย
8. IS02 ก็จะมีหน้าที่ Couple สัญญาน Current Loop ให้อยู่ในรูปของสัญญาน TTL เพื่อส่งผ่าน Driver เบอร์ 1488 (ที่เป็น RS232 Output) ออกไปยังเครื่อง PC
9. IS03 ก็จะมีหน้าที่ Couple สัญญาน TTL ให้เป็น Current Loop เพื่อส่งข้อความผ่านชุมสายไปยังเครื่องเทเล็กซ์ปลายทาง

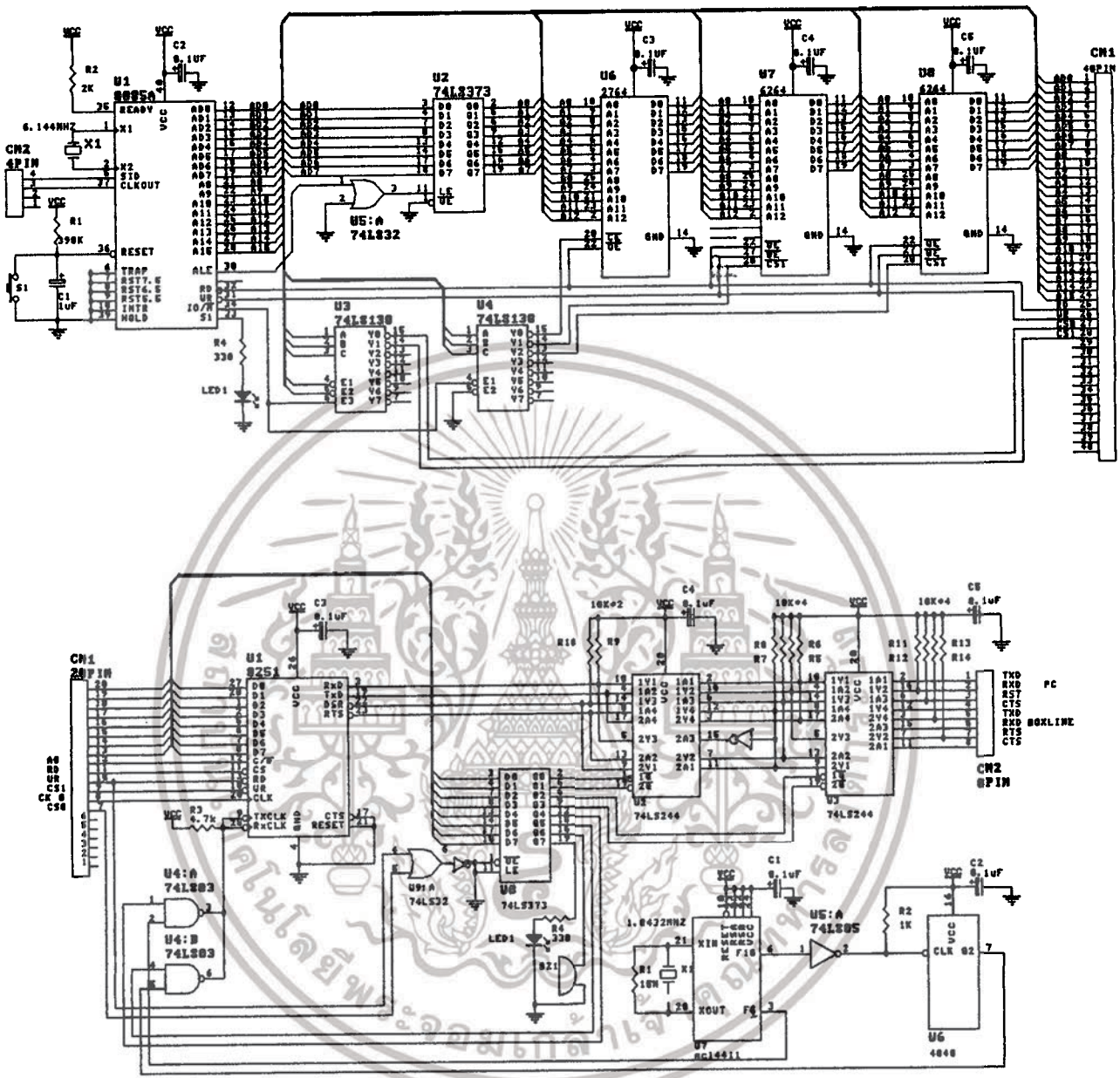
- การทำงานของส่วนอื่น ๆ ของชุด Current Loop

1. วงจรกลุ่ม U1B,U1C จะต่อเป็นลักษณะของวงจร Delay ที่มีค่าของ Time Delay เข้ามายุ่ง มีผลด้านการ Cut Line Telex ด้วยการกด ESC
2. ลักษณะการกด ESC หมายถึงการ Break Line นั่นคือการ Clear ให้ TX คืนสภาพจาก High ("1") เป็น Low ("0")
3. ลอจิก "0" ของ TX จะผ่านวงจร Delay Time มา Trig Gate U5A เพื่อให้ปล่อยสัญญาน "0" ไป OR กับสัญญาน TX เดิม เพื่อให้ Output ที่ได้คือจะเป็นลอจิก "1" AND กับสัญญานเทเล็กซ์ ให้ผลของลอจิกไป Set สภาพของ RX และ CTS ให้กลับคืนสภาพเดิม คือจาก

Low เป็น High และจาก High เป็น Low เสมือนทำให้เกิดการ Cut Line ให้ขาดกัน

4. IS01 โดยสภาพปกติจะ "ON" อยู่ ในกรณีที่กระแสเป็น -5 mA,-20 mA หลอด LED ที่แสดงสภาวะ Idle จะติดสว่าง และเมื่อชุมสายกลับ Line จาก -20 mA เป็น +20 mA มาทำให้ IS01 "OFF" LED ที่แสดงสภาวะ Idle จะดับ

2. การทำงานของวงจร CONTROL



รูปที่ 23 แสดงวงจรชุด Control ทั้งหมด

การติดต่อ 8085 กับ หน่วยความจำ (ดูรูปที่ 23 ประกอบ)

หน่วยความจำที่ใช้จะมี 2 แบบ คือ ชนิด EPROM เบอร์ 2764 ขนาด 8 KByte 1 ตัว และ RAM เบอร์ 6264 ขนาด 8KByte 2 ตัว

2764 เป็น EPROM ขนาด 8KX8 บิท ที่ผู้ใช้สามารถเขียนข้อมูลลงไปได้โดยเครื่องโปรแกรม EPROM (EPROM PROGRAMMER) ผู้ใช้ไม่สามารถใช้คำสั่งการทำงานของ 8085 เขียนข้อมูลลงไปใน EPROM แบบนี้ได้ ขาสัญญาน A_0 ถึง A_{12} เป็นขาสัญญาณที่ชี้ตำแหน่ง(Address) ของหน่วย

ความจำในกรณีนี้มีทั้งหมด 13 ขา จึงชี้ตำแหน่งหน่วยความจำได้ 2^{13} เท่ากับ 8192 หรือ 8 KByte ตำแหน่ง D_0 ถึง D_7 เป็นขาที่ใช้ส่งข้อมูลจากตำแหน่งที่ชี้โดยสัญญาณที่ขา A_0 ถึง A_{12} ออกมาภายนอก2764 ดังนั้นข้อมูลจะส่งได้ที่ละ 8 บิท ส่วนสัญญาน $\overline{CS}$ และ $\overline{OE}$(Active Low)ใช้ควบคุมการอ่านข้อมูลกับ 2764

6264 เป็น RAM ขนาด 8KX8 บิท หน่วยความจำนี้สามารถเขียนหรืออ่านข้อมูลได้ขา Address A_0-A_{12} จะใช้สำหรับป้อนค่าตำแหน่งหน่วยความจำที่ต้องการติดต่อด้วย (2^{13} = 8 KByte) ตำแหน่ง D_0 ถึง D_7 ใช้สำหรับส่งข้อมูลเข้าไปยัง 6264 หรือรับข้อมูลออกจาก 6264 ส่วนสัญญาน $\overline{CS}$,$\overline{OE}$ และ $\overline{WR}$ (ทั้งหมดจะ Active Low) ใช้ควบคุมการอ่านหรือเขียนข้อมูลกับ 6264

วงจรถอดรหัส (Decoder) เป็นวงจรที่ทำหน้าที่เปลี่ยนรหัสจากเลขฐาน 2 เป็นรหัสอื่น จากวงจรจะใช้ TTL เบอร์ 74LS138 (U4) เป็นตัวถอดรหัสแบบ 3 to 8 Line Decoder ทำการถอดรหัสตำแหน่งหน่วยความจำ รูปที่ 24 เป็น Diagram และตาราง Function ของ 74LS138

INPUT					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

<u>รูปที่ 24</u> Dagram ภายในของ 74LS138

จากตารางจะแสดงว่าถ้าขา G1 มีลอจิกเป็น "1" และ G2* มีลอจิก "0" จะทำให้ 74LS138 ทำการถอดรหัสเลขฐาน2 ที่เข้ามาทางขา A,B และ C G2* เป็นผลลัพธ์การ OR กันของลอจิกที่เข้ามาที่ขา G2A และ G2B ถ้าสัญญานที่เข้ามาที่ขา A,B และ C เป็น 000 จะทำให้สถานะลอจิกที่เอาท์พุทที่ขา Y0 เป็น "0" (L) ส่วนขาอื่น ๆ จะมีลอจิกเป็น "1" (H) แต่ถ้าสัญญานที่ขา G1 ไม่มีลอจิกเป็น "1" หรือ G2 เป็น "0" จะไม่เกิดการถอดรหัสทำให้ขา Y0 ถึง Y7 มีลอจิกเป็น "1"

74LS138 ในรูปที่ 24 ใช้ถอดรหัสตำแหน่งหน่วยความจำของ 8085 โดยขาสัญญาน A_{13} ถึง A_{15} จะถูกต่อมาที่ขา A,B,C ของ 74LS138 จากวงจรจะเห็นว่า G1 ต่อไว้ที่สถานะลอจิก "1" อยู่แล้วเมื่อมีการติดต่อกับหน่วยความจำ ขาสัญญาน $IO/\overline{M}$ ของ 8085 ซึ่งต่อกับขา G2A ของ 74LS138 (G2B ต่อลง Ground) จะให้ลอจิก "0" ออกมา 74LS138 ก็จะทำการถอดรหัสหน่วยความจำตามที่ผู้ใช้กำหนดออกมา

74LS373 จำทำหน้าที่ Lacth ตำแหน่งหน่วยความจำ 8 บิทล่าง (AD_0 ถึง AD_7) ข้อมูลที่ออกจาก 74LS373 จะเป็นค่า 8 บิทล่างของตำแหน่งหน่วยความจำที่ต้องการติดต่อ และในวงจรได้ต่อตำแหน่งหน่วยความจำ 8 บิทล่างเข้าหน่วยความจำ (2764 และ 6264) และ Address A_8 ถึง A_{12} ของ 8085 จะต่อโดยตรงเข้ากับ A_8 ถึง A_{12} ของหน่วยความจำ เพราะตำแหน่งหน่วยความจำ 8 บิทบนที่ออกมาจาก A_8 ถึง A_{15} จะคงที่ตลอดเวลา (A_{13} ถึง A_{15} จะต่อเข้าขา A,B,C ของ 74LS138 เพื่อทำการถอดรหัสหน่วนความจำ)

ในการติดต่อกับหน่วยความจำเริ่มจาก 8085 ทำการส่งตำแหน่ง Address มาพร้อมกับสัญญาน Io/M (Io/M = "0"),$\overline{RD}$ และ $\overline{WR}$ (RD,WR Active ที่ Low) มา ทำให้ 74LS138 ทำการถอดรหัสตำแหน่งหน่วยความจำที่ต่อจาก A_{13} ถึง A_{15} มายัง A,B,C ของ 74LS138 ตามลำดับ ดังนั้นถ้า 8085 ทำการติดต่อกับหน่วยความจำ ณ ตำแหน่ง

0000 0000 0000 0000 (0000H)

0001 1111 1111 1111 (1FFFH)

จะทำให้ Y0 มีสภาวะเป็นลอจิก "0" และถ้า 8085 ทำการติดต่อกับหน่วยความจำตำแหน่ง

0010 0000 0000 0000 (2000H)

0011 1111 1111 1111 (3FFF)

จะทำให้ Y1 มีสภาวะเป็นลอจิก "0" และถ้า 8085 ทำการติดต่อกับหน่วยความจำตำแหน่ง

0100 0000 0000 0000 (4000H)

0101 1111 1111 1111 (5FFFH)

จะทำให้ Y2 มีสภาวะเป็นลอจิก "0"

ขา Y0 จะถูกต่อเข้ากับขา CE ของ 2764 และขา Y1 และ Y2 จะต่อเข้ากับขา CE ของ 6264 ดังนั้นเมื่อ 8085 ทำการติดต่อกับหน่วยความจำ 0000 ถึง 1FFFH จะเป็นการเลือกติดต่อกับ 2764 หรือ EPROM และถ้าเป็นการติดต่อกับหน่วยความจำ 2000 ถึง 5FFFH จะเป็นการติดต่อกับ 6264 หรือ RAM

การติดต่อระหว่าง 8085 กับ 8251

ขาสัญญานของ 8251 สามารถที่จะต่อเข้ากับ 8085 ได้โดยตรงจากรูป ขาสัญญาน D_0 ถึง D_7 ,WR,RD,CLK และ C/D จะต่อเข้ากับขาสัญญาน AD_0 ถึง AD_7 ,WR,RD,CLOCK OUT และ A_8 ของ 8085 โดยตรง

การเลือกติดต่อกับชิพ 8251 จะถูกกำหนดโดยขาสัญญาน CS (จะ Active ที่ลอจิก "0")ส่วนขาสัญญาน C/D ถ้าเป็น "1" คือสัญญานควบคุมใช้คู่กับ RD และ WR เป็นการเขียนอ่านรหัสควบคุม(Control) แต่ถ้าเป็น "0" จะใช้ร่วมกับ RD และ WR หมายถึงการอ่านข้อมูล

การกำหนดอัตราบอดของ 8251 จะใช้สัญญานนาฬิกาจากภายนอกเป็นตัวกำหนดโดยจะป้อนเข้าทางขา TxC และ RxC โดยใช้ไอซีเบอร์ MC14411 เป็นตัว Generate ความถี่ที่ใช้ อัตราบอดที่ใช้ในที่นี้จะใช้เป็น 50 บอด สำหรับการติดต่อระหว่าง Line Telex กับ 8251 โดยเลือกตัวหารภายใน 8251 ไว้เท่ากับ 64X ดังนั้น MC14411 จะต้องกำเนิดความถี่เท่ากับ 3200 Hz (เมื่อหารด้วย 64 จะได้เท่ากับ 50 บอด) ส่วนอัตตราบอด 6400 บอด จะใช้ติดต่อระหว่าง 8251 กับ PC ดังนั้น MC14411 จะต้องกำเนิดความถี่ออกมาเท่ากับ 409.6 KHz (เมื่อหารด้วย 64 เท่ากับ 6,400 บอด)

การเชื่อมต่อเอาท์พุทของ 8251 กับ RS232 จะเชื่อมต่อกันโดยตรงไม่ได้ เอาท์พุทของ 8251 จะมีลอจิกเป็นระดับ TTL ซึ่งเป็นระดับลอจิก "0" และ "1" เป็น 0 และ 5 Volt ตามลำดับ ดังนั้นจึงต้องแปลงระดับลอจิกไปเป็นบวกลบ 12 Volt เพื่อให้ตรงกับมาตรฐานของ RS232 โดย RS232 จะใช้ระดับแรงดันที่ไม่เท่ากับ TTL เพราะต้องการส่งให้ได้ระยะทางไกลขึ้น RS232 ใช้วิธีการที่แตกต่างจากลอจิกธรรมดา คือลอจิก "1" เป็น -12 Volt และลอจิก "0" เป็น +12 Volt ดังนั้นจึงต้องมีการ Inverse ลอจิกของเอาท์พุทของ RS232 โดยใช้ไอซีเบอร์ MC1488 แปลงลอจิก TTL ไปเป็นสัญญาน RS232 และ MC1489 แปลงสัญญาน RS232 มาเป็น TTL

ไอซี 74LS373 (U8) จะ Latch ขาสัญญาน AD_0 ถึง AD_7 ของ 8085 เพื่อทำหน้าที่เป็นตัวเลือกความถี่ให้ 8251 เพื่อผลิตอัตราบอดเลท(50 หรือ 6,400 บอด) และจะเป็นตัวเลือกให้ 74LS244 ชุดใดทำงาน

ไอซี 74LS244 เป็น Buffer ที่มีลักษณะเป็นวงจรลอจิกสามสถานะสองทิศทาง จากวงจรใช้ 74LS244 สองตัวในการเลือกติดต่อระหว่าง 8251 กับ Line Telex หรือ 8251 กับ PC หรือ Line Telex กับ PC

ไอซี 74LS138 (U3) จะเป็นตัวถอดรหัสเพื่อกำหนดพอร์ทให้ 8251 หรือ 74LS373 (U8)ทำงาน โดยขา G2A,G2B,A,B,C และ G1 จะต่อเข้ากับขา A_{13} ถึง A_{15} และ Io/M ของ 8085 ตามลำดับ สำหรับการกำหนดหมายเลขพอร์ตสำหรับการติดต่อกับอินพุทเอาท์พุทพอร์ตจะเป็นดังนี้

A15 A14 A13 A12 A11 A10 A9 A8

0 0 0 0 0 0 0 0 ==> PORT 00H

จะกำหนดการทำงานของ 74LS373 (U8)

A15 A14 A13 A12 A11 A10 A9 A8

0 0 0 0 1 0 0 0 ==> PORT 08H

จะกำหนดการทำงานของ 8251 ในการเขียนอ่านรหัสควบคุม

A15 A14 A13 A12 A11 A10 A9 A8

0 0 0 0 1 0 0 1 ==> PORT 09H

จะกำหนดการทำงานของ 8251 ในการเขียนอ่านข้อมูล

วิจารณ์และสรุป

จากโครงงานนี้ทางผู้จัดทำได้ดำเนินการสร้าง ทดลองและแก้ปัญหาเพื่อให้โครงงานนี้ ทำงานได้อย่างมีประสิทธิภาพมากที่สุด

ปัญหาเรื่องการออกแบบ ในส่วนของชุด Interface ที่ทำหน้าเปลี่ยน Current Loop ไปเป็นสัญญาน TTL และ RS232 ตามลำดับนั้น จะต้องคำนึงถึงอุปกรณ์ที่นำมาใช้เพื่อรักษาระดับ Current ไว้ที่ -5 mA ในขณะ Idle และ +20,-20 mA ขณะใช้งาน นอกจากนี้ต้องคำนึงถึงการหน่วงเวลาของระบบ Signalling ของชุมสายเทเล็กซ์เราจะต้องปรับค่า Resistor เกือกม้าให้สัมพันธ์กับการตัด Line ของระบบเมื่อสิ้นสุดการใช้งาน

ส่วนชุด Control ปัญหาอยู่ที่การเลือกใช้อุปกรณ์ให้สอดคล้องกับระบบ เช่น การใช้ 74LS244 ทำหน้าที่เป็น Data Switch 3 ทางให้ระบบเพื่อเลือกเส้นทาง การที่เลือกใช้ 74LS244 เพราะง่ายต่อการ Control Switch ได้รวดเร็วถูกต้องไม่ต้องมี Port เพิ่มเติมให้ยุ่งยาก CPU ใช้ 8085 ซึ่งมีขาพิเศษที่รับข้อมูลแบบ Series คือ SID (Series Input Data) ที่สอดคล้องกับระบบเทเล็กซ์ ไม่ต้องใช้ Port เพิ่มเติม

ปัญหาด้านการเขียนโปรแกรมส่วนใหญ่จะมุ่งเน้นไปที่การ Synchronous กับ Signalling ของชุมสายเทเล็กซ์ เช่นการหน่วงเวลา เพื่อรอช่วงจังหวะการตัด Line การรับ BKK GA ของเครื่องตลอดจนช่วงเวลาต่าง ๆ มีการออกแบบการรับส่งข้อมูลโดยใช้ระบบ Hand Checking หรือระบบตรวจสอบข้อมูลก่อนการรับส่งข้อมูล

บริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์ต่อหน่วยงานทั้งภาครัฐบาลและภาคเอกชน โดยโครงงานที่สร้างขึ้นนี้จะเป็นการประหยัดต่อค่าใช้จ่าย และสะดวกต่อการใช้งานอย่างมาก ต่อการที่จะต้องทำการจัดซื้อเครื่องใหม่ หรือในการเช่ายืมของบริษัทต่าง ๆ ที่จำเป็นต้องใช้บริการเทเล็กซ์ และยังรวมถึงค่าบำรุงรักษาเครื่องเทเล็กซ์ สำหรับโครงงานนี้เพียงแต่ทางหน่วยงานต่าง ๆ มีคู่สายเทเล็กซ์ และเครื่องไมโครคอมพิวเตอร์ (ซึ่งตามหน่วยงานหรือบริษัทส่วนมากจะมีเครื่องไมโครคอมพิวเตอร์อยู่แล้ว) ก็สามารถที่จะนำโครงงานนี้ไปใช้งานได้เหมือนเครื่องเทเล็กซ์ทุกประการ โดยไม่ต้องติดตั้งอุปกรณ์ใด ๆ เพิ่มเติม และในกรณีที่เครื่องไมโครคอมพิวเตอร์ปิดอยู่ หรือช่วงหลังเวลาทำงาน ก็ยังสามารถที่จะรับข้อความเทเล็กซ์ได้ โดยข้อความเทเล็กซ์จะถูกเก็บไว้ในหน่วยความจำเมื่อต้องการเรียกออกมาดูหรือพิมพ์ออกเครื่องพิมพ์ก็สามารถทำได้โดยการทำงานบนเครื่องไมโครคอมพิวเตอร์

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่องนี้ได้รับแนวความคิดจาก อาจารย์ชวลิต เบญจางคประเสริฐ และ อาจารย์ไพศาล สิทธิโยภาสกุล ทั้งยังให้คำแนะนำ และสนับสนุนมาโดยตลอด จึงขอกราบขอบพระคุณ อาจารย์ชวลิต เบญจางคประเสริฐ และ อาจารย์ไพศาล สิทธิโยภาสกุล เป็นอย่างสูงไว้ ณ ที่นี้

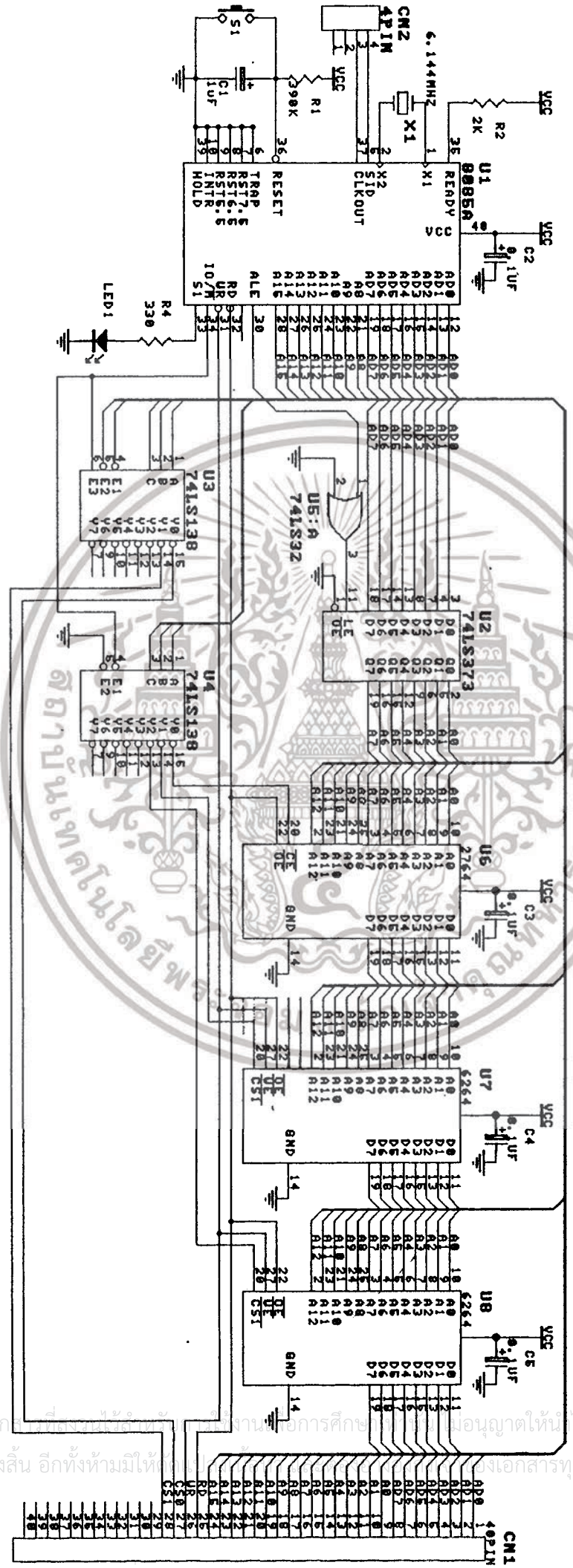
ขอขอบคุณอาจารย์ภาควิชาภาควิชาอิเล็กทรอนิกส์ที่ให้ความร่วมมือและช่วยเหลืออำนวยความสะดวกในการทำปริญญานิพนธ์ด้วยดีตลอดมา

เอกสารอ้างอิง

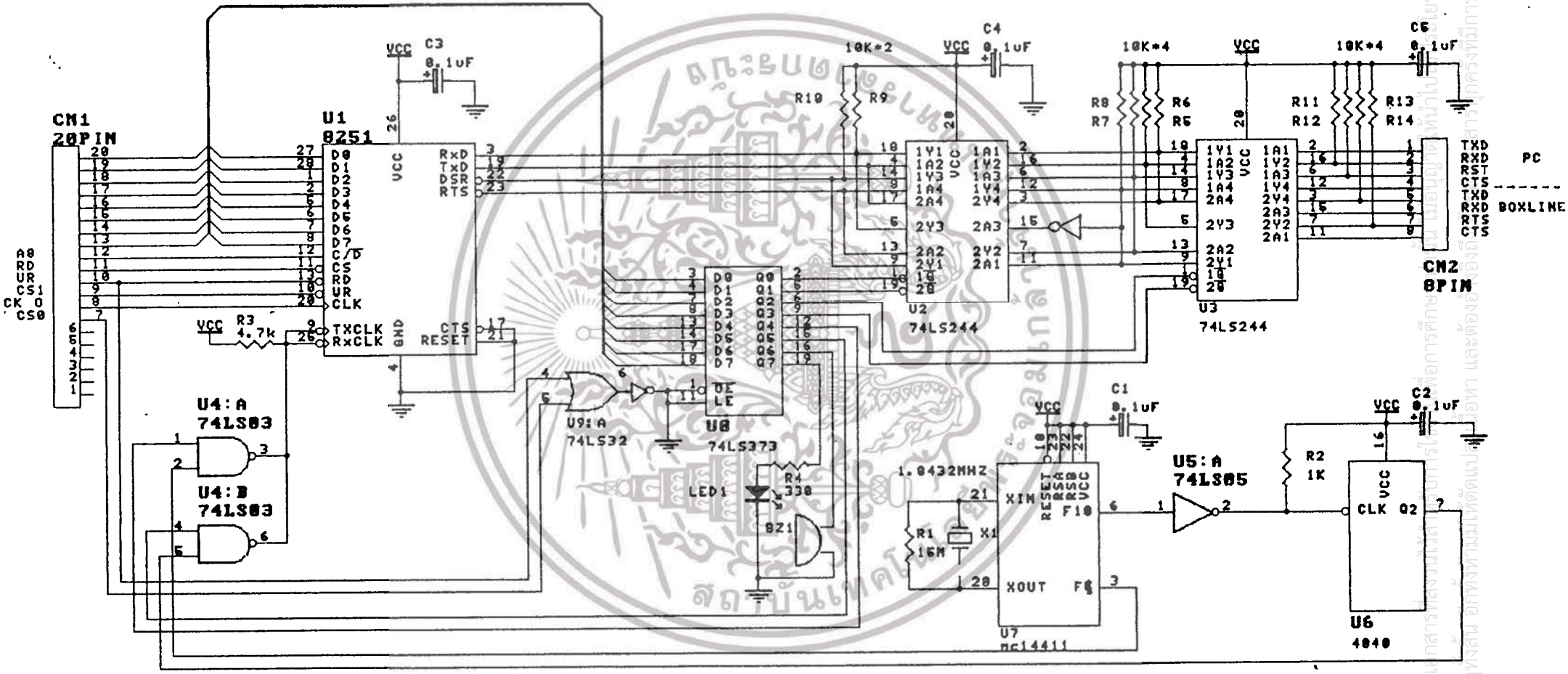
1. Herbert Schildt,"C Power User's Guide"
2. Borland,Turbo C 2.0 Referance And User's Guide
   Borland International,Inc,U.S.A.,1986
3. ยืน ภู่วรวรรณ,วัฒนา เชียงกุล,"ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์"
   ,บริษัทซีเอ็ด ยูเคชั่น จำกัด,พ.ศ. 2521
4. กฤษดา วิศวธีรานนท์,ยืน ภู่วรวรรณ,"ไมโครโปรเซสเวอร์",สมาคมส่งเสริม
   เทคโนโลยี (ไทย-ญี่ปุ่น)
5. ยืน ภู่วรวรรณ,"เทคโนโลยีฮาร์ดแวร์ IBM PC",บริษัทซีเอ็ด ยูเคชั่น จำกัด
   ,พ.ศ. 2521
6. ยืน ภู่วรวรรณ,ดร.ชัยยงค์ วงศ์ชัยสุวัฒน์,ดร.ไพศาล สงวนหมู่,"ไมโครคอม
   พิวเตอร์ 16 บิท",บริษัทซีเอ็ด ยูเคชั่น จำกัด,พ.ศ. 2521
7. ศ.ดร.ไพรัช รัชยพงษ์,"ไมโครคอมพิวเตอร์", บริษัทซีเอ็ด ยูเคชั่น จำกัด,
   พ.ศ. 2521
8. ศิวัฒน์ ศิวะบวร,พรชัย จักรธำรงค์,จิรศักดิ์ ชัยวิริยกุล,"การประยุกต์ใช้งาน
   ภาษาซี"บริษัทซีเอ็ด ยูเคชั่น จำกัด,พ.ศ. 2521

ภาคผนวก ก

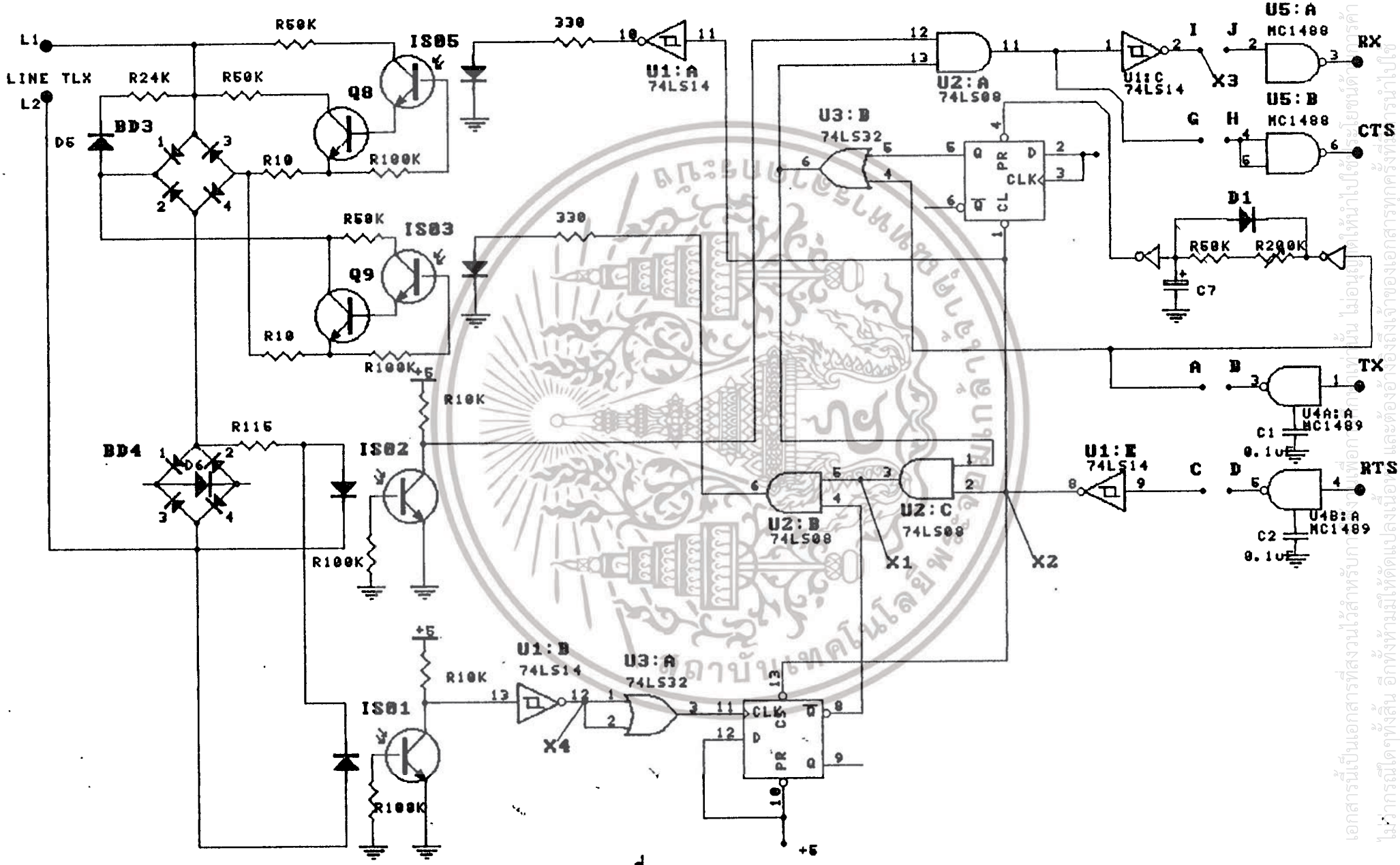
วงจร
ไฟล์วชาร์ท
โปรแกรม



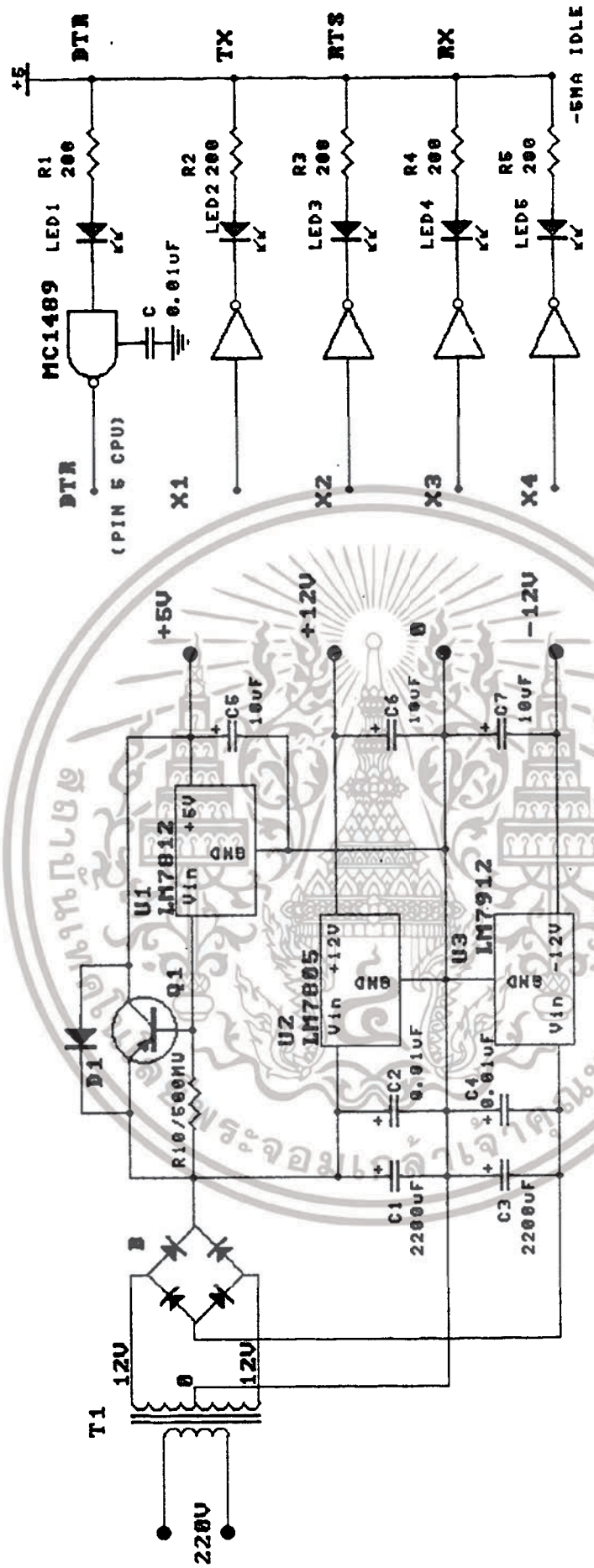
รูปที่ 25 วงจรชุด CONTROL 1



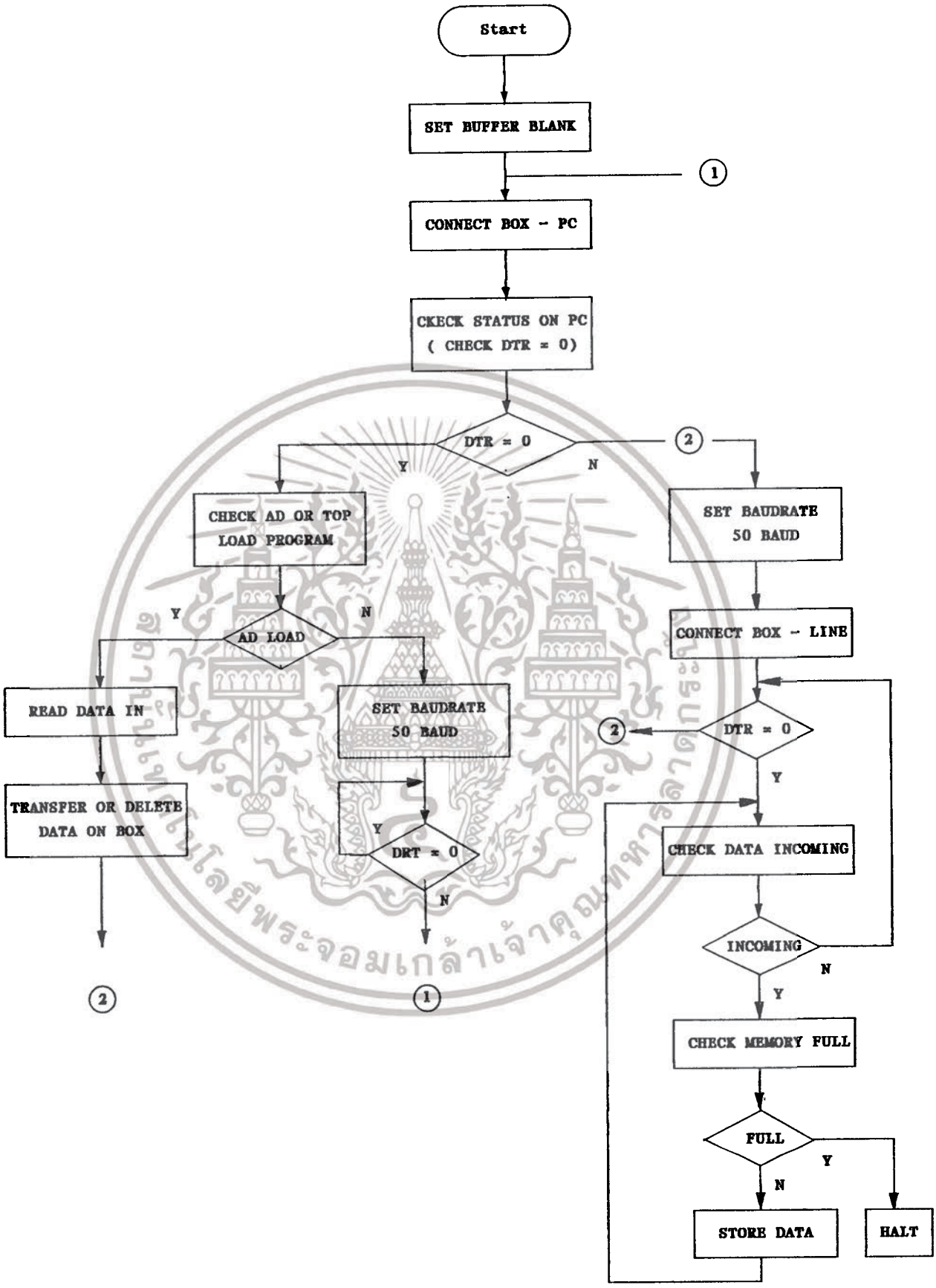
รูปที่26 วงจรชุด CONTROL 2

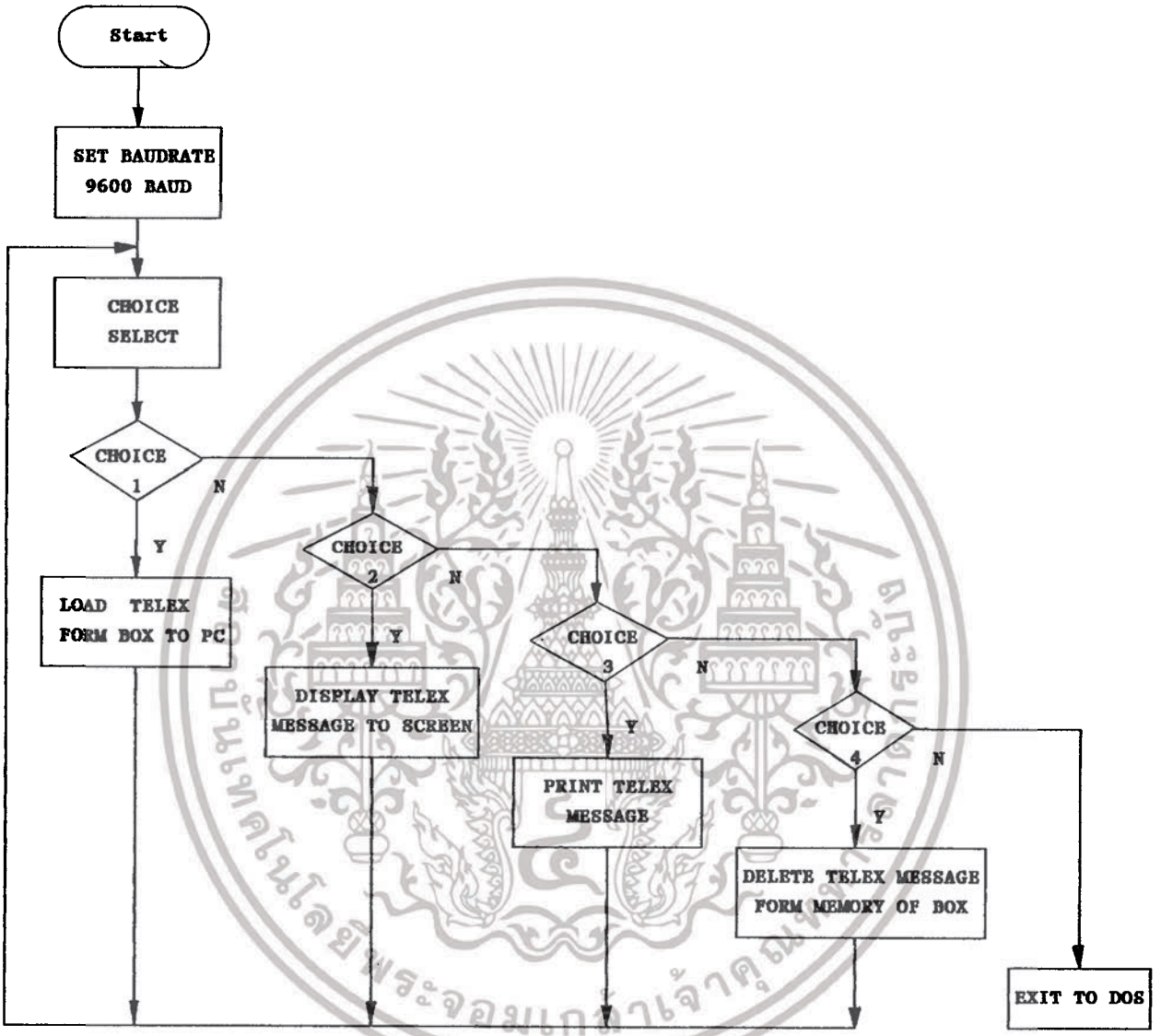


รูปที่27 วงจร INTERFACE



รูปที่28 วงจร POWER SUPPLY





โฟล์วชาร์ทแสดงการทำงานของโปรแกรม AD

โปรแกรม ADVANCE TELEX (ใน ROM)

```
          .8080
          aseg
START     equ     100h          ; address start run program

; DEFIND TELEX STATUS CONSTANT
WRU       equ     9             ; telex : who are you code
FS        equ     11011b        ; shift 1...
LS        equ     11111b        ; shift a...
S_INCM    equ     0a5h          ; status incomming
S_ONLI    equ     082h          ; status online
S_ZERO    equ     00            ; no message in memory
S_FULL    equ     080h
; box send code to pc

SOH       equ     1             ; start of heading
STX       equ     2             ; starrt of message
ETX       equ     3             ; end of message
EOT       equ     4             ; end transfer data box

; hardware box receive code for pc
ENQ       equ     5             ; receive, delete msg on box
ACK       equ     6             ; transfer msg from box to pc
DC1       equ     17            ; set anserback code from pc
DC2       equ     18            ; see anserback code
NAK       equ     21            ; see memory on box
CAN       equ     24            ; bad command
RAM       equ     8*1024        ; ram = 8 KB
PC_BOX    equ     00011010b     ; pc connect box
PC_L      equ     00010011b     ; pc connect line
BOX_LI    equ     00100101b     ; box connect line and bell off
BOX_LO    equ     01100101b     ; box connect line and bell on
MLAMPO    equ     10011010b     ; LED on ,bell off mem good
MLAMP     equ     01011010b     ; led off,bell on  mem bad
```

```
;Command set 8251 to 50 b/s 5 unit 1 start 1.5 stop for TLX MSG

SP50        equ     10000011b     ; set mode word 8251

; Command set 8251 to 9600 b/s 7 unit 1 start 1 stop for PC

SPHI        equ     01001011b     ; set mode word 8251

; command Instruction format for 8251
; D6=IR, D5=RTS, D4=ER ,D3=SBRK ,D2=RxE ,D1=DTR ,D0=TxEN

RTS1        equ     00010101b     ; com.rts=1,rx=tx=en ( use inverse )
RTS0        equ     00110101b     ; com.rts=0,er=reset,rx=tx=en (inverse)
RESET       equ     01000000b     ; reset command 8251
BRKLI       equ     00111101b     ; break line for line cut

; I/O port 8251 (UART), LS373 (switch box_line_pc)

LS373       equ     00h           ; port control box
IN_OUT      equ     08h           ; port in/out data 8251
CTRL        equ     09h           ; port read/write command 8251

; USER address RAM  16 byte

SYS_S       equ     RAM+1         ; store system status
POINTER     equ     SYS_S+1       ; store address memory data 2 byte
RAMMAX      equ     POINTER+2     ; store ram memory max for check. 2 byte
STBL        equ     RAMMAX+2      ; store teble converse code
RXSK        equ     STBL+2        ; test shift key
STACK       equ     RXSK+2        ; store stack pointer
COUNTER     equ     STACK+2       ; use counter
RXRMAX      equ     COUNTER+2     ; compare 1 KB before rammax
MFULL       equ     RXRMAX+2
S_MEM       equ     02050h        ; start memory use streage TLX MSG
```

```
;************************************;
;*          Start program           *;
;************************************;
          org     START             ; start program
          lda     MFULL
          cpi     S_FULL
          jz      fon
          lda     SYS_S             ; load status system for Check
          cpi     S_INCM            ; give incomming,not clear mem
          jz      n_set             ; set new system
          cpi     S_ONLI            ; check reg a = on ,incomming
          jz      nrecd             ; loop online continous
; find max memory ram
          mvi     a,095h            ;
f_ram:    lxi     h,((RAM*3)-1)     ;
          mov     m,a               ; load a-> (hl)
          cmp     m                 ; compare a with (hl)
          jz      s_st              ; set stack pointer
          jmp     fon               ; led on bad memory
; set stack pointer 30 byte
s_st:     shld    STACK             ; save address stack pointer
          sphl                      ; mov hl -> sp
          ora     a                 ; clear cy=0
          mov     a,l
          sui     30
          mov     l,a
          mov     a,h
          sbb     0
          mov     h,a               ; hl is address ram max
          shld    RAMMAX            ; this is ram max
          ora     a                 ; clear cy
          mov     a,h               ; set point end rxrmax mem
          sui     04h               ; 400h = 1024
          mov     h,a
          shld    RXRMAX            ; 1Kb before rammax
```

```
; set memory data start

            lxi     h,S_MEM         ; move start memory to end address
            shld    POINTER         ; pointer of DATA
            mvi     m,SOH           ; start of heading
            inx     h
            shld    POINTER
            mvi     a,S_ZERO
            sta     SYS_S
            sta     MFULL
            call    ledon
            jmp     pcbox

; set box with line and set 8251 speed 50 baud

sbox_1:     lda     MFULL
            cpi     S_FULL
            jz      fon
            call    set51           ; on opto iso5 and off iso2

; Program run loop telex
; check DTR active low ,connect pc with line or box
; D7 = dsr, D5 = fe ,D1 = RxRDY, D0 =TxRDY

ploop:      rim
            ral                     ; shift bit 7 -> cy
            jnc     pcbox           ; check dtr, if cy = 0 goto up speed.

; the DSR pin active lo ,the status bit is '1'.
; check DSR (line is CTS ) active low ,normal rts is '1'

            in      CTRL            ; DSR  bit 7  -> cy
            ral
            jnc     ploop           ; pin DSR = 1,loop
```

```
; read 8251 , RxRDY active low befor goto receive message data
; delay > 250 msec. wait received wru by check fe error

          mvi     a,RTS0          ; on opto iso2 and off is05
          out     CTRL            ; set command word
rdel:     lxi     b,060ffh        ; set time delay
          call    delay1
          jc      rx_d            ; dsr status hi
          mvi     a,RTS1          ; on opto iso5 and off iso2
          out     CTRL            ;
          jmp     ploop           ; loop program

; end program loop

; case 1 ,receive telex
; receive WRU ,give incomming status on

rx_d:     in      CTRL
          ani     00100010b       ; check RxRdy bit 1 with fe
          cpi     00100010b
          jz      rdel
          cpi     00100000b
          jz      rdel
          cpi     00000010b
          jnz     rx_d
          in      IN_OUT
          ani     01fh            ; clear bit 7,6,5
          cpi     FS              ; check  shift key 1... or  a...
          jnz     next
          sta     RXSK            ; stored shift 1...
          jmp     rx_d
next:     cpi     LS
          jnz     next1
          sta     RXSK            ; stored shift A...
          jmp     rx_d
```

```
next1:      cpi     WRU             ; check WHO ARE YOU
            jnz     rx_d            ; rereceive
            lda     RXSK            ;
            cpi     FS              ; check shift key
            jnz     rx_d            ; nz go receive,zero wru ok

; receive wru ok go on bell

            mvi     a,BOX_LO        ; port bell on
            out     LS373
            lhld    POINTER
            call    ch_rxm
            mvi     a,STX
            mov     m,a
            inx     h               ; pointer to start data
            mvi     a,42            ; equ wru ( * ascii = 42 )
            mov     m,a

; check mem full

            inx     h
            shld    POINTER         ; rem next address memory
            mvi     a,S_ONLI        ; set status incomming on
            sta     SYS_S

; send WRU 20 character to line
; hl address mem

; de address answerback code
; b counter 20 cha.

send:       mvi     b,20
            lxi     d,abs           ; de point to addr abs
send1:      in      CTRL
            rar                     ; check TxRdy bit 0 -> cy
```

```
            jnc     send1
            ldax    d               ; (de) to a
            out     IN_OUT
            call    ascii           ; input reg. a
            mov     m,a             ; a to (hl)
            inx     h
            shld    POINTER         ; rem address memory
            inx     d

;check max mem

            dcr     b
            jnz     send1
sen:        in      CTRL
            rar                     ; check TxRdy bit 0 -> cy
            jnc     sen

; off bell
            mvi     a,BOX_LI        ; port belloff
            out     LS373
rec1:       in      CTRL
            ani     10000010b
            ral
            jnc     rec
            rar
            rar
            rar
            jnc     rec1
            in      IN_OUT          ; bit 1 = 1 ,get char.
            mov     c,a             ; store a tlx code
            call    ascii
            mov     m,a
            mov     a,c
            inx     h
            shld    POINTER         ; rem address memory
```

```
; check max mem
            call    ch_m            ; check mem max
            cpi     FS              ; check  shift key 1... or  a...
            jnz     next2
            sta     RXSK            ; stored shift 1...
            jmp     rec1
next2:      cpi     LS
            jnz     next3
            sta     RXSK            ; stored shift A...
            jmp     rec1
next3:      cpi     WRU             ; check WHO ARE YOU
            jnz     rec1            ; rereceive
            lda     RXSK            ;
            cpi     FS              ; check shift key
            jz      send
            jmp     rec1

; the after wru,received msg continous.
rec:        call    delay           ; check fe (check line cut)
            jnz     rec1

; check break line

break:      mvi     m,ETX           ; set end of message
            inx     h
            mvi     m,EOT           ; end of message
            shld    POINTER
            mvi     a,S_INCM        ; no online ,give incomming
            sta     SYS_S

; check max mem
            call    ch_m
            mvi     a,RTS1          ; on opto iso5 and off iso2
            out     CTRL
            jmp     ploop
```

```
; routine delay > 250 msec. wait by check fe error
; return nz ,not pin DSR = 1 and give data in rx.
;         z ,break line.

delay:    lxi    b,080ffh          ;
del1:     xthl                     ; 16
          xthl                     ; 16
          dcr    c                 ; 4   clk =3*10^6
          jnz    del1              ; 7,[256(32+11)]=11008.,t=0.0037
          in     CTRL              ; 0010 0010b
          ani    010000000b
          cpi    000000000b
          rnz
          xthl                     ; 16
          xthl                     ; 16
          dcr    b                 ; 0.4411
          jnz    del1              ; 11,del=[120*[19187] = 2302440
          ret                      ; t=1/f=del/f=0.7499
delay1:   xthl                     ; 16
          xthl                     ; 16
          dcr    c                 ; 4  clk =3*10^6
          jnz    delay1            ; 7,[256(32+11)]=11008.,t=0.0037
          in     CTRL              ; 0010 0010b
          ral
          rc                       ; check Fe bit 5,'1',loop (14+5=19)
          xthl                     ; 16
          xthl                     ; 16
          dcr    b                 ; 0.4411
          jnz    delay1            ; 11,del=[120*[19187] = 2302440
          ret                      ; t=1/f=del/f=0.7499

fon:      mvi    a,MLAMPO
          out    LS373
          rim
          ral
```

```
            jnc     n_set
            jmp     fon

; routine on led

ledon:      mvi     a,MLAMPO
            out     ls373
time:       lxi     b,060ffh
on1:        xthl                ; 16
            xthl                ; 16
            dcr     c           ; 4   0.0037
            jnz     on1         ; 7,[255(64+11)]=19125,t=19125/3*10^6=0.0064
            xthl                ; 16
            xthl                ; 16
            dcr     b           ; 0.4411
            jnz     on1         ; 11,del=[120*[19187] = 2302440
            ret                 ; t=1/f=del/f=0.7499
 routine off led
ledoff:     mvi     a,MLAMP
            out     ls373
            call    time
            ret                 ; t=1/f=del/f=0.7499

; check memory max
; store reg. a , hl
; input reg. hl  address

ch_m:       push    h
            push    psw
            lda     RAMMAX      ; read address ram lo
            cmp     l
            jnz     ch_m2
            lda     RAMMAX+1    ; read address ram hi
            cmp     h           ; (rammax) - hl = 0   ,mem full
            jnz     ch_m2
```

```
; mem full
; set sys halt ,set status ,set 8251 break line cts


            mvi     m,EOT
            pop     psw
            pop     h
            pop     h
            call    sbrk

ch_m2:      pop     psw
            pop     h
            ret

; routine set 8251
; normal speed 50 b/s and connect line - box

set51:      mvi     a,BOX_LI        ; set connect line-box
            out     LS373

; set 8251, 50 b/s ,5 unit ,1 start,1.5 stop,/64,rts=0,cts=1

            mvi     a,RESET         ; internal reset 8251
            out     CTRL            ; to 8251
            mvi     a,SP50          ; set speed 50 baud
            out     CTRL
            mvi     a,RTS1          ; ready or stanby
            out     CTRL
            ret

; set new instrution 8251

n_set:      lhld    STACK
            sphl
            jmp     pcbox
```

```
; user reset system cpu clear new stack pointer
cstack:     lhld    STACK
            sphl
            jmp     ploop

;user reset cpu clear stack and load hl,end_a for receive msg continous
nrecd:      lhld    STACK
            sphl
recd:       lhld    POINTER
            jmp     rec1

; end routine box with line

; check connect pc with box or pc with line
; frist connected ,chage box - line is box - pc

pcbox:      mvi     a,RESET             ; internal reset 8251
            out     CTRL
            mvi     a,PC_BOX
            out     LS373

bench:      mvi     a,SPHI              ; up speed hi
            out     CTRL
            mvi     a,RTS0              ; rts  = 0
            out     CTRL

; check over run error, if error ,to connect box line
; receive code connect from pc

            lxi     h,060ffh            ; time: check loop

;jmp pcb_2  ;test transfer data to box

pbox_:      rim                         ; read DTR active low
            ral                         ; shift bit 7 -> cy
```

```
            jc      sbox_1      ; pin dtr active low,hi go set box line
            in      CTRL
            rar                 ; bit 0 -> cy
            rar                 ; check bit 1 is RxRDY
            jc      box_r1      ; cy '1', go rec. control code
            dcr     l
            jnz     pbox_
            dcr     h
            jz      pc_li       ; go connect box line with top on pc
            jmp     pbox_

;  box check and transfer data to pc
; connection box - pc
; send

pcb_2:      rim                 ; read DTR active low
            ral                 ; shift bit 7 -> cy
            jc      sbox_1      ; cy=1,go set box line
            in      CTRL
            rar
            rar                 ; check bit 1 is RxRDY
            jnc     pcb_2       ; cy '1', go rec. msg

; Box : receive code from pc

box_r1:     in      IN_OUT      ; rec. msg from pc
            cpi     DC1         ; code transfer msg
            jz      trans
            cpi     DC2         ; code delete msg on box
            jz      del

; routine box send bad command to pc
nocom:      mvi     b,CAN       ; send bad command or ready
            call    sendtr
            jmp     pcb_2
```

```
; routine box transfer msg to pc by ascii code.
; used send start mem to end addr
; reg de = start mem for transfer
; reg hl = stored addr counter (mem use)

trans:    lda     SYS_S           ; ld system status to reg a.
          cpi     S_INCM
          jnz     no_msg

; send total mem used to pc
          mvi     b,ACK           ; send ready
          call    sendtr
          lxi     d,S_MEM
send4:    ldax    d               ; move (de) to a
          cpi     00              ; null char not send
          jz      out_
          cpi     EOT
          jz      ccaa
          mov     b,a
          call    sendtr          ; send transfer
ready:    rim                     ; read DTR active low
          ral                     ; shift bit 7 -> cy
          jc      sbox_1          ; cy=1,go set box line
          in      CTRL
          rar
          rar                     ; check bit 1 is RxRDY
          jnc     ready           ; cy '1', go rec. msg
          in      IN_OUT          ; wait
          cpi     ENQ             ; pc send ctrl a for send
          jnz     error1
out_:     inx     d
          jmp     send4
ccaa:     mvi     b,EOT           ; end send code or box ready
          call    sendtr
          jmp     pcb_2
```

```
; routine not incomming in tlx box
no_msg:   mvi   b,NAK          ; no have data in box
          call  sendtr
          jmp   pcb_2
error1:   mvi   b,CAN          ; no have data in box
          call  sendtr
          jmp   pcb_2
; routine converse to ascii
; input data reg a. is baudot code
; out data reg a. is ascii
ascii:    push  h
          push  d
          push  b
          cpi   FS                  ; check fig 1...
          jnz   asc
          lxi   h,tbl1
          shld  STBL
          jmp   ascn1
asc:      cpi   LS                  ; check let a..
          jnz   ascd                ; go check code
          lxi   h,tbla
          shld  STBL
ascn1:    mvi   a,0
          jmp   ascn
ascd:     lhld  STBL                ; load table
          add   l
          mov   l,a
          mvi   a,0
          adc   h
          mov   h,a
          mov   a,m                 ; output  reg a. is ascii
ascn:     pop   b
          pop   d
          pop   h
          ret
```

```
; routine send 1 byte to pc
; input reg. b is data memory.
; store reg. a

sendtr:   rim                    ; check DTR on
          ral
          jc      sbox_1         ; cy = 1 ,exit
send2:    in      CTRL
          rar                    ; check TxRdy = 1
          jnc     send2
          mov     a,b
send3:    out     IN_OUT         ; send msg
          ret


; routine delete msg all in box or set system memory

del:      lxi     h,S_MEM+1
          shld    POINTER        ; clear pointer memory
          mvi     m,EOT
          lhld    STACK
          sphl
          mvi     a,S_ZERO
          sta     SYS_S          ; system clear no message memory
          mvi     a,015h
          sta     MFULL
          mvi     b,NAK          ; end send code and box ready
          call    sendtr
          jmp     pcb_2
```

```
; converse code hex to ascii
; input reg. a is hex
; store reg. hl,bc,de
; out reg.   hl
; not clear
; case 4 : set connect pc with line tlx

pc_li:      mvi     a,RESET
            out     CTRL
            mvi     a,PC_L      ; change sw line
            out     LS373
pcli1:      rim                 ; read DTR active low loop
            ral                 ; shift bit 7 -> cy
            jc      pclo1       ; check cut dtr loop, pass to set box-pc
            jmp     pcli1       ; cy = 1 exit to box received tlx data

;add 14/2/91 delay dtr off line when the pc sending the wru to line.

pclo1:      lxi     d,08ffh     ; 15 sec delay
pclo:       call    tim
            xthl                ; 16
            xthl                ; 16
            dcr     e           ; 4  clk =3*10^6
            jnz     pclo        ; 7,[256(32+11)]=11008.,t=0.0037
            rim
            ral
            jnc     pcli1
            xthl                ; 16
            xthl                ; 16
            dcr     d           ; 0.4411
            jnz     pclo        ; 11,del=[120*[19187] = 2302440
            jmp     pcbox
```

```
;delay for time sent AAb for use top

tim:    lxi   b,001ffh    ;
on2:    xthl              ; 16
        xthl              ; 16
        dcr   c           ; 4  0.0037
        jnz   on2         ; 7,[255(64+11)]=19125,t=19125/3*10^6=0.0064
        xthl              ; 16
        xthl              ; 16
        dcr   b           ; 0.4411
        jnz   on2         ; 11,del=[120*[19187] = 2302440
        ret               ; t=1/f=del/f=0.7499

;routine check mem 1 KB before rammax

ch_rxm: push  h
        push  psw
        lhld  RXRMAX
        lda   POINTER+1   ; read address ram hi
        cmp   h           ; (rammax) - hl = 0  ,mem full
        jc    rxmax
        pop   psw
        pop   h
        pop   h
        call  sbrk
rxmax:  pop   psw
        pop   h
        ret

sbrk:   lhld  RAMMAX
        shld  POINTER
        mvi   a,S_INCM
        sta   SYS_S
        mvi   a,S_FULL
        sta   MFULL
```

```
        mvi     a,BRKLI
        out     CTRL
        call    time
        call    time
        call    time
        call    time
        call    time
        call    time
        call    set51
        pop     h               ; clear call
        hlt

; define abs for test receive msg to mem boxs.

abs:    db      011011b         ; 1...  1
        db      001000b         ; <     2
        db      000010b         ; =     3
        db      000110b         ; 8     4
        db      010110b         ; 0     5
        db      000110b         ; 8     6
        db      010110b         ; 0     7
        db      000111b         ; 7     8
        db      011111b         ; A...  9
        db      000100b         ; sp    10
        db      001110b         ; C     11
        db      000011b         ; A     12
        db      010000b         ; T     13
        db      000001b         ; E     14
        db      010010b         ; L     15
        db      000001b         ; E     16
        db      011101b         ; X     17
        db      000100b         ; sp    18
        db      010000b         ; T     19
        db      010100b         ; H     20
```

```
; table baudot code to ascii

tbla:     db      00,69,10,65,32    ; shift a...
          db      83,73,85,13,68
          db      82,74,78,70,67
          db      75,84,90,76,87
          db      72,89,80,81,79
          db      66,71,00,77,88
          db      86,00
tbl1:     db      00,51,10,45,32    ; shift 1...
          db      39,56,55,13,42
          db      52,07,44,00,58
          db      40,53,43,41,50
          db      00,54,48,49,57
          db      63,00,00,46,47
          db      61,00
          end
```

โปรแกรม ADVANCE TELEX บนเครื่องไมโครคอมพิวเตอร์

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#define  norm_vid 23
#define  rev_vid 7
#define  menu_vid 112
#define  high_vid 30
#define  ESC 27
#define  BORDER 1
#define  com_port 0x3f8
#define  ABORT 0
#define TIME_OUT_BIT    0x01
#define IO_ERROR_BIT    0x08
#define SELECTED_BIT    0x10
#define PAPER_OUT_BIT   0x20
#define ACKNOWLEDGE_BIT 0x40
#define NOT_BUSY_BIT    0x80
#define ERROR_BIT      (TIME_OUT_BIT | IO_ERROR_BIT | PAPER_OUT_BIT)
#define PRINTER_OFF    (SELECTED_BIT | ACKNOWLEDGE_BIT)
#define NOT_ON_LINE    (SELECTED_BIT | IO_ERROR_BIT)
#define OUT_OF_PAPER   (SELECTED_BIT | PAPER_OUT_BIT)
#define PRINTER_BUSY   (NOT_BUSY_BIT | PAPER_OUT_BIT)
#define PORT_LPT1      0
#define FUNCTION_OO    0
#define MIN_TIME_OUT   1
#define TIME_OUT_ADDR 0x00400078
unsigned char far *time_out_pointer = (unsigned char far *)TIME_OUT_ADDR;
char *error_type[5] = {"Printer switch is off.",
                       "Printer is out off paper.",
                       "Printer is off line and cannot receive data.",
                       "Printer is bust or Printer queue is not empty.",
                       "Printer is not ready."};
union REGS r;
```

```
int data,mem,data1;                    /* exiternal data          */
char start_mem;                        /* stored address memory   */
int SOH = 01;                          /* start of heading        */
int STX = 02;                          /* start of text           */
int ETX = 03;                          /* end of text             */
int EOT = 04;                          /* end of transmission     */
int ENQ = 05;                          /* enquiry or ready        */
int ACK = 06;                          /* acknowledge             */
int DC1 = 17;                          /* display,save,print msg  */
int DC2 = 18;                          /* del msg on box          */
int NAK = 21;                          /* no acknowledge,no msg   */
int CAN = 24;                          /* error transmission msg  */
int i    = 1;
int ok   = 1;
int pass = 1;
char key;
FILE *tlxfile,*tlxfile1,*file,*fopen();

int j,x,y,attri,startx,starty,endx,endy;
char p,c,head_name[78],file_name[13];

char *me[]={
        " Load TLX message to PC ",
        " View TLX message       ",
        " Print TLX message      ",
        " Delete message on box  ",
        " Exit to dos            ",
        };
char *men[]={"L","V","P","D","E",};
char *rem[]={"Transfer all TLX. message from BOX to PC          ",
                        "View all TLX. message on screen        ",
                        "Printing all TLX. message              ",
                        "Delete all TLX. message on box         ",
                        "Exit ATAT program to dos               ",
                       };
```

```
/*************************************/
/*            pop_up menu            */
/*************************************/

int popup(menu,keys,count,x,y,border)
char *menu[];                  /* menu text                          */
char *keys;                    /* hot keys                           */
int  count;                    /* number of menu items               */
int  x,y;                      /* x,y coordinate of left hand corner */
int  border;                   /* no border if 0                     */

{
  register int i,len=0;
  int endx,endy,choice;
  unsigned int *f;
  for(i=0;i<count;i++)
  if(strlen(menu[i]) > len)
  len= strlen(menu[i]);
  endx = len + 1 + x;
  endy = count + 1 + y;

 /* allocte enough memory for it */
 f=(unsigned int *) malloc((endx-x+1) * (endy-y+1));
 if(!f)exit(1);          /*  install your own error handler here */

 do
    {
         /* save the current screen data  */
         save_video(x,endx+1,y,endy+1,f);
         if(border)
         draw_border(x,y,endx,endy);

         /* display the menu  */
         display_menu(menu,x+1,y+1,count);
         choice=get_resp(x+1,y,count,menu,keys);
```

```
        switch (choice) {
                        case 0:
                                {
                                    menu0();
                                    break;
                                }
                        case 1:
                                {
                                    menu1();
                                    break;
                                }
                        case 2:
                                {
                                    menu2();
                                    break;
                                }
                        case 3:
                                {
                                    menu3();
                                    break;
                                }
                        case 4:
                                {
                                    pass=0;
                                    break;
                                }
                    }
        /*  restore the original screen  */
        if(pass!=0)restore_video(x,endx+2,y,endy+2, (char *) f);
        free(f);
       }
 while(pass);

}
```

```
/*************************************/
/*              draw border          */
/*************************************/
int draw_border(startx,starty,endx,endy)
int startx,starty,endx,endy;
{ unsigned int i;
  for(i=startx+1;i<endx;i++)
        {
          write_vid(i,starty,196,menu_vid);
          write_vid(i,endy,196,menu_vid);
        }
  for(i=starty+1; i<endy;i++)
        {
          write_vid(startx,i,179,menu_vid);
          write_vid(endx,i,179,menu_vid);
        }
  write_vid(startx,starty,218,menu_vid);
  write_vid(startx,endy,192,menu_vid);
  write_vid(endx,starty,191,menu_vid);
  write_vid(endx,endy,217,menu_vid);
}

/*************************************/
/*        input selection key        */
/*************************************/
get_resp(x,y,count,menu,keys)
int x,y,count;
char *menu[];
char *keys;
{
  union inkey
        {
          char  ch[2];
          int i;
        }c;
```

```
int arrow_choice = 0 ,key_choice;
y++;
/*  highlight the first selection  */
write_video(x,y,menu[0],rev_vid);      /*  reverse video  */
write_video(2,23,rem[0],norm_vid);
goto_xy(0,0);no_cursor();
for(;;)
   {
     while(!bioskey(1));              /* WAIT FOR KEY STROKE */
     c.i=bioskey(0);                  /* read the key  */
     /*  reset the selection to normal video  */
     goto_xy(x,y+arrow_choice);
     write_video(x,y+arrow_choice,
     menu[arrow_choice],menu_vid);          /* redisplay  */
     strcpy(head_name,men[arrow_choice]);
     write_str(x+1,y+arrow_choice,118);
     if(c.ch[0])
      {
       /* is normal key   */
       /*   see if it is a hot key  */
       key_choice = is_in(keys, tolower(c.ch[0]));
       if(key_choice) return key_choice-1;
       /*  check for enter or space bar  */
       switch(c.ch[0] )
                {
                  case '\r': return arrow_choice;
                  case ' ' : arrow_choice++;
                             break;
                  case ESC : printf("\n");
                             /* pin rts ,dtr to lo */
                             outportb(com_port+4,0x0);
                             /* pin rts ,dtr to lo */
                             outportb(com_port+4,0x0);
                             clrs();
                             have_cursor();
```

```
                                        clrs();goto_xy(0,0);
                                        exit(0);
                                }
                        }
                        else
                           {
                              /* is special key */
                               switch(c.ch[1])
                                     {
                                          case 72: arrow_choice--;  /* up arrow  */
                                                   break;
                                          case 80: arrow_choice++;  /* down arrow  */
                                                   break;
                                     }
                           }
                        if(arrow_choice == count) arrow_choice=0;
                        if(arrow_choice < 0) arrow_choice=count-1;
                        /* highlight the next selection  */
                        goto_xy(x,y+arrow_choice);
                        write_video(x,y+arrow_choice,menu[arrow_choice],rev_vid);
                        write_video(2,23,rem[arrow_choice],norm_vid);
                        goto_xy(0,0);no_cursor();
                }
}

/*************************************/
/*          display string           */
/*************************************/
write_video(y,x,f,attrib)
char *f;
int x,y,attrib;
{
  union REGS r;
  register int i,j;
```

```
/*************************************/
/*           restore screen          */
/*************************************/

int restore_video(startx,endx,starty,endy,buf_ptr)
int startx,endx,starty,endy;
unsigned int *buf_ptr;
{
  union REGS r;
  register int i,j;
  for(i=starty;i<endy;i++)
        for(j=startx;j<endx;j++)
           {
             goto_xy(j,i);
             r.h.ah = 9;              /* write character function  */
             r.h.bh = 0 ;             /* assume active display page is 0  */
             r.x.cx = 1;              /* number of time to write the character*/
             r.h.al = *buf_ptr++;     /* character  */
             r.h.bl = *buf_ptr++;     /* attribute  */
             int86(0x10,&r,&r);
           }
}


/*************************************/
/*                is_in              */
/*************************************/

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

```
/*************************************/
/*                break              */
/*************************************/

int c_break(void)
{
    printf("USER Break by CTRL C   aborting ...\n");
    return(ABORT);
}

/*************************************/
/*             sent byte             */
/*     send data 1 byte is ascii     */
/*************************************/

send_byte(data)
{
    outportb(com_port,data);          /* pin rts ,dtr to hi          */
}

/*************************************/
/*            receive byte           */
/*       receive 1 byte ascii        */
/*************************************/

rec_byte()                            /* receive 1 byte ascii        */
{
  int tlx_sts;
  tlx_sts = inportb(com_port+5);      /*   read status               */
  while ((tlx_sts & 0x1) ==0)
                {
                  tlx_sts = inportb(com_port+5); /* read status              */
                }
  data = inportb(com_port);           /*   read data                 */
 }
```

```
/*************************************/
/*              any keys             */
/*************************************/
anykey()
{
  outportb(com_port+4,0x0);          /* pin dtr ,rts to lo */
  do
        {
          goto_xy(24,24);
          printf("\007press SPACE BAR for countinous");
        }
  while(getch() != ' ');
}

int get_time_out(int port)
  {   return(time_out_pointer[port]); }
void set_time_out(int value,int port)
  {   time_out_pointer[port] = value; }
int define_error(int status)
  {   int tem = status;
      if ((tem &= PRINTER_OFF) == PRINTER_OFF) return 0;
      else {
            tem = status;
            if ((tem &= OUT_OF_PAPER) == OUT_OF_PAPER) return 1;
            else {
                  tem = status;
                  if (( tem &= NOT_ON_LINE) ==NOT_ON_LINE) return 2;
                  else {
                        tem = status;
                        if ((tem &= PRINTER_BUSY) == PRINTER_BUSY) return 3;
                       else return 4;
                       }
                 }
          }
  }
```

```
int print_ch(char ch,int port)
{
    unsigned status;
    char ans;
    status = biosprint(FUNCTION_OO,ch,port);
        if (status & ERROR_BIT)
           {
             goto_xy(0,0);have_cursor();cls();
             printf("\007ERROR: %s\n",error_type[define_error(status)]);
             return 1;
           }
        else
            return 0;
}

void print()
{
    FILE *file;
    int old_value,x,ch;
    char answer;
    if ((file = fopen(file_name, "rt")) == NULL) {
       goto_xy(0,0);
       printf("Select Load TLX message before printing\n");
    }
    else {
            old_value = get_time_out(PORT_LPT1);
            set_time_out(MIN_TIME_OUT,PORT_LPT1);
            do {
                 ch = fgetc(file);
                 if (ch != EOF)
                    {
                       if (print_ch(ch,PORT_LPT1))
                          {
                             printf("\007Retry or not.? [r/n]");
                             answer = getch();
```

```
                    if ( answer != 'r')
                       ch = EOF;
                    else
                        {
                           x = ftell(file)-1;
                           fseek(file,x,SEEK_SET);
                        }
                           cls();
                  }
               }
            } while (ch != EOF);
              fclose(file);
              set_time_out(old_value,PORT_LPT1);
        }
}

/*************************************/
/*              menu 0               */
/*************************************/
menu0()  /*  save in file tlxbox.rec  */
{
  i = 1;
  ok= 1;
  cls();
  file = fopen("tlxbox.box","rt");
  if(file != NULL)
    {
       rename("tlxbox.box","tlxbox.old");
       fclose (file);
    }
    file = fopen("tlxbox.rec","rt");
    if(file != NULL){
       rename("tlxbox.rec","tlxbox.bak");
       fclose (file);
       }
```

```
goto_xy(1,1);
outportb(com_port+4,0x3);              /* pin rts ,dtr to hi */
delay(3);
send_byte(DC1);                        /* pin rts ,dtr to hi */
rec_byte();
if (data==ACK)
   {
     if ((tlxfile=fopen("tlxbox.rec","w")) == NULL){
        printf("Cannot open TLXBOX.REC file");
        anykey();main(); }
        if ((tlxfile1=fopen("tlxbox.box","w")) == NULL){
           printf("Cannot open TLXBOX.BOX file");
           anykey();main();
           }
           printf("\nSAVE files name TLXBOX.REC\n");
           do{
              rec_byte();
              putc(data,tlxfile1);
                 switch (data) {
                    case 01: break;          /*  SOH  */
                    case 02: printf("\nTLX MSG. NO: %d\n",i);
                             fprintf(tlxfile,"\nTLX MSG. NO: %d\n\n\n\n\n",i);
                             break;
                    case 03: fprintf(tlxfile,"\n\n\n\nEND TLX MSG. NO: %d\n\n\n\n\n",i);
                             i++;
                             break;
                   case 04: ok=0;--i;
                            fprintf(tlxfile,"\n\nTLX MSG. COMPLETE TOTAL => %d MSG\n",i);
                            printf("\nTLX MSG. COMPLETE TOTAL => %d MSG\n",i);
                            break;
                   case 24: ok=0;
                            fprintf(tlxfile,"\n ERROR DATA TRANSMITTER\n");
                            break;
                            default: putc(data,tlxfile);
                            }
```

```
            if (ok==1) send_byte(ENQ);        /*pin rts ,dtr to hi*/
            } while (ok);
            fclose(tlxfile);
            fclose(tlxfile1);
    }
    else if (data==NAK)
            printf("\n NO DATA MEMORY ON BOX \n");
    anykey();
    main();
}
/*************************************/
/*          menu1()                  */
/*************************************/
menu1()
{
   cls();
   main1();
}
/*************************************/
/*          menu2()                  */
/*************************************/
menu2()
{
   cls();
   get_view();
}
/*************************************/
/*              menu 3               */
/*************************************/
menu3()
{
  char opt,opt1;
  cls();
  goto_xy(0,0);
  have_cursor();
```

```
  printf("\nWARNING! TLX message on box will be delete\n");
  printf("\nDo you wish to delete (Y/N): ");
  opt = getch();
  if ( opt == 'Y' || opt == 'y')
     {
        printf("\n\nAre you sure! (Y/N): ");
        opt1 = getch();
        if (opt1 == 'Y' || opt1 == 'y')
           {
              outportb(com_port+4,0x3);   /* pin rts ,dtr to hi*/
              delay(2);
              send_byte(DC2);
              rec_byte();
              if (data==NAK)
                 {
                    cls();
                    goto_xy(1,1);
                    printf("\n DELETE ALL MESSAGE ON BOX !!!");
                    delay(100);
                    printf("\b\b\b OK !\n");
                 }
           }
     }
     main();
}

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

```
/*  Function Clear Screen */
cls()
{
  union REGS r;
  r.h.ah = 6;
  r.h.al = 0;
  r.h.ch = 0;
  r.h.cl = 0;
  r.h.dh = 24;
  r.h.dl = 79;
  r.h.bh = 31;
  int86(0x10,&r,&r);
}

/* Draw Character & Border */
write_vid(i,j,p,attri)
{
  union REGS r;
  goto_xy(i,j);
  r.h.ah = 9;
  r.h.bh = 0;
  r.x.cx = 1;
  r.h.al = p;
  r.h.bl = attri;
  int86(0x10,&r,&r);
}

/* Function nocursor */
no_cursor()
{ union REGS r;
  r.h.ah = 1;
  r.h.ch = 0;
  r.h.cl = 0;
  int86(0x10,&r,&r);
}
```

```
getwin(startx,starty,endx,endy)
{ for(i=startx;i<endx;i++)
     { write_vid(i,starty,205,norm_vid); write_vid(i,endy,205,norm_vid);}
  for(j=starty;j<endy;j++)
     { write_vid(startx,j,186,norm_vid); write_vid(endx,j,186,norm_vid);}
  write_vid(startx,starty,201,norm_vid);write_vid(endx,starty,187,norm_vid);
  write_vid(startx,endy,200,norm_vid);write_vid(endx,endy,188,norm_vid);
  goto_xy(0,0);no_cursor();
}

write_str(startx,starty,attri)
{ j = strlen(head_name);
  for(i=0;i<j;i++){p = head_name[i];write_vid(startx+i,starty,p,attri);}
}

/*************************************/
/*          display menu             */
/*************************************/
display_menu(menu,x,y,count)
char *menu[];
int x,y,count;
{  int z=y;
   register int i;
   for(i=0;i<count; i++,y++)
         {
               /*goto_xy(x,y);printf(menu[i]);*/
               strcpy(head_name,menu[i]);
               write_str(x,y,menu_vid);
         }
  for(i=0;i<count;i++,z++)
         {
            strcpy(head_name,men[i]);
            write_str(x+1,z,118);
         }
}
```

```
clrs()
{
  union REGS r;
  r.h.ah = 6;
  r.h.al = 0;
  r.h.ch = 0;
  r.h.cl = 0;
  r.h.dh = 24;
  r.h.dl = 79;
  r.h.bh = 7;
  int86(0x10,&r,&r);
}

have_cursor()
{ union REGS r;
  r.h.ah = 1;
  r.h.ch = 12;
  r.h.cl = 13;
  int86(0x10,&r,&r);
}

#include <process.h>
get_view()
{
   char choice,value,value1;
   cls();goto_xy(0,0);have_cursor();
   printf("1: All file.\n");
   printf("2: Some file.\n");
   printf("3: Interval.\n");
   printf("4: Return to main menu.\n");
   printf("Please select option : ");
   scanf("%d",&choice);
   switch(choice){
         case  1 : strcpy(file_name,"tlxbox.rec");
                   print();
```

```
                break;
        case  2 : printf("\nSelect TLX MSG. NO.  ");
                scanf("%d",&value);cls();goto_xy(0,0);
                selected(value,value);
                strcpy(file_name,"tlxbox.tmp");
                print();
                break;
        case  3 : cls();goto_xy(0,0);
                printf("\nSelect TLX MSG. NO.  ");
                scanf("%d",&value);goto_xy(23,1);
                printf(" to NO.  ");
                scanf("%d",&value1);
                selected(value,value1);
                strcpy(file_name,"tlxbox.tmp");
                print();
                break;
        case  4 : break;
                }
   main();
}

selected(int x,int y)

{
   int i=0,j=0,k=1;
   char data;
   if((tlxfile=fopen("tlxbox.box","r"))==NULL){
       cls();goto_xy(0,0);
       printf("Please select Load TLX MSG.\n");
       anykey();main();}
   if((tlxfile1=fopen("tlxbox.tmp","w"))==NULL){
       cls();goto_xy(0,0);
       printf("Not enough memory\n");
       anykey();main();}
       fprintf(tlxfile1,"\n\n\n");
```

```
        do {
            data = getc(tlxfile);
            switch(data){
                  case 01: break;
                  case 02: if ( i >= x ) fprintf(tlxfile1,"\n\n\n\n\n\n");
                           i++;
                           break;
                  case 03: if ( i >= x ) fprintf(tlxfile1,"\n\n\n");
                           j++; if( j == y) k=0;
                           break;
                  case 04: k=0; break;
                  case 24: k=0; break;
                           default: if( i >= x)
                           putc(data,tlxfile1);
                           if( j == y)  k=0;
                           break;
                     }
            } while (k);
            fprintf(tlxfile1,"\n\n\n");
            fclose(tlxfile);
            fclose(tlxfile1);
}

main1()
{
        int stat;
        goto_xy(0,0);
        if((file=fopen("temp","rt"))==NULL){
        file=fopen("temp","w");
        fclose(file);}
        execl("wpview.exe","wpview.exe","tlxbox.rec",NULL);{
        printf("\n\t\tVIEW FILE ERROR \n");
        exit(0);}
        getch();
}
```

```
main()
{
  int i;
  unsigned int *p;
  cls();
  ctrlbrk(c_break);              /* initial com port                */
  outportb(com_port+4,0x0);      /* pin rts ,dtr to lo              */
  delay(5);
  outportb(com_port+3,0x80);     /* set DLAB = 1  at 3fbh           */
  outportb(com_port,0x0C);       /* set boud rate lsb               */
  outportb(com_port+1,0x0);      /* set ========= msb               */
  outportb(com_port+3,0x02);     /* set baud rate 1 st 7 u 9600 np */

  for(i=0;i<79;i++)
     {
       write_vid(i,0,205,norm_vid); write_vid(i,24,205,norm_vid);
       write_vid(i,22,205,norm_vid);
     }
       strcpy(head_name," Comment ");
       write_str(35,22,norm_vid);
       for(j=0;j<24;j++)
           {
             write_vid(0,j,186,norm_vid); write_vid(79,j,186,norm_vid);
           }
             write_vid(0,0,201,norm_vid); write_vid(79,0,187,norm_vid);
             write_vid(0,24,200,norm_vid);write_vid(79,24,188,norm_vid);
             write_vid(0,22,204,norm_vid);write_vid(79,22,185,norm_vid);
             getwin(20,8,59,18);
             strcpy(head_name,"A D V A N C E   T E L E X   ON   PC");
             write_str(23,3,high_vid);
             strcpy(head_name,"KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG");
             write_str(16,6,high_vid);
             strcpy(head_name,"TELEX SUBSCRIBER OFFICE.  TELEX DIVISION");
             write_str(18,20,high_vid);
             popup(me,"1vpde",5,27,10,BORDER);
```

```
        printf("\n");
        outportb(com_port+4,0x0);       /* pin rts ,dtr to lo */
        outportb(com_port+4,0x0);       /* pin rts ,dtr to lo */
        clrs();
        have_cursor();
        clrs();goto_xy(0,0);
        exit(0);

}
```

ภาคผนวก ซ

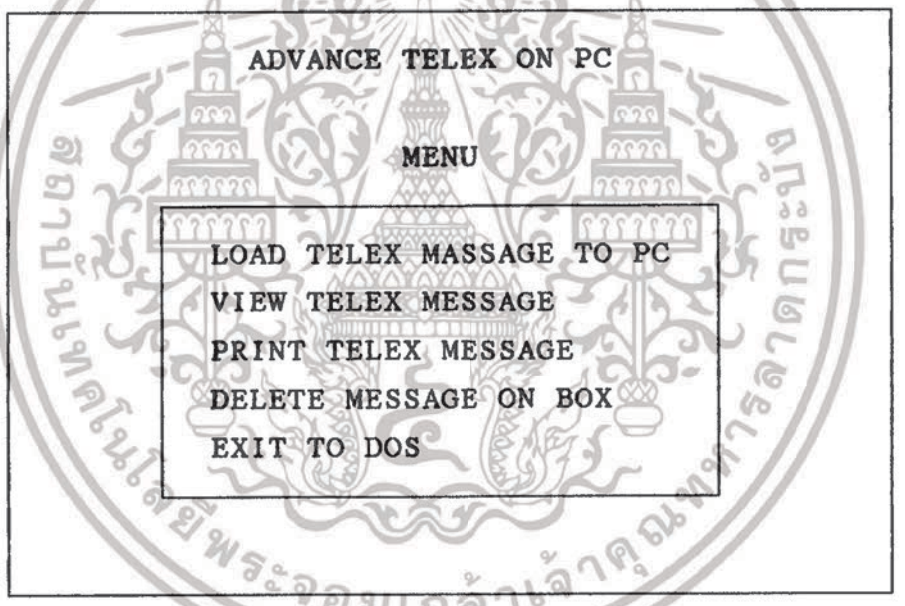
การใช้งาน ADVANCE TELEX
การใช้งานโปรแกรม TOP

การใช้โปรแกรม ADVANCE TELEX ON PC

จากประโยชน์ของชุด ADVANCE TELEX ON PC จะเห็นว่า เราไม่จำเป็นต้องเปิดเครื่องไมโครคอมพิวเตอร์ตลอดเวลา โดยชุด ADVANCE TELEX ON PC จะทำหน้าที่รับข้อความเทเล็กซ์ที่เข้ามาไว้ทั้งหมด ถ้าต้องการดูข้อมูลก็สามารถเรียกมาดูได้ด้วย Software ของ ADVANCE TELEX ON PC ซึ่งโปรแกรมนี้จะทำงานบนเครื่องไมโครคอมพิวเตอร์ ดังต่อไปนี้

การเข้าสู่โปรแกรม

เมื่อผู้ใช้ RUN PROGRAM ADVANCE TELEX ON PC ภาพที่ปรากฏบนจอภาพจะเป็นดังรูปที่ 1



รูปที่ 1 MAIN MENU

จาก MENU ประกอบด้วยโหมดต่าง ๆ 5 โหมด ดังนี้

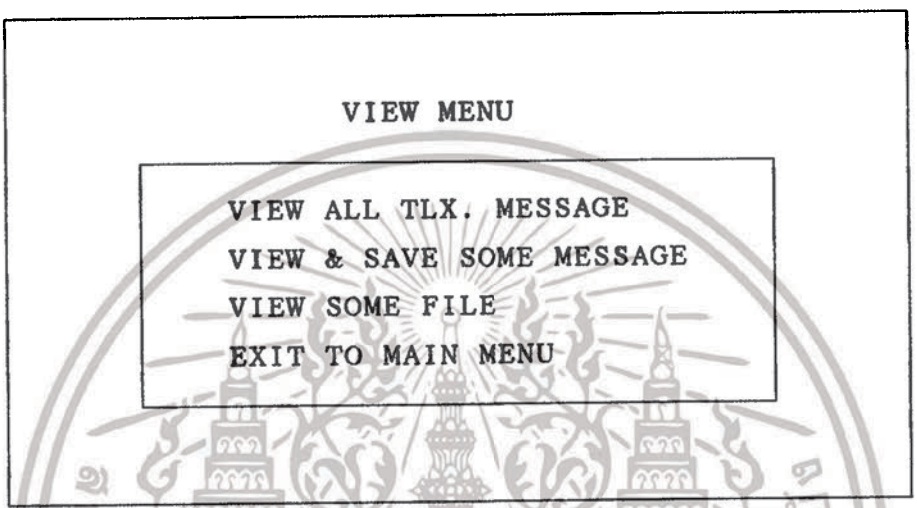
1. LOAD TELEX MASSAGE TO PC
2. VIEW TELEX MESSAGE
3. PRINT TELEX MESSAGE
4. DELETE MESSAGE ON BOX
5. EXIT TO DOS

การเลือกใช้งานให้เลื่อน High Light Bar โดยการกดลูกศรขึ้นหรือลง หรือ Space Bar มายังตำแหน่งที่ต้องการ แล้วกด ENTER

รายละเอียดของโหมดต่าง ๆ

1. LOAD TELEX MESSAGE ON PC ในโหมดนี้เป็นการ LOAD ข้อความจาก MEMORY ของชุด ADVANCE TELEX ON PC ทั้งหมดมาไว้บนเครื่องไมโครคอมพิวเตอร์

2. VIEW TLX MESSAGE จะปรากฏ Menu ดังรูปที่ 2



รูปที่ 2 แสดง VIEW MWNU

VIEW ALL TLX. MESSAGE เป็นการเรียกดูข้อความเทเล็กซ์ ทั้งหมดที่ LOAD มาบนเครื่องไมโครคอมพิวเตอร์

VIEW & SAVE SOME MESSAGE เป็นการเรียกข้อความเทเล็กซ์มาดูเหมือน MENU VIEW ALL TLX. MESSAGE แต่ข้อความใน MENU นี้สามารถเจาะจงข้อความได้

ตัวอย่าง เมื่อทำการเลือก MENU VIEW & SAVE SOME MESSAGE จะปรากฏข้อความบนจอภาพ

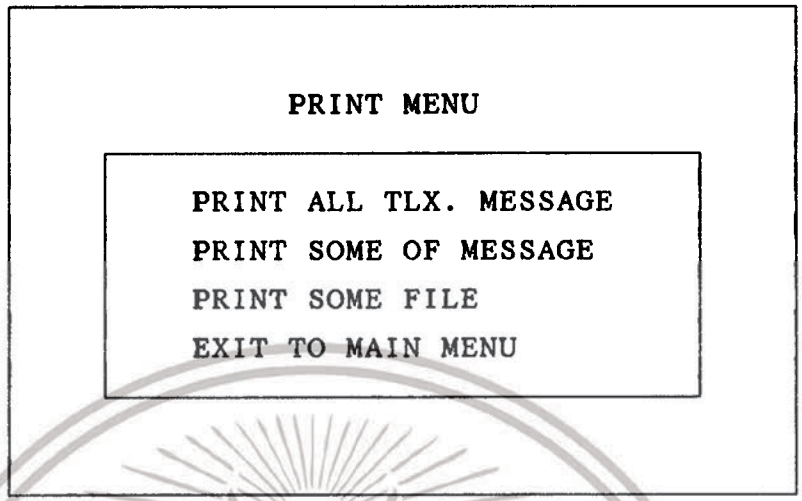
TLX. MESSAGE NO.

ให้ใส่หมายเลขลงไป จะปรากฏข้อความเทเล็กซ์ตามต้องการ และถ้าต้องการเก็บข้อความส่วนนี้ไว้ก็ให้ตอบ "Y" (โดยจะขึ้นข้อความ DO YOU WANT TO SAVE [Y/N] :) พร้อมกับทำการตั้งชื่อ

VIEW SOME FILE เป็นการแสดงข้อความใน FILE ที่ถูกเก็บไว้โดยให้ใส่ชื่อ FILE ตามที่ต้องการ

EXIT TO MAIN MENU เป็นการออกจาก VIEW MENU สู่ MAIN MENU

3. PRINT TLX. MESSAGE จะมี MENU ให้เลือก 4 MENU ดังรูปที่ 3



รูปที่ 3 แสดง PRINT MENU

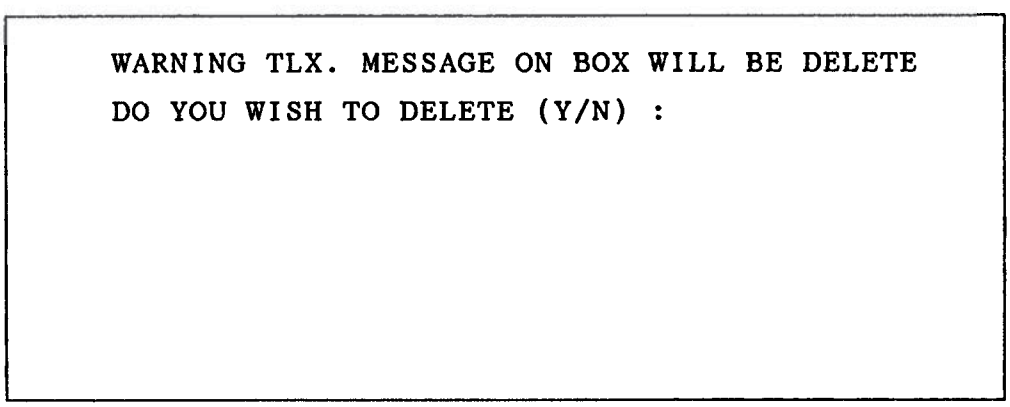
PRINT ALL TLX. MESSAGE เป็นการพิมพ์ข้อความเทเล็กซ์ทั้งหมดออกทางเครื่องพิมพ์

PRINT SOME OF MESSAGE เป็นการพิมพ์ข้อความเทเล็กซ์โดยสามารถเจาะจงข้อความได้

PRINT SOME FILE เป็นการทิมพ์ข้อความใน FILE ที่ถูกเก็บไว้โดยให้ใส่ชื่อ FILE ที่ต้องการจะพิมพ์

EXIT TO MAIN MENU เป็นการออกจาก PRINT MENU สู่ MAIN MENU

4. DELETE MESSAGE ON BOX MENU นี้จะเป็นการลบข้อความใน MEMORY ของ ADVANCE TELEX ออกทั้งหมด



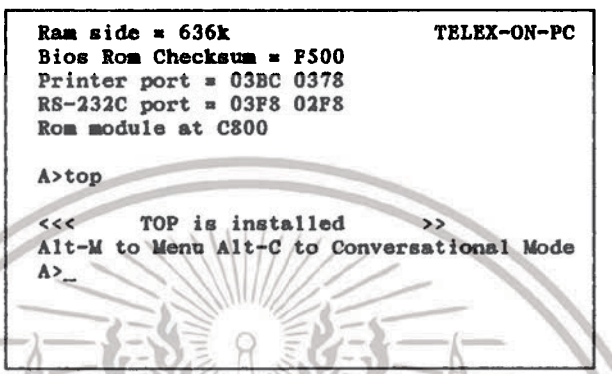
รูปที่ 4 แสดงข้อความที่ปรากฏ

โดยเครื่องจะถามเพื่อให้แน่ใจว่าต้องการที่จะลบข้อความออกหรือไม่ ถ้าไม่ต้องการลบข้อความออกให้ตอบ N

5. EXIT TO DOS เป็นการออกไปสู่ระบบ DOS

การใช้โปรแกรม TOP (TELEX ON PC)

หลังจากโปรแกรม TOP โหลดเรียบร้อยแล้วระบบจะกลับเข้าสู่ DOS ตามเดิมซึ่งขณะนี้ผู้ใช้สามารถใช้โปรแกรมอื่นๆ ได้โดยที่มุมขวามือของจอภาพจะปรากฏข้อความ TELEX ON PC(กระพริบตลอด) เมื่อมีเทเล็กซ์เข้ามามุมขวามือของจอภาพก็จะเปลี่ยนเป็น INCOMING CALL และถ้ามีการส่งเทเล็กซ์ออกมุมขวามือของจอภาพจะมีข้อความ OUT GOING CALL มาแทน ดังรูปที่ 5



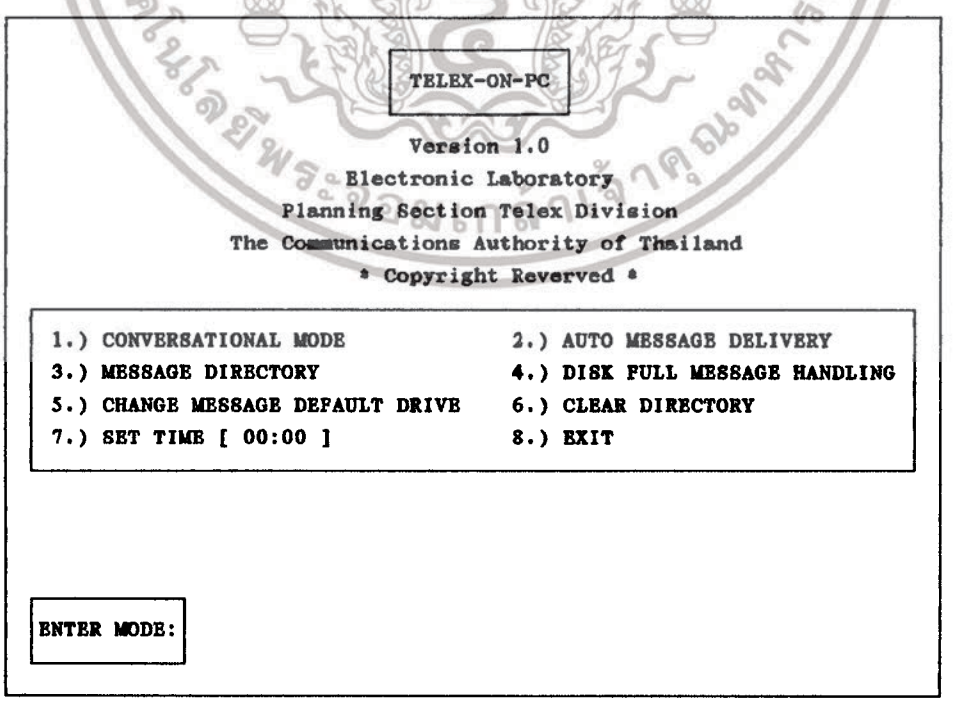
รูปที่ 5 การโหลดโปรแกรม TOP

การเข้าโหมดเทเล็กซ์มี 2 วิธี คือ

กด ALT M จอภาพจะเปลี่ยนไปที่ MAIN MENU ดังรูปที่ 6

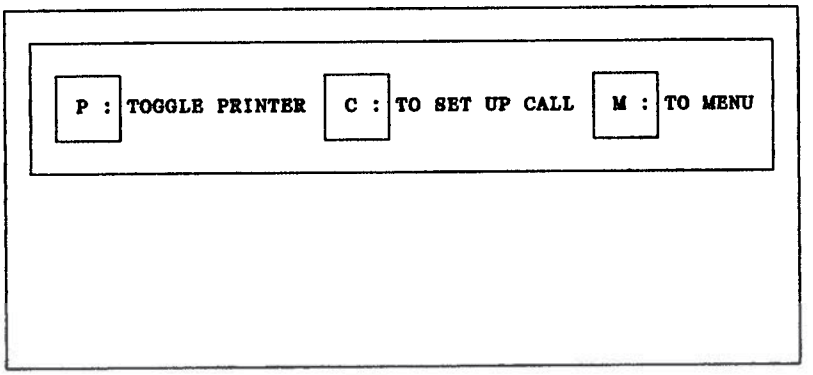
กด ALT C จอภาพจะเปลี่ยนไปที่ CONVERSATION MODE ดังรูปที่ 7

MAIN MENU ประกอบด้วยโหมดต่างๆ ดังรูปที่2 เมื่อเลือกใช้โหมดใดๆ ให้กดหมายเลขนั้น แล้วตามด้วย ENTER



รูปที่ 6 MAIN MENU

1. CONVERSATION MODE จะเป็นการใช้งานอย่างเครื่องเทเล็กซ์ จอภาพจะเปลี่ยนเป็นดังรูปที่ 7

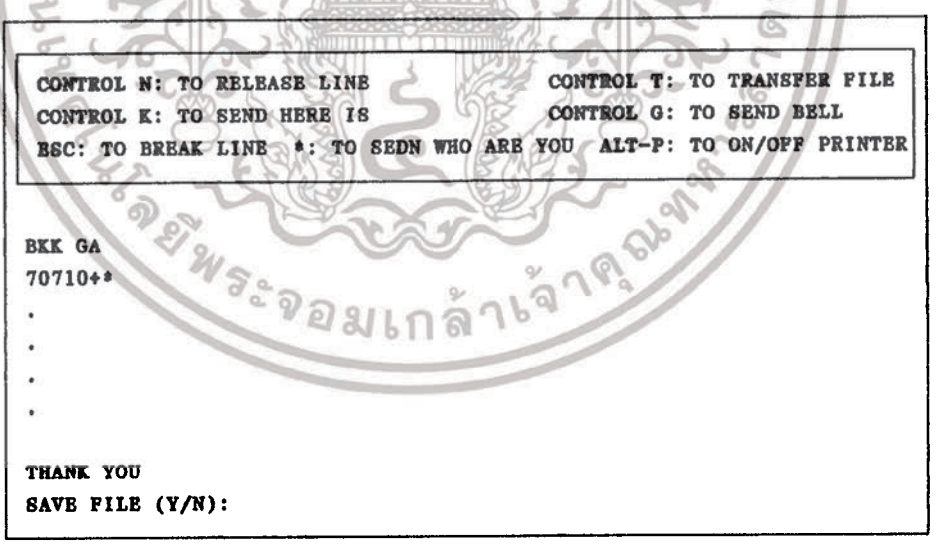


รูปที่ 7 CONVERSATION MODE

P เมื่อกด KEY นี้การรับส่งข้อความเทเล็กซ์จะพิมพ์ออกที่ PRINTER

C จะเป็นการส่งข้อความเทเล็กซ์ ซึ่งทำได้ 2 วิธี คือ

1. ส่งเทเล็กซ์จากไฟล์ (โดย WORD STAR) ที่เตียมไว้ โดยพิมพ์ชื่อไฟล์ที่ต้องการจะส่ง หลังจากนั้น Program จะทำการติดต่อกับชุมสายและจอภาพจะเปลี่ยนไปดังรูปที่ 8 จะมี BBK GA ส่งมาจากชุมสายบนจอภาพ แล้วให้กด SELECTION NUMBER (เลขหมายปลายทาง) แล้วตามด้วยเครื่องหมาย + หลังจากชุมสายให้ ANSWERBACK CODE แล้วให้กด CTRL และ T พร้อมกัน ข้อมูลที่อยู่ในไฟล์ก็จะถูกส่งออกไป หากมีความจำเป็นต้องหยุดการส่งก็ให้กด CTRL T อีกครั้ง



รูปที่ 8 จอภาพขณะ COMPUTER ติดต่อกับชุมสายเทเล็กซ์

2. ส่งเทเล็กซ์โดยการเรียกติดต่อตามปกติ (เหมือนเครื่องเทเล็กซ์) เมื่อปรากฏข้อความ "FILENAME : (ESC TO CANCAL)" ให้กด ENTER หลังจากนั้นการทำงานของเครื่องก็จะเหมือนกับการทำงานของเครื่องเทเล็กซ์ตามปกติ

หลังจากเลิกการติดต่อแล้วจะปรากฏข้อความ "SAVE FILE (Y/N)" ถ้าหากไม่ต้องการเก็บให้ตอบ "N" แต่ถ้าต้องการเก็บข้อความเทเล็กซ์ให้ตอบ "Y"

ฟังก์ชั่นอื่น ๆ

CTRL N การเลิกติดต่อกับชุมสายชนิดได้รับ TIME CHANGE

CTRL K ส่ง ASNWERBACK CODE ออกจากเครื่อง

CTRL T ส่งไฟล์หรือหยุดการส่งไฟล์

CTRL G เป็นการส่ง BELL เรียกปลายทาง

AALT P ให้เครื่องพิมพ์ทำงาน

ESC ขณะมีการติดต่อแล้วกด KEY นี้ LINE TELEX จะถูกตัดโดยไม่ได้ TIME CHANGE

* เป็นสัญญาน WRU (WHO ARE YOU)

2. AUTO MESSAGE DELIVERY โหมดนี้เป็นการส่งเทเล็กซ์โดยวิธีอัตโนมัติ โดยที่ผู้ใช้สามารถกำหนดเวลาหรือตั้งเวลาในการส่งข้อมูลในไฟล์ที่กำหนดไว้ได้ ดังรูปที่ 9

```
        AUTO MESSAGE DELIVERY
          CURRENT TIME: 16:00

SEND FILE (Y/N): Y
FILE NAME TELEX.OUT
TIME: 23:00
TELEX NUBER: 70710
ACCEPT MESSAGE REFERENCE: A CONFIRM (Y/N):

        ESC: TO CANCEL PROCESS
```

รูปที่ 9 ตัวอย่างรูปแบบการตั้งเวลาส่งข้อมูล

3. MESSAGE DIRECTORY ในโหมดนี้ PROGRAM TOP จะแสดงข้อความเทเล็กซ์ต่าง ๆ ดังรูปที่ 10

```
        MESSAGE DIRECTORY

1.) AUTO DELIVERY MESSAGE LIST
2.) INCOMPLETED DELIVERY MESSAGE LIST
3.) COMPLETED DELIVERY MESSAGE LIST
4.) INCOMING TELEX MESSAGE LIST (TLXMSGxx.INC)
5.) OUTGOING TELEX MESSAGE LIST (TLXMSGxx.OUT)
6.) ABBREVIATE SELECTION
7.) MENU


ENTER NUMBER:
```

รูปที่10 แสดง MENU ของ MESSAGE DIRECTORY

3.1 AUTO DELIVERY MESSAGE LIST จะแสดงชื่อของไฟล์ที่ต้องการส่งออก เวลาที่ต้องการส่ง หมายเลขเทเล็กซ์ที่ต้องการติดต่อและหมายเลขประจำข้อความ (กำหนดได้ที่โหมด AUTO MESSAGE DELIVERY) ดังรูปที่ 11

```
        AUTO DELIVERY MESSAGE LIST

FILENAME      TIME      TLX#      MSG#
 B:TEST       13:50     70710      A
 B:TEST       13:52     80848      B

FILE CANCELLED (Y/N):
```

รูปที่11 แสดงข้อความที่ต้องการส่งออก

3.2 INCOMPLETED DELIVERY MESSAGE LIST ในกรณีที่ข้อความที่ตั้งไว้ไม่สามารถส่งออกไปได้ MENU นี้จะแสดงออกมาให้ทราบ

3.3 COMPLETED DELIVERY MESSAGE LIST เป็นการรายงานผลของข้อความเทเล็กซ์ที่ตั้งไว้ ที่สามารถส่งออกไปโดยเรียบร้อย

3.4 INCOMING TELEX MESSAGE LIST จะแสดงข้อความเทเล็กซ์ที่รับเข้ามา

3.5 OUTGOING TELEX MESSAGE LIST จะแสดงข้อความทั้งหมดไม่ว่าด้วยวิธีใดก็ตาม

3.6 ABBREVIATE SELECTION เป็นการใช้ฟังก์ชั่น KEY (F1-F10) แทนหมายเลขเทเล็กซ์ที่ใช้อยู่เป็นประจำ โดยสามารถกำหนดได้สูงสุด 10 หมายเลข

3.7 MENU เป็นการกลับไปสู่ MAIN MENU

4. DISK FULL MESSAGE HANDLING ข้อความเทเล็กซ์ที่รับส่งจะถูกบันทึกลงในแผ่น DISKETTE ภายหลังการเลิกการติดต่อถ้าเกิดกรณีที่ DISKETTE เต็ม เมื่อโปรแกรมตรวจพบก็จะแสดงข้อความเตือนผู้ใช้ดังรูปที่ 12

```
V: TO DISPLAY MESSAGE    P: TO DISPLAY&PRINT MESSAGE
S: TO SAVE               M: TO MENU

ENTER MODE:
```

รูปที่12 DISK FULL MESSAGE HANDING

ในโหมดนี้ผู้ใช้จะต้องเปลี่ยน DISKETTE ใหม่มาใส่แทนของเก่า หลังจากเปลี่ยนแล้ว ผู้ใช้สามารถที่จะเลือกจัดการกับข้อมูลสุดท้าย ที่ยังไม่ได้จัดเก็บได้ โดยการกด KEY ตามคำสั่งดังนี้

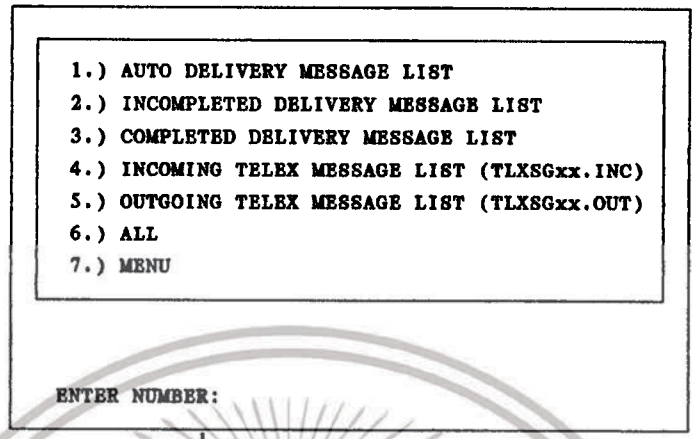
V เป็นการตรวจสอบข้อความบนจอภาพ

P เป็นการตรวจสอบข้อความบนจอภาพพร้อมทั้งพิมพ์ออกเครื่องพิมพ์

S เป็นการเก็บข้อความลงบนแผ่น DISKETTE ที่เปลี่ยนเข้ามาใหม่

M เป็นการออกสู่ MAIN MENU โดยไม่มีการจัดเก็บข้อความเทเล็กซ์

5. CHANGE MESSAGE DEFAULT DRIVE เป็นการกำหนด DISK DRIVE ที่ใช้ในการเก็บข้อมูล
6. CLEAR DIRECTORY ในโหมดนี้ผู้ใช้สามารถเลือกลบข้อความที่ตั้งขึ้นในโหมด MESSAGE DIRECTORY โดยจะมี MENU ดังรูปที่ 9



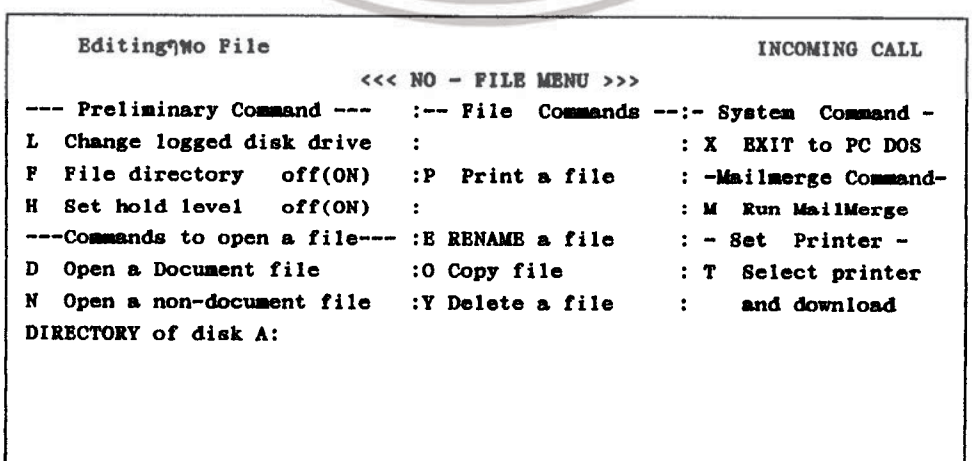
รูปที่9 CLEAR DIRECTORY

6.1 AUTO DELEVERY MESSAGE LIST ยกเลิกทุกข้อมูลทุกเวลาที่จะส่ง
6.2 INCOMPLETED DELIVERY MESSAGE LIST ยกเลิกทุกข้อมูลที่ไม่สามารถส่งได้
6.3 COMPLETED DELIVERY MESSAGE LIST ยกเลิกทุกข้อมูลที่ฝากส่งไปแล้ว
6.4 INCOMING TELEX MESSAGE LIST ยกเลิกทุกข้อมูลที่รับเข้ามา
6.5 OUTGOING TELEX MESSAGE LIST ยกเลิกทุกข้อมูลที่ได้ส่งไปแล้ว
6.6 ALL ยกเลิกทุกข้อความที่ปรากฏใน MESSAGE DIRECTORY
6.7 MENU กลับเข้าสู่ MAIN MENU

7. SET TIME เป็นการตั้งเวลาเพื่อใช้เป็นฐานในการส่งเทเล็กซ์
8. EXIT ออกสู่ระบบ DOS

การรับเทเล็กซ์ขณะที่ผู้ใช้เครื่องทำงานอื่น

ในขณะที่ผู้ใช้เครื่องไมโครคอมพิวเตอร์ทำงานอื่นอยู่ และมีสัญญาณเทเล็กซ์เข้ามา (โดยที่มุมขวาของจอภาพจะปรากฏข้อความ INCOMING CALL) ผู้ใช้สามารถที่จะหยุดการทำงานและมารับเทเล็กซ์โดยทันที ก็สามารถจะทำได้โดยกดคีย์ ALT C จอภาพจะเปลี่ยนมาอยู่ใน CONVERSATIONAL MODE ผู้ใช้สามารถโต้ตอบหรืออ่านข้อความเทเล็กซ์ได้ทันที ดังแสดงในรูปที่ 10



รูปที่ 10 การรับเทเล็กซ์ที่เรียกเข้า

ภาคผนวก ค

รายละเอียดข้อมูลของอุปกรณ์

8085A/8085A-2
SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSORS

- Single +5V Power Supply
- 100% Software Compatible with 8080A
- 1.3 μs Instruction Cycle (8085A); 0.8 μs (8085A-2)
- On-Chip Clock Generator (with External Crystal, LC or RC Network)
- On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control
- Four Vectored Interrupt Inputs (One is non-Maskable) Plus an 8080A-compatible interrupt
- Serial In/Serial Out Port
- Decimal, Binary and Double Precision Arithmetic
- Direct Addressing Capability to 64k Bytes of Memory

The Intel® 8085A is a complete 8 bit parallel Central Processing Unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085A (CPU), 8156 (RAM/IO) and 8355/8755A (ROM/PROM/IO)] while maintaining total system expandability. The 8085A-2 is a faster version of the 8085A.

The 8085A incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a high level of system integration.

The 8085A uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8155/8156/8355/8755A memory products allow a direct interface with the 8085A.

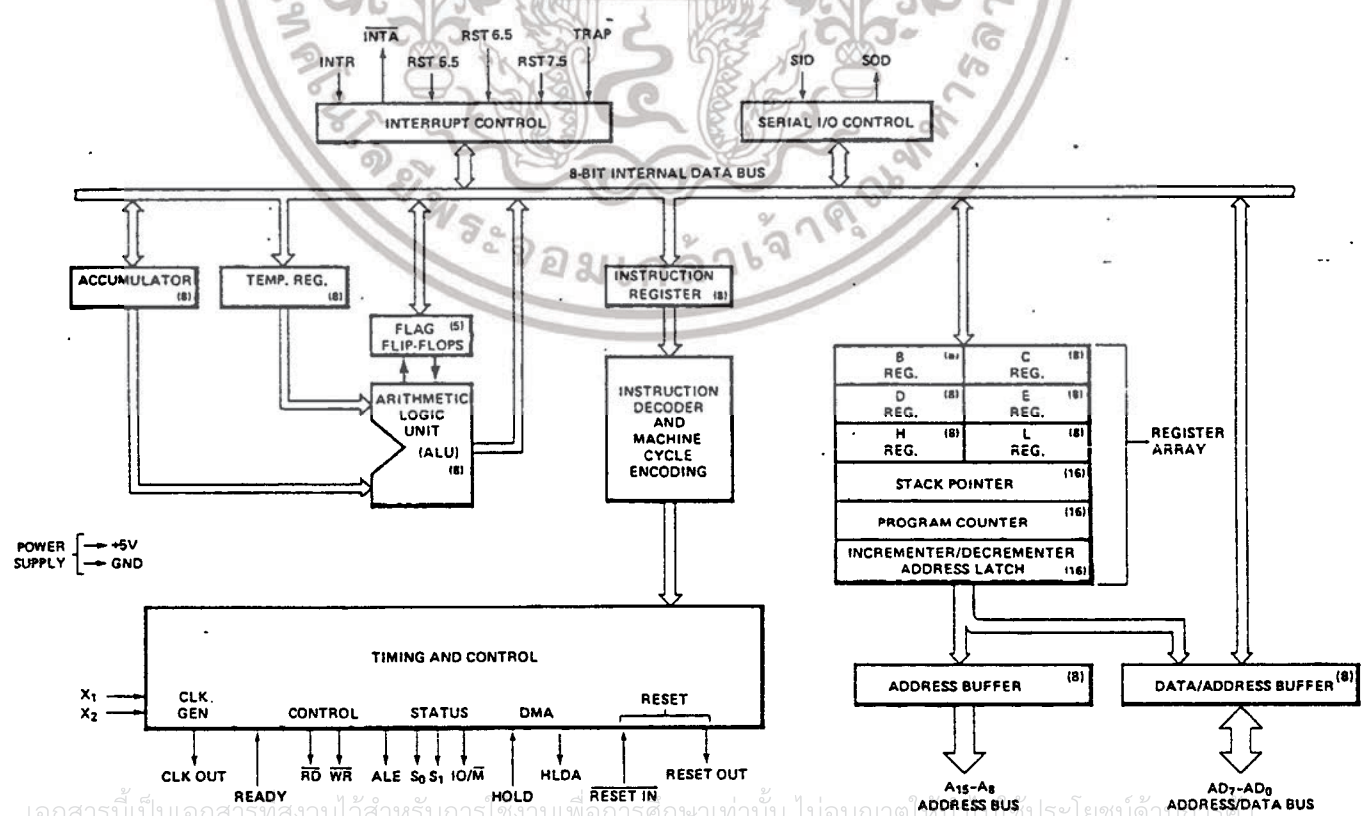


Figure 1. 8085A CPU Functional Block Diagram

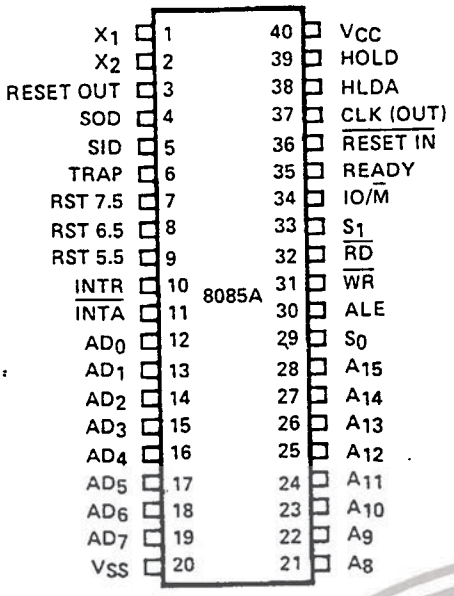


Figure 2. 8085A Pinout Diagram

8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin:

Symbol	Function
A_8–A_{15} (Output, 3-state)	Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.
AD_{0-7} (Input/Output, 3-state)	Multiplexed Address/Data Bus: Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.
ALE (Output)	Address Latch Enable: It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1, and IO/$\overline{M}$ (Output)	Machine cycle status:

IO/$\overline{M}$	S_1	S_0	Status
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt Acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

* = 3-state (high impedance)
X = unspecified

Symbol	Function
	S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$ (Output, 3-state)	READ control: A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$ (Output, 3-state)	WRITE control: A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY (Input)	If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle.
HOLD (Input)	HOLD indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA (Output)	HOLD ACKNOWLEDGE: Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.
INTR (Input)	INTERRUPT REQUEST: is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

8085A FUNCTIONAL PIN DESCRIPTION (Continued)

Symbol	Function
$\overline{INTA}$ (Output)	INTERRUPT ACKNOWLEDGE: Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5 (Inputs)	RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP (Input)	Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 1.)
$\overline{RESET\ IN}$ (Input)	Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{RESET\ IN}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay. The cpu is held in the reset condition as long as $\overline{RESET\ IN}$ is applied.
RESET OUT (Output)	Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2 (Input)	X_1 and X_2 are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK (Output)	Clock Output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID (Input)	Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD (Output)	Serial output data line. The output SOD is set or reset as specified by the SIM instruction.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

TABLE 1. INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2).	High level until sampled.

NOTES:

(1) The processor pushes the PC on the stack before branching to the indicated address.
(2) The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085A or 8085A-2 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The driving frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085A is operated with a 6 MHz crystal (for 3 MHz clock), and the 8085A-2 can be operated with a 10 MHz crystal (for 5 MHz clock). If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired
C_L (load capacitance) ≤ 30 pf
C_S (shunt capacitance) ≤ 7 pf
R_S (equivalent shunt resistance) ≤ 75 Ohms
Drive level: 10 mW
Frequency tolerance: ±.005% (suggested)

Note the use of the 20 pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the 8085A, providing that its frequency tolerance of approximately ±10% is acceptable. The components are chosen from the formula:

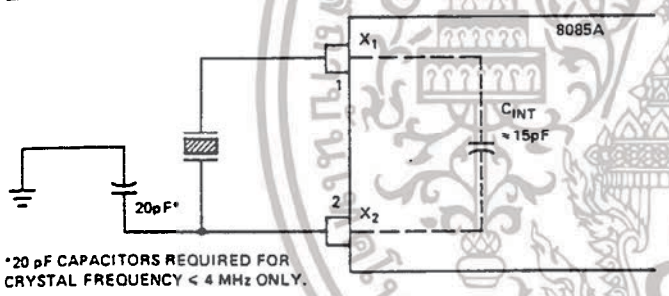
$$f = \frac{1}{2\pi\sqrt{L(C_{ext} + C_{int})}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

An RC circuit may be used as the frequency-determining network for the 8085A if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

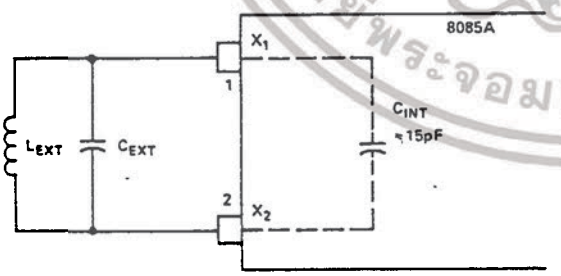
Figure 4 shows the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4 V.

For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figue 4D). If the driving frequency is from 6 MHz to 10 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 4E). To prevent self-oscillation of the 8085A, be sure that X_2 is not coupled back to X_1 through the driving circuit.

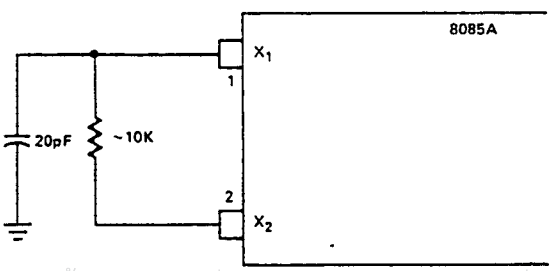


*20 pF CAPACITORS REQUIRED FOR CRYSTAL FREQUENCY < 4 MHz ONLY.

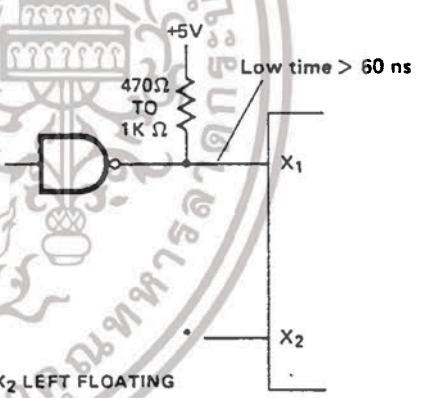
A. Quartz Crystal Clock Driver



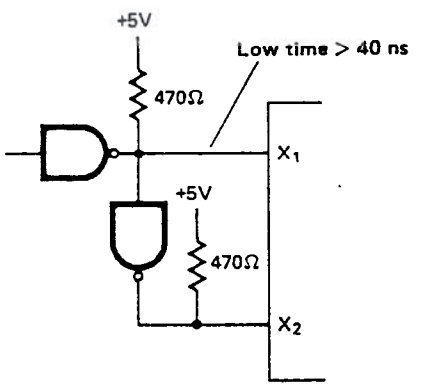
B. LC Tuned Circuit Clock Driver



C. RC Circuit Clock Driver



D. 1-6 MHz Input Frequency External Clock Driver Circuit



E. 1-10 MHz Input Frequency External Clock Driver Circuit

Figure 4. Clock Driver Circuits

BASIC SYSTEM TIMING

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 9 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ($IO/\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$). (See Table 2.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each machine cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of $\overline{READY}$ or HOLD inputs). Any T state must be one of ten possible states, shown in Table 3.

TABLE 2. 8085A MACHINE CYCLE CHART

MACHINE CYCLE			STATUS			CONTROL		
			$IO/\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
OPCODE FETCH	(OF)		0	1	1	0	1	1
MEMORY READ	(MR)		0	1	0	0	1	1
MEMORY WRITE	(MW)		0	0	1	1	0	1
I/O READ	(IOR)		1	1	0	0	1	1
I/O WRITE	(IOW)		1	0	1	1	0	1
ACKNOWLEDGE OF INTR	(INA)		1	1	1	1	1	0
BUS IDLE	(BI):	DAD	0	1	0	1	1	1
		ACK. OF RST,TRAP	1	1	1	1	1	1
		HALT	TS	0	0	TS	TS	1

TABLE 3. 8085A MACHINE STATE CHART

Machine State	Status & Buses				Control		
	S1,S0	$IO/\overline{M}$	A_8-A_{15}	AD_0-AD_7	$\overline{RD}$,$\overline{WR}$	$\overline{INTA}$	ALE
T_1	X	X	X	X	1	1	1*
T_2	X	X	X	X	X	X	0
T_{WAIT}	X	X	X	X	X	X	0
T_3	X	X	X	X	X	X	0
T_4	1	0†	X	TS	1	1	0
T_5	1	0†	X	TS	1	1	0
T_6	1	0†	X	TS	1	1	0
T_{RESET}	X	TS	TS	TS	TS	1	0
T_{HALT}	0	TS	TS	TS	TS	1	0
T_{HOLD}	X	TS	TS	TS	TS	1	0

0 = Logic "0" TS = High Impedance
1 = Logic "1" X = Unspecified

* ALE not generated during 2nd and 3rd machine cycles of DAD instruction.
† IO/M = 1 during T_4-T_6 of INA machine cycle.

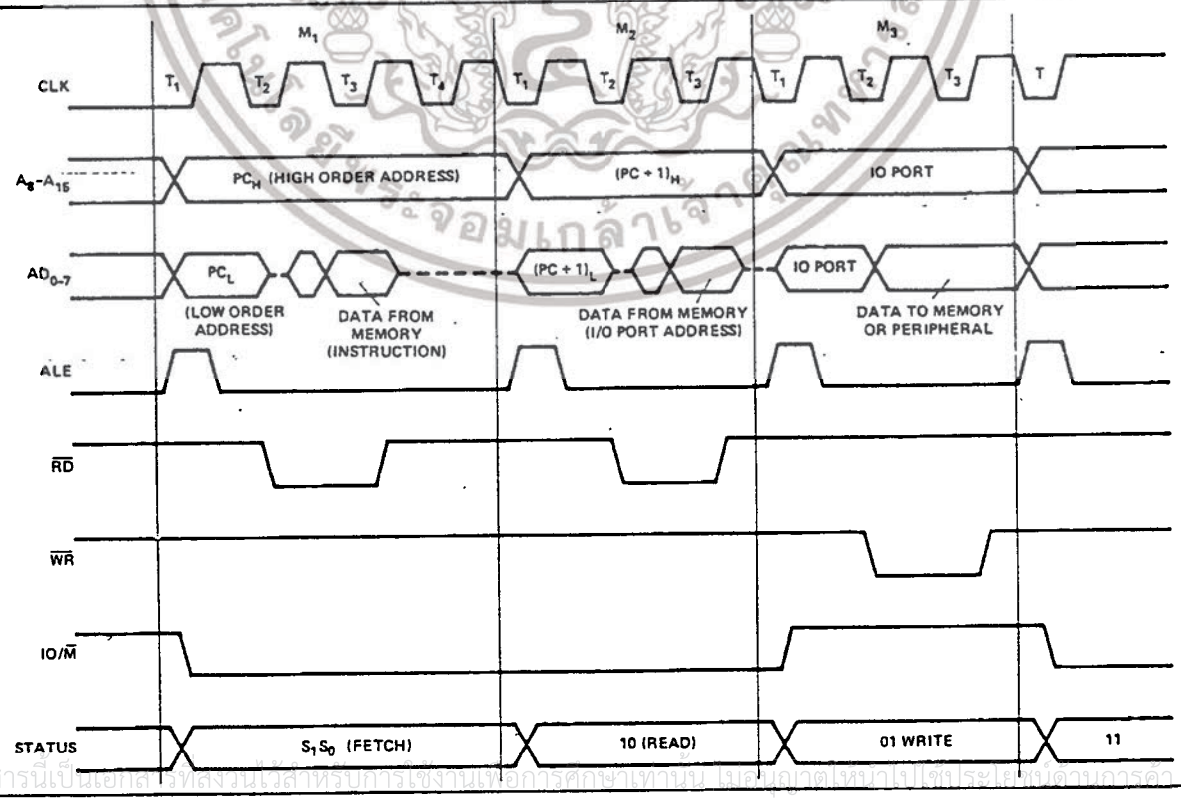


Figure 9. 8085A Basic System Timing

TABLE 4. ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . −0.5V to +7V
Power Dissipation . . . . . 1.5 Watt

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions 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.

TABLE 5. D.C. CHARACTERISTICS

(T_A = 0°C to 70°C; V_{CC} = 5V ±5%; V_{SS} = 0V; unless otherwise specified)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{CC}	Power Supply Current		170	mA	
I_{IL}	Input Leakage		±10	μA	$V_{in} = V_{CC}$
I_{LO}	Output Leakage		±10	μA	$0.45V \leq V_{out} \leq V_{CC}$
V_{ILR}	Input Low Level, RESET	-0.5	+0.8	V	
V_{IHR}	Input High Level, RESET	2.4	V_{CC}+0.5	V	
V_{HY}	Hysteresis, RESET	0.25		V	

TABLE 6. A.C. CHARACTERISTICS

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

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{CYC}	CLK Cycle Period	320	2000	200	2000	ns
t_1	CLK Low Time (Standard CLK Loading)	80		40		ns
t_2	CLK High Time (Standard CLK Loading)	120		70		ns
t_r, t_f	CLK Rise and Fall Time		30		30	ns
t_{XKR}	X_1 Rising to CLK Rising	30	120	30	100	ns
t_{XKF}	X_1 Rising to CLK Falling	30	150	30	110	ns
t_{AC}	A_{8-15} Valid to Leading Edge of Control[1]	270		115		ns
t_{ACL}	A_{0-7} Valid to Leading Edge of Control	240		115		ns
t_{AD}	A_{0-15} Valid to Valid Data In		575		350	ns
t_{AFR}	Address Float After Leading Edge of $\overline{READ}$ ($\overline{INTA}$)		0		0	ns
t_{AL}	A_{8-15} Valid Before Trailing Edge of ALE[1]	115		50		ns
t_{ALL}	A_{0-7} Valid Before Trailing Edge of ALE	90		50		ns
t_{ARY}	READY Valid from Address Valid		220		100	ns
t_{CA}	Address (A_{8-15}) Valid After Control	120		60		ns
t_{CC}	Width of Control Low ($\overline{RD}$, $\overline{WR}$, $\overline{INTA}$)	400		230		ns
t_{CL}	Trailing Edge of Control to Leading Edge of ALE	50		25		ns
t_{DW}	Data Valid to Trailing Edge of $\overline{WRITE}$	420		230		ns
t_{HABE}	HLDA to Bus Enable		210		150	ns
t_{HABF}	Bus Float After HLDA		210		150	ns
t_{HACK}	HLDA Valid to Trailing Edge of CLK	110		40		ns
t_{HDH}	HOLD Hold Time	0		0		ns
t_{HDS}	HOLD Setup Time to Trailing Edge of CLK	170		120		ns
t_{INH}	INTR Hold Time	0		0		ns
t_{INS}	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		ns
t_{LA}	Address Hold Time After ALE	100		50		ns
t_{LC}	Trailing Edge of ALE to Leading Edge of Control	130		60		ns
t_{LCK}	ALE Low During CLK High	100		50		ns
t_{LDR}	ALE to Valid Data During Read		460		270	ns
t_{LDW}	ALE to Valid Data During Write		200		120	ns
t_{LL}	ALE Width	140		80		ns
t_{LRY}	ALE to READY Stable		110		30	ns

Table 6. A.C. Characteristics (Cont.)

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{RAE}	Trailing Edge of $\overline{READ}$ to Re-Enabling of Address	150		90		ns
t_{RD}	$\overline{READ}$ (or $\overline{INTA}$) to Valid Data		300		150	ns
t_{RV}	Control Trailing Edge to Leading Edge of Next Control	400		220		ns
t_{RDH}	Data Hold Time After $\overline{READ}$ $\overline{INTA}$[7]	0		0		ns
t_{RYH}	READY Hold Time	0		0		ns
t_{RYS}	READY Setup Time to Leading Edge of CLK	110		100		ns
t_{WD}	Data Valid After Trailing Edge of $\overline{WRITE}$	100		60		ns
t_{WDL}	LEADING Edge of WRITE to Data Valid		40		20	ns

Notes:

1. A_8-A_{15} address Specs apply to IO/$\overline{M}$, S_0, and S_1 except A_8-A_{15} are undefined during T_4-T_6 of OF cycle whereas IO/$\overline{M}$, S_0, and S_1 are stable.
2. Test conditions: t_{CYC} = 320 ns (8085A)/200 ns (8085A-2); C_L = 150 pF.
3. For all output timing where C_L = 150 pF use the following correction factors:
   25 pF ≤ C_L < 150 pF: −0.10 ns/pF
   150 pF < C_L ≤ 300 pF: +0.30 ns/pF
4. Output timings are measured with purely capacitive load.
5. All timings are measured at output votage V_L = 0.8V, V_H = 2.0V, and 1.5V with 20 ns rise and fall time on inputs.
6. To calculate timing specifications at other values of t_{CYC} use Table 7.
7. Data hold time is guaranteed under all loading conditions.

Input Waveform for A.C. Tests:

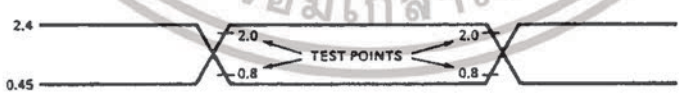


TABLE 7. BUS TIMING SPECIFICATION AS A T_{CYC} DEPENDENT

	8085A			8085A-2 (Preliminary)		
t_{AL}	–	(1/2) T - 45	MIN	(1/2) T - 50	–	MIN
t_{LA}	–	(1/2) T - 60	MIN	(1/2) T - 50	–	MIN
t_{LL}	–	(1/2) T - 20	MIN	(1/2) T - 20	–	MIN
t_{LCK}	–	(1/2) T - 60	MIN	(1/2) T - 50	–	MIN
t_{LC}	–	(1/2) T - 30	MIN	(1/2) T - 40	–	MIN
t_{AD}	–	(5/2 + N) T - 225	MAX	(5/2 + N) T - 150	–	MAX
t_{RD}	–	(3/2 + N) T - 180	MAX	(3/2 + N) T - 150	–	MAX
t_{RAE}	–	(1/2) T - 10	MIN	(1/2) T - 10	–	MIN
t_{CA}	–	(1/2) T - 40	MIN	(1/2) T - 40	–	MIN
t_{DW}	–	(3/2 + N) T - 60	MIN	(3/2 + N) T - 70	–	MIN
t_{WD}	–	(1/2) T - 60	MIN	(1/2) T - 40	–	MIN
t_{CC}	–	(3/2 + N) T - 80	MIN	(3/2 + N) T - 70	–	MIN
t_{CL}	–	(1/2) T - 110	MIN	(1/2) T - 75	–	MIN
t_{ARY}	–	(3/2) T - 260	MAX	(3/2) T - 200	–	MAX
t_{HACK}	–	(1/2) T - 50	MIN	(1/2) T - 60	–	MIN
t_{HABF}	–	(1/2) T + 50	MAX	(1/2) T + 50	–	MAX
t_{HABE}	–	(1/2) T + 50	MAX	(1/2) T + 50	–	MAX
t_{AC}	–	(2/2) T - 50	MIN	(2/2) T - 85	–	MIN
t_1	–	(1/2) T - 80	MIN	(1/2) T - 60	–	MIN
t_2	–	(1/2) T - 40	MIN	(1/2) T - 30	–	MIN
t_{RV}	–	(3/2) T - 80	MIN	(3/2) T - 80	–	MIN
t_{LDR}	–	(4/2) T - 180	MAX	(4/2) T - 130	–	MAX

NOTE: N is equal to the total WAIT states.
$T = t_{CYC}$.

NOTE: N is equal to the total WAIT states.
$T = t_{CYC}$.

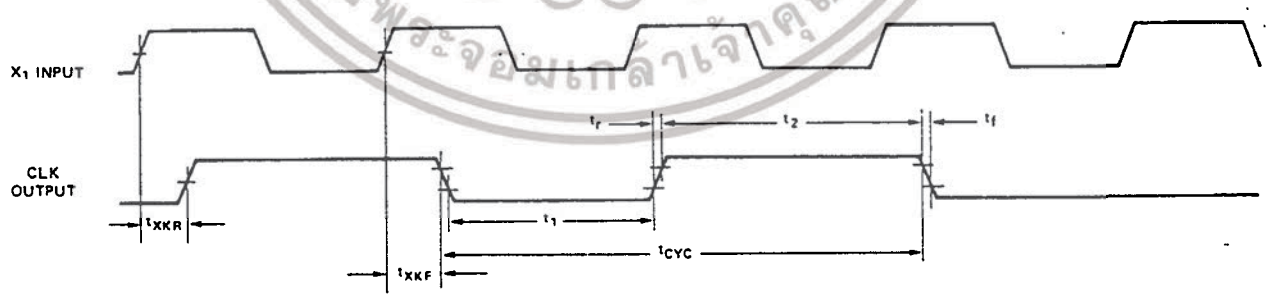
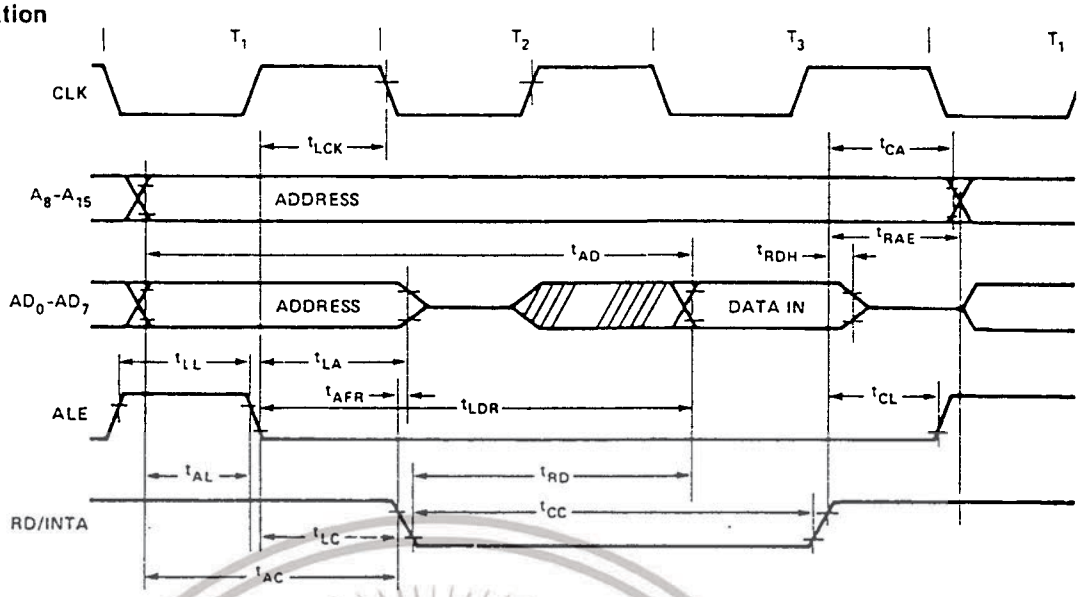
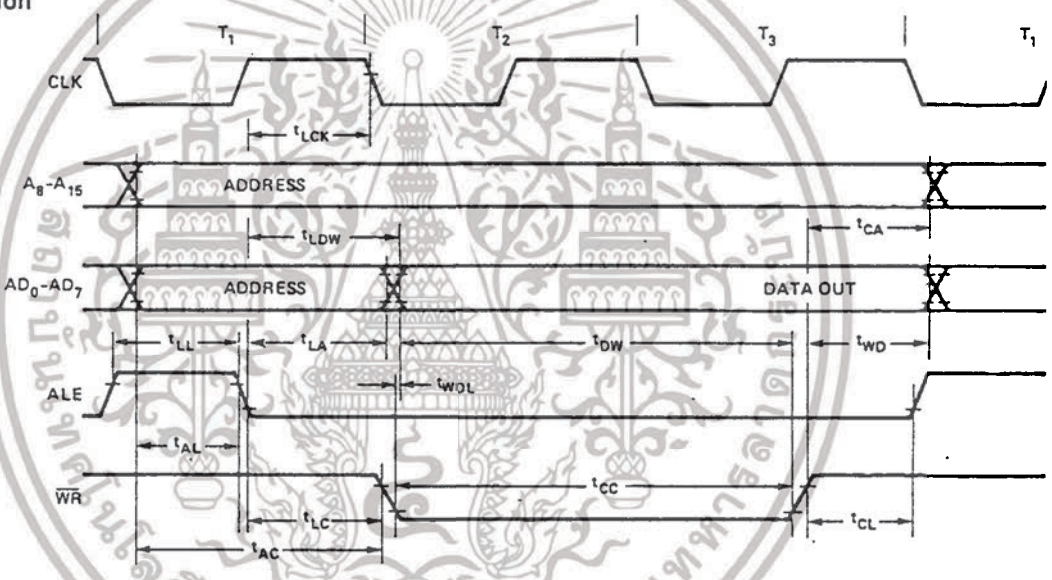


Figure 10. Clock Timing Waveform

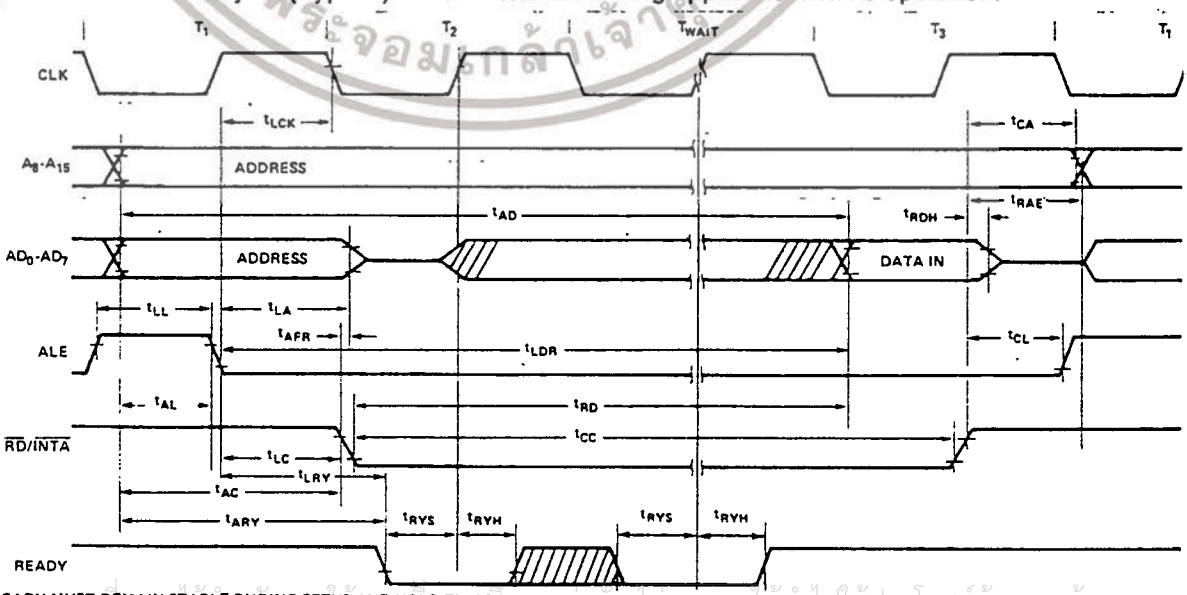
Read Operation



Write Operation



Read operation with Wait Cycle (Typical) — same READY timing applies to WRITE operation.



NOTE 1: READY MUST REMAIN STABLE DURING SETUP AND HOLD TIMES.

Figure 11. 8085A Bus Timing, With and Without Wait

MOTOROLA

MC14411

BIT RATE GENERATOR

The MC14411 bit rate generator is constructed with complementary MOS enhancement mode devices. It utilizes a frequency divider network to provide a wide range of output frequencies.

A crystal controlled oscillator is the clock source for the network. A two-bit address is provided to select one of four multiple output clock rates.

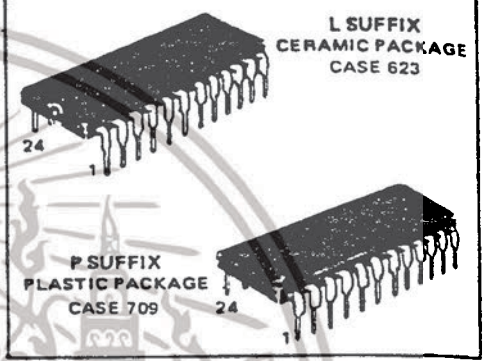
Applications include a selectable frequency source for equipment in the data communications market, such as teleprinters, printers, CRT terminals, and microprocessor systems.

- Single 5.0 Vdc (± 5%) Power Supply
- Internal Oscillator Crystal Controlled for Stability (1.8432 MHz)
- Sixteen Different Output Clock Rates
- 50% Output Duty Cycle
- Programmable Time Bases for One of Four Multiple Output Rates
- Buffered Outputs Compatible with Low Power TTL
- Noise Immunity = 45% of V_{DD} Typical
- Diode Protection on All Inputs
- External Clock May be Applied to Pin 21

CMOS LSI

(LOW-POWER COMPLEMENTARY MOS)

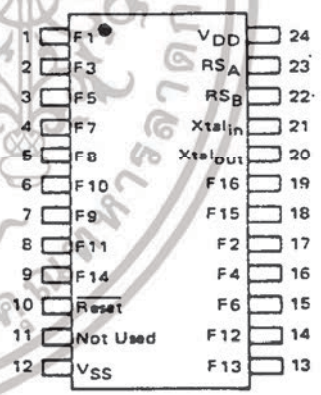
BIT RATE GENERATOR



MAXIMUM RATINGS (Voltages referenced to V_{SS}, Pin 12.)

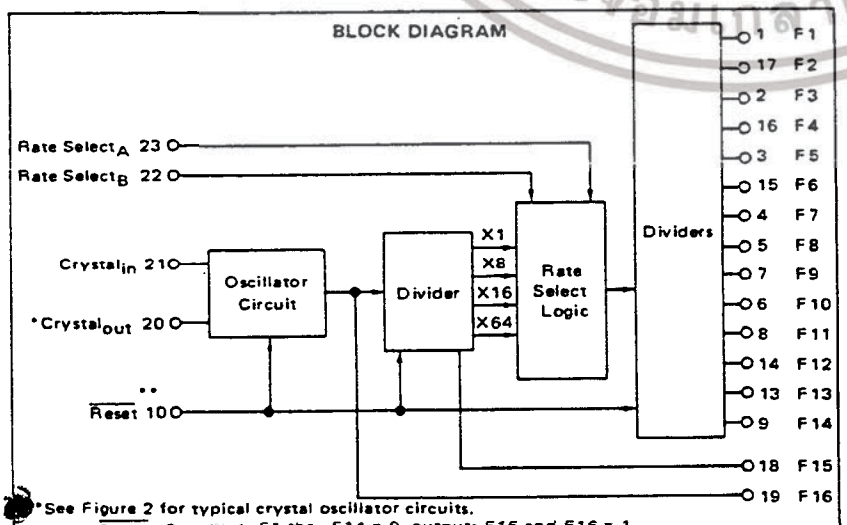
Rating	Symbol	Value	Unit
DC Supply Voltage Range	V_{DD}	5.25 to -0.5	Vdc
Input Voltage, All Inputs	V_{in}	V_{DD} + 0.5 to V_{SS} -0.5	Vdc
DC Current Drain per Pin	I	10	mAdc
Operating Temperature Range	T_A	-40 to +85	°C
Storage Temperature Range	T_{stg}	-65 to +150	°C

PIN ASSIGNMENT



V_{DD} = Pin 24
V_{SS} = Pin 12

BLOCK DIAGRAM



*See Figure 2 for typical crystal oscillator circuits.
**When Reset = 0, outputs F1 thru F14 = 0, outputs F15 and F16 = 1.

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{DD}$.

Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V_{DD} Vdc	-40°C Min	-40°C Max	25°C Min	25°C Typ	25°C Max	+85°C Min	+85°C Max	Unit
Supply Voltage	V_{DD}	–	4.75	5.25	4.75	5.0	5.25	4.75	5.25	Vdc
Output Voltage "0" Level	V_{out}	5.0	–	0.05	–	0	0.05	–	0.05	Vdc
Output Voltage "1" Level		5.0	4.95	–	4.95	5.0	–	4.95	–	Vdc
Input Voltage (V_O = 4.5 or 0.5 Vdc)	V_{IL}	5.0	–	1.5	–	2.25	1.5	–	1.5	Vdc
Input Voltage (V_O = 0.5 or 4.5 Vdc)	V_{IH}	5.0	3.5	–	3.5	2.75	–	3.5	–	Vdc
Output Drive Current (V_{OH} = 2.5 Vdc) Source	I_{OH}	5.0	-0.23	–	-0.20	-1.7	–	-0.16	–	mAdc
(V_{OL} = 0.4 Vdc) Sink	I_{OL}	5.0	0.23	–	0.20	0.78	–	0.16	–	mAdc
Input Current	I_{in}	–	–	±0.1	–	±0.00001	±0.1	–	±1.0	µAdc
Input Capacitance (V_{in} = 0)	C_{in}	–	–	–	–	5.0	–	–	–	pF
Quiescent Dissipation	P_Q	5.0	–	2.5	–	0.015	2.5	–	15	mW
Power Dissipation**† (Dynamic plus Quiescent) (C_L = 15 pF)	P_D	5.0	(P_D = (7.5 mW/MHz) f + P_Q)							mW
Output Rise Time** t_{TLH} = (3.0 ns/pF) C_L + 25 ns	t_{TLH}	5.0	–	–	–	70	200	–	–	ns
Output Fall Time** t_{THL} = (1.5 ns/pF) C_L + 47 ns	t_{THL}	5.0	–	–	–	70	200	–	–	ns
Input Clock Frequency	f_{CL}	5.0	–	1.85	–	–	1.85	–	1.85	MHz

† For dissipation at different external load capacitance (C_L) refer to corresponding formula:

$P_T(C_L) = P_D + 2.6 \times 10^{-3} (C_L - 15\ pF) V_{DD}^2 f$

where: P_T, P_D in mW, C_L in pF, V_{DD} in Vdc, and f in MHz.

* *The formula given is for the typical characteristics only.

TABLE 1 – OUTPUT CLOCK RATES

Rate Select B	Rate Select A	Rate
0	0	X1
0	1	X8
1	0	X16
1	1	X64

Output Number	Output Rates (Hz) X64	X16	X8	X1
F1	614.4 k	153.6 k	76.8 k	9600
F2	460.8 k	115.2 k	57.6 k	7200
F3	307.2 k	76.8 k	38.4 k	4800
F4	230.4 k	57.6 k	28.8 k	3600
F5	153.6 k	38.4 k	19.2 k	2400
F6	115.2 k	28.8 k	14.4 k	1800
F7	76.8 k	19.2 k	9600	1200
F8	38.4 k	9600	4800	600
F9	19.2 k	4800	2400	300
F10	12.8 k	3200	1600	200
F11	9600	2400	1200	150
F12	8613.2	2153.3	1076.6	134.5
F13	7035.2	1758.8	879.4	109.9
F14	4800	1200	600	75
F15	921.6 k	921.6 k	921.6 k	921.6 k
F16*	1.843M	1.843M	1.843M	1.843M

*F16 is buffered oscillator output.

FIGURE 1 – DYNAMIC SIGNAL WAVEFORMS

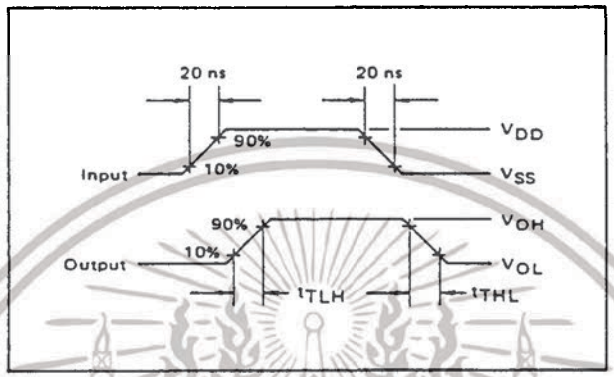
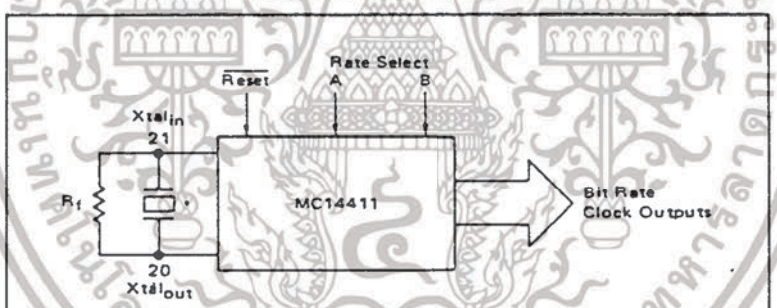


FIGURE 2 – TYPICAL CRYSTAL OSCILLATOR CIRCUIT



R_f = 15 MΩ±10%

CRYSTAL SPECIFICATION

Crystal Mode	Parallel
Frequency	1.8432 MHz ±0.05% @ 13 pF
R_S	540 Ω max
C_0	7.0 pF max
Temperature Range	0 to 70°C
Test Level	1 mW
Test Set	TS – 330/TSM or Equivalent

*Suggested Crystal Suppliers:
Tyco CTS Knights
Motorola Crystal Products

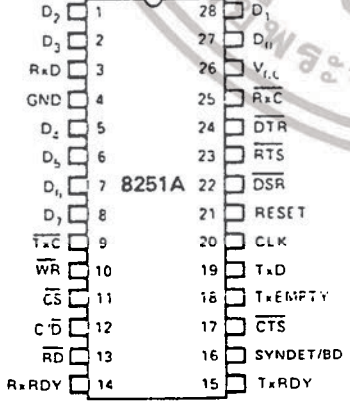
intel®

8251A
PROGRAMMABLE COMMUNICATION INTERFACE

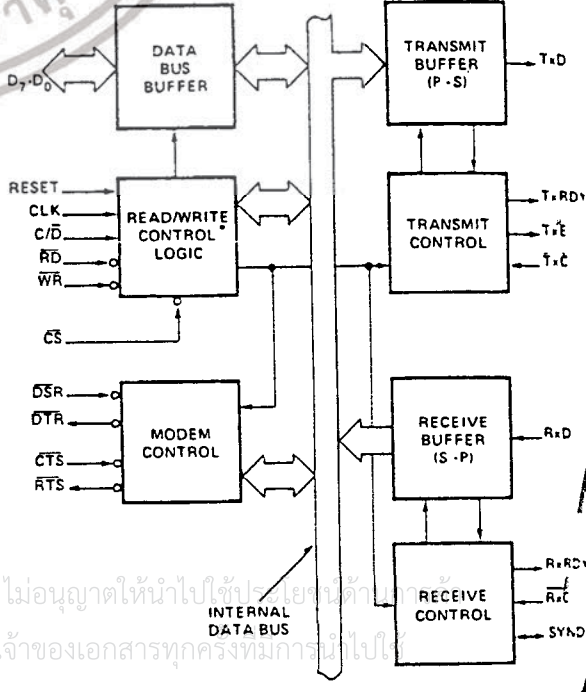
- Synchronous and Asynchronous Operation
- Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling.
- Synchronous Baud Rate — DC to 64K Baud
- Asynchronous Baud Rate — DC to 19.2K Baud
- Full Duplex, Double Buffered, Transmitter and Receiver
- Error Detection — Parity, Overrun and Framing
- Fully Compatible with 8080/8085 CPU
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Single +5V Supply
- Single TTL Clock

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronou Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family o microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operat using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accept data characters from the CPU in parallel format and then converts them into a continuous serial data stream fo transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for th CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it ha received a character for the CPU. The CPU can read the complete status of the USART at any time. These include da transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silico gate technology.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char from 8080)

$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground