เครื่องพิมพ์แบบไร้สาย

INFRARED WIRELESS PRINTER

โดย

นาย ณัฐวุฒิ ไชยประสงค์สุข

นาย เทวินทร์ วีระพลเทพ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เครื่องพิมพ์แบบไร้สาย

INFRARED WIRELESS PRINTER

โดย

นาย ณัฐวุฒิ ไชยประสงค์สุข เลขประจำตัวนักศึกษา 35104140

นาย เทวินทร์ วีระพลเทพ เลขประจำตัวนักศึกษา 35104162

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ ปีการศึกษา 2538

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องพิมพ์แบบไร้สาย

ผู้จัดทำ

1. นาย ณัฐวุฒิ ไชยประสงค์สุข

2. นาย เทวินทร์ วีระพลเทพ

รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา

(.....)

ชื่อโครงงานภาษาไทย	เครื่องพิมพ์แบบไร้สาย	
ชื่อโครงงานภาษาอังกฤษ	INFRARED WIRELESS PRINTER	
นาย ณัฐวุฒิ ไชยประสงค์สุข	เลขประจำตัวนักศึกษา	35104140
นาย เทวินทร์ วีระพลเทพ	เลขประจำตัวนักศึกษา	35104162

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(.....)

อาจารย์ที่ปรึกษา

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

หน้า

เครื่องพิมพ์แบบไร้สาย

ณัฐวุฒิ ไชยประสงค์สุข

เทวินทร์ วีระพลเทพ

รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2538

บทคัดย่อ

เครื่องพิมพ์แบบไร้สายในโครงงานนี้เป็นการสื่อสารข้อมูลอนุกรมแบบไร้สายโดยผ่านทางแสงอินฟราเรดสามารถส่งได้ไกลกว่าแบบขนานซึ่งใช้กันอยู่ทั่วไปในการติดต่อระหว่างคอมพิวเตอร์กับเครื่องพิมพ์ ในโครงงานนี้ข้อมูลจะส่งออกจากคอมพิวเตอร์เป็นแบบขนาน จากนั้นเครื่องส่งจะแปลงเป็นแบบอนุกรมส่งออกทางอินฟราเรด เครื่องรับจะรับข้อมูลที่ส่งด้วยแสงอินฟราเรดซึ่งเป็นข้อมูลแบบอนุกรมมาแปลงเป็นแบบขนานแล้วจึงส่งให้เครื่องพิมพ์ต่อไป

INFRARED WIRELESS PRINTER

Nuttawut Chaiprasongsook

Thaywin Weerapholthep

Assoc.Prof.Dr. Manas Sangworasilp

Advisor

1995

ABSTRACT

Infrared wireless printer in this project is a serial wireless data communication. By using infrared ray can be send long distance than communicate between computer and printer. In this project is send out data from computer to parallel after that transfrom to serial to infrared. At printer it will transfrom infrared ray from serial to parallel and send to printer again.

บทที่ 1
บทนำ

เนื่องจากในปัจจุบันการใช้งานทางคอมพิวเตอร์ได้นิยมใช้การอย่างแพร่หลาย การติดต่อสื่อสารระหว่างคอมพิวเตอร์กับคอมพิวเตอร์หรืออุปกรณ์อำนวยความสะดวกต่าง ๆ จึงเป็นสิ่งจำเป็นต่อการใช้งานทางด้านคอมพิวเตอร์ การติดต่อสื่อสารจึงได้มีการพัฒนาไปอย่างรวดเร็วและมีรูปแบบในการติดต่อสื่อสารต่าง ๆ หลายรูปแบบมากมาย เพื่อทำให้การใช้งานเฉพาะอย่างมีความสะดวกรวดเร็วกว้างขวางและข้อมูลที่ส่งมีความถูกต้องแม่นยำสูง

การสื่อสารในปัจจุบันแบ่งออกอย่างกว้าง ๆ ได้เป็น 2 แบบคือ การติดต่อแบบขนานและการติดต่อแบบอนุกรม ซึ่งในโครงงานนี้เป็นการสื่อสารข้อมูลแบบอนุกรมโดยใช้แสงอินฟราเรดเป็นตัวส่งข้อมูลระหว่างคอมพิวเตอร์กับเครื่องพิมพ์ เป็นการสื่อสารอีกแบบหนึ่งที่ประยุกต์โดยการนำเอาการสื่อสารแบบอนุกรมและประโยชน์ของแสงอินฟราเรดมารวมกันเป็นการสื่อสารอนุกรมแบบไร้สาย โดยเครื่องพิมพ์จะสามารถรับข้อมูลจากคอมพิวเตอร์ผ่านทางแสงอินฟราเรด ซึ่งใช้ไมโครคอนโทรลเลอร์เป็นตัวจัดการในการรับและการส่งข้อมูลทั้งหมด

ทางคณะผู้จัดทำโครงงานนี้จึงหวังว่าจะเป็นประโยชน์ในการศึกษาและประยุกต์ใช้งานการติดต่อสื่อสารข้อมูล

บทที่ 2 การติดต่อเครื่องพิมพ์แบบขนาน

2.1 การเชื่อมต่อเครื่องพิมพ์แบบขนาน

ปัจจุบันในระบบการติดต่อระหว่างเครื่องพิมพ์กับคอมพิวเตอร์หรือโครงข่ายเครื่องพิมพ์จะเป็นการติดต่อแบบขนาน การติดต่อแบบขนานได้มีการพัฒนาขึ้นมาเป็นเวลาหลายปีแล้วแต่เริ่มจริงจังเมื่อเครื่องพีซี (PC) เริ่มเป็นที่นิยมใช้ บริษัทผู้ผลิตเครื่องพิมพ์ที่ชื่อ "เซ็นทรอนิกส์" (CENTRONICS)ได้เสนอเครื่องพิมพ์ที่สามารถใช้ได้กับส่วนติดต่อแบบขนานได้ทุกชนิดทำให้ผู้ผลิตคอมพิวเตอร์และอุปกรณ์เชื่อมต่อต่างๆเริ่มผลิต พอร์ต(PORT) ที่สามารถใช้ได้กับเครื่องพิมพ์ของเซ็นทรอนิกส์ คำว่า " ส่วนคิดต่อแบบขนานเซ็นทรอนิกส์ " จึงเริ่มเป็นที่รู้จักและการเชื่อมต่อแบบขนานในปัจจุบันก็จะต้องสร้างขึ้นเพื่อให้เข้ากันได้กับเซ็นทรอนิกส์

ในส่วนต่อไปนี้จะกล่าวถึงส่วนประกอบของการติดต่อแบบขนานแบบเซ็นทรอนิกส์ ซึ่งเป็นที่นิยมใช้หันที่วไปในคอมพิวเตอร์และอุปกรณ์การพิมพ์ต่างๆ

2.2 ส่วนติดต่อแบบขนาน

ส่วนติดต่อแบบขนานของ เซ็นทรอนิกส์ มี 36 ขา ความยาวของสายจะยาวประมาณ 6 ถึง 9 ฟุต แต่ความยาวนี้ไม่กำหนดตายตัว สายที่ใช้เชื่อมต่อกับส่วนติดต่อแบบเซ็นทรอนิกส์ ก็จะเป็นประเภทเดียวกับ RS-232 คือจะประกอบด้วย สายกราวด์(GROUND) ,ข้อมูล(DATA), ควบคุม(CONTROL) และเวลาแต่จะมีส่วนที่แตกต่างกันบ้างก็คือแบบ เซ็นทรอนิกส์ ข้อมูลจะถูกส่งไปทางเดียวโดยส่งออกจากคอมพิวเตอร์ไปสู่อุปกรณ์ภายนอก แต่การติดต่อแบบอนุกรมจะส่งข้อมูลได้สองทิศทาง

2.3 สายกราวด์และข้อมูล

ในการส่งแบบอนุกรมข้อมูลจะถูกส่งไปครั้งละ 1 บิต เครื่องรับจะต้องรู้ความยาวตัวอักษรบิตเริ่ม/บิตหยุด ความเร็วในการส่งข้อมูล เพื่อจะทำให้ถอดระหัสได้ถูกต้อง ในการต่อแบบขนานจะตรงกันข้ามกับการส่งข้อมูลแบบอนุกรม โดยในการส่งเครื่องคอมพิวเตอร์จะส่งบิตทั้งหมดของตัวอักษรในเวลาเดียวกันแทนที่จะส่งไปทีละครั้งพิมพ์ไปให้เครื่องพิมพ์ไปให้เครื่องพิมพ์ไปให้เครื่องพิมพ์ใส การส่งทั้ง 2 แบบถ้าเริ่มที่เวลาเดียวกัน บิตข้อมูลจะส่งในรูปแบบเดียวกันโดย RS-232 จะส่งทีละบิตจนครบ 8 บิต ส่วนในการส่งแบบขนานจะต้องเซ็ตขา 2-9 สำหรับส่งข้อมูลโดยข้อมูลแต่ละเส้นจะต้องมีระดับสายกราวด์ที่สัมพันธ์กัน แต่ RS-232 จะเซ็ตขาที่ 7 ขาเดียว ส่วนแบบขนานจะมีขา 20-27 เป็นขากราวด์ที่อ้างอิงถึงข้อมูลในแต่ละเส้น จากตารางที่ 2.1 เป็นสายคู่กันของกราวด์และข้อมูลในแต่ละบิต แต่ใน RS-232 สายกราวด์จะไม่มีการกำหนดเหมือนแบบขนาน โดยสายกราวด์จะมีระดับเป็น 0 โวลต์

เนื่องจากสายข้อมูลมี 9 เส้น บิตของข้อมูลทั้งหมดของตัวอักษรจะถูกส่งจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ในเวลาเดียวกัน ซึ่งในการส่งแบบนี้จะรวดเร็วมาก ดังนั้นหน่วยความเร็วจะไม่แสดงในรูปจำนวนบิตต่อนาที แต่จะมีหน่วยเป็นจำนวนอักษรต่อวินาที (CPS) โดยมีความเร็วปกติที่ 500 CPS ความเร็วในการส่งแบบนี้ไม่ใช่ว่าจะรวดเร็วกว่าการส่งแบบอนุกรม ความเร็วของเครื่องพิมพ์จะเป็นตัวชี้ที่แท้จริงว่าจะสามารถพิมพ์ข้อมูลออกมามากแค่ไหน

สายกราวด์และสายอื่นๆจะมีที่ขา 14 ,16 และ 17 โดยที่ขา 14 หรือขา16 จะเป็นสายกราวด์อ้างอิงของสายควบคุม ส่วนขา 17 จะเป็นสายกราวด์ของเครื่อง

ข้อมูลบิตที่	คู่สายของข้อมูล
1	2,20
2	3,21
3	4,22
4	5,23
5	6,24
6	7,25
7	8,26
8	9,27

ตารางที่ 2.1 แสดงสายของข้อมูลและกราวด์ในแต่ละบิต

2.4 สายควบคุม

ส่วนอินเทอร์เฟสจะมีสายควบคุมสำหรับเครื่องพิมพ์เพื่อคอยแจ้งให้คอมพิวเตอร์ทราบว่าได้รับตัวอักษรจากคอมพิวเตอร์แล้ว ซึ่งขา 10 จะเป็นขา acknowledge และเป็นขาออกของสัญญาณจากเครื่องพิมพ์ไปยังคอมพิวเตอร์ ครั้งแรกจะมีรหัสที่ส่งยังเครื่องพิมพ์และคอมพิวเตอร์จะได้รับสัญญาณพัลส์ (pulse)ก่อนที่รหัสใหม่จะถูกส่งเข้าไป ขา 28 (return) จะเป็นขาที่ติดต่อกับขาสำหรับการรับรู้ (ackwonledge)

หน้าที่ของสายควบคุมหน้าที่อื่นมีดังนี้ เช่นในกรณีที่กระดาษหมดหรือในขณะที่เครื่องพิมพ์อยู่ในกรณีที่ยังไม่พร้อมจะทำงาน เพราะความเร็วในการติดต่อแบบขนาน ขาควบคุมจะมีความจำเป็นที่จะต้องควบคุมการส่งข้อมูลระหว่างอุปกรณ์ 2 ตัว สัญญาณควบคุมจะรักษาสภาวะที่คอมพิวเตอร์รับรู้เมื่อข้อมูลถูกส่งออกมา

2.5 สัญญาณ BUSY

ถ้าเครื่องพิมพ์ไม่สามารถรับข้อมูลได้อีกแล้ว สัญญาณนี้ก็จะถูกส่งไปยังคอมพิวเตอร์ โดยสายควบคุมในการติดต่อกับอุปกรณ์ต่างๆนั้นจะเหมือนกับการติดต่อแบบอนุกรม RS-232 จะถูกควบคุมทางฮาร์ดแวร์หรือซอฟแวร์ก็ได้ โดยถ้าเครื่องพิมพ์รับข้อมูลเต็มแล้วจะต้องควบคุม โดยซอฟแวร์โดยใช้ xon/xoff หรือ etx/ack เพราะว่าตัวอักษรจะถูกส่งจากเครื่องพิมพ์กลับไปยังคอมพิวเตอร์ เพราะว่าการติดต่อแบบขนานจะเป็นทิศทางเดียวคือให้เฉพาะสัญญาณควบคุมทางฮาร์ดแวร์ โดยขา 11 จะใช้สำหรับทำหน้าที่นี้ ขา 11 คือขาสัญญาณการทำงาน ถ้าไม่ทำงานจะเป็น " 0 " ข้อมูลจะถูกส่งจากคอมพิวเตอร์ ถ้ามีสัญญาณจากขา ack มาโดยสภาวะ busy นี้ จะทำให้เครื่องพิมพ์เปิด ขา 11 จะเป็นตัวบอกคอมพิวเตอร์ให้หยุดส่งข้อมูล ถ้าเครื่องพิมพ์ปิดจะทำให้มีการผิดพลาด หรือบัฟเฟอร์ของเครื่องพิมพ์เต็มและเครื่องพิมพ์จำเป็นจะต้องส่งข้อมูลออกไป เมื่อเครื่องพิมพ์อยู่ในโหมด online หรือในสภาวะที่ไม่มีข้อผิดพลาดแล้วขาสัญญาณ (ขา11) จะเป็น " 0 " อีกครั้ง สัญญาณนี้จะเป็นสัญญาณนำร่องการส่งข้อมูล ขา 29 ก็คือขากราวด์ของสัญญาณในการทำงาน

สายสัญญาณ busy ก็จะเหมือนสายควบคุมทางฮาร์ดแวร์ของ RS-232 ความแตกต่างก็คือสภาวะของสายสัญญาณในการตอบรับขา 11,19 หรือ 20 ใน RS-232 จะเป็นตัวชี้ว่าตอนนี้คอมพิวเตอร์สามารถส่งข้อมูลออกไปได้หรือไม่ ขาสัญญาณการทำงานแบบขนานถ้าปิดอยู่จะเป็นการบอกว่ายังสามารถส่งข้อมูลออกไปได้

ขา 32 คือ สัญญาณบอกว่ามีการผิดพลาดเกิดขึ้น เมื่อขานี้เปิด (on) แปลว่าไม่มีการผิดพลาดเกิดขึ้นที่เครื่องพิมพ์และขานี้ปิด (off) เมื่อเครื่องพิมพ์กระดาษหมด , เครื่องพิมพ์ในสภาวะที่ไม่พร้อม สัญญาณผิดพลาดและสัญญาณในการทำงานจะคล้ายกันแต่มีข้อแตกต่างกันบ้าง ขาสัญญาณในการทำงานจะเปิดเพื่อที่จะหยุดส่งข้อมูลจากคอมพิวเตอร์ สถานะต่างๆสามารถทำให้ขาสัญญาณนี้เปิดรวมทั้งกรณีกระดาษหมดหรือเครื่องพิมพ์ปิดอยู่และอื่นๆส่วนใหญ่ขาสัญญาณความผิดพลาดจะปิด นอกจากก่อนหน้านั้นจะมีสถานะนี้มาก่อน สายควบคุม buffer เต็มสายสัญญาณการทำงานจะเปิดเพื่อส่งข้อมูลจากคอมพิวเตอร์ ฉะนั้นขาสัญญาณในการทำงานจะถูกส่งออกไประหว่างที่ขาสัญญาณความผิดพลาดเป็นตัวชี้ว่าเกิดการผิดพลาดขึ้น

2.6 โหมดในการเปิดเลือก

การติดต่อแบบขนานของเครื่องพิมพ์ จะมีสัญญาณที่ใช้บอกคอมพิวเตอร์ว่าขณะนี้เปิดเครื่องและอยู่ในสถานะที่พร้อมที่จะทำงานและ ถ้าขา 13 เปิดคอมพิวเตอร์จะรู้ว่าขณะนี้พร้อมที่จะทำงานแล้ว โดยขาสัญญาณนี้ก็คือขา Select โดยรหัสการเลือกจะส่งให้เครื่องพิมพ์ ถ้าเครื่องพิมพ์

พร้อมขานี้จะเปิด แต่ถ้าเครื่องพิมพ์เปิดเครื่องอยู่ขานี้จะปิดและสัญญาณการทำงานจะเปิด คอมพิวเตอร์จะรับรู้โดยการส่งสัญญาณควบคุมและหยุดการส่งข้อมูล

กรณีที่กระดาษหมด เมื่อเครื่องพิมพ์กระดาษหมดเครื่องพิมพ์จะแจ้งให้คอมพิวเตอร์ให้ทราบ ซึ่งจะรู้ได้จากขา 12 (pe) สัญญาณนี้จะสร้างจากเครื่องพิมพ์ ถ้าขานี้เปิดเครื่องพิมพ์จะหยุดส่งข้อมูลเพื่อป้องกันไม่ให้ข้อมูลสูญหาย เครื่องพิมพ์จะทำให้ขานี้เปิดพร้อมขา busy โดยขา Select จะไม่ลดลงเป็น 0 ดังนั้นเมื่อกระดาษหมดขาสัญญาณที่จะเปิดมีดังนี้ ack , busy , pe , sel และ fault ดังนั้นถ้าเติมกระดาษไปแล้วเครื่องพิมพ์จะกลับมาอยู่ในสภาวะเปิดตามเดิม แต่ขา ack , selจะเปิดเหมือนเดิม ขา ack จะเปิดและปิดในการรับข้อมูลในแต่ละครั้งซึ่งจะใช้ขานี้ต่อกับขาของสัญญาณการผิดพลาดในหารติดต่อกับคอมพิวเตอร์ เพื่อเป็นการบอกเมื่อเกิดปัญหาที่เครื่องพิมพ์

2.7 การรีเซ็ตเครื่องพิมพ์

ถ้าคอมพิวเตอร์ต้องการรีเซ็ตเครื่องพิมพ์และเคลียร์ Buffer สัญญาณที่ใช้ในการติดต่อสำหน้าที่นี้คือ ขา31 โดยขา 31 จะเปิดอยู่นอกจากเกิดการรีเซ็ต ถ้าขานี้เป็นศูนย์จะทำให้เครื่องพิมพ์ปิดและกลับมาเปิดใหม่ ขาสัญญาณนี้คอมพิวเตอร์จะส่งไปยังเครื่องพิมพ์ โดยขากราวด์สัญญาณนี้จะอยู่ที่ขา 30

2.8 การจำลองเครื่องพิมพ์

การที่จะต้องจำลองเครื่องพิมพ์นั้นเป็นเพราะมีการใช้เครื่องพิมพ์แบบต่างๆมากมายการใช้รหัสควบคุมและขนาดของเครื่องพิมพ์ต่างๆโดยการติดต่อแบบขนานนี้ แสดงในตาราง 1.2

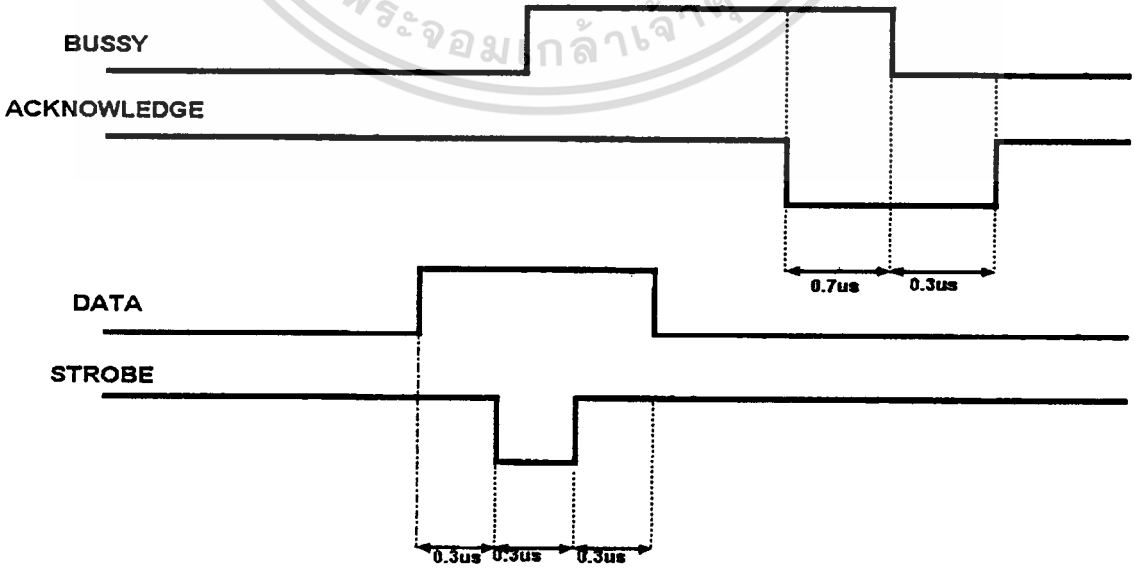
Pin	Signal	Pin	Signal
1	DATASTROBE	19	Twisted Pair Ground
2	Data Bit 1	20	Twisted Pair Ground
3	Data Bit 2	21	Twisted Pair Ground
4	Data Bit 3	22	Twisted Pair Ground
5	Data Bit 4	23	Twisted Pair Ground
6	Data Bit 5	24	Twisted Pair Ground
7	Data Bit 6	25	Twisted Pair Ground
8	Data Bit 7	26	Twisted Pair Ground
9	Data Bit 8	27	Twisted Pair Ground

10	ACKNOWLADGE	28	Twisted Pair Ground
11	BUSY	29	Twisted Pair Ground
12	PAPER OUT	30	INPUT PRIME RETURN
13	SELECT	31	INPUT PRIME
14	GROUND	32	FAULT
15	Not Use	33	GROUND
16	GROUND	34	Not Use
17	CHASSIS GROUND	35	Not Use
18	+ 5 V	36	Not Use

ตารางที่2.2 แสดงการติดต่อแบบขาน

2.9 ขาสัญญาณเวลา

RS-232 จะมีสัญญาณเวลาในการส่งและการรับสำหรับการส่งแบบนี้พร้อมกัน ส่วนการส่งแบบขนานจะมีสัญญาณการส่งอย่างเดียว ขา 1 หรือขารับรู้การส่งข้อมูลจะต้องได้จังหวะพอดีกับการรับรู้โดยสัญญาณนาฬิกาส่งให้กับเครื่องพิมพ์ และการอ่านข้อมูลของบิตต่างๆ การรับรู้จะอยู่ในช่วงเวลาสั้นๆประมาณ 1 ไมโครเซ็ค โดยเมื่อตัวอักษรถูกอ่านเข้ามา ขารับรู้จะเปิดในระยะเวลาสั้นๆจากนั้นจะเป็น " 0 " อีกครั้งเมื่อได้รับอักษรตัวต่อไปเข้ามา จากรูปที่ 2.1 เป็นตารางเวลาของการติดต่อแบบขนาน โดยดูจากขาสัญญาณ ack และ busy



รูปที่ 2.1 แสดงตารางเวลาของการติดต่อแบบขนาน

2.10 บัฟเฟอร์ (BUFFER)

เครื่องพิมพ์จะทำงานช้ากว่าคอมพิวเตอร์มาก ดังนั้นถ้าต้องการพิมพ์งานจำนวนมากผู้ใช้จำเป็นต้องเก็บข้อมูลเร็วๆ และส่งออกไปทำงาน บัฟเฟอร์จะทำหน้าที่เก็บข้อมูลที่ส่งออกจากคอมพิวเตอร์เพื่อให้คอมพิวเตอร์ทำงานได้ต่อไปโดยไม่ต้องรอส่งข้อมูลออก

บัฟเฟอร์จะมีขนาดหน่วยความจำขนาดต่างๆและแบบต่างๆบัฟเฟอร์สามารถรับข้อมูลแบบขนานหรือแบบอนุกรมได้ เช่น กรณีที่คอมพิวเตอร์ติดต่อกันแบบอนุกรม ส่วนเครื่องพิมพ์ติดต่อแบบขนาน บัฟเฟอร์จะถูกนำมาใช้เพื่อแก้ไขปัญหาในการติดต่อระหว่างอุปกรณ์ทั้งสองนี้

คอมพิวเตอร์สามรถส่งข้อมูลแบบอนุกรมไปยังบัฟเฟอร์และส่งสัญญาณควบคุมไปด้วย บัฟเฟอร์จะส่งข้อมูลออกไปยังเครื่องพิมพ์เป็นแบบขนาน ซึ่งคอมพิวเตอร์ที่ต่ออยู่กับบัฟเฟอร์ก็เหมือนต่ออยู่กับเครื่องพิมพ์ส่วนสัญญาณควบคุม และเกิดการผิดพลาดบัฟเฟอร์จะเป็นตัวจัดการในทางกลับกันคอมพิวเตอร์ติดต่อกันแบบขนานส่วนเครื่องพิมพ์ติดต่อกันแบบอนุกรมบัฟเฟอร์ก็สามารถทำงามให้คอมพิวเตอร์กับเครื่องพิมพ์ติดต่อกันได้ ซึ่งทำให้การติดต่อระหว่างอุปกรณ์ความเหมาะสม

2.11 สวิตช์ ABC

สวิตช์ ABC เป็นอุปกรณ์ที่ใช้สำหรับเชื่อมอุปกรณ์หลายๆตัวโดยใช้สวิตช์ เช่น คอมพิวเตอร์หนึ่งเครื่องเชื่อมกับสวิตช์ไปยังเครื่องพิมพ์ 2 ตัว หรือเครื่องพิมพ์ 1 เครื่องไปยังคอมพิวเตอร์หลายๆเครื่องโดยอุปกรณ์ต่างๆจะถูกต่อกับสวิตช์ เช่น ในกรณีที่อุปกรณ์ของพอร์ต A,B ต่อกับพอร์ต C เมื่อสวิตช์เปิดของพอร์ต A จะได้ว่า พอร์ต A และพอร์ต C ต่อถึงกัน ถ้าพอร์ต B เปิดจะได้ว่า พอร์ต B และพอร์ต C ติดต่อกันอยู่ ซึ่งสวิตช์แบบนี้สามารถต่อกับอุปกรณ์อื่นๆได้มากกว่า 2 ตัวเข้ากับอุปกรณ์ภายใน 1 ตัว

ซึ่งชี้ให้เห็นว่าสวิตช์นี้เป็นอุปกรณ์ที่เป็นเส้นทางไว้ส่วนผ่าน โดยห้ามมีสายสัญญาณมาตัดผ่าน โดยการต่อของ A-C และ B-C ถือว่าเป็นเส้นทางตรงที่ผ่านได้ ถ้าสวิตช์ ABC เป็นแบบขนานขาทั้ง 36 ขาจะแยกว่าขาใดไปที่พอร์ตใด

2.12 การต่อแบบขนานแบบ IBM

การส่งข้อมูลแบบขนานเป็นมาตรฐาน โดย IBM ได้เป็นผู้ออกแบบไว้ซึ่งเป็นการเชื่อมต่อโดย DB25 มี 25 ขา สำหรับการติดต่อแบบขนาน หมายความว่าขา 36 ขานั้นไม่ได้ใช้ทั้งหมด เครื่องพิมพ์ส่วนใหญ่ที่ติดต่อแบบขนานจะใช้แบบแอมฟีนอล(Amphenol) เพราะDB25 ไม่สามารถใช้ แทน แบบแอมฟีนอลได้

การต่อขาของ DB25 จะเป็นตัวผู้และพอร์ตของคอมพิวเตอร์จะเป็นตัวเมีย ดังนั้นในการต่อออกจากคอมพิวเตอร์จะต้องดูด้วยจะต่อแบบใด ฉะนั้นตัวเมียจะเป็นพอร์ตแบบขนานส่วนตัวผู้จะ

เป็นพอร์ตอนุกรมโดยขาของ DB25 แสดงไว้ในตารางที่ 2.3 ส่วนอีกตารางเป็นตัวปรับให้การต่อของ DB25 กับ Amphenol เหมือนกัน

DB25 Pin	Function
1	Stobe
2	Data bit 0
3	Data bit 1
4	Data bit 2
5	Data bit 3
6	Data bit 4
7	Data bit 5
8	Data bit 6
9	Data bit 7
10	Acknowledge
11	Busy
12	Paper end (out of paper)
13	Select
14	Auto feed
15	Error
16	Initialize printer (reset)
17	Select input
18-25	Ground

ตารางที่ 2.3 แสดงขาของพอร์ตแบบขนานของ IBM

2.13 การติดต่อกับอุปกรณ์ต่างๆ

เครื่องคอมพิวเตอร์มีหลายพอร์ตได้เพื่อประโยชน์ในการใช้งานติดต่อกับอุปกรณ์ต่างๆเมื่อจะเลือกใช้อุปกรณ์ทางพอร์ตไหนก็สามารถใช้ได้

2.14 สัญญาณควบคุม

การเลือกแบบนี้จะไม่สัมพันธ์กันเมื่อใช้การติดต่อแบบขนาน โดยสายสัญญาณจะใช้ทางฮาร์ดแวร์มาตรฐานนี้จะใช้กับแบบอนุกรม ,พอร์ต

2.15 ความเร็ว

เมื่อใช้การติดต่อแบบขนานจะเกิดการไม่สัมพันธ์กัน แต่ถ้าเป็นการติดต่อแบบอนุกรม ความเร็วจะต้องได้จังหวะกันพอดี เพื่อไม่ให้มีการผิดพลาดเกิดขึ้นในการส่งสัญญาณ สัญญาณการรับรู้ในการติดต่อแบบขนานจะขึ้นอยู่กับการส่งข้อมูล โดยการส่งสายสัญญาณแบบขนานจะไม่นำความเร็วมาคิด

DB25	Amphenon
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	32
16	31
17	36
18	33
19	19
20	21
22	25
23	27
24	29
25	30

ตารางที่ 2.4 แสดงการปรับต่อขาของDB25 และ Amphenol

บทที่ 3

การสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมเป็นการรับหรือส่งข้อมูลในลักษณะบิตหรือกลุ่มของบิต ครา ละหนึ่งบิตเป็นลำดับเรื่อยไปจนสิ้นสุด การสื่อสารแบบนี้จะมีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมาก เนื่องจากข้อมูลมีการโอนย้ายมาพร้อมกันจึงมีความจำเป็นต้องใช้จำนวนเส้นสัญญาณเพียงสองหรือสามเส้นเท่านั้น ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะสมในการสื่อสารกับอุปกรณ์ภายนอกเป็นระยะทางไกลๆเพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก

รูปที่ 3.1 ข้อมูลสื่อสารแบบอนุกรม ข้อมูลหนึ่งไบต์จะถูกส่งออกคราวละบิต

3.1 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม

เนื่องจากการสื่อสารแบบอนุกรมเป็นการรับ/ส่งข้อมูล ในลักษณะกลุ่มของบิตข้อมูล (Bit Stream) ดังนั้นจึงต้องให้ความสนใจในการพิจารณาถึงเรื่องของอัตราความเร็วในการรับ/หนึ่งวินาทีเรียกว่า อัตราบอด (Baud Rate) ตามค่ามาตรฐานเหล่านี้ ได้แก่ 110 ,150 ,300 ,1200 ,2400, 4800 และ9600

3.2 รูปแบบของข้อมูลอนุกรม

วิธีการที่จะทำให้ข้อมูลสื่อสารอนุกรมมีความถูกต้องมากยิ่งขึ้น จะใช้การเพิ่มเติมบิตข้อมูลบางอย่างร่วมไปกับข้อมูลจริงได้แก่

3.2.1. บิตเริ่มต้น (Start Bit)

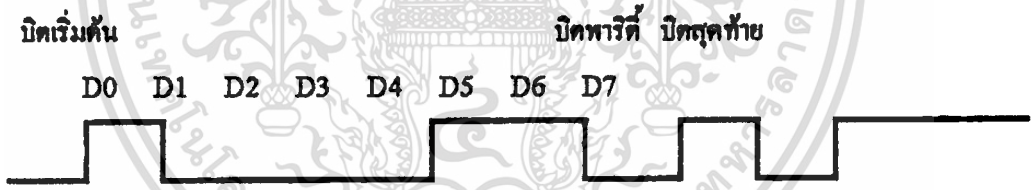
บิตนี้มีหน้าที่สำหรับการบ่งบอกให้วงจรฮาร์ดแวร์รหัสด้านรับทราบถึง ตำแหน่งจุดเริ่มต้นของบิตข้อมูลกลุ่มใหม่ เพื่อที่จะทำการปรับจังหวะของสัญญาณการรับข้อมูลให้ตรงกัน ดังนั้นบิตเริ่มต้นนี้จึงถูกเพิ่มเข้าไปก่อนมีการส่งข้อมูลจริง ตามปกติแล้วค่าของบิตเริ่มต้นมักจะเป็นระดับลอจิก (Logic) ที่ตรงข้ามกับระดับลอจิกของสภาวะของสายสื่อสาร ขณะเมื่อไม่มีการส่งข้อมูล (Idle State)

3.2.2 บิตแสดงภาวะความเป็นเลขคู่หรือเลขคี่ (Parity Bit)

บิตนี้มีหน้าที่เพื่อการตรวจสอบความถูกต้องของข้อมูล โดยทั่วไปมักเรียกว่า บิตพาริตี้ และจะนำไปแทรกต่อท้ายบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของบิตข้อมูลที่เป็น 1 ซึ่งจะเป็นได้สองลักษณะ คือ พาริตี้คู่ (Event Parity) และ พาริตี้คี่ (Odd Parity) ตัวอย่าง เช่น ระบบที่ติดต่อกัน โดยระบุว่าจะใช้พาริตี้คู่ ทางด้านส่งจะนำค่าข้อมูลที่จะส่งมาพิจารณาหากจำนวนของบิตที่มีค่าเป็น 1 เป็นเลขจำนวนคู่อยู่แล้วค่าของบิตพาริตี้จะมีค่าเป็น 0 แต่หากว่าจำนวนของบิตที่มีค่าเป็น 1 เป็นเลขจำนวนคี่ ค่าของบิตพาริตี้จะมีค่าเป็น 1 การพิจารณาถ้ามีค่าเป็นเลขจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามานี้ถูกต้อง แต่หากไม่เป็นเลขจำนวนคู่แสดงว่าเกิดความผิดพลาดของข้อมูลขึ้น

3.2.3. บิตสุดท้าย (Stop Bit)

สมบูรณ์ คือ บิตเริ่มต้น บิตพาริตี้ และ บิตสุดท้าย รวมทั้งสิ้น 12 บิต บิตสุดท้ายเป็นบิตที่เพิ่มเติมขึ้นเพื่อระบุถึงขอบเขตการสิ้นสุดของกลุ่มบิตข้อมูลบิตสุดท้ายนี้อาจจะมีจำนวนมากกว่าหนึ่งบิตได้ คือ 1 บิต 11/2 บิต และ 2 บิต ดังนั้นกรณีของการส่งข้อมูล 8 บิตพร้อมบิตที่เพิ่มเติมเข้าไปโดย



รูปที่ 3.2 แสดงลักษณะของข้อมูลแบบอนุกรมพร้อมบิตเริ่มต้นบิตพาริตี้และบิตสุดท้าย

3.3 หลักการและโครงสร้างของการสื่อสารข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมเป็นการสื่อสารที่การรับส่งข้อมูลจะใช้สายสัญญาณจำนวนน้อย ซึ่งปกติจะใช้เพียง 1 คู่ คือสายสัญญาณที่เป็นข้อมูล และ สายกราวด์เปรียบเทียบกัน โดยข้อมูลจะส่งออกหรือรับเข้าในลักษณะที่เป็นบิตต่อบิต ซึ่งเมื่อเราเปรียบเทียบกับการสื่อสารข้อมูลแบบขนานที่จำนวนข้อมูลและอัตราเร็วในการสื่อสารข้อมูลเท่ากันแล้วการสื่อสารข้อมูลแบบอนุกรมจะต้องใช้เวลาในการรับ-ส่งข้อมูลมากกว่าอย่างแน่นอน แต่ข้อดีของการสื่อสารข้อมูลแบบอนุกรมนี้ คือ การใช้สายสัญญาณน้อยกว่า และสามารถส่งสัญญาณได้ในระยะทางไกลกว่าแม้ว่าอัตราการลดทอนหรือผิดเพี้ยนของสัญญาณที่มีผลจากความยาวของสายสัญญาณจะมีค่าเท่ากับการ

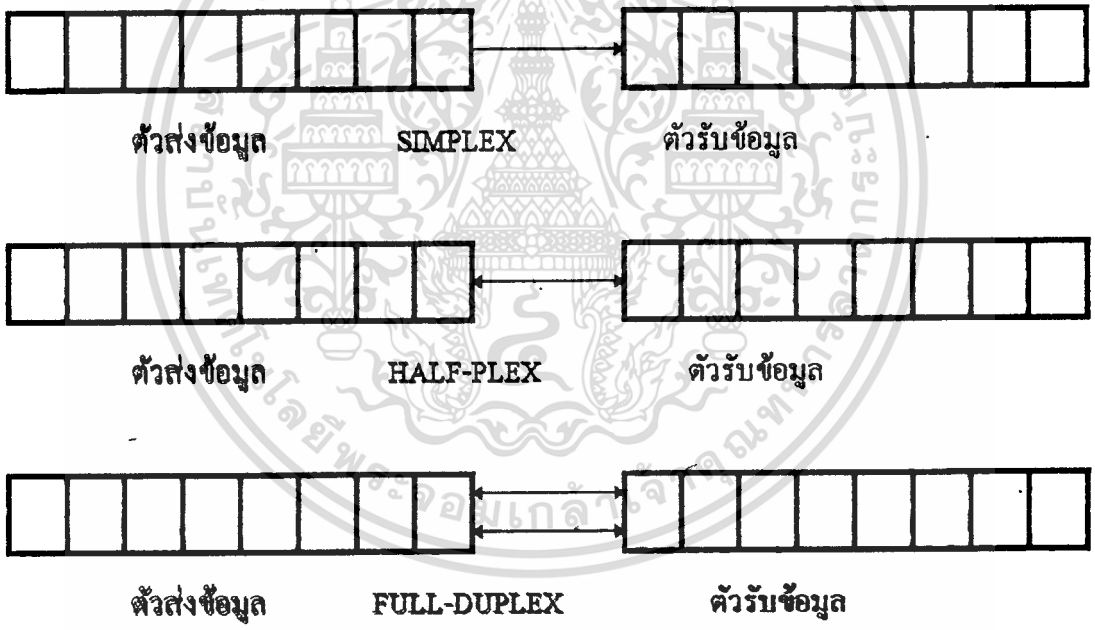
สื่อสารข้อมูลแบบขนาน แต่การสื่อสารข้อมูลแบบอนุกรมมีวิธีในการที่จะลดผลจากการลดทอนของสัญญาณนี้โดยอาศัยหลักการรับ-ส่งสัญญาณแบบดิฟเฟอเรนเชียล ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงเหมาะสำหรับใช้กับการสื่อสารข้อมูลระยะไกล หรือการสื่อสารที่ต้องการใช้สายหรือช่องสัญญาณในการรับ-ส่งข้อมูลจำนวนน้อย เช่น การสื่อสารข้อมูลโครงข่ายแบบท้องถิ่น (Local Area Network : LAN)

จากที่ได้กล่าวไปแล้วการสื่อสารข้อมูลแบบอนุกรมยังสามารถที่จะแบ่งตามลักษณะของทิศทางในการสื่อสารข้อมูล ตามโครงสร้างและความต้องการของระบบ ได้ดังนี้คือ

3.3.1. การสื่อสารข้อมูลในทิศทางเดียวตลอดเวลา : simplex

3.3.2. การสื่อสารข้อมูลใน 2 ทิศทางแต่สลับเวลา : half - duplex

3.3.3. การสื่อสารข้อมูลใน 2 ทิศทางตลอดเวลา : full-duplex



รูปที่ 3.3 แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรม

3.3.1 simplex : เป็นการสื่อสารในทิศทางใดก็จะใช้ทิศทางนั้นตลอดเวลาไม่มีการเปลี่ยนแปลงทิศทาง เช่นการส่งสัญญาณภาพจากสถานีไปยังเครื่องรับ หรือการส่งข้อมูลไปยังวิทยุติดตามตัว

3.3.2 half-duplex : เป็นการสื่อสารข้อมูลใน 2 ทิศทาง แต่ในขณะเวลาหนึ่งนั้นสัญญาณจะไปได้ในทิศทางเดียวเท่านั้น ดังนั้นอุปกรณ์แต่ละตัวที่จะเชื่อมต่อหรือสื่อสารข้อมูลในลักษณะนี้จะต้องเป็นได้ทั้งตัวรับและตัวส่ง ซึ่งมีชื่อเรียกว่า ทรานซีฟเวอร์ (transceiver) และจะต้องมีวงจรที่เลือกว่า ณ เวลานั้นจะทำงานเป็นตัวรับหรือตัวส่ง

3.3.3 full-duplex : เป็นการสื่อสารข้อมูลที่คล้ายกับ half-duplex แต่เป็นการสื่อสารข้อมูลใน 2 ทิศทางตลอดเวลา

แสดงลักษณะของการสื่อสารข้อมูลอนุกรมแบบต่างๆในรูปที่ 3.3

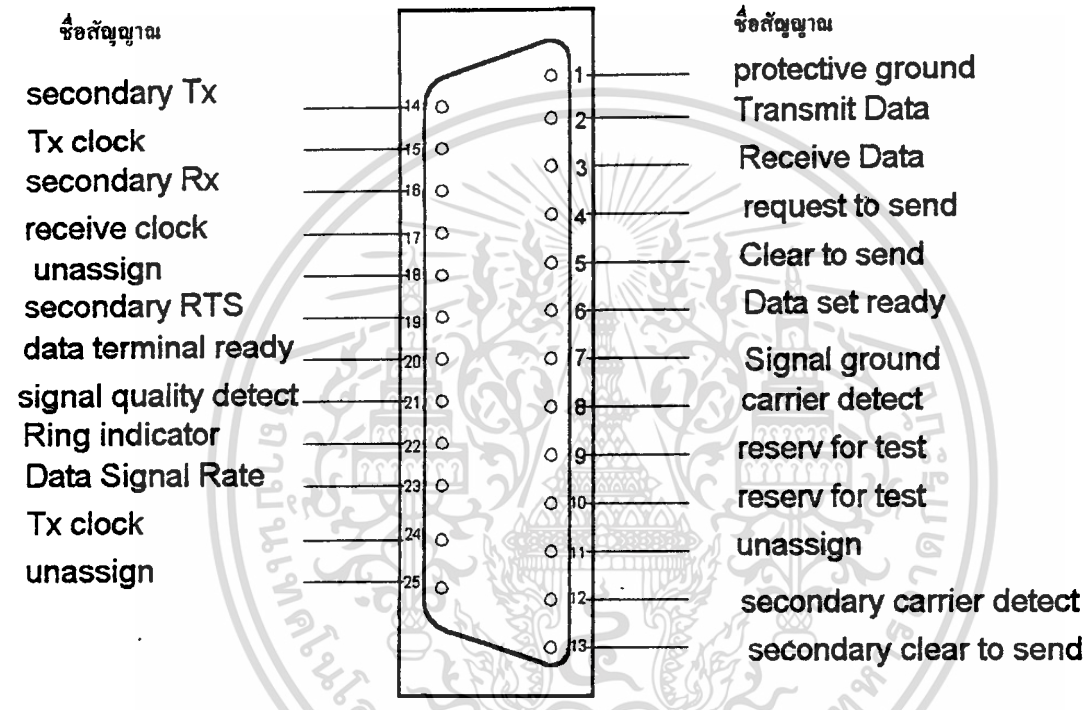
3.4 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

การสื่อสารข้อมูลแบบอนุกรมที่ใช้งานอยู่ในปัจจุบันนั้นได้มีการกำหนดมาตรฐานการรับ-ส่งข้อมูลไว้หลายแบบด้วยกัน แต่ที่ได้รับความนิยมนำมาใช้งานอย่างมาก คือการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C และที่มาตรฐานนี้เป็นที่นิยมเนื่องจากเป็นระบบการสื่อสารข้อมูลที่ใช้ในเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งเป็นคอมพิวเตอร์ที่มีใช้อย่างแพร่หลายจากอดีตมาถึงปัจจุบัน มาตรฐาน RS-232Cมีโครงสร้างการสื่อสารเป็นแบบจุดต่อจุดเท่านั้น โดยมีลักษณะสมบัติทางไฟฟ้าและทางกายภาพ

รูปที่ 3.4 แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

3.5มาตรฐาน RS 232

มาตรฐานของ RS 232 กำหนดข้อต่อแบบ DB-25 แต่ละขาของข้อต่อกำหนดไว้ดังรูป 3.5 อย่างไรก็ตามผู้ผลิตไมโครคอมพิวเตอร์อาจจะใช้ข้อต่อชนิดอื่นที่นอกเหนือไปจาก DB-25 สัญญาณต่าง ๆ ถูกมอบหมายให้ทำหน้าที่ดังนี้



รูปที่ 3.5 แสดงสัญญาณต่าง ๆ ของ DB-25

และสามารถแสดงรายละเอียดเกี่ยวกับ pin ที่สำคัญที่ใช้ในการส่งข้อมูลกับเครื่องพิมพ์ได้ดังนี้

3.5.1 **Transmit data** เป็นสัญญาณที่ส่งออกจาก DTE (หรือตัวไมโครคอมพิวเตอร์)ไปยัง

โมเด็มหรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออก สถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ "1"

3.5.2 **Receive data** เป็นทางของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพทางลอจิกเป็น "1"

3.5.3 **Request to sent** ใช้สำหรับส่งสัญญาณไปยังเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้ใช้คู่กับ CTS หรือ Clear to sent ที่อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็ส่งสัญญาณออกไปที่สายCTS

3.5.4 **Clear to sent** คังอธิบายไว้ใน RTS เมื่อสัญญาณนี้อยู่ในสถานะออฟ (negative voltage) หรือ ลอจิก "1" หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

3.5.5 **Signal ground** ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุก ๆ สาย ของสัญญาณ จะมีแรงดันเป็น "0" เมื่อเทียบกับสัญญาณตัวอื่น

บทที่ 4

การใช้งาน Z80-PIO Z80-SIO Z80-CTC

4.1 การใช้งาน Z80 PIO

Z80 PIO เป็นซิพซัพพอร์ตที่สร้างขึ้นมาสำหรับใช้กับ Z80 โดยเฉพาะ สำหรับ Z80 PIO นี้จะทำหน้าที่ I/O (INPUT/OUTPUT PORT) ที่เราสามารถจะเลือกโปรแกรมให้ทำงานในลักษณะต่างๆกันได้คือ อาจจะทำให้ทำงานเป็นพอร์ตอินพุท พอร์ตเอ้าท์พุท หรือเป็นทั้งพอร์ตอินพุทและพอร์ตเอ้าท์พุทได้ในเวลาเดียวกัน และจะทำให้มีการแฮนด์เชค (HANDSHAKE) หรือไม่ก็ได้ขึ้นกับโหมดการทำงานที่เราโปรแกรมได้ ซึ่งมีอยู่ถึง 4 โหมด คือ โหมด 0 ถุง โหมด 3 แต่เราจะกล่าวถึงโหมด 1 เท่านั้น

4.1.1 การจัดเรียงขาบน Z80 PIO

เราจะสามารถแยกขาต่างๆบน Z80 -PIO ออกเป็นกลุ่มๆตามลักษณะการใช้งานได้ดังนี้คือ

1) กลุ่มที่ใช้เชื่อมต่อกับ CPU : หน้าที่ของขาในกลุ่มนี้จะทำการติดต่อกับ CPU โดยตรงซึ่งจะประกอบไปด้วย

1.1) D7-DO:ทำหน้าที่ในการรับหรือส่งข้อมูลให้กับ CPU

1.2) B/A SELECT: (PORT B/A SELECT) : หน้าที่ของขานี้ก็คือทำให้ PIO ทราบว่า CPU ต้องการติดต่อกับพอร์ตB (กรณีที่ได้รับลอจิก "1")หรือพอร์ต A (ในกรณีที่ได้รับลอจิก "0")โดยปกติแล้วขา B/A SELECT นี้จะต่อกับขา AO ของ CPU โดยตรง

1.3) C/D SELECT : (CONTROL/DATA SELECT): ใช้ในการบอก PIO ข้อมูลที่ CPU ส่งให้กับPIO นั้นเป็นคำสั่งควบคุม(CONTROL WORD : ในกรณีที่ได้รับลอจิก "0") สำหรับคำสั่งควบคุมนั้นจะใช้ในการโปรแกรม PIO เพื่อเลือกโหมดการทำงานนี้โดยปกติจะต่อเข้ากับขานี้โดยปกติจะต่อเข้ากับขา A1 ของ CPU โดยตรง

1.4) CE: (CHIP ENABLE INPUT): การทำงานของขา CE นี้ก็คือเมื่อขานี้ได้รับลอจิก "0"PIO ก็สามารถจะรับหรือส่งข้อมูลให้กับอุปกรณ์อินพุทได้และเมื่อขา CE นี้ได้รับลอจิก "1" PIO ก็ไม่สามารถที่จะรับหรือส่งข้อมูลได้

1.5) M1: (MACHINE CYCLE 1) : สำหรับขานี้โดยปกติจะต่อกับขา M1 ของ Z80 โดยตรงซึ่งPIO จะใช้สัญญาณที่ขานี้มาควบคุมการทำงานหลายๆอย่างภายในตัว PIO โดยที่เมื่อทั้ง M1 และ RD เป็นลอจิก "0"ทั้งคู่ (FETCH OPCODE) อยู่ซึ่งเมื่อ PIO ได้รับสัญญาณคู่นี้ก็จะคอยอ่านข้อมูลบนบัสข้อมูล(Data Bus) ว่าเป็นออฟโคดของคำสั่ง RET หรือไม่ ถ้าใช่ก็จะไปทำให้ขา IEO มีสถานะเป็น "1"ซึ่งจะกล่าวถึงการใช้งานในภายหลังและในกรณีที่ทั้งขา M1 และ IORQ ได้รับลอจิก "0"ทั้งคู่ก็แสดงว่า Z80 ส่งสัญญาณตอบสนองการอินเทอร์รัพท์ออกมา และ

เนื่องจาก PIO ไม่มีขาที่ใช้ในการรีเซ็ตโดยตรงดังนั้นจึงได้มีการกำหนดได้ว่าขา M1 แอคทีฟเพียงขาเดียว(IORQ และ RD ไม่แอคทีฟตาม)PIO จะถือว่าเป็นการรีเซ็ตซึ่งจะกล่าวถึงการใช้งานในขานี้ภายหลัง

1.6)IORQ: สำหรับขานี้จะต่อโดยตรงกับขา IORQ ของ Z80 และใช้ร่วมกับ B/A SELECT C/D SELECT และ CE เพื่อการแลกเปลี่ยนข้อมูลคือเมื่อ IORQ กับ CE ได้รับลอจิก "0"ทั้งคู่ (ACTIVE) ก็จะแสดงว่า Z80 ต้องการที่จะติดต่อPIO ตัวนั้นนั่นเองและเมื่อ IORQ กับM1 ได้รับลอจิก "0" (ACTIVE) ทั้งแสดงว่าZ80ได้รับการตอบรับการอินเทอร์รัพท์มาซึ่งถ้า PIO ถูกโปรแกรมได้ให้ตอบสนองต่อการอินเทอร์รัพท์PIO ก็จะต้องได้รับอินเทอร์รัพท์เวคเตอร์ไปบนบัสข้อมูลอัตโนมัติ

1.7) RD: ขา RDนี้ก็จะต่อโดยตรงกับ RD ของZ80เช่นเดียวกัน เราทราบมาแล้วว่าถ้าสัญญาณจากขา RDของ Z80 นี้เป็น "0" ก็แสดงว่าZ80 ต้องการที่จะอ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์I/O ดังนั้นในกรณีที่ขาทั้ง IORQ และ CE เป็น "0" ก็แสดงว่าZ80ต้องการที่จะอ่านข้อมูลจากPIOและเนื่องจากบนPIOไม่มีขาWR ดังนั้นPIO จึงต้องหลักการที่ว่าขา RDและ WRจะไม่แอคทีฟพร้อมกัน(ไม่เป็น "0" พร้อมกัน) คือถ้าขาRDเป็น "0"ขา WR ก็จะเป็น "1" (ซึ่งจะเป็นช่วงเวลาที่Z80อ่านข้อมูลจากภายนอก)และถ้าขา RD เป็น "1" และขา WR ก็จะเป็น "0" (ซึ่งจะเป็นช่วงเวลาที่Z80 ทำการส่งข้อมูลออก)ดังนั้นPIO จึงใช้วิธีตรวจสอบที่ว่า IORQ และ CE แอคทีฟแต่ RDไม่แอคทีฟตามก็แสดงว่า Z80 ต้องการจะส่งข้อมูลให้กับPIO นั้นเอง

1.8):ขาคล็อก(CLOCK) ของPIOจะต่อเข้ากับคล็อกของระบบซึ่งใช้สำหรับZ80ด้วย

2)กลุ่มที่เกี่ยวกับการอินเทอร์รัพท์ สำหรับขาในกลุ่มนี้จะทำหน้าที่ในการควบคุมการขออินเทอร์รัพท์ของ PIOซึ่งประกอยด้วย

2.1) INT สำหรับขา INT นี้ เป็นขาเอาท์พุทของPIOซึ่งจะแอคทีฟเมื่อ PIO ต้องการที่จะขออินเทอร์รัพท์ โดยสามารถที่จะควบคุมโดยวิธีการทางซอฟแวร์ และยังขึ้นอยู่กับขา IEI ของPIOด้วย (ซึ่งจำกล่าวถึงในภายหลัง)

2.2) IEI (INTERRUPT ENABLE IN)และIOE(INTERRUPT ENABLE OUT) :สำหรับขาทั้งสองนี้จะใช้สำหรับการทำ DAISY CHAIN PRIORITY

3) กลุ่มที่ใช้เกี่ยวกับการติดต่อระหว่างอุปกรณ์ภายนอกขาในกลุ่มนี้จะทำหน้าที่ในการติดต่อรับหรือส่งข้อมูลให้กับอุปกรณ์ภายนอกซึ่งสามารถที่จะแยกได้เป็น2พอร์ตคือ พอร์ตA พอร์ต B

3.1) พอร์ตA ประกอบด้วย บัสข้อมูล 8 เส้นและบัสควบคุมอีก 2 เส้นคือ

3.1.1) A7-A0:ทำหน้าที่ในการรับหรือส่งข้อมูลให้กับอุปกรณ์ภายนอก โดยที่ A0 จะเป็นบิตที่มีนัยสำคัญสูงสุด (LEAST -SIGNIFICANT BIT) ของข้อมูล

3.1.2) ASTB และ ARDY (PORT A STROBE PULSE INPUT REGISTER A READY):เป็นขาที่ใช้สำหรับการทำงานในโหมดที่มีการทำ HANDSHAKE

3.2) พอร์ตB ประกอบด้วยบัสข้อมูล 8 เส้น และบัสควบคุม 2 เส้น เช่นกันคือ

3.2.1) B0-B7 ทำหน้าที่รับหรือส่งข้อมูลของพอร์ตB กับอุปกรณ์ภายนอก

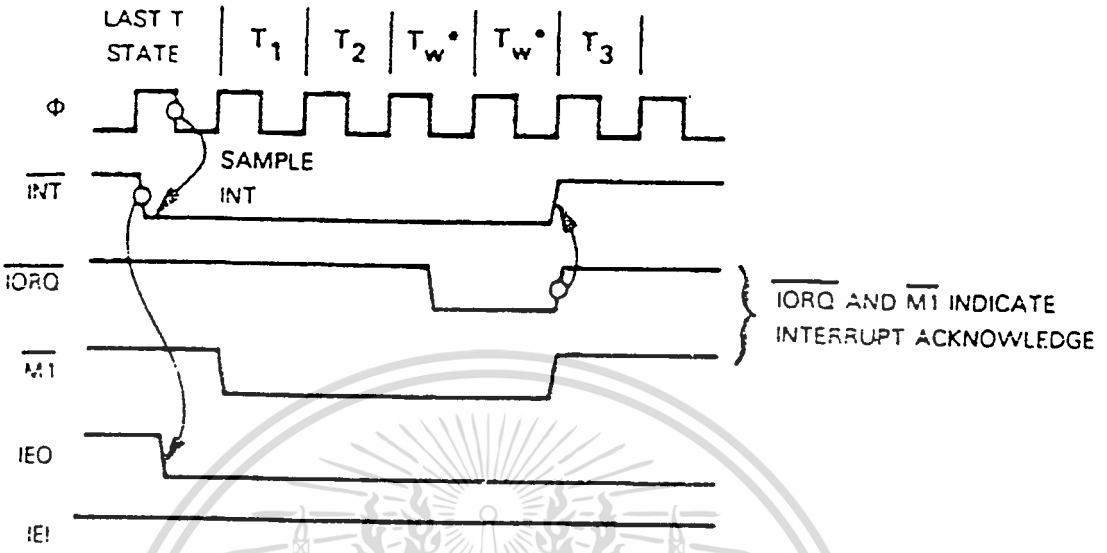
3.2.2) BSTB:(PORT B STROBE PULSE INPUT) และBRDY:(REGISTER B READY): ใช้สำหรับการทำงานในโหมดที่มีการทำแฮนด์เชค เช่นเดียวกับ ASTB ซึ่งจะกล่าวถึงในภายหลัง

4.1.2 การทำงานของ Z80 ในโหมด 1

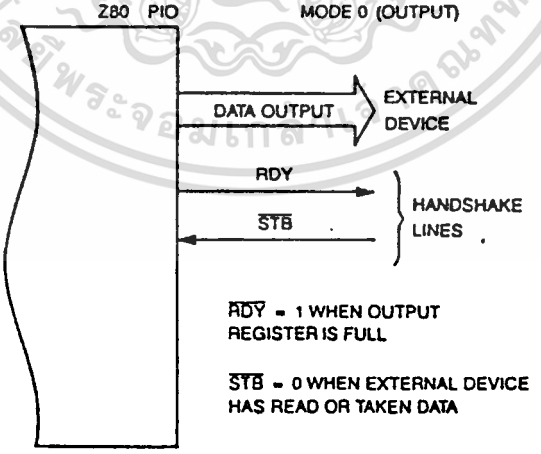
ลักษณะการทำงานของPIOในโหมดนี้คือการทำให้พอร์ตที่ถูกโปรแกรมทำหน้าที่เป็นพอร์ตอินพุทคือ รับข้อมูลจากอุปกรณ์ภายนอกได้เพียงอย่างเดียว สำหรับคำสั่งควบคุมที่ใช้โปรแกรมPIOให้ทำงานในโหมดนี้คือ 4FH (D7=0,D6=1) โดยส่งไปยังพอร์ตแอดเดรส(Adress Port) 92H หรือ 93H (พอร์ต A หรือ B CONTROL) สำหรับในที่นี้เราจะให้พอร์ต B เป็นพอร์ตอินพุท ดังนั้นเราจะต้องส่งข้อมูล 4FH ไปยังพอร์ต 93H ในโหมดนี้ข้อมูลจากอุปกรณ์ภายนอกจะส่งให้กับ PIOที่ขา B7-B0ของ PIO (ขา A0-A7 ในกรณีที่ใช้พอร์ต A เป็นพอร์ตอินพุท)

ในกรณีที่เราไม่ต้องการที่จะทำแฮนด์เชค ในโหมดนี้ ก็เพียงแต่ทำให้ขา BSTB ได้รับลอจิก "0" ซึ่งอาจจะทำได้โดยการต่อขา BSTB กับ GROUND ทำเช่นนี้จะทำให้ PIO รับข้อมูลที่อยู่บน B7-B0 (หรือ A7-A0) โดยไม่ต้องรอ BSTBจากอุปกรณ์ภายนอก

สำหรับกรณีที่เราต้องการที่จะทำแฮนด์เชค นั้น เราจะต้องใช้งานขา ASTB กับ ARDY (สำหรับพอร์ต A)และ BSTB กับ BRDY (สำหรับพอร์ต B) ในที่นี้เราจะต้องใช้ขา BSTB กับ BRDY โดยเมื่อขา BSTB แอคทีฟ (ได้รับลอจิก "0" PIO ก็จะทำการอ่านข้อมูลจากบัสข้อมูลของพอร์ตทำหน้าที่เป็นพอร์ตอินพุท) ซึ่งในที่นี้คือ B7-B0 (หรือ A7-A0 สำหรับพอร์ต A) นั่นเอง ส่วนขา BRDY นั้นจะเป็นเอาท์พุทที่ทำหน้าที่ในการแสดงให้อุปกรณ์ภายนอกทราบว่าอินพุทรีจิสเตอร์ (INPUT REGISTER)นั้น พร้อมที่จะรับข้อมูลต่อไปได้ (CPU ได้ทำการอ่านข้อมูลใน INPUT REGISTER แล้ว)จากรูปแสดงบล็อกไดอะแกรมการต่อบัสข้อมูล BSTB และ BRDY กับอุปกรณ์ภายนอก



รูปที่ 4.1 ไดอะแกรมเวลาแสดงการขออินเทอร์รัพท์ของ PIO

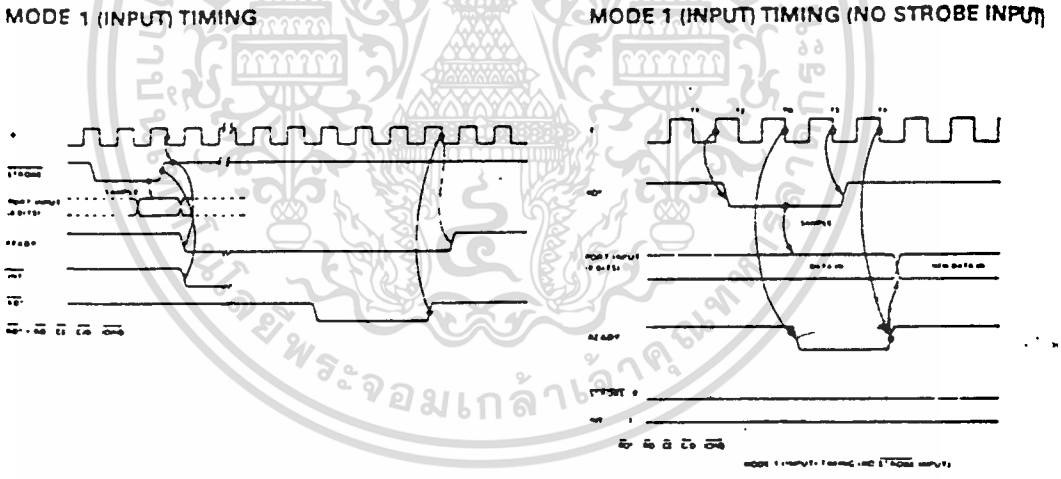


รูปที่ 4.2 แสดงไดอะแกรมการต่อ PIO กับอุปกรณ์ภายนอก

เมื่ออุปกรณ์ภายนอกส่งข้อมูลให้แก่ PIOแล้ว PIO จะทำการขออินเทอร์รัพท์ (ในกรณีที่ PIOทำงานในลักษณะที่ไม่ทำ HANDSHAKE นั้น PIO จะไม่ทำการขออินเทอร์รัพจาก Z80)

โดยทำให้ขา INT แอคทีฟ (แอคทีฟที่ละจิก “0”) จากนั้นก็จะเป็นหน้าที่ของโปรแกรมตอบสนองการอินเทอร์รัพท์ที่จะทำให้ Z80 อ่านข้อมูลจากพอร์ตอินพุทแล้ว PIO (ในกรณีนี้คือพอร์ต B)เมื่อ Z80 อ่านข้อมูลจากพอร์ตอินพุทแล้ว PIOก็จะทำให้ขา BRDY แอคทีฟ (ลอจิก “1”)เพื่อที่จะทำให้อุปกรณ์ภายนอกทราบว่า PIOพร้อมที่รับข้อมูลชุดใหม่แล้ว (อุปกรณ์ภายนอกจะไม่ทำการส่งข้อมูลให้กับ PIOจนกว่าขา BRDY ของ PIO จะมีระดับลอจิกเป็น “1”)

สิ่งหนึ่งที่ต้องคำนึงถึงคือเมื่อเราโปรแกรมเลือกการทำงานในโหมดนี้แล้ว เราจะต้องทำการอ่านข้อมูลจาก PIO ในทันที เนื่องจากเมื่อ PIO ถูกรีเซ็ตนั้นขา BRDY ของ PIO จะถูกทำให้เป็น “0” ดังนั้นการอ่านข้อมูลจาก PIO ก็เพื่อที่จะทำให้ขาBRDY มีระดับลอจิกเป็น “1”นั้นเอง อย่างไรก็ตามข้อมูลที่อ่านเข้ามานี้เป็นข้อมูลที่เราไม่สนใจว่าจะมีค่าเป็นอะไร เนื่องจากเราทำการอ่านข้อมูลนี้เพื่อทำให้ขา BRDY ลอจิกเป็น “1” เท่านั้น สำหรับไดอะแกรมเวลาการทำงานของ PIO ในโหมดนี้ได้แสดงได้ในรูป



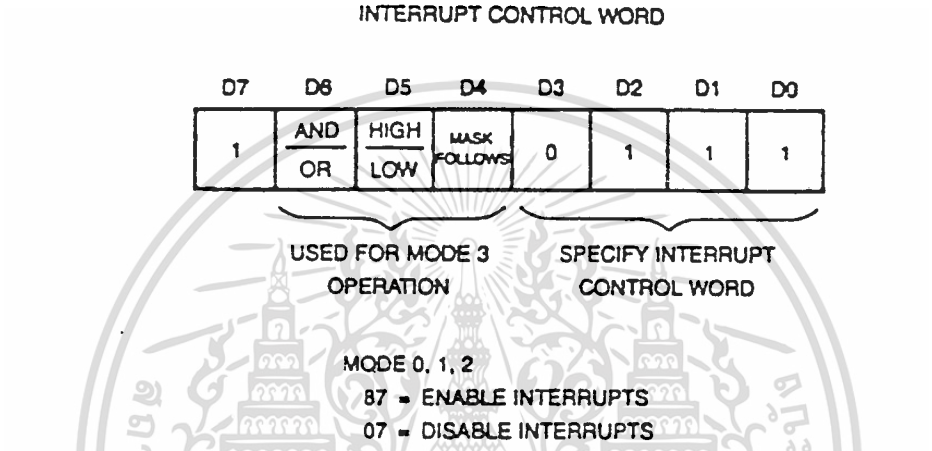
รูปที่ 4.3 ไดอะแกรมเวลาแสดงการทำงานของ PIO ในโหมด 1

4.1.3 การควบคุมคำสั่งการอินเทอร์รัพท์ (INTERRUPT CONTROL WORD)

สำหรับหัวข้อที่ผ่านมานั้นเราได้ศึกษาวิธการโปรแกรม PIO ให้เลือกการทำงานในโหมด 0 หรือ โหมด 1 ให้กับพอร์ต A หรือพอร์ต B ในหัวข้อนี้จะศึกษาถึงวิธีการที่จะโปรแกรมการอินเทอร์รัพท์และวิธีการที่จะใช้กับการทำงานของ PIO ในโหมด 0 และ โหมด 1

เช่นเดียวกับการเลือกโหมดขั้นแรกจะต้องทำการส่งข้อมูลให้กับรีจีสเตอร์ควบคุม (CONTROL REGISTER) เพื่อโปรแกรมการทำงานให้กับ PIO เสียก่อน แต่ข้อมูลที่ส่งไปนี้จะมี

บิต D3-D0 เป็น 0111Bแทนที่จะเป็น 0011B สำหรับบิต D7-D4 นั้นจะถูกใช้ในการเขียนโปรแกรมการทำงานของ PIO ในการอินเทอร์รัฟท์และแรงเงาเรียกข้อมูลที่ถูกส่งให้กับรีจีสเตอร์ควบคุมในลักษณะนี้ว่า "INTERRUPT CONTROL WORD"จากรูปจะแสดงหน้าที่ของแต่ละบิตบน INTERRUPT CONTROL WORD



รูปที่ 4.4 แสดงการจัดเรียงบิตบน INTERRUPT CONTROL WORD

ถ้าบิต D7 ของ INTERRUPT CONTROL WORD เป็น "1" จะทำให้ PIOพร้อมที่จะทำการขออินเทอร์รัพท์ แต่ถ้าบิต D7 นี้เป็น "0" PIO จะไม่สามารถทำการขออินเทอร์รัพท์ได้

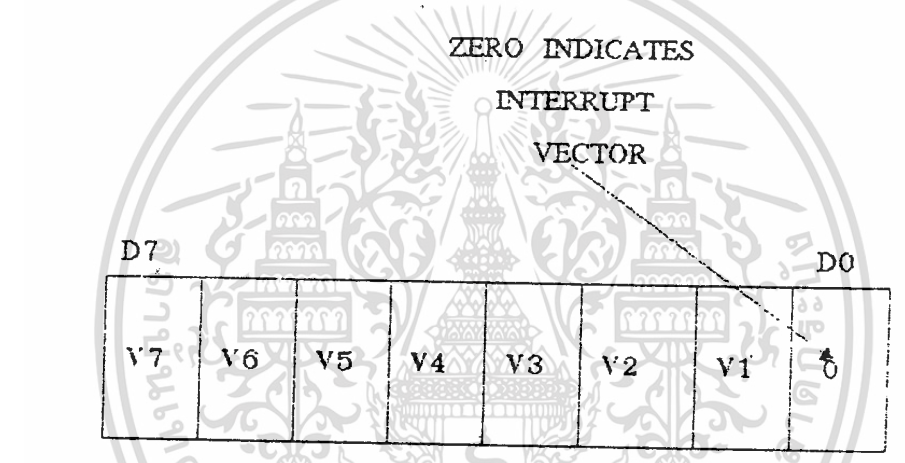
สำหรับโหมด 0 และโหมด 1 ของ PIO นั้นเราจะสนใจแต่เพียงการENABLE หรือการ DISABLEการขออินเทอร์รัพท์เท่านั้น ดังนั้นในกรณีของโหมด 0 และโหมด 1 ค่าของบิต D6-D4 เป็น "0" ทั้งหมด ดังนั้นจึงมีข้อมูลเพียง 2 ค่าที่จะเลือกส่งให้กับรีจีสเตอร์ควบคุมเท่านั้น

87H สำหรับการ ENABLE INTERRUPT

07H สำหรับการ DISABLE INTERRUPT

ในกรณีที่เราขอมให้ PIO ของอินเทอร์รัพท์ได้นั้น (ส่งข้อมูล 87H ให้กับ CONTROL REGISTER)สิ่งที่เราต้องทำขั้นตอนต่อไปคือการกำหนดค่าของอินเทอร์รัพท์เวคเตอร์

(INTERRUPT VECTOR) ที่จะต้องส่งให้กับ Z80 สำหรับ อินเทอร์รัพท์เวคเตอร์นี้เราสามารถที่จะส่งให้กับPIOได้โดยวิธีเดียวกับการเลือกโหมดหรือการเขียนโปรแกรม INTERRUPT CONTROL WORD เพียงแต่ว่าในกรณี การโปรแกรม อินเทอร์รัพท์เวคเตอร์นั้นค่าบิต D0 จะมีค่าเป็น "0" (ค่าของ อินเทอร์รัพท์เวคเตอร์จะต้องเป็นเลขคู่เท่านั้น) สำหรับรูปแบบของบิตต่างๆ บนอินเทอร์รัพท์เวคเตอร์จะแสดงดังรูปที่ 4.5



VECTOR TO BE PLACED ON DATA BUS WHEN AN INTERRUPT OCCURS

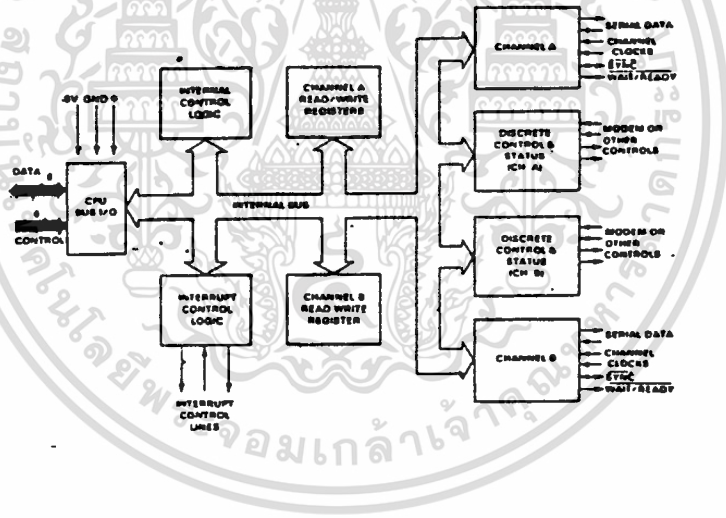
รูปที่ 4.5 การจัดเรียงบิตบน INTERRUPT VECTOR

4.2 การใช้งาน Z80-SIO

4.2.1 บล็อกไดอะแกรมของ Z80-SIO

SIO ประกอบด้วยแชนแนลการระบบส่งข้อมูลแบบอนุกรมซึ่งทำงานเป็นอิสระต่อกันถึง 2 แชนแนล(Channel) คือ แชนแนล A และ B โดยแต่ละบล็อกจะทำงานร่วมกับหน่วยควบคุม (Control Block) และหน่วยแสดงสถานะ (Status Block) โดยหน่วยควบคุมและหน่วยแสดงสถานะจะเชื่อมโยงกับอินพุทและเอาท์พุทจำนวนมาก

พิจารณาบล็อกทางด้านซ้าย 2 บล็อกก่อน คือ Channel A และ Channel B Read/Write Register ซึ่งบล็อกเหล่านี้จะเป็นลอจิกภายใน (Internal Logic) ที่ช่วยให้การโปรแกรมใช้งาน Z80-SIO เป็นไปอย่างมีประสิทธิภาพ โดยที่ทุกๆโหมดการทำงานของ Z80-SIO สามารถควบคุมได้ด้วยซอฟแวร์ ก่อนที่จะมีการใช้งาน Z80-SIO จะต้องมีการโปรแกรมบล็อกเหล่านี้เพื่อจัดลักษณะการทำงานให้พร้อมก่อนที่จะถูกนำไปใช้งาน



รูปที่ 4.6 บล็อกไดอะแกรมของ Z80-SIO

บล็อก Interrupt Control Logic บล็อกนี้ช่วยให้ Z80-SIO สามารถเชื่อมต่อกับโครงสร้างการอินเทอร์รัพท์ของ Z80 และ Peripheral Device ได้ซึ่งโครงสร้างทางด้านการอินเทอร์รัพท์ของ SIO นี้มีประสิทธิภาพสูงมาก

บล็อกของ Internal Logic จะใช้ในการควบคุมการส่งผ่านข้อมูลภายในตัว Z80-S10

สุดท้ายเป็นบล็อกของ CPU I/O BUS ซึ่งใช้ในการอินเทอร์เฟส ระหว่าง Z80-CPU กับ SIO นอกจากนี้ยังสามารถที่ใช้เป็นบัฟเฟอร์ได้อีกด้วย สิ่งที่จะกล่าวถึงต่อไปจากนี้คือการเชื่อมต่อ SIO กับZ80-CPU

4.2.2 รายละเอียดขอบขาต่างๆของ SIO

B/A : (B / A SELECT): ถ้าขานี้เป็นลอจิก "1"จะเลือกแชนแนล B และถ้าเป็นลอจิก "0" จะเลือกแชนแนล A ขานี้ใช้เป็นขาอินพุท ซึ่งโดยปกติแล้วจะต่อขานี้เข้ากับA0 จากบัสแอดเดรสของ Z80 CPU

C / D : (CONTROL/DATA SELECT): ขานี้เป็นขาอินพุทที่ยงใช้ในการเลือกว่าข้อมูลที่ CPU ส่งให้กับ ISO เป็นข้อมูลที่ใช้ในการควบคุมการทำงานของ ISO หรือเป็นข้อมูลที่ต้องการจะส่งให้กับอุปกรณ์ภายนอกโดยผ่าน ISO ถ้าขานี้ได้รับหลอจิก "1" ก็จะเป็นการติดต่อระหว่าง คอนโทรลรีจิสเตอร์ของแชนแนล A หรือ B เข้ากับ Z80 CPU ถ้าเป็นลอจิก "0" Z80-S10 จะอยู่ในโหมดของการรับส่งข้อมูล ชานี้ปกติจะต่ออยู่กับขา A1 ของZ80CPUและเนื่องจากการที่ขา A0 และ A1 ถูกต่อเข้ากับ SIO จึงทำให้ SIO มีพอร์ตแอดเดรสถึง 4 ค่า

CE : (CHIP ENABLE) : เป็นขาอินพุทซึ่งจะแอคทีฟ รับ ลอจิก "0" ในกรณีที่ซีพียูจะติดต่อกับ SIO ขานี้จะต้องถูกทำให้แอคทีฟ เพื่อให้สามารถที่จะเขียนและอ่านจากรีจิสเตอร์ภายใน(Internal Register)ได้ ปกติสัญญาณที่ ขา CE นี้จะได้จากการถอดรหัสแอดเดรส 6 บิต บนของแอดเดรสไบต์ต่ำ (A7-A2)

CLOCK: เป็นขาอินพุทของสัญญาณคล็อกที่ใช้ภายในตัว SIO ซึ่งสัญญาณคล็อกนี้เป็นสัญญาณเดียวกับสัญญาณคล็อกที่ป้อนให้กับ CPU เพื่อใช้ในการควบคุมการติดต่อภายในของ SIO ให้มีความสัมพันธ์กัน ซึ่งขานี้จะมีลักษณะทาง DC เหมือนกับ Z80

M1:(MACHINE CYCLE ONE): ขานี้เป็นขาอินพุทที่ต่อกับขา M1ของ Z80-CPUในกรณีที่ขา M1 และขา RDของ SIO ได้รับลอจิก "0" แสดงว่า Z80 กำลังทำขบวนการเฟทซ์ออฟโคด (Fetch Op Code) จากหน่วยความจำในช่วงเวลานี้ SIO จะทำการตรวจสอบออฟโคดบนบัสข้อมูลว่าตรงกับออฟโคดของคำสั่ง RETI หรือไม่ ถ้าเป็นคำสั่ง RETI SIO จะทำการถอนสัญญาณ IEOโดยการทำให้สัญญาณที่ขานี้เป็นลอจิก "1" ถ้าขา M1กับขา IORQมีระดับลอจิกเป็น "0" Z80 จะอยู่ในขบวนการตอบรับการอินเทอร์รัพท์ (Interrupt Acknowledge) และ SIO จะถอนการอินเทอร์รัพท์เองเมื่อระดับสัญญาณทั้งสองนี้เปลี่ยนจากลอจิก "0" เป็น "1" (ขอบขาขึ้นของสัญญาณ)

IORQ : (INPUT/OUTPUT REQUEST): ถาอินพุทขานี้จะต่อโดยตรงกับขา IORQ ของ Z80

RD : (READ): เป็นขาอินพุทที่ต่อเข้ากับขา RD ของ Z80 โดยตรงถ้าขา RD ขา IORQ และขา CE มีระดับลอจิก เป็น "0"CPU จะอ่านข้อมูลจาก SIO จะสังเกตได้ว่า SIO นี้ไม่มีขาอินพุท WR ที่ใช้เขียนข้อมูล ฉะนั้นถ้าต้องการเขียนข้อมูลให้กับ SIO ก็สามารถทำได้โดยให้ขา RDนี้มีลอจิก "1"ในขณะที่ IORQ และ CE เป็นลอจิก "0"ข้อมูลก็จะถูกส่งไปยังพอร์ตที่ถูกเลือกโดยการควบคุมจากขา C / D และ B / A

RESET : ถ้ามีระดับลอจิกเป็น "0" s10 จะทำการ Disable การรับส่งข้อมูลของ ทั้งแชนแนล A และ B ที่เอาท์พุทของทั้งสองแชนแนลนี้จะอยู่ในสภาวะ Marking การอินเทอร์รัพท์จะถูก Disable ในการนี้สัญญาณควบคุม Modem จะถูกเซ็ตให้เป็น "1" ซึ่ง S10 จะต้องถูกโปรแกรมใหม่ (ใน คอนโทรล รีจิจิสเตอร์) หลังจากที่มีการรีเซ็ตแล้ว

IE1: (INTERRUPT ENABLE INPUT): ขานี้จะถูกใช้ในการอินเทอร์รัพท์แบบ Daisy chain

IEO: (INTERRUPT ENABLE OUTPUT): ขานี้จะใช้งานร่วมกับขา IE1ในการอินเทอร์รัพท์แบบ Daisy Chain

INT : (INTERRUPT REQUEST OUT) : เป็นขาเอาท์พุทแบบ Open Drain และแอคทีฟที่ระดับลอจิก "0" ขานี่จะต่อเข้าโดยตรงกับขา INT ของ Z80

W/RDYA, W/RDYB : (WAIT/READY CHANNEL A/B): เป็นขา WAIT/READY แบ่งเป็นทางแชนแนล A และ B ซึ่งสามารถจะควบคุมได้โดยวิธีการทางซอฟแวร์ ถ้าขานี้ถูกโปรแกรมให้เป็น Ready Function ในจะสามารถทำตัวเป็นบัฟเฟอร์ที่สามารถขับกระแสได้โดยอาจใช้เป็นสัญญาณ Ready ให้กับส่วนควบคุม DMA(DMAController) เพื่อแสดงว่าเมื่อใดที่ SIO มีข้อมูลที่จะส่งออกไปหรือพร้อมที่จะรับข้อมูลเข้ามาจากอุปกรณ์ภายนอก,Wait ,Function จะใช้ในการติดต่อระหว่าง Z80-CPU กับ SIO เพื่อแสดงว่าเมื่อใดที่ SIO มีข้อมูลพร้อมที่จะส่งให้แก่ CPU หรือพร้อมที่จะรับข้อมูลจาก CPU เพื่อส่งออกไปยังอุปกรณ์ภายนอก

CTSA,CTSB : (CLEAR TO SEND A/B) : ขาอินพุทขานี้จะแอคทีฟที่ลอจิก "0" สามารถควบคุมการทางซอฟแวร์ ให้เป็น Auto Enable ได้ คืออุปกรณ์ภายนอกจะสามารถเริ่มทำการส่งข้อมูลเข้ามาได้ แต่ถ้าสัญญาณอินพุทนี้ไม่ได้ถูกโปรแกรมให้อยู่ในโหมด การทำงานแบบ Auto Enable แล้วก็จะถูกใช้เป็นขาอินพุทเอนกประสงค์ ของ SIO

DCDA,DCDB : (DATA CARRIER DETECT A/B) : ใช้เป็น Receiver Enable (ในหมด Auto Enable) ทำงานที่ลอจิก "0" และถ้าเป็นลอจิก "0" จะใช้เป็นขาอินพุทเอนกประสงค์ได้

RTSA,RTSB : (REQUEST TO SEND A/B) : ขาเอาท์พุททั้ง 2 ขานี้เป็นขาที่ใช้บอกอุปกรณ์ภายนอกว่า SIO พร้อมที่จะส่งข้อมูลได้และสัญญาณนี้จะมีสภาวะทางลอจิกเป็น "1" (In Active) เมื่อ Register transmitter ว่ากล่าวคือ SIO ได้ส่งข้อมูลออกไปหมดเรียบร้อยแล้ว แต่เมื่อนำขาที่กล่าวมาทั้งหมดนี้จะเป็นจริงเมื่อ RTS บิตใน Control Register ถูกเซ็ตให้เป็นลอจิก "1"เท่านั้น แต่ในกรณีติดต่อแบบ Synchronous สภาวะที่ขาRTS จะเป็นไปตามสภาวะทางลอจิกของ RTS บิตเท่านั้น นอกจากนี้ขาทั้ง 2 ยังสามารถโปรแกรมให้ใช้งานอื่นๆได้

DTRA,DTRB : (DATA TERMINAL READY A/B) : ขาเอาท์พุททั้งสองนี้จะมีระดับสัญญาณตามสถานะลอจิกของโปรแกรมบิต (Register)เพื่อแสดงว่า SIO พร้อมที่จะรับ,ส่งข้อมูลหรือยัง โดยสัญญาณนี้จะแอคทีฟที่ลอจิก "0"

RxDA, RxDB : (RECEIVE DATA INPUTS): ขานี้ใช้เป็นขาอินพุทที่ใช้ในการรับข้อมูลแบบอนุกรมจากระบบสายส่ง

TxDA,TxDB : (TRANSMITTER DATA OUTPUTS): ขาเอาท์พุทคู่นี้ใช้ในการส่งข้อมูลแบบอนุกรมจาก SIO ไปยังระบบสายส่ง

RxCA,RxCB : (RECEIVE CLOCK A/B) : ขาทั้งสองนี้เป็นขาอินพุทซึ่งใช้ในการรับสัญญาณคล็อก เพื่อใช้ควบคุมการรับส่งข้อมูลอนุกรม โดยสามารถโปรแกรมให้ความถี่ของสัญญาณคล็อกที่ขานี้เป็น 1,16,32,หรือ 64 เท่าของอัตราการรับข้อมูลที่ใช้

TxCA,TxCB : (TRANSMITTER CLOCK A/B): เป็นขาอินพุทซึ่งใช้ในการรับสัญญาณคล็อกเพื่อใช้ในการส่งข้อมูลอนุกรม โดยสามารถโปรแกรมความถี่ของสัญญาณคล็อกที่ขานี้ให้เป็น 1,16,32,หรือ 64เท่าของอัตราการส่งข้อมูลที่ใช้

4.2.3 SIO REGISTER

Z80-SIO ประกอบด้วย Internal Write Register 8 ตัวสำหรับแชนแนล B คือ WR0-WR7 และสำหรับแชนแนล A จะมี Internal Write Register 7 ตัว คือ WR0-WR7 และWR3-WR7 โดยไม่มี WR2

WR2เป็นรีจิสเตอร์ที่ใช้เก็บอินเทอร์รัพท์เวคเตอร์ (Interrupt Vector) สำหรับทั้ง 2 แชนแนลและบิตที่มีนัยสำคัญสูงสุดของแต่ละรีจิสเตอร์ คือ บิต D7 เราจะต้องเขียนข้อมูลลงในแต่ละรีจิสเตอร์เป็นจำนวน 2 ไบต์ ยกเว้น WRC คือ ไบต์แรกเขียนลงที่ WR0 , โดยไบต์นี้จะถูกถอดรหัสโดย SIO (Internally Decode) เพื่อบ่งบอกว่า Write Register ตัวใดที่จะถูกเขียนเป็นลำดับต่อไป เหตุที่ต้องอาศัยวิธีดังกล่าวนี้เนื่องจาก SIO ไม่มีบัสแอดเดรสที่จะใช้เป็น Index Internal Register

ในแชนแนล B จะมีกลุ่ม Read Register (Label RR0-RR2) ซึ่งจะเป็นที่เก็บสถานะของ SIO อยู่ โดยในแชนแนล A ไม่มี RR1 อยู่

4.2.4 ขั้นตอนทั่วไปในการ INITIALIZE สำหรับ SIO

ในหัวข้อนี้จะกล่าวถึงวิธีการโปรแกรมใช้งานและการเชื่อมต่อ SIO ในการทำงานแบบ Serial Poll

SIO สามารถส่งหรือรับข้อมูลครั้งละ 5,6,7 หรือ 8 บิตต่อคาแรคเตอร์ ซึ่งสามารถโปรแกรมจำนวนบิตได้ทั้งทางด้านส่งและรับ

ในกรณีการส่งข้อมูลอนุกรมแบบ Asynchronous SIO จะสร้าง Start Bit ซึ่งเป็นลอจิก "0" และสร้าง Stop Bit ขนาด 1,3/2 หรือ 2 บิต ซึ่งเราสามารถโปรแกรมขนาดของ Stop Bit ในแต่ละคาแรคเตอร์ที่ส่งออกมาได้

Stop Bit มีลอจิกเป็น "1" ซึ่งเป็นระดับลอจิกของสภาวะ Marking ก่อนที่จะมีการเพิ่ม Stop Bit เข้าไปในแต่ละคาแรคเตอร์ อาจจะมีการเพิ่ม Parity Bit ก่อน ซึ่งพาริตี้นี้อาจจะเป็นพาริตี้คู่หรือคี่ก็ได้

เนื่องจาก CPU รับข้อมูลเข้ามาทีละ 8 บิตแต่คาแรคเตอร์หนึ่งๆอาจจะใช้ไม่ถึง 8 บิต เช่น ในกรณีที่คาแรคเตอร์มีเพียง 6 บิต D0-D5 จะเป็นข้อมูลจริง (DOจะเป็น LSB ; Least Significant Bit) ส่วนบิตที่เหลือจะเป็น Stop Bit และ Parity Bit จำนวน 1 บิต หมายความว่า คาแรคเตอร์ที่รับเข้ามามี

จำนวน 5 บิต +1 Stop Bit ,2 บิตที่เหลือจะถูกกำหนดโดย SIO ให้เป็น Marking Level หรือลอจิก "1" โดยอัตโนมัติ

สัญญาณคล็อกที่ป้อนให้กับ SIO เพื่อนำไปใช้ในการควบคุมอัตราการรับส่งข้อมูล จะมี 4 สัญญาณด้วยกัน ทั้ง 4 สัญญาณนี้จะเกี่ยวข้องกับขา RxA,RxB,TxA,TxBซึ่งคล็อกที่ป้อนให้กับ SIO นี้จะมีความถี่เป็น 1,16,32 หรือ 64 เท่าของอัตราการส่งหรือรับข้อมูล

เราสามารถสรุปขั้นตอนเมื่อใช้ SIO ใน Asynchronous Mode ได้ดังนี้

1. เขียนข้อมูลให้แก่ WR0 เพื่อเป็นการรีเซ็ต SIO ส่วนมากในการเขียนคำสั่งการรีเซ็ตจะใช้ความยาวมากกว่า 1 ไบต์ ซึ่งใช้ในการเขียนคำสั่งพิเศษ

2. เขียนข้อมูลให้แก่ WR1 เพื่อทำการเซ็ต Baud Rate Fator (ตัวหารความถี่) ที่ใช้ในการส่งและรับข้อมูล,ขนาด ของ Stop Bit และชนิดของ Parity Bit

3. เขียนข้อมูลให้แก่ WR3 เพื่อทำการเซ็ตจำนวนบิตของข้อมูลที่จะรับ,ทำ Auto Enable และ Receiver Enable

4.เขียนข้อมูลให้แก่ WR5 เพื่อทำการเซ็ตจำนวนบิตของข้อมูลที่จะส่ง และทำ Transmitter Enable

5. เขียนข้อมูลลงใน WR2 ซึ่งใช้สำหรับแชนแนล B เท่านั้นใช้ในการกำหนดอินเตอร์รัพท์เวคเตอร์

6. เขียนข้อมูลลงใน WR1 เพื่อทำการ Enable หรือ Disable การขออินเตอร์รัพท์ของ SIO

ในกระบบการส่งข้อมูลแบบ Asynchronous นั้นเราไม่จำเป็นต้องใช้ WR6,WR7 แต่สำหรับในการส่งข้อมูลแบบ Synchronous จะต้องมีการป้อนข้อมูลให้แก่รีจิสเตอร์คู่นี้ด้วย

4.2.4 ขบวนการอินเทอร์รัพท์ใน SIO

ในส่วนที่จะกล่าวถึงต่อไปนี้ จะเป็นการนำเอา SIO ไปเกี่ยวข้องกับการอินเทอร์รัพท์ในขั้นแรกจะมาดูโครงสร้างการอินเทอร์รัพท์ใน SIO และจะแสดงตัวอย่างการ INITIALIZE SIO, และการรับ/ส่งข้อมูลในโหมดของการอินเทอร์รัพท์

ใน SIOรีจิสเตอร์ WR2 จะใช้เป็นที่เก็บอินเทอร์รัพท์เวคเตอร์ ซึ่งเวคเตอร์ตัวนี้จะถูกส่งให้แก่ CPU เมื่อ SIO ให้รับสัญญาณตอบรับการอินเทอร์รัพท์ (Interrupt Acknowledge)จาก Z80 CPU

SIO สามารถให้อินเตอร์รัพท์เวคเตอร์ได้ต่างๆกัน 8 ตัว ซึ่งอินเทอร์รัพท์เวคเตอร์ทั้ง 8 ตัวนี้จะถูกสร้างมาจากเวคเตอร์เพียงตัวเดียวที่ถูกส่งไปให้กับ SIO ใน Initialization Phase (การส่งอินเทอร์รัพท์เวคเตอร์ให้กับ WR2 ในตอนแรกสุด) เพื่อที่จะให้ทั้ง 8 ตัวนี้ถูกสร้างขึ้น , บิต D2 ของรีจิสเตอร์ WR1 (Status Effects Vector) จะต้องถูกเซ็ตให้เป็นลอจิก "1"

เมื่อบิตถูกเซ็ตและยอมให้มีการขออินเทอร์รัพท์ได้แล้ว ก็จะกำหนดชนิดของการอินเทอร์รัพท์จากนั้นจะมีการเปลี่ยนแปลงข้อมูลของบิตต่างๆกันถึง 8 แบบ ซึ่งจะแบ่งออกให้เป็น 2 กลุ่มด้วยกัน โดยแต่ละกลุ่มจะใช้ในแชนแนลต่างกัน เพราะฉะนั้นจึงสามารถที่จะให้รายละเอียดของการอินเทอร์รัพท์แก่ทั้งแชนแนล A และB ซึ่งลำดับการอินเทอร์รัพท์จะถูกจัดได้ภายในตัว SIO โดยอัตโนมัติ

แสดงรายละเอียดของบิตต่างๆ จะก่อให้เกิดอินเทอร์รัพท์เวคเตอร์ต่างๆกัน 8 ชนิด และจัดลำดับความสำคัญจากสูงไปต่ำ

เงื่อนไขของการอินเทอร์รัพท์ ซึ่งประกอบด้วย Parity Error , Receive Overrun,Framing Error หรือ End Frame SDLC ซึ่งเรียกว่า "เงื่อนไขพิเศษในการรับข้อมูล (Special Receive Condition)"

4.2.5 การเริ่มต้นให้ SIO ทำการอินเทอร์รัพท์

ลักษณะการ Initialize SIO เมื่อให้ SIO อยู่ในโหมดการอินเทอร์รัพท์จะมีลักษณะเหมือนกับ Polling Mode แต่แตกต่างกันตรงที่ใน Polling Mode ไม่ต้องส่งอินเทอร์รัพท์เวคเตอร์ให้แก่ WR2 และต้องเขียน WR1 ให้ทำการ Disable การขออินเทอร์รัพท์

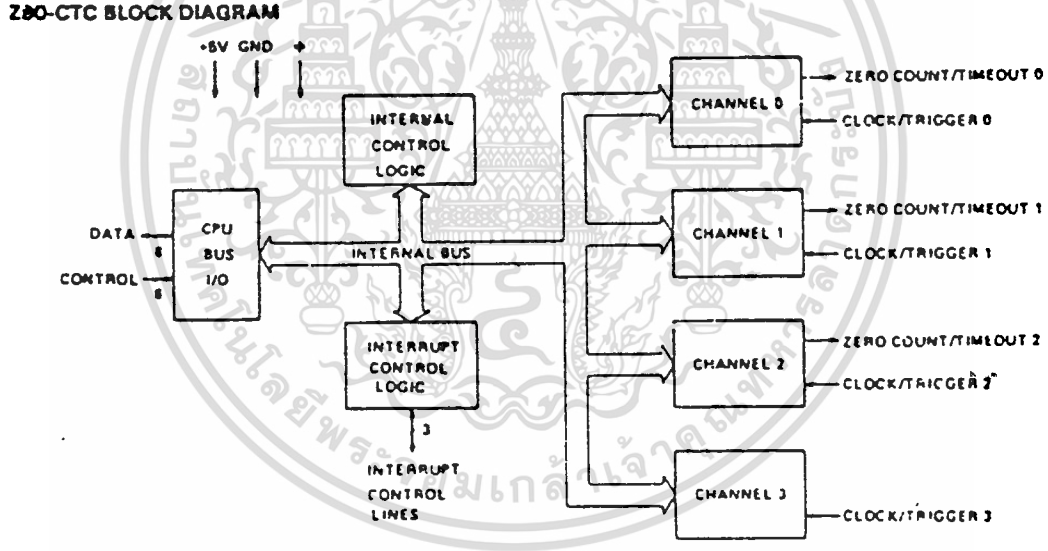
หลังจากที่ SIO ได้รับการ Initialize เพียงครั้งหนึ่งแล้ว SIO ก็พร้อมที่จะใช้งานในโหมดการอินเทอร์รัพท์ ซึ่งอินเทอร์รัพท์เซอวิสรูทีน (Interrupt Service Routine)จะต้องสอดคล้องกับชนิดของการอินเทอร์รัพท์ เช่น เมื่อเกิดการอินเทอร์รัพท์ขึ้นจากเงื่อนไขที่ตัวส่งว่างโปรแกรม Subroutine "Reset TX Interrupt Pending"จะถูก Execute คาแร็คเตอร์ตัวต่อไปจะถูกส่งให้แก่ SIO

4.3 การใช้งาน Z80-CTC

4.3.1 บล็อกไดอะแกรมของ CTC

CTC เป็นชื่อที่มาจากที่มาจากลักษณะการทำงานพื้นฐาน 2 ประการของชิพคือ การนับและการเป็นฐานเวลา (Counter/Timer) โดยภายในจะมีเคาน์เตอร์และไทม์เมอร์ที่เป็นอิสระต่อกันอยู่ 4 แชนแนล

จากรูปจะเห็นว่า CTC แบ่งออกเป็น 4 แชนอนที่เป็นอิสระต่อกัน เรียกว่า Ch0,Ch1,Ch2 และ Ch3 ซึ่งใน 3 แชนแนลแรก (Ch0,Ch1 และ Ch2) นั้นมีเส้นสัญญาณที่ใช้สำหรับเชื่อมต่อ (Interface)กับระบบภายนอกอยู่ 2 เส้นเรียกว่า ZERO COUNT/TIME OUT (เป็นเอาท์พุท) และ CLOCK/TRIGGER (เป็นอินพุท) ส่วนใน Ch3 มีเพียงCLOCK/TRIGGER เพียงเส้นเดียว



รูปที่ 4.7 บล็อกไดอะแกรมของ Z80-CTC

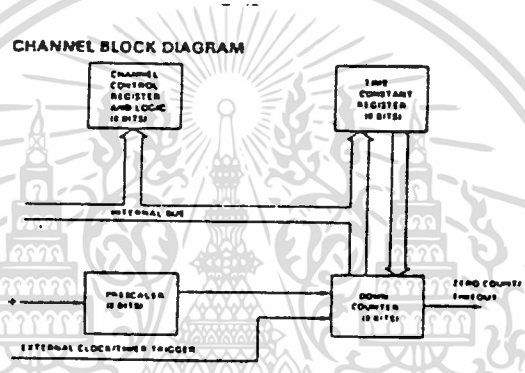
บล็อกที่มีชื่อว่า IBTERNAL CONTROL LOGIC ทำหน้าที่ควบคุมการส่งผ่านข้อมูลบนบัสภายใน CTC ให้ถูกต้อง

บล็อกของ INTERRUPT CONTROL LOGIC ทำหน้าที่ควบคุมการอินเตอร์รัพท์ ซึ่งการอินเทอร์รัพท์ของ CTC นี้จัดว่ามีความสำคัญมาก

บล็อกสุดท้าย คือ CPU BUS I/O ซึ่งใช้ในการเชื่อมต่อระหว่าง Z80 กับ CTC ประกอบด้วย บัสข้อมูล 8 เส้นและบัสควบคุม 6 เส้น การติดต่อระหว่า Z80 กับ CTC จะต้องผ่านบล็อกนี้ ดังนั้น บล็อกนี้สามารถที่จะถูกโปรแกรมให้ทำตามจุดประสงค์ของผู้โปรแกรมได้

4.3.2 รายละเอียดของแต่ละ CHANNEL BLOCK

เป็นการแสดงบล็อกไดอะแกรมภายในของแต่ละแชนแนล ซึ่งประกอบไปด้วย CHANNEL CONTROL REGISTER, TIME CONSTANT DOWN และ COUNTER



รูปที่ 4.8 บล็อกไดอะแกรมแสดงโครงสร้างภายในของเคาน์เตอร์

บล็อก CHANNEL CONTROL REGISTER เป็นรีจิสเตอร์ที่ผู้โปรแกรมใช้เขียนข้อมูลเข้าไปเพื่อกำหนดลักษณะการทำงานของแชนแนล

บล็อก TIME CONSTANT REGISTER เป็นรีจิสเตอร์ที่มีขนาด 8 บิต ซึ่งมีค่าได้ตั้งแต่ 00H ไปจนถึง 0FFH ค่าที่อยู่ในรีจิสเตอร์นี้จะนำไปใช้สำหรับเซ็ตค่าที่ใช้เริ่มต้นการนับจำนวนคล็อกของ DOWN COUNTER

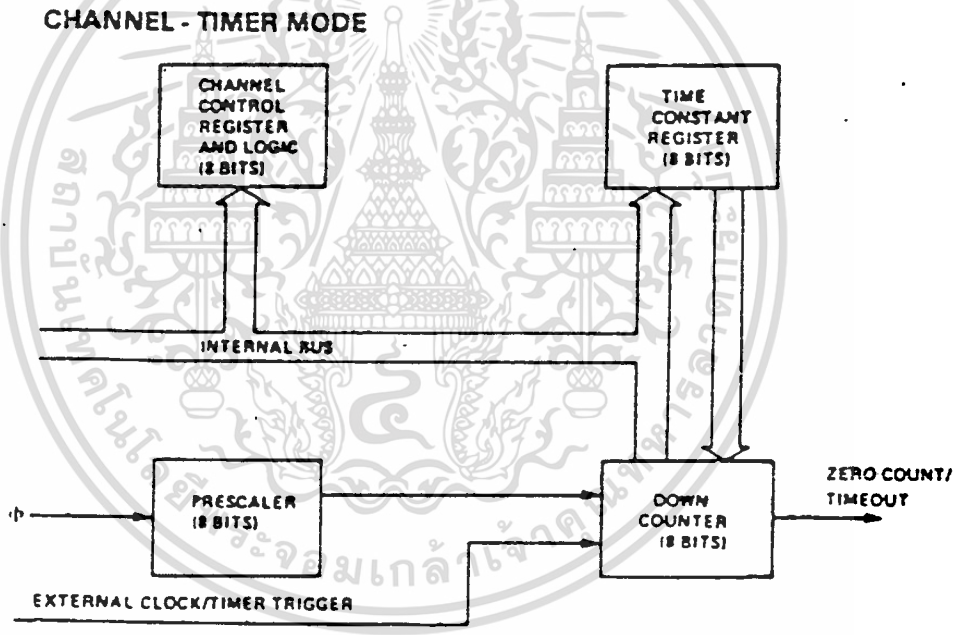
บล็อก DOWN COUNTER เป็นเคาน์เตอร์ที่มีขนาด 8 บิต มีลักษณะการนับแบบลง ที่บล็อกนี้มีเส้นสัญญาณอินพุทชื่อว่า EXTERNAL CLOCK/TIMER TRIGGER สัญญาณนี้เป็นสัญญาณคล็อกที่จะป้อนให้กับ DOWN COUNTER โดยตรง หรือ อาจจะใช้เป็นสัญญาณ ENABLE สำหรับให้ PRESCALER ทำงานก็ได้ขึ้นอยู่กับการโปรแกรม CTC ส่วนเส้นสัญญาณ

ZERO COUNT/TIMER OUT ใช้เป็นเอาท์พุทที่จะแอคทีฟเมื่อค่าที่นับในเคาน์เตอร์มีค่าเป็น 00H

บล็อกซึ่งอยู่หน้า DOWN COUNTER มีชื่อว่า PRESCALAR มีขนาด 8 บิต หน้าที่ของบล็อกนี้ จะใช้สำหรับเป็นตัวหารเริ่มแรกให้กับสัญญาณคล็อกที่จะป้อนให้กับ DOWN COUNTER โดยสามารถเลือกได้ 2 ค่าคือ 16 หรือ 256 ซึ่งขึ้นกับการโปรแกรมของผู้ใช้

4.3.3 ตัวอย่างการใช้งาน CTC ในโหมดของไทม์เมอร์

ในหัวข้อนี้จะเป็นการแสดงให้เห็นว่า จะใช้ CTC ในโหมดไทม์เมอร์ได้อย่างไรจากบล็อกไดอะแกรมของ CTC ในโหมดไทม์เมอร์ที่แสดงในรูปที่ เราจะทำการเซ็ทให้ CTC เป็นเสมือนคล็อกตัวหนึ่งที่สามารถให้เอาท์พุทเป็นคล็อกพัลส์ (CLOCK PULSE) ที่มีความถี่เป็น 2400 Hz ซึ่งการใช้งานในลักษณะนี้นิยมใช้เป็น BOUD RATE GENERATER หรือไทม์เมอร์ให้กับระบบใด ๆ โดยในตัวอย่างนี้ เราจะสมมติว่าความถี่ของคล็อกของระบบที่ป้อนเข้ามามีค่าเท่ากับ 1 Mhz



รูปที่ 4.9 บล็อกไดอะแกรมในการใช้ Z80-CTC ใน TIMER MODE

ที่กล่าวมานี้หมายความว่าเราจะต้องหาตัวเลขจำนวนหนึ่ง ที่จะใช้สำหรับเป็นตัวหารความถี่ของคล็อกที่ป้อนเข้ามาให้มีความถี่เป็น 2400 Hz ฉะนั้นคาบของคล็อกที่มีความถี่ 2400 Hz จะมีค่าเท่ากับ 416.7 nanoseconds (หรือเพื่อให้ง่ายจึงใช้เป็น 417 nanoseconds) คาบของสัญญาณคล็อกที่ป้อนเข้าไปจะมีค่าเท่ากับ 1 microseconds จากบล็อกไดอะแกรมรูป เป็นการบอกถึงลักษณะการทำงานของ CTC ในขณะที่อยู่ใน TIMER MODE

จาก PRESCALER ของ CTC สามารถกำหนดค่าตัวหารความถี่ได้เป็น 16 หรือ 256 เช่น เราหาร 417 ด้วย 16 จะได้เท่ากับ 26.06 ปัดให้เป็นเลขจำนวนเต็มได้เท่ากับ 26 หมายความว่า เราจะต้องใช้ค่า TIME CONSTANT เท่ากับ 26 ฉะนั้นเราจะได้ตัวหารทั้งหมดเป็น 16 x 26 = 416 ซึ่งจะได้ว่าทุก ๆ 416 microseconds เอาท์พุทของ ZC/TO จะได้ลอจิกเป็น "1" และกลับเป็น "0" (คือพัลส์ 1 ลูก) ซึ่งการทำเช่นนี้ทำให้เราได้ความถี่ที่มีใกล้เคียงกับ 2400 Hz มากและรายละเอียดทั้งหมดนี้แสดงในรูป 4.10

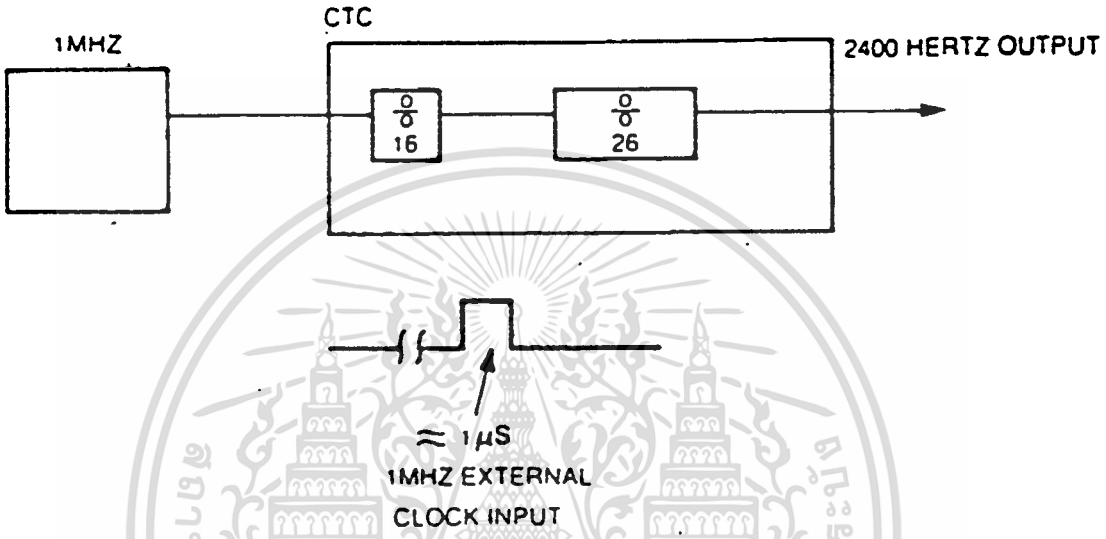
ต่อไปนี้เราจะมาดูวิธีการเซ็ต CTC สำหรับใช้ใน TIMER MODE สำหรับตัวอย่างนี้ เริ่มแรกเราจะต้องเซ็ต CONTROL WORD REGISTER ก่อน โดยเราจะใช้แชนแนล 2 ค่าของบิทใน CONTROL WORD จะถูกเซ็ตดังต่อไปนี้

บิท D7 = 0 : ไม่มีการอินเทอร์รัพท์
บิท D6 = 0 : เป็นโหมด TIMER
บิท D5 = 0 : ตั้ง PRESCALE FACTOR = 16
บิท D4 = 0 : ไม่สนใจ เพราะเราไม่ได้ใช้เป็น TRIGGER
บิท D3 = 0 : ให้เริ่มต้นทำงานเมื่อ TIME CONSTANT ถูกโหลด
บิท D2 = 1 : จะมีค่า TIME CONSTANT ตามหลัง CONTROL WORD
บิท D1 = 1 : แชนแนลจะถูกรีเซ็ท
บิท D0 = 1 : เป็นบิทที่ใช้กำหนดว่าข้อมูลทั้งหมดนี้เป็น CONTROL WORD

สมมติค่า TIME CONSTANT มีค่าเป็น 26 (ฐานสิบ) หรือ 1AH ตามวิธีข้างต้นเราสามารถให้ CTC ทำงานได้ตามต้องการ ดังโปรแกรมข้างล่างนี้

```
LD A, 07H
OUT (042H) ,A  : เอาท์พุท CONTROL WORD  ไปยัง CH2
LD A, 1AH
OUT (042H) ,A  : เซ็ทค่า TIME CONSTANT , เริ่มต้นทำงาน
```

ฉะนั้นขณะนี้ CTC จะทำงานในโหมดไทม์ที่แชนแนล 2 ด้วยความถี่ประมาณ 2400 เฮริตซ์ สำหรับการต่อทางฮาร์ดแวร์สามารถทำได้ดังรูป 4.10



รูปที่ 4.10 บล็อกไดอะแกรมแสดงการใช้งาน CTC ในการสร้าง BOAD RATE ถ้าความถี่เป็น 2400 Hz จะต้องป้อนความถี่ที่อินพุท 1 MHz

บทที่ 5

ไมโครคอนโทรลเลอร์ MCS-51

5.1 คุณสมบัติของ MCS-51

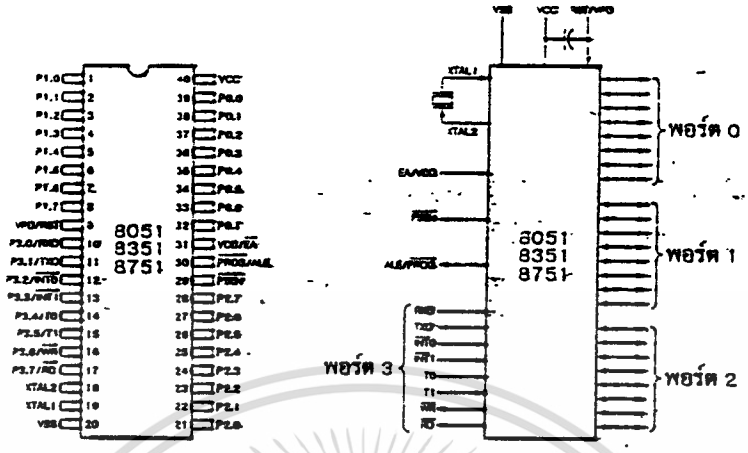
คุณสมบัติที่สำคัญๆของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีดังนี้

- ต้องการแหล่งจ่ายไฟ 5 โวลต์ เพียงชุดเดียว
- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงาน (ROM) อยู่ภายในชิป
- มีหน่วยความจำสำหรับเก็บข้อมูลทั่วไป (RAM) อยู่ภายในชิปจำนวน 128 ไบต์
- สามารถใช้หน่วยความจำสำหรับโปรแกรมและข้อมูลที่อยู่ภายนอกชิปได้อย่างละ 64 กิโลไบต์ แยกจากกัน
- คำสั่งส่วนใหญ่ใช้เวลาทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกกะเฮิรตซ์
- มีพอร์ตที่สามารถรับหรือส่งข้อมูลได้ทั้ง 2 ทิศทาง จำนวน 4 พอร์ตๆละ 8 บิต หรือสามารถใช้ งานเป็นพอร์ตขนาด 1 บิต แยกจากกันทำให้เสมือนมีพอร์ตขนาด 1 บิต ใช้งานรวมทั้งสิ้น 32 พอร์ต
- รับและส่งข้อมูลแบบอนุกรมได้ในตัว โดยสามารถกำหนดอัตราเร็วในการรับและส่งข้อมูล (baud rate) ได้ตั้งแต่ 300-375 กิโลบิต ต่อ วินาที
- จัดลำดับความสำคัญของสัญญาณอินเตอร์รัปต์ได้ 2 ระดับ
- มีรีจิสเตอร์สำหรับใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์เพื่อนับจำนวนสัญญาณนาฬิกาภายในชิป หรือนับการเปลี่ยนสถานะของสัญญาณภายนอกขนาด 16 บิต จำนวน 2 ตัว เพื่อใช้สำหรับนับจำนวนพัลส์ วัดความกว้างของพัลส์หรือใช้วัดช่วงเวลา
- หน่วยความจำสำหรับเก็บข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลได้ทั้งระดับไบต์และระดับบิตเพื่อให้การออกแบบโปรแกรมและการควบคุมระบบทำได้ง่ายขึ้น
- มีคำสั่งคูณและหารเลขขนาด 8 บิต ในตัวเอง
- สามารถประมวลผลแบบบูลีนเพื่อใช้ในงานควบคุมโดยเฉพาะ
- ใช้โปรแกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-48 ได้

5.2 โครงสร้างของไมโครคอนโทรลเลอร์ตระกูล MCS-51

5.2.1 ตำแหน่งขาของ MCS-51

MCS-51 ทุกเบอร์จะมีขาพื้นฐานเหมือนกันดังแสดงในรูปที่ 5.1



รูปที่ 5.1 แสดงตำแหน่งขาของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51

หน้าที่การใช้งานแต่ละขาของ MCS-51 มีดังนี้

- **ขา Vss** (ขา 20) สำหรับต่อลงกราวด์

- **ขา Vcc** (ขา 40) สำหรับต่อแหล่งจ่ายแรงดันกระแสตรง 5 โวลต์

- **ขาพอร์ต 0** (ขา 32-39) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7)พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ high -impedance)โดยมีวงจรพูลอัพภายใน นอกจากนั้นยังใช้พอร์ต 0 ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลภายนอกชิป โดยการส่งค่าแอดเดรสไบต์ต่ำ (A0-A7) และรับส่งข้อมูล (D0-D7) จากหน่วยความจำภายนอก

- **ขาพอร์ต 1** (ขา 1-8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 1 ขนาด 8 บิต (P1.0-P1.7). พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากใช้งานเป็นอินพุตพอร์ตต้อง

โหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ high -impedance) โดยมีวงจรพูลอัพภายใน

- **ขาพอร์ต** 2 (ขา 21-28) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 2 ขนาด 8 บิต (P2.0-P2.7) พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ high - impedance) โดยมีวงจรพูลอัพภายใน นอกจากใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปแล้ว พอร์ต 2 ยังใช้ในการติดต่อ ROM และ RAM ภายนอกด้วย โดยใช้สำหรับส่งค่าแอดเดรสไบต์สูง (A8-15)

- **ขาพอร์ต** 3 (ขา 10-17) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 3 ขนาด 8 บิต (P3.0-P3.7) พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ high - impedance) โดยมีวงจรพูลอัพภายใน นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆดังนี้

ขา P3.0 ใช้รับข้อมูลจากภายนอกแบบอนุกรม

ขา P3.1 ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม

ขา P3.2 ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิดที่ 0

ขา P3.3 ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิดที่ 1

ขา P3.4 ใช้เป็นสัญญาณอินพุตให้เคาต์เตอร์ของไทม์เมอร์ 0

ขา P3.5 ใช้เป็นสัญญาณอินพุตให้เคาต์เตอร์ของไทม์เมอร์ 1

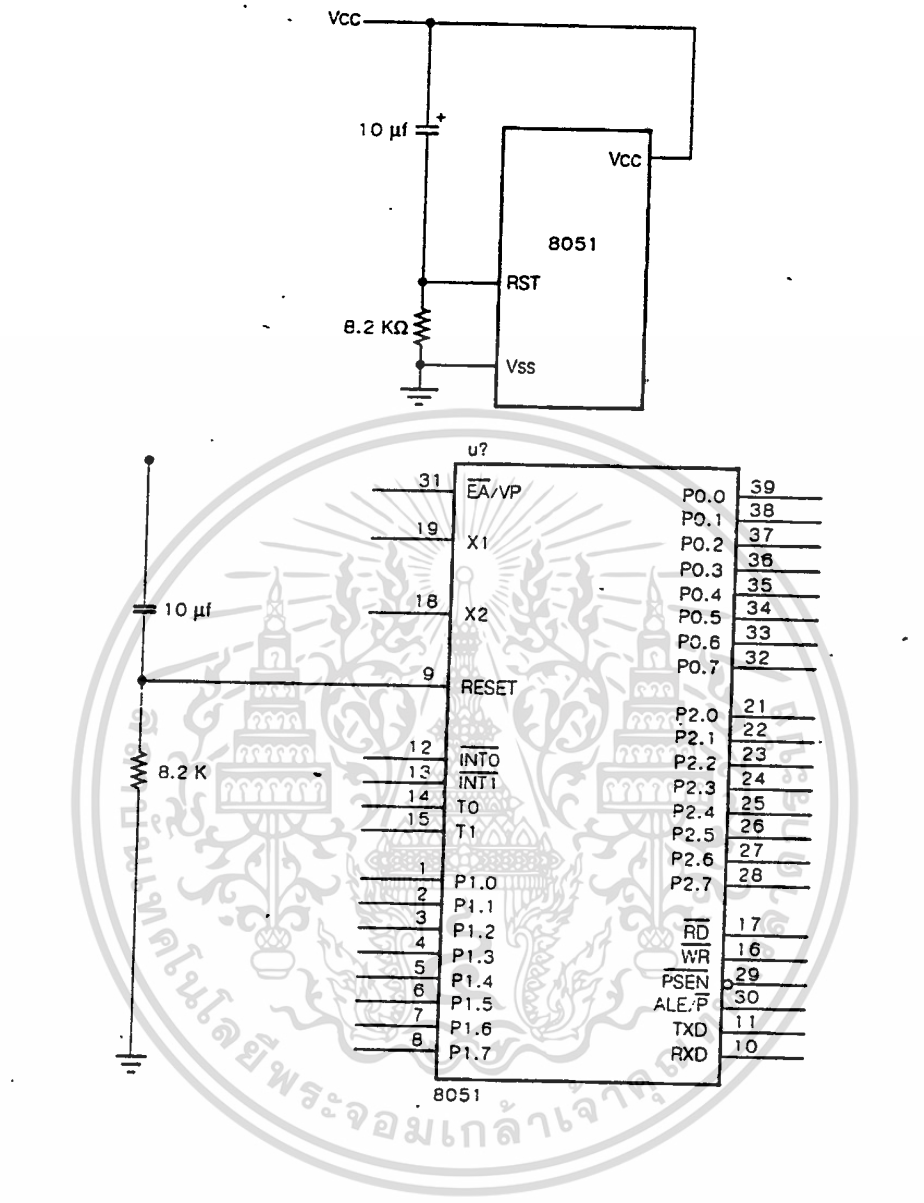
ขา P3.6 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลสำหรับเก็บข้อมูลภายนอกชิป

ขา P3.7 ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลสำหรับเก็บข้อมูลภายนอกชิป

การใช้งานพอร์ต 3 ในหน้าที่พิเศษนี้จะต้องโหลดค่า 1 ไปยังแต่ละบิตที่ต้องการใช้ก่อนทุกครั้ง

- **ขา RST** (ขา 9) ใช้สำหรับการรีเซตวงจรทุกอย่างภายในชิปเพื่อเริ่มต้นทำงานใหม่ โดยมีการต่อวงจรดังรูปที่ 5.2

- **ขา ALE / PROG** (ขา 30) ใช้สำหรับส่งสัญญาณออกไปภายนอกเพื่อควบคุมการแลตช์ค่าแอดเดรสไบต์ต่ำ (Address Latch Enable) จากพอร์ต 0 ในระหว่างการติดต่อ ROM หรือ RAM ภายนอก



รูปที่ 5.2 แสดงวงจรสำหรับรีเซตชิปไมโครคอนโทรเลอร์ MCS-51 เมื่อเริ่มจ่ายพลังงานโดยอัตโนมัติ

- **ขา PSEN** (ขา 29) ใช้ส่งสัญญาณสโตรบเพื่ออ่านคำสั่งจากโปรแกรมที่เก็บไว้ใน ROM ภายนอก

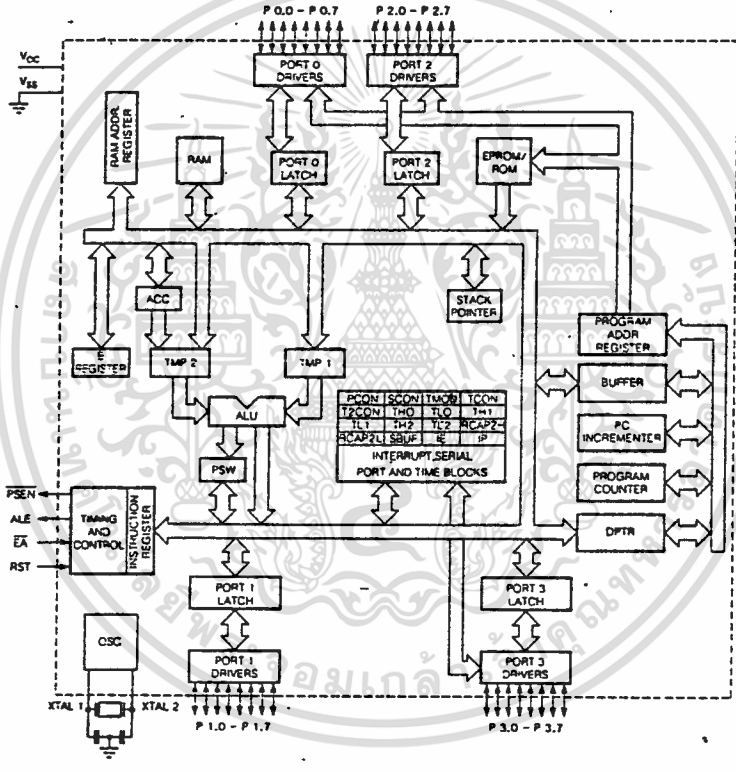
- **ขา EA / Vpp** (ขา 31) เป็นขาสำหรับใช้เลือกให้ MCS-51 ทำงานจากโปรแกรมที่อยู่ภายในหรือภายนอกชิป โดยหากขานี้มีสถานะเป็น 0 ให้ใช้โปรแกรมจาก ROM ภายนอก ถ้าขานี้มี

สถานะเป็น 1 หมายถึงบังคับให้ MCS-51 ใช้โปรแกรมจาก RAM ภายในชิป ส่วนใน MCS-51 ที่ไม่มี ROM ภายในให้ต่อขานี้ลงกราวด์เสมอ

- **ขา XTAL 1** (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุทเข้าสู่วงจรออสซิลเลเตอร์
- **ขา XTAL 2 (ขา 18)** ใช้ต่อคริสตอลภายนอกโดยเป็นเอาท์พุทออกจากวงจรออสซิลเลเตอร์

5.2.2 โครงสร้างภายในของ MCS-51

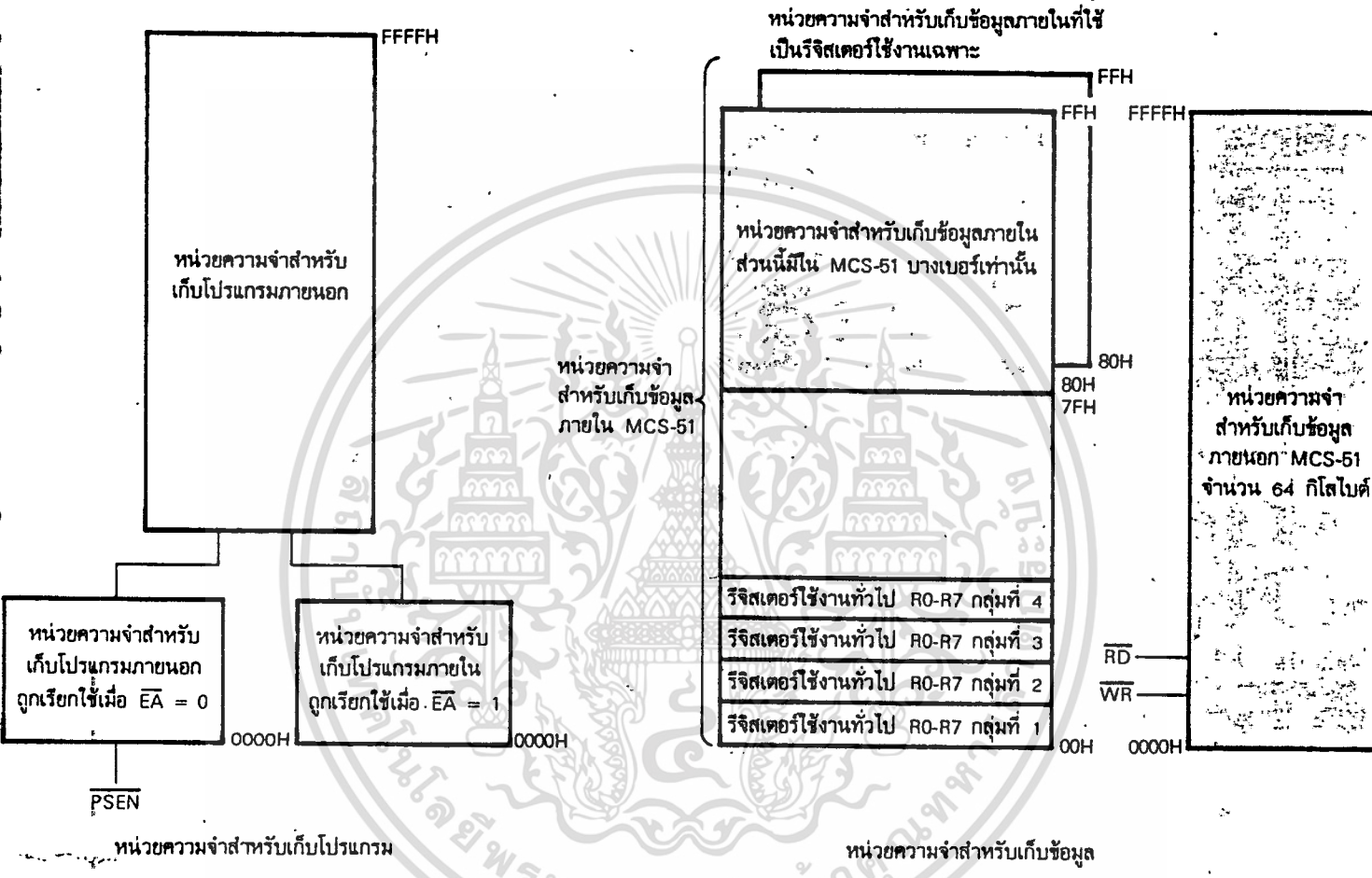
โครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังในรูปที่ 5.3



รูปที่ 5.3 แสดงโครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ MCS-51

5.2.3 โครงสร้างหน่วยความจำภายในของ MCS-51

โครงสร้างหน่วยความจำทั้งหมดของ MCS-51 แสดงดังในรูปที่ 5.4



รูปที่ 5.4 แสดงโครงสร้างหน่วยความจำทั้งหมดของ MCS-51

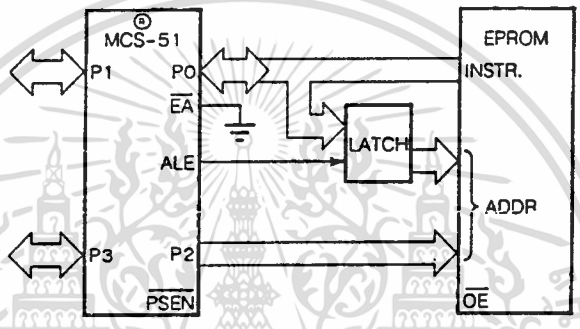
ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะแบ่งหน่วยความจำออกเป็น 2 ส่วน คือ

- **หน่วยความจำสำหรับเก็บโปรแกรม (program memory)**

หน่วยความจำสำหรับเก็บโปรแกรมจะใช้เก็บโปรแกรมควบคุมการทำงานของชิป MCS-51 บางเบอร์จะมีหน่วยความจำนี้อยู่ภายในชิป (internal program memory) ซึ่งมีขนาดได้ตั้งแต่

0 , 4 , 8 , 16 กิโลไบต์ แต่บางเบอร์จะไม่มี ทำให้ต้องเก็บโปรแกรมไว้ในหน่วยความจำภายนอก (external program memory) ทั้งหมด

สัญญาณควบคุมการเฟตช์คำสั่งกับโปรแกรมที่เก็บไว้ในหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป(read strobe) คือสัญญาณ PSEN (Program Strobe Enable) จากขา 29 ซึ่งจะนำไปใช้ต่อกับขา RD ของหน่วยความจำสำหรับเก็บโปรแกรมภายนอกแต่สัญญาณ PSEN จะไม่ถูกใช้งาน เมื่อ MCS-51 ทำงานจากโปรแกรมซึ่งอยู่ในหน่วยความจำสำหรับเก็บโปรแกรมที่อยู่ภายในชิป การใช้โปรแกรมจากหน่วยความจำภายนอกชิปแสดงดังในรูปที่ 5.5



รูปที่ 5.5 แสดงการใช้หน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป

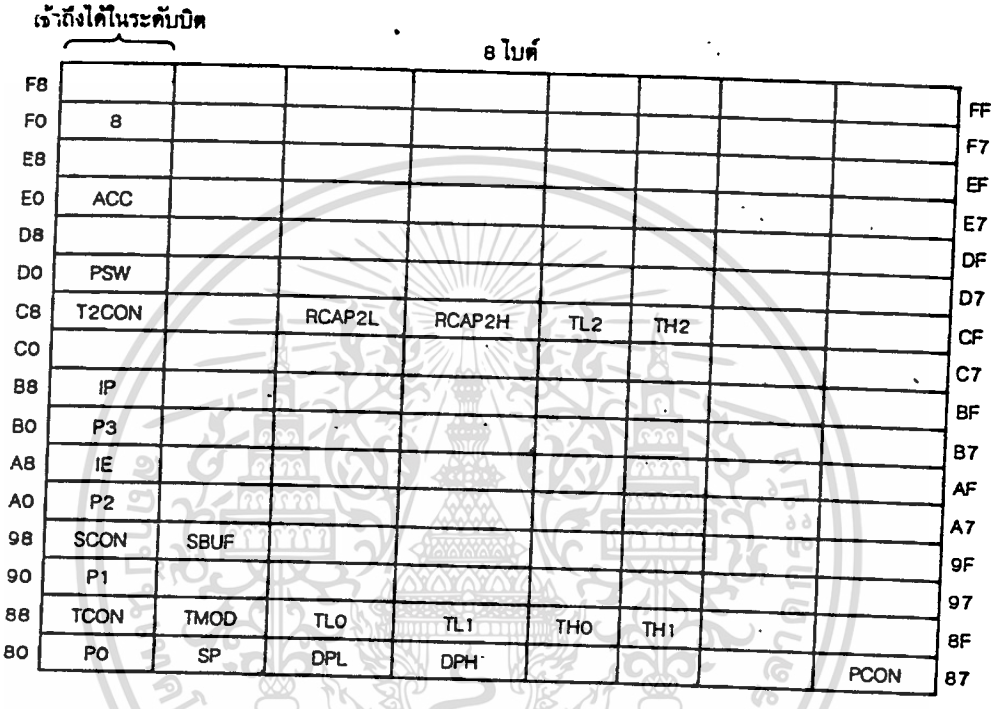
- หน่วยความจำสำหรับเก็บข้อมูล (data memory)

หน่วยความจำสำหรับเก็บข้อมูล ซึ่งใช้สำหรับเก็บข้อมูลระหว่างการทำงาน MCS-51 ทุกเบอร์จะมีหน่วยความจำนี้อยู่ภายในชิปจำนวนหนึ่ง แต่จะมากหรือน้อยเท่าไรขึ้นอยู่กับเบอร์ของชิป

แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิปแสดงดังในรูปที่ 5.4 โดยแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิป และหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป หน่วยความจำสำหรับเก็บข้อมูลภายในชิปยังแบ่งออกได้เป็น 2 ส่วนย่อย คือ

- ส่วนที่ใช้เก็บข้อมูลทั่วไป (internal ram) อยู่ภายในชิปบริเวณ 128 ไบต์ แรก
- ส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (special function register : SFR) อยู่ภายในชิป

บริเวณ 128 ไบต์ หลังซึ่งเป็นส่วนที่มีเพิ่มขึ้นมาใน MCS-51 บางเบอร์ดังแสดงในรูปที่ 5.6

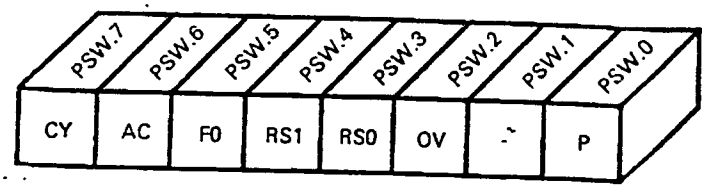


รูปที่ 5.6 แสดงโครงสร้างและตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะใน MCS-51

5.2.4 รีจิสเตอร์ที่ใช้ควบคุม MCS-51

1) รีจิสเตอร์ที่ใช้งานเฉพาะ PSW (Program Status Word)

เข้าถึงข้อมูลได้ในระดับบิตดังแสดงในรูปที่ 5.7



รูปที่ 5.7 รีจิสเตอร์ใช้งานเฉพาะ PSW

PSW.7 CY carry flag

PSW.6 AC auxillary carry flag

PSW.5 F0 flag 0 เป็นบิตบอกสถานะที่ผู้ใช้สามารถกำหนดการใช้งานเองได้

PSW.4 RS1 ใช้เลือกกลุ่มรีจิสเตอร์ R0-R7

PSW.3 RS0 ใช้เลือกกลุ่มรีจิสเตอร์ R0-R7

0 0 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 0

0 1 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 1

1 0 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 2

1 1 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 3

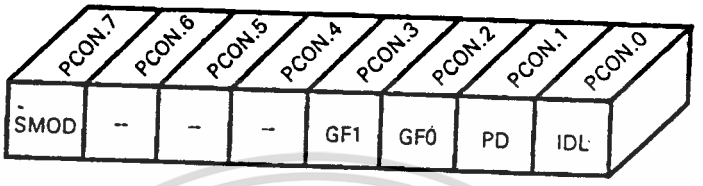
PSW.2 OV บิตแสดงการเกิด over flow

PSW.1 - บิตที่ผู้ใช้กำหนดการใช้งานเองได้

PSW.0 P parity flag ถูกเซตหรือเคลียร์โดยวงจรภายใน MCS-51 ในแต่ละไซเกิลของคำสั่งแต่ละคำสั่ง ใช้เป็นตัวบอกให้ทราบว่าในรีจิสเตอร์ ACC มีข้อมูลที่เป็น 1 อยู่เป็นจำนวนคู่หรือคี่

2) รีจิสเตอร์ใช้งานเฉพาะ PCON (Power Control Register)

ไม่สามารถเข้าถึงข้อมูลในระดับบิตได้ ดังแผนภาพที่แสดงในรูปที่ 5.8



รูปที่ 5.8 รีจิสเตอร์ใช้งานเฉพาะ PCON

PCON.7 SMOD เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate หากบิตนี้มีค่าเป็น 1 ในการใช้งานพอร์ตสื่อสารอนุกรมโหมด 1,2 และ 3 ค่า baud rateจะพิ่มขึ้นเป็น 2 เท่า

PCON.6 - ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ๆในอนาคต)

PCON.5 - ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ๆในอนาคต)

PCON.4 - ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ๆในอนาคต)

PCON.3 GF1 บิตที่ผู้ใช้สามารถกำหนดการใช้งานได้เอง

PCON.2 GF0 บิตที่ผู้ใช้สามารถกำหนดการใช้งานได้เอง

PCON.1 PO บิตควบคุมการใช้พลังงานแบบ Power Down Mode

1 : MCS-51 จะอยู่ในสภาวะ Power Down Mode (หยุดทำงานจนกว่าจะมีการรีเซต) ใช้ได้เฉพาะ MCS-51 ที่ผลิตโดยใช้เทคโนโลยี CHMOS เท่านั้น

0 : MCS-51 จะอยู่ในสภาวะการทำงานปกติ

PCON.0 IDL บิตควบคุมการทำงานแบบพลังงานต่ำแบบ idle mode

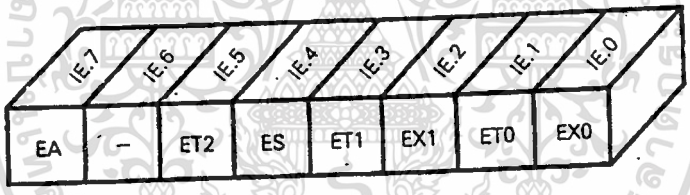
1 : MCS-51 อยู่ในสภาวะ idle mode คือหยุดรอการทำงานชั่วคราวจนกว่าจะมีสัญญาณอินเตอร์รัปต์ ในระหว่างที่รอสัญญาณเตอร์รัปต์ MCS-51 จะใช้พลังงานน้อยมาก

0 : MCS-51 อยู่ในสภาวะการทำงานปกติ (บิต PD ต้องเป็น 1)

5.2.5 รีจิสเตอร์สำหรับใช้ควบคุมการทำงานเกี่ยวกับอินเตอร์รัปต์

1) รีจิสเตอร์ใช้งานเฉพาะ IE (Interrupt Enable-Register)

เข้าถึงข้อมูลได้ในระดับบิตดังแสดงในรูป 5.9



รูปที่ 5.9 รีจิสเตอร์ใช้งานเฉพาะ IE

IE.7 EA ใช้ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ทั้งหมด

0 : MCS-51 จะไม่ตอบสนองต่อสัญญาณอินเตอร์รัปต์ใด ๆ ทั้งสิ้น

1 : อินเตอร์รัปต์แต่ละชนิดถูกควบคุมการตอบสนองจากบิตในรีจิสเตอร์นี้

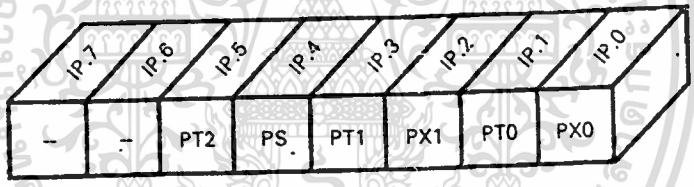
IE.6 – ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคต)

IE.5 ET2 ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 2 เมื่อเกิด overflow หรือเกิดอินเตอร์รัปต์ของไทม์เมอร์ 2 (มีในเฉพาะ MCS-51 บางเบอร์ที่มีไทม์เมอร์2 เช่น 8052)

IE.4	ES	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม
IE.3	ET1	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 1 เมื่อเกิด overflow
IE.2	EX1	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1
IE.1	ET0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0 เมื่อเกิด overflow
IE.0	EX0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 0

2) รีจิสเตอร์ใช้งานเฉพาะ IP (Interrupt Priority Register)

เข้าถึงข้อมูลได้ในระดับบิต ดังแสดงในรูปที่ 5.10



รูปที่ 5.10 รีจิสเตอร์ใช้งานเฉพาะ IP

IP.7	--	ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคต)
IP.6	--	ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคต)
IP.5	PT2	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์2
IP.4	PS	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม

IP.3 PT1 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์1

IP.2 PX1 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1

IP.1 PT0 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0

IP.0 PX0 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1

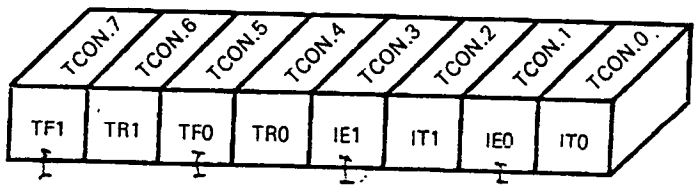
การให้บิตกำหนดความสำคัญของอินเตอร์รัปต์เป็น 0 หมายถึงให้อินเตอร์รัปต์ชนิดนั้นมีลำดับความสำคัญต่ำ ส่วนการให้บิตกำหนดความสำคัญของอินเตอร์รัปต์เป็น 1 หมายถึงให้อินเตอร์รัปต์ชนิดนั้นมีลำดับความสำคัญสูง

รีจีสเตอร์ที่ควบคุมการทำงานของไทม์เมอร์/เคาน์เตอร์ รีจีสเตอร์ที่ควบคุมการทำงานของไทม์เมอร์หรือเคาน์เตอร์ใน MCS-51 ประกอบด้วนรีจีสเตอร์ใช้งานเฉพาะ TCON,TMOD และ T2CON (มีใน MSC-51 ที่มีไทม์เมอร์ 2 เท่านั้น)

5.2.6 รีจิสเตอร์ที่ควบคุมการทำงานของไทม์เมอร์ / เคาท์เตอร์

1) รีจิสเตอร์ใช้งานเฉพาะ TCON (Timer / Counter Control Register)

เข้าถึงข้อมูลได้ในระดับบิต ดังแสดงในรูปที่ 5.10



รูปที่ 5.10 รีจิสเตอร์ใช้งานเฉพาะ TCON

TCON.7 TF1 บิตแสดงการเกิด over flow ของไทม์เมอร์ 1 จะถูกเซตเองเมื่อไทม์เมอร์ 1 เกิด over flow เมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการการอินเตอร์รัปต์

TCON.6 TR1 บิตควบคุมการนับของไทม์เมอร์ 1 (ควบคุมการทำงานจากโปรแกรม)

1 : ไทม์เมอร์ 1 เริ่มทำงานต่อ (นับต่อ)

0 : ไทม์เมอร์ 1 หยุดทำงาน (หยุดนับสัญญาณนาฬิกาภายใน หรือ จำนวนพัลส์ภายนอกที่ขา INT1)

TCON.5 TF0 บิตแสดงการเกิด over flow ของไทม์เมอร์ 0 ถูกเซตเองเมื่อไทม์เมอร์ 0 เกิด over flow และจะถูกเคลียร์เองเมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัปต์

TCON.4 TR0 บิตควบคุมการนับของไทม์เมอร์ 0 (ควบคุมจากโปรแกรม)

1 : ไทม์เมอร์ 0 เริ่มทำงานต่อ (นับต่อ)

0 : ไทม์เมอร์ 0 หยุดทำงาน (หยุดนับสัญญาณนาฬิกาภายในหรือ จำนวนพัลส์ภายนอกที่ขา INT0)

TCON.3 IE1 บิตแสดงสถานะสัญญาณอินเทอรัปต์ภายนอกชนิดที่ 1 ซึ่งจะถูกเซ็ตเองโดยฮาร์ดแวร์เมื่อมีสัญญาณอินเทอรัปต์ที่ได้จากการตรวจสอบการเปลี่ยนสถานะและจะถูกเคลียร์เองโดยคำสั่ง RETI ที่อยู่ในโปรแกรมส่วนบริการอินเตอรัปต์

TCON.2 IT1 บิตเลือกประเภทการตรวจสอบสัญญาณอินเทอรัปต์ที่เกิดขึ้นที่ขา INT1 โดย

1 : ตรวจสอบการเปลี่ยนระดับสัญญาณจาก 1 เป็น 0 ที่ขา INT1

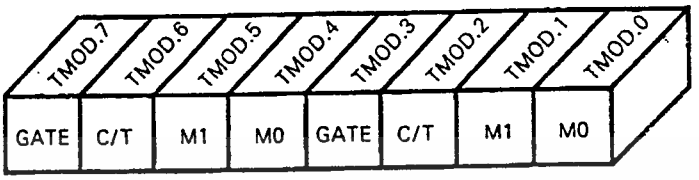
0 : ตรวจสอบระดับของสัญญาณที่ขา INT 1

TCON.1 IE0 บิตแสดงสถานะสัญญาณอินเทอรัปต์ภายนอกชนิดที่ 0 ซึ่งจะถูกเซ็ตเองโดยฮาร์ดแวร์เมื่อมีสัญญาณอินเทอรัปต์ที่ได้จากการตรวจสอบการเปลี่ยนสถานะและจะถูกเคลียร์เองโดยคำสั่ง RETI ที่อยู่ในโปรแกรมส่วนบริการอินเตอรัปต์

TCON.0 IT0 บิตเลือกประเภทการตรวจสอบสัญญาณอินเตอรัปต์ที่เกิดขึนที่ขา INT1 เหมือนบิต IT1

2) รีจิสเตอร์ใช้งานเฉพาะ TMOD (Timer/Counter Mode Control Regiser)

ไม่สามารถเข้าถึงข้อมูลในระดับบิตได้ ดังแสดงในรูปที่ 5.11



รูปที่ 5.11 รีจิสเตอร์ใช้งานเฉพาะ TMOD

GATE บิตเลือกการควบคุมให้รีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์ หรือเคาน์เตอร์ ทำงาน โดยควบคุมจากฮาร์ดแวร์หรือซอฟต์แวร์ดังนี้

- เมื่อบิต TRx (TR0 , TR1) และ GATE ถูกเซต ไทม์เมอร์หรือเคาท์เตอร์จะทำงานต่อเมื่อสถานะที่ขา INTx (INT0 , INT1) มีค่าเป็น 1 (ควบคุมจากฮาร์ดแวร์)
- เมื่อบิต GATE ถูกเคลียร์ ไทม์เมอร์หรือเคาน์เตอร์จะทำงานก็ต่อเมื่อบิต TRx ถูกเซตโดยไม่ขึ้นกับสถานะที่ขา INTx (ควบคุมจากซอฟต์แวร์)

C / T บิตเลือกการทำงานของรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ดังนี้

0 : หมายถึงทำงานเป็นไทม์เมอร์ (นับจำนวนแมชชีนไซเกิล)

1 : หมายถึงทำงานเป็นเคาน์เตอร์ (นับจำนวนพัลส์ภายนอกที่ขา Tx)

M1 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือไทม์เมอร์ 1

M0 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือไทม์เมอร์ 1

0 0 โหมด 0 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 13 บิต

0 1 โหมด 1 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิต

1 0 โหมด 2 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิต ที่มีการโหลดค่าเองเมื่อมี overflow

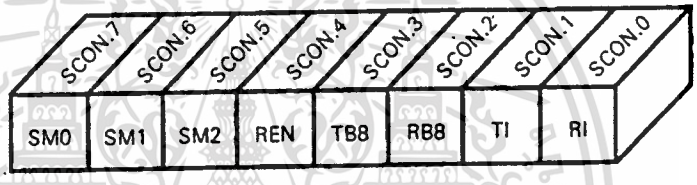
1 1 โหมด 3 : ไทม์เมอร์ 0

- รีจิสเตอร์ TL0 ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิต ที่ควบคุมการทำงานได้จากบิตของไทม์เมอร์ 0 เอง
- รีจิสเตอร์ TH0 ใช้เป็นไทม์เมอร์ขนาด 8 บิต ที่ควบคุมการทำงานได้จากบิตของไทม์เมอร์ 1

1 1 โหมด 3 : ไทม์เมอร์ 1 หยุดการทำงาน

3) รีจิสเตอร์ใช้งานเฉพาะ SCON (Serial Port Control Register)

สามารถเข้าถึงข้อมูลได้ระดับบิต ดังแสดงในรูปที่ 5.12



รูปที่ 5.12 รีจิสเตอร์ใช้งานเฉพาะ SCON

SCON.7 SM0 บิตเลือกการทำงานของพอร์ตสื่อสารอนุกรมในโหมดต่างๆ

SCON.6 SM1 บิตเลือกการทำงานของพอร์ตสื่อสารอนุกรมในโหมดต่างๆ

0 0 โหมด 0 : ทำงานเป็น shift register มี baud rate = 1/12 ของความถี่ออสซิลเลเตอร์

0 1 โหมด 1 : 8 บิต UART buad rate กำหนดเองได้

1 0 โหมด 2 : 9 บิต UART buad rate = 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์

1 1 โหมด 3 : 9 บิต UART buad rate กำหนดเองได้

SCON.5 SM2 บิตเลือกการใช้งานพอร์ตสื่อสารอนุกรมในโหมด 2 และ 3 เพื่อใช้ติดต่อระหว่างซีพียูด้วยกันเอง

1 : ใช้พอร์ตสื่อสารอนุกรมในการติดต่อระหว่างซีพียูด้วยกันเอง

0 : ใช้พอร์ตสื่อสารอนุกรมตามปกติ

SCON.4 REN บิตควบคุมการอนุญาตให้มีการรับข้อมูล ดังนี้

1 : อนุญาตให้มีการรับข้อมูลจากภายนอกได้

0 : ไม่อนุญาตให้มีการรับข้อมูลจากภายนอก

SCON.3 TB8 บิตข้อมูลบิตที่ 9 ซึ่งจะถูกส่งออกไปในการทำงานของพอร์ตสื่อสารอนุกรมโหมด 2 และ 3 การเซต หรือ เคลียร์จะกระทำด้วยคำสั่งในโปรแกรมเท่านั้น

SCON.2 RB8 บิตข้อมูลบิตที่ 9 ที่ได้รับมาจากภายนอกในการทำงานของพอร์ตสื่อสารอนุกรมโหมด 2 และ 3 ส่วนในการทำงานโหมด 1 ถ้าหากบิต SM2 = 0 บิตนี้จะเป็นบิตสิ้นสุดของข้อมูลที่รับเข้ามาได้ และไม่ถูกกำหนดการใช้งานในโหมด 0

SCON.1 TI บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการส่งข้อมูลถูกเซตโดยฮาร์ดแวร์ เมื่อข้อมูลบิตที่ 8 ถูกส่งออกไปแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่นๆจะถูกเซตโดยฮาร์ดแวร์ เมื่อเริ่มส่งบิตสิ้นสุดของข้อมูลออกไปและจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

SCON.0 RI บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการรับข้อมูลถูกเซตโดยฮาร์ดแวร์ เมื่อได้รับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 หรือที่จุดกึ่งทางของช่วงรับบิตสิ้นสุดของข้อมูลในการทำงานโหมดอื่น (มีข้อยกเว้นในกรณีใช้พอร์ตสื่อสารอนุกรมติดต่อระหว่างซีพียูด้วยกันเอง) และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

บทที่ 6

การตรวจวัดและการส่งข้อมูลด้วยแสงอินฟราเรด

อุปกรณ์เครื่องมือเครื่องใช้หลายอย่างในปัจจุบันมีการติดต่อระหว่างกัน เช่น การใช้รีโมทคอนโทรล , การตรวจวัดตำแหน่ง-ระยะทาง แสงอินฟราเรดนำมาใช้ในกรณีข้างต้นได้เป็นอย่างดี เนื่องจากราคาถูก , ง่ายต่อการใช้,อุปกรณ์ต่างๆหาได้ง่าย และยังไม่มีข้อจำกัดต้องขออนุญาตการใช้งาน (ที่จำเป็นสำหรับระบบคลื่นวิทยุ) ระบบแสงอินฟราเรดนี้จริงๆแล้วทำได้ไม่ยากแต่นักออกแบบยังไม่ค่อยคุ้นเคย จุดประสงค์ของบทความนี้ก็เพื่อเป็นแนวทางทำความเข้าใจในระบบแสงอินฟราเรดนี้

6.1 อุปสรรคทั่วไป

จากรูปที่ 1 แสดงระบบอินฟราเรดทั่วๆไป ส่วนการส่งแสดงโดยมีอุปสรรคคือการขับต้นกำเนิดแสงให้ได้พลังงานที่เพียงพอที่ส่วนรับจะสามารถรับได้

รูปที่ 6.1 ระบบอินฟราเรดอย่างง่าย

ในส่วนของการรับจะมีตัวแปรหลายตัวที่จะต้องนำมาพิจารณา สิ่งแวดล้อมที่มีแสงล้อมรอบจะเป็นประเด็นหลักที่จะต้องนำมาพิจารณา ในกรณีของสัญญาณอินฟราเรดที่ถูกส่งมามีกำลังอ่อน เมื่อเทียบกับพลังงานแสงของสิ่งแวดล้อม อย่างเช่น หลอดไฟ หลอดฟลูออเรสเซนท์ และแสงอาทิตย์

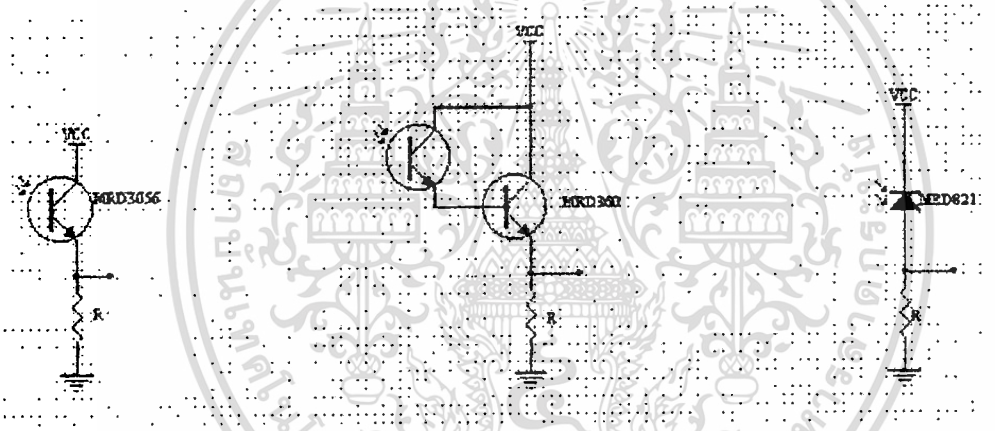
มีวิธีแก้ปัญหานี้ได้ 2 ทาง

วิธีแรก คือ การสร้างการกระตุ้นของระดับรอบข้างเพื่อที่จะตรวจจับซึ่งจะเปรียบเทียบได้กับไฟฉ่วงกระแสตรง ซึ่งเป็นผลให้ความไวในการรับลดลง และที่เลวร้ายยิ่งกว่านั้นอาจทำให้เกิดการอิ่มตัวของเครื่องตรวจจับบางชนิด

· วิธีที่สอง คือ การสร้างระดับของสัญญาณรบกวน ที่มีค่ามากกว่า สัญญาณข้อมูลอยู่ 60 เดซิเบล โดยเฉพาะสัญญาณรบกวนที่มีความถี่ 50 - 60 เฮิรตซ์ หรืออาจกล่าวได้ว่าความไวของ silicon photo detector จะเพิ่มขึ้นในช่วงที่สามารถมองเห็นได้ ความไวนี้จะลดลงเนื่องจากการรบกวนของ หลอดไฟ และ แสงอาทิตย์

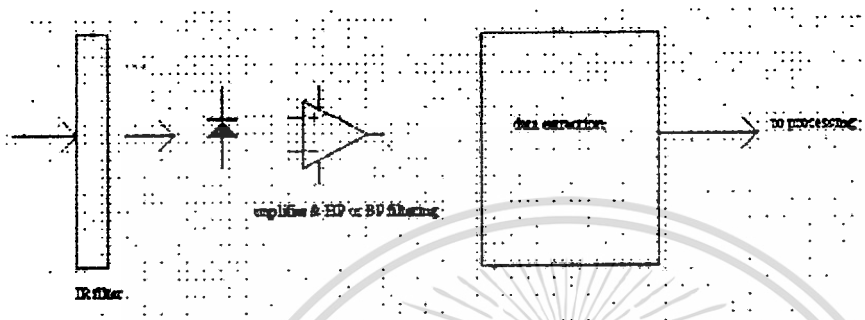
6.2 เครื่องรับ

เมื่อมีโฟตรอนมากระทบกับรอยต่อ จึงเกิดเป็นกระแสขึ้น



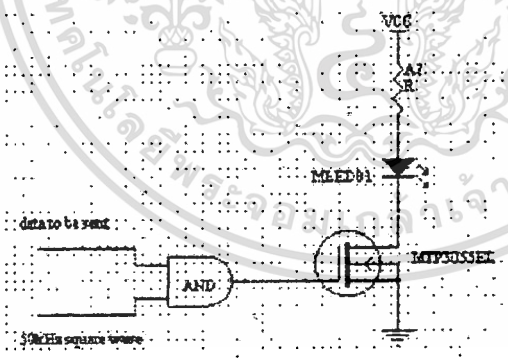
รูปที่ 6.2 แสดงรูปแบบของเครื่องรับไว้ 3 แบบ ซึ่งทั้ง 3 วิธีจะผลิตคู่ อิเลคตรอนและโฮล

แบบ Darlington และแบบ Phototransistor จะใช้ในระบบปิดซึ่งแหล่งกำเนิดจะต้องมีกำลัง มากพอ ส่วนแบบ Photodiode จะสามารถใช้ในระยะทางที่ไกลกว่าทั้งสองแบบข้างต้น



รูปที่ 6.3 เครื่องรับอินฟราเรดอย่างง่าย

6.3 เครื่องส่ง



รูปที่ 6.4 แสดงรูปแบบของเครื่องส่ง ซึ่งก่อนที่เราจะทำการส่งสัญญาณข้อมูลออกไปนั้น

เราจะต้องทำการ มอดูเลท สัญญาณข้อมูล กับสัญญาณพาหะเสียก่อน จากรูปเราจะเห็นว่าเรานำสัญญาณข้อมูล มาทำการมอดูเลทกับ สัญญาณ Square wave หลังจากนั้นจึงค่อยนำสัญญาณที่มอดูเลทแล้วมาผ่านอุปกรณ์ Driver ซึ่งในที่นี้เราจะใช้ FET เป็นตัว Driver และหลังจากนั้นจึงนำไปผ่าน Infrared emitting diode ต่อไป

6.4 การส่งระยะไกล

ถ้าต้องการส่งให้ได้ระยะไกลๆ จะต้องมีวิธีเพิ่มเติมซึ่งมีด้วยกันหลายวิธีดังต่อไปนี้

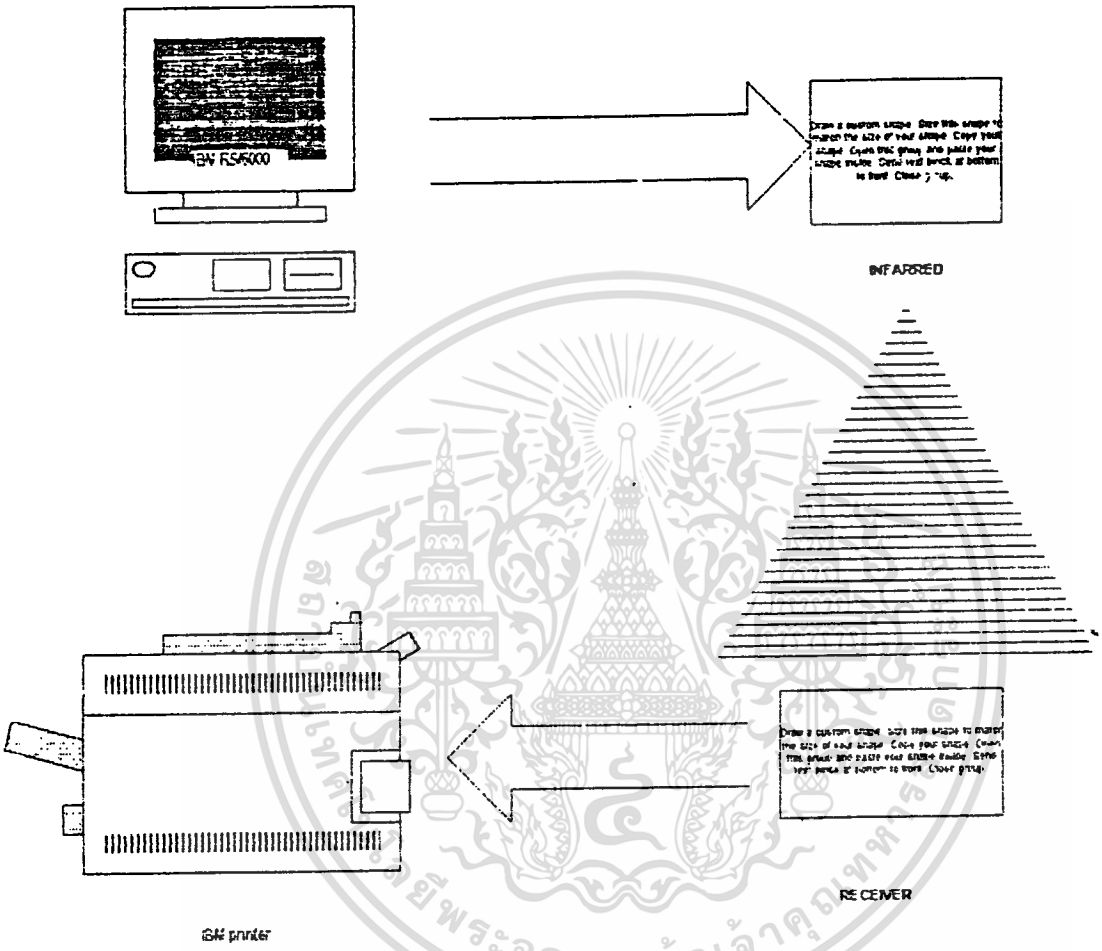
- เพิ่มกระแสขับตัวส่งอินฟราเรด
- ใช้ไดโอดหลายๆ ตัวต่ออนุกรมกัน ข้อดีคือไม่ต้องเพิ่มแหล่งจ่ายกระแสแต่มีข้อเสียคือ ไดโอดถ้าเอามาต่อหลายตัวแล้วทิศทางจะไม่ดี เพราะแต่ละตัวจะหันหน้าไม่ตรงตำแหน่งกัน
- ใส่ pulse ที่มี duty circle สั้นมากๆ ใช้กระแสแค่ไม่กี่แอมป์ก็พอแต่ใช้ pulse ที่มีคาบมีหน่วยเป็น μsec โดยมี duty circle 5% หรือต่ำกว่า แต่ต้องมีภาครับที่ออกแบบเป็นพิเศษ

ในภาครับก็จะมีช่องรับแสงกว้างขึ้นเพื่อให้รับพลังงานมาสะสมได้มากขึ้น โดยสามารถต่อไดโอดขนานกันได้หลายๆ ตัวเพื่อช่วยกันรับเอากระแสมารวมกัน แต่ไม่สามารถคำนึงเรื่องทิศทางได้ วิธีอื่นอีกก็คือใช้เลนส์โดยวางไดโอดที่จุดโฟกัสจะทำให้สามารถเพิ่มกระแสได้

บทที่ 7

เครื่องรับส่งข้อมูลแบบไร้สาย (INFARRED WIRELESS PRINTER)

7.1 บล็อคไดอะแกรมแสดงการทำงานของเครื่องรับส่งข้อมูลแบบไร้สาย



รูปที่ 7.1 ไดอะแกรมแสดงการทำงานของเครื่องรับส่งข้อมูลแบบไร้สาย

เครื่องรับส่งข้อมูลกับเครื่องพิมพ์แบบไร้สายเป็นอีกทางเลือกหนึ่งในการติดต่อสื่อสารข้อมูลระหว่างเครื่องพิมพ์กับคอมพิวเตอร์ นอกเหนือจากการส่งผ่านข้อมูลแบบขนานโดยใช้สาย ซึ่งการส่งข้อมูลแบบไร้สายมีข้อดี คือ สามารถส่งได้ไกลกว่าแบบขนานและประหยัดสายในการส่งข้อมูล เนื่องจากการส่งแบบนี้เป็นการส่งแบบอนุกรม การติดต่อระหว่างเครื่องพิมพ์ในปัจจุบันเป็นการติดต่อแบบขนาน การทำงานของเครื่องส่งนี้จึงต้องมีการแปลงการส่งแบบขนานมาเป็นเป็นแบบอนุกรมก่อนแล้วจึงส่งผ่านข้อมูลไปให้เครื่องรับ เครื่องรับจึงแปลงจากการส่งแบบอนุกรมเป็นแบบขนานอีกครั้งหนึ่งเพื่อให้สามารถติดต่อกับเครื่องพิมพ์ได้

7.2 การทำงานของเครื่องรับส่งข้อมูลแบบไร้สาย

แบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

7.2.1 ส่วนของการส่งข้อมูล

7.2.2 ส่วนของการรับข้อมูล

7.2.1 การทำงานของการส่งข้อมูล

การทำงานของการส่งข้อมูลนี้จะใช้ไมโครคอนโทรลเลอร์ เบอร์ Z84C015 เป็นตัวควบคุมการทำงานในส่วนของการส่งข้อมูลซึ่งในชิบของ Z80-PIO ,Z80-SIO และ CTC ใช้การทำงานนี้การทำงานเมื่อคอมพิวเตอร์ส่งข้อมูลในการพิมพ์มา โดยการส่งสัญญาณสโตรบ (STROBE) มายังเครื่องส่งข้อมูล (ซึ่งขณะส่งสัญญาณนี้จะมีข้อมูลมารอพร้อมอยู่ที่ขา DATAของ PIOพอร์ตอยู่แล้ว) ขา STROBE นี้จะต่ออยู่กับขา ASTB ของพอร์ต PIOจากนั้นเครื่องส่งข้อมูลจะรับข้อมูลแบบขนานโดยผ่านทาง PIO ใน Z84C015 ซึ่งในที่นี้ใช้ PIO พอร์ต A เป็นพอร์ตในการรับข้อมูลจากคอมพิวเตอร์ แล้ว Z84C015 จะเป็นตัวจัดการข้อมูลนำไปเก็บไว้ในบัฟเฟอร์

โดยขณะที่ Z84C015ยังไม่ได้รับข้อมูลนั้น พอร์ต A จะส่งสัญญาณ BUSY ขา ARDY เป็น LOW ผ่าน Inverted ก็จะเป็น High ไปยังขาBUSYของคอมพิวเตอร์ ขา 11 ทำให้ในช่วงนี้คอมพิวเตอร์ ไม่ส่งข้อมูลตัวใหม่มา

เมื่อพอร์ต A ว่าง สัญญาณจากขา ARDY จะเป็น High ผ่าน Inverterไปยังขา BUSY ของคอมพิวเตอร์ ทำให้คอมพิวเตอร์สามารถส่งข้อมูลตัวใหม่เข้ามาได้ (ถ้ามี)

จากนั้น Z84C015จะทำการส่งข้อมูลออกทาง SIO พอร์ต เป็นการส่งข้อมูลแบบอนุกรมซึ่งในที่นี้ใช้พอร์ต B ของSIO SIOจะทำการส่งอินเทอร์รัพท์มายัง Z80C015 เพื่อขอใช้การบริการจาก Z80C015 เมื่อพอร์ต SIOว่าง

การขอการอินเทอร์รัพท์ซึ่งในส่วนของเครื่องส่งนี้จะมีการขอการอินเทอร์รัพท์อยู่สองพอร์ต คือ พอร์ต A ของ PIOและพอร์ต B ของ SIO ในกรณีที่มีการของการอินเทอร์รัพท์มากกว่าสองตัวพร้อมกัน จะมีการจัดลำดับการขออินเทอร์รัพท์ โดยการจัดลำดับการอินเทอร์รัพท์ของส่วนเครื่องส่งนี้ พอร์ต B ของ SIO จะมีความสำคัญสูงสุดรองลงมาคือ พอร์ต A ของPIO

สมมติว่ามีการขออินเทอร์รัพท์ของพอร์ตสองพอร์ตในเวลาเดียวกัน จะดูว่ามีPRIORITLY BIT ว่าของพอร์ตไหนมีความสำคัญกว่าในที่นี้คือ SIO มี PRIOITY BIT สูงกว่าจึงทำให้ Z80C015 ทำการบริการการขออินเทอร์รัพท์ ของพอร์ต SIOก่อนหลังจากนั้นจึงบริการการขออินเทอร์รัพท์ตัวที่ต่ำกว่าตัวต่อไป

เครื่องส่งข้อมูลรี้จะส่งผ่านข้อมูลในอัตรา BAUD RATE 1200 BIT PER SECOND มี PARITY BIT เป็นแบบ ODD ข้อมูลจะถูกส่งผ่านโดยทาง INFARRED ไปยังส่วนรับข้อมูล

ส่วนหน่วยความจำ แสดงในรูปที่ 7.2 ซึ่งประกอบไปด้วย

1) ROM ขนาด 64k

2) RAM ขนาด 256K

โดยที่ ROM ใช้ในการเก็บโปรแกรมการทำงานRAM ใช้ในการเก็บค่าพารามิเตอร์ต่างๆที่ต้องการใช้ในส่วนของการส่งข้อมูล

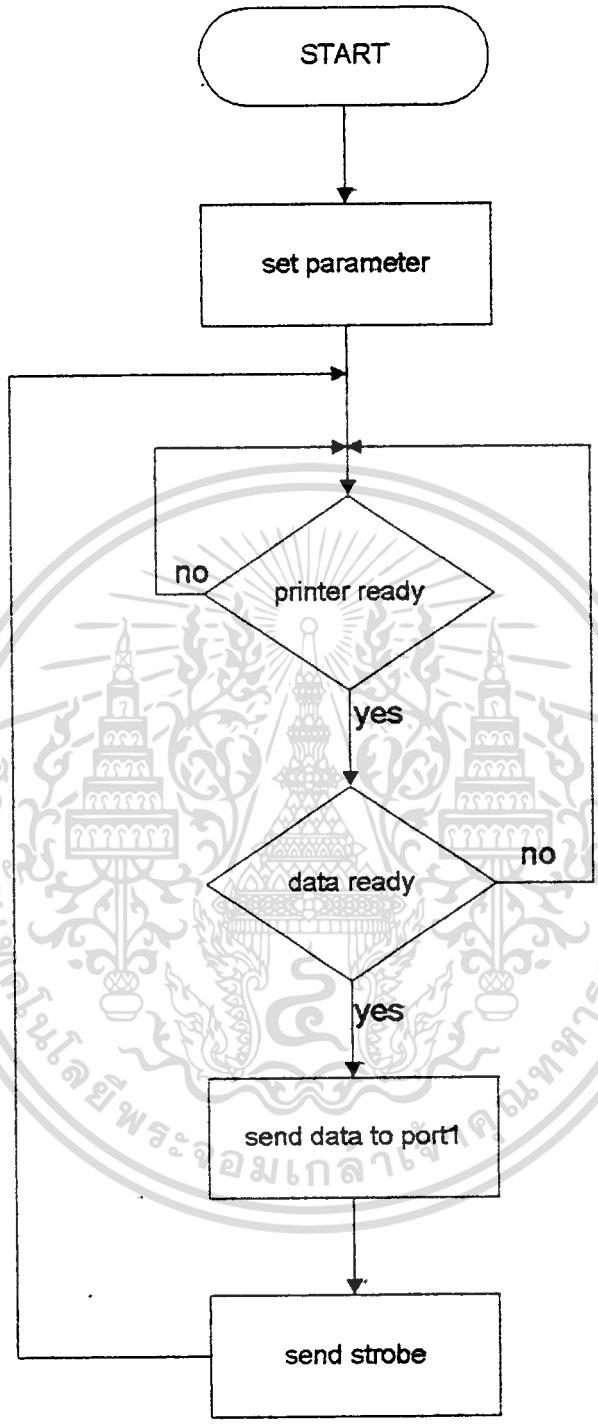
Memmory map ของเครื่องส่งข้อมูลมี ค่า Address ดังนี้

(0000)h-(1FFF)h เป็นROM ขนาด 64 K* 8 bit

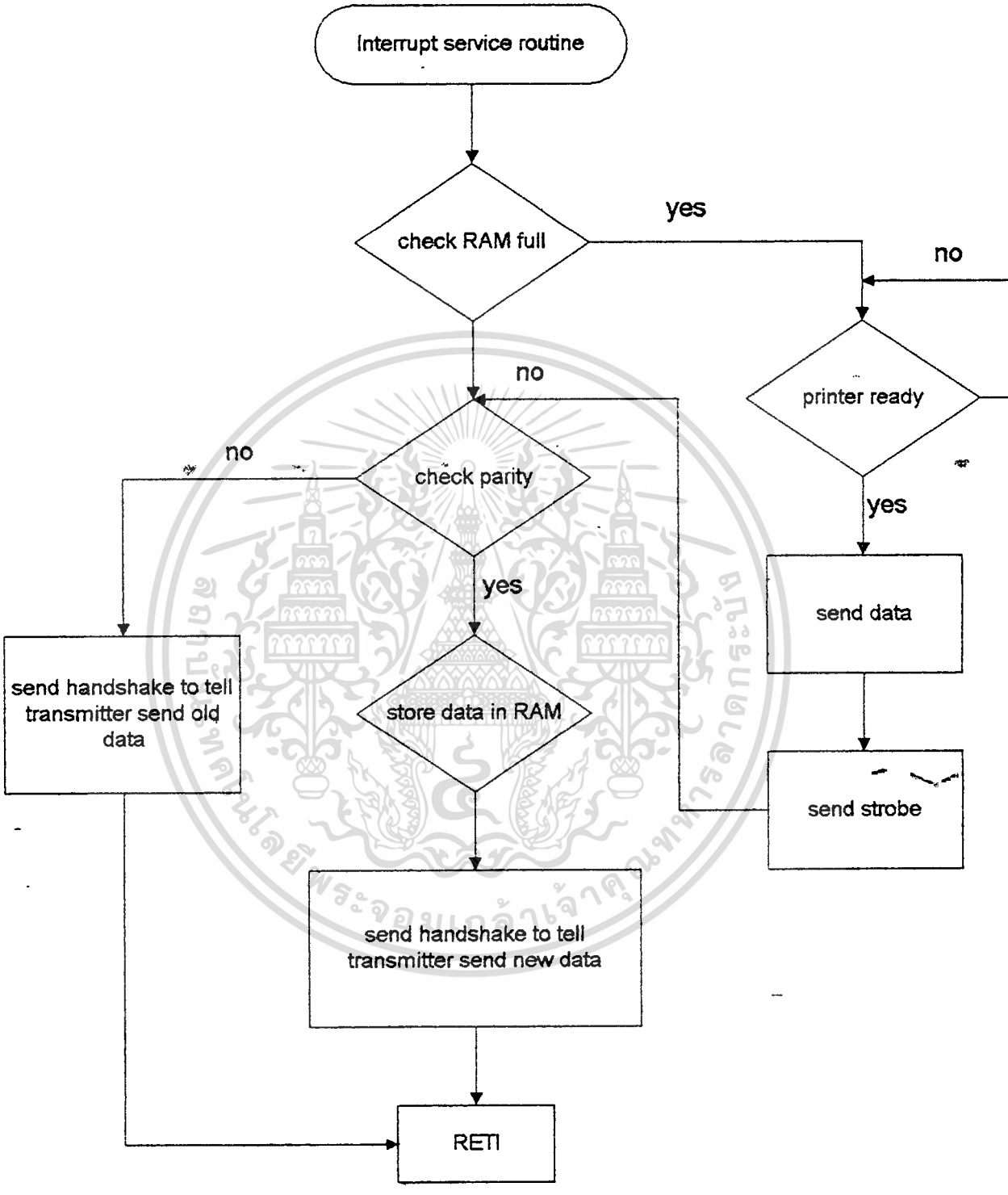
(8000)h-(F000)h เป็นRAM ขนาด256 K* 8bit

7.2.2 การทำงานในส่วนรับข้อมูล

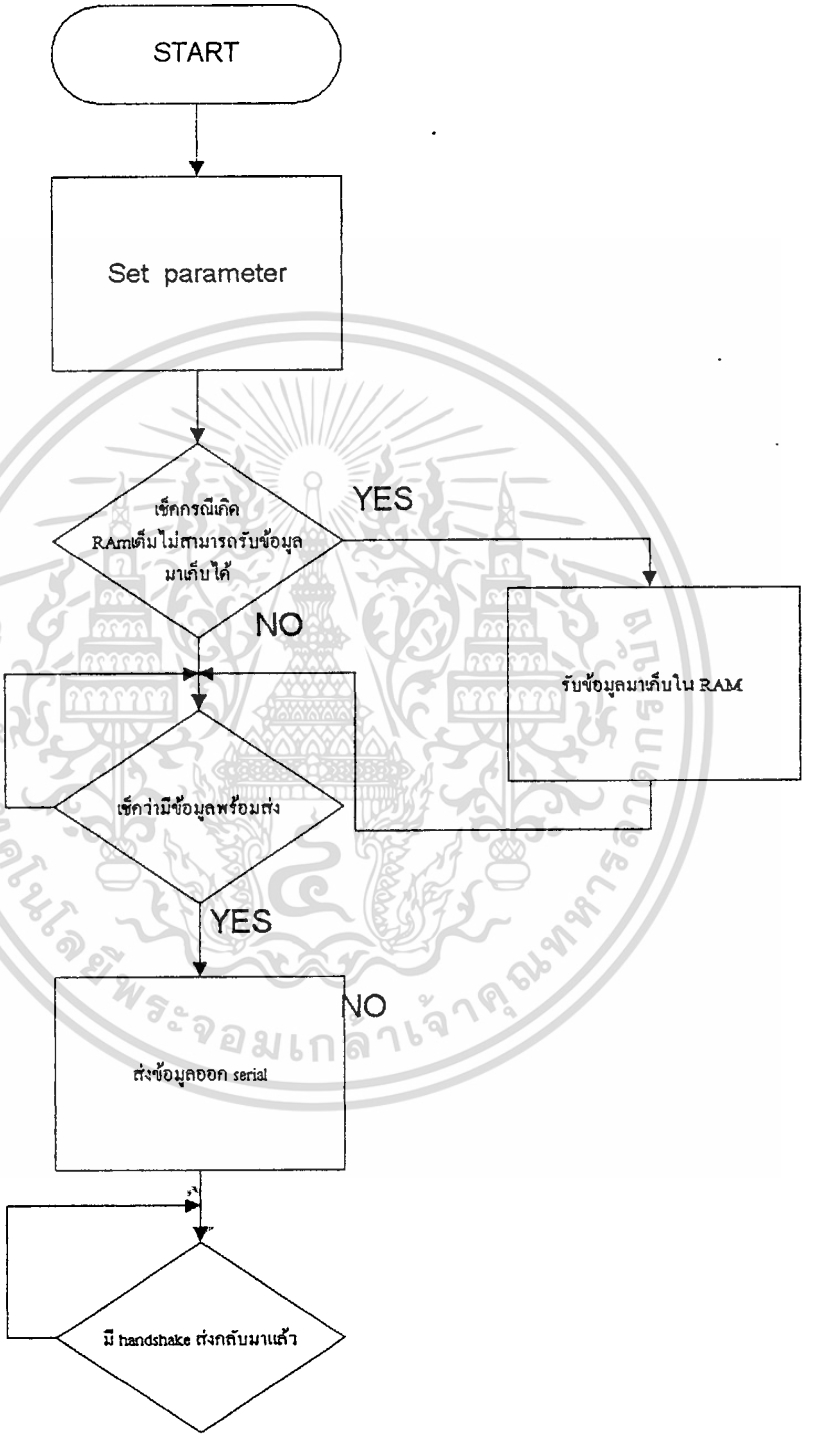
การทำงานของภาครับข้อมูลนี้จะใช้ไปโครคอนโทรลเลอร์เบอร์ 8031 เป็นตัวควบคุมการทำงานในส่วนของการรับข้อมูล โดยจะรับข้อมูลเข้ามาทางขา RxD ของ 8031 จะเริ่มต้นเมื่อวงจร 1-to-0 Transition Detector ทำการตรวจสอบข้อมูลแล้วพบว่าสัญญาณที่ขา RxDเปลี่ยนจาก 1 เป็น 0 ทำให้วงจรหาร 16 นับถึง 7,8และ9 จะมีการตรวจสอบข้อมูลที่เข้ามาทางขา RxD ถ้า 2 ใน 3 เหมือนกันก็ถือว่าข้อมูลที่เข้ามาคือค่านั้น ถ้าข้อมูลที่รับเข้ามาบิตแรกไม่เป็น 0 คือไม่ใช่ Start Bit(เพราะอาจตรวจพบการเปลี่ยนจากสัญญาณจากขา RxDแต่การตรวจสอบที่เวลา 7,8และ9 ส่วนใหญ่เป็น 1)ก็จะเกิดการรีเซ็ตส่วนรับข้อมูลแล้วกลับไปเริ่มต้นการตรวจสอบการเปลี่ยนสถานะจาก 1 เป็น 0 ใหม่ แต่ถ้าถูกต้องก็จะรับข้อมูลเข้ามาทีละ 1 บิต ข้อมูลนี้จะถูกเลื่อนเข้าไปเก็บทางขวาของ Input Shift Register และ 1 จะถูกเลื่อนออกไปทางซ้ายโดยสัญญาณ Shift จนกระทั่งสัญญาณ Stat Bit ซึ่งเป็น 0 ถูกเลื่อนเข้ามาถึงซ้ายสุดของ Input Shift Register จะบอกให้กับ Rx Control รู้ว่าจะต้องมีข้อมูลบิตสุดท้ายเข้ามาอีก 1 บิต เมื่อข้อมูลบิตสุดท้ายเข้ามาจะเกิดการโหลดข้อมูลจาก InputShift Register ไปยัง SBUF,RB8 และเซ็ต RI ให้เป็น 1 ในส่วนของซอฟแวร์ ก็จะนำ Bit RxB8 นี้ ซึ่งเป็นบิตพาริตี้ไปทำการเช็คว่าส่งข้อมูลมาถูกต้องหรือเปล่า ถ้าส่งมาถูกต้องก็จะนำข้อมูลใน SBUF ไปเก็บได้ ถ้าผิดพลาดก็จะส่งสัญญาณกลับไปบอกทางภาคส่งให้ส่งสัญญาณมาให้ใหม่อีกครั้งหนึ่ง เมื่อได้ข้อมูลแล้วก็จะนำไปเก็บใน RAM ซึ่งมีขนาด 32K bytes เพื่อรอที่จะส่งไปให้ยังเครื่องพิมพ์อีกทีหนึ่ง โดยพอร์ทขนาน P1 โดยจะมีโปรแกรมควบคุมการทำงานทั้งหมด ซึ่งจะกล่าวถึงต่อไป



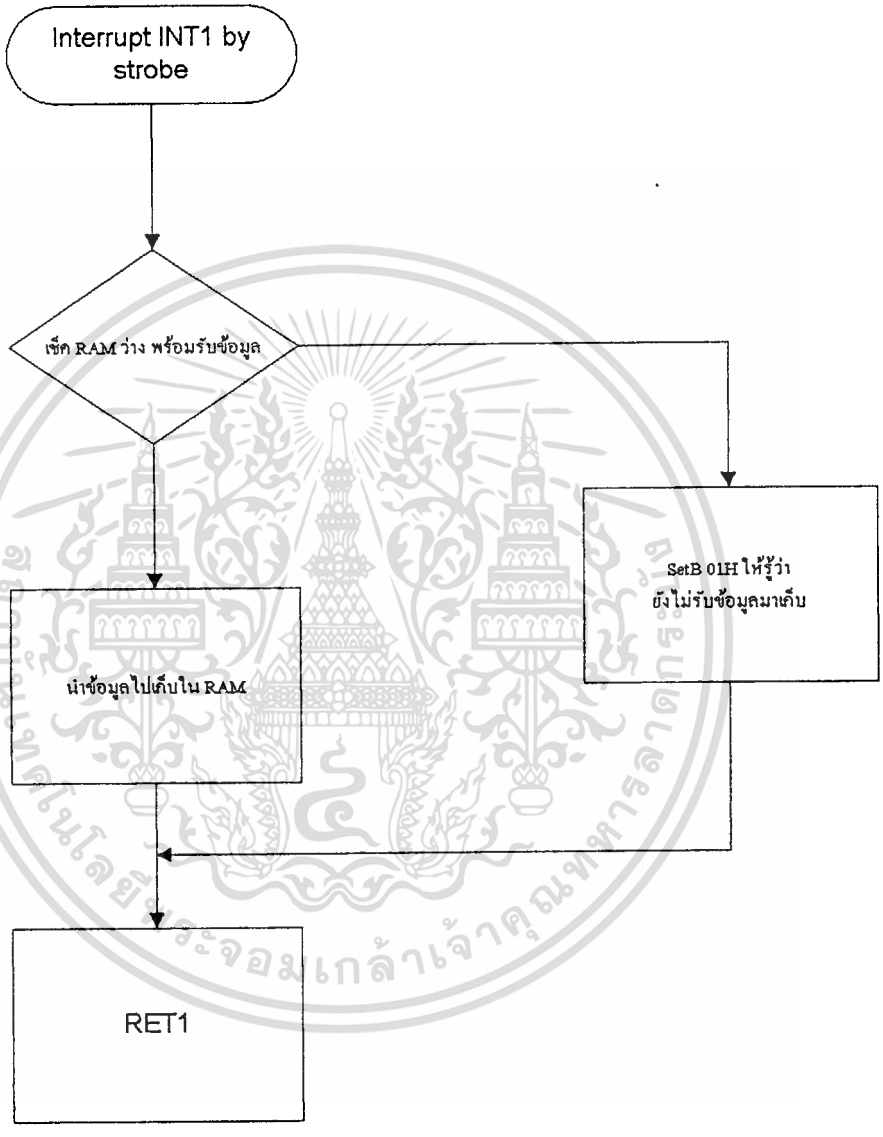
รูปที่ 7.2 โฟล์วชาร์ทแสดงการทำงานของโปรแกรมหลัก



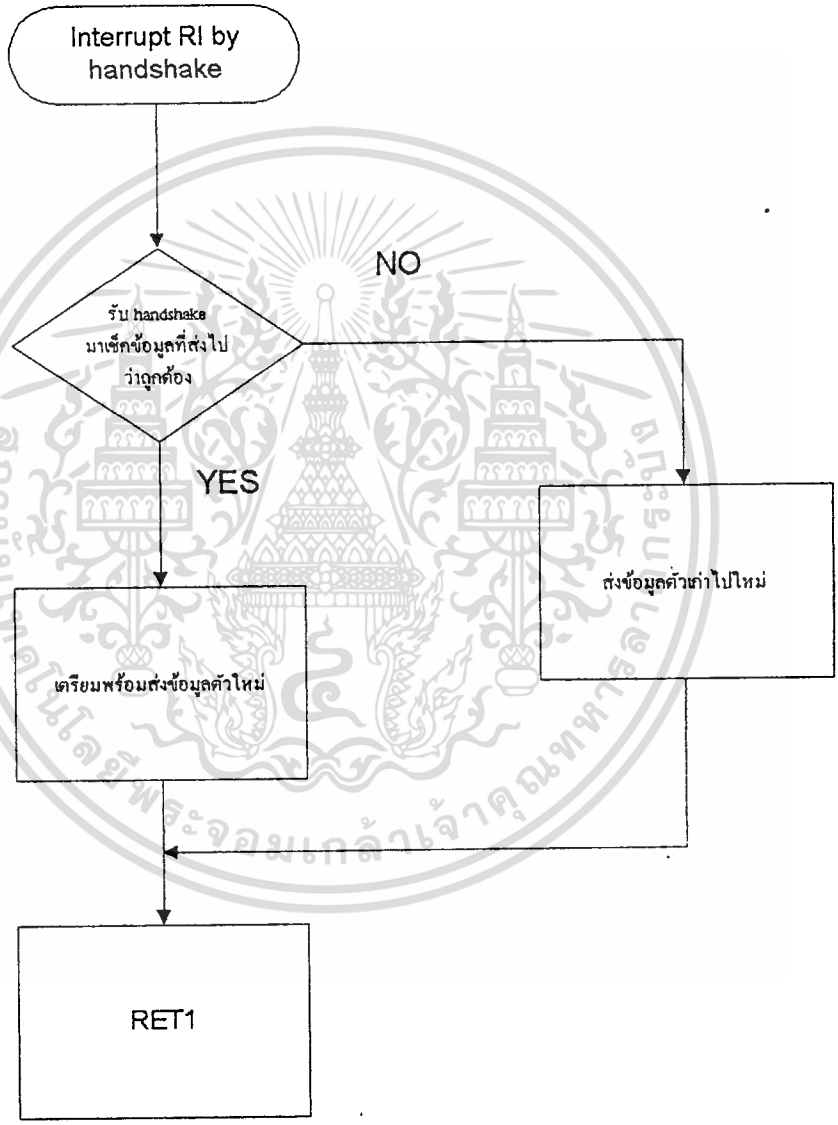
รูปที่ 7.3 โฟล์วชาร์ทแสดงการทำงานของโปรแกรมย่อย



รูปที่ 7.4 แสดงโฟล์วชาร์ทโปรแกรมหลักของภาคส่ง



รูปที่ 7.5 แสดงโฟล์วชาร์ทโปรแกรมย่อยรับข้อมูลเก็บในแรมของภาคส่ง



รูปที่ 7.6 แสดงโฟล์วชาร์ทโปรแกรมย่อยรับแฮนด์เชกของภาคส่ง

7.3 แนวคิดในการเขียนโปรแกรม

7.3.1 **ภาคส่ง** จากทฤษฎีที่ผ่านมาก็จะรู้ถึงหลักการในการส่งข้อมูลระหว่างคอมพิวเตอร์กับเครื่องพิมพ์ ซึ่งจะมีส่วนที่ใช้สำคัญมากๆ คือ สัญญาณสโตรบ(Strobe)กับสัญญาณบิสซี(Busy) ดังนั้นนสการเขียนโปรแกรมภาคส่ง ก็คือการรับข้อมูลจากคอมพิวเตอร์มาเก็บไว้ในแรม(Ram) และนำข้อมูลจากแรมส่งออกอนุกรมไปยังภาครับโดยจะใช้ สโตรบมาเป็นตัวอินเทอร์รัพท์ให้รับข้อมูลมาเก็บในแรม ส่วนโปรแกรมหลักก็จะเป็นโปรแกรมส่งข้อมูลโดยจะเช็คแรมตลอดเวลาว่ามีข้อมูลอยู่หรือไม่ ถ้ามีก็จะนำข้อมูลส่งออกทางพอร์ทอนุกรม ถ้าไม่มีก็วนลูปเช็คไปเรื่อยๆ จนกว่าจะมีการอินเทอร์รัพท์เข้ามาก็จะมีข้อมูลมาเก็บเพื่อรอการส่งออกต่อไป

7.3.2 **ภาครับ** เป็นการรับข้อมูลอนุกรมเข้ามาเก็บในแรมและนำข้อมูลจากแรมส่งไปให้เครื่องพิมพ์โดยการส่งนั้นจะเช็กว่ามีข้อมูลอยู่หรือไม่ จากนั้นจะเช็กขาบิสซีของเครื่องพิมพ์ว่าพร้อมที่รับข้อมูลหรือไม่ ถ้าไม่ก็จะวนลูปเช็กไปเรื่อยๆ ถ้าพร้อมก็จะทำการส่งข้อมูลไปให้เครื่องพิมพ์ ซึ่งการส่งข้อมูลนี้จะอยู่ในส่วนโปรแกรมหลัก โดยจะต้องสร้างสัญญาณสโตรบขึ้นมาเองเพื่อบอกให้เครื่องพิมพ์ทราบว่ามีข้อมูลส่งมาให้แล้ว ในส่วนรับข้อมูลอนุกรมจะใช้อินเทอร์รัพท์โดยพอร์ตอนุกรม เมื่อมีข้อมูลส่งมาครบก็จะอินเทอร์รัพท์ไปทำงานในการรับข้อมูลเข้ามาเก็บในแรม

ในการเช็คข้อมูลว่ามีข้อมูลอยู่หรือเช็คว่าแรมเต็มนั้นจะใช้เคาน์เตอร์เป็นตัวนับจำนวนข้อมูล โดยเมื่อมีข้อมูลเข้ามาเก็บในแรมเคาน์เตอร์ก็จะเพิ่มขึ้นทีละหนึ่ง ถ้ามีข้อมูลส่งออกจากแรมเคาน์เตอร์ก็จะลดลงทีละหนึ่ง

7.4 แนวคิดในการออกแบบวงจรรับส่งสัญญาณแสงอินฟราเรด

การส่งสัญญาณแสงอินฟราเรด ต้องมีการดัดแปลงรูปแบบของการส่งสัญญาณให้มีลักษณะคล้ายกับการส่งสัญญาณแบบคลื่นวิทยุ โดยมีการส่งสัญญาณแบบโทเบิรสต์ (Tone Burst) แทนการส่งสัญญาณแบบพัลส์ธรรมดาเพื่อทำให้มีการรบกวนในการส่งการรับข้อมูล

7.4.1 สัญญาณโทนเบิรสต์

สัญญาณโทนเบิรสต์ประกอบด้วยพัลส์ความถี่สูงแบบต่อเนื่องตลอดช่วงความกว้างของบิตที่เป็น "1" ในขณะที่บิตข้อมูลอยู่ในสภาวะต่ำสัญญาณจะคงเดิมไม่มีการเปลี่ยนแปลง

สำหรับวงจรรับต้องออกแบบให้มีการตอบสนองความถี่ที่เหมาะสมกัน จึงจะสามารถรับข้อมูลจากเครื่องส่งได้ถูกต้องโดยไม่มีการรบกวนจากอุปกรณ์อื่นๆ ในพื้นที่ใกล้เคียงกันหรือการรบกวนจากแสงดวงอาทิตย์ หลอดไฟฟ้า

ข้อดีในการส่งสัญญาณแบบโทนเบิรสต์ก็คือสามารถขจัดสัญญาณรบกวนจากภายนอกได้ดีมาก ซึ่งในระบบที่ใช้สัญญาณแสงอินฟราเรดที่ส่งด้วยพัลส์ธรรมดา อาจมีแสงเข้ามารบกวนที่เครื่องรับจนอาจเกิดการผิดพลาดในการรับสัญญาณควบคุมได้ แต่ถ้าหากมีการใช้สัญญาณโทนเบิรสต์แล้ว สิ่งรบกวนต่าง ๆ เหล่านี่จะถูกขจัดออกไปได้โดยสิ้นเชิง

7.4.2 วงจรส่งสัญญาณโทนเบิรสต์

เนื่องาจากสิ่งที่ถูกเพิ่มเข้ามาจากสัญญาณพัลส์ธรรมดาก็คือพัลส์ความถี่สูง ดังนั้นวงจรส่งสัญญาณความถี่สูงเข้ามาด้วย วงจรกำเนิดสัญญาณอย่างง่าย ๆ ซึ่งในวงจรนี้ใช้ IC เบอร์ 555 ต่อเป็นวงจรชนิดอะสเตเบิลมัลติไวเบรเตอร์ดัง สัญญาณเอ้าพุทความถี่สูงสามารถนำไปขับหลอดอินฟาเรดเพื่อส่งเป็นสัญญาณ หากต้องการเพิ่มกำลังส่งให้ทรานซิสเตอร์เป็นตัวขับสัญญาณ

การส่งสัญญาณแบบโทนเบิรสต์สามารถทำได้โดยการรวมสัญญาณระหว่างอนุกรมพัลส์ความถี่สูง(จากขาเอ้าของไอซี 555) เข้าด้วยกันด้วยวงจรแนนด์เกต

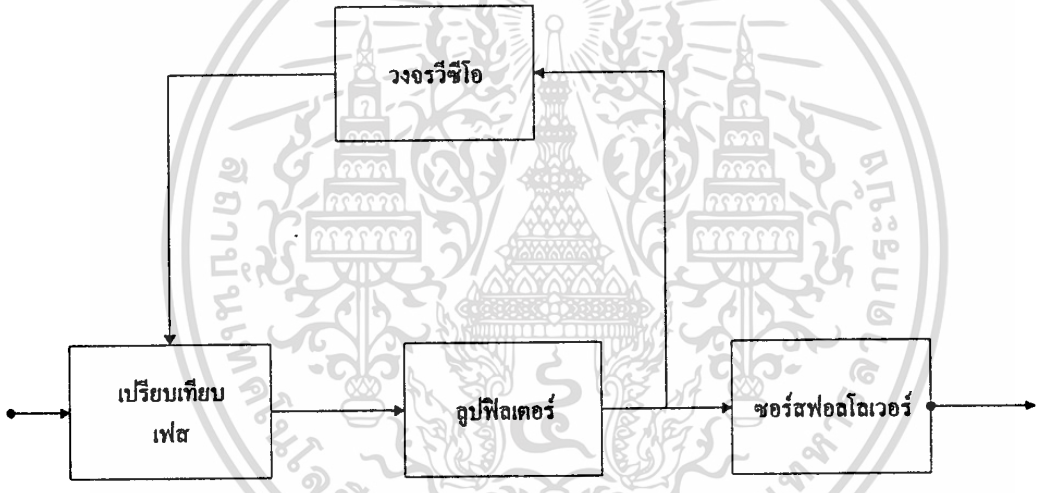
7.4.2.1 วงจรรับอย่างง่าย

การรรับสัญญาณโทนเบิรสต์ที่ส่งมาทำได้ด้วยวงจรซึ่งประกอบด้วยโฟโต้ไดโอดต่ออนุกรมกับตัวต้านทานเท่านั้นก็จะได้เอ้าพุต แค่สัญญาณที่ได้จะมีขนาดสัญญาณอ่อน

7.4.3 เฟสล็อคลูป

ในส่วนของวงจรรับต้องมีส่วนที่สามารถเลือกค่าความถี่ที่เหมาะสมกับช่องสัญญาณโดยการใช้วงจรหรือไอซีชนิดเฟสล็อคลูปเข้ามาเสริมการทำงาน

รูปที่ 7.7 เป็นบล็อกไดอะแกรมแสดงการทำงานของวงจรเฟสล็อคลูป หรือเรียกย่อ ๆ ว่าพีแอล แอล (PLL : Phase Locked Loop) เริ่มจากบล็อกของวงจรวีซีโอ (VCO : Voltage controlled Oscillator) เป็นวงจรกำเนิดสัญญาณความถี่ที่ถูกควบคุมด้วยแรงดัน สัญญาณที่ถูกสร้างขึ้นจากสัญญาณวีซีโอจะถูกนำไปเปรียบเทียบกับสัญญาณความถี่ที่เป็นอินพุทของระบบด้วยวงจรเปรียบเทียบเฟส สัญญาณแตกต่างของเฟสที่เกิดขึ้นจะถูกส่งไปยังวงจร หรือลูปฟิลเตอร์เพื่อผลิตแรงดันไฟตรงออกมาก่อนผ่านวงจรซอร์สฟอลโลเวอร์เป็นเอ้าพุตออกไป



รูปที่ 7.7 แสดงบล็อกไดอะแกรมวงจรเฟสล็อคลูป

ในขณะเดียวกันสัญญาณจากลูปฟิลเตอร์ในรูปของขนาดหรือแมกนิจูดของแรงดันที่เปลี่ยนแปลงตามค่าเฟสที่แตกต่างกันในตอนต้น จะถูกป้อนกลับไปยังวงจรวีซีโอเพื่อปรับค่าความถี่ของวงจรวีซีโอสำหรับเปรียบเทียบกับสัญญาณอินพุทใหม่เป็นเช่นนี้ไปเรื่อย ๆ

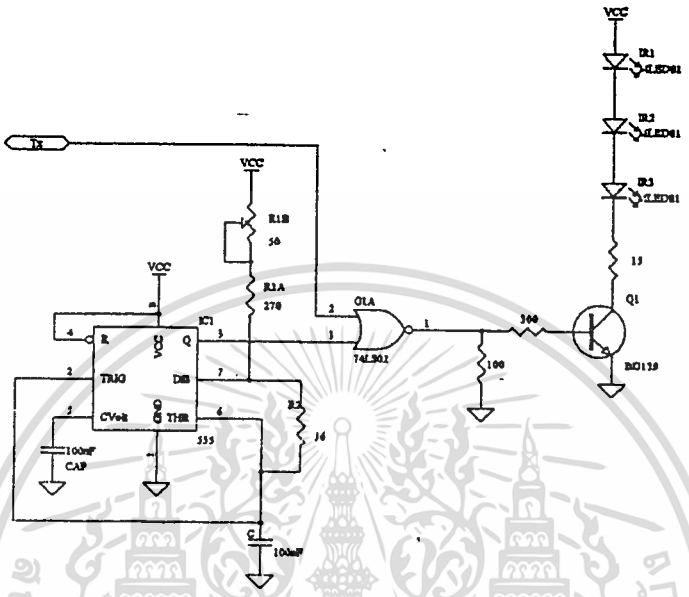
หากสัญญาณอินพุตมีค่าความถี่เท่ากับหรือสูงกว่า (ขึ้นกับวงจรที่ออกแบบ) ความถี่ที่กับหนดไว้ในวงจรวีซีโอ (บางครั้งอาจออกแบบให้ทำงานเฉพาะความถี่ที่เท่ากับ หรือเป็นฮาร์มอนิกกับความถี่กำหนดเท่านั้น) จะได้สัญญาณเอาต์พุทแสดงออกมาจากเฟสล็อคลูป แต่ถ้าสัญญาณอินพุตมีค่าความถี่นอกเหนือจากนี้ ก็จะไม่มีสัญญาณเอาต์พุตออกมาในภาคสุดท้าอาจออกแบบให้ทำงานเฉพาะความถี่ที่เท่ากับ หรือเป็นฮาร์มอนิกกับความถี่กำหนดเท่านั้น) จะได้

สัญญาณเอาต์พุทแสดงออกมาจากเฟสล็อคลูป แต่ถ้าสัญญาณอินพุตมีค่าความถี่นอกเหนือจากนี้ ก็จะไม่มีสัญญาณเอาต์พุตออกมาในภาคสุดท้าย

7.5 วงจรอินฟาเรดของภาคส่ง

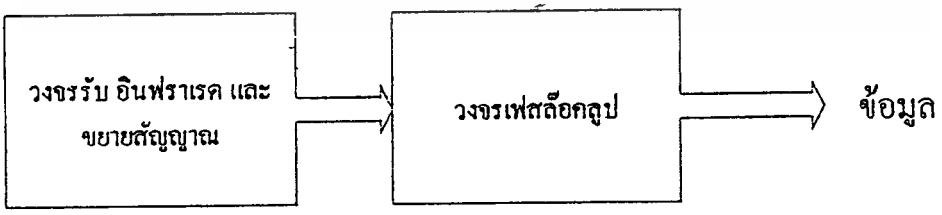
รูปที่ 7.8 แผนภาพแสดงโครงสร้างส่วนส่งสัญญาณอินฟราเรด

ในวงจรใช้ไอซีเบอร์ 555 เป็นตัวกำเนิดความถี่ในการโทนเบิตร์สซึ่งในวงจรนี้ออกแบบให้กำเนิดความถี่ประมาณ 40 กิโลเฮิร์ตนำไปรวมกับสัญญาณข้อมูลของเครื่องส่ง การรวมสัญญาณโดยการใช้นอร์เกต(ไอซีเบอร์ 74LS02) จากนั้นสัญญาณจะผ่านมาที่ทรานซิสเตอร์ที่ขาเบสเพื่อไบอัสทรานซิสเตอร์ให้ทำงาน หลอดอินฟราเรดจะทำงานตามการไบอัสของทราซิสเตอร์ วงที่ใช้ในการส่งข้อมูลด้วยแสงอินฟราเรดแสดงดังรูปที่ 7.9



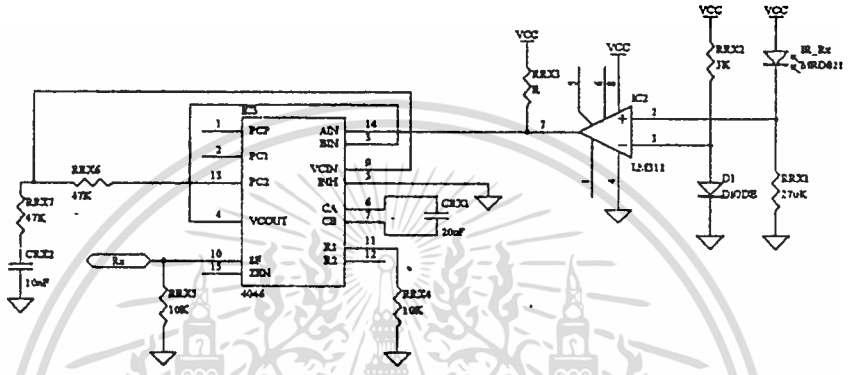
รูปที่ 7.9 แสดงวงจรส่งข้อมูลด้วยแสงอินฟราเรด

7.6 วงจรรับข้อมูลทางแสงอินฟราเรด



รูปที่ 7.10 แผนภาพแสดงโครงสร้างส่วนรับข้อมูลสัญญาณอินฟราเรด

สัญญาณแสงอินฟราเรดจากเครื่องส่ง จะรับด้วยตัวรับอินฟราเรดผ่านเฟสล็อคลูปส่ง แปลงเป็นข้อมูลออกมาส่งไปยังคอนโทรลเลอร์เพื่อจัดการข้อมูลต่อไป รูปที่ 7.11 แสดงวงจรรับอินฟราเรด



รูปที่ 7.11 แสดงวงจรรับข้อมูลแสงอินฟราเรด

การทดลองการจัดเก็บข้อมูลในบัฟเฟอร์ของเครื่องส่ง

จุดประสงค์ในการทดลอง

การทดลองนี้จะทำการจับเวลาในการส่งพิมพ์ เพื่อที่จะวัดความเร็วในการจัดเก็บข้อมุลของเครื่องส่ง (เครื่องส่งมีบัฟเฟอร์ 32 กิโลไบต์เก็บข้อมูลโดยผ่านทางพอร์ตขนานของเครื่องคอมพิวเตอร์)

วิธีทดลอง

1. ทำการต่อเครื่องส่งเข้ากับเครื่องคอมพิวเตอร์โดยการต่อพอร์ต DB25 ของเครื่องส่งเข้ากับพอร์ตเครื่องพิมพ์ของเครื่องคอมพิวเตอร์ และจ่ายไฟขนาด 5 โวลต์ ให้กับเครื่องส่ง

2. เตรียมไฟล์ที่ใช้พิมพ์โดยมี ขนาดไฟล์ประมาณ 25 กิโลไบต์ ในการทดลองนี้ใช้การส่งพิมพ์โดยการใช้คำสั่ง TYPE ของ DOS6.2 ในการส่งพิมพ์

3. จับเวลาในการส่งพิมพ์ การจับเวลาเริ่มจับตั้งแต่ได้กด ENTER ในคำสั่ง TYPE ไฟล์สั่งให้พิมพ์ไฟล์ที่เตรียมไว้ในการทดลอง ทำการจับเวลาจนกว่าจะรับข้อมูลจากเครื่องเข้าไปเก็บในแรมจนหมดทำการทดลองซ้ำ 5 ครั้งบันทึกผลการทดลองลงในการทดลอง

ผลการทดลอง

ตารางบันทึกการทดลองการรับข้อมูลเข้าเก็บไว้ในบัฟเฟอร์

ลำดับในการทดลอง	ความเร็วในการเก็บข้อมูล (วินาที)
1	1.15
2	1.14
3	1.15
4	1.15
5	1.15

สรุปผลการทดลอง

การเก็บข้อมูลลงในบัฟเฟอร์ของเครื่องส่งสามารถทำงานได้ดีคือสามารถพักข้อมูลออกก่อนส่งออกทางอินฟาเรดสามารถทำให้อินฟราเรดทำงานได้ทันในการส่งโดยคอมพิวเตอร์และคอมพิวเตอร์สามารถทำงานอย่างอื่นได้โดยไม่ต้องรอพิมพ์เหมือนในการต่อแบบธรรมดากับที่ไม่มีบัฟเฟอร์

วิเคราะห์ผลการทดลอง

การทดลองการส่งออกพิมพ์โดยผ่านบัฟเฟอร์ของเครื่องส่งซึ่งมีแรมขนาด 32กิโลไบต์ ข้อบกพร่องที่พบคือการส่งไฟล์ในการพิมพ์ที่ขนาดใหญ่กว่าขนาดของบัฟเฟอร์มากๆ อาจจะทำให้ข้อมูลในการส่งพิมพ์มีปัญหาคือพิมพ์ออกมาไม่ได้อักษรตามที่ต้องการ

หนังสืออ้างอิง

1. ศูนย์ภาษาคอมพิวเตอร์, "การใช้งาน Z80",

2. สุเจตน์ จันทรังษ์, "ไมโครคอนโทรลเลอร์ชิพเดี่ยว 8051"

3. ปรเมษฐ์ ประนยานันทน์, ปิยพงศ์ เผ่าวณิช, "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51

4. NEC PINWRITER P3200/P3300 USER'GUIDE

5. ZiLOG INTELLIGENT PERIPHERAL CONTROLLERS

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จได้ ด้วยความสนับสนุนและความเอื้อเฟื้อจากบุคคลต่าง ๆ มากมาย ดังต่อไปนี้

1. รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษาโครงงาน
2. อ. ขนิษฐา แซ่ตั้ง อาจารย์ประจำภาควิชาอิเล็กทรอนิกส์
3. อ. พลผดุง ผดุงกุล อาจารย์ประจำภาควิชาอิเล็กทรอนิกส์
4. อ. กิติพล ชิตสกุล อาจารย์ประจำภาควิชาอิเล็กทรอนิกส์
5. คณาจารย์ทุกท่าน, รุ่นพี่ปริญญาโทภาคอิเล็กทรอนิกส์, เพื่อนๆ ภาคอิเล็กทรอนิกส์ โดยเฉพาะ คุณปรีชา, คุณธัชพงศ์, คุณธนาคม, คุณมรกต, คุณศรีเมษ, คุณสุรพันธ์, คุณอรวรรณ, คุณอธิป, คุณบุญโชค, คุณจิรวัฒน์, คุณณัฐพล, คุณณัฐวุฒิ, คุณคนัย, คุณสราวุฒิ, คุณสมชาย

ภาคผนวก ก.

โปรแกรมภาษาแอสแซมบลีของภาคส่ง

```
                ORG 0000H

                LJMP INITIAL

                ORG 0013H

                LJMP STB

;*****************************************************************
;       RECIEVE HANDSHAKE FROM PRINTER (INTERRUPT RI)
;*****************************************************************

                ORG 0023H

                SETB P3.2
                CLR RI
                CLR IE.4
                MOV A,SBUF
                JNZ TRUTH
                MOV SBUF,R6

WAITS:  JBC TI,XXX
                AJMP WAITS

XXX:    SETB IE.4
                CLR P3.2
                RETI

TRUTH:  MOV A,#01H
                ADD A,R5
```

```
        MOV R5,A
        JNC DDD
        CLR C
        MOV A,#01H
        ADD A,R4
        MOV R4,A
        JNB PSW.2,DDD
        CLR PSW.2
        MOV R4,#00H
        CLR PSW.5

DDD:    SETB IE.4
        CLR 00H
        CLR P3.2
        RETI
;*************************************************************************
;       INITIAL
;*************************************************************************

INITIAL:  MOV SCON,#11000000B
        MOV TMOD,#00100000B
        MOV TH1,#0E8H
        SETB IE.7
        SETB IE.2
        SETB IE.4
        SETB TR1
        CLR 00H
        CLR 01H
        SETB PS
        MOV R2,#00H
```

```
        MOV R3,#00H
        MOV R4,#00H
        MOV R5,#00H
        CLR P3.2

;*************************************************************************
;       SEND DATA TO PRINTER BY SERIAL PORT
;*************************************************************************

PASS:   JNB 01H,EEE
        MOV A,P1
        MOV DPH,R2
        MOV DPL,R3
        MOVX @DPTR,A
        MOV A,#01H
        ADD A,R3
        MOV R3,A
        JNC FFF
        CLR C
        MOV A,#01H
        ADD A,R2
        MOV R2,A
        JNB PSW.2,FFF
        CLR PSW.2
        MOV R2,#00H
        SETB PSW.5
FFF:    SETB IE.2
        CLR P3.2
        CLR 01H

EEE:    JB PSW.5,CCC
```

```
        MOV A,R2
        XRL A,R4
        JNZ CCC
        MOV A,R3
        XRL A,R5
        JNZ CCC
        AJMP EEE

CCC:    MOV DPH,R4
        MOV DPL,R5
        MOVX A,@DPTR
        MOV R6,A
        JB PSW.0,ODD
        CLR RB8
        AJMP ZZZ

ODD:    SETB RB8

ZZZ:    CLR REN
        CLR IE.4
        MOV SBUF,A
        SETB 00H

WAIT:   JBC TI,YYY
        AJMP WAIT

YYY:    CLR TI
        SETB IE.4
        SETB REN

WWW:    JNB 00H,PASS
```

```
        AJMP WWW

;*************************************************************************
;       RECIEVE DATA FROM PC TO RAM (INTERRUPT INT1 BY STROBE)
;*************************************************************************

STB:    SETB P3.2
        CLR IE.2
        JNB PSW.5,AAA
        MOV A,R2
        XRL A,R4
        JNZ AAA
        MOV A,R3
        XRL A,R5
        JNZ AAA
        SETB 01H
        RETI

AAA:    MOV A,P1
        MOV DPH,R2
        MOV DPL,R3
        MOVX @DPTR,A
        MOV A,#01H
        ADD A,R3
        MOV R3,A
        JNC BBB
        CLR C
        MOV A,#01H
        ADD A,R2
        MOV R2,A
        JNB PSW.2,BBB
```

```
        CLR PSW.2
        MOV R2,#00H
        SETB PSW.5

BBB:    SETB IE.2
        CLR P3.2
        RETI

        END
```

```
                ORG 0000H

        LJMP INITIAL

;*************************************************************************
;       RECIEVE DATA FROM PC TO RAM (INTERRUPT INT1 BY STROBE)
;*************************************************************************

                ORG 0013H

        SETB P3.2       ;       busy  'high'
        MOV A,P1
        MOV DPH,R2
        MOV DPL,R3
        MOVX @DPTR,A    ;       move data from PC to RAM

;-----------INCREMENT POINTER DATA IN-----------
        INC R3
        CJNE R3,#00H,BBB
        INC R2
        CJNE R2,#80H,BBB
        MOV R2,#00H
;-----------------------------------

;-----------INCREMENT COUNTER-----------
BBB:    INC R7
        CJNE R7,#00H,AAA
        INC R6
        CJNE R6,#80H,AAA
;-----------------------------------
```

```
        RETI                ; RAM full   (busy 'high')

AAA:    CLR P3.2            ; RAM not full   (clear busy P3.2)
        RETI

;*************************************************************************
;       INITIAL
;*************************************************************************
db          'send1'
INITIAL:  MOV SCON,#11000000B
        MOV TMOD,#00100000B
        MOV TH1,#0E8H       ;  baud rate 1200
        MOV TL1,#0E8H
        SETB EA             ;  EA  = IE.7
        SETB EX1            ;  EX1 = IE.2 (INT1)
        SETB TR1            ; timer1 on
        MOV R2,#00H
        MOV R3,#00H
        MOV R4,#00H
        MOV R5,#00H
        MOV R6,#00H
        MOV R7,#00H
        CLR P3.2            ; busy 'low'

;*************************************************************************
;     SEND DATA TO PRINTER BY SERIAL PORT
;*************************************************************************

CHECKD: CJNE R7,#00H,CCC        ; check data in RAM
```

```
        CJNE R6,#00H,CCC
        SJMP CHECKD
                        ; have data in RAM
CCC:    MOV DPH,R4
        MOV DPL,R5
        MOVX A,@DPTR

        MOV SBUF,A              ; send data to serial port

;-----------INCREMENT POINTER DATA OUT-----------
        INC R5
        CJNE R5,#00H,YYY
        INC R4
        CJNE R4,#80H,YYY
        MOV R4,#00H
;------------------------------------

;-----------DECREMENT COUNTER-----------
YYY:    DEC R7
        CJNE R7,#0FFH,WAIT
        DEC R6
;------------------------------------

WAIT:   JBC TI,ZZZ
        AJMP WAIT

ZZZ:    CLR TI
        CLR P3.2                ; clear busy because RAM not full
        AJMP CHECKD

        END
```

ภาคผนวก ช.

โปรแกรมภาษาแอสแซมบลีของภาครับ

```
        ORG 0000H

        MOV SCON,#0D0H
        MOV TMOD,#22H
        MOV TH1,#0E8H
        SETB IE.7
        SETB IE.4
        SETB TR1
        MOV R2,#00H
        MOV R3,#00H
        MOV R4,#00H
        MOV R5,#00H
        CLR P3.2
        LJMP CHECKB
;*****************************************************************
;       INTERRUPT SERVICE ROUTINE
;*****************************************************************

        ORG 0023H

        CLR RI
        CLR IE.4
        CLR REN
        JNB PSW.5,AAA
        MOV A,R2
        XRL A,R4
        JNZ AAA
        MOV A,R3
        XRL A,R5
        JNZ AAA
```

```
CHECKBU:    JNB P3.2,CHECKFA
            AJMP CHECKBU

CHECKFA:    JB P3.3,DDD
            AJMP CHECKBU

DDD:        MOV DPH,R4
            MOV DPL,R5
            MOVX A,@DPTR
            MOV P1,A
            NOP
            CLR P3.4
            NOP
            NOP
            SETB P3.4
            NOP
            MOV A,#01H
            ADD A,R5
            MOV R5,A
            JNC AAA
            CLR C
            MOV A,#01H
            ADD A,R4
            MOV R4,A
            JNB PSW.2,AAA
            CLR PSW.2
            MOV R4,#00H
            CLR PSW.5

AAA:        MOV A,SBUF
```

```
        JB RB8,ODD
        JB PSW.0,SEND0

SEND1:  MOV DPH,R2
        MOV DPL,R3
        MOVX @DPTR,A
        MOV A,#01H
        ADD A,R3
        MOV R3,A
        JNC BBB
        CLR C
        MOV A,#01H
        ADD A,R2
        MOV R2,A
        JNB PSW.2,BBB
        CLR PSW.2
        MOV R2,#00H
        SETB PSW.5

BBB:    MOV SBUF,#0FFH
WAIT:   JBC TI,EEE
        AJMP WAIT
EEE:    SETB IE.4
        SETB REN
        RETI

ODD:    JB PSW.0,SEND1

SEND0:  MOV SBUF,#00H
        AJMP WAIT
```

```
;*********************************************************************
;*********************************************************************

CHECKB:     JNB P3.2,CHECKF
            AJMP CHECKB

CHECKF:     JB P3.3,PASS
            AJMP CHECKB

PASS:       JB PSW.5,CCC
            MOV A,R2
            XRL A,R4
            JNZ CCC
            MOV A,R3
            XRL A,R5
            JNZ CCC
            AJMP CHECKB

CCC:        MOV DPH,R4
            MOV DPL,R5
            MOVX A,@DPTR
            MOV P1,A
            NOP
            CLR P3.4
            NOP
            NOP
            SETB P3.4
            NOP
            MOV A,#01H
            ADD A,R5
            MOV R5,A
```

```
JNC CHECKB
CLR C
MOV A,#01H
ADD A,R4
MOV R4,A
JNB PSW.2,CHECKB
CLR PSW.2
MOV R4,#00H
CLR PSW.5
AJMP CHECKB
END
```

```
                ORG 0000H

        MOV SCON,#11010000H
    MOV TMOD,#00100000H
        MOV TH1,#0E8H          ; baud rate 1200
        MOV TL1,#0E8H
        SETB EA                ; EA = IE.7
        SETB ES                ; ES = IE.4  (serial interrupt)
    SETB TR1
    MOV R0,#21H
        MOV R1,#21H
        MOV R2,#00H
        CLR P3.2
        LJMP CHECKD
;*************************************************************************
;           INTERRUPT SERVICE ROUTINE BY RI
;*************************************************************************

                ORG 0023H

        CLR RI

        MOV A,SBUF
    MOV @R0,A       ; move data to store in RAM

;————————INCREMENT POINTER DATA IN————————————
        INC R0
        CJNE R0,#80H,BBB
        MOV R0,#21H
```

```
;-------------------------------------

;-----------INCREMENT COUNTER-----------------
BBB:        INC R2
            CJNE R2,#5FH,AAA
;-------------------------------------

AAA:        RETI

;*************************************************************************
;           SEND DATA TO PRINTER
;*************************************************************************

CHECKD:     CJNE R2,#00H,CHECKB     ; check data in RAM
            SJMP CHECKD
                                    ; have data in RAM
CHECKB:     JNB P3.2,CHECKF         ; check busy
            AJMP CHECKB

CHECKF:     JB P3.3,PASS            ; check fault
            AJMP CHECKB

PASS:       MOV A,@R1
            MOV P1,A                ; send data to printer
            NOP
            CLR P3.4                ; strobe
            NOP
            NOP
            SETB P3.4
            NOP
```

```
;————————INCREMENT POINTER DATA OUT————————
                INC R1
                CJNE R1,#80H,YYY
                MOV R1,#21H
;————————————————————————————

;————————DECREMENT COUNTER————————————
YYY:            DEC R2
                CJNE R2,#5FH,CHECKD
;————————————————————————————

                AJMP CHECKD

                END
```

ภาคผนวก ค.

วงจรไมโครคอนโทรลเลอร์ และข้อมูลที่ใช้ในการทำโครงงาน

Parallel Interface Lines Description

Table E-1 describes the signal lines and pin assignments.

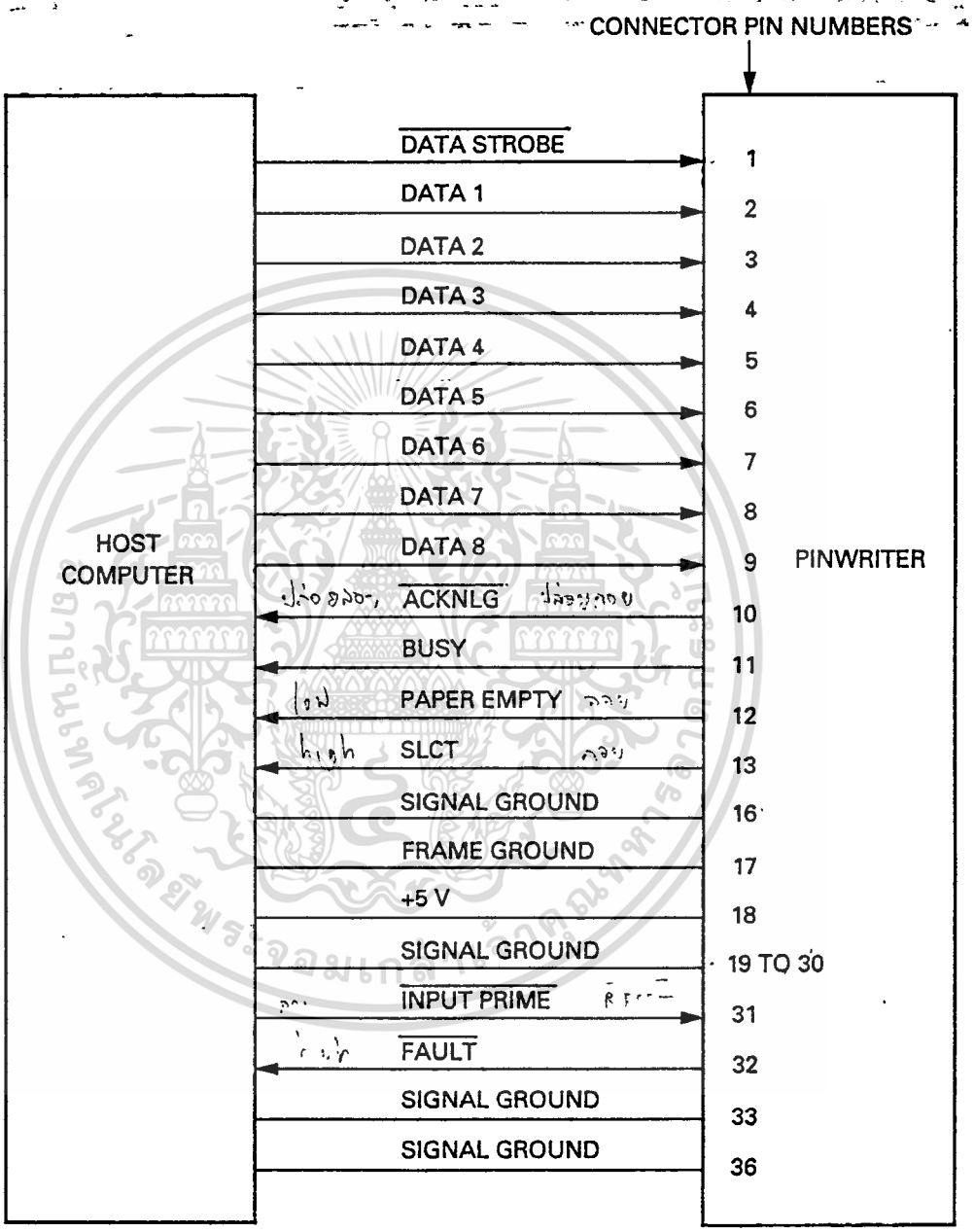


Figure E-1 Parallel Interface Lines Diagram

Table E-1 Parallel Interface Lines Description

SIGNAL PIN	SIGNAL	DIRECTION	DESCRIPTION
1	$\overline{\text{Data Strobe}}$	To printer	Strobes a data byte to the printer. The minimum pulse width of the signal is 1 µs, active low.
2	Data 1	To printer	Data lines 1 to 8 transmit ASCII code. The lines are high (true) at logic 1 and low (false) at logic 0. The minimum pulse width of the signal is 2 µs.
3	Data 2		
4	Data 3		
5	Data 4		
6	Data 5		
7	Data 6		
8	Data 7		
9	Data 8		
10	$\overline{\text{Acknlg}}$	From printer	An 8-µs pulse on this status line indicates that the printer received a data byte and it can receive another.
11	Busy	From printer	This signal goes high when the printer is busy because • the buffer is full, or • it is initializing, or • it is deselected, or • a printer fault has occurred, or • a data is received and next data can not be received yet.
12	Paper Empty	From printer	This status line goes high when the printer is out of paper.
13	Slct	From printer	This status line goes high when the printer is selected. The line is low when the printer is deselected.

Table E-1 Parallel Interface Lines Description (cont'd)

SIGNAL PIN	SIGNAL	DIRECTION	DESCRIPTION
14, 15	—	—	Unused.
16	SG	—	Signal ground.
17	FG	—	Frame ground.
18	+5 V	—	+5 V.
19, 20 21, 22 23, 24 25, 26 27, 28 29, 30	SG		Ground. Pin 19 is ground for $\overline{\text{Strobe}}$. Pins 20 to 27 are ground for data 1 to 8, pin 28 is ground for Acknlg, pin 29 is ground for Busy, and pin 30 is ground for Input Prime.
31	$\overline{\text{Input Prime}}$	To printer	A low on this line initializes the printer. The signal pulse width must be more than 15 µs.
32	$\overline{\text{Fault}}$	From printer	This line goes low when the printer is • out of paper, • deselected, or • in a fault state.
33	SG	—	Signal ground.
34	—	—	Unused.
35	—	—	Unused.
36	SG	—	Signal ground.

Parallel Interface Timing

The timing relationships of the signals for data input, paper empty, and deselect are shown below.

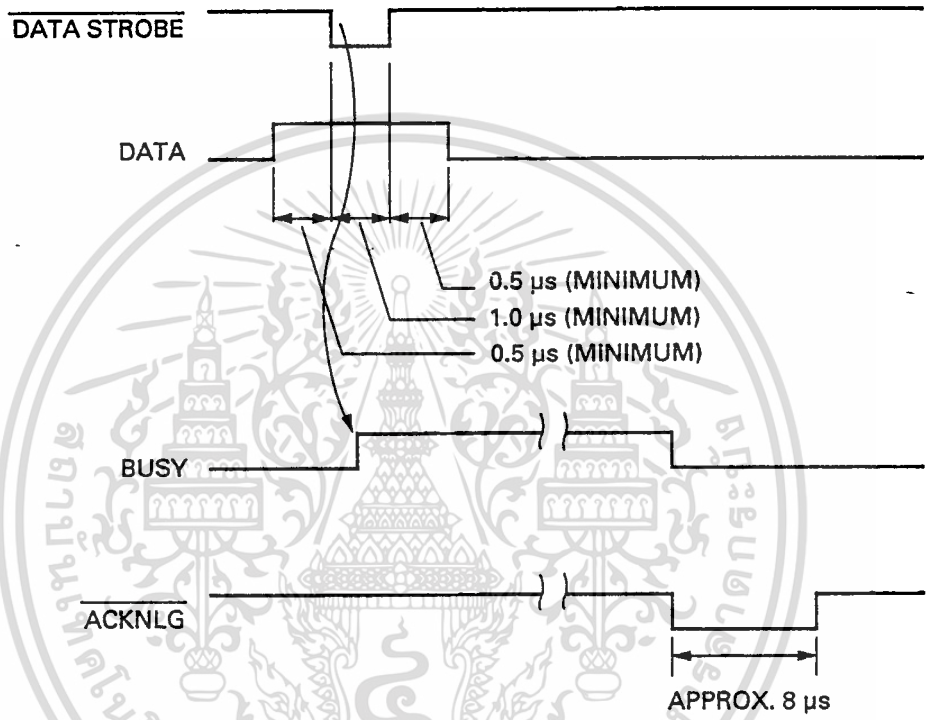


Figure E-2 Timing for Data Input Signals

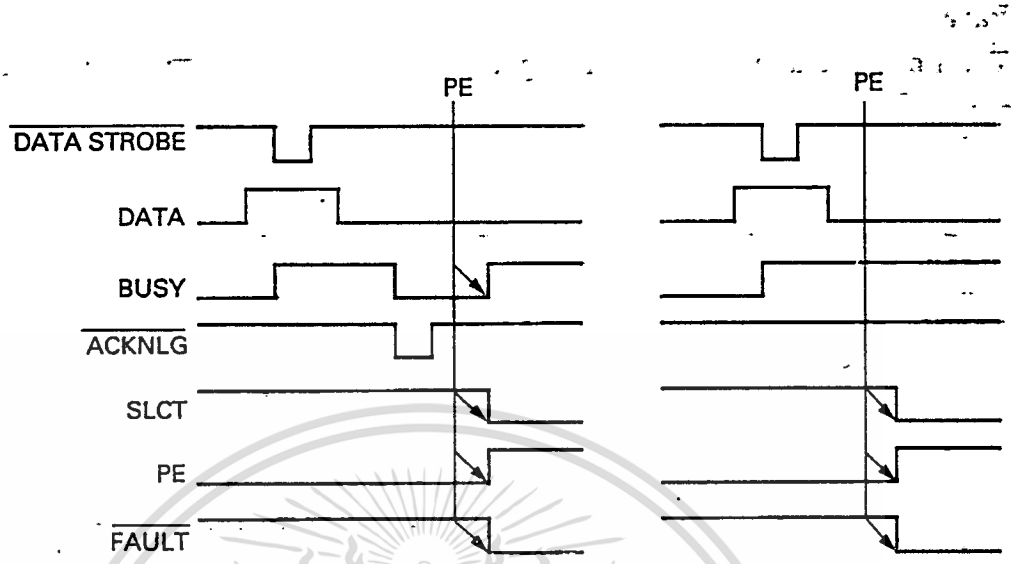


Figure E-3 Timing for Paper Empty Signals

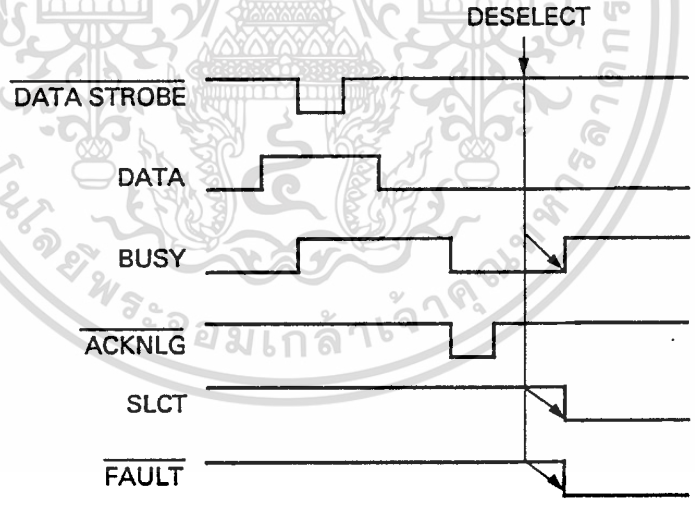
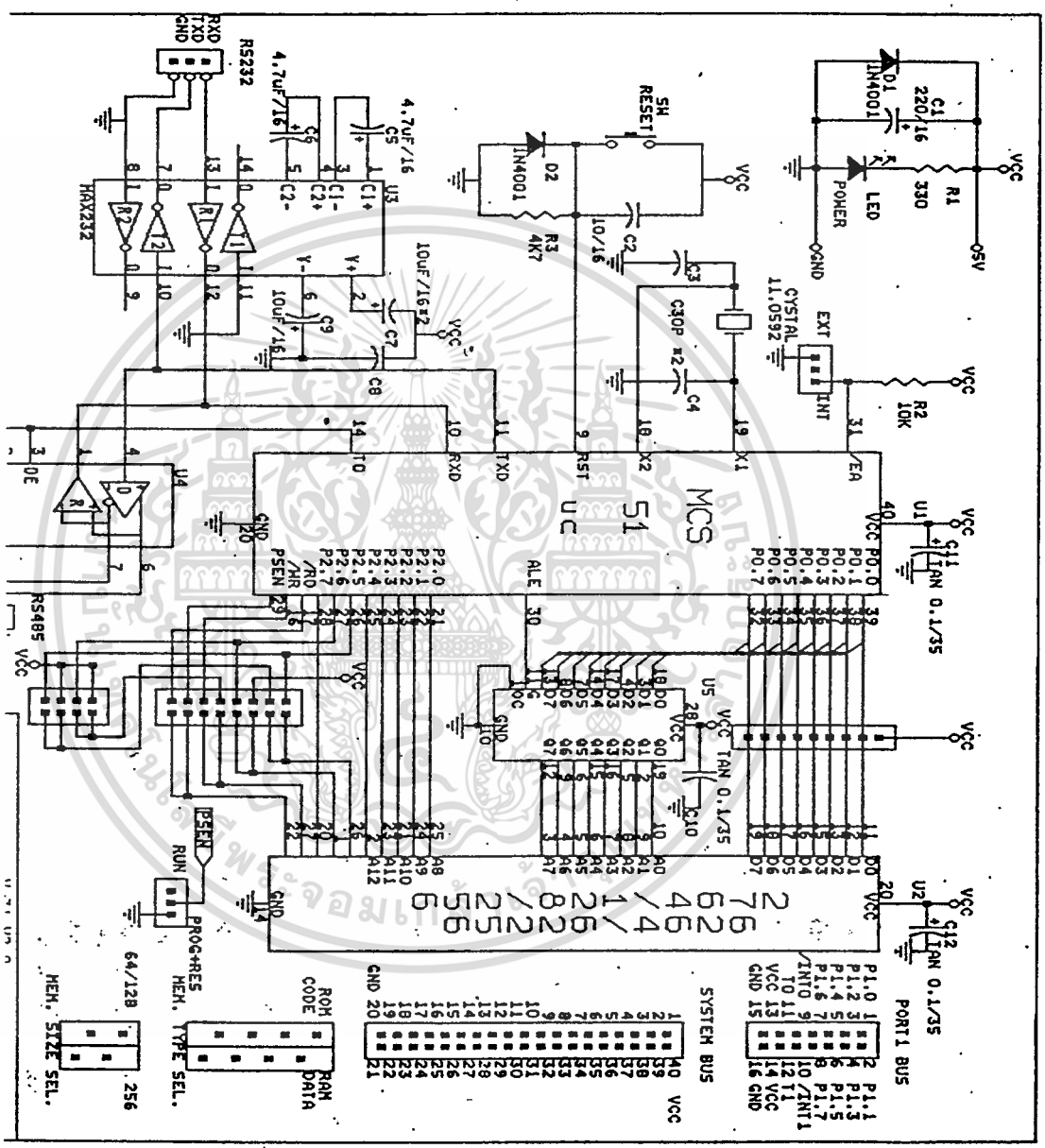
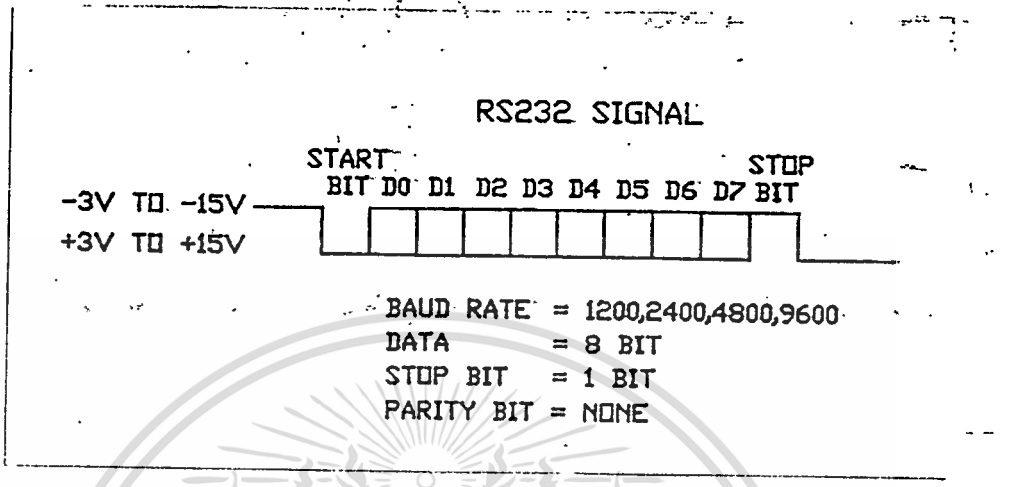
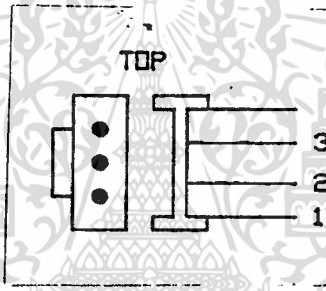


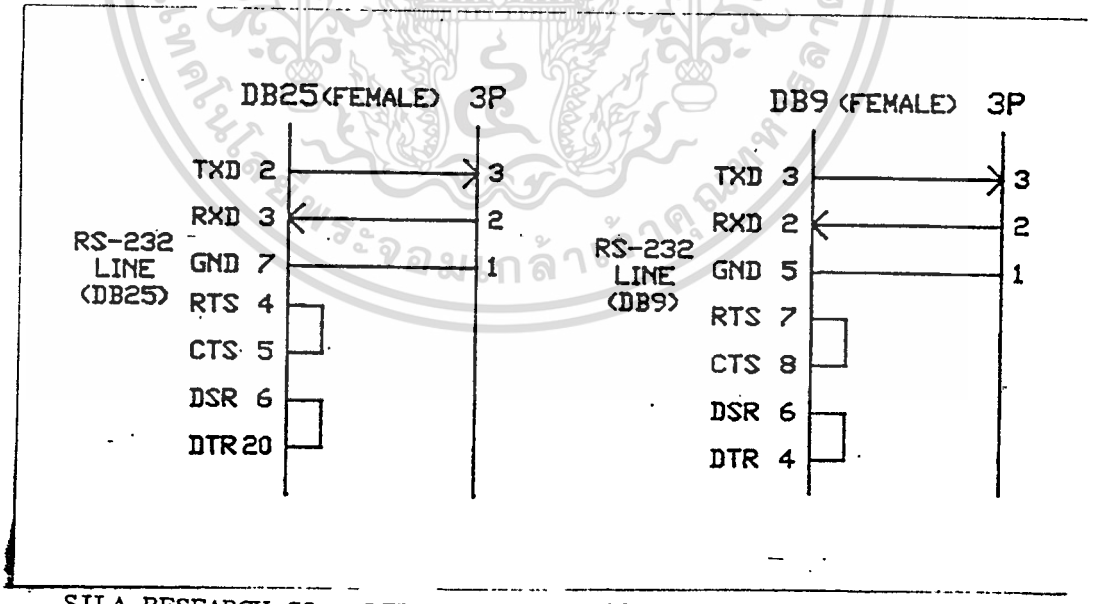
Figure E-4 Timing for Deselect Signals



ภาพแสดงสัญญาน RS232 และการต่อสาย







ภาพแสดงรายละเอียดของ CONNECTOR และ CHIP