โครงข่ายข้อมูลไร้สาย

WIRELESS NETWORK

โดย

นาย ธานี เอิบอาบ

นาย นคร โชติธนานุรักษ์

นาย บัณฑิต ผ่องฉาย

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปริญญานิพนธ์ปีการศึกษา 2535

เรื่อง โครงข่ายข้อมูลไร้สาย

WIRELESS NETWORK

ผู้จัดทำ

1.นาย ธานี	เอิบอาบ	32.1121
2.นาย นคร	รัชติธนานุรักษ์	32.1133
3.นาย บัณฑิต	ฝ่องฉาย	32.1157

..... อาจารย์ที่ปรึกษา

(อาจารย์ พลพดุง ผดุงกุล)

โครงข่ายข้อมูลไร้สาย

WIRELESS NETWORK

ธานี เอิบอาบ 32.1121
นคร โชติธนานุรักษ์ 32.1133
บัณฑิต ผ่องฉาย 32.1157
อ.พลผดุง ผดุงกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2535

บทคัดย่อ

ปริญญานิพนธ์เรื่องโครงข่ายข้อมูลไร้สายนี้ เป็นการนำเอาการสื่อสารโดยใช้ระบบวิทยุมาประยุกต์ใช้เพื่ออุปกรณ์ประเภทอุปกรณ์ปลายทางข้อมูล(DATA TERMINAL EQUIPMENT (DTE)) สามารถสื่อสารข้อมูลถึงกันได้ ซึ่งมีประโยชน์สำหรับการสื่อสารข้อมูลเป็นระยะทางไกลๆ โดยไม่เกิดปัญหาทางด้านการวางสาย ลักษณะการส่งข้อมูลเป็นฮาล์ฟดูเพล็ก (HALF DUPLEX)โดยผ่านทางอินเตอร์เฟส RS-232C และระบบมีการเข้าถึงข้อมูลแบบเซนทรัลไลซ์โพลลิ่ง (Centralized Polling) ซึ่งในระบบการสื่อสารระหว่างคอมพิวเตอร์เป็นระบบที่สามารถทำการส่งข่าวสารทั้งในรูปของข้อความ และบันทึกในรูปของไฟล์ไปยังคอมพิวเตอร์เครื่องอื่นๆ ที่ต่ออยู่ในระบบได้ โดยจะมีไมโครโปรเซสเซอร์ Z-80 ทำหน้าที่เป็นศูนย์กลาง (server)ในการรับส่งข้อความข่าวสารหรือไฟล์ต่างๆให้แก่เครื่องคอมพิวเตอร์ที่เป็นเครื่องปลายทาง(terminal) โดยตัวศูนย์กลางจะเป็นตัวคอยตรวจสอบเครื่องปลายทางอยู่ตลอดเวลาว่าต้องการติดต่อหรือไม่ซึ่งมีลักษณะคล้ายระบบโครงข่ายข้อมูลของคอมพิวเตอร์ พร้อมทั้งได้ทำการเขียนโปรแกรมเรซิเดนซ์(resident) ฝังไว้ในเครื่องปลายทาง เพื่อให้การรับส่งข้อมูลทำได้ทันที โดยไม่ต้องรอให้เครื่องรับว่างจากการทำงานของโปรแกรมอื่นก่อน เมื่อทำการรับส่งข้อมูลเสร็จ ก็จะกลับไปทำงานที่โปรแกรมเดิมโดยไม่ทำให้การทำงานของโปรแกรมเดิมเสียสถานะไป

WIRELESS NETWORK

Abstract

This thesis is wireless network which concerning about the application of communication system by radio signal. This system will enable the DATA TERMINAL EQUIPMENTS (DTES) to communicate between one another. It is most useful for transmission of data in a far distance. It eleminates the problem of placing cable and wiring. The transfer of data uses half-duplex communication system, and uses RS-232C as an interface. The system has accessed to the data with centralized polling. The transmission can be in the form of both pieces of data and file transfering to other computers connected in the system. The microprocessor Z-80 is the server for the transmission of data or file to the terminal. The server will check with the terminal all the time whether the terminal wants to send any data or file. This is similar to the computer network system. The system also has the resident program for immediate interruption so that the transmission of data can be done immediately. There is no need to wait for the computer while other program is in process. After the transmission has completed, the computer will go back the program which was in process previously without interfering with that program.

สารบัญ

บทที่ 1

บทนำ

การติดต่อสื่อสารข้อมูลระหว่างคอมพิวเตอร์นั้น ปกติสามารถกระทำได้โดยวิธีการเชื่อมโยงสายเข้าด้วยกันโดยตรง ดังจะแบ่งวิธีการส่งตามลักษณะการเดินทางของข้อมูลได้เป็น 2 วิธี คือ

1.การส่งข้อมูลแบบขนาน ข้อมูลจะถูกส่งออกไปทีเดียวทั้งไบท์ โดยบิทที่อยู่ใน 1 ไบท์ จะถูกส่งออกไปพร้อมๆกัน ซึ่งจำเป็นที่จะต้องใช้ช่องสัญญาณในการส่งอย่างน้อยเท่ากับ 1 ไบท์ เช่น IEEE-488

2.การส่งข้อมูลแบบอนุกรม ข้อมูลจะถูกส่งออกไปทีละ 1 บิท ช่องสัญญาณ ในการส่งอาจจะใช้เพียง 1 ช่องสญญาณเท่านั้น เช่น อินเตอร์เฟส EIA RS-232C อินเตอร์เฟส RS-422A

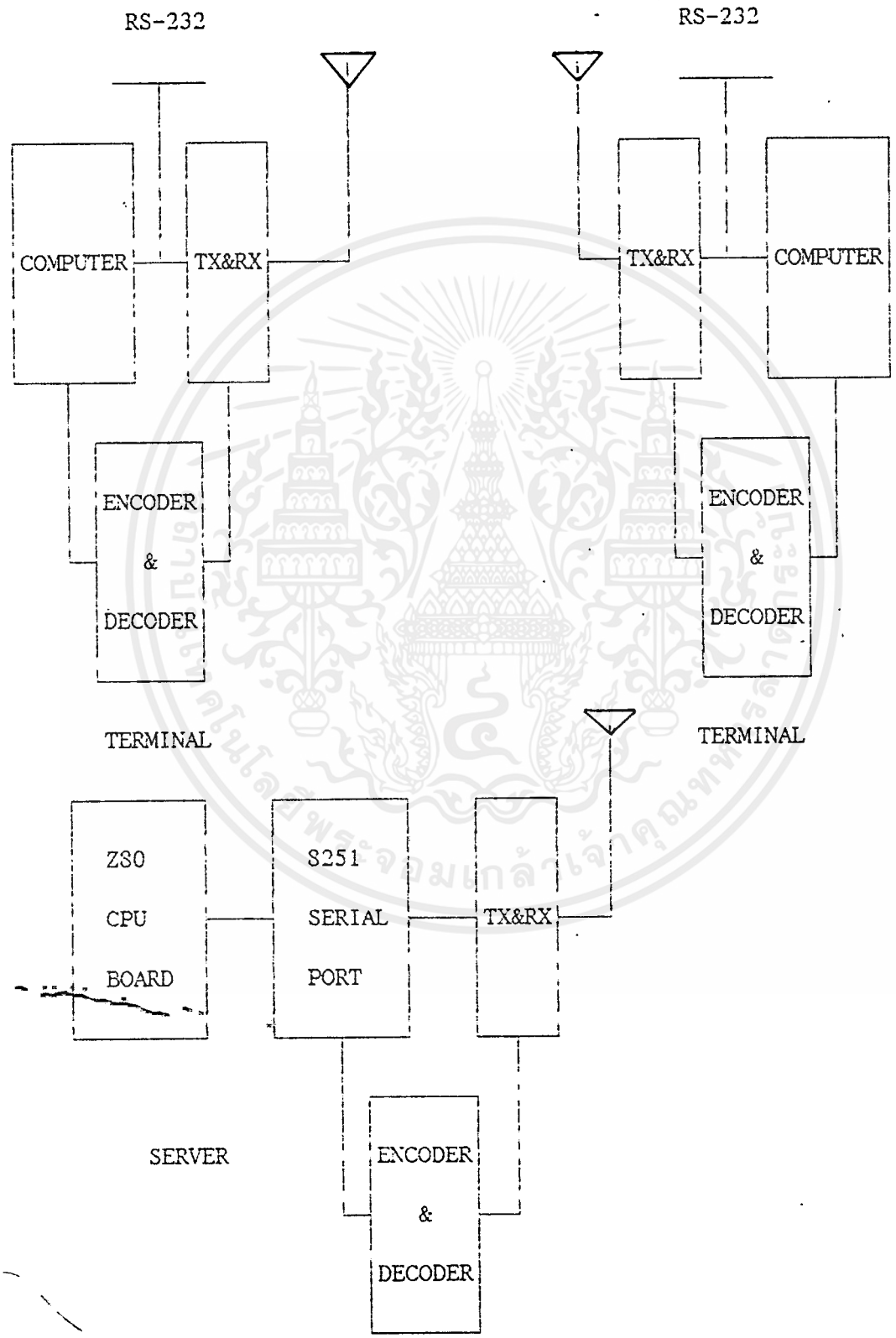
ด้วยเหตุที่ว่า การสื่อสารข้อมูลระหว่างคอมพิวเตอร์โดยใช้สายเชื่อมต่อกันโดยตรงนั้นไม่สามารถกระทำได้ในระยะทางไกลๆ เนื่องจากสภาพความเป็นตัวเก็บประจุที่เกิดขึ้นในสาย(stray capacitance) เมื่อนำมาใช้ในการส่งข้อมูลที่มีความเร็วสูง อาจจะเกิดการผิดพลาดของข้อมูลขึ้นได้เมื่อถึงปลายทาง ประกอบกับความเจริญก้าวหน้าทางด้านคอมพิวเตอร์ พัฒนาเป็นโครงข่ายคอมพิวเตอร์(COMPUTER NETWORK) โดยที่คอมพิวเตอร์จะต้องอาศัยข้อมูลจากคอมพิวเตอร์ตัวอื่นที่ไกลออกไป จึงจำเป็นต้องหาวิธีอื่น เพื่อให้การติดต่อสื่อสารในระยะทางไกลบรรลุผลได้ วิธีหนึ่งที่เหมาะสมสำหรับโครงข่ายคอมพิวเตอร์ก็คือ ทำการสื่อสารข้อมูลผ่านสายโทรศัพท์ ทั้งนี้เป็นเพราะว่า สายโทรศัพท์วางเป็นโครงข่ายครอบคลุมพื้นที่ทั่วไปอยู่แล้ว แต่ปัญหาจากการใช้ข่ายสายโทรศัพท์นี้ยังมีอยู่ เนื่องจากช่องสัญญาณโทรศัพท์สามารถส่งผ่านสัญญาณที่มีความถี่สูงสุดได้ประมาณไม่เกิน 4 kHz จึงต้องพยายามทำให้สัญญาณข้อมูลมีแถบกว้างความถี่ (BANDWIDTH) เหมาะสมกับสายโทรศัพท์ โดยใช้วิธีแปลงสัญญาณข้อมูลซึ่งมีลักษณะเป็นสัญญาณดิจิตอลให้เป็นสัญญาณเสียงพูด เรียกวิธีนี้ว่า โมดูเลชั่น (MODULATION) อุปกรณ์ที่ทำหน้าที่นี้คือ โมเด็ม (MODEM)

การวางโครงข่ายสายโทรศัพท์นั้นยังไม่สามารถวางได้ทั่วถึงทุกจุด ปัญหานี้สามารถแก้ไขได้โดยการนำ"ระบบวิทยุ"เข้ามาช่วย เพราะระบบวิทยุสามารถทำการติดต่อได้เป็นระยะทางไกลๆ โดยไม่มีปัญหาในเรื่องของการวางสาย

การส่งข้อมูลผ่านสายโทรศัพท์และระบบวิทยุนั้น และทำการส่งข้อมูลแบบอนุกรม เพื่อเป็นการประหยัดช่องสัญญาณ เพราะใช้เพียงแค่ 1 หรือ 2 ช่องสัญญาณเท่านั้น

โครงงานนี้เป็นโครงงานการสื่อสารข้อมูลผ่านระบบวิทยุ และทำการแปลงสัญญาณข้อมูลเป็น

สัญญาณความถี่ที่มีการโมดูเลชั่นแบบ แอมปลิจูดชิฟคีย์อิ่ง(Amplitude shift keying modulation)โดยระบบจะมีไมโครโปรเซสเซอร์เป็นตัว SERVER และมีเครื่องคอมพิวเตอร์เป็นตัว TERMINAL ซึ่งสื่อกลางของการติดต่อทั้งหมดเป็นคลื่นความถี่วิทยุ โดยมีการติดต่อในระบบโครงข่ายแบบ star เราสามารถเขียนระบบการติดต่อนี้เป็น BLOCK DIAGRAM ได้ดังรูป



เริ่มต้นจากตัว server จะต้องคอยตรวจสอบตัว terminal ว่าต้องการติดต่อกับเครื่อง terminal ตัวใดตัวหนึ่งหรือไม่ โดยการใช้หลักการของ เอ็นโค๊ดเดอร์และดีโคดเดอร์ คือจะส่ง ADDRESS ของตัว terminal ที่ต้องการจะติดต่อด้วย (เลือกโดย software)ไปเข้ารหัสโดยใช้ IC ENCODERจากนั้นก็จะนำสัญญาณจากตัวเข้ารหัสนี้ไป Modulate กับคลื่นพาห์ส่งออกอากาศไป

ที่ตัว terminal ทุกตัวจะสามารถรับสัญญาณที่ส่งออกไปนี้ได้ จากนั้นตัว terminalจะนำสัญญาณนี้ไปทำการ Demodulate แล้วนำสัญญาณนี้ไปถอดรหัสก็โดยใช้ IC DECODER ซึ่งเราตั้งรหัสไว้ก่อนแล้ว ตัวถอดรหัสก็จะทำการเปรียบเทียบรหัสของสัญญาณที่เข้ามากับรหัสที่ตั้งเอาไว้ว่าตรงกันหรือไม่ ถ้าตรงกัน ตัวถอดรหัสก็จะส่ง output ออกมาค่าหนึ่งซึ่งเราจะนำเอา output นี้ไปใช้ในการ INTERRUPT CPU ของตัว terminal ตัวนั้น เมื่อ COMPUTER ถูก INTERRUPT แล้วจะส่งสัญญาณ ACK ตอบรับการติดต่อไปยังเครื่อง server แล้วจากนั้น ก็จะมีการรับส่งข้อมูลกัน โดยผ่านทางอินเตอร์เฟส EIA RS-232 ในขณะที่เครื่องserver ตัวอื่น ๆ ก็จะสามารถทำงานอย่างอื่นได้โดยไม่เกี่ยวข้องกับการติดต่อครั้งนี้และยังสามารถตั้ง priority ของเครื่อง terminal แต่ละตัวได้ และที่ตัว terminal จะมีการเขียนโปรแกรมฝังตัว เพื่อให้การรับส่งข้อมูลทำได้ทันทีโดยไม่ต้องรอให้เครื่อง terminal ว่างจากการทำงานโปรแกรมอื่นก่อน เมื่อทำการรับส่งเสร็จ ก็จะกลับไปทำงานที่โปรแกรมเดิม โดยไม่ทำให้การทำงานของโปรแกรมเดิมเสียสถานะไป พร้อมทั้งยังมีการใส่โปรแกรม watchdog ทั้งเครื่อง server และเครื่อง terminal เพื่อเป็นการรีเซ็ตตัวเอง เมื่อมีการผิดพลาดขึ้นในขณะการรับส่ง ทำให้เกิดการหยุดชะงัก(hang)ขึ้นในระบบ

การติดต่อระหว่างโครงข่ายคอมพิวเตอร์นี้ มีการเข้าถึงข้อมูลแบบcentralized polling โดยจะต้องมีการใช้ PROTOCOL มาตรฐานของระบบด้วยซึ่งจะกล่าวรายละเอียดในบทต่อไป

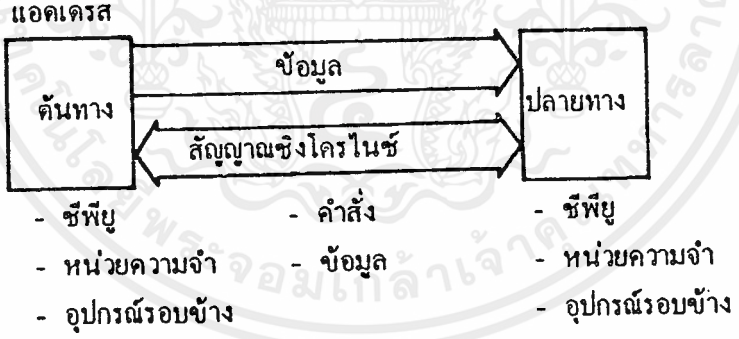
บทที่ 2

ทฤษฎีการรับส่งข้อมูลแบบอนุกรม

2.1 การอินเตอร์เฟสไมโครโปรเซสเซอร์

ดังที่ได้ทราบมาแล้วว่าขบวนการทางคณิตศาสตร์และลอจิกของระบบคอมพิวเตอร์เกิดขึ้นจากการทำงานของหน่วยประมวลผลกลาง ซึ่งเป็นวงจรรวม(Integrated Circiut:IC)หรือที่เรียกว่า ซีพียู (Central Processing Unit:CPU) นั่นเอง โดยไอซีตัวนี้จะติดตั้งอยู่บนแผ่นวงจรร่วมกับอุปกรณ์อื่นๆ เช่น หน่วยความจำ ROM (Read Only Memory), RAM (Ramdom Access Memory) และอุปกรณ์อื่นๆ ดังนั้นการอินเตอร์เฟสไมโครโปรเซสเซอร์ก็คือ การทำงานร่วมกันระหว่าง ซีพียูกับอุปกรณ์อื่นๆ ในการโอนย้ายข้อมูลระหว่างอุปกรณ์ต่างๆ บนแผ่นวงจรนั่นเอง

นอกจาก ซีพียู จะต้องทำงานสอดคล้องกับ ROM,RAM แล้วยังต้องอินเตอร์เฟสเข้ากับอุปกรณ์อินพุท,เอาว์พุท ต่างๆ อีกด้วย เพื่อเพิ่มประสิทธภาพการทำงานให้สมบูรณ์ยิ่งขึ้น ในขบวนการต่างๆ ของการทำงานร่วมกันของอุปกรณ์อิเลคทรอนิคส์ จะต้องเนื่องกันเป็นแบบลูกโซ่ เช่นในการส่งข้อมูลจากซีพียูไปยังอุปกรณ์อื่น เป็นต้น



รูปที่ 2.1 การอินเตอร์เฟสคอมพิวเตอร์

ข้อมูลที่จะโอนย้ายทุกตัวจะต้องมีแหล่งส่งข้อมูล และแหล่งรับข้อมูลเสมอ ซึ่งในขบวนการต่างๆ จะมีหลักสำคัญอยู่ว่า ข้อมูลนั้นเป็นแอดเดรส หรือคาต้า จะส่งไปยังจุดไหน เช่นส่งไปยังหน่วยความจำ หรืออุปกรณ์ I/O และจะส่งเมื่อไหร่ ขบวนการเหล่านี้ในขบวนการทั่วๆไปจะต้องมีสัญญาณในการตรวจสอบอุปกรณ์พร้อมที่จะส่งหรือรับข้อมูล หรือก่อนเสมอ ซึ่งจุดส่งข้อมูลและจุดรับข้อมูลจะต้องมีสัญญาณตรวจสอบความพร้อมเสมอเพื่อให้ข้อมูลมีการใช้งานอย่างเป็นระเบียบ เช่น ส่งจากซีพียูไปยังอุปกรณ์รอบข้าง เป็นต้น ซึ่งจุดรับส่งคู่หนึ่งๆ อาจจะเป็นระหว่างซีพียูด้วยกัน,ซีพียูกับหน่วยความจำ, ซีพียูกับอุปกรณ์รอบข้าง,ระหว่างอุปกรณ์รอบข้างด้วยกัน,หรือระหว่างอุปกรณ์รอบข้างกับหน่วยความจำ

ก็ได้ โดยข้อมูลที่รอนย้ายไปมานั้นอยู่ในลักษณะของเลขฐานสอง เช่น 0011010 เลขแต่ละตัวแทนด้วย 1 บิท(bit) อาจจะเป็น 8 บิท หรือ 16 บิท ก็ขึ้นอยู่กับระบบนั้นๆ

2.2 <u>การส่งข้อมูลแบบขนานและอนุกรม</u>

โดยทั่วๆไปหลักใหญ่ของการส่งข้อมูลในคอมพิวเตอร์ หรือคอมพิวเตอร์ด้วยกันจะมีลักษณะการส่งข้อมูลอยู่ 2 แบบคือ ส่งแบบขนาน และส่งแบบอนุกรม ดังกล่าวมาแล้วว่าคำสั่งหรือข้อมูลในรูปของบิท คือหลายๆ บิทประกอบกันเป็นคำๆ หนึ่ง(word) หรือคำสั่งหนึ่งๆดังในรูป 3.2ได้แสดงถึงกลุ่มของบิทที่มีการใช้งานในไมโครคอมพิวเตอร์ โดยในการกำหนดแอดเดรสของหน่วยความจำหรือการเขียนคำสั่ง และขบวนการอื่นๆ ล้วนแต่ต้องแปลงให้อยู่ในรูปของเลขฐานยี่กับเลขหนึ่งเสมอจึงจะทำให้ซีพียูรับรู้ และปฏิบัติตามได้ จึงได้มีการกำหนดลักษณะมาตรฐานของข้อมูลได้ดังนี้

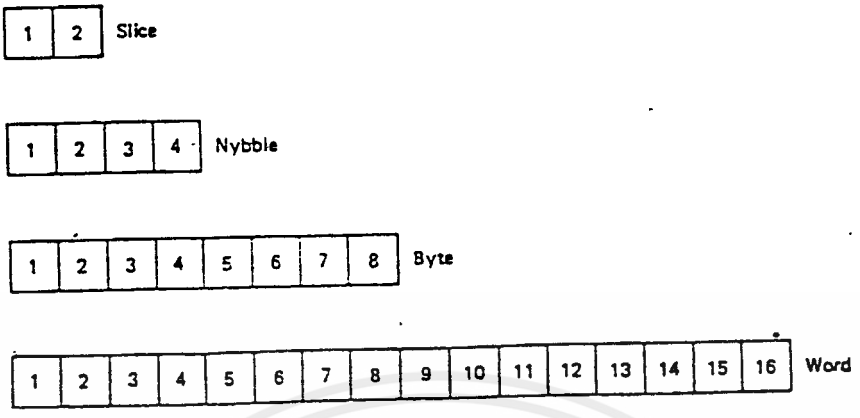
"ถ้าข้อมูลหนึ่งตัวเมื่อแปลงให้อยู่ในรูปของเลขฐานสองแล้ว ประกอบด้วย 4 บิท เราเรียกว่า 4 บิทไมโครหรือ 1 นิบเบิล(Nybble)"

"และถ้าข้อมูลประกอบด้วยกลุ่มของบิทที่มี 8 บิท เราเรียกว่าเป็น 1ไบท์(byte)"เป็นต้น แต่ในระบบอื่นอาจจะมี 16บิท หรือ 32บิท เป็น 1ไบท์ก็ได้

เพราะฉะนั้นเมื่อเรารู้ลักษณะของข้อมูลแล้ว ต่อไปเราจะมาดูถึงข้อแตกต่างระหว่างการส่งข้อมูลในแบบขนาน และอนุกรมว่าเป็นอย่างไร

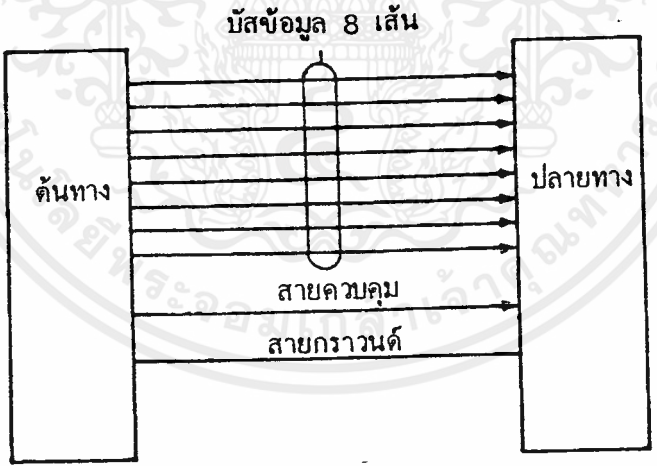
การส่งข้อมูลแบบอนุกรม : ข้อมูลแต่ละบิทจะถูกส่งเรียงกันออกไปเป็นลำดับ ต่อเนื่องกันทีละบิท เช่น ถ้าข้อมูลเป็น 1010 : เลข 0 ทางขวามือสุดซึ่งเป็น LSB(Least Significant Bit) ส่งออกไปก่อน ตามด้วยบิทที่สองคือเลข 1 และบิทที่3 คือเลข 0 และบิทสุดท้ายคือเลข 1 ซึ่งเป็น MSB (Most Significant Bit)ตามลำดับ โดยสายส่งข้อมูลจะมีอยู่เพียงเส้นเดียวเท่านั้น

การส่งข้อมูลแบบขนาน : ข้อมูลทุกๆบิทจะถูกส่งออกไปพร้อมๆกันในครั้งเดียว เช่น ถ้าข้อมูลเป็น 1010 ทั้งสี่บิทจะถูกส่งออกไปพร้อมกัน โดยผ่านสายส่งข้อมูลที่มี 4เส้น โดยแต่ละบิทจะส่งในสายส่งคนละเส้น



รูปที่ 2.2 แสดงรูปแบบของข้อมูล

ในระบบไมโครคอมพิวเตอร์ทั่วไป การส่งผ่านข้อมูลระหว่างอุปกรณ์ต่างๆ ที่อยู่บนแผ่นวงจรเดียวกัน จะส่งในแบบขนานทั้งสิ้น ถ้าเป็นไมโครคอมพิวเตอร์ขนาด ๘ บิท สายส่งข้อมูลภายในก็จะมี ๘ เส้น ซึ่งเรียกว่า "บัสข้อมูล" (data bus) นอกจากบัสข้อมูลทั้ง ๘ เส้นนี้แล้วยังต้องมีสายส่งข้อมูลอย่างน้อยอีก ๒ เส้น ร่วมกันอีกด้วย ซึ่งจะใช้เป็นสาย data ready และสายกราวนด์ดังรูปที่



รูปที่ 2.3 ระบบการส่งข้อมูลแบบขนาน

ซึ่งความจำเป็นของสายกราวนด์นี้มีไว้เพื่อใช้เป็นจุดอ้างอิงของศักดาไฟฟ้าเพื่อแสดงสถานะทางลอจิกของข้อมูลจะมีเพียง ๐ กับ ๑ เท่านั้น สำหรับสาย data ready นั้นจะบอกถึงความพร้อมของจุดส่งข้อมูลว่า ขณะนี้มีข้อมูลพร้อมที่จะส่งแล้ว ทางจุดรับจะตอบรับต่อสัญญาณนี้ว่าพร้อมที่จะรับหรือไม่ ซึ่งจะต้องมีการตรวจสอบสัญญาณเหล่านี้ก่อนเสมอ ถึงจุดนี้แล้วคงพอจะมองเห็นภาพของข้อมูลที่ถูกส่งแล้วว่าจะถูกส่งออกไปครั้งละ ๑ ไบท์ ตลอดเวลาจึงทำให้ปฏิบัติตามได้อย่างรวดเร็วถูกต้อง

ข้อที่ควรสังเกตจากรูป... อีกข้อหนึ่งก็คือ ในการส่งข้อมูลแบบขนานนี้ ควรจะต้องมีสัญญาณ

ควบคุมเพื่อใช้ควบคุมการปฏิบัติงานให้ถูกต้อง โดยสัญญาณควบคุมนี้จะส่งในทิศทางเดียวตลอดเวลา แต่ถ้าหากทิศทางของข้อมูลมีการแลกเปลี่ยนกันคือ ส่งกลับไปมาได้ทั้งสองทิศทางในลักษณะนี้สัญญาณควบคุมก็จะต้องเป็นลักษณะสองทิศทางด้วย จึงต้องใช้สายสัญญาณควบคุมถึง ๒ เส้นพร้อมกัน คือสัญญาณ IN และ OUT จะเป็นตัวชี้ว่า แชนแนลหรือช่องสัญญาณไหนที่ทำการส่งข้อมูลและช่องสัญญาณไหนที่รับข้อมูล เป็นต้น

สำหรับในระบบไมโครคอมพิวเตอร์ ๘ บิทนั้น ซีพียู Z-80 ถูกใช้แพร่หลายที่สุดในการใช้งาน ซีพียู Z-80 บัสข้อมูลทั้ง ๘ เส้นจะต่ออยู่กับรีจิสเตอร์ภายในของซีพียู โดยหน้าที่ของซีพียูนี้จะทำหน้าที่ในการแปลคำสั่งที่รับเข้ามา แล้วเปลี่ยนให้อยู่ในรูปของสัญญาณลอจิกส่งออกไปสั่งงานให้ขาต่างๆ ของตัวซีพียู(Z-80) มีขาที่ติดอยู่กับตัวถังทั้งหมด ๔๐ ขา ประกอบกันเรียกว่าแพ็กเกจปฏิบัติงานส่งสัญญาณออกไปให้อุปกรณ์ต่างๆ บนแผ่นวงจรพิมพ์ สำหรับโครงสร้างอื่นๆ ของซีพียูZ80 ก็ประกอบด้วยหน่วยความจำ หน่วยความจำของระบบนี้จะจัดเป็นกลุ่มๆ ละ 8 บิท,การส่งข้อมูลผ่านพอร์ท อินพุท/เอาท์พุตพอร์ทจะส่งครั้งละ ๘ บิท รีจิสเตอร์ภายในที่เป็นรีจิสเตอร์ใช้งานทั่วไปจะมีขนาด ๘ บิท แต่ในโครงสร้างภายในสามารถใช้รีจิสเตอร์ ๘ บิทสองตัวรวมเป็น ๑๖ บิทได้ เป็นต้น

ในปัจจุบันไมโครคอมพิวเตอร์ขนาด ๑๖ บิท และซุปเปอร์ไมโครคอมพิวเตอร์ตลอดจนถึงมินิคอมพิวเตอร์ และซุปเปอร์มินิคอมพิวเตอร์ได้เข้ามามีบทบาทในงานคอมพิวเตอร์มาก ดังนั้นหลักการต่างๆจึงสามารถประยุกต์ ใช้งานกับคอมพิวเตอร์เหล่านี้ได้

ในหัวข้อก่อนเราทราบแล้วว่าการส่งข้อมูลระหว่างซีพียูกับอุปกรณ์ต่างๆ บนแผ่นวงจรจะส่งเป็นแบบขนาน แต่ถ้าในกรณีที่เป็นการส่งข้อมูลจากไมโครคอมพิวเตอร์ไปยังอุปกรณ์รอบข้างต่างๆ การส่งข้อมูลมักจะเป็นการส่งแบบอนุกรมแทบทั้งสิ้น เพื่อเป็นการประหยัดค่าใช้จ่ายและสามารถได้ระยะทางไกลๆดังนั้นจึงใช้สายส่งข้อมูลเพียงเส้นเดียว ส่วนสายสัญญาณที่เหลือจะเป็นสายส่งสัญญาณควบคุม และสายกราวน์ด

ซึ่งมาตรฐานสากลที่กำหนดขึ้นมาควบคุมขบวนการส่งข้อมูลแบบนี้ประกอบด้วยมาตรฐานของสัญญาณระดับต่างๆ เช่น ระดับสัญญาณที่ใช้อุปกรณ์ TTL (Transistor-Transistor Logic), ระดับสัญญาณที่ใช้กับมาตรฐานของ EIA RA-232 และระดับสัญญาณที่ใช้รับระบบ 20mA Currentloop มาตรฐานเหล่านี้ใช้กันแพร่หลายในการส่งข้อมูลแบบอนุกรมในระบบไมโครคอมพิวเตอร์โดยเฉพาะมาตรฐาน RS-232C

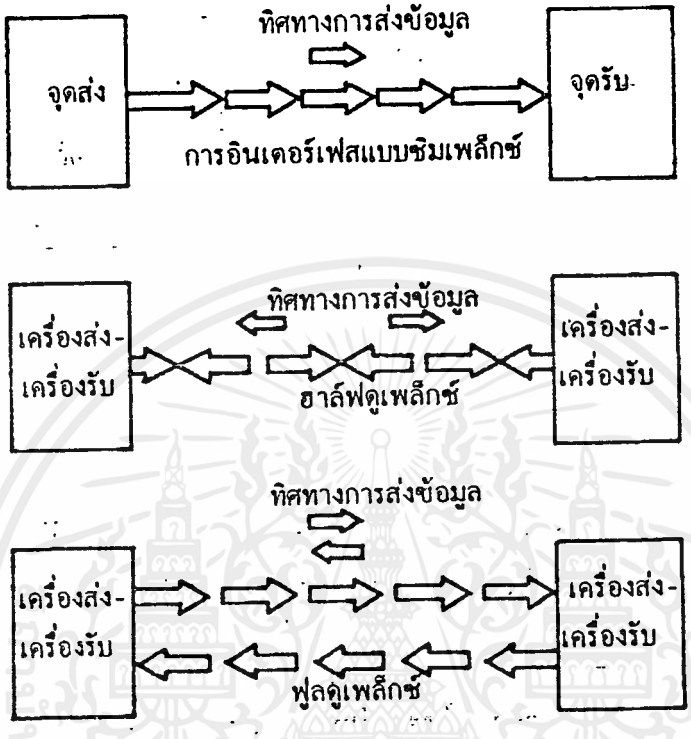
ข้อเปรียบเทียบระหว่างการส่งข้อมูลแบบขนานและแบบอนุกรม

	แบบขนาน	แบบอนุกรม
๑. ระยะทาง	ปกติจะน้อยกว่า ๑๐๐ ฟุต	ส่งได้ตั้งแต่ระยะสั้นๆจนถึงระนะทางเป็นไมล์
๒. ความเร็ว	อัตราความเร็วสูงมากในระยะที่ไม่ไกลมากนัก กำหนดได้เป็นจำนวนบิทต่อวินาที	อัตราความเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วง ๐ ถึง ๒ ล้านบิทต่อวินาที
๓.ระดับของสัญญาณ	ในการอินเตอร์เฟสจะใช้ระดับสัญญาณที่ใช้กับอุปกรณ์ TTL คือสัญญาณลอจิก ๑ และ ๐ จะแทนด้วยระดับแรงดัน +5V และ 0V ตามลำดับ	ใช้มาตรฐานของ EIA-RS232C คือมีระดับสัญญาณไฟฟ้าขนาด 12V หรืออาจจะใช้มาตรฐาน 20mA currentloop หรืออาจจะใช้ระดับสัญญาณของ TTL ก็ได้(ใช้กันน้อยมาก)
๔.ความผิดพลาดของสัญญาณ	ถ้าส่งในระยะทางไกลๆ ความผิดพลาดของข้อมูลจะเกิดขึ้นง่าย	ความผิดพลาดของสัญญาณจะมีน้อยหลัง
๕.ค่าใช้จ่าย	ถ้าส่งในระยะทางไกลๆจะสิ้นเปลืองค่าใช้จ่ายมาก เพราะต้องใช้สายส่งสัญญาณหลายเส้น	สิ้นเปลืองน้อยกว่าหลายเท่า ถึงแม้ว่าจะใช้อุปกรณ์เปลี่ยนสัญญาณของข้อมูลจากแบบขนานไปเป็นแบบอนุกรม แล้วส่งผ่านสายส่งใช้อุปกรณ์ในการแปลงสัญญาณกลับมาเป็นแบบขนานอีก ก็ยังลงทุนน้อยกว่า

ในระบบการสื่อสารข้อมูลนั้น ข้อมูลที่ส่งออกไปจะอยู่ในรูปของสัญญาณไฟฟ้า (เป็นสัญญาณอนาล็อก) วิ่งผ่านไปตามสายส่ง ซึ่งมักจะเกิดปัญหาเรื่องเกิดความผิดพลาดของสัญญาณขึ้นมาซึ่งปัจจัยอย่างหนึ่งที่เป็นสาเหตุก็คือ ระยะทางที่ส่งผ่านข้อมูล ถ้าระยะทางยิ่งไกลความผิดพลาดของข้อมูลก็จะเกิดพลาดเป็นเงาตามตัว ฉะนั้นในการแก้ไขจุดบกพร่องนี้วิธีต่างๆ ที่ใช้แก้ปัญหานี้ก็คือ ส่งสัญญาณข้อมูลที่มีความแรงมากๆ ออกไป หรือมีการใช้วงจรขยายข้อมูลเป็นช่วงๆ เพื่อให้ความแรงของสัญญาณคงที่ตลอดเวลา เป็นต้น ซึ่งการขยายสัญญาณที่เป็นแบบอนุกรมจะยุ่งยากน้อยกว่าแบบขนานมาก นอกจากนี้ปัญหาอื่นๆ ที่พบอยู่เสมอก็ได้แก่ เฟสของสัญญาณและปัญหาเกี่ยวกับการหน่วงเวลาของสัญญาณข้อมูล ซึ่งล้วนแต่มีผลให้ข้อมูลที่รับได้ทางปลายทางผิดพลาดได้และในการส่งข้อมูลแบบขนานมัก

จะพบกับปัญหาการบิดเบือนของสัญญาณ (Skewing) เสมอ

2.3 ทิศทางของการส่งข้อมูล



รูปที่ 2.4 การส่งข้อมูลในลักษณะต่างๆ

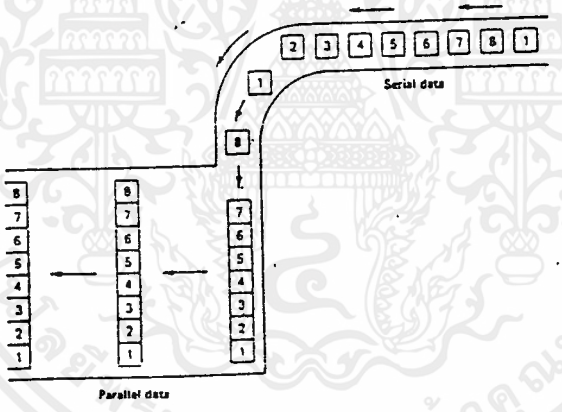
จากรูปที่2.4จะเห็นว่ามีทิศทางของการส่งข้อมูลต่างๆกัน 3 ลักษณะ คือ แบบทิศทางเดียวหรือซิมเพล็ก,แบบสองทางแต่รับกันไม่ได้ หรือฮาล์ฟดูเพล็กซ์ และแบบที่สามเป็นแบบส่งข้อมูลได้สองทางและรับตอบกันได้ในเวลาเดียวกัน หรือแบบฟูลดูเพล็กซ์

วิธีที่ง่ายที่สุดก็คือ การส่งข้อมูลในทิศทางเดียว เช่น ส่งข้อมูลจากเทอร์มินอล A ไปยังเทอร์มินอล B ในกรณีนี้เทอร์มินอล A จะต้องเป็นเครื่องส่งและ B จะต้องเป็นเครื่องรับเท่านั้น ส่วนในกรณีที่สองจะแตกต่างจากกรณีแรกคือ เป็นการส่งในลักษณะที่เทอร์มินอล A และ B สามารถทำหน้าที่ได้ทั้งเครื่องรับและเครื่องส่ง เช่น เทอร์มินอล A ส่งไปให้เทอร์มินอล B ได้และB ก็ส่งข้อมูลตอบกลับมาให้ A ได้เช่นกัน แต่ตัวส่งอยู่คนละช่วงเวลา ซึ่งต่างจากวิธีซิมเพล็กที่เทอร์มินอล B ทำหน้าที่เป็นฝ่ายรับข่าวสารได้เพียงอย่างเดียวเท่านั้น ในลักษณะที่สามวิธีนี้เรียกว่าเป็นแบบฟูลดูเพล็กซ์ จากรูปที่ 2.4เราจะเห็นความแตกต่างจากสองกรณีแรกได้ว่า เทอร์มินอล A และ B สามารถจะรับตอบกันได้คือ เป็นเครื่องรับและเครื่องส่งในเวลาเดียวกันได้คือเทอร์มินอล A ส่งข้อมูลไปให้

เทอร์มินอล B ซึ่งเป็นเวลาเดียวกันที่เทอร์มินอล B ก็ส่งข้อมูลไปให้เทอร์มินอล A เทอร์มินอล A และ B จะมีการ ทำงานที่เป็นอิสระต่อกัน และสายส่งข้อมูลที่ใช้ติดต่อระหว่างเทอร์มินอล A และ Bจะประกอบไปด้วย สายส่ง 3 เส้นคือ สายสัญญาณรับส่งและกราวด์ โดยระบบของเทอร์มินอล A และ B จะมีสายส่งร่วมกัน จะเรียกว่าเป็นฟูลดูเพล็ก 4- เส้น (Full duplex 2-wires) ส่วนอีกระบบหนึ่งจะใช้สายสัญญาณส่งและรับร่วมกัน และในระบบจะมีสายสัญญาณเพียง 2 เส้น ใน ลักษณะนี้เรียกว่าเป็นระบบฟูลดูเพล็กซ์ 2-เส้น (Full duplex 2-wires) เป็นต้น

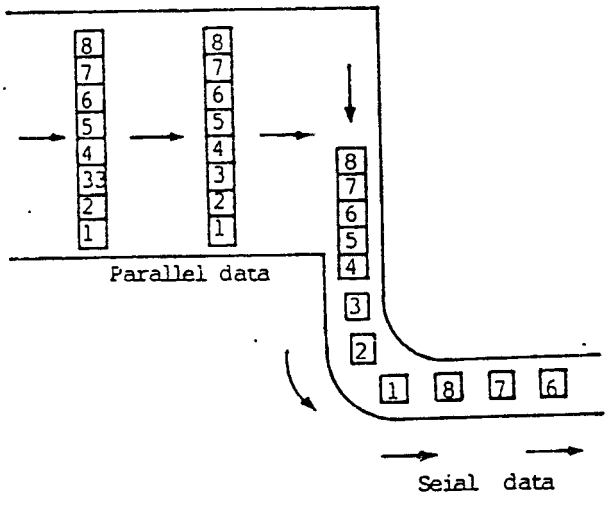
2.4 <u>การแปลงรูปแบบข้อมูลแบบขนานไปเป็นแบบอนุกรม</u>

หลักการทำงานของการแปลงรูปแบบข้อมูลนี้ก็โดยอาศัย ชิฟท์จิสเตอร์โดยมีหลักการดังนี้คือข้อมูล ที่ส่งข้อมูลเข้ามาจะเป็นแบบอนุกรม โดยส่งเข้ามาทีละบิทเมื่อเข้ามาถึงรีจิสเตอร์ บิทที่ทยอยกันเข้า มาก็จะถูกจัดเก็บเรียงกันอยู่ในรีจิสเตอร์ จนกระทั่งครบจำนวนบิทที่ต้องการ รีจิสเตอร์ก็จะส่งข้อมูลนั้น ออกไป ดังในรูปที่ 2.5



รูปที่ 2.5 การแปลงรูปข้อมูลจากอนุกรมไปเป็นแบบขนาน

และเมื่อต้องการจะแปลงรูปแบบข้อมูลกลับไปเป็นแบบอนุกรมอีก ก็ทำตามขบวนการที่ตรงกัน ข้ามดังในรูปที่2.6



รูปที่ 2.6 การแปลงข้อมูลจากแบบขนานไปเป็นแบบอนุกรม

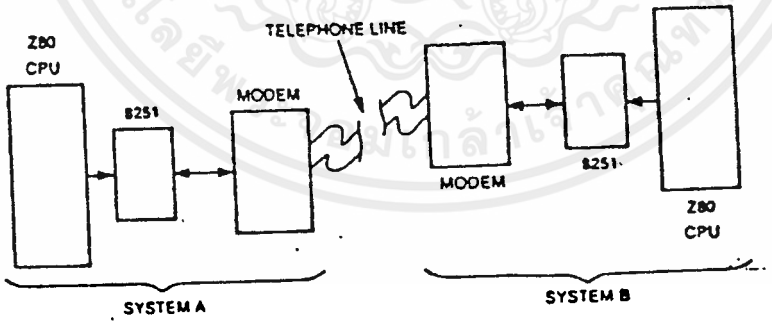
ในการส่งข้อมูลลักษณะนี้จำเป็นจะต้องรู้จักคำ 3 คำต่อไปนี้คือ

.UART : Universal Asynchronous Receiver-Transmitter

.USART: Universal Synchronous Asynchronous Receiver-Transmitter

.SIO : Serial input/output circuit

ซึ่งอุปกรณ์เหล่านี้นอกจากจะต้องใช้งานร่วมกับซิพรีจิสเตอร์แล้ว ยังมีบระโยชน์ต่อระบบบัส, ความเร็วในการส่งข้อมูล, พาริตี้และฟอร์แมทของข้อมูล เป็นต้น ตัวอย่างของวงจรรวมที่ใช้งานในลักษณะนี้ก็เช่น 8251 USART ซึ่งใช้งานร่วมกับซีพียู Z-80 โดยเฉพาะเพื่อใช้สื่อสารข้อมูลผ่านโมเด็มโดยแสดงการอินเตอร์เฟสดังรูป



รูปที่ 2.7 การใช้งาน 8251 ร่วมกับโมเด็ม

2.5 การซิงค์โครไนซ์อุปกรณ์

ระบบ 2 ระบบที่ติดต่อสื่อสารกันอยู่จะรับส่งข้อมูลที่อยู่ในรูปสัญญาณไฟฟ้าสองระดับคือระดับ0และระดับ 1 ซึ่งเป็นระดับสัญญาณทางดิจิตอล ดังนั้นถ้าระบบ A ส่งระดับสัญญาณ 0 และ 1 ผ่านไปตามสาย และระบบ B จะสามารถรับสัญญาณได้โดยไม่คลาดเคลื่อนนั้น ทั้งสองระบบจะต้องมีเวลาการ

ส่ง/รับข้อมูลที่สอดคล้องกันตลอดเวลา ในการสื่อสารข้อมูลระดับสัญญาณ 0 และ 1 จะแทนข้อมูลต่างๆ เช่น ตัวอักษร,สัญลักษณ์,ตัวเลข และอื่นๆ ในระบบคอมพิวเตอร์จะมีการนำเอารหัสต่างๆมาใช้แทน เช่น รหัสแอสกี้ ซึ่งในการสื่อสารระหว่างระบบทั้งด้านรับและด้านส่งจะต้องรหัสมาตรฐานเดียวกันจึงจะสามารถติดต่อกันได้

.ระดับสัญญาณบวก/ลบ : ในการใช้งานอุปกรณ์ TTL (Transistor Transistor Logic) ระดับลอจิก 1 แทนด้วยระดับสัญญาณ +5 โวลท์ และระดับลอจิก 0 แทนระดับสัญญาณ0โวลท์ กรณีนี้จึงถือว่าเป็นสถานะสัญญาณแบบบวก ส่วนในการอินเตอร์เฟสแบบมาตรฐาน RS-232ระดับลอจิก 1 จะแทนด้วยระดับสัญญาณ -12 โวลท์ และระดับลอจิก 0 แทนด้วยระดับสัญญาณ +12 โวลท์ ในกรณีนี้ถือว่าเป็นสถานะแบบลบ

2.6 <u>การส่งข้อมูลแบบอะซิงค์โครนัสและซิงค์โครนัส</u>

2.6.1การรับส่งข้อมูลแบบซิงโครนัส

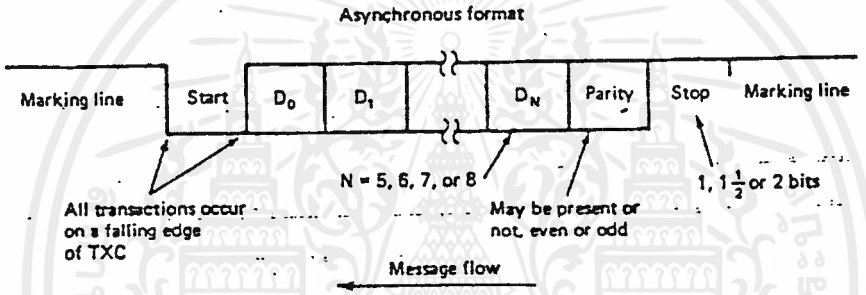
ไม่ว่าในการส่งข้อมูลจะเป็นแบบขนานหรืออนุกรม การส่งข้อมูลแบบซิงโครนัสก็คือระบบการส่งที่ข้อมูลแต่ละเวิร์ดถูกส่งออกไปตามเวลาที่แน่นอน ซึ่งหมายถึงระยะเวลาระหว่างข้อมูลแต่ละเวิร์ดที่ถูกส่งไปมีค่าแน่นอน

2.6.2 การรับส่งข้อมูลแบบอะซิงโครนัส

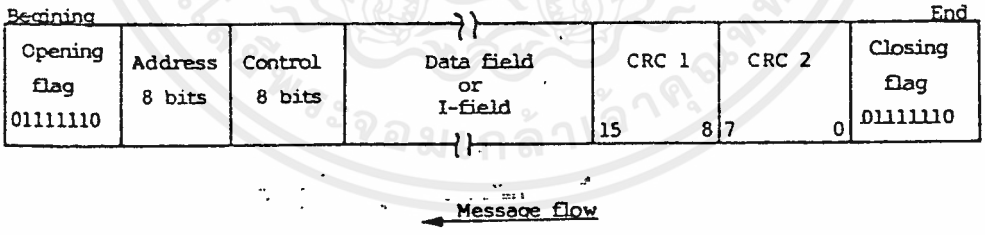
คือระบบการรับส่งข้อมูลที่แต่ละคำ ถูกส่งออกไปอย่างไม่มีกำหนดเวลาแน่นอน นั่นคือระยะเวลาระหว่างข้อมูลแต่ละคำที่ถูกส่งออกไปมีค่าไม่แน่นอน ตัวอย่างเช่น การส่งข้อมูลจากคอมพิวเตอร์ A ไปยังคอมพิวเตอร์ B ในการป้อนข้อมูลจากคีย์บอร์ดนั้นผู้ที่พิมพ์ดีดไม่สามารถพิมพ์ด้วยอัตราเร็วคงที่ได้แต่ละครั้งที่กดจะห่างไม่เท่ากัน ดังนั้นสิ่งที่กำหนดเวลาในการส่งข้อมูลก็คือ ความพร้อมเพรียงของเครื่องรับและเครื่องส่ง

ในการส่งข้อมูลแบบอะซิงโครนัสนั้น โครงสร้างของข้อมูลที่จะส่งจะมีลักษณะเป็นบล็อกๆ ซึ่งแต่ละบล็อกประกอบด้วยบิทเริ่มต้น (start bit) ส่วนของข้อมูลและบิทสิ้นสุดข้อมูล (stop bit) โดยบิทเริ่มต้นจะแสดงถึงการเริ่มต้นของกลุ่มข้อมูล แล้วตามด้วยส่วนของกลุ่มข้อมูล และบางกรณีอาจมีการเพิ่มพาริตี้บิทเพื่อใช้ตรวจสอบความถูกต้องของข้อมูล และบิทสิ้นสุดข้อมูลก็เป็นการบอกว่าข้อมูลในบล็อกๆ นี้หมดลงเพียงเท่านี้ ส่วนในการส่งข้อมูลแบบซิงโครนัสนั้นจะต้องมีการส่งสัญญาณไปพร้อมๆกับสัญญาณนาฬิกาไปพร้อมกับสัญญาณข้อมูล ในการส่งข้อมูลระยะสั้นๆ สัญญาณนาฬิกาซึ่งใช้เป็นสัญญาณซิงค์ อาจจะส่งแยกไปในการสายส่งสัญญาณอีกเส้นหนึ่งไม่ส่งรวมไปในสายส่งข้อมูลก็ได้

แต่ถ้าเป็นการสื่อสารในระยะไกลๆ แล้วสัญญาณนาฬิกาจะถูกเข้ารหัสส่งร่วมไปกับสัญญาณข้อมูลในสายส่งเส้นเดียวกัน และไม่ว่าจะเป็นการส่งข้อมูลแบบซิงโครนัสหรืออะซิงโครนัส ข้อมูลในบล็อกหนึ่งๆจะต้องมีบิทที่แสดงให้รู้ว่าเป็นการเริ่มต้น และสิ้นสุดข้อมูลซึ่งลักษณะของข้อมูลเช่นนี้เรากล่าวมาแล้วในการส่งข้อมูลแบบอะซิงโครนัส แต่ถ้าเป็นการส่งแบบซิงโครนัสจะไม่มีบิทเริ่มต้นและบิทหยุด แต่บิทที่อยู่ตัวท้ายของข้อมูลบล็อกหนึ่ง ๆ (บล็อกหนึ่ง ๆ ประกอบด้วยข้อมูลหลายชุด) จะแสดงถึงจุดเริ่มต้นและจุดสิ้นสุดของข้อมูลเท่านั้น เพราะฉะนั้นถ้าเป็นการส่งข้อมูลอนุกรมแบบอะซิงโครนัส เราจะเพิ่ม Framing bits รวมเข้าไปในแต่ละคาร์แรกเตอร์ และถ้าเป็นการส่งข้อมูลอนุกรมแบบซิงโครนัส เราจะเพิ่มFraming charactors เข้าไปร่วมกับบล็อกของข้อมูลแต่ละบล็อก ซึ่งแสดงความแตกต่างได้ดังรูปที่2.8 และรูปที่2.9



รูปที่2.8 การส่งข้อมูลอนุกรมแบบอะซิงโครนัส



รูปที่2.9 การส่งข้อมูลอนุกรมแบบซิงโครนัส

2.6.2.1Asynchronous Serial Link-Level Protocal

ในหัวข้อนี้จะกล่าวถึงโปรโตคอลของการส่งข้อมูลอนุกรมแบบอะซิงโครนัสโดยละเอียด ข้อมูลแต่ละคาร์แรกเตอร์ที่ส่งไปมีกำหนดเวลาไม่แน่นอน ดังนั้นหลักการของโปรโตคอลชนิดนี้จึงต้องมีการรีซิงโครไนซ์ก็คือ ความแตกต่างของสัญญาณนาฬิการะหว่างด้านส่งและด้านรับจะมีได้ไม่เกิน 10เปอร์เซ็นต์ ซึ่งหมายความว่าหลังจากที่ทางด้านส่งๆข้อมูลไปแล้ว 10 บิท โดยสมมติให้ทางด้านส่งมีความถี่ของสัญญาณนาฬิกาเร็วกว่าทางด้านรับ ทางด้านรับจะรับข้อมูลได้เกินมา 1 บิท คือรับได้จำนวนบิท

11 บิท หรือหมายความว่า ด้วยระยะเวลาของสัญญาณนาฬิกาเท่าๆกัน ทางด้านส่งจะส่งข้อมูลได้จำนวนบิทมากกว่า ดังนั้นในบล็อกๆ หนึ่งของข้อมูลที่ส่งไปจึงมีการเพิ่มบิทเริ่มต้น และบิทหยุดรวมเข้าไปตรงหัว/ท้ายของข้อมูล โดยปกติบิทเริ่มต้นจะมีเพียงบิทเดียว ในขณะที่บิทหยุดอาจจะมี 2 บิท หรือ 3 บิทก็ได้

- บิทเริ่มต้น

ในโปรโตคอลของการส่งข้อมูลอนุกรมแบบอะซิงโครนัส กำหนดให้สถานะมาร์ค (Marking State) เป็นสัญญาณลอจิก 1 เมื่อทางด้านส่งจะทำการส่งข้อมูลก็จะต้องส่งเป็นบิตเริ่มต้นโดยแทนด้วยสถานะสเปส (Space State) หรือสัญญาณลอจิก 0 จำนวน 1 บิทไปก่อน ซึ่งจะทำให้ทางด้านรับดิเท็กซ์สถานะของสายส่งได้ว่าขณะนั้นสายส่งกำลังมีข้อมูลส่งมา สำหรับปัญหาที่เกิดขึ้นและมีผลต่อสัญญาณข้อมูลคือสัญญาณสไปค์ (Spike) ซึ่งทำให้สถานะลอจิกของสายส่งมีช่วงเวลาสั้นเกินไปทำให้ทางด้านรับไม่สามารถดีเท็กซ์สถานะของสายส่งหรือสถานะของบิทเริ่มต้นได้ ดังนั้น ส่วนใหญ่ทางด้านรับจะมีส่วนของวงจรสไปค์ ดีเทคชั่น (Spike Detection) ที่ทำหน้าที่สุ่มจับสัญญาณสถานะของสายส่งด้วยความถี่ของการสุ่มค่าหนึ่งในระหว่างบิทต่อบิท ซึ่งอาจจะเป็น 2,4 หรือ 16 ครั้งในระหว่าง 1 บิทก็ได้

ดังนั้นสรุปได้ว่า หน้าที่ของบิทเริ่มต้นนั้นจะเป็นตัวบอกว่าข้อมูลเริ่มต้นตรงไหน และเมื่อใช้ร่วมกับบิทหยุด ซึ่งจะกล่าวถึงต่อไปก็จะทำให้ทราบได้ว่าข้อมูลสิ้นสุดตรงไหน โดยความกว้างของบิทเริ่มต้นนี้จะมีความกว้างเท่ากับ 1 บิทข้อมูล แสดงดังรูปที่ 2.13

- บิทข้อมูล

หลังจากที่ด้านรับสามารถดีเท็กซ์สัญญาณบิทเริ่มต้นได้แล้ว ก็จะทำการเซ็ทสถานะของชิฟรีจิสเตอร์ ให้พร้อมที่จะรับบิทข้อมูลได้ โดยบิทข้อมูลจะมีจำนวนบิทเป็น 5,6,7 หรือ 8 บิต ขึ้นกับจำนวนคาร์แรคเตอร์ที่ใช้ดังแสดงตามตารางต่อไปนี้

จำนวนบิทข้อมูลใน ๑ คาร์แรกเตอร์	จำนวนคาร์แรกเตอร์
5 บิท	32
6 บิท	64
7 บิท	128
8 บิท	256

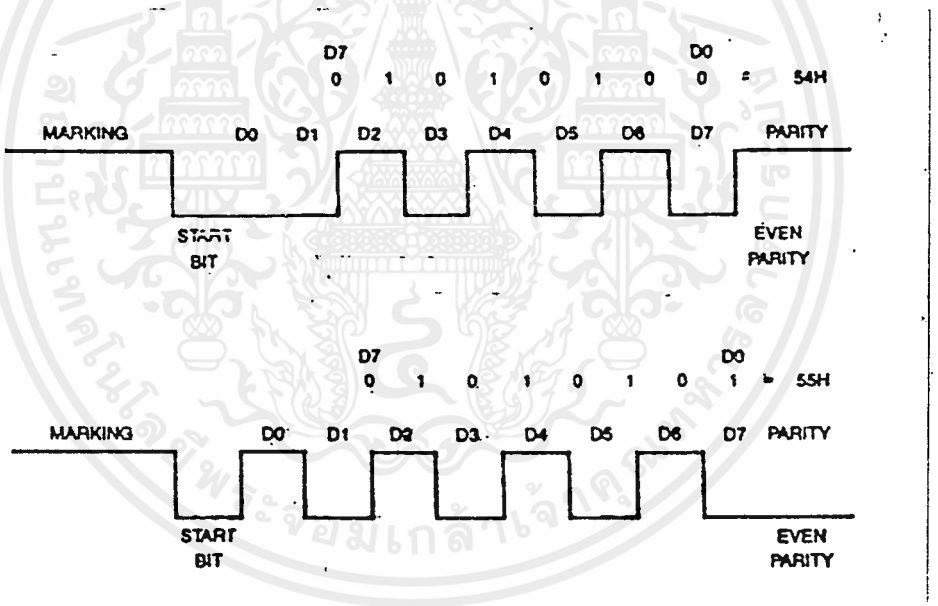
นอกจากนั้นรหัสต่างๆที่ใช้อาจจะแทนด้วย 5 บิท ซึ่งเป็นมาตรฐานของรหัส Baudotโดยประกอบด้วยกลุ่มของคาร์แรกเตอร์ต่างๆ จำนวน 32 คาร์แรกเตอร์ และถ้าเป็นรหัสขนาด7 บิทจะประ

กอบด้วยกลุ่มของคาร์แรกเตอร์ต่างๆ จำนวน 128 ตัว ซึ่งเป็นมาตรฐานของรหัสASCIIและใช้กันแพร่หลายมาก นอกจากนี้ก็ยังมีรหัสขนาด 8 บิท หรือมาตรฐานของรหัสEBCDICโดยมาตรฐานนี้ประกอบด้วยกลุ่มของกลุ่มของคาร์แรกเตอร์ 256 คาร์แรกเตอร์ เป็นต้น

- **บิทพาริตี้**

บิทนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่า ข้อมูลที่รับเข้ามาผิดหรือไม่ บิทพาริตี้นี้จะถูกส่งออกมาพร้อมกับบิทข้อมูล ซึ่งบิทนี้จะเป็น "1"หรือ "0" นั้นขึ้นกับข้อมูลที่ส่งออกมาว่ามีจำนวนบิทที่เป็น "1" เป็นจำนวนคู่ หรือ คี่ และยังขึ้นกับอมปกรณ์รับส่งข้อมูลด้วยว่าถูกออกแบบไว้ให้รับส่งบิทพาริตี้ในลักษณะของพาริตี้คู่ หรือ คี่อีกด้วย

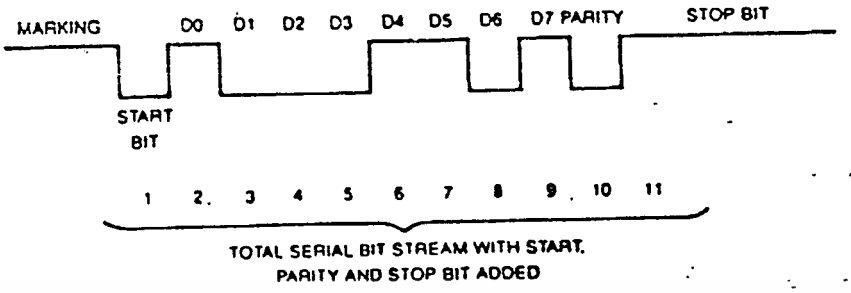
สิ่งสำคัญคือ ถ้าอุปกรณ์ส่งข้อมูลทำการส่งในลักษณะพาริตี้คู่ หรือคี่ก็ตามส่วนรับข้อมูลในลักษณะพาริตี้เดียวกับอุปกรณ์ส่งข้อมูลด้วย เช่น ในกรณีที่อุปกรณ์ส่งข้อมูลทำการส่งข้อมูลในลักษณะพาริตี้คู่อุปกรณ์รับข้อมูลก็จะต้องทำการรับข้อมูลในลักษณะของพาริตี้คู่ด้วย เป็นต้น ดังรูป



รูปที่2.10 การเพิ่มพาริตี้เข้าไปในข้อมูลแต่ละไบท์

- **บิทสิ้นสุดข้อมูล**(Stop bit)

บิทสุดท้ายที่เพิ่มเข้าไปนี้ จะใช้ในการตรวจสอบจุดสิ้นสุดของข้อมูล บิทนี้จะถูกเพิ่มเข้าไปหลังพาริตี้บิท ถ้าอุปกรณ์รับข้อมูลตรวจไม่พบบิทนี้ ก็แสดงว่าข้อมูลที่รับเข้านั้นเกิดข้อผิดพลาดขึ้น สำหรับบิทสิ้นสุดข้อมูลนี้อาจจะมีจำนวนบิทเป็น 1,1.5 หรือ 2 บิทก็ได้ ซึ่งสรุปได้ว่าข้อมูลที่ส่งออกมาในแต่ละไบท์นั้นไม่ใช้มีแต่ข้อมูล 8 บิท เท่านั้น แต่อาจจะมีได้ถึง 12 บิท หรือมากกว่านั้นก็ได้ ดังรูป



รูปที่ 2.11 รูปแบบของข้อมูลในแต่ละไบท์ในการรับส่งข้อมูลอนุกรมแบบอะซิงโครนัส

2.7 อุปกรณ์ที่ใช้ในการสื่อสารแบบอนุกรม

อุปกรณ์สนับสนุนหรือที่เรียกว่า ชิพซัพพอร์ท (ship support) ที่จำเป็นในการส่งข้อมูลแบบอนุกรมก็คือ SIO,UARTและUSARTโดยการทำงานของ UART นั้นเราสามารถเพิ่ม Framing bit เข้าไปรวมกับข้อมูลเพื่อใช้สำหรับการส่งข้อมูลแบบอะซิงโครนัส ส่วน USART เป็นพอร์ทที่ใช้สำหรับส่งข้อมูลแบบอนุกรม เช่น ไอซี 8251 USART สามารถทำงานได้ทั้งการส่งข้อมูลแบบซิงโครนัส และอะซิงโครนัส

สำหรับการทำงานของ USART ในโหมดซิงโครนัสนั้น จะทำหน้าที่ตรวจจับสัญญาณซิงค์คาร์แรกเตอร์ในกรณีที่เป็นเครื่องรับ และทำหน้าที่ส่งสัญญาณซิงค์คาร์แรกเตอร์ออกไปเมื่อซีพียูไม่ได้ส่งข้อมูล ในกรณีที่เป็นเครื่องส่ง การใช้งานUSART นี้ทำให้สามารถประหยัดอุปกรณ์สนับสนุนในการส่งผ่านข้อมูลได้มาก

2.8 การตรวจจับข้อผิดพลาด

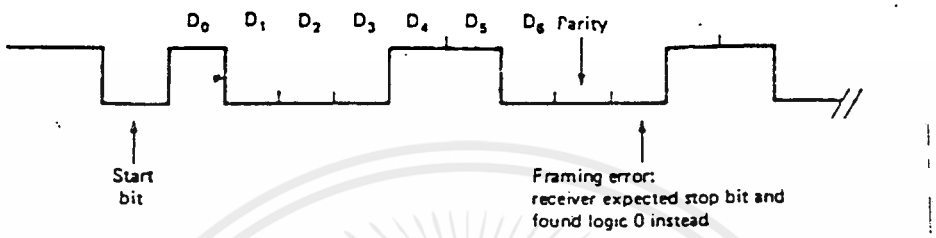
หน้าที่ที่สำคัญของวงจรสนับสนุนพอรท์ I/O ก็คือ การตรวจจับข้อผิดพลาด ซึ่งข้อผิดพลาดที่จะเกิดขึ้นได้มีหลายประกอบด้วยกัน เช่น

Framming error, Parity error และ Recciver Overrun Error เป็นต้น

2.8.1Framming Errors

การตรวจจับ Framming errors เกิดขึ้นดังนี้ เมื่ออุปกรณ์ด้านรับจะทำการรับสัญญาณของบิทสิ้นสุดข้อมูล (stop bit) แต่ปรากฏว่าสัญญาณที่รับได้เป็นสัญญาณลอจิก 0 แทนซึ่งในการส่งข้อ

มูลทางด้านรับจะรู้ว่า สัญญาณตัวแรกที่รับได้เป็นบิทเริ่มต้น (start bit) แล้วตามด้วยบิทข้อมูล (data bit) ซึ่งอาจจะมาบิทพาริตี้หรือไม่ก็ได้ แล้วสิ้นสุดด้วยบิทสิ้นสุดข้อมูล ดังนั้นเมื่อด้านรับตรวจพบว่า บิทสิ้นสุดข้อมูลที่รับได้ไม่เป็นตามที่กำหนดไว้ (คือรับได้เพียงลอจิก 0 ตัวเดียวซึ่งจริงๆ แล้วต้องเป็น 1) ด้านรับก็จะรู้ทันทีว่าเกิดข้อผิดพลาดของข้อมูลขึ้นแล้ว ดังแสดงในรูป



รูปที่ 2.12 เงื่อนไขการเกิด Framing Error

2.8.2 Parity Errors

ใน Framing Error ด้านรับจะตรวจจับข้อผิดพลาดได้ เพราะแทนที่ด้านรับจะรับสัญญาณสิ้นสุดข้อมูลได้ แต่กลับรับได้สัญญาณที่เป็นลอจิก 0 แทนสำหรับใน parity Error นี้ ทางด้านรับต้องสอดคล้องกับทางด้านส่งคือทางด้านส่งๆเป็นพาริตี้/คี่/หรือไม่มีพาริตี้ ทางด้านรับก็จะต้องมี การตรวจสอบไปตามนี้ โดยทำการเปรียบเทียบค่าของพาริตี้ที่รับได้กับค่าพาริตี้บิทของข้อมูลจริง ดังนั้นถ้าเปรียบเทียบค่าคู่นี้แล้วไม่ตรงกัน ก็จะรู้ได้ทันทีว่าเกิดข้อผิดพลาดขึ้นกับข้อมูลที่ส่งมา

2.8.3 Overrun Errors

เราทราบแล้วว่าการใช้งาน SIO นั้นจะต้องมีส่วนของบัฟเฟอร์เป็นแบบ FIFO (First In First out)หรือเป็นแบบเข้าก่อนออกก่อน สมมุติว่ารีจิสเตอร์ตัวนี้รับข้อมูลเต็มที่ได้ 5 ตัว เพราะฉะนั้นถ้าข้อมูลถูกส่งมาทั้งหมด 6 ตัว ข้อมูล 5 ตัวแรกจะถูกจัดเก็บไว้ภายในรีจิสเตอร์ก่อนแล้วถึงจะส่งต่อไปใช้งาน แต่ทว่าในขณะนั้นยังไม่มีข้อมูลตัวที่หกถูกย้ายออกไป จึงมีข้อมูลอีกตัวหนึ่งที่เหลือค้างอยู่ จึงทำให้เกิดข้อผิดพลาดขึ้น ดังนั้นในการใช้งานทั่วไปแล้ว การใช้งาน SIO จะต้องทำการแฮนด์เชค (Hand-shake) กันขึ้นระหว่างด้านส่งและด้านรับ เพื่อป้องกันการรับส่งข้อมูลที่ไม่ทัน และป้องกัน Overrun Errors ไม่ให้เกิดขึ้นอีกด้วย

บทที่ 3

ทฤษฎีการทำงานของโปรแกรมการรับส่งข้อมูลแบบอนุกรมและวงจรเครื่องรับส่ง

3.1 ทฤษฎีและการทำงานของไอซี 8251 USART

3.1.1 8251 กับพอร์ทอนุกรม

ไอซี 8251 Universal Synchronous Asynchronous Receiver/Transmitter เป็นอุปกรณ์ที่เปลี่ยนข้อมูลดิจิตอลแบบขนานจาก Z80 เข้าไปในไอซีที่ทีแอล โดยการส่งข้อมูลดิจิตอลแบบอนุกรม ซึ่งไอซี 8251 นี้จะให้สัญญาณไปควบคุมโมเด็ม, ตรวจสอบ error และ hand-shaking ข้อมูลที่จะเข้าไปสู่ Z80

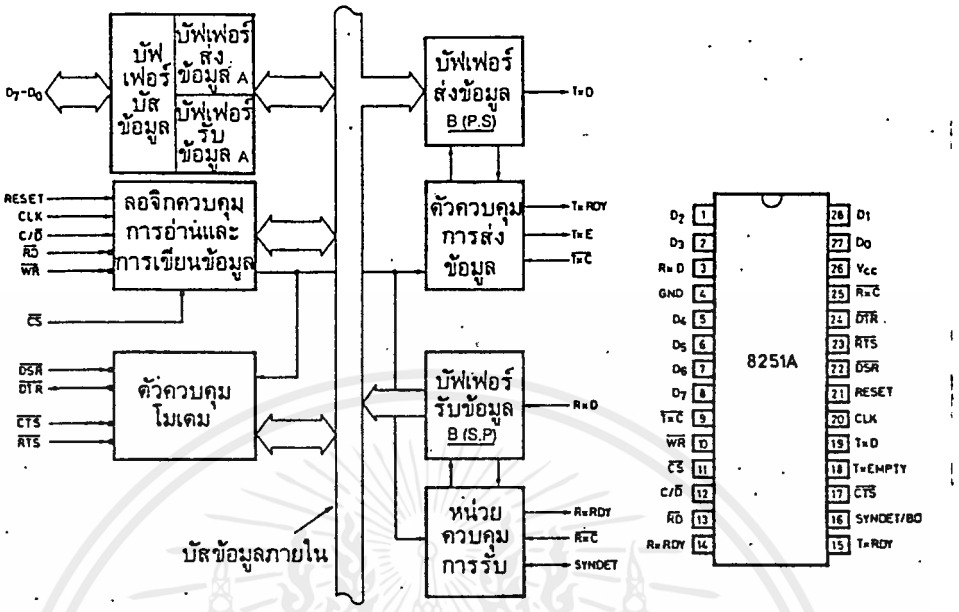
ไอซี 8251 สามารถที่จะส่งหรือรับข้อมูลด้วยอัตราการส่งระหว่าง dc-64k baud สำหรับข้อมูลแบบซิงโครนัส และ dc-19.2k baud สำหรับข้อมูลแบบอะซิงโครนัส ตัวรับและตัวส่งสามารถแยกที่จะรับหรือส่งข้อมูลที่ baud rate ที่แตกต่างกันได้ แต่ทั้งสองแบบนั้น (รับ/ส่ง) จำเป็นต้องปฏิบัติงานพร้อมกัน เพราะทั้งตัวรับและตัวส่งจะถูกโปรแกรมให้เหมือนกัน ไม่ว่าจะเป็นโหมดอะซิงโครนัส หรือ โหมดซิงโครนัสก็ตาม

ไอซี 8251 เป็นชิปขนาด 28 ขา มีการจัดขาและโครงสร้างภายใน ดังแสดงในรูปที่ 3.1

จากสัญญาณต่างๆ ในรูปที่ 3.1 สามารถสรุปหน้าที่ได้ดังตารางที่ 3.1

ชื่อสัญญาณ	หน้าที่โดยย่อ	ชนิดของสัญญาณ
D_0-D_7	บัสข้อมูล	สองทิศทาง
RESET	สัญญาณรีเซต	อินพุต
CLK	สัญญาณนาฬิกา	อินพุต
C/D	สัญญาณเลือก	อินพุต
RD	สัญญาณแสดงการอ่าน	อินพุต
WR	สัญญาณแสดงการเขียน	อินพุต
CS	สัญญาณเลือกชิป 8251	อินพุต
DSR	data set ready	อินพุต
DTR	data terminal ready	เอาต์พุต
CTS	clear to send	อินพุต
RTS	request to send	เอาต์พุต
TxD	ข้อมูลเอาต์พุตแบบอนุกรม	เอาต์พุต
TxRDY	พร้อมจะรับข้อมูลไปส่ง	เอาต์พุต
TxEMPTY	บัฟเฟอร์ข้อมูลว่าง	เอาต์พุต
TxC	สัญญาณนาฬิกากำหนดการส่ง	อินพุต
RxD	ข้อมูลอินพุตแบบอนุกรม	อินพุต
RxRDY	ข้อมูลพร้อมที่จะส่งไปยังบัสข้อมูล	เอาต์พุต
RxC	สัญญาณนาฬิกากำหนดการรับ	อินพุต
SYNDET/BD	synchronous/break detect	สองทิศทาง
V_{cc},GND	ไฟเลี้ยง	-

ตารางที่ 3.1สรุปสัญญาณต่างๆ ของ 8251



รูปที่ 3.1 โครงสร้างภายในและการจัดขาของไอซี 8251

8251 รับข้อมูลแบบขยสยจากไมโครโปรเซสเซอร์แล้วส่งออกไปยังอุปกรณ์ภายนอกแบบอนุกรม และยังสามารุรับข้อมูแบบอนุกรมจากอุปกรณ์ภายนอกมาเปลี่ยนเป็นแบบขนานแล้วส่งไปยังไมโครโปรเซสเซอร์ได้ด้วย 8251 จะส่งสัญญาณบอกไปโครโปรเซสเซอร์เมื่อพร้อมที่จะรับข้อมูลใหม่เพื่อจะส่งต่อไป หรือเมื่อมีข้อมูลซึ่งรับมาจากอุปกรณ์ภายนอกและพร้อมจะส่งให้ไมโครโปรเซสเซอร์ ไมโครโปคเซสเซอร์สามารถตรวจสอบสถานะของ 8251 ได้ตลอดเวลา โดยขบวนการอ่านข้อมูลภายในชิป 8251

รายละเอียดของสัญญาณต่างๆ ในรูปที่ 3.1 ที่เพิ่มเติมจากตารางที่ 3.1 มีดังนี้

1 บัฟเฟอร์บัสข้อมูล(data bus buffer) คือสัญญาณที่ใช้เชื่อมโยงกับบัสข้อมูลของไมโครโปรเซสเซอร์ ข้อมูลที่ส่งมาจากซีพียูจะถูกเก็บไว้ในบัฟเฟอร์ส่งข้อมูล A (transmit buffer A) ซึ่งจะถูกส่งต่อไปยังวงจรบัฟเฟอร์ B เพื่อจะเลื่อนออกไปแบบอนุกรมทาง TxD ในทางตรงกันข้ามข้อมูลที่ 8251 รับมาแบบอนุกรมจะถูกเปลี่ยนเป็นแบบขนานโดยบัฟเฟอร์รับข้อมูล B (receiver buffer B) ก่อน แล้วส่งต่อมาเก็บไว้ในบัฟเฟอร์รับข้อมูล A เพื่อส่งให้ซีพียูต่อไป นอกจากนี้ซีพียูยังสามารถบังคับให้ 8251 ทำหน้าที่ต่างๆ ได้ และข้อมูลหรือสถานะของ 8251 จะได้รับการอ่านจากซีพียูโดยผ่านเข้าทางบัสข้อมูลได้

2 ลอจิกควบคุมการอ่านและการเขียน (read and write control logic)มีหน้าที่ในการ

ควบคุมการอ่านหรือการเขียนข้อมูลมายัง 8251

3 ตัวควบคุมโมเดม (modem control) ทำหน้าที่ควบคุม 8251 เมื่อ 8251 ต้องติดต่อกับโมเดม

4 บัฟเฟอร์ส่งข้อมูล B (transmit buffer B) ทำหน้าที่รับข้อมูลจากบัฟเฟอร์ A แล้วเปลี่ยนจากบัฟเฟอร์ส่ง

5 ตัวควบคุมการส่งข้อมูล (transmit control) ทำหน้าที่ควบคุมการส่งข้อมูลแบบอนุกรมจากบัฟเฟอร์ส่ง

6 บัฟเฟอร์รับข้อมูล B (receive buffer B) เป็นที่เก็บข้อมูลที่ส่งมาจากอุปกรณ์ภายนอกและอนุกรมทาง RxD แล้วเปลี่ยนเป็นแบบขนานให้บัฟเฟอร์รับข้อมูล A

7 ตัวควบคุมการรับข้อมูล (receive control) ทำหน้าที่ควบคุมการรับข้อมูลแบบอนุกรม

8 บัสข้อมูลภายใน (internal data bus) เป็นเส้นทางสำหรับการติดต่อภายใน

3.1.2 สัญญาณติดต่อกับซีพียู

สัญญาณของไอซี 8251ที่ติดต่อกับซีพียูมีรายละเอียดดังนี้

D0 - D7 เป็นสัญญาณที่จะต่อกับบัสของไมโครโปรเซสเซอร์

RESET สัญญาณนี้จะแอคตีฟที่ลอจิก "1" และจะบังคับให้ 8251 อยู่ในสภาวะรีเซต หรือ วงจรภายในจะอยู่ในสภาพเหมือนขณะที่เริ่มต้น คำสั่งเก่าๆจากซีพียู

CLK เป็นสัญญาณนาฬิกาสำหรับวงจรต่างๆภายใน 8251 สัญญาณนี้จะต่อเข้ากับสัญญาณ clk ของ Z80 ได้

WR สัญญาณนี้แอคตีฟที่ลอจิก "0" และเป็นสัญญาณที่เขียนข้อมูลมายังพอร์ต

RD สัญญาณนี้แอคตีฟที่ลอจิก "0" เมื่อซีพียูต้องการอ่านข้อมูล

C/D ถ้าเป็น"1"คือสัญญาณควบคุมซึ่งจะใช้คู่กับสัญญาณ RD และ WR หมายถึงการเขียนและการอ่านรหัสควบคุม แต่ถ้าเป็น "0" จะใช้ร่วมกับสัญญาณ RD และ WR ซึ่งหมายถึงการเขียนและการอ่านข้อมูล

CS เป็นสัญญาณที่ใช้ในการเลือกชิป 8251 ซึ่งแอคตีฟที่ลอจิก "0"

ในการใช้งานจะใช้ร่วมกับระหว่างสัญญาณ C/D,RD,WR และ CS ดังตารางที่ 3.2

C/D	RD	WR	CS	ความหมาย
0	0	1	0	ข้อมูลจาก 8251 ไปสู่ บัสข้อมูล
0	1	0	0	บัสข้อมูลไปสู่ 8251
1	0	1	0	ข้อมูลบอกสถานะไปสู่บัสข้อมูล
1	1	0	0	บัสข้อมูลไปสู่รีจิสเตอร์
*	*	*	1	บัสข้อมูลลอยตัว

ตารางที่3.2 ความหมายในการทำงานร่วมกันของสัญญาณควบคุม 8251

3.1.3 สัญญาณติดต่อกับโมเดม

สัญญาณที่ 8251ใช้ติดต่อกับโมเดม ประกอบด้วย สัญญาณ DSR,DTR,RTS และ CTSซึ่งมีความหมายดังนี้

DSR(data det ready) เป็นสัญญาณที่อินพุตเข้า 8251 สถานะของสัญญาณ DSR (ลอจิก "0" หรือ "1") สามารถตรวจสอบได้โดยโปรแกรมของซีพียู จึงสามารถใช้สัญญาณ DSR เป็นสัญญาณพอร์ตอินพุตขนาด 1 บิตได้ แต่ในระบบโมเดม DSR จะเป็นสัญญาณที่ตอบสนองความพร้อมจากระบบโมเดม

DTR(data terminal ready) เป็นสัญญาณเอาต์พุตจาก 8251 สถานะของสัญญาณ DTR สามารถควบคุมโดยการโปรแกรมจากซีพียู จึงใช้เป็นพอร์ตเอาต์พุตขนาด 1 บิตได้ แต่ในระบบโมเดมจะใช้ DTR เป็นสัญญาณแสดงเพื่อบอกโมเดมถึงความพร้อมของ 8251

RTS(request to send) เป็นสัญญาณเอาต์พุตจาก 8251 สถานะของสัญญาณ RTS สามารถควบคุมได้โดยโปรแกรมจากซีพียู เราจึงสามารถใช้ RTS เป็นสัญญาณพอร์ตเอาต์พุตขนาด 1 บิตได้ แต่ถ้าใช้งานกับโมเดม RTS จะเป็นสัญญาณออกไปยังโมเดมเพื่อเตรียมส่งข้อมูล

CTS(clear to send) เป็นสัญญาณอินพุตจากภายนอก เมื่อสัญญาณ CTS เป็น "0" ตัว 8251 จึงจะส่งข้อมูลออกไปได้ (ถ้าสัญญาณ TxEN ในรหัสควบคุมเป็น "1" ด้วย)

3.1.4สัญญาณติดต่อกับภาครับและภาคส่ง

สัญญาณนี้จะทำหน้าที่ในการรับและส่งข้อมูล แสดงสถานะการทำงาน และควบคุมการทำงานของภาครับและส่ง สัญญาณต่างๆ มีดังนี้

TxRDY(transmitter ready)มีความสัมพันธ์กับบัฟเฟอร์ส่งข้อมูล A และบัฟเฟอร์ส่งข้อมูล B

คือ TxRDY จะเป็น "1" เมื่อข้อมูลในบัฟเฟอร์ส่งข้อมูล A ถูกส่งต่อไปยังบัฟเฟอร์ส่งข้อมูล B แล้ว นั่นคือ บัฟเฟอร์ส่งข้อมูล A ว่างและพร้อมที่จะรับข้อมูลใหม่จากซีพียูได้ สัญญาณ TxRDY จะเป็น "0" เมื่อบัฟเฟอร์ส่งข้อมูล A รับข้อมูลจากซีพียู แต่มีข้อควรระวังคือ บิตข้อมูลของ TxRDYในไบต์แสดงสถานะจะไปออกยังสัญญาณ TxRDY ขา 15 ก็ต่อเมื่อ 8251 ได้รับอรนาเบิลให้ส่งข้อมูลได้ หรือสัญญาณ TxEN ในไบต์แสดงสถานะเป็น "1" และสัญญาณ CTS เป็น"0"

TxEMPTY(transmitter empty) จะเป็น "1" เมื่อข้อมูลในบัฟเฟอร์ส่งข้อมูล B ถูกเลื่อนออกไปทางสัญญาณ TxD หมดแล้วซึ่งหมายความว่าบัฟเฟอร์ส่งข้อมูล A ว่างนั่นเอง และสัญญาณ **TxEMPTY** จะเป็น "0" เมื่อบัฟเฟอร์ส่งข้อมูล B รับข้อมูลมาจากบัฟเฟอร์ส่งข้อมูล A

TxC (transmitter clock)เป็นสัญญาณนาฬิกาฐานเวลาของข้อมูลที่ส่งออกไป ในระบบซิงโครนัส ความถี่ของสัญญาณ TxC ก็คือ อัตราโบด์ของการส่งเลย (1*) แต่ในระบบอะซิงโครนัส ความถี่ของสัญญาณ TxC จะเพิ่มเป็น1,16 หรือ 64เท่าของอัตราโบด์จริงก็ได้ ซึ่งเลือกได้โดยการโปรแกรมเช่น

ถ้าอัตราโบด์เป็น 100 โบด์จะได้

- ความถี่ของสัญญาณ TxC จะต้องเป็น100 Hz ถ้าเลือก(1*)
- ความถี่ของสัญญาณ TxC จะต้องเป็น1.6kHz ถ้าเลือก(16*)
- ความถี่ของสัญญาณ TxC จะต้องเป็น6.4kHz ถ้าเลือก(64*)

RxRDY (receiver ready)เป็นสัญญาณเอาต์พุตที่ส่งไปยังซีพียูเพื่อบอกว่ารับข้อมูลจากภายนอกเข้ามาอยู่ในบัฟเฟอร์รับข้อมูล A แล้ว และ พร้อมที่จะให้ซีพียูมาอ่านไปได้ สัญญาณ RxRDYนี้อาจจะใช้แทนสัญญาณอินเตอร์รัพต์ก็ได้ หรือในระบบการหยั่งเสียง(polling)ซีพียูอาจจะมาตรวจสอบสถานะของสัญญาณ RxRDY ได้โดยกระบวนการอ่าน และสัญญาณ RxRDY จะเป็น "0"ทันทีที่ซีพียูอ่านข้อมูลไปแล้ว

RxC(receiver clock)เป็นสัญญาณนาฬิกาที่กำหนดการรับข้อมูลจากภายนอก มีคุณลักษณะเช่นเดียวกับสัญญาณ ในการใช้งานของ สัญญาณนาฬิกาที่กำหนดการรับและส่งคือ สัญญาณ และ ซึ่งอาจจะเท่ากันหรือไม่ก็ได้

3.1.5การโปรแกรม 8251

การโปรแกรม 8251 คือ การสั่งให้ 8251ทำงานตามรูปแบบที่ระบบเราต้องการก่อนที่จะเริ่มการรับและการส่งทั้งหมด ซึ่งซีพียูต้องออกคำสั่งมายัง 8251 โดยมีขั้นตอนและประเภทของคำสั่ง คือ

1 สัญญาณ RESET เป็นการเริ่มต้นก่อนสัญญาณใดๆ

2 ทำคำสั่งเลือกกระบวนการทำงาน(mode select)

3.ทำคำสั่งควบคุม (command)

หลังจากใช้สัญญาณ RESET แล้ว ซีพียูต้องออกคำสั่งเลือกโหมด จบแล้วจะตามด้วยรหัสซิงก์หรือรหัสคำสั่งควบคุม แล้วแต่ว่าจะเลือกการทำงานในแบบซิงโครนัสหรืออะซิงโครนัสโดยซีพียูจะต้องส่งรหัสคำสั่งหรือข้อมูลเป็นลำดับขั้นตอน ดังตารางที่ 3.3

ขั้นที่	สัญญาณ	รหัสคำสั่งหรือข้อมูลที่ส่งมาให้ 8251
1	C/D = 1	Mode Instruction : เลือกโหมด
2	C/D = 1	Sync Character 1*
3	C/D = 1	Sync Character 2*
4	C/D = 1	Command Instruction :รหัสควบคุม
5	C/D = 0	Data
6	C/D = 1	Command Instruction : ทำเมื่อเปลี่ยนรหัส
7	C/D = 0	Data
8	C/D = 1	Command Instruction : ทำเมื่อเปลี่ยนรหัส

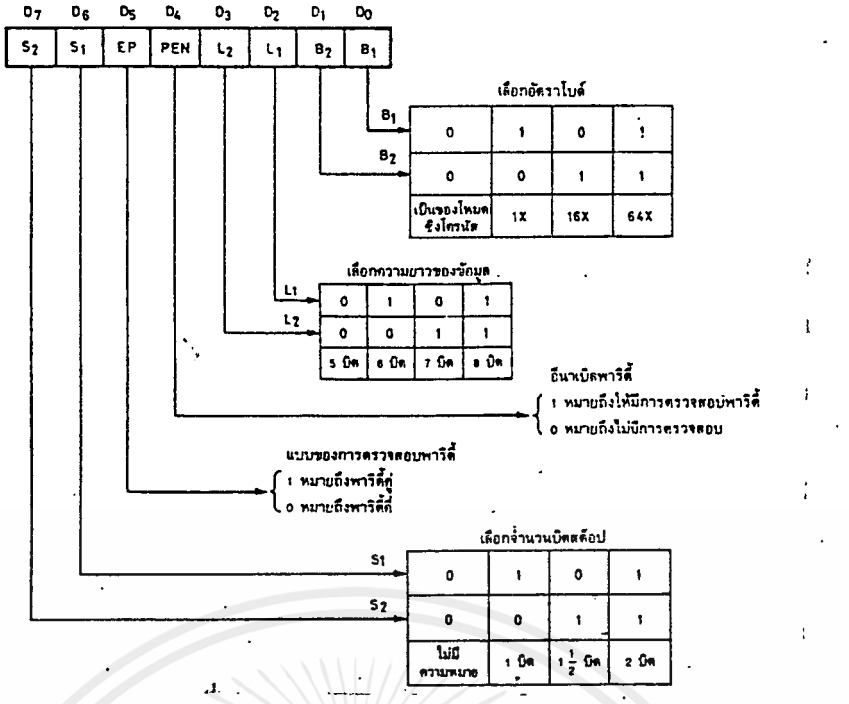
* ทำเมื่อเลือกโหมดซิงโครนัส

ตารางที่3.3 ลำดับที่ซีพียูติดต่อกับ 8251

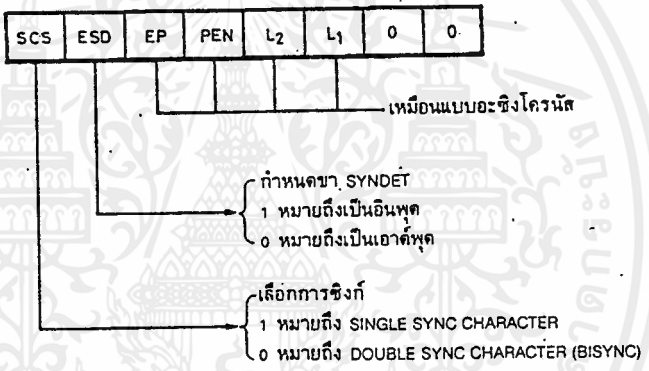
ขั้นตอนที่ 2 และ 3 ให้ข้ามไปได้เลยถ้าเลือกโหมดอะซิงโครนัส และขั้นตอนที่ 3 อาจไม่ต้องทำถ้าเลือกกระบวนการซิงโครนัสแบบใช้รหัสซิงก์ตัวเดียว สำหรับขั้นตอนที่ และ หมายถึง การรับหรือการส่งข้อมูลต่อเนื่องกันไป จนกว่าจะเปลี่ยนโหมดหรือรหัสควบคุม ส่วนขั้นตอนที่ 5 หรือ 7 จะทำถ้าต้องการเปลี่ยนรหัสควบคุมแต่ไม่เปลี่ยนแปลงโหมด

3.1.6คำสั่งเลือกโหมดและรหัสควบคุม

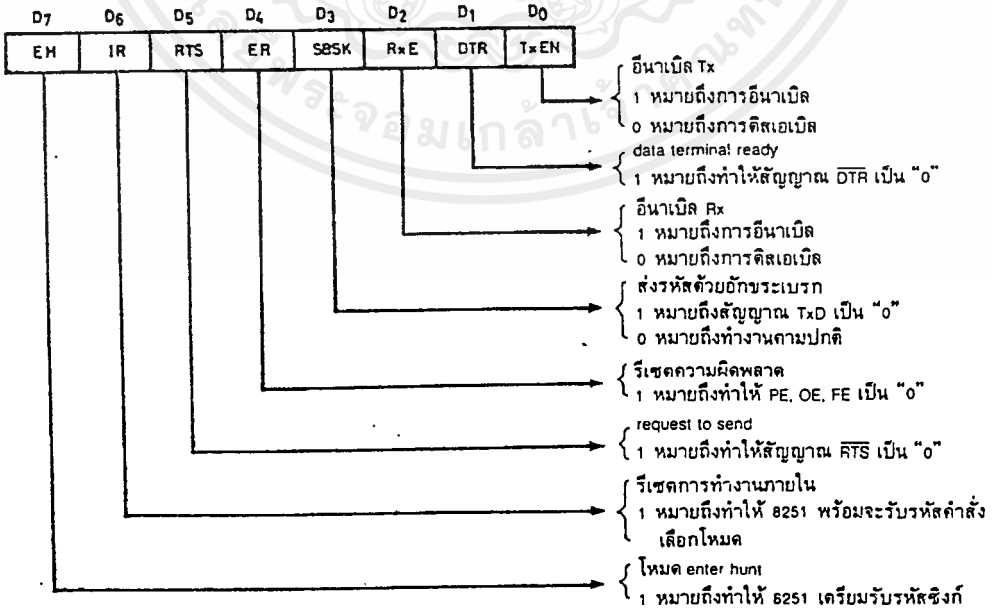
ลักษณะคำสั่งเลือกโหมดทั้ง 8 บิตของคำสั่งที่ซีพียูส่งมาให้ 8251 เพื่อเริ่มต้นการทำงานจะมีความหมายดังรูปที่ 3.2 สำหรับโหมดแบบอะซิงโครนัส ดังรูปที่ 3.3 สำหรังโหมดแบบซิงโครนัส ต่อจากนั้นจึงเป็นการส่งรหัสควบคุม ซึ่งมีความหมายดังรูปที่ 3.4



รูปที่ 3.2 รหัสเลือกโหมดแบบอะซิงโครนัส



รูปที่ 3.3 รหัสเลือกโหมดแบบซิงโครนัส



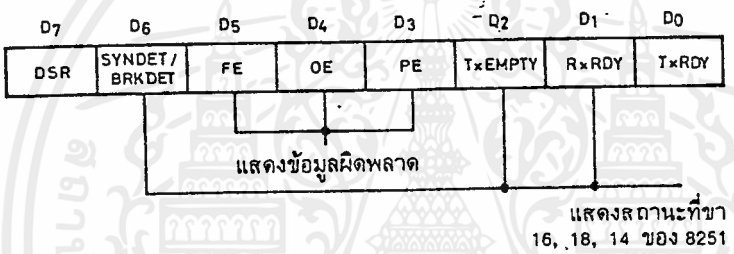
รูปที่ 3.4 รหัสคำสั่งของแต่ละบิตที่ควบคุมการทำงานภายในของ 8251

สังเกตที่การเลือกอัตราโบด์ของบิต D_0 และ D_1 ในรูปที่ 3.2 ความหมายของ 16* และ 64* คือ การนำความถี่ของสัญญาณจากเวลา TxC หรือ RxC มาหารด้วย 16* และ 64* ตามลำดับ แล้วนำไปกำหนดอัตราโบด์ ซึ่งมีประโยชน์ในกรณีที่ต้องใช้วงจรกำเนิดสัญญาณนาฬิกาความถี่สูงๆ แล้วต้องนำมาหารให้เหลือความถี่เท่ากับอัตราโบด์ การเลือก 16* และ 64* จะช่วยลดวงจรหารให้น้อยลงได้

เนื่องจากวงจรกำเนิดความถี่ดังกล่าว ต้องมีความเที่ยงตรงสูง จึงนิยมใช้คริสตอลเป็นตัวกำหนดความถี่ และสำหรับค่าความถี่ที่สามารถนำมากำหนดอัตราโบด์ได้ก็คือค่า 18.432 MHz

3.1.7 <u>โบด์แสดงสถานะ</u>

โบด์แสดงสถานะนี้มีขนาด 8 บิต แต่ละบิตจะแสดงสถานะของ 8251 โดยมีความหมายเป็น on หรือ offความหมายของแต่ละบิตเป็นดังรูปที่ 3.5



รูปที่ 3.5โบด์แสดงสถานะของ 8251

ความหมายของบิตในรูปที่3.5 มีดังนี้

TxRDY(D_0) จะเป็น "1"ทันทีที่บัฟเฟอร์ส่งข้อมูล A ว่าง ซึ่งหมายถึง พร้อมที่จะรับข้อมูลไปส่ง และยังแสดงถึงสัญญาณTxRDY ที่ขา 15 อีกด้วย โดยถ้าสัญญาณ CTS เป็น "0" สัญญาณTxRDY ที่ขา 15 จะมีลอจิกเหมือนขา 15 แต่ถ้าสัญญาณ CTS เป็น "1" สัญญาณ TxRDY ที่ขา15 จะเป็น"0" ทันที

RxRDY แสดงถึงได้รับข้อมูลเข้ามาครบโบด์แล้ว ให้ซีพียูมาอ่านไปได้ นอกจากนี้RxRDY ยังมีลอจิกเหมือนกับสัญญาณ RxRDYที่ขา 14 อีกด้วย

TxEMPTY แสดงลอจิกของสัญญาณTxEMPTY ที่ขา 13 โดยจะเป็น "1" เมื่อ 8251 ว่าง หรือไม่มีข้อมูลจะส่ง แต่จะกลับเป็น "0" เมื่อได้รับข้อมูลจากซีพียูในขณะที่ภาคส่งได้รับการอีนาเบิล

PE จะเป็น "1" เมื่อเกิดการผิดพลาดจากการตรวจสอบบิตพาริตี้ และจะถูกรีเซ็ตเป็น "0"โดย ER ในรหัสควบคุมจากซีพียู ซึ่งPE นี้จาก parity error

OE จะเป็น"1" ถ้าซีพียูไม่มาอ่านข้อมูลที่ 8251 ซึ่ง 8251 จะรับ OE เข้ามาไว้ในบัฟเฟอร์รับข้อมูล A แล้วมีข้อมูลใหม่เข้ามาอีก และ OE จะถูกรีเซ็ตเป็น "0" โดย ER ในรหัสควบคุม OE มา

จาก overrun error

FE ใช้ในกระบวนการอะซิงโครนัสเท่านั้น ถ้าข้อมูลที่รับมาขาดบิตสต็อปไป FE จะเป็น "1"และ FE จะถูกรีเซ็ตเป็น "0" โดย ER ในรหัสควบคุม ซึ่ง FE นี้มาจาก framing error

ทั้ง PE,OE และ FE จะไม่ขัดขวางการทำงานของ 8251แต่ประการใด เป็นเพียงการเตือนให้ทราบถึงการเกิดความผิดพลาดแล้วเท่านั้น

3.2 RS-232-C

RS-232-C เป็นอุปกรณ์เชื่อมต่อ DTE (Data Terminal Equipment) เข้ากับ DCE(Data Communication Equipment) ซึ่งเป็นมาตรฐานของ EIA (Electronic Industries Association) RS-232-C เป็นตัวเชื่อมต่อแบบ DB 25 pin สัญญาณต่างๆ ที่ผ่าน RS-232-C นี้จะเป็นไปตามมาตรฐาน EIA ซึ่งต่างไปจากสัญญาณ TTL ดังนี้

- สัญญาณจะมีสถานะ "1"ได้จะต้องมีแรงดัน ต่ำกว่า -3 Vdc แต่ไม่เกิน -15 Vdc ซึ่งเรียกว่า Marking
- สัญญาณจะมีสถานะ "0"ได้จะต้องมีแรงดัน ต่ำกว่า +3 Vdc แต่ไม่เกิน +15 Vdc ซึ่งเรียกว่า space

สัญญาณที่ใช้กับ RS-232-C

RS-232-C เป็นตัวเชื่อมต่อที่มี 25 ขา แต่สัญญาณที่ใช้นั้นมีเพียง 8 สัญญาณที่จำเป็น ดังนั้นขาที่เหลือจึงปล่อยว่างเอาไว้ สัญญาณต่างๆ มีดังนี้

ขา	สัญญาณ
1	NC
2	TRANSMITTED DATA
3	RECEIVED DATA
4	REQUEST TO SEND
5	CLEAR TO SEND
6	DATA SET READY
7	SIGNAL GROUND
8	RECEIVED LINE SIGNAL DETECTOR
9	+TRANSMIT CURRENT LOOP DATA
11	-TRANSMIT CURRENT LOOP DATA

18	+RECEIVE CURRENT LOOP DATA
20	DATA TERMINAL READY
22	RING INDICATOR
25	-RECEIVE CURRENT LOOP DATA

สำหรับขาที่ไม่กล่าวถึง เป็นขาที่ปล่อยลอยไว้

การส่งข้อมูลด้วย RS-232-C นั้น ส่งได้ 2 แบบคือ ส่งในรูปของแรงดันและในรูปของกระแส โดยปกติจะส่งในรูปของแรงดัน (ขา2,ขา3)แต่ในกรณีที่สายส่งมีความยาวมากๆ จะทำให้เกิดแรงดันตกคร่อมสายทำให้แรงดันลดลง บางครั้งอาจลดลงเป็นศูนย์ได้ ถ้าระยะทางไกลมากๆ ดังนั้นจึงมีการคิดแก้ไขโดยส่งในรูปของกระแส เพราะกระแสจะมีค่าคงที่ตลอดความยาวของสายส่ง(ขา9,ขา11,ขา 18 และขา 25)

ขาสัญญาณต่างๆ มีหน้าที่ดังนี้

ขา 2 Transmitted Data

: เป็นขาส่งข้อมูลแบบอนุกรมในรูปของแรงดัน

ขา 3 Received Data

: เป็นขารับข้อมูลแบบอนุกรมในรูปของแรงดัน

ขา 4 Request to Send

: เป็นขาส่งสัญญาณที่บ่งว่า DTE ต้องการส่งข้อมูล

ขา 5 Clear to Send

: เป็นขารับสัญญาณที่บ่งว่าอุปกรณ์ภายนอกพร้อมที่จะรับข้อมูล

ขา 6 Data Set Ready

: เป็นขารับสัญญาณที่บ่งว่าอุปกรณ์ภายนอกพร้อมที่จะติดต่อด้วย

ขา 7 Signal Ground

: ขาสัญญาณอ้างอิง(0 V)

ขา 8 Received Line Signal Detector

: ขารับสัญญาณที่บ่งว่า DCE จับสัญญาณที่จะ demodulate ได้

ขา 20 Data Terminal Ready

: ขาส่งสัญญาณขอติดต่อสื่อสารกับอุปกรณ์ภายนอก

ขา 22 Ring Indicator

: ขารับสัญญาณเรียกทางโทรศัพท์

สำหรับขา 9, 11 Transmit Current Loop Data เป็นขาส่งข้อมูลในรูปของกระแส และขา 18 +Receive Current Loop Data กับ ขา 25

- Receive Current Loop Return เป็นขารับข้อมูลในรูปของกระแส

3.3 การใช้งานพอร์ตอนุกรมผ่านBIOS

พอร์ตอนุกรม เป็นอุปกรณ์ที่ใช้รับส่งข้อมูลที่สร้างปัญหายุ่งยากให้แก่ผู้เขียนโปรแกรมมากที่สุดอุปกรณ์หนึ่ง ซึ่งต่างจากพอร์ตแบบขนาน ที่มีการทำงานและใช้งานที่ง่ายกว่ามาก แต่ถึงกระนั้นพอร์ตแบบอนุกรมก็ยังถูกนำมาใช้งานอย่างมาก เพราะค่าใช้จ่ายที่ถูกกว่า เมื่อเทียบกับพอร์ตแบบขนาน

ในบทนี้จะได้อธิบายถึงพื้นฐานของการทำงานของพอร์ตแบบอนุกรมรวมทั้งการเตรียมสถานะริ่มต้นของอุปกรณ์(initialization) การส่งข้อมูล(transmittion) การรับข้อมูล (reception) และความผิดพลาดทั่วๆ ไปของการรับส่งข้อมูล หลังจากนั้นจะพัฒนาโปรแกรมแอพพลิเคชันที่ใช้งานพอร์ตแบบอนุกรม ๒ โปรแกรม ซึ่งได้แก่ โปรแกรมโอนย้ายไฟล์ (file transfer) ที่สามารถโอนย้ายไฟล์ระหว่างคอมพิวเตอร์ ๒ เครื่องได้ทุกชนิด และอีกโปรแกรมหนึ่งคือ ระบบลน(Local Area Network:LAN) ซึ่งประกอบด้วย โปรแกรมที่ใช้ในศูนย์กลางบริการการใช้ฟล์ หรือไฟล์เซิร์ฟเวอร์(file server) และคำสั่งที่ใช้สำหรับเรียกใช้ไฟล์และส่งไฟล์ไปเก็บที่ฟล์เซิร์ฟเวอร์

การใช้งานพอร์ตอนุกรมสามารถทำได้ ๓ วิธีคือ ผ่านทางดอส ผ่านทาง BIOS และสุดท้ายคือ การเขียนโปรแกรมควบคุมพอร์ตอนุกรมโดยตรง

การเรียกใช้พอร์ตอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนัก เพราะว่าดอสไม่มีวิธีที่จะดูสถานะของการรับส่งและตัวพอร์ตอนุกรมว่าการรับส่งข้อมูลถูกต้องหรือไม่เพียงใด

การเขียนโปรแกรมควบคุมการทำงานของพอร์ตอนุกรมโดยตรงนั้นคงไม่จำเป็น เพราะมีวิธีดีกว่าและง่ายกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพหมายเลข ๑๔

3.3.1การเตรียมสถานะเริ่มต้นของพอร์ต

ในการเตรียมสถานะของพอร์ตอนุกรมเราสามารถทำได้ โดยผ่านอินเตอร์รัพต์หมายเลข ๑๔ ฟังก์ชันหมายเลข ๐ โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังก์ชัน(ในที่นี้คือ ๐) รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ตอนุกรม โดยมีความหมายของแต่ละบิตดังในตารางที่3.4

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

ตารางที่ 3.5

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ตอนุกรมได้มากถึง 7 พอร์ต โดยหมายเลขพอร์ตที่จะใช้สามารถกำหนดผ่าน รีจิสเตอร์ DX พอร์ตแรกมีหมายเลข 0 พอร์ตต่อไปหมายเลข 1 เช่นนี้ต่อไปเรื่อยๆ ฟังก์ชันที่แสดงต่อไปนี้มีชื่อว่า init_port() มีหน้าที่เตรียมสถานะของพอร์ตในระบบ

```
/* ฟังก์ชันนี้ทำหน้าที่เตรียมสถานะเริ่มต้นของพอร์ต */
viod port_init(port,code)
int port;
unsigned char code;
{
   union  REGS r;
   r.x.dx = port; /* หมายเลขพอร์ตอนุกรม */
   r.H.ah = 0;    /* เรียกฟังก์ชันหมายเลข 0 ทำหน้าที่เตรียมสถานะ */
   r.h.al = code; /* รหัสตั้งสถานะเริ่มต้น */
   int86(0x14,&r,&r);
}
```

3.3.2 การส่งข้อมูลผ่านพอร์ตอนุกรม 1ไบต์

อินเตอร์รัพต์หมายเลข 14 ฟังก์ชันหมายเลข 1 ของ BIOS ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ตอนุกรม หมายเลขของพอร์ตอนุกรมที่จะทำการส่งข้อมูล สามารถกำหนดได้ผ่านรีจิสเตอร์ DX และข้อมูลที่ต้องการส่งจะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเสร็จเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่าการส่งข้อมูลถูกต้องหรือไม่

รายละเอียดของฟังก์ชัน sport() ดังนี้

```
/* ฟังก์ชัน sport() ทำหน้าที่ส่งตัวอักษรออกทางพอร์ตอนุกรม */
```

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตรารับส่งข้อมูล 000 = 110 baud 001 = 150 baud 010 = 300 baud 011 = 600 baud 100 = 1200 baud 101 = 2400 baud 110 = 4800 baud 111 = 9600 baud
4,3	พาริตี้ 00 หรือ 10 = ไม่มีพาราตี้ 01 = พาริตี้คี่ 11 = พาริตี้คู่
2	จำนวนของบิตสิ้นสุด 0 = 1 บิตสิ้นสุด 1 = 2 บิตสิ้นสุด
1,0	จำนวนบิตในข้อมูล 1 ไบต์ 10 = 7 บิต 11 = 8 บิต

ตารางที่ 3.4

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะของพอร์ตอนุกรม เช่น ถ้าต้องการตั้งให้พอร์ตมีอัตรา 9600 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุด และใช้ 8 บิตต่อ 1 ไบต์ข้อมูล จะได้รูปแบบของรหัสดังตารางที่ 3.5

```
void sport(port,c)
int port;  /* หมายเลขของพอร์ต */
char c;    /* ตัวอักษรที่ต้องการส่ง*/
{
   union  REGS  r;
   r.x.dx = port;  /* กำหนดหมายเลขพอร์ต */
   r.h.al = c;     /* ตัวอักษรที่ต้องการส่ง */
   r.h.ah = 1;     /* ฟังก์ชันหมายเลข 1 ทำหน้าที่ส่งข้อมูล 1 ไบต์ */
   int86(0x14,&r,&r);
   if(r.h.ah & 128) {   /*  ตรวจดูบิตที่ 7 */
   printf("send error detected in serial port");
   exit(1);
   }
 }
```

ถ้าบิต 7 ของรีจิสเตอร์ AH มีค่า 1 เมื่อส่งข้อมูลเสร็จสิ้น แสดงว่าเกิดความผิดพลาดอะไร ดังในหัวข้อต่อไป

3.3.3 การตรวจสอบสถานะของพอร์ตอนุกรม

ฟังก์ชันหมายเลข 3 ของอินเตอร์รัพต์หมายเลข 14 ของ BIOS ใช้ตรวจดูสถานะของพอร์ตอนุกรม รีจิสเตอร์ DX ใช้สำหรับหมายเลขพอร์ตที่จะตรวจสอบ สถานะของพอร์ตสามารถแปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL ดังในตารางที่ 3.3

สถานะของสายส่ง (AH)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data Ready	0
Overrun Error	1
Parity Error	2
Framming Error	3
Break-Detect Error	4
Transfer hold-registor empty	5
Transfer shift-registor empty	6
Time-out Error	7
สถานะของโมเด็ม (AL)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring detector	2
Change in line signal	3
Clear-to-send	4
Data-set-ready	5
Ring indicator	6
Line signal detector	7

ตารางที่3.6

จะเห็นว่าสัญญาณสถานะต่างๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดังนั้นเมื่อนำมาใช้กับอุปกรณ์อื่นจึงลดความสำคัญลง แต่ก็ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญมากคือ Data Readyซึ่งจะเป็นตัวบอกว่า เมื่อไหร่ที่ข้อมูลว่า rport() มีหน้าที่อ่านข้อมูลจากพอร์ต โดยจะใช้สถานะData Ready นี้เป็นตัวบอกว่า ข้อมูลพร้อมที่จะถูกอ่านหรือยัง

3.3.4 การรับข้อมูลผ่านพอร์ตอนุกรม 1 ไบต์

การรับข้อมูลจากพอร์ตอนุกรม สามารถทำได้โดยเรียกผ่าน BIOS อินเตอร์รัพต์หมายเลข 14 ฟังก์ชันหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ต ข้อมูลที่อ่านได้จะเก็บในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ตจะสามารถตรวจดูได้ที่บิต 7ของรีจิสเตอร์ AH

ต่อไปนี้คือฟังก์ชัน rport() มีหน้าที่อ่านข้อมูลเข้า 1 ไบต์

```
/* ฟังก์ชันนี้ทำหน้าที่อ่านตัวอักษรจากพอร์ตอนุกรม */
rport(port)
int port; /* หมายเลขพอร์ต */
{
   union REGS r;
   /*  */
   while(!(check_stat(port)&256))
   if(kbit()) {    /* ยกเลิกเมื่อมีการกดคีย์  */
   getch();
   exit();
   }
   r.x.dx = port; /* กำหนดหมายเลขพอร์ต */
   r.h.ah = 2;     /* ฟังก์ชันหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
   int86(0x14,&r,&r);
   if(r.h.ah & 128)
   printf("read error detected in serial port");
   return r.h.al;
}
```

การทำงานของฟังก์ชันนี้คือจะรอจนกระทั่งข้อมูลถูกรับมาผ่านพอร์ตอนุกรมแล้วส่งค่าอักษรกลับมา แต่ว่าการทำงานเช่นนี้อาจทำให้โปรแกรมไม่หลุดจากลูปการทำงาน เช่น เมื่อสายส่งของพอร์ตเกิดเสีย ดังนั้นจึงต้องตรวจสถานะของพอร์ตอนุกรมเสียก่อน ดังได้กล่าวไว้ในหัวข้อที่ผ่านมา ฟังก์ชัน kbhit() ใช้ตรวจสอบการกดคีย์ ถ้าหากไม่มีข้อมูลผู้ใช้ก็สามารถกดคีย์ใดคีย์หนึ่งเพื่อออกจากลูปได้ แต่ถ้ามีข้อมูลรับเข้ามา ฟังก์ชันก็จะผ่านไปเรียกอินเตอร์รัพท์เพื่ออ่านข้อมูลเข้ามา และเช่นเดียวกับการส่งข้อมูลบิต 7 ของรีจิสเตอร์ AH ใช้บอกว่า การอ่านข้อมูลมีข้อมูลมีข้อผิดพลาดหรือไม่

3.4 การโอนย้ายไฟล์ระหว่างคอมพิวเตอร์

3.4.1 การส่งไฟล์ข้อมูล

ฟังก์ชันแรกซึ่งจำเป็นสำหรับการโอนย้ายไฟล์ผ่านพอร์ตอนุกรมก็คือฟังก์ชัน send_file() ที่ทำหน้าที่เปิดไฟล์ที่จะทำการส่ง นับจำนวนไบต์ในไฟล์นั้น ส่งขนาดของไฟล์นั้นให้แก่คอมพิวเตอร์ด้านรับ สุดท้ายก็ทำการโอนย้ายไฟล์ตามขนาดที่ส่งไป

```
/* ฟังก์ชันนี้ทำหน้าที่ส่งไฟล์ที่กำหนด */
void send_file(fname)
char *fname
{
   FILE *fp;
   char  ch;
   union {
   char c[2];
   unsigned int count;
   } cnt;
   if(!(fp=fopen(fname,"rb"))){
   printf("cannot open input file\n");
   exit(1);
   }
   send_file_name(fname); /* ส่งชื่อไฟล์ */
```

```
wait(port); /* รอจนกว่าจะมีการตอบรับ */
/* หาขนาดของไฟล์ */
cnt.count = filesize(fp);
/* ส่งขนาดของไฟล์ */
sport(port,cnt.c[0]);
wait(port);
sport(port,cnt.c[1]);
 do{
  ch=getc(fp);
  if(ferror(fp)){
  printf("error reading input file");
  break;
  }
/* รอจนกว่าคอมพิวเตอร์ด้านรับจะพร้อม */
if(!feof(fp)){
wait(port);
sport(port,ch);
}
} while(!feof(fp));
wait(port); /* อ่านเครื่องหมายจุด (.) ตัวสุดท้ายจากพอร์ต */
fclose(fp);
}
```

ฟังก์ชัน send_file_name() ทำหน้าที่ 2 อย่างคือ ทำหน้าที่ติดต่อกับคอมพิวเตอร์ด้านรับ โดยการส่งเครื่องหมาย (?) จนกว่าคอมพิวเตอร์ด้านรับจะตอบรับกลับมาด้วยเครื่องหมาย(.) (เครื่องหมายจุดเป็นเพียงสัญลักษณ์การตอบรับในโปรแกรมนี้เท่านั้น เราใช้สัญลักษณ์เป็นตัวอักษรตัวอื่นๆได้) และหน้าที่อีกอย่างคือ เมื่อทำการติดต่อกับคอมพิวเตอร์ด้านรับ

สำหรับฟังก์ชัน check_stat() ใช้สำหรับตรวจสอบสถานะของพอร์ตอนุกรมที่เราทำ

หนด ส่วนฟังก์ชัน wait() ใช้สำหรับรอการตอบรับจากคอมพิวเตอร์ด้านรับซึ่งเป็นการทำงานของซอฟแวร์แฮนด์เชคกิ้งนั่นเอง

```
/* ฟังก์ชันนี้ทำหน้าที่รอจนกว่าจะมีการตอบรับ */
void wait(port)
int port;
{
  if(rport(port) !='.') {
  printf("communication error\n");
  exit(1);
  }
 }
```

ถ้าเกิดข้อมูลผิดพลาดในการส่งข้อมูลนั้น ฟังก์ชันนี้จะหลุดจากการทำงาน แล้วกลับไปสู่โปรแกรมหลัก แต่ถ้าหากว่าต้องการจัดการกับข้อผิดพลาดเอง ก็สามารถทำได้เช่นกันจากรหัส exit(1)

ฟังก์ชัน file_size() ทำหน้าที่หาขนาดของไฟล์ในหน่วยไบต์ ถ้าหากว่าคอมไพเลอร์ที่ใช้อยู่มีฟังก์ชันที่ใช้หาขนาดของไฟล์อยู่แล้ว ก็สามารถใช้แทนฟังก์ชันนี้ได้เลย มีข้อสังเกตว่าตัวแปร cnt ซึ่งเป็นตัวแปรแบบยูเนียน(union) นั้นจำเป็นต้องใช้ เพราะว่าขนาดของไฟล์นั้นเป็นตัวเลขขนาด 2 ไบต์ แต่โปรแกรมสามารถส่งข้อมูลทีละไบต์เท่านั้น

3.5.2 การรับไฟล์ข้อมูล

การรับไฟล์ข้อมูลมีกรรมวิธีที่ย้อนกลับกับการส่งไฟล์ข้อมูลคือ เริ่มด้วยการรอจนกว่าจะได้รับ เครื่องหมายคำถามฟังก์ชันนี้ก็จะตอบรับด้วยเครื่องหมายจุด ซึ่งเป็นเครื่องหมายตอบรับการส่งข้อมูล จากนั้นจะอ่านไฟล์เข้ามา ตามด้วยขนาดของไฟล์ สุดท้ายจะเป็นข้อมูลแต่ละไบต์ของไฟล์เมื่อมีการอ่านเข้ามาแต่ละไบต์ จะมีการตอบรับกลับไปสู่คอมพิวเตอร์ด้านส่งเสมอ เพื่อเป็นการบอกว่าได้รับข้อมูลเรียบร้อยแล้ว

ฟังก์ชัน rec_file() มีรายละเอียดดังนี้

```
/* ฟังก์ชันนี้ทำหน้าที่รับไฟล์ */
void rec_file()
```

```
{
  FILE *fp;
  char  ch;
  char  fname[14];
  union  {
  char  c[2];
  unsigned int count;
   }cnt;
  get_file_name(fname); /* รับชื่อของไฟล์ */
  printf("receiving file %s\n",fname);
  remove(fname);
  if(!(fp=fopen(fname,"wb"))){
  printf("cannot open output file\n");
  exit(1)
  }
  /* หาขนาดของไฟล์ */
  sport(port,'.');/* ตอบรับ */
  cnt.c[0] = rport(port);
  sport(port,'.');/* ตอบรับ */
  cnt.c[1] = rport(port);
  sport(port,'.'); /* ตอบรับ */
  for (; cnt.count;cnt.count--){
  ch = rport(port);
  putc(ch,fp);
  if(ferror(fp)){
  printf("error writing file");
  exit(1);
  }
```

```
    sport(port,'.');  /* ตอบรับ */
   }
   fclose(fp);
  }
```

ฟังก์ชันที่ทำหน้าที่รับชื่อไฟล์เข้ามามีรายละเอียดดังนี้

```
/* ฟังก์ชันนี้ทำหน้าที่รับชื่อไฟล์ */
void get_file_name(f)
char  *f;
{
  printf("receiver waiting... \n");
  while(rport(port) != '?');]
  sport(port,'.'); /* ตอบรับ */
  while((*f = rport(port))) {
  if(*f != '?') {
     f++;
    sport(port,'.');   /* ตอบรับ */
   }
  }
 }
```

3.5 พิธีการและรูปแบบในการติดต่อรับส่งข้อมูล

ในการส่งข้อมูลระหว่าง PERSONAL COMPUTER สิ่งที่หลีกเลี่ยงไม่ได้คือความผิดพลาดต่างๆ ที่อาจจะเกิดขึ้นได้ในระหว่างการติดต่อ ดังนั้นจึงได้มีการจัดรูปแบบการรับส่งข้อมูลเพื่อให้ได้ข้อมูลที่ถูกต้องและใช้เวลารวดเร็วในการรับส่ง ซึ่งจะมีการตรวจสอบความถูกต้องของข้อมูลโดยมีการส่งข้อมูลเป็นบล็อคๆให้มีการเช็คผลบวกของข้อมูลในแต่ละบล็อค รวมทั้งมีการใช้โปรโตคอลแบบ XMODEM ในการติดต่อดังจะมีรายละเอียดดังนี้

3.5.1 การส่งผ่านข้อมูลทีละบล็อค

ในขั้นตอนการทำงานของการรับส่งข้อมูลเมื่อมีการโหลดไฟล์ที่ต้องการส่งออกไปเราจะมีการจัดการเกี่ยวกับข้อมูลในไฟล์โดยจัดแบ่งเป็นบล็อคขนาด 128 ไบท์ ทั้งนี้เพื่อให้ง่ายต่อการส่ง เนื่องจากในแต่ละบล็อคจะมีข้อมูลขนาดไม่มากนักซึ่งเมื่อมีข้อมูลที่ผิดพลาดในการส่งของบล็อคที่ผ่านมาก็จะเสียเวลามากในการส่งข้อมูลบล็อคเดิมลงไป และข้อผิดพลาดที่อาจจะเกิดขึ้นสามารถเช็คได้จากผลบวกของข้อมูลในบล็อคนั้นๆ และการควบคุมการส่งข้อมูลในแต่ละบล็อคจะมีรูปแบบที่เรียกว่าโปรโตคอล ดังจะได้กล่าวต่อไป

3.5.2 การเช็คผลบวกของข้อมูล

ในแต่ละบล็อคของข้อมูลเราจะทำการบวกข้อมูลในแต่ละไบท์เข้าด้วยกันทั้งนี้จนครบ 128 ไบท์ และจะเอาผลรวมที่ได้เฉพาะไบท์ต่ำไบท์เดียวเท่านั้น แล้วนำมาลบด้วย 100h นำค่าที่ได้ให้เป็นผลของการเช็คผลบวก เพื่อส่งออกไปให้ตัวรับ และในทำนองเดียวกันเมื่อตัวรับได้รับข้อมูลจนครบ 128 ไบท์ พร้อมกับค่าเช็คผลบวกนั้น แล้วก็จะทำการบวกข้อมูลทั้งหมดนั้นเข้าด้วยกันและจะเอาผลรวมที่เป็นเฉพาะไบท์ต่ำนั้น มาทำการเปรียบเทียบกับค่า 00h ว่าเท่ากันหรือไม่ ถ้าเท่ากันก็จะรับข้อมูลบล็อคต่อไปแต่ถ้าไม่เท่ากันก็จะให้ตัวส่งส่งข้อมูลบล็อคเดิมกลับมาใหม่ โดยวิธีการนี้จะทำให้ข้อมูลมีโอกาสผิดพลาดน้อยมาก และเป็นวิธีการที่นิยมใช้กันมาก โดยจะใช้โปรโตคอลแบบ XMODEM ควบคุมการติดต่อ

3.5.3 โปรโตคอลแบบ XMODEM

เป็นเทคนิคในการควบคุมการรับส่งข้อมูล และจะต้องมีการกำหนดรูปแบบให้เหมือนกันทั้งฝ่ายรับและฝ่ายส่ง โดยการใช้อักขระควบคุมในตารางของแอสกี้ เพื่อควบคุมและการกำหนดการรับส่งบล็อคของข้อมูล อักขระที่ใช้ในการควบคุมนี้ได้แก่

ACK (ACKNOWLEDGE) เป็นรหัสที่ใช้บอกตัวส่งว่าพร้อมแล้วต้องการจะติดต่อด้วย และเป็นสัญญาณที่ใช้บอกตัวส่งว่าข้อมูลถูกต้องให้ส่งข้อมูลบล็อคต่อมาได้ ซึ่งมีค่าเท่ากับ 06 ในรหัสแอสกี้

NAK (NEGETIVE ACKNOWLEDGE) เป็นรหัสที่ใช้บอกฝ่ายส่งว่าข้อมูลในบล็อคที่ผ่านมา ผิดพลาดให้ส่งข้อมูลบล็อคเดิมมาอีกครั้ง ซึ่งมีค่าเท่ากับ 21 ในรหัสแอสกี้

ETB (END OF TRANSMISSION BLOCK) ใช้ระบุจุดจบของข้อมูลในบล็อคนั้น ซึ่งมีค่าเท่ากับ 23 ในรหัสแอสกี้

ETX (END OF TEXT) เป็นรหัสที่ใช้บอกฝ่ายส่งว่าข้อมูลที่ส่งให้บล็อคนั้นเป็นบล็อค

สุดท้ายของการส่งข้อมูล ซึ่งมีค่าเท่ากับ 03 ในรหัสแอสกี้

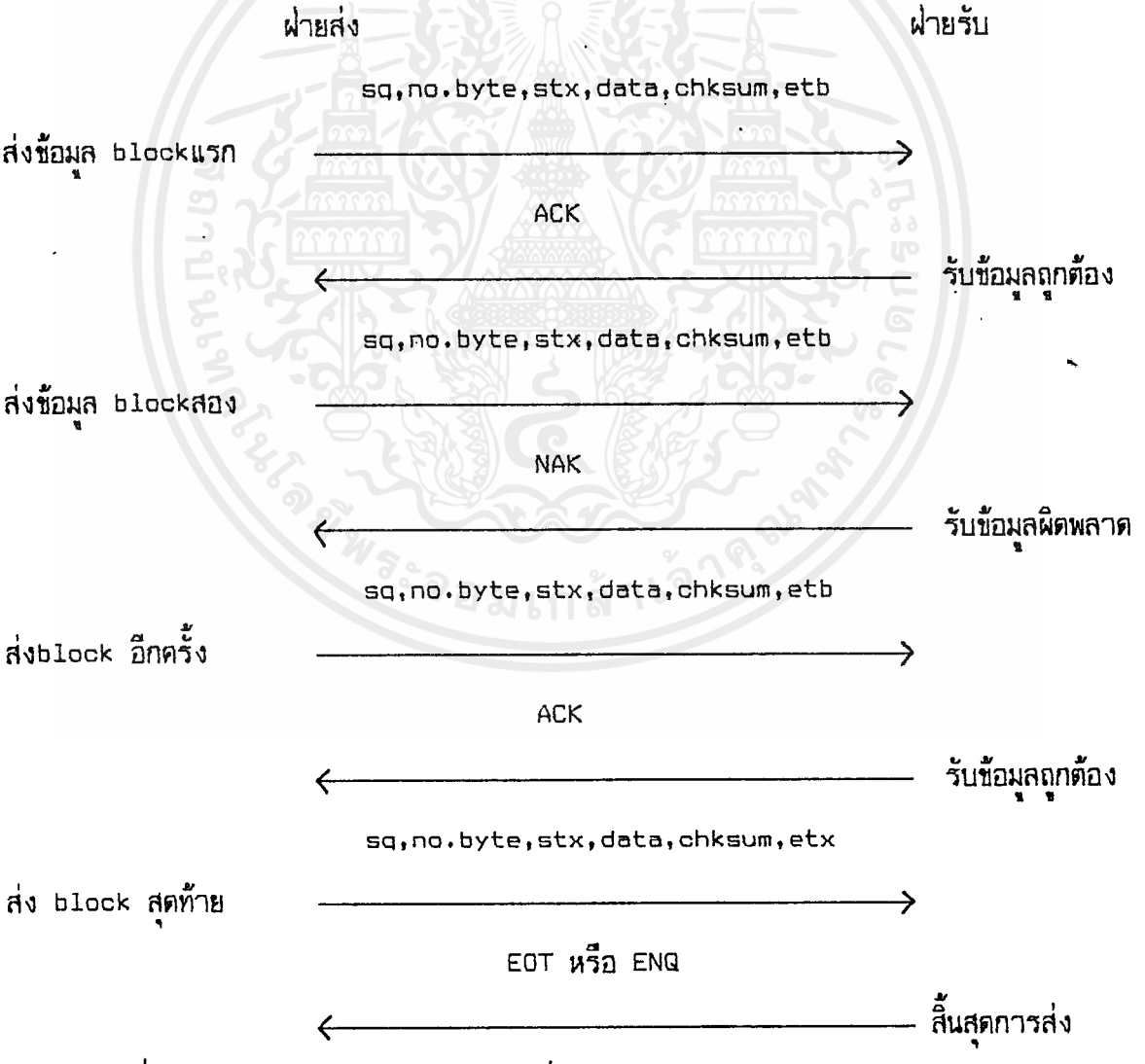
ENQ (ENQUIRY) เป็นรหัสที่ส่งมาจากฝ่ายรับ เพื่อมาบอกฝ่ายส่งว่าต้องการส่งข้อมูลให้ฝ่ายส่ง ซึ่งมีค่าเท่ากับ 05 ในรหัสแอสกี้

STX (START OF TEXT) เป็นรหัสที่ส่งไปพร้อมกับบล็อคข้อมูล เพื่อบอกว่าเป็น bit เริ่มต้นของข้อมูลในบล็อค ซึ่งมีค่าเท่ากับ 01 ในรหัสแอสกี้

SOH (STRAT OF HEAD) เป็นรหัสที่ส่งไปพร้อมกับบล็อคคำสั่ง เพื่อบอกว่าเป็นการเริ่มต้นของบล็อคส่วนหัว ซึ่งเป็นคำสั่ง ซึ่งมีค่าเท่ากับ 02 ในรหัสแอสกี้

EOT (END OF TRANSMISSION) เป็นรหัสที่ใช้ระบุว่าสิ้นสุดการส่งผ่านข้อมูลหรือข้อมูลได้รับเรียบร้อยแล้วให้ยกเลิกการติดต่อ ซึ่งมีค่าเท่ากับ 04 ในรหัสแอสกี้

ลักษณะการติดต่อแบบใช้โปรโตคอล แสดงดังในรูปที่ 3.6



รูปที่ 3.6 แสดงการส่งผ่านข้อมูลทีละบล็อคของการส่งสัญญาณ half-duplex

โดย sq (sequene number) คือ หมายเลขของบล็อคข้อมูล

no.byte (number of byte) คือ จำนวนของข้อมูลที่ส่งในบล็อคนั้น

data คือ ข้อมูล

chksum (checksum) คือ ค่าของการตรวจสอบการผิดพลาด

โดยที่ทางตัวส่งจะไม่ส่งข้อมูลไปให้จนกว่าฝ่ายรับจะทำการส่งสัญญาณ ACK มาให้ฝ่ายส่ง เมื่อฝ่ายส่งได้รับสัญญาณนี้แล้วจะทำการส่งข้อมูลบล็อคแรกไปให้ ซึ่งจะเป็นข้อมูล 128 ไบท์ และจะส่งสัญญาณ ETB ปิดท้ายไปเพื่อทำการบอกฝ่ายรับว่าสิ้นสุดข้อมูล บล็อคที่หนึ่งแล้ว จากนั้นฝ่ายส่งจะทำการส่งผลบวกของข้อมูลทั้ง 128 นั้น ไปให้ฝ่ายรับ เมื่อได้รับผลรวมของข้อมูลแล้วก็จะทำการเช็คกับผลรวมของข้อมูลที่ฝ่ายรับได้รวมเอาไว้แล้วว่าตรงกันหรือไม่ ถ้าตรงกันแสดงว่าข้อมูลที่ได้รับมานั้นถูกต้อง ทางฝ่ายรับก็จะส่งสัญญาณ ACK ตอบกลับไปให้ฝ่ายรับเพื่อให้ส่งข้อมูลบล็อคที่ 2 มายังฝ่ายรับต่อไปและจะทำอย่างนี้ต่อไปเรื่อยๆ จนกว่าทางฝ่ายรับจะเจอ ETX (END OF TEXT) ซึ่งเป็นการบอกให้ฝ่ายรับรู้ว่ามันเป็นไบท์สุดท้ายของข้อมูล เมื่อฝ่ายรับทำการเช็คผลบวกแล้วว่าถูกต้อง ก็ทำการส่งสัญญาณ EOT ตอบกลับไปเพื่อยกเลิกการติดต่อหรือส่งสัญญาณ ENQ กลับไปเมื่อฝ่ายรับต้องการส่งข้อมูลให้ฝ่ายส่ง

สำหรับในกรณีที่ข้อมูลที่ส่งมาเมื่อผ่านการเช็คผลรวมของข้อมูลแล้วไม่ตรงกัน ทางฝ่ายรับก็จะส่งสัญญาณ NAK ตอบกลับไป

XMODEM เหมาะสำหรับ IBM PC เหนือโปรโตคอลชนิดอื่น 3 ข้อ คือ

1 ใช้อักขระควบคุมที่มีอยู่แล้วใน ASCII

2 สามารถจะใช้ภาษาในระดับสูงควบคุมได้ เช่น ภาษาเบสิก ภาษาปาสคาล

3 ต้องการบัฟเฟอร์สื่อสารแค่ 256 ไบต์

4 ระบบบริการข่าวสารด้วยคอมพิวเตอร์ โดยทั่วไปใช้โปรโตคอล XMODEM

ข้อดีของระบบการควบคุมการรับส่งข้อมูลแบบอนุกรมโดยการใช้ระบบการใช้โปรโตคอลก็คือ

1 โปรโตคอลบางชนิดสามารถเลือกขนาดของกลุ่มข้อมูลได้

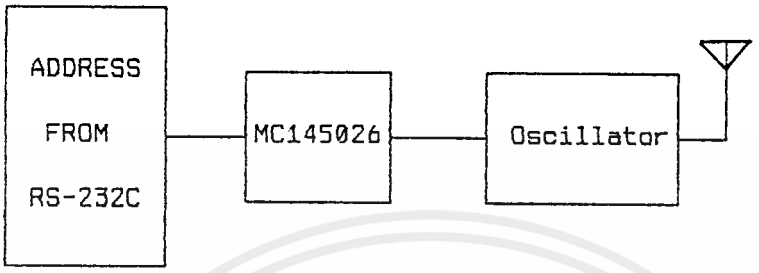
2 สามารถส่งข้อมูลที่ไม่ใช่ ASCII ได้ โดยไม่ต้องกลัวว่ารหัสนั้นจะไปทับกับรหัสควบคุมของ ASCII

3 การตรวจสอบโดยวิธี checksum มีความสามารถตรวจสอบความผิดพลาดได้ดีกว่าใช้บิตพาริตี้ในอะซิงโครนัส ในขณะที่บิตพาริตี้สามารถให้ประสิทธิผลได้ 95 เปอร์เซ็น แต่ checksum สามารถให้ประสิทธิผลถึง 99.5 เปอร์เซนต์ หากพบการผิดพลาดด้วยบิตพาริตี้ ไม่ทำให้เกิดการส่งใหม่เกิดขึ้นแต่ข้อผิดพลาดจาก checksum ทำให้เกิดการส่งข้อมูลมาใหม่

3.6 เครื่องรับส่ง

3.6.1 เครื่องส่ง

จะขออธิบายเฉพาะเครื่องส่งและเครื่องรับ R.F. เท่านั้น ส่วนระบบการติดต่อระหว่าง cpu และ port (8251) จะกล่าวต่อไปภายหลัง ลักษณะของภาคส่งเป็นดังรูป



ตัว MC14026 เป็นตัวEncoder ซึ่งจะอธิบายตัวของมันต่อไป ในที่นี้ขอให้ทราบว่า MC14026 เป็นตัวบังคับให้เกิดการออสซิลเลท คือเมื่อเอาพุทของMC14026 มีสถานะสูง(ใกล้ระดับไฟเลี้ยง) จะมีผลทำให้วงจรออสซิลเลท เกิดการออสซิลเลทที่ความถี่ในย่าน300MHz แต่ถ้าเอาพุทของ MC14026 อยู่ในสถานะต่ำ(ใกล้ระดับกราวด์) จะมีผลทำให้วงจรOSCหยุดการทำงาน เมื่อวงจร OSC เกิดการออสซิลเลทที่ความถี่นี้ คลื่นนี้จะเป็นคลื่นพาร์ของสัญญาณเอาพุทจาก MC14026 เพื่อส่งออกอากาศไป ลักษณะเช่นนี้อาจกล่าวได้ว่าเครื่องส่งแบบนี้ทำงานในแบบของ Pulse Modulation

หัวใจของวงจรนี้ คือวงจรกำเนิดความถี่ (Oscillator) หลักการเกิดการ oscillate มี 2 ประการ คือ

1. เฟสชิฟท์ (Phase shift) รอบลูป (Loop) ของระบบมีค่ารวมกันเป็น 0° หรือ 2nπ เรเดียน

2. อัตราการขยายรอบลูป (Loop gian) มีค่าเป็น 1 พอดี ถ้าต้องการให้ขนาดของสัญญาณที่ได้จากการออสซิลเลทมีขนาดคงที่

วงจรออสซิลเลทแบ่งเป็น 2 แบบ

1. วงจรออสซิลเลทคลื่นรูป sine

2. วงจรออสซิลเลทแบบไม่ใช้คลื่นรูป sine เช่น รูปสี่เหลี่ยม เป็นต้น

ในโครงงานนี้เราใช้วงจร osc แบบคลื่นรูป sine

วงจรoscillatorคลื่นรูปsine จะให้กำเนิดสัญญาณคลื่นรูปsine ออกมาตลอดเวลา โดยมีขนาดและความถี่คงที่ ชนิดของวงจร oscillator แบบนี้แบ่งได้จากอุปกรณ์ที่ใช้ ซึ่งแบ่งได้เป็น

_ วงจร oscillator ที่ใช้ LC

_ วงจร oscillator ที่ใช้ RC

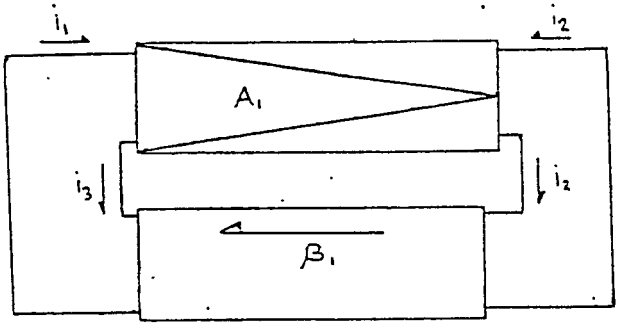
_ วงจร oscillator ที่ใช้ crystal

ลักษณะเฉพาะของแต่ละแบบคือ

1. วงจร oscillator ที่ใช้ LC
   - ความถี่เปลี่ยนแปลงได้ง่าย
   - โดยทั่วไปเสถียรภาพของความถี่ไม่ดี
   - มักใช้กันโดยทั่วไป
   - ใช้ในย่านความถี่สูง
2. วงจร oscillator ที่ใช้ RC
   - ใช้ในช่วงความถี่ต่ำ
   - ความถี่สามารถเปลี่ยนแปลงได้
   - เสถียรภาพของความถี่ไม่ดี
3. วงจร oscillator ที่ใช้ crystal
   - เสถียรภาพของวงจรดีเป็นพิเศษ
   - ใช้ในย่านความถี่สูง
   - การทำให้ความถี่เปลี่ยนแปลงทำได้ยาก

จากคุณสมบัติของวงจรทั้ง 3 แบบ จะเห็นว่าเราต้องการวงจร oscillator ที่ความถี่สูง เราจึงเลือกใช้วงจร oscillator แบบที่ใช้ LC

ต่อไปจะกล่าวถึงการออสซิลเลท โดยอุปกรณ์ที่ใช้เป็นลักษณะของกระแส คือเป็นทรานซิสเตอร์ เนื่องจากทรานซิสเตอร์ทำงานในลักษณะกระแสเช่นกัน เราสามารถเขียนโครงสร้างของวงจรได้ ดังรูป



จากรูปค่า A_1 และ β_1 จะเท่ากับ

$$A_1 = i_1 / i_2$$

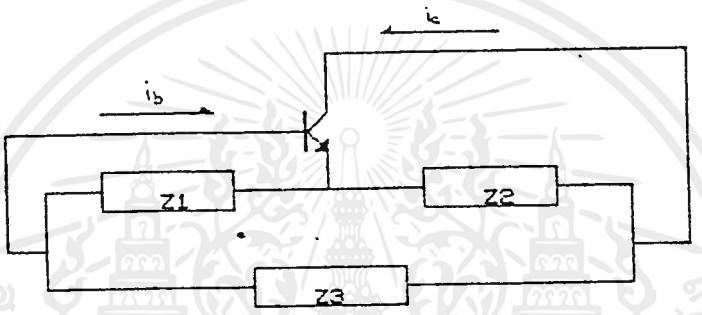
$$\beta_1 = i_3 / i_2$$

การออสซิลเลทจะเริ่มเมื่อ $i_1 < i_3$ และยะที่ $i_1 = i_3$ การออสซิลเลทจะเริ่มมีค่าดังที่

$A_1 * \beta_1 = i_2 / i_1 > 1$ (เริ่มการออสซิลเลท)

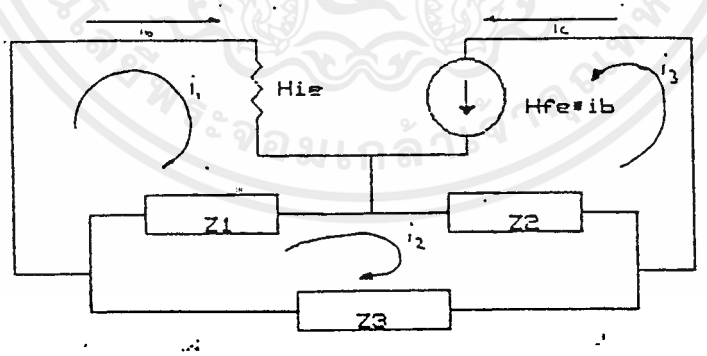
$A_1 * \beta_1 = i_3 / i_1 = 1$ (ออสซิสเลทแบบต่อเนื่อง)

ในวงจรที่ใช้ในโครงงานนี้เป็นวงจร oscillator แบบ Hartley จะขออธิบายวงจรแบบนี้จากรูป



รูปที่ 3.7 แสดงวงจร oscillator แบบเชื่อมต่อจุดสามจุด

เราสามารถเขียนเป็นวงจรสมมูลได้ดังรูป



รูปที่ 3.8 แสดงวงจรสมมูลของวงจรรูปที่ 3.7

จากรูปจะได้ว่า

$$(h_{ie} + Z_1)i_1 - (Z_1 * i_2) = 0$$

$$(-Z_1 * i_1) + (Z_1 + Z_2 + Z_3)i_3 + (Z_2 * i_3) = 0$$

เพราะว่า $i_1 = i_b$, $i_3 = i_c = h_{fe} * i_b$ ดังนั้นสมการ จะได้เป็น

$$(h_{ie} + Z_1)i_b - (Z_1 * i_2) = 0$$

$$(h_{fe} * Z_2 - Z_1)i_b + (Z_1 + Z_2 + Z_3)i_2 = 0$$

ตามสมการบนจะได้คำตอบก็ต่อเมื่อ $i_b = 0$ และ $i_2 = 0$ดังนั้นสัมประสิทธิ์ดีเทอมิแนนท์ จะต้องเป็น 0

$$(h_{ie} + Z_1)(Z_1 + Z_2 + Z_3) + Z_1(h_{fe} * Z_2 - Z_1)$$

$$= h_{ie}(Z_1 + Z_2 + Z_3) + z_1(z_2(1+h_{fe}) + z_3) = 0 \quad \text{--------(3)}$$

ถ้าค่า Z_1, Z_2 และ Z_3 เป็นค่า reactance จริงๆแล้ว jX_1, jX_2 และjX_3 จากสมการที่ (3) จะมีค่าเป็น

$$jh_{ie}(X_1 + X_2 + X_3) - X_1 * X_2(1+h_{fe}) - X_1 * X_3 = 0$$

$$X_1 + X_2 + X_3 = 0 \quad \text{--------(4)}$$

$$X_3(1 + h_{fe}) + X_3 = 0 \quad \text{--------(5)}$$

สมการที่(4)จะแสดงข้อกำหนดของความถี่สัญญาณ ส่วนสมการที่ (5)จะแสดงข้อกำหนดของศักดาสัญญาณ จากสมการที่(5) จะได้ว่า

$$X_2 = -(X_3)/(1 + h_{fe}) \quad \text{--------(6)}$$

จากสมการนี้จะเห็นว่าค่า X และ Xเป็นค่า reactance ต่างกันแทนสมการที่ (6) ลงในสมการที่ (4)ทำให้

$$X_1 = -(h_{fe} * X_3)/(1 + h_{fe}) \quad \text{--------(7)}$$

จากสมการนี้จะเห็นว่าค่า X และ X เป็นค่า reactance ต่างชนิดกันและจากสมการที่(6) และ (7) จะได้ว่า

$$h_{fe} = X_1 / X_2$$

ผลจากสมการเหล่านี้สรุปได้ว่า

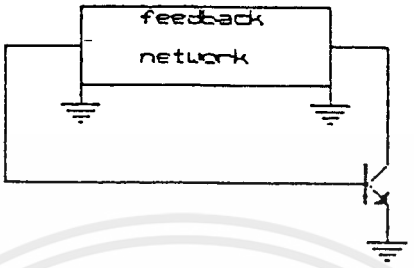
1. ค่า reactance ของ X_1และ X_2 จะต้องเป็นคนละชนิดกับ X_3
2. ค่า reactance ต่างๆ ควรจะเป็นตามข้อกำหนดดังนี้

$$|X_2| < |X_3|, |X_1| < |X_3|$$

3. h_{fe} ควรจะมีค่าเป็น $h_{fe} = X_1/X_2$

ค่า h_{fe} ในข้อ 3 เป็นค่าที่การออสซิลเลท เป็นแบบอิ่มตัวแล้ว ถ้าในกรณีของการเริ่มการออสซิลเลท ค่า h_{fe} ควรจะเป็นมากกว่า X_1 / X_2

ส่วนข้อ 1 กับข้อ 2 จะทำให้ได้วงจรพื้นฐานเป็น2 แบบคือแบบ Hartley กับ Colpoits คือ เป็น Hartley เมื่อ X_1 และ X_2 เป็น L และ X_3 เป็น C ส่วน Coplits เมื่อ X_1 และ X_2 เป็น C และ X_3 เป็น L จากรูป

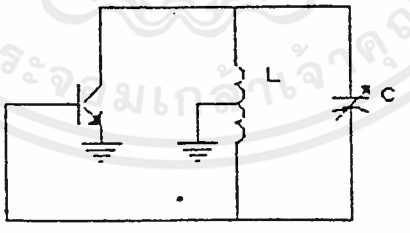


รูปที่ 3.9 แสดงวงจร Oscillator แบบ Common Emitter

เป็นวงจรแบบอิมิตเตอร์ร่วม (common emitter) จะเห็นว่า phase shiftของวงจรป้อนกลับ ต้องมีค่าเป็น 180 เพราะตัวขยายแบบคอมมอนอิมิตเตอร์เองก็มีเฟสที่ต่างกันระหว่างอินพุทกับเอาพุทอยู่ 180 อยู่แล้ว ซึ่งจะทำให้ phase shift รอบลูป มีค่าเป็น 360 พอดี

ในโครงงานนี้เราใช้วงจร Oscillator แบบ common emitter ชนิด Hartley (Hartley Oscillator)

จากโครงงานนี้ใช้วงจร Oscillator แบบ Harltey ซึ่งมีลักษณะดังรูป



รูปที่ 3.10 แสดงวงจร OSC แบบ HARTLEY

จากรูปสามารถอธิบาย ได้ว่า เมื่อตัวเหนี่ยวนำ และตัวเก็บประจุเกิด Resonance จะเสมือนเป็นความต้านทานค่าหนึ่งต่ออยู่ทางคอลเลคเตอร์ของทรานซิสเตอร์ จะเกิดการป้อนกลับมายังขาเบสของทรานซิสเตอร์โดยมี phase shift เป็น 180 เนื่องจากเป็นวงจรแบบ Common emitter ซึ่งมี phase shift อีก 180 ทำให้เกิด Oscillate ได้โดยความถี่ของการ

Oscillate คือ

$$f_o = 1/2\pi LC$$

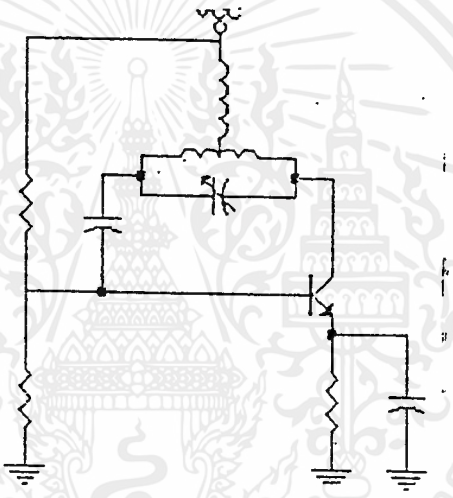
ในโครงงานนี้เราเลือกใช้C เป็นแบบปรับค่าได้ขนาด5-30pF เพื่อให้ง่ายต่อการปรับเลือกความถี่อื่นๆ มาใช้ ฉนั้นเราจึงสามารถคำนวณหาค่า L ได้เมื่อกำหนดให้ค่า C มีค่าเท่ากับ 15 pF จากสมการ $f_o = 1/2\pi LC$

$$300*10^{6} = 1/2\pi L*15*10^{-12}$$

$$L = 18.8 \text{ nH}$$

ซึ่งเป็นค่าที่น้อยจึงเลือกทำ L บนลายทองแดง

วงจรที่ใช้ในโครงงานนี้เป็นดังรูป

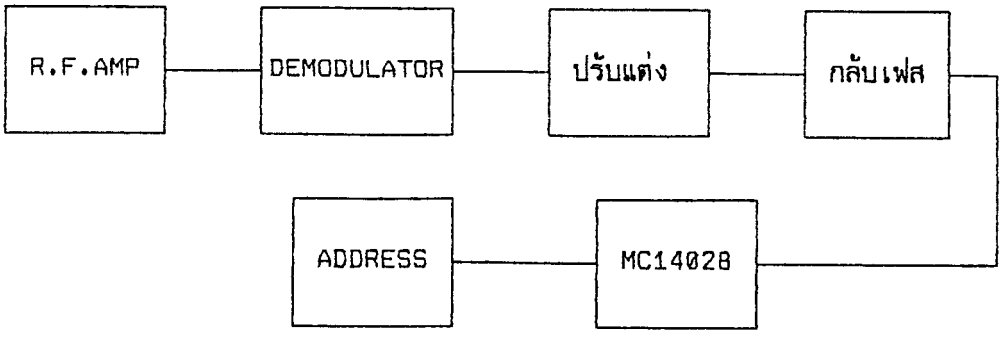


รูปที่ 3.11 แสดงวงจรจริงที่ใช้พร้อมการ bias

วงจรจริงที่ใช้ในโครงงานนี้จะมีการต่อตัวเก็บประจุ Bypass ไว้ที่ขาอีมิตเตอร์ เพื่อไว้ในการเพิ่ม gain ที่ความถี่สูง

3.6.2 <u>เครื่องรับ</u>

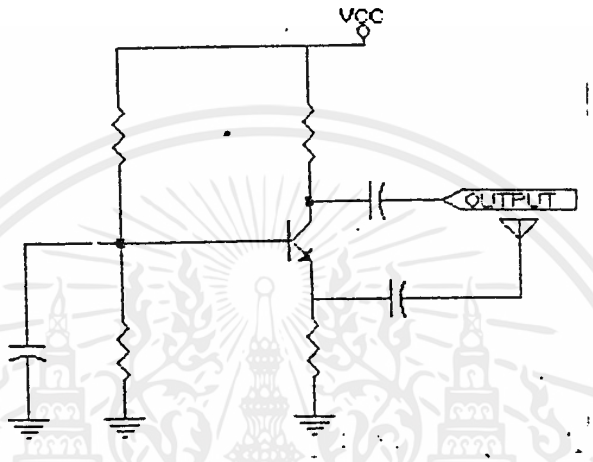
หลักการของเครื่องรับสามารถเขียนเป็น block diagram ได้ดังนี้



ซึ่งอธิบายได้เป็นส่วนๆ ได้ดังนี้

3.6.2.1 RF Amplifier

เนื่องจากสัญญาณที่เข้ามาทางสายอากาศนั้นมีค่าต่ำมาก ดังนั้นก่อนที่จะนำสัญญาณนี้ไปใช้จึงควรทำการขยายสัญญาณก่อน ซึ่งวงจรขยายเป็นดังรูป



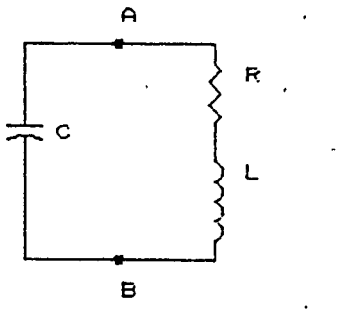
รูปที่ 3.12 แสดงวงจร R.F. Amplifier

เป็นวงจรขยายแบบ common base เนื่องจากวงจรขยายแบบนี้จะใช้ Frequency Response สูงกว่าวงจรขยายแบบอื่นๆ กล่าวคือ มีช่วงของ Bandwidth กว้าง จึงนำมาใช้ขยายสัญญาณ ในโครงงานนี้ได้เป็นอย่างดี อัตราขยายของวงจรนี้ เท่ากับ

$$A_v = Z_C/Z_E$$

3.6.2.2 วงจรเลือกความถี่ (Demodulator)

เมื่อได้ขนาดของสัญญาณใหม่พอสมควรแล้ว เราจะนำสัญญาณนี้มาแยก เอาคลื่นพาร์ออกโดยใช้วงจรเลือกความถี่ ซึ่งวงจรนี้ประกอบด้วย Tank Circuit ซึ่งประกอบด้วยตัวเหนี่ยวนำ และตัวเก็บประจุดังรูป



รูปที่ 3.13 แสดง PARALLEL RESONANCE

ที่ความถี่ Resonance จะทำให้ ซึ่งหาความถี่นี้ได้จากสมการ

$$W_c = 1/\ LC$$

ที่ความถี่นี้ ทำให้ค่าImpedance ระหว่างจุดA และ B มีค่าสูงสุด ทำให้กระแสไหลผ่านน้อยสุด ในที่นี้เราตั้ง f_c ไว้ที่ 300 MH_Z ทำให้สามารถคัดคลื่นพาร์ออกไปได้

3.6.2.3 วงจรปรับแต่งสัญญาณ

เมื่อได้สัญญาณo/p จากdemodulator แล้ว จะยังไม่สามารถนำสัญญาณนี้ไปใช้งานได้ เนื่องจากขนาดสัญญาณ และรูปร่างของสัญญาณเพี้ยนไปจากที่ส่งออกมา ดังนั้นจึงต้องนำสัญญาณนี้ไปเข้าวงจรปรับแต่งสัญญาณก่อน โดยนำสัญญาณมาเข้าวงจรขยายแบบ Common Emitter ซึ่งจะกลับเฟสไป 180 จากนั้น นำไปเข้าในวงจร Comparator เปรียบเทียบแรงดัน จากนั้นจึงนำสัญญาณมาเข้าวงจร Common emitter เพื่อกลับเฟส 180 อีกครั้งหนึ่ง เพื่อให้ได้สัญญาณที่สามารถนำไปใช้งานได้จริง

3.7 การทำงานของ เอ็นโค็ด เดอร์และดีโค็ด เดอร์

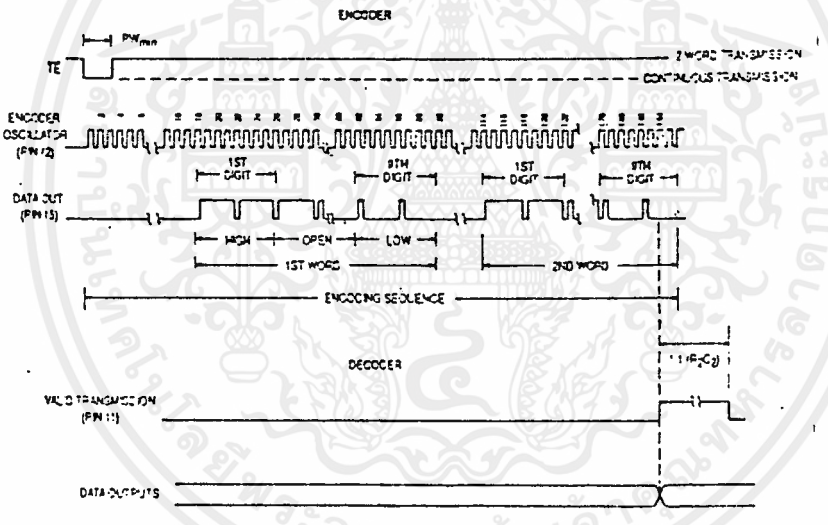
ในส่วนที่เกี่ยวข้องกับการ เข้ารหัสอาศัยการทำงานของไอซี เข้ารหัส เบอร์ MC14026 ซึ่งมีการรับข้อมูลแบบขนาน และส่งข้อมูลออกแบบอนุกรม รหัสทางอินพุตของ MC14026 นี้ จะสามารถ เข้ารหัสได้ถึง 3 สถานะคือ เป็นได้ทั้งระดับลอจิก "0" ระดับลอจิก "1" และสภาวะอิมพีแดนซ์สูงคือปล่อยลอยไว้

โครงสร้างของ MC14026 นี้จะมีขาแอดเดรสอยู่ 9 ขา คือ A1-A9 และ มีขาเอาพุต 1 ขา คือขา 15 ซึ่ง เป็นขาสำหรับส่งสัญญาณจากขา A1-A9 ที่เป็นข้อมูลแบบขนานไปยังวงจรภายนอก

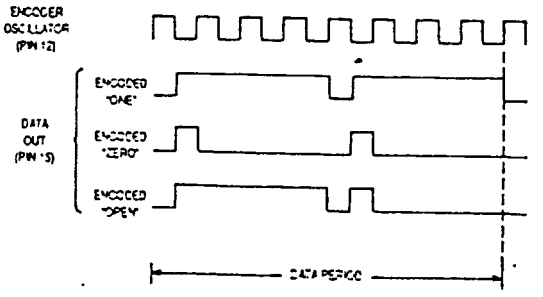
แบบอนุกรม ขาที่ควบคุมการทำงานของ MC14026 คือ ขา TE สัญญาณดิจิตอลทางอินพุตจะถูกส่งไปทางเอาพุตได้ต่อเมื่อขา TE อยู่ในสถานะ "0" เท่านั้น

จากคุณสมบัติที่ว่าอินพุตของไอซีเป็นได้ทั้งสามสถานะ ดังนั้นจึงสามารถเข้ารหัสได้ถึง $(2^3) = 19,683$ ที่ไม่ซ้ำกัน

ในส่วนการถอดรหัสอาศัยการทำงานของไอซีถอดรหัสเบอร์ MC14028 ซึ่งมีการรับข้อมูลแบบอนุกรม และเมื่อตรงรหัสจะทำให้เกิดสัญญาณเอาพุตขึ้น โครงสร้างของ MC14028 นี้จะมีขาแอดเดรสอยู่ 9 ขา คือ A1-A9 และ มีขาเอาพุต 1 ขา คือขา 11(V_T) ซึ่งเป็นขาสำหรับส่งสัญญาณบอกว่ารหัสที่ส่งมาจากตัวเข้ารหัส ตรงกับ A1-A9 ที่ตั้งเอาไว้โดยข้อมูลที่ส่งมาเป็นข้อมูลแบบอนุกรมจะเข้าขา data in ก็คือขา 9 ซึ่งถ้าเป็นรหัสเดียวกัน จะทำให้ขา V_T มีสถานะเป็น "1" ถ้ารหัสไม่ตรงกัน จะทำให้มีสถานะเป็น "0"ซึ่ง block diagram เป็นดังรูป



รูปที่ 3.14 block diagram



รูปที่ 3.15 รูปแสดงการเข้ารหัสข้อมูล

3.8 โปรแกรมเรสซิเดนต์

โปรแกรมเรซิเดนต์โครงงานนี้เป็นโปรแกรมที่ใช้งานบนระบบปฏิบัติการ (Operating System) MS-DOS ซึ่ง MS-DOS เป็นระบบปฏิบัติการที่ถูกออกแบบมาในลักษณะของ Single Task คือ ทำงานกับโปรแกรมเพียงหนึ่งโปรแกรมในเวลาหนึ่ง นั่นคือ ในการที่จะเรียกใช้งานโปรแกรมใดโปรแกรมหนึ่งบน MS-DOS จะต้องโหลดโปรแกรมนั้นจากดิสก์ลงมาในหน่วยความจำก่อนและการที่จะใช้งานโปรแกรมอื่นนั้น จะต้องรอจนกว่าโปรแกรมเดิมที่โหลดลงในหน่วยความจำนั้น ทำงานเสร็จสิ้นเสียก่อน ดังนั้นเพื่อที่จะให้สามารถทำงานหลายอย่างได้ หรือ เรียกใช้โปรแกรมอื่นในขณะที่ทำงานอีกโปรแกรมหนึ่งอยู่วิธีทำก็คือ จะต้องโหลดเอาโปรแกรมที่ต้องการเรียกใช้งานนั้นฝัง หรือค้างอยู่ในหน่วยความจำ (StayResident) ก่อนที่จะโหลดเอาโปรแกรมอื่นลงไปเพื่อทำงาน เนื่องจากโปรแกรมยังค้างอยู่ในหน่วยความจำ จึงทำให้โปรแกรมพร้อมที่จะถูกเรียกใช้งานได้ตลอดเวลา

หลักการ และเทคนิคโปรแกรมเรซิเดนต์นั้น ความจริงถูกเก็บไว้เป็นความลับของผู้พัฒนาโปรแกรมพวกนี้ ในโปรแกรมมีการเรียกใช้ฟังก์ชั่นของดอส ซึ่งเป็นฟังก์ชันที่ไม่มีบอกในคู่มือการใช้งานของดอสซึ่งจะได้กล่าวต่อไป

3.8.1 การฝังโปรแกรมลงหน่วยความจำ

ในการทำงานโปรแกรมตามปกตินั้น โปรแกรมจะถูกอ่านจากดิสก์ลงในหน่วยความจำที่ดอสจัดให้เรียกว่า Transient Program Area และหลังจากจบการทำงานโปรแกรม หน่วยความจำส่วนนี้จะกลับเป็นที่ว่างอีกเช่นเดิม การที่จะฝังโปรแกรมลงในหน่วยความจำนั้น ดอสเองเตรียมบริการไว้ให้ 2 วิธีด้วยกันคือ

-อินเตอร์รัพท์ 27H

DX = Offset ของไบท์สุดท้ายของโปรแกรมบวกด้วย 1

(อ้างอิงจาก Program Segment Prefix)

CS = Segment ของ Program Segment Prefix

การฝังโดยวิธีนี้ จะได้ขนาดสูงสุด 64 กิโลไบท์เท่านั้น

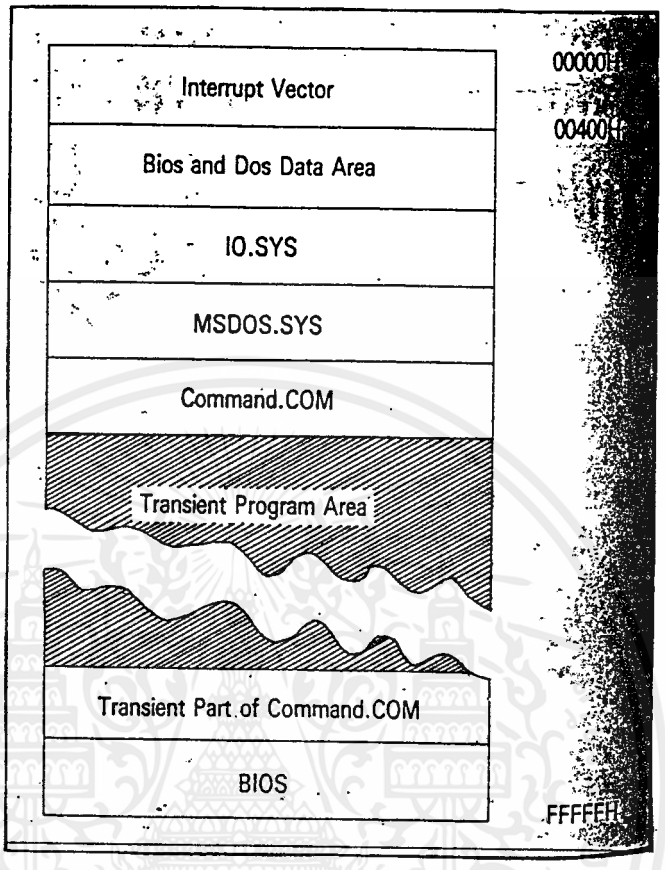
-อินเตอร์รัพท์ 21H ฟังก์ชันบริการ 31H

AH = 31h

AL = Exit Code

DX = ขนาดของโปรแกรมที่จะฝังลงในหน่วยความจำเป็นพารากราฟ (คือจำนวนไบท์ของโปร

แกรมที่จะฝังหารด้วย 16)



รูปที่ 3.16 แสดงหน่วยความจำหลังโหลดโปรแกรมเพื่อจะทำงาน

3.8.2 การเรียกใช้งานโปรแกรมเรสซิเดนต์

หลังจากฝังโปรแกรมลงในหน่วยความจำแล้ว การเรียกใช้งานโปรแกรมเรสซิเดนต์จะถูกเรียกผ่านทางอินเตอร์รัพต์ ดังนั้น โปรแกรมเรสซิเดนต์จึงต้องทำการเปลี่ยนแปลงค่าในตารางอินเตอร์รัพต์เวกเตอร์ ให้มาชี้มาที่ฟังก์ชันที่จะให้ทำงานแทน

โปรแกรมเรสซิเดนต์ส่วนใหญ่แล้ว มักจะปลุกหรือเรียกโปรแกรมรูทีนที่จะป๊อปอัพให้ขึ้นมาทำงานโดยการกดคีย์พิเศษ (Hot Key) ที่กำหนดเอาไว้ การทำงานลักษณะนี้ทำได้โดยการเขียนโปรแกรมอินเตอร์รัพต์ที่เกี่ยวกับคีย์บอร์ดใหม่ เมื่อมีการกดคีย์พิเศษโปรแกรมอินเตอร์รัพต์คีย์บอร์ดใหม่ที่ถูกเขียนขึ้นก็จะทำการเรียกไปยังรูทีนหรือฟังก์ชันที่ต้องการให้ทำงาน อินเตอร์รัพต์ที่เกี่ยวกับคีย์บอร์ดที่ใช้กันอยู่มี 2 อินเตอร์รัพต์ คือ อินเตอร์รัพต์ 09h ซึ่งเป็นฮาร์ดแวร์อินเตอร์รัพต์ และอินเตอร์รัพต์ 16h ซึ่งเป็นซอฟแวร์อินเตอร์(เป็นบริการของไบออสในการอ่านคีย์) ความแตกต่างระหว่างอินเตอร์

รัพท์ทั้งสองคือ อินเตอร์รัพท์ 16h นั้นเป็นซอฟแวร์อินเตอร์รัพท์ จะเกิดขึ้นก็ต่อเมื่อมีการเรียกใช้จากโปรแกรมอื่นๆ (อินเตอร์รัพท์ 16h มีฟังก์ชันบริการอยู่หลายอย่างที่สำคัญ คือ การอ่านตัวอักษรถัดไปจากบัฟเฟอร์ของคีย์บอร์ด ตรวจสอบว่ามีตัวอักษรพร้อมจะนำเข้าหรือไม่ และรายงานสถานภาพของคีย์บอร์ด ซึ่งได้แก่ ปุ่ม SHift,Caps-lock,Scroll-lock,Insert,Ctrl และ Alt) ส่วนอินเตอร์รัพท์ 09h นั้น เป็นฮาร์ดแวร์อินเตอร์รัพท์ซึ่งจะเกิดขึ้นทุกครั้งที่มีการกดหรือปล่อยคีย์บอร์ด การทำงานของอินเตอร์รัพท์นี้จะอ่านค่ารหัสสแกน (Scan Code) ของคีย์ที่ถูกกดจากพอร์ต 60h (อ่านมา 1 ไบต์) แล้วทำการเปลี่ยนเป็นรหัสแอสกี (เปลี่ยนเป็น 2 ไบต์ คือ ไบต์ต่ำ เป็นรหัสแอสกีของคีย์ที่กดและไบต์สูง เป็นรหัสสแกน) แล้วนำไปเก็บไว้ในบัฟเฟอร์ของคีย์บอร์ด สำหรับใช้งานโดยโปรแกรมอื่นๆซึ่งการอ่านก็จะผ่านทางอินเตอร์รัพท์หมายเลข 16h นั่นเอง

จากการทำงานของอินเตอร์รัพท์ทั้งสองที่กล่าวมาแล้ว จะพบว่าอินเตอร์รัพท์หมายเลข 09h จะตอบสนองทันทีทันใดได้ดีกว่าอินเตอร์รัพท์ 16h แต่ปัญหาที่สำคัญคือ ถ้าใช้อินเตอร์รัพท์หมายเลข 09h เป็นตัวปลุกรูทีนป๊อปอัพ ในขณะที่โปรแกรมเรสซิเดนต์ตื่นขึ้นมาพร้อมที่จะทำงาน สภาวะนั้นดอสอาจจะทำงานอยู่ในฟังก์ชันที่เป็น Non-reentran ก็ได้ ซึ่งจะไม่ปลอดภัยสำหรับการทำงานของโปรแกรมหรือรูทีนป๊อปอัพ แต่อย่างไรก็ตาม สามารถจะหลบเลี่ยงได้โดยการตรวจสอบสภาวะของดอสก่อนที่จะให้โปรแกรมหรือรูทีนป๊อปอัพ

การกำหนดคีย์พิเศษนั้น สามารถกำหนดเป็นคีย์รวมระหว่างคีย์ Ctrl,Alt,Shift-left หรือ Shift-right กับคีย์อื่นๆได้ ซึ่งคีย์ Ctrl, Alt, Shift-left และ Shift-right สามารถตรวจสอบได้จากตำแหน่งในพื้นที่ของไบออส 0:417h โดยค่าต่างๆจะอยู่ในตารางที่ 3.7 (สาเหตุที่ไม่ใช้คีย์ Insert,Num lock, Scroll-lock และ Cap-lock ก็เพราะว่า สถานะของคีย์พวกนี้จะเปลี่ยนแปลงไปมาเมื่อกดคีย์แต่ละครั้ง แต่สำหรับคีย์ Ctrl, Alt, Shift-left และ Shift-right สถานะจะถูกเซ็ตเมื่อได้รับการกดคีย์ค้างเอาไว้และเมื่อปล่อยคีย์สถานะจะถูกเคลียร์ซึ่งสามารถตรวจสอบได้ง่ายกว่า) และสำหรับคีย์อื่นนั้นจะถูกอ่านตรงๆจากพอร์ต61h ค่าที่ได้จะเป็นรหัสสแกนดังในตารางที่ 3.8 ซึ่งแสดงรหัสสแกนประจำคีย์แต่ละคีย์

ในการปลุกโปรแกรมเรสซิเดนต์ของโครงงานนี้จะใช้อินเตอร์รัพท์ ที่เกิดจากสัญญาณนาฬิกา (Timeinterrupt) เป็นตัวตรวจสอบสถานะของขา DSR ของพอร์ตอนุกรม และตรวจสอบสภาวะของดอสด้วย ซึ่งถ้าดอสอยู่ในสภาวะหรือทำงานฟังก์ชันที่ปลอดภัยจึงจะเรียกโปรแกรมหรือรูทีนป๊อปอัพขึ้นมาทำงานเพื่อการรับส่งข้อมูล ส่วนคีย์พิเศษ (hot key) นั้นใช้ในกรณีที่ต้องการส่งข้อมูลไปให้ terminal ตัวอื่น

บิตที่ 7 6 5 4 3 2 1 0	ความหมาย	
*	Insert	
. *	Caps lock	
. . *	Num lock	Active = 1
. . . *	Scroll lock	Non Active = 0
. . . . * . . .	Alt	
. . . . . * . .	Ctrl	
. . . . . . * .	Left Shift	
. . . . *	Right Shift	

ตารางที่ 3.7 สถานะของคีย์บอร์ดที่ตำแหน่งหน่วยความจำ 0000:0417h

คีย์ฟังก์ชัน		คีย์ที่เป็นตัวอักษร								คีย์ตัวเลข	
ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส
F1	59	'	41	Tab	15	A	30	Z	44	ESC	1
F2	60	1	2	Q	16	S	31	X	45	Home	71
F3	61	2	3	W	17	D	32	C	46	Up	72
F4	62	3	4	E	18	F	33	V	47	PgUp	73
F5	63	4	5	R	19	G	34	B	48	PrtSc	55
F6	64	5	6	T	20	H	35	N	49	←	75
F7	65	6	7	Y	21	J	36	M	50	5	76
F8	66	7	8	U	22	K	37	,	51	→	77
F9	67	8	9	I	23	L	38	.	52	Minus	74
F10	68	9	10	O	24	:	39	/	53	End	79
		0	11	P	25	"	40	Right Shift	54	Down	80
		-	12	[	26	left Shift	42	ALT	56	PgDn	81
		=	13	]	27			Spacbar	57	Ins	82
		\	43	Enter	28			Caps look	58	Del	83
		back space	14							Plus	78

ตารางที่ 3.8 รหัสแกนประจำคีย์บอร์ดแต่ละคีย์

3.8.3 ข้อจำกัดของดอส (Non reentant)

เนื่องจากดอสถูกออกแบบมาเป็นโอเอสที่ทำงานแบบ Single task ดังนั้นข้อจำกัดของดอสที่สำคัญก็คือ ฟังก์ชันบริการต่างๆของดอส (int 21h) นั้นไม่สามารถจะเรียกซ้ำซ้อนได้ (Nonreentrant) เนื่องจากดอสสร้างตัวแปรและกำหนดใช้สแต็กในหน่วยความจำที่แน่นอนตำแหน่งหนึ่ง การเรียกใช้ฟังก์ชันของดอสซ้ำซ้อนก่อนที่ดอสจะทำงานของฟังก์ชันนั้นเสร็จ ก็จะทำให้เกิดการทำลายข้อมูลและสแต็กของเดิมได้ ซึ่งทำให้เครื่องหยุดการทำงานได้ (แต่ก็มีวิธีการแก้ไข เพื่อที่จะสามารถเรียกใช้ฟังก์ชันของดอสบางอย่างได้อย่างปลอดภัย ซึ่งจะกล่าวถึงต่อไป) ดังนั้นในฟังก์ชันที่เป็นอินเตอร์รัพต์รูทีนในโปรแกรมเรสซิเดนต์จึงไม่สามารถใช้ฟังก์ชั่น printf, puts ฯลฯ ได้

การเรียกใช้ฟังก์ชันของดอส ดอสจะเก็บค่าของสแต็กรีจิสเตอร์เอาไว้ แล้วเปลี่ยนมาใช้สแต็กของดอสซึ่งดอสสร้างขึ้นเอง พร้อมกับเก็บค่ารีจิสเตอร์ที่สำคัญลงสแต็กเมื่อทำงานเสร็จสิ้นจึงจะคืนค่าต่างของรีจิสเตอร์รวมทั้งสแต็กเดิมให้กับสแต็กรีจิสเตอร์ดังเช่นในระหว่างที่อยู่ในการทำงานของฟังก์ชันของดอส ถ้าโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาทำงานและมีการเรียกใช้ฟังก์ชันของดอสอีกในโปรแกรมหรือรูทีนป๊อปอัพ ดอสก็จะเก็บรีจิสเตอร์ต่างๆของโปรแกรมเรสซิเดนต์ ทับรีจิสเตอร์ของโปรแกรมที่เรียกใช้ดอสครั้งแรก ดังนั้นหลังจากจบฟังก์ชันดอสและออกจากโปรแกรมหรือรูทีนป๊อปอัพแล้ว การทำงานของฟังก์ชันดอสที่เกิดจากการเรียกใช้ครั้งแรกก็จะไม่ถูกต้อง เนื่องจากค่าในสแต็กเดิมถูกทำลายอาจทำให้เครื่องหยุดทำงานได้

อย่างไรก็ตาม ฟังก์ชันของดอส (int 21h) ถูกแบ่งออกเป็น 2 กลุ่มด้วยกันคือ ฟังก์ชันที่เกี่ยวกับคีย์บอร์ด จอภาพ และเครื่องพิมพ์ (Character I/O) ตั้งแต่ฟังก์ชัน 00h ถึง 0Ch และฟังก์ชันตั้งแต่ 0Dh ขึ้นไป ฟังก์ชันทั้งสองกลุ่ม จะใช้สแต็กแยกกันคนละส่วน ดังนั้นถ้าโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาในขณะที่ดอสทำงานอยู่ในฟังก์ชันกลุ่มใดกลุ่มหนึ่ง โปรแกรมหรือรูทีนป๊อปอัพก็สามารถจะใช้ฟังก์ชันบริการของดอสอีกกลุ่มหนึ่งได้อย่างปลอดภัย ซึ่งโดยทั่วไปแล้วนั้นฟังก์ชันในกลุ่มแรกคือ ฟังก์ชัน 00h ถึง 0Ch ซึ่งเกี่ยวกับคีย์บอร์ด จอภาพและเครื่องพิมพ์ นั้นสามารถหลบเลี่ยงไปใช้ไบออสสำหรับอ่านคีย์หรือกระทำต่อหน่วยความจำของจอโดยตรงได้สำหรับการแสดงผล ดังนั้นโปรแกรมเรสซิเดนต์ทั่วไปจึงนิยมทำงานในขณะดอสอยู่ในฟังก์ชันกลุ่มแรก (00h-0Ch)

ค่าในรีจิสเตอร์ ES:BX จะชี้ไปยังหน่วยความจำ InDOS ในระหว่างที่ดอสทำงานในฟังก์ชันกลุ่มที่ 2 ของอินเตอร์รัพต์ 21h (ฟังก์ชัน 0Dh ขึ้นไป) ค่าใน InDOS จะมีค่าเป็น 1 และเมื่อดอสทำงานในฟังก์ชันกลุ่มที่ 2 นี้เสร็จสิ้น ก็จะเคลียร์ค่าใน InDOS ให้เป็น 0

จะเห็นว่าสถานะของดอสที่ปลอดภัยต่อการทำงานของโปรแกรมเรสซิเดนต์ก็คือ ในขณะที่ค่าใน InDOS เป็น 0 ดังนั้นเมื่อโปรแกรมได้รับการกดคีย์พิเศษเพื่อปลุกโปรแกรมเรสซิเดนต์ขึ้นมา ดังที่กล่าวในหัวข้อที่แล้ว ในอินเตอร์รัพต์ของคีย์บอร์ด (อินเตอร์รัพต์ 09h) จะเซ็ตค่าแฟล็ก pop_flag ให้มีค่าเป็น 1 เพื่อเตรียมพร้อมสำหรับการเรียกโปรแกรม หรือรูทีนป๊อปอัพขึ้นมาทำงาน จากนั้นอินเตอร์รัพต์ไทม์เมอร์(1Ch)ซึ่งกระตุ้น 18.2 ครั้งต่อนาที จะคอยตรวจสอบ pop_flag และ InDOS (ค่าเป็น 0) ถ้าพร้อมก็จะเรียกโปรแกรมหรือรูทีนป๊อปอัพขึ้นมาทำงานทันที แต่จะต้องตรวจสอบสภาพของการทำงานดิสก์ก่อน ซึ่งจะกล่าวในต่อไป

มีกรณีพิเศษอยู่กรณีหนึ่งที่โปรแกรมหรือรูทีนป๊อปอัพสามารถทำงานได้ทันทีเมื่อค่าใน InDOS เป็น 1 คือในขณะที่ดอสรอการกดคีย์(ขณะอยู่ในเครื่องหมายเตรียมพร้อม) ดอสจะเรียกอินเตอร์รัพต์ 28h ซึ่งคือสถานะที่ดอสว่างอยู่โปรแกรมหรือรูทีนป๊อปอัพสามารถทำงานได้โดยไม่ต้องตรวจสอบเงื่อนไขอื่นอีก

3.8.4 การทำงานภายในรูทีนป๊อปอัพ

หลังจากที่สภาวะต่างๆรวมทั้งดอส พร้อมที่จะใช้รูทีนป๊อปอัพขึ้นมาทำงานโปรแกรมเรซิเดนต์ จะต้องทำการเก็บค่าต่างๆของสิ่งแวดล้อมที่เป็นอยู่เอาไว้ก่อนเรียกรูทีนป๊อปอัพ และเมื่อทำงานในรูทีนป๊อปอัพเสร็จเรียบร้อยแล้วก็ต้องคืนค่าต่างๆที่เก็บไว้ให้ถูกต้องด้วย ซึ่งงานที่เกี่ยวข้องต้องทำนั้นมีดังนี้

* สแต็ก
* อินเตอร์รัพต์ของ Control-Break และ Control-C
* อินเตอร์รัพต์ของ Critical error
* Program Segment Prefix
* พื้นที่รับส่งข้อมูลกับดิสก์ (Disk transfer address, DTA)
* จอภาพและเคอร์เซอร์

จะอธิบายรายละเอียดแยกเป็นหัวข้อต่อไปนี้

– การเปลี่ยนแปลงสแต็ก

ในขณะที่โปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาทำงาน สแต็กที่ใช้งานอยู่จะเป็นสแต็กของโปรแกรมที่

หางานค้างอยู่หรืออาจจะเป็นสแต็กของตัวดอสเอง ซึ่งขนาดของสแต็กอาจจะมีไม่เพียงพอสำหรับรูทีนป๊อบอัพจะทำงานได้ ดังนั้นจำเป็นต้องทำการเปลี่ยนสแต็กให้มาใช้งานสแต็กของโปรแกรมเรสซิเดนต์เอง โดยการเปลี่ยนค่าในรีจีสเตอร์สแต็ก (SS และ SP) จากการกระทำนี้จะเห็นว่าโปรแกรมเรสซิเดนต์ที่จะสร้าง เป็นโปรแกรมที่ไม่สามารถเรียกซ้ำได้ (Non-reentrant) ซึ่งเราจะแก้ปัญหานี้ได้ โดยการสร้างแฟล็กขึ้นมาเพื่อใช้ตรวจสอบว่ากำลังทำงานอยู่ในรูทีนป๊อบอัพ โดยจะต้องกำหนดค่าแฟล็กนี้ให้เป็น 1 ก่อนจะทำการเปลี่ยนค่าสแต็กและหลังจากทำงานในรูทีนป๊อบอัพเสร็จสิ้นและคืนค่าสแต็กเดิมเรียบร้อยแล้ว จึงจะกำหนดแฟล็กนี้ให้เป็น 0

- อินเตอร์รัพต์ของ Control-Break และ Control-C

ตามปกติการที่ผู้ใช้โปรแกรมกดคีย์ Ctrl-Break หรือ Ctrl-C จะทำให้โปรแกรมที่ทำงานอยู่ในขณะนั้นเลิกการทำงานดังนั้นเพื่อป้องกันเหตุการณ์นี้เกิดกับโปรแกรมเรสซิเดนต์ก่อนจะเข้ารูทีนป๊อบอัพจะต้องเปลี่ยนอินเตอร์รัพต์ทั้งสองนี้ (อินเตอร์รัพต์ 1bh สำหรับ Control-Break และอินเตอร์รัพต์ 23h สำหรับ Control-C) โดยทำการเปลี่ยนค่าในตารางอินเตอร์รัพต์เวกเตอร์ให้มาชี้ที่ฟังก์ชันต่างๆของโปรแกรมเรสซิเดนต์ เมื่อป๊อบอัพขึ้นจะไม่สนใจ Control-Break และ Control-C เมื่อออกจากรูทีนป๊อบอัพก็คืนค่าเวกเตอร์อินเตอร์รัพต์ทั้งสองคืน นอกจากนี้อีกสิ่งหนึ่งที่ควรทำก่อนเรียกรูทีนป๊อบอัพก็คือ การบอกดอสให้เลิกสนใจ Control-Break ซึ่งทำได้โดยใช้ฟังก์ชันดอสอินเตอร์รัพต์ 21h ฟังก์ชันที่ 33h แต่อย่างไรก็ตามต้องเก็บสถานะของดอสในการตอบสนอง Control-Break เอาไว้เพื่อคืนให้แก่ดอสหลังจากออกจากรูทีนป๊อบอัพ ซึ่งในเทอร์โบซีได้เตรียมฟังก์ชันเอาไว้แล้วคือ getcbrk อ่านค่าการตอบสนอง Control-Break และsetcbek กำหนดค่าการตอบสนอง Control-Break

- อินเตอร์รัพต์ของ Critical Error

โปรแกรมที่ทำงานภายใต้ดอสเมื่อเกิดข้อบกพร่องในการติดต่ออุปกรณ์ต่างๆ เช่น ดิสก์ เครื่องพิมพ์ เป็นต้น ตัวอย่างของข้อพร่องได้แก่ การพยายามเขียนหรืออ่านดิสก์ขณะที่ฝาดิสก์ไดรฟ์ไม่ได้ปิด การพยายามพิมพ์ข้อความออกทางเครื่องพิมพ์แต่เครื่องพิมพ์ไม่พร้อมที่จะทำงาน เหตุการณ์เหล่านี้จะทำให้ดอสเรียกใช้โปรแกรมอินเตอร์รัพต์ 24h คืออินเตอร์รัพต์ Critical Error ซึ่งการทำงานของอินเตอร์รัพ 24h นี้คือ จะพิมพ์ข้อความแสดงข้อผิดพลาดที่เกิดจากอุปกรณ์เหล่านั้น พร้อมทั้งรอคำตอบจากผู้ใช้โปรแกรมว่า จะยกเลิกงานหรือจะพยายามทำงานนั้นอีกครั้ง หรือจะทำงานผิดไปต่อโดยไม่สนใจข้อผิดพลาดนั้นๆ Abort, Retry, Ignore? ซึ่งคำตอบที่ได้รับจากผู้ใช้โปรแกรมนั้น

อินเตอร์รัพต์ 24h จะส่งกลับไปยังดอสโดยผ่านทางรีจิสเตอร์ AL คำตอบคือ Abort ซึ่งถ้าเกิดขึ้นขณะโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาทำงานอาจจะทำความเสียหายให้กับระบบได้ ดังนั้นเพื่อแก้ไขปัญหานี้ผู้เขียนโปรแกรมเรสซิเดนต์จะต้องจัดการแก้ไขอินเตอร์รัพต์ 24h เสียใหม่เอง แต่สำหรับในโปรแกรมในตัวรงงานนี้จะทำการเปลี่ยนแปลงอินเตอร์รัพต์ 24h ใหม่ โดยส่งคำตอบให้ดอสทำงานต่อไปโดยไม่สนใจข้อผิดพลาด(ซึ่งทำได้โดยการกำหนดค่า "ศูนย์" ให้รีจิสเตอร์ AL ก่อนออกจากอินเตอร์รัพต์ 24h)

-. Program Segment Prefix (PSP) และพื้นที่รับส่งข้อมูลกับดิสก์ (Disk Transfer Area, DTA)

ตามปกติก่อนดอสจะนำโปรแกรมลงหน่วยความจำ เพื่อให้โปรแกรมทำงาน ดอสจะสร้างพื้นที่หน่วยความจำขนาด 256 ไบต์ เรียกว่า Program Segment Prefix (PSP) เอาไว้ส่วนบนสุดของโปรแกรม ซึ่งภายในจะมีข้อมูลสำหรับควบคุมการทำงานของโปรแกรมโดยเฉพาะที่สำคัญ คือ เรื่องดิสก์หรือไฟล์ โครงสร้างรายละเอียดของPSP ให้ไว้ในตารางที่ 3.9 และก็เป็นเพราะว่าดอสนั้นจะควบคุมการทำงานของโปรแกรมได้เพียงหนึ่งโปรแกรม ซึ่งขณะที่ปลุกโปรแกรมเรสซิเดนต์ขึ้นมาดอสก็ยังคงใช้ PSP ของโปรแกรมเดิมอยู่ ถ้าหากโปรแกรมเรสซิเดนต์มีการเขียนหรืออ่านไฟล์ก็อาจจะทำความเสียหายให้กับข้อมูลของโปรแกรมเดิมที่ทำอยู่ก่อนการปลุกโปรแกรมเรสซิเดนต์ได้ ดังนั้นในโปรแกรมเรสซิเดนต์ก่อนจะเรียกรูทีนป๊อปอัพ จึงควรจะบอกดอสให้เปลี่ยนการใช้ PSP จากของเดิมมาเป็น PSPของโปรแกรมเรสซิเดนต์

ในพื้นที่ของดอสจะมีหน่วยความจำส่วนหนึ่ง สำหรับเก็บตำแหน่งของ PSP ที่ดอสใช้ควบคุมโปรแกรมที่ทำงานอยู่ เช่นเดียวกับที่เก็บแฟล็ก InDOS ที่ใช้บอกสถานะการทำงานของดอสดังที่ได้กล่าวในหัวข้อที่แล้ว ในการเปลี่ยนแปลง PSP นั้นก็ทำโดยเปลี่ยนค่าในพื้นที่ส่วนนี้นั่นเอง ซึ่งดอสได้มีฟังก์ชันสำหรับใช้งานนี้อยู่ด้วยกัน 3 ฟังก์ชัน (เป็นฟังก์ชันที่ไม่เปิดเผย) คือฟังก์ชัน 50h กำหนด PSP ใหม่ ฟังก์ชัน 51h อ่านค่า PSP และฟังก์ชัน 62h สำหรับการอ่านค่า PSP เช่นกัน ตารางที่ 3.10 เป็นตารางสรุปฟังก์ชันที่ไม่เปิดเผยและใช้งานโปรแกรมเรสซิเดนต์

ฟังก์ชัน 50h และ 51h เริ่มมีตั้งแต่ดอสรุ่น 2.0 แต่ฟังก์ชัน 51h เมื่อใช้งานในอินเตอร์รัพต์ 28h พบว่ามีปัญหา(Bug)เกิดขึ้น เพราะฟังก์ชันนี้ไปใช้สแต็กร่วมกับฟังก์ชันอื่น อย่างไรก็ตามต่อมาดอสได้เพิ่มฟังก์ชัน 62h ขึ้นมาในดอสรุ่น 3.0 ซึ่งการทำงานจะเหมือนกับฟังก์ชัน 51h แต่ได้แก้ไขปัญหาที่เกิดขึ้นกับฟังก์ชัน 51h

เซตที่	ออฟเซ็ต	ขนาด (ไบต์)	ความหมาย
1	0000H-0001H	2	เป็นที่เก็บคำสั่ง INT 20H ซึ่งเป็นคำสั่งจบการทำงานของโปรแกรมเพื่อกลับไปสู่เอ็มเอสดอส
2	0002H-0003H	3	เซกเมนต์แอดเดรสของหน่วยความจำสูงสุด
3	0004H	1	สงวนไว้
4	0005H-0009H.	5	เป็นคำสั่ง LONG CALL ไปยังฟังก์ชันของเอ็มเอสดอส เราสามารถใช้คำสั่ง JMP กระโดดมาที่แอดเดรสนี้ เพื่อเรียกใช้ฟังก์ชันของเอ็มเอสดอสได้
5	000AH-000DH	4.	เก็บแอดเดรสของการเลิกงาน เมื่อโปรแกรมจบการทำงานด้วยคำสั่ง INT 20H หรือ INT 21H ด้วยฟังก์ชัน 00H, 31H หรือ 4CH
6	000EH-0011H	4	เก็บแอดเดรสที่โปรแกรมจะกระโดดไปทำงานเมื่อมีการกด CTRL-BREAK
7	00012H-0015H	4	เก็บแอดเดรสที่โปรแกรมจะกระโดดไปทำงานเมื่อเกิดข้อผิดพลาด ซึ่งส่วนใหญ่จะเป็นข้อผิดพลาดที่เกี่ยวข้องกับอุปกรณ์ไอโอ เช่น ดิสค์ไม่พร้อมทำงาน เนื่องจากไม่ได้ปิดฝา หรือเกี่ยวกับเครื่องพิมพ์ เช่นกระดาษหมด เป็นต้น
8	0016H-002BH	7	สงวนไว้
9	002CH-002DH	2	เป็นตัวชี้แอดเดรสของสภาพแวดล้อม (Environment) ว่าเก็บอยู่ที่ใด สภาพแวดล้อมนี้เป็นบริเวณที่เก็บข้อมูลต่าง ๆ เช่น เส้นทาง (PATH) ของการค้นหาไฟล์ หรือตัวแปรที่เกิดจากการใช้คำสั่ง SET หรือรูปแบบของสัญลักษณ์เตรียมพร้อมที่กำหนดด้วยคำสั่ง PROMPT เป็นต้น ข้อมูลสภาพแวดล้อมนี้จะเก็บอยู่ในรูป -ASCIIZ ซึ่งหมายถึงรหัสอัสกี้ที่ปิดท้ายด้วยศูนย์
10	002EH-005BH	45	สงวนไว้
11	005CH-006BH	15	เป็นที่เก็บบล็อกการควบคุมไฟล์ (File Control Block) ตัวที่ 1
12	006CH-007FH	19	เป็นที่เก็บบล็อกการควบคุมไฟล์ตัวที่ 2 จะสังเกตว่า FCB ตัวที่ 2 นี้จะซ้อนทับ FCB ตัวที่ 1 ในกรณีที่ใช้ FCB เพียงตัวเดียวจะไม่มีปัญหาใด แต่ถ้าต้องใช้ตัวที่ 2 ด้วย ต้องเก็บ FCB ตัวแรกก่อน
13	0080-00FFH	128	เป็นที่เก็บพารามิเตอร์ของบรรทัดคำสั่ง และใช้เป็นพื้นที่ส่งผ่านของดิสค์ (Disk Transfer Area) พื้นที่นี้จะเป็นบัฟเฟอร์หรือกันชนระหว่างดิสค์กับบริเวณที่เก็บข้อมูลจริง เมื่อมีการอ่านข้อมูลจากดิสค์สู่หน่วยความจำ ข้อมูลจะถูกอ่านลงมาที่นี่ก่อน แล้วจึงย้ายไปสู่บริเวณเก็บข้อมูลจริง ในกรณีการเขียนก็เช่นเดียวกัน

ตารางที่ 3.9 ส่วนต่างๆของ PSP และความหมาย

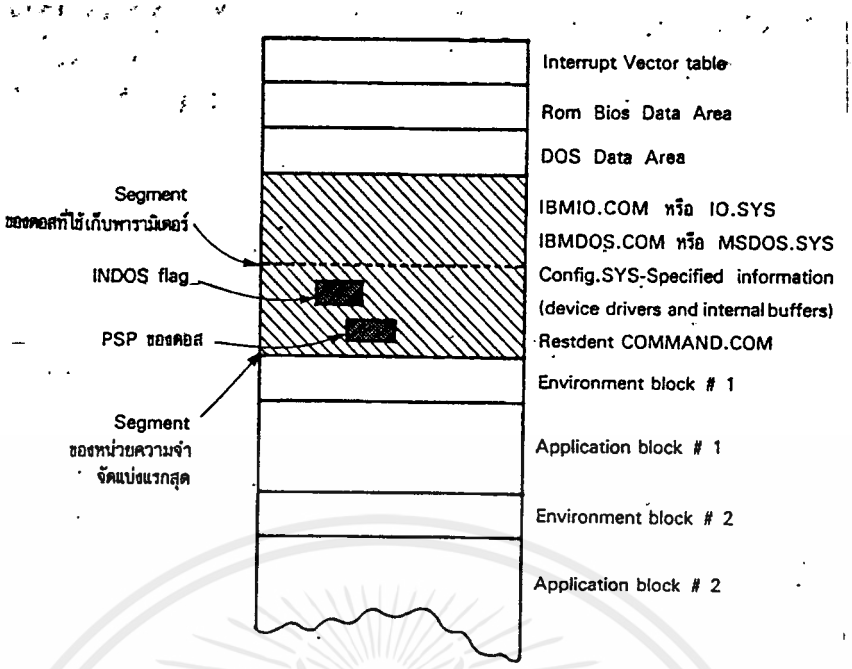
Function No.	Name of Function	Imput	Output	DOS version
034 h	Get INDOS Flag	AH = 034 h	ES = Segment of INDOS Flag BX = offset of INDOS Flag	2.0 and later
050 h	Set Current PSP	AH = 050 h BX = Segment of PSP to be made current	none	2.0 and later
051 h	Get PSP Segment	AH = 051 h	BX = Current PSP	2.0 and later
052 h	Get First MCB	AH = 052 h	ES : [BX−2] point to First MCB	2.0 and later
062 h	Get PSP Segment	AH = 062 h	BX = Current PSP	3.0 and later

ตารางที่ 3.10 สรุปฟังก์ชันของดอส(int 21h)ที่ไม่เปิดเผยและใช้งานในโปรแกรมเรสซิเดนต์

สำหรับพื้นที่รับส่งข้อมูลกับดิสก์(DTA)นั้น ปกติ(Default)แล้วจะอยู่เป็นส่วนหนึ่งของ PSP ซึ่งสามารถจะกำหนดที่ใหม่ได้โดยจะต้องจองเนื้อที่จากดอสก่อน พื้นที่รับส่งข้อมูลกับดิสก์(DTA)นั้นจะใช้เป็นที่สำหรับพักข้อมูล(Buffer)ในการเขียนหรืออ่านดิสก์ และเป็นไบท์ที่โปรแกรมที่ทำงานอยู่ขณะที่โปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมานั้น อาจจะจองหรือสร้าง DTA ขึ้นมาใหม่สำหรับใช้งานเอง ดังนั้นเมื่อโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาจึงต้องเก็บ DTA ของโปรแกรมที่ทำงานอยู่ก่อน เพื่อคืนให้กับดอสหลังจากที่รูทีนป๊อปอัพทำงานเสร็จ ฟังก์ชันสำหรับอ่านและกำหนด DTA อยู่ในตารางที่ 3.11 แต่สำหรับไลบรารี่ของเทอร์โบซีได้เตรียมฟังก์ชันไว้ให้แล้ว คือฟังก์ชัน getdta และ setdta

- จอภาพและเคอร์เซอร์

เรื่องของจอภาพและเคอร์เซอร์กับโปรแกรมเรสซิเดนต์ก็เป็นส่วนหนึ่งที่สำคัญ ซึ่งการทำงานของโปรแกรมเรสซิเดนต์อาจต้องใช้พื้นที่บางส่วนของจอภาพ ดังนั้นก่อนที่จะให้โปรแกรมเรสซิเดนต์ขึ้นมาทำงาน จะต้องเก็บสถานภาพของจอภาพเดิมเอาไว้เพื่อคืนให้หลังจากที่โปรแกรมเรสซิเดนต์ทำงานเสร็จ



รูปที่ 3.17 แสดงที่อยู่ของ InDOS flag ซึ่งได้จากฟังก์ชัน 34h และ PSP ของดอสซึ่งค้นหาโดยเริ่มจากเซกเมนต์ที่เก็บพารามิเตอร์ของดอสจนถึงหน่วยความจำจัดแบ่งอันแรกสุด

Function No	Function Name	input	out put
1Ah	Set Disk Transfer	AH = 1Ah	None
	Address	DS : DX = Disk Transfer Address	
2Fh	Get Disk Transfer	AH = 2Fh	ES : BX
	Address		Points to Disk Transfer Address

ตารางที่ 3.11 ฟังก์ชันของดอส(Int 21) ใช้กำหนดและอ่านพื้นที่ DTA

3.8.5 การป้องกันการจำลองโปรแกรมซ้ำ

การป้องกันการจำลองโปรแกรมเรซิเดนซ์ซ้ำสองทำได้ โดยการตรวจสอบสภาพโปรแกรมก่อนจะทำการติดตั้งโปรแกรม ซึ่งมีด้วยกัน 2 วิธี คือ

1.การค้นหาโปรแกรมในหน่วยความจำซึ่งดอสควบคุม คือ ค้นหาชื่อตัวโปรแกรมจากการจำลองครั้งแรกใน Memory Control Block ของดอส

2.ในการจำลองโปรแกรมครั้งแรกนั้น โปรแกรมจะติดตั้งชื่อหรือสัญญาลักษณ์ของโปรแกรมเอาไว้

ใน vector interrupt ที่ว่างอยู่ (ไว้สำหรับรูปโปรแกรมที่จะใช้งาน)ซึ่งได้แก่ interruptหมายเลข 60hถึง 67h โดยการเลือกเอา interrupt ที่ว่าง จากนั้นในการโหลดโปรแกรมครั้งต่อไปก็จะทำการตรวจสอบกับ interrupt หมายเลขต่างๆดังกล่าว ซึ่งถ้าพบก็จะไม่ติดตั้งโปรแกรม แต่จะเลิกโปรแกรมไปตามปกติ ทั้งสองวิธีนั้นวิธีแรกเป็นวิธีที่ถูกต้องที่สุด และวิธีที่ 2 นั้นง่ายสะดวกต่อการติดตั้งและตรวจสอบ

3.8.6 การติดต่อส่งข้อมูลให้กับโปรแกรมเรซิเดนซ์ที่ฝังโปรแกรมไปแล้ว

การส่งผ่านข้อมูลหรือ parameter ให้กับโปรแกรมเรซิเดนซ์ที่ฝังโปรแกรมไปแล้วนั้น ทำได้โดยการโหลดหรือเรียกใช้งานโปรแกรมเรซิเดนซ์อีกครั้งหนึ่ง ซึ่งส่งผ่านข้อมูลทาง register ผ่านทางอินเทอร์รัพ (60h. - 67h) ซึ่งโปรแกรมเรซิเดนซ์ที่โหลดก่อนได้ทำการเปลี่ยนแปลงให้เป็นรูทีนสำหรับควบคุมการทำงานของโปรแกรม ภายในรูทีนนี้จะทำการตรวจสอบค่า register ที่ได้รับมาว่าจะให้ทำอะไร

3.8.7 กฎและข้อหลีกเลี่ยงสำหรับการเขียนโปรแกรมด้วย TURBO C

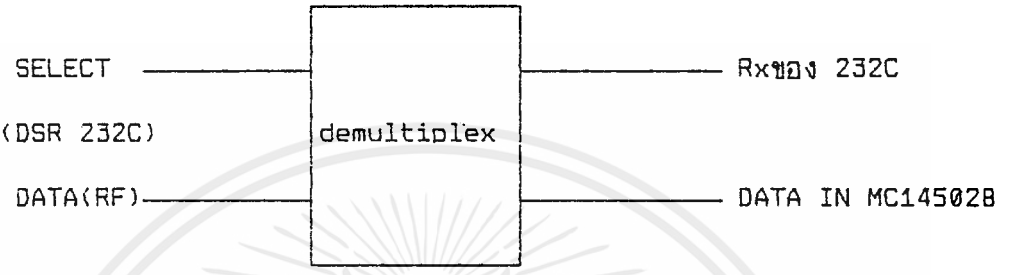
ใน TURBO C นั้น ก่อนจะทำงานที่ฟังชั่น main นั้นจะทำงานส่วน Start-up Code ของTURBO จบฟังชั่น main จึงจะคืนค่าเดิมให้ แต่ในโปรแกรมเรซิเดนซ์นั้นไม่ได้จบโปรแกรมตามปกติ ดังนั้น interrupt บางตัวอาจจะไม่ได้คืน ซึ่งจะทำให้ระบบเสียหายได้ ตัวอย่างเช่น interrupt หมายเลขศูนย์ (Divide-by-Zero) ได้ถูกเปลี่ยนก่อนเรียกฟังก์ชั่น main ซึ่งก่อนจะใช้ฟังก์ชั่น keepจะต้องคืนค่าให้ถูกต้องก่อน และเนื่องจาก TURBO C รุ่น 1.5 ขึ้นไป เก็บค่า interrup vectorไว้ที่ตัวแปร Zero DIv Vector ดังนั้นเราจึงจะสามารถคืน interrupt vector ได้อย่างถูกต้อง ปัญหาอีกอันหนึ่งก็คือเรื่องไลบรารี่คณิตศาสตร์ (floating point math library) ซึ่งจะใช้ interrupt บางหมายเลข และทำการเปลี่ยนแปลง interrupt vector ใน Start-up Code เช่นกัน ดังนั้นไม่ควรใช้ฟังก์ชั่นทางคณิตศาสตร์ (floating point) ในโปรแกรมเรซิเดนซ์ และที่สำคัญ อย่าใช้ฟังก์ชั่นใน library ที่ใช้ฟังก์ชั่นของดอส คือ ฟังก์ชั่น 0 ถึง 0ch

บทที่ 4

การทำงานของวงจรและซอฟแวร์ในส่วนอินเตอร์เฟสคอมพิวเตอร์และZ80

4.1 การทำงานของวงจรในส่วน Terminal

ในโครงงานนี้ใช้เครื่อง computer เป็นเครื่อง terminal โดยมีการทำงานดังนี้



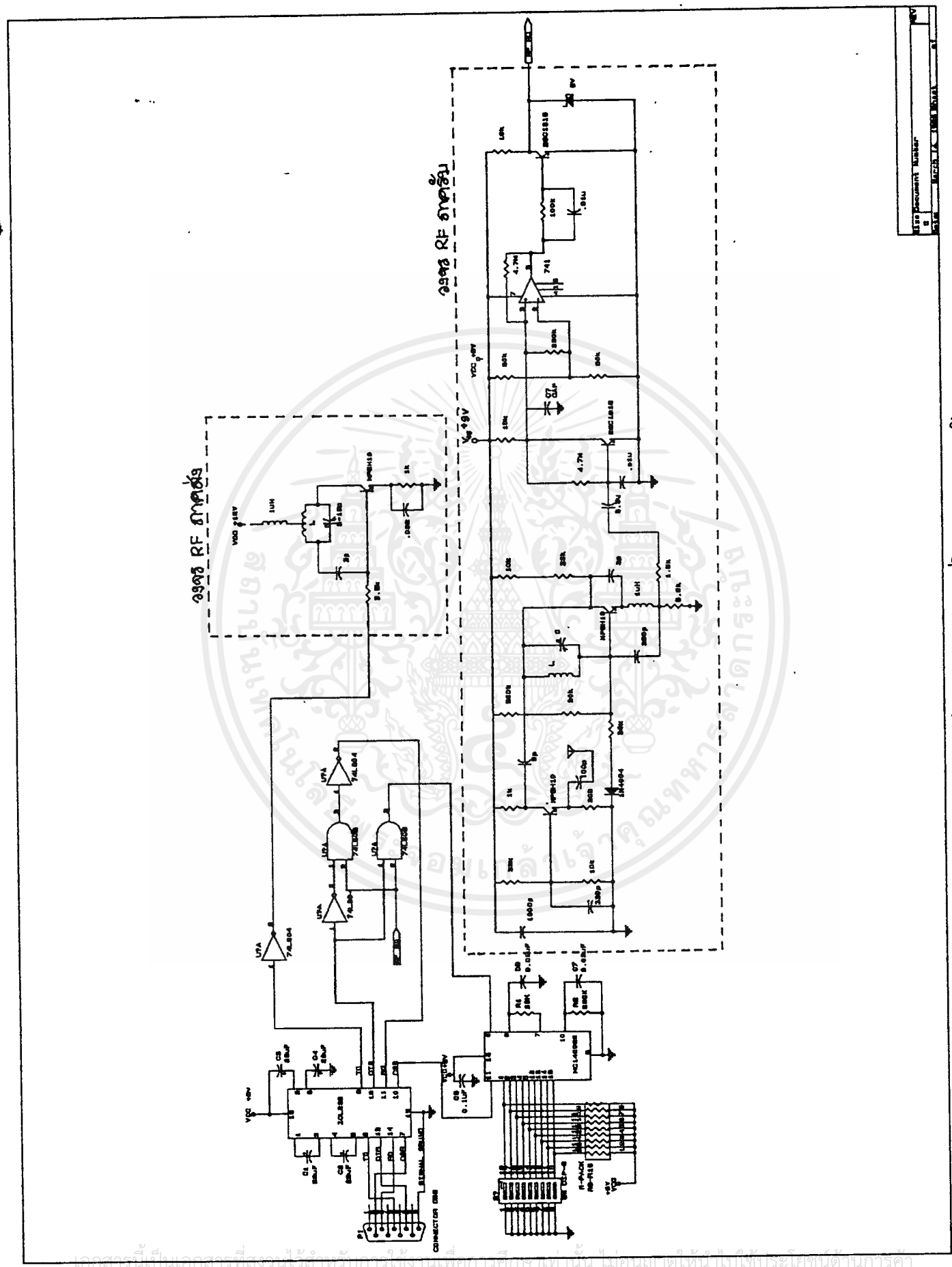
รูปที่ 4.1 แสดงวงจร demultiplex

รูปที่ 4.2 แสดงการต่อขา DSR ของ RS-232C

รูปที่ 4.3 แสดงการต่อขา Tx ของ RS-232C

จากรูปที่ 4.1 วงจร demultiplex ซึ่งปกติ data จากเครื่องรับจะป้อนเข้าที่ขา DATA IN (ขา 9 ของ MC 145028) เมื่อต้า server สแกนมาถึงเครื่อง Terminal นั้น (code ตรงกับ code ของ terminal นั้น)ต้าMC 145028ก็จะส่งสัญญานจากขา VT (ขา11) ไปยังขา DSR ของ 232C(ดังรูป4.2) จากนั้นเครื่องterminal นั้นจะถูก interrupt พร้อมกับส่งสัญญาน select เพื่อให้ data ไปเข้าที่ขา Rx ของ 232C จากนั้น serverจะบอกว่ามี dataให้เครื่องterminalนั้นหรือไม่ และจะถาม terminal นั้น ว่าต้องการส่ง data ให้กับ terminal อื่นหรือไม่ ถ้า server ไม่มี data ที่จะส่งให้ และ terminal นั้นๆ ไม่ต้องการติดต่อกับ terminal อื่นๆ ต้า server ก็จะสแกนไปถามต้าอื่นๆ ต่อไป พร้อมกับ terminal นั้นๆ ก็จะสามารถทำงาน ต่อไปได้

ในกรณีของการส่ง data จาก terminal จะเป็นการส่งตรงจาก terminalไปยังเครื่องส่งเพื่อส่งออกอากาศ ไปยัง server (ดังรูปที่ 4.3)



ภาคผนวกที่ ข วงจร Interface กับ PC

4.2 การทำงานของวงจรในส่วน server

จะเห็นว่า hardware ที่เราใช้ interface กับ Z80 cpu board เราต้องใช้ทั้งพอรต์ขนานโดยจะใช้ไอซีเบอร์ 8255 และพอรต์อนุกรม โดยจะใช้ไอซีเบอร์ 8251 ซึ่งการต่อพอรต์อนุกรมจะมีไว้สำหรับส่งโค๊ดออกจากตัว server ไปยัง terminal ที่อยู่ในระบบ และใช้ส่งและรับข้อมูลจากเครื่อง server กับเครื่อง terminal ที่กำลังติดต่ออยู่ โดยหลักการทำงานและโปรแกรมไอซีเบอร์ 8251 ได้อธิบายไว้ตอนต้นแล้ว ส่วนพอรต์ขนานมีหน้าที่ดังนี้

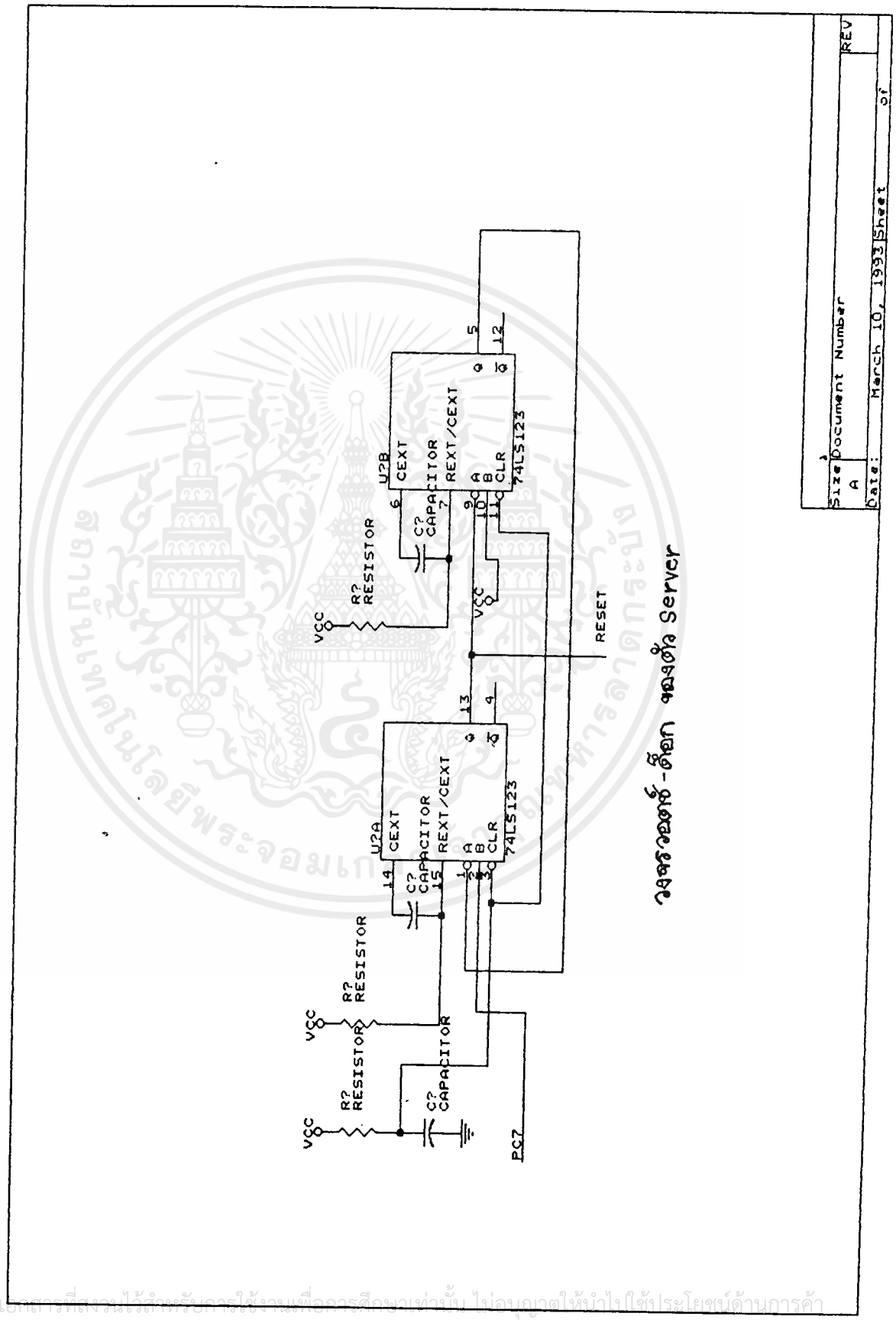
1. เป็นตัวกำหนดค่าโค๊ด ที่จะต้องส่งให้ IC encoder 14026 เพื่อส่งออกไปสู่ตัวเครื่อง terminal ในระบบ โดยเราใช้ Port A ของ IC 8255

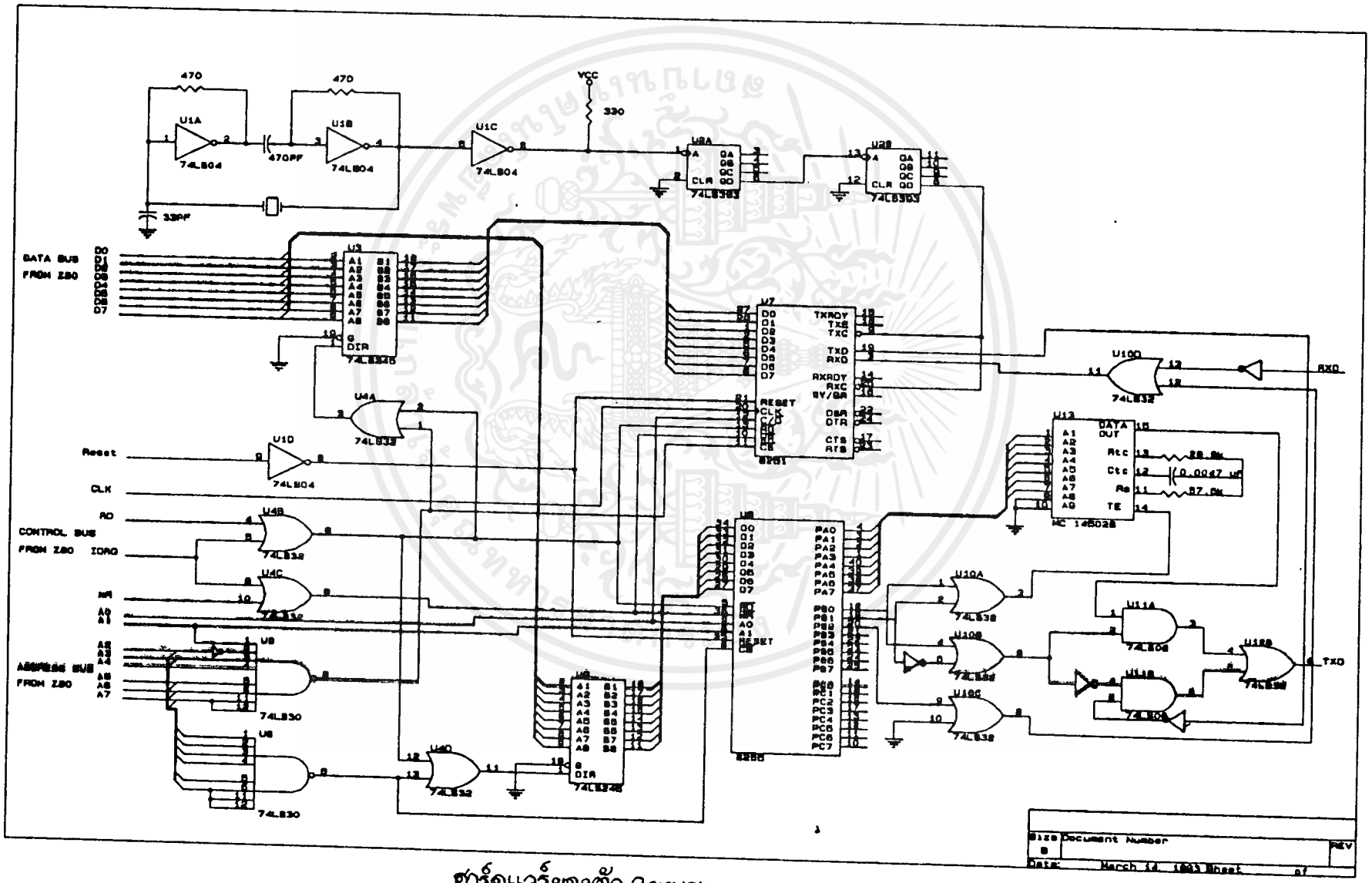
2. เป็นตัวควบคุมการ multiplex และ demultiplex โดยใช้ Port B ในวงจร โดยการส่งข้อมูลเป็น

- 04 H ไว้ใช้ในการควบคุมเกทต่างๆ ให้ส่ง code
- 06 H ไว้ใช้ในการควบคุมเกท ในการส่งข้อมูลออกไปให้เครื่องรับ
- 02 H ไว้ใช้ในการควบคุมเกท ในการรับข้อมูลจากเครื่องรับเข้ามา

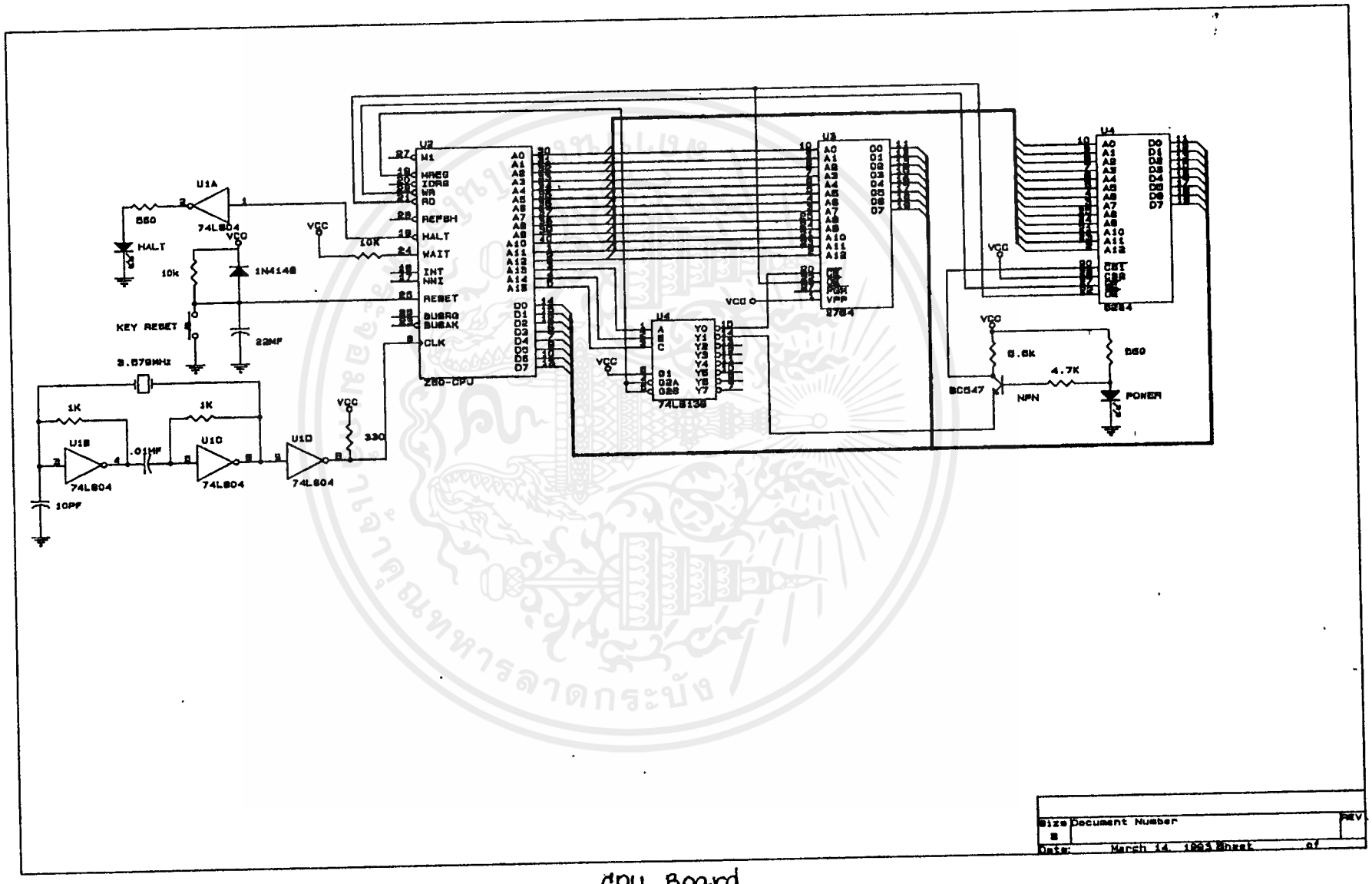
3. เป็นตัวช่วยในการทำงานของวงจร Watchdog โดยการโปรแกรมออก Port C

4.3 ส่วนซอฟแวร์ของโครงงาน



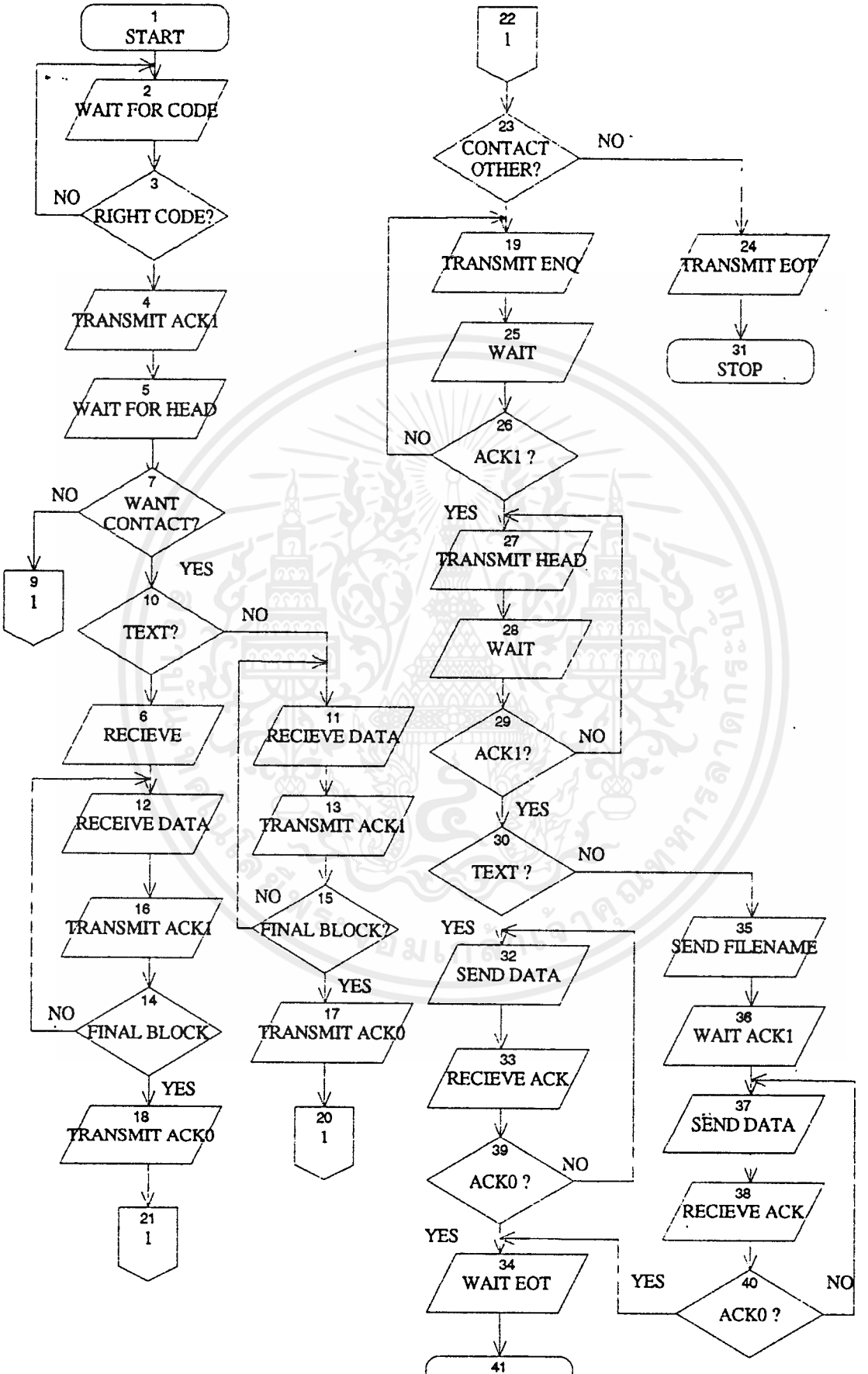


ฮาร์ดแวร์ของตัว Server

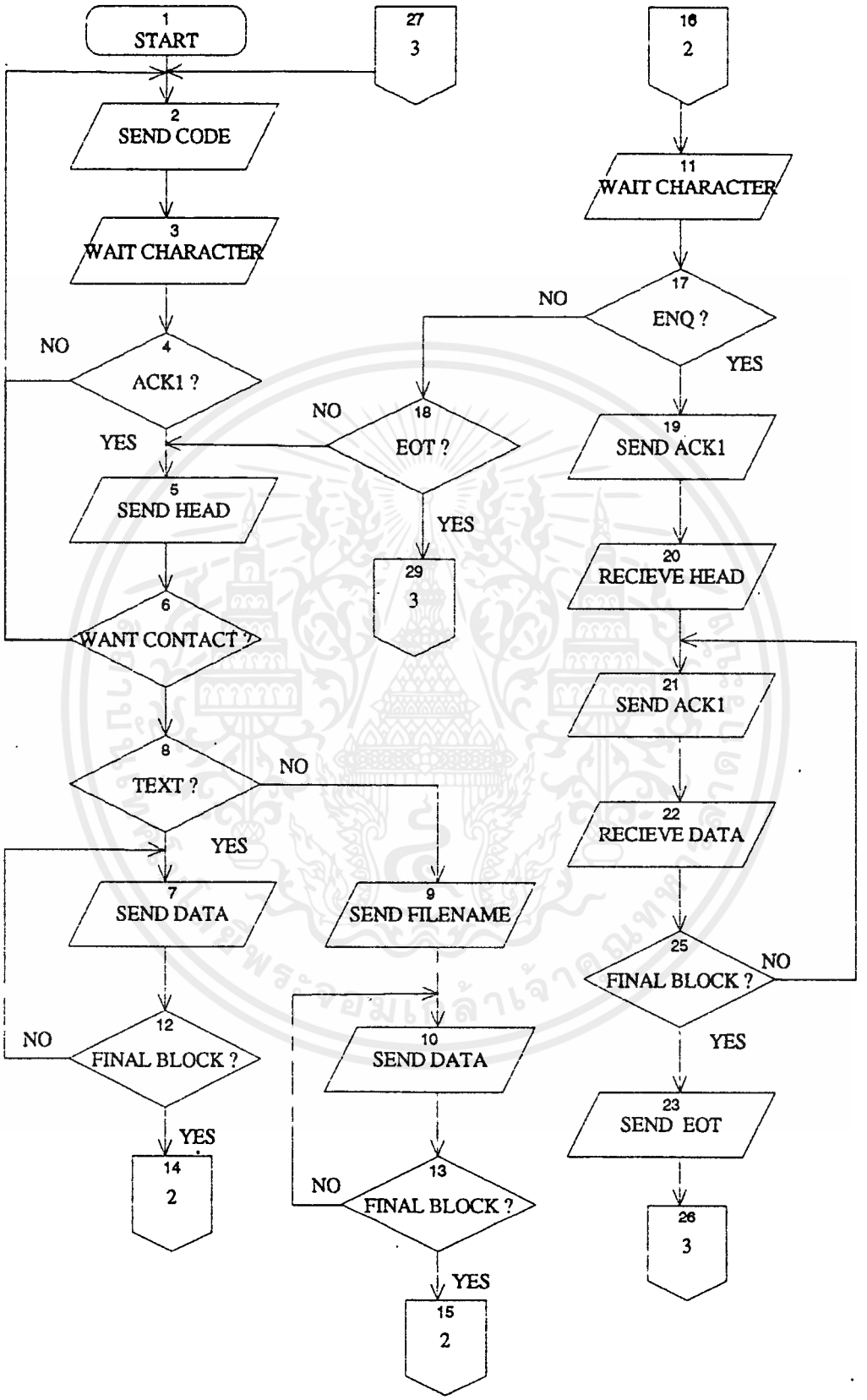

U1A
74LS04
VCC
HALT
1N4148
KEY RESET
22MF
3.579MHz
1K
U1B
U1C
U1D
.01MF
10PF
330
U2
Z80-CPU
M1
MREQ
IORQ
WR
RD
REFSH
HALT
WAIT
INT
NMI
RESET
BUSRQ
BUSAK
CLK
10K
U3
2764
U4
74LS138
6264
8.6k
4.7K
BC547
NPN
POWER
Size
Document Number
REV
Date: March 14, 1993 Sheet of

CPU Board

4.3.1 FLOWCHART ของเครื่อง TERMINAL



4.3.2 FLOWCHART ของเครื่อง SERVER



บทที่ 5

ผลการทดลอง

ในการถ่ายโอนไปรษณีย์อิเล็คทรอนิกส์ (electronic mail)

ในการทดลองรับส่งข้อความสั้นๆ ด้วยอัตรารับส่งข้อมูล 4,800 บิทต่อวินาที จำนวน 20 ครั้ง พบว่าสามารถรับส่งข้อความได้ถูกต้องสมบูรณ์ 14 ครั้ง ส่วนอีก 6 ครั้งไม่สามารถรับส่งได้ เพราะเกิดจากความไม่เน่นอนของเครื่องรับส่ง

ในการถ่ายโอนการรับส่งแฟ้มข้อมูล (file)

ทดลองรับส่งแฟ้มข้อมูลสั้นๆ ขนาดไม่เกิน 2 กิโลไบท์ ด้วยอัตรารับส่งข้อมูล 4,800 บิทต่อวินาทีจำนวน 20 ครั้ง พบว่าสามารถรับส่งแฟ้มข้อมูลได้ถูกต้องสมบูรณ์ 15 ครั้ง ส่วนอีก 5 ครั้งไม่สามารถรับส่งได้ เพราะความไม่เน่นอนของเครื่องรับส่ง

บทที่ 6

สรุปและวิจารณ์ผล

สำหรับโครงงานโครงข่ายข้อมูลไร้สายนี้ เป็นการรับส่งข้อความและแฟ้มข้อมูล(file) ผ่านคลื่นวิทยุ ปัญหาส่วนใหญ่ของโครงงานนี้อยู่ที่เครื่องรับส่งวิทยุ ซึ่งไม่มีความแน่นอน อีกทั้งระยะที่สามารถส่งได้ก็เป็นระยะสั้นๆ ประมาณ 5-10 เมตร ซึ่งถ้าสามารถปรับปรุงส่วนของเครื่องรับส่งให้ดี จะทำให้การรับส่งข้อมูลไม่มีความผิดพลาด ถูกต้องและเป็นประโยชน์มากขึ้น

นอกจากนี้ในกรณีของการส่งแฟ้มข้อมูล (file) ในโครงงานนี้เป็นเพียงการศึกษาการทำงานของการส่ง file สั้นๆไม่เกิน 2 Kbyte เพราะ buffer ที่ตัว Server มีน้อย ซึ่งถ้าขยายการส่ง file ให้ใหญ่ขึ้นก็จะมีประโยชน์มากขึ้น

ในโครงงานนี้ได้ทำส่วนของ watch dog ทั้งของ Server และ Terminal ซึ่งมีประโยชน์มากในกรณีที่ระบบเกิดการรับส่งผิดพลาดขึ้นมา watch dog ก็จะไปทำการรีเซ็ต CPU ให้เริ่มทำงานใหม่

กิตติกรรมประกาศ

สำหรับโครงงานโครงข่ายข้อมูลไร้สายนี้ ทางคณะผู้จัดทำได้รับความช่วยเหลือและให้คำแนะนำจาก อาจารย์ พลพุง ผดุงกุล และอาจารย์ชนิษฐา แซ่ตั้ง อีกทั้งได้รับความช่วยเหลือในด้านเครื่องมือที่ใช้ในการทำโครงงานจาก คุณยุทธนา ศิริแสงมงคล นักศึกษาคณะวิศวกรรมศาสตร์ ขอบคุณเพื่อนๆ น้องๆชุมนุมยิเลคทรอนิกส์ ที่ช่วยสนับสนุนและให้กำลังใจ ขอบคุณเพื่อนร่วมห้องโปรเจ็กท์ทุกคนที่ให้ความช่วยเหลือในทุกๆด้าน

ทางคณะผู้จัดทำจึงขอขอบคุณทุกๆท่านที่มีส่วนช่วยในโครงงานนี้

คณะผู้จัดทำ