การเช็คเงินโดยอัตโนมัติ

AUTOMATIC BILLING

โดย

นางสาวศิริพร ภาวะเวช 33100376

นางสาวสมนึก จันภู 33100396

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

การเช็คเงินโดยอัตโนมัติ

AUTOMATIC BILLING

โดย

นางสาวศิริพร ภาวะเวช 33100376

นางสาวสมนึก จันภู 33100396

อาจารย์ที่ปรึกษา สมยศ จุณณะปิยะ

บทคัดย่อ

โครงงานนี้เสนอหลักการทำงาน และการออกแบบสร้างวงจร Automatic Billing ซึ่งเป็นส่วนหนึ่งในการบันทึกข้อมูลคิดค่าบริการในโรงแรม ทำให้ลดความล่าช้า ในการเช็คเอาท์ เพราะไม่ต้องใช้พนักงานขึ้นไปตรวจเช็คห้อง โดยในห้องจะมีช่องบรรจุเครื่องดื่มที่ออกแบบพิเศษ (ในโครงงานนี้ไม่ได้กล่าวถึงการจัดทำช่องบรรจุอาหาร)และมีอุปกรณ์ส่งสถานะการเปิด ปิดช่องบรรจุเครื่องดื่ม คือ ตัว ACS(Automatic Check Service) ซึ่งทำหน้าที่ส่งข้อมูล แบบอะซิงโครนัส และสามารถตั้ง แอดเดรสประจำตัวได้ ทำให้สามารถรับข้อมูลของห้องต่างๆ ได้อย่างถูกต้อง และข้อดีของระบบนี้อีกคือ การนำสัญญาณส่งไปในสายส่งสัญญาณเพื่อแปลงเป็นไฟตรง +5 โวลต์ เลี้ยงวงจรทั้งหมด และสามารถตั้งแอดเดรสได้ถึง 128 ห้อง

ABSTRACT

This project propose the theory and principle of automatic billing circuit, part of store value of service in hotel. This system save time for check out because official does not check room before check out.In room have special boxes of drink and ASC (Automatic Check Service) is asynchronous transmitted/received, send status of open/close of boxes of drink. We can receive data from other room correctly and the advantage of this system is to use the transmitting signal in signal wire, which is converted to be the DC 5 volts for supplying the automatic check service and can connect parallel 128 room.

สารบัญ

บทที่ 1
บทนำ

ในการให้บริการของโรงแรมชั้นหนึ่งจะเน้นที่การให้บริการแก่ลูกค้า โดยให้ลูกค้าได้รับความพึงพอใจที่สุด ไม่ว่าจะเป็นการให้บริการอาหาร, เครื่องดื่มในห้องพัก หรือแม้กระทั่งการให้บริการในเวลาเช็คเอ้าท์ซึ่งลูกค้าต้องการที่จะได้บริการที่รวดเร็วทันใจ

ในอดีตที่ผ่านมาจะประสบปัญหาความล่าช้า ในการให้บริการแก่ลูกค้าอย่างมาก บางครั้งอาจจะใช้เวลาเป็นชั่วโมงในการเช็คเอ้าท์เพราะว่าต้องใช้พนักงานของโรงแรมขึ้นไปตรวจห้องพักของลูกค้าว่าได้ใช้บริการอาหารหรือเครื่องดื่มในห้องพักที่ทางโรงแรมจัดไว้ให้ชนิดใดบ้าง เป็นมูลค่าเท่าไร แล้วเอามารวมกับค่าห้องพักและบริการอื่น ๆอีก จึงทำให้ใช้เวลานาน จากปัญหาอันนี้จึงทำให้ได้แนวความคิดที่จะแก้ปัญหาความล่าช้าในการเช็คเอ้าท์ โดยการใช้เทคโนโลยีเข้ามาช่วยแทนการใช้พนักงาน ขึ้นไปตรวจเช็คอาหารและเครื่องดื่ม โดยการใช้ระบบการเก็บข้อมูลไว้ในคอมพิวเตอร์ ข้อมูลที่จัดเก็บเป็นเครื่องดื่มชนิดต่าง ๆเมื่อเช็คเอ้าท์ก็จะเรียกข้อมูลนั้นออกมาประมวลผล และพิมพ์ใบเสร็จออกมาทันที

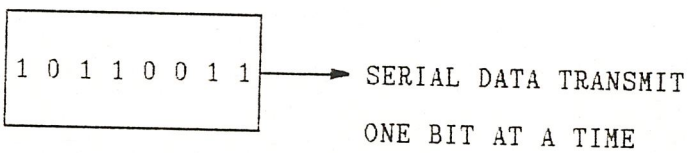
โดยในห้องพักจะออกแบบช่องบรรจุเครื่องดื่มให้มีลักษณะพิเศษ ที่สามารถจะบอกสถานะการเอาเครื่องดื่มชนิดนั้น ๆไปบริโภค หรือว่ายังคงอยู่ไม่ได้ถูกนำไปบริโภค คล้ายกับลักษณะลิ้นชัก การเปิดลิ้นชักก็คือการแสดงสถานะเอาสินค้านั้นไปบริโภค และจะมีอุปกรณ์อีกตัวหนึ่งเรียกว่าเอซีเอส (ACS:AUTOMATIC CHECK SERVICE) ทำหน้าที่เก็บสถานะการเปิดลิ้นชัก เมื่อมีสัญญาณมาจากคอมพิวเตอร์เพื่อถามว่าห้องนั้น ๆอยู่ในสถานะอะไร เอซีเอสก็จะส่งสัญญาณตอบกลับไป ว่ามีสถานะอย่างไร ซึ่งรายละเอียดของเอซีเอสจะกล่าวในบทต่อไป

บทที่ 2

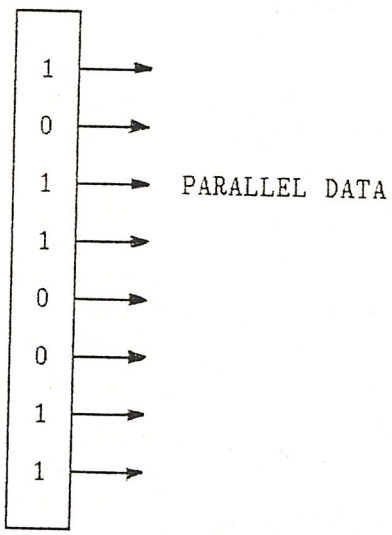
การสื่อสารข้อมูลแบบอนุกรม

การติดต่อสื่อสารแบบอนุกรมเป็นการส่งข้อมูลทีละบิทแบบต่อเนื่องกันไปการสื่อสารแบบขนานต่างกับอนุกรม คือแบบขนานจะส่งข้อมูลครั้งละหลายบิทในเวลาเดียวกัน ตัวอย่างของการสื่อสารแบบขนาน เช่น ซีพียู อ่านและเขียนข้อมูลกับหน่วยความจำ ซึ่งข้อมูลที่ติดต่อกันในไมโครโปรเซสเซอร์ทั้งหมดจะเป็นแบบขนาน

ความแตกต่างของการรับส่งข้อมูลแบบขนานกับอนุกรม แสดงได้ดังรูปที่ 1A ซึ่งรูปแสดงการส่งข้อมูลแบบอนุกรมโดยใช้สายเพียงเส้นเดียวในการส่งข้อมูลทุกบิท และรูปที่1B แสดงการส่งข้อมูลแบบขนานซึ่งใช้สาย 1เส้นสำหรับข้อมูล 1บิท



(A)การส่งข้อมูลแบบอนุกรม



(B)การส่งข้อมูลแบบขนาน

รูปที่ 1 แสดงการส่งข้อมูลแบบอนุกรมและขนาน

SERIAL TIMING

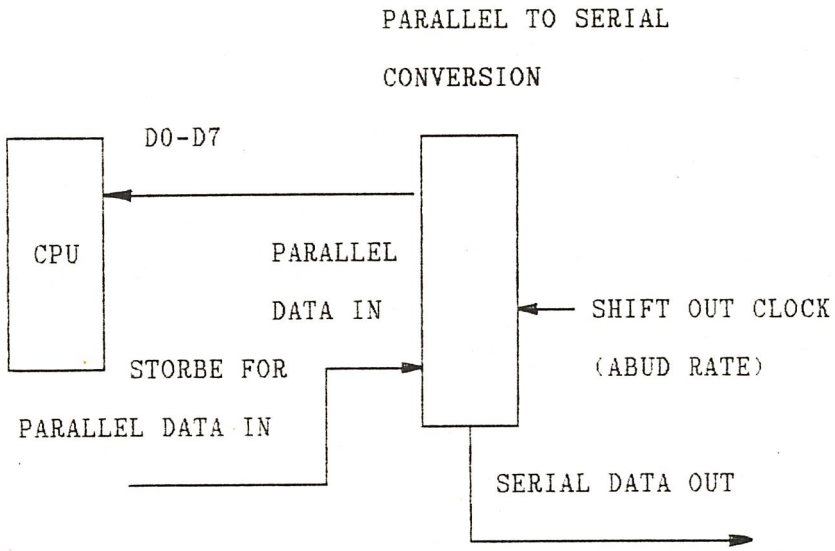
การส่งข้อมูลแบบอนุกรมดังที่กล่าวมาแล้ว ยังมีโครงสร้างที่สำคัญอีกหลายอย่างที่ใช้ในการส่งข้อมูลหนึ่งในโครงสร้างเหล่านั้นคือ ความถี่ในการส่งข้อมูลออกไป ซึ่งความถี่นี้ เรียกว่า Baud rate ซึ่งเป็นตัวกำหนดจำนวนบิทที่ส่งไปใน 1 วินาที Baud rate มีค่าต่างๆกันดังนี้ 110, 150, 300, 600, 900... สมมุติว่าเราต้องการส่งข้อมูล 8 บิทที่ Baud rate 2400 หมายถึงข้อมูลจะถูกส่งออกไปแบบอนุกรม โดยแต่ละบิทมีความกว้างเท่ากับ 1/2400 วินาที จากเวลาดังกล่าว เราสามารถหาเวลาในการส่งข้อมูลขนาด 8 บิท ได้โดยใช้ 8*(1/2400) ซึ่งได้เท่ากับ 3,328 ไมโครวินาที แต่ถ้าวิธีการส่งข้อมูลแบบขนาน ในการส่งข้อมูล 8 บิทออกไปพร้อมกัน จะใช้เวลาน้อยกว่า 1/2400 วินาที

การเปลี่ยนข้อมูลแบบขนานเป็นอนุกรม

หลักการส่งข้อมูลแบบอนุกรมจะต้องเปลี่ยนข้อมูลจากขนานเป็นอนุกรมก่อนแล้วจึงส่งออกไปซึ่งมีหลักการทำงานดังนี้

1. เก็บข้อมูลขนาด 8 บิทมาเก็บในชิพรีจีสเตอร์
2. เลื่อนข้อมูลออกไปทีละบิทในเวลาที่ถูกต้องตาม Baud rate

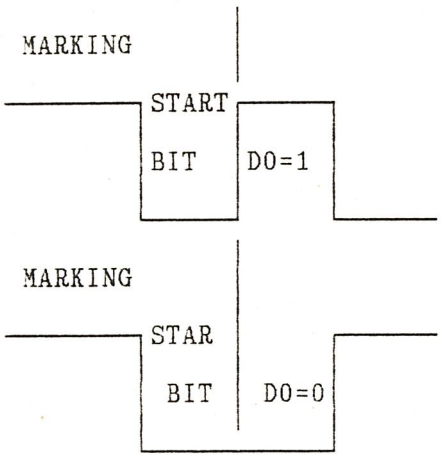
การทำงานในลักษณะนี้แสดงได้ดังรูปที่ 2 ซึ่งเริ่มแรก ข้อมูลส่งจากไมโครโปเซสเซอร์ แบบขนานเข้ามายังชิพรีจีสเตอร์ แล้วจึงส่งออกไปเป็นแบบอนุกรม โดยส่งบิท D0 ออกไปเป็นบิทแรกและส่ง D7 ออกไปเป็นบิทสุดท้าย



รูปที่ 2 แสดงการเปลี่ยนข้อมูลจากขนานเป็นอนุกรม

สตาร์ทบิท (START BIT)

ในการส่งข้อมูลแบบอนุกรมนั้นด้านรับจะต้องรับและแปลความหมายของข้อมูลที่รับมาได้ถูกต้อง ดังนั้นการส่งข้อมูลจึงจำเป็นต้องส่งสตาร์ทบิท เข้ามาอีก 1 บิทเพื่อทำหน้าที่บอกให้ทางด้านรับทราบว่าข้อมูลใหม่กำลังตามมา และทำหน้าที่เป็น Clock synchornize ในการรับข้อมูล แต่ละตัวอักษร ซึ่งเหมือนข้อมูล 1ไบต์ ที่ส่งแบบขนาน เมื่อเราเพิ่มสตาร์ทบิทเข้าไป จะทำให้การส่งข้อมูล 1 ตัวอักษรมีมากกว่า 8 บิทการส่งข้อมูลแบบอนุกรมถ้าหากไม่มีการส่งข้อมูลออกไปเราจะเรียกช่วงนี้ว่า Marking ซึ่งเราใช้ช่วง Marking มีค่าลอจิก 1 สตาร์ทบิทที่เราเพิ่มเข้าไปต้องมีค่าตรงข้ามกับ Marking คือจะมีค่าลอจิก 0 สตาร์ทบิท ที่ใส่เพิ่มเข้าไปจะมาขนาดความกว้างเท่ากับบิทข้อมูล ดังแสดงในรูปที่ 3 ด้านรับจะต้องตรวจสอบสตาร์ทบิท และรับข้อมูลที่ตามมา

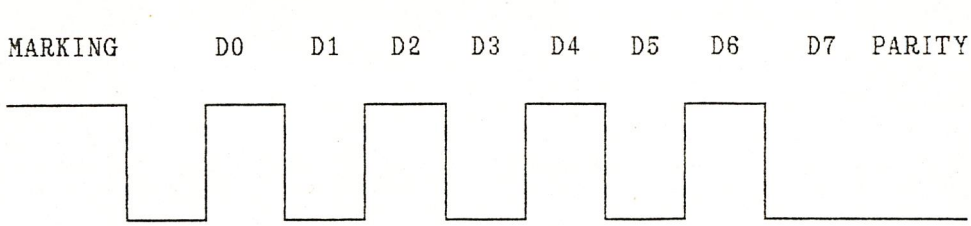


รูปที่ 3 แสดง START BIT

พาริตี้บิท(PARITY BIT)

พาริตี้บิทเป็นบิทตรวจสอบที่เพิ่มเข้าไปทางด้านส่งเพื่อให้ด้านรับตรวจสอบความถูกต้องของข้อมูล ข้อมูลที่ส่งออกไปทางด้านส่งนั้น จะประกอบด้วยจำนวนบิท ที่มีค่าลอจิก 1 ซึ่ง อาจจะเป็นคู่หรือคี่ (EVEN or ODD) เช่นข้อมูล 8 บิทค่า 54H (01011000) จะมีลอจิก 1 จำนวน 3 บิท และค่า 55H (01010101) มีลอจิก 1จำนวน 4 บิท ด้านรับจะรับข้อมูลและทำการนับจำนวนของลอจิก 1 ว่า เป็นคู่หรือคี่ สมมุติด้านรับถูกกำหนดให้รับข้อมูลที่มีค่าลอจิก 1 เป็นเลขคู่ ค่า 55H จะถูกต้องและค่า 54H จะผิด ดังนั้นด้านส่งจะต้องเพิ่ม 1 เข้าไปรวมกับข้อมูลอีก 1 บิทข้อมูลใหม่ ก็จะเป็น 9 บิท ก่อนที่จะส่งออกไปค่าของพาริตี้บิทอาจจะเป็น 0 หรือ 1 ก็ได้เพื่อทำให้จำนวนบิทในข้อมูลทั้งหมดมีลอจิก 1เป็นเลขคู่ รูปที่4 แสดงพาริตี้บิท ในการส่งข้อมูล 55H

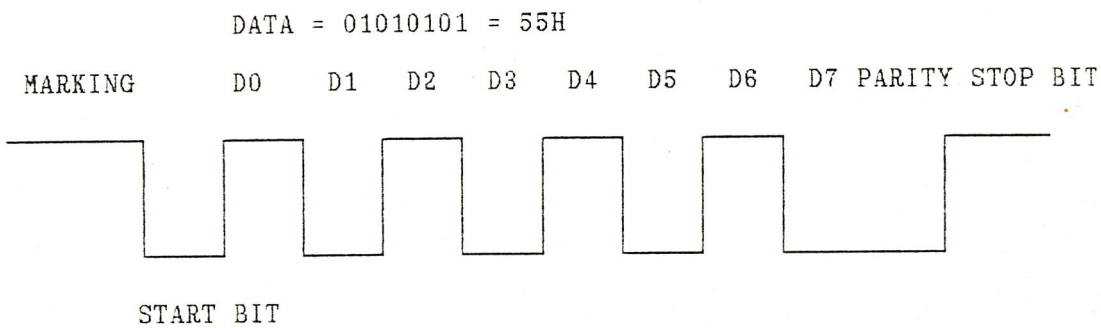
DATA = 01010101 =55H



รูปที่ 4 แสดง PARATY BIT

สต๊อพบิท (STOP BIT)

บิทสุดท้ายที่เพิ่มเข้าไปในการส่งข้อมูลปแบบอนุกรมคือสต๊อพบิท ทางด้านรับจะคอยตรวจสอบสต๊อพบิท ซึ่งอยู่ท้ายสุดของข้อมูล จำนวนของสต๊อพบิทอาจจะมี 1, 1/2 หรือ 2 บิทก็ได้ รูปที่5 แสดงการส่งข้อมูลขนาด 8 บิทที่รวมกับสตาร์ทบิท, พาริตี้บิท และสต๊อพบิทอีก 2 บิทดังนั้นจำนวนบิททั้งหมดจะเท่ากับ 12 บิท ที่ Baud rate 2400 เวลาที่ใช้ในการส่งข้อมูลเท่ากับ 12*416 ไมโครวินาที หรือ 4.99 มิลลิวินาที



รูปที่ 5 แสดงการส่งข้อมูล 1 ตัวอักษร

มาตรฐาน RS-232C

RS-232 เป็นมาตราฐานการสื่อสารข้อมูลแบบอนุกรมที่กำหนดโดย EIA (Electronics Industries Association)ได้ถูกตีพิมพ์ในปี ค.ศ.1969 ตัวอักษร RS แทน" Recomm Standard" 232 แทนหมายเลขของมาตราฐาน ส่วนอักษร C แสดงให้รู้มาตรฐานได้รับการแก้ไขกี่ครั้ง

-ถูกออกแบบให้ใช้กับอุปกรณ์พวก discrete

-ใช้การอินเตอร์เฟสแบบ Unbalanced

-ในแต่ละเซอร์กิตใช้ลวดนำในการนำสัญญาณ 1 เส้น และมีสายกราวน์รวมของทุกเซอร์กิตเพียงเส้นเดียว

-อัตราเร็วในการส่งข้อมูลมีค่า <20 kbps

-ระยะทางสูงสุดที่ใช้ในการส่งข้อมูลมีค่า < 15 เมตร

มาตรฐาน RS 232 C ที่ถูกตีพิมพ์โดย EIA ได้กล่าวถึงการสื่อสารข้อมูล ระหว่าง Data Terminal Equipment(DTE) และ Data Communication Equipment (DCE)คำจำกัดความของ DCE และ DTEได้คัดมาจากคำแปลศัพท์ (glossary) ในหนังสือ "Technical Aspect of Data Communication"ซึ่งเขียนโดย John Mcnamara(Digital press, 1977) ดังนี้

DCE: อุปกรณ์ที่มีฟังก์ชันการทำงานต่างๆ ที่ทำให้เกิดการเชื่อมต่อยังดำรงต่อไป และยุติการเชื่อมต่อ นอกจากนี้ยังใช้เปลี่ยนลักษณะของสัญญาณ และสร้างรหัสสัญญาณต่างๆ ที่จำเป็นต้องใช้ในการสื่อสารข้อมูลระหว่าง DTE และ Data circuit โดย DCE อาจเป็นส่วนใดส่วนหนึ่ง ของคอมพิวเตอร์หรือไม่ก็ได้

DTE: 1.เป็นอุปกรณ์ที่ประกอบไปด้วยตัวส่งข้อมูล(data source)หรือตัวรับข้อมูล(data sink) หรือเป็นทั้งตัวส่งและตัวรับข้อมูลก็ได้

2.เป็นอุปกรณ์ที่ประกอบด้วย function unit ต่อไปนี้ control logic,buffer store และอุปรณ์อินพุทหรือเอ้าท์พุทจำนวนหนึ่งตัวหรือมากกว่าก็ได้ หรือรวมเครื่องคอมพิวเตอร์เข้าไปด้วยก็ได้ DTE อาจจะรวมส่วน errror control synchronization และความสามารถในการบ่งหรือระบุว่าต้องการเกี่ยวข้องกับอุปกรณ์ตัวใด(station identification capability)เข้าไปด้วยก็ได้

ถ้าเทอร์มินัล และคอมพิวเตอร์ของเราเป็น DCE และ DTE ทั้งคู่เราจะทำการสื่อสารข้อมูลได้อย่างไรปัญหาที่เกิดขึ้นเราแก้ไขได้โดยใช้สายเคเบิล (Cable) ที่เรียกว่า

"null modem"

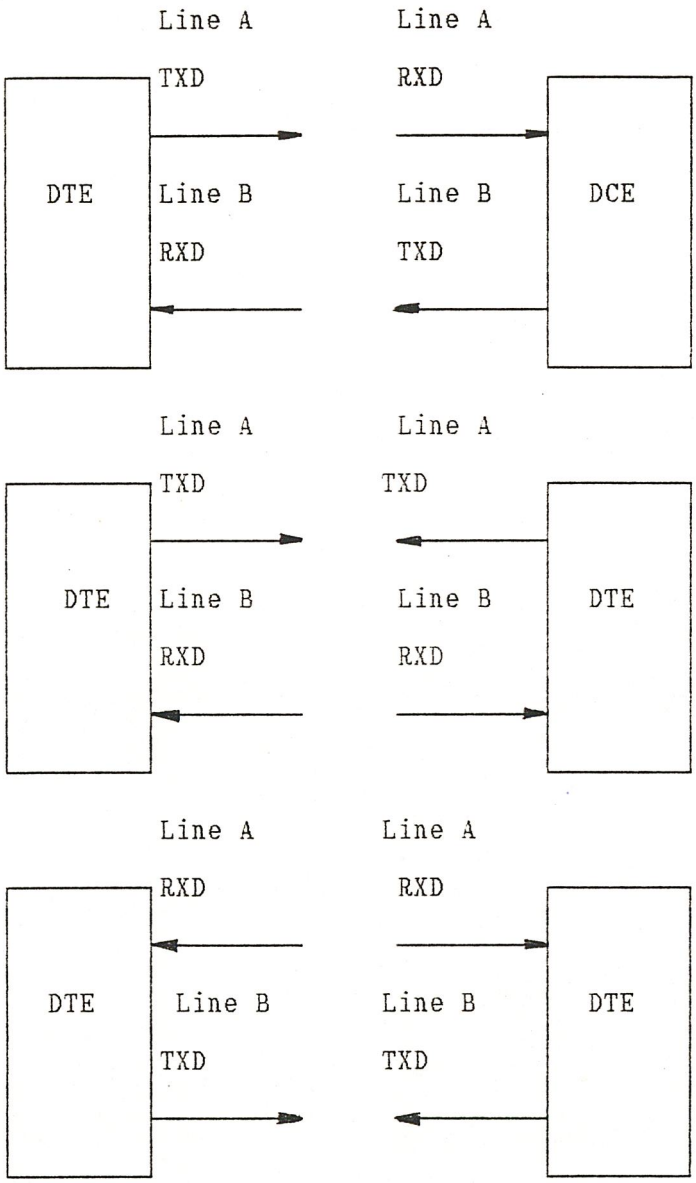
1.null หมายความว่า อุปกรณ์ตัวนี้ไม่สามารถทำงานอะไรได้เราใช้อุปกรณ์ตัวนี้ เมื่อต้องการเปลี่ยนทิศทางการเคลื่อนที่ของข้อมูลเท่านั้น

2.modem แสดงว่าอุปกรณ์ตัวนี้เป็น DCE เหตุนี้เราจึงใช้ null modem แทรกเข้าไประหว่าง DTE สองตัว เพื่อทำให้เราสามารถทำการสื่อสารข้อมูลโดยผ่าน RS-232C ได้

สำหรับลักษณะของสาย (line) ที่ใช้ในการรับและส่งข้อมูลของ DTE และ DCE เป็นดังนี้

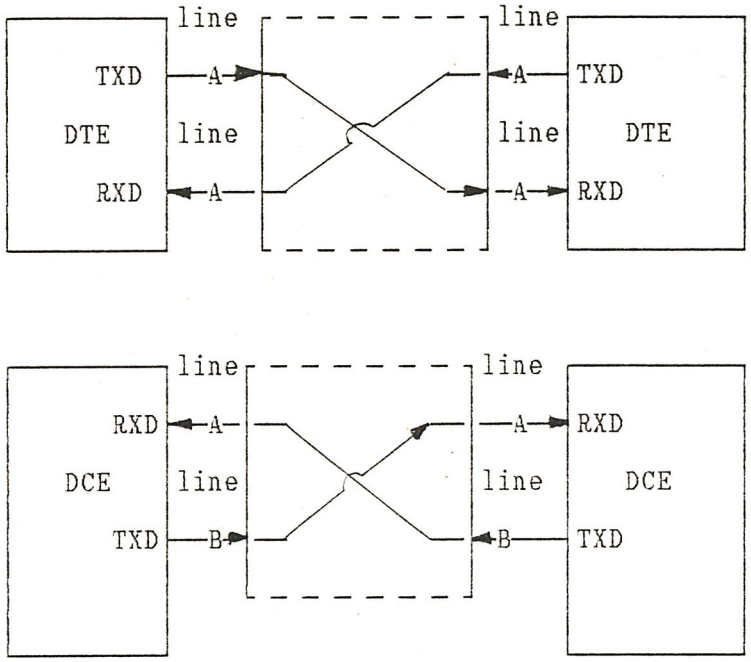
1.สายที่ใช้ในการส่งและรับข้อมูลของ DTE และ DCE มีอยู่สองเส้น แต่ละเส้น จะมีทิศทางการเคลื่อนที่ของข้อมูลกำหนดได้ต่างทิศทางกัน

2.DTE จะส่งข้อมูลทาง line A และ DCE จะรับข้อมูลทาง line A เช่นเดียวกัน DCE จะOส่งข้อมูลทาง line B และ DTE รับข้อมูลทาง line B ดังแสดงในรูปที่ 8



จากรูปที่ 6 ลักษณะการส่งข้อมูลและรับข้อมูลของ DTE และ DCE

จากรูปที่6 ถ้าDTE 2 ตัว ทำการแลกเปลี่ยนข้อมูลกัน ข้อมูลจะถูกส่งออกสองทาง Line A และรับทาง Line B ทั้งคู่ ดังนั้นการสื่อสารข้อมูล จะไม่สามารถเกิดขึ้นได้ ในกรณีที่ DCE 2 ตัวทำการแลกเปลี่ยนข้อมูลกัน เมื่อเราต่อ Line A เข้าด้วยกันและต่อ Line B เข้าด้วยกัน ข้อมูลใน Line B จะต้านกันเอง ส่วนใน Line A ไม่มีข้อมูลที่จะรับ ปัญหานี้แก้ได้โดยการใช้ NULL MODEM เข้ามาช่วย NULL MODEM CABLE จะทำการไขว้ Line A เข้ากับ Line B ดังรูปที่ 7



รูปที่ 7 ลักษณะการทำงานของ null modem ที่ใช้การอินเตอร์เฟส DTE หรือ DCE สองตัว

เราใช้มาตราฐาน RS-232-C ในการสื่อสารข้อมูลแบบอนุกรมระหว่าง DCE กับ DTE โดยการส่งข้อมูลจะถูกกำหนดให้อยู่ระหว่าง 0 ถึง 20000 บิทต่อวินาที ในการประยุกต์ใช้งาน RS-232C เราเร็วสูงสุดที่ควรใช้จะมีค่าไม่เกิน 19.2 กิโลบิทต่อวินาที

คุณสมบัติของสัญญาณไฟฟ้า

สัญญาณที่ขาทุกขาที่คอนเนคเตอร์ของ RS-232-C จะเป็นสภาวะใดสภาวะหนึ่งในแต่ละคู่ของคู่ต่อไปนี้

MARK/SPACE

ON/OFF

LOGIC 0/LOGIC 1

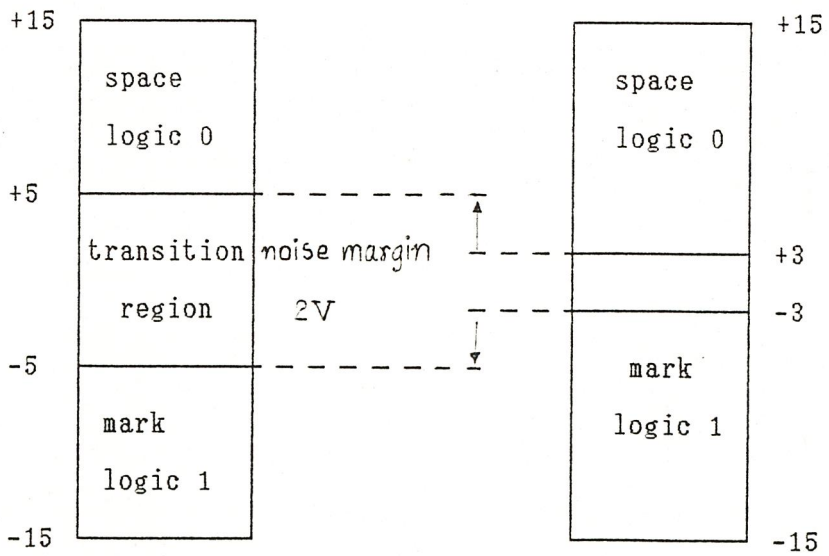
ความสัมพันธ์ระหว่างสถานะของสัญญาณคู่ต่างๆ กับระดับแรงดันได้แสดงไว้ในตารางที่ 1

STATUS	SIGNAL VOLTAGE	
	$-25V<V_1<-3V$	$3V<V_1<25V$
Binary logic		
State	1	0
Signal condition	MARK	SPACE
Function	OFF	ON

ตารางที่ 1

ช่องของระดับแรงดันระหว่าง -3 ถึง +3 โวลต์ จะเป็นช่วงของการเปลี่ยนลอจิกดังนั้นจึง ไม่มีการระบุสภาวะของสัญญาณในช่วงนี้ ในการแทนลอจิก 1 หรือสถานะ mark ตัวขับสัญญาณ(driver) ต้องจ่ายจ่ายแรงดันระหว่าง-5 ถึง-15 โวลต์ส่วนในการแทนลอจิก 0 หรือ space ตัวขับสัญญาณต้องจ่ายแรงดันระหว่างถึง +5 ถึง +15 โวลต์

RS-232C ยอมให้มี noise margin ได้ไม่เกิน 2 โวลต์ สำหรับความสัมพันธ์ระหว่างระดับแรงดันและสถานะได้แสดงไว้ในรูปที่ 8



รูปที่ 8 คุณสมบัติทางไฟฟ้าของการอินเตอร์เฟสแบบ RS-232-C

จากรูปจะเห็นได้ว่า ถ้า line driver หรือตัวกำเหนิดสัญญาณต้องการส่งลอจิก 0 line driver จะต้องจ่ายแรงดัน ระหว่าง +5 ถึง +15 โวลต์ ส่วน line receive หรือตัวรับสัญญาณปลายทางจะถือว่าแรงดันที่อยู่ในช่วง +3 ถึง +15 โวลต์ แทนลอจิก 0 จากการเปรียบเทียบระดับสัญญาณของตัวส่งและตัวรับจะเห็นว่า RA-232C ยอมให้มีการ drop ของสัญญาณในช่วง 2 โวลต์เกิดขึ้นได้สำหรับในด้านการส่งลอจิก 1 ก็เป็นเช่นเดียวกัน

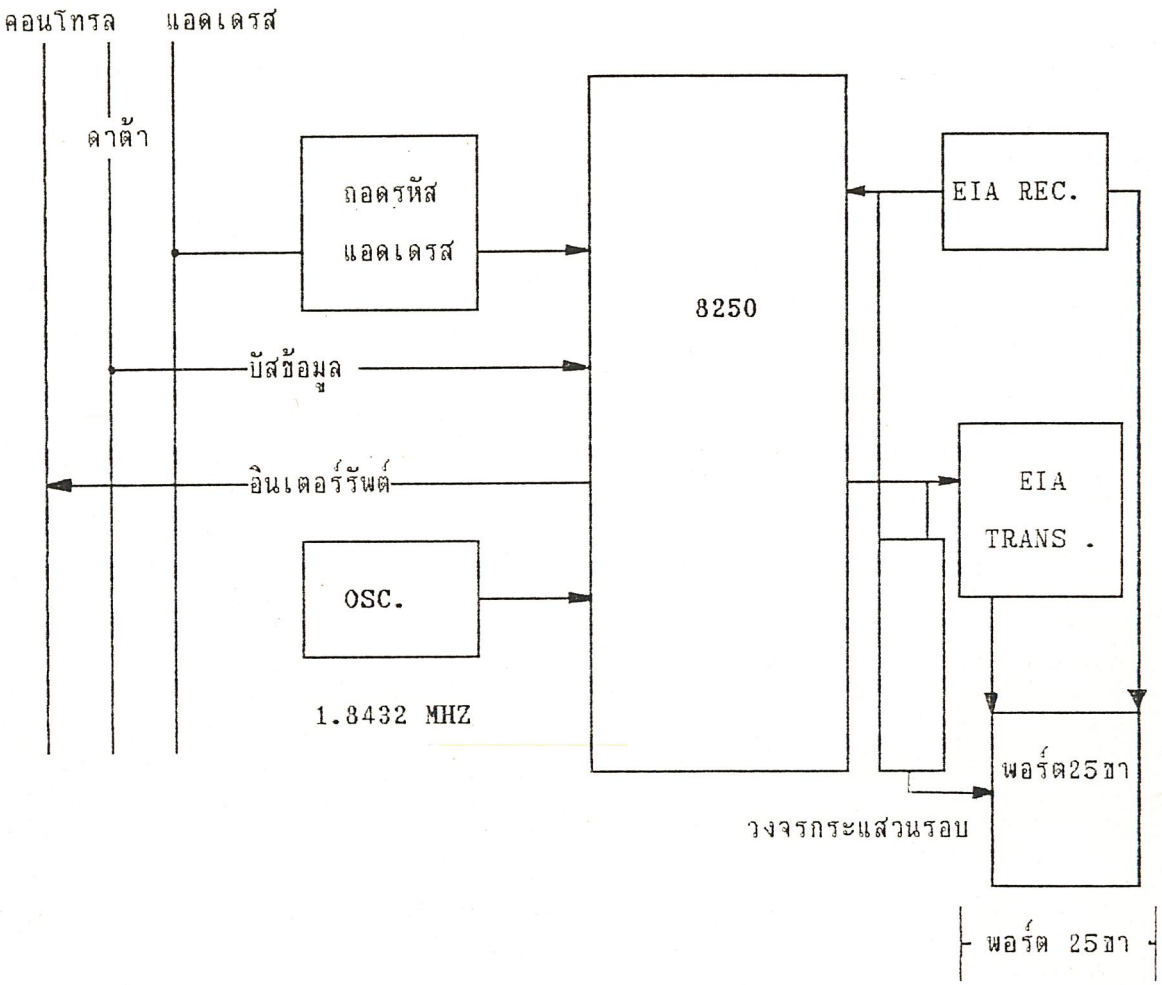
จึงกล่าวได้ว่า **สัญญาณทางไฟฟ้า** หมายถึงการรวมข้อมูลเกี่ยวกับการจาก แรงดันและกระแสของแต่ละขาสัญญาณด้วย ซึ่งถ้าเกิดการลัดวงจรกันระหว่าง 2 ขาขึ้นไป แล้ว จะต้องไม่ทำให้อุปกรณ์เกิดความเสียหาย (ไม่ได้หมายความว่าอุปกรณ์นั้นจะต้องไม่พัง)สำหรับส่วนของหน้าที่ซึ่งอาจเป็นส่วนที่สำคัญที่สุดก็ได้นั้น เป็นส่วนกำหนดลำดับของสัญญาณและการตอบสนองการทำงานของ DTE และ DCE

พอร์ตสื่อสารข้อมูล

พอร์ตสื่อสารเป็นส่วนสำคัญส่วนหนึ่งที่มีอยู่บนไมโครคอมพิวเตอร์ 16 บิต พอร์ตสื่อสารนี้มีชื่ออีกอย่างหนึ่งว่าคอม-พอร์ต (COM PORT) ผู้ออกแบบพอร์ตสื่อสารต้องการให้เป็นไป ตามมาตรฐานการเชื่อมต่อแบบอนุกรมที่เรียกว่า RS-232C พอร์ตนี้เป็นทางออกของข้อมูลที่ผู้ใช้สามารถส่ง หรือรับกับระบบอื่นได้ พอร์ตสื่อสาร RS-232C บนเครื่องไมโครคอมพิวเตอร์ 16 บิต มีโครงสร้างที่สามารถโปรแกรมด้วยการส่งรหัสคำสั่งให้กับชิปหลักได้สำหรับตัว RS-232C นี้เป็นมาตรฐานแบบอะซิงโครนัสที่โปรแกรมสตาร์ตบิตสต็อปบิตและพาริตี้อัตราการส่งก็อัตราการส่งก็สามารถกำหนดได้ตั้งแต่ 50 บอด ถึง 9600 บอด ลักษณะพิเศษของวงจรคือสามารถส่งสัญญาณ มาอินเตอร์รัพต์ซีพียู ตามเงื่อนไขได้ และยังมีโครงสร้างฮาร์ดแวร์ป้อนกลับ เพื่อใช้ในการตรวจสอบระบบว่า ทำงานปกติดีหรือไม่ได้อีกด้วย วงจรพอร์ตสื่อสารของไมโครคอมพิวเตอร์ 16 บิตนี้ใช้ไอซีหมายเลข 8250 เป็นตัวสำคัญของระบบ

โครงสร้างของพอร์ตสื่อสาร

พอร์ตสื่อสารของเครื่องไมโครคอมพิวเตอร์ 16 บิต ที่จะกล่าวถึงนี้ เป็นระบบมาตราฐานตามเครื่องไอบีเอ็มพีซีเอ็กที โครงสร้างบล็อกไดอะแกรมแสดงดังรูปที่ 9



พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือสล็อตขนาด (31*2)62 ขาของระบบนั่นเอง ซีพียูติดต่อกับ 8250 ในลักษณะพอร์ตที่เป็นอินพุตเอาต์พุต การจัดพอร์ตนี้กำหนดหมาย เลขพอร์ตอย่างเจาะจงระบบไมโครคอมพิวเตอร์ 16 บิต มีพอร์ตสื่อสารสองพอร์ตคือ คอม1(COM1) และคอม2 (COM2)ทั้งคอม 1 และคอม 2 มีหมายเลขอินพุตเอาต์พุตพอร์ตดังตาราง

ตาราง หมายเลขอินพุต เอาต์พุตพอร์ตของ COM_1 และ COM_2

อินพุตเอาต์พุตพอร์ต		เลือกรีจีสเตอร์	สถานะ DLAB
com 1	com 2		
3F8	2F8	บัฟเฟอร์ IX	DLAB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB = 0 (อ่าน)
3F8	2F8	แลตช์ตัวหาร (LSB)	DLAB = 1
3F9	2F9	แลตช์ตัวหาร (MSB)	DLAT = 1
3F9	2F9	อีนาเบิลอินเตอร์รัพต์	
3FA	2FA	กำหนดอินเตอร์รัพต์	
3FB	2FB	ควบคุมสายสื่อสาร	
3FC	2FC	ควบคุมโมเด็ม	
3FD	2FD	แสดงสายสถานะสื่อสาร	
3FE	2FE	แสดงสถานะโมเด็ม	

การเลือกหมายเลขอินพุตเอาต์พุตพอร์ต แยกเป็นสองกลุ่ม กลุ่มหนึ่งคือCOM_1 จะกำหนด หมายเลขพอร์ตจาก 3F8 ถึง 3F9 อีกกลุ่มหนึ่งถ้ากำหนดหมายเลขพอร์ตเป็น 2F8 ถึง 2FE

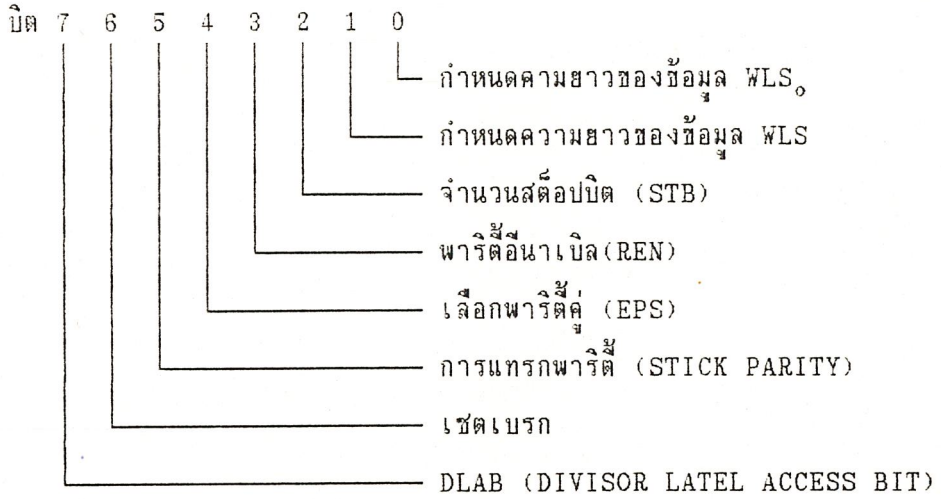
การใช้งานรีจีสเตอร์ต่างๆ บน 8250

การใช้งานบอร์ดอะแดปเตอร์สื่อสารจะต้องโปรแกรมค่าไมโครโค้ดเข้าใน 8250 ก่อน ซึ่งรีจีสเตอร์ต่างๆมีความหมายดังนี้

รีจีสเตอร์ควบคุมสายสื่อสาร (LINE CONTROL REGISTER)

ในการควบคุมรูปแบบของข้อมูลอะซิงโครนัสนั้น จะต้องโปรแกรมค่าลงในรีจีสเตอร์ควบคุมสายสื่อสาร รีจีสเตอร์ตัวนี้มี 8 บิต โดยแต่ละบิตมีความหมายดังนี้

พอร์ตแอดเดรสหมายเลข 3FB



ค่าของบิตในรีจีสเตอร์ควบคุมสายการสื่อสาร

บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	ความหมาย
0	0	หมายถึงข้อมูลขนาด 5 บิต
0	1	หมายถึงข้อมูลขนาด 6 บิต
1	1	หมายถึงข้อมูลขนาด 7 บิต
1	1	หมายถึงข้อมูลขนาด 8 บิต

บิต2เป็นบิตที่ใช้ในการกำหนดจำนวนสต็อปบิต ถ้าเป็น"0" หมายถึงการใช้สต็อปบิต 1 บิต แต่ถ้าบิต2 เป็น "1"ในกรณีส่งแบบ 5 บิต จะมีความหมายของสต็อปบิตเป็น 1.5 บิต แต่ถ้าส่งแบบ 6,7 หรือ 8 บิต ความยาวของสต็อปบิตจะเป็น 2

บิต 3 บิตนี้ เป็นบิตแสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้ โดยถ้าบิตนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้

บิต 4 มีค่าเป็น "0" และ **บิต** 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น"1"จะเป็นพาริตี้คู่

บิต5เมื่อบิต3 มีค่าเป็น "1"และบิต 5 มีค่าเป็น "1" และบิต 4 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี้(stick parity)ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น"1" และบิต 5 มีค่าเป็น "1" จะมีการกำหนดพาริตี้เป็น "1"

บิต 6 เป็นบิตที่ควบคุมการเบรก เมื่อบิต 6 มีค่าเป็น"1"

บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิตที่จะมีผลต่อการแลตช์ตัวหาร

การโปรแกรมอัตราบอด (BOUD RATE GENNERATOR)

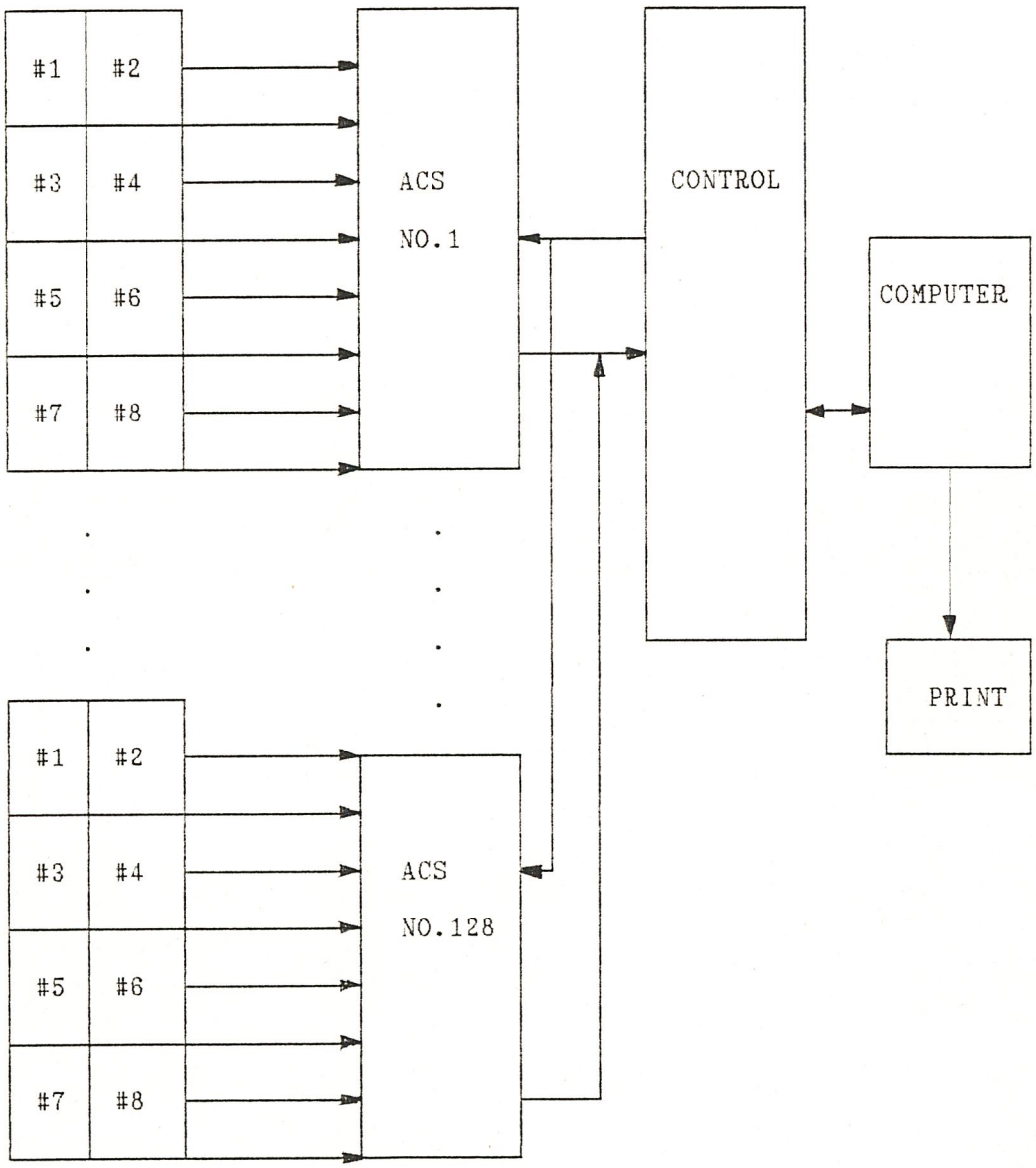
อัตราบอดได้รับการกำหนด เทียบกับสัญญาณนาฬิกา1.8432 MHZ และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1-(2^{16}-1) ค่าความถี่เอาต์พุต ของตัวกำหนดอัตราบอดมีค่าเท่ากับ 16*อัตราบอด ดังนั้นตัวหาร= ความถี่สัญญาณนาฬิกา/(อัตราบอด*16)การกำหนดอัตราบอดด้วยการกำหนดตัวหาร จึงเป็นกำหนดตัวหารในรีจีสเตอร์ 2 ตัว ตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจีสเตอร์นี้ การกำหนดต้องให้ DLAB = 1 แล้วให้โหลดลงมาในรีจีสเตอร์ $3F_8$ ซึ่งเรียงกันเป็น LSB ของ ตัวหาร ส่วน $3F_9$ เมื่อ DLAB = 1จะเป็นค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHZ เป็นดังตารางค่ากำหนดอัตราบอด

อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	-
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

บทที่ 3

โครงสร้างของระบบ AUTOMATIC BILLING

ส่วนประกอบต่างๆ ของระบบ ACS จะแสดงได้ดังบล็อคไดอะแกรมดังรูป



รูปที่ 10

จากบล็อกไดอะแกรม แบ่งการทำงานออกเป็น

1.**ช่องบรรจุอาหารประเภทเครื่องดื่ม** มีลักษณะเป็นช่องๆแต่ละช่องบรรจุเครื่องดื่ม 1 ชนิด พร้อมกับรายละเอียดของสินค้าที่ฝากล่องในโครงงานนี้ได้ออกแบบให้บรรจุเครื่องดื่มได้ 8 ช่องในการออกแบบทาง Mechanic ช่องบรรจุเครื่องดื่มจะออกแบบให้มีลักษณะที่เมื่อมีการเปิดฝากล่องออกแล้วหยิบเครื่องดื่มออกมาแล้วไม่สามารถจะเก็บกลับคืนได้ เพราะทันทีที่เปิดฝากล่องออกจะเอาข้อมูลเข้าไปเก็บในส่วน ACS ซึ่งหมายถึงเมื่อเปิดฝากล่องจะคิดเงินทันที สำหรับโครงงานนี้จะไม่พิจารณาถึง Mechanic ของกล่องบรรจุเครื่องดื่มจะพิจารณาเฉพาะการนำข้อมูลไปเก็บใน ACS เท่านั้นโดยจะใช้ Dip Switch แทนสถานะของการเปิด หรือปิดฝากล่องซึ่งจะเป็นข้อมูลนำไปประมวลผล

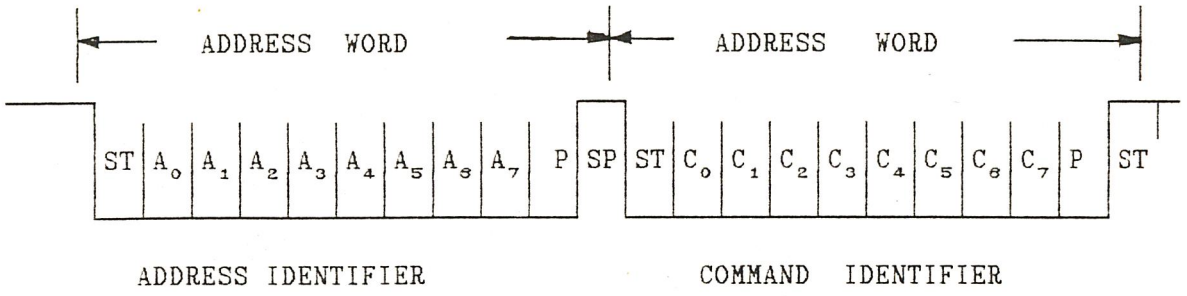
2.**ส่วน ACS (Automatic Check Servise)** ในส่วนนี้จะใช้ IC MC14469 ซึ่งเป็น Address Asynchronous Receive/Transmitter การทำงานในส่วนนี้ จะได้กล่าวถึงรายละเอียดต่อไป

3.**ส่วน master หรือ ส่วนควบคุม**(Control)ในส่วนนี้จะสร้างแหล่งจ่ายไฟเพื่อจ่ายไปในสายโดย ACS ทุกตัวนี้ไม่จำเป็นต้องมีแหล่งจ่ายไฟเลี้ยงเอง และในกรณีที่สายเกิด shortจะได้ไม่เกิดอันตรายต่อ ACS ทั้งหมดที่ต่อขนานกันเมื่อสายเกิด shortจะเสียหายเฉพาะส่วน **control** เท่านั้น

4.**ส่วน computer** ทำหน้าที่ประมวลผลข้อมูลที่มาจากห้องต่างๆพร้อมทั้งพิมพ์ใบเสร็จเก็บเงินในการคิดค่าบริการต่างๆ

รูปแบบและข้อกำหนดในการสื่อสารระหว่าง ACS กับ CPU

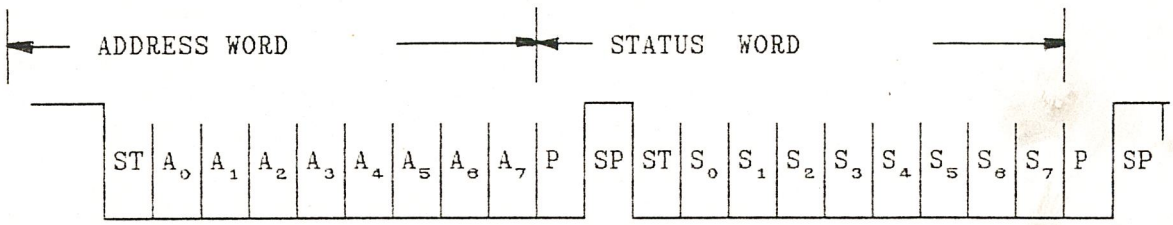
เนื่องจาก ACS มีจำนวนหลายตัว ACS แต่ละตัวจะมี ADDRESS ตัวเอง สามารถเลือกตั้งได้ตามต้องการโดย DIP SWITCH เมื่อ CPU ต้องการทราบสภาวะของ ACS ตัวไหนก็จะส่งADDRESS และ COMMAND WORD ออกไปตามคู่สายที่มี ACS ต่อขนานอยู่จำนวนมาก ACS ทุกตัวจะรับข้อมูลที่เป็น ADDRESS WORD พร้อมกันทุกตัว แต่จะมี ACS เพียงตัวเดียวเท่านั้นที่มี ADDRESS ตรงกับ ADDRESS ที่ CPU ส่งมา (ADDRESS MATCH) ACS ตัวนั้นก็จะส่ง ADDRESS ของตัวมันเอง ไปยัง CPU เพื่อเป็นการยืนยันว่า CPU ติดต่อกับ ACSไม่ผิดตัว ขณะเดียวกันก็จะส่ง STATUS WORD ตามไป CPU ก็จะทราบสภาวะของ ACS ว่าอยู่ในสภาวะอะไร ดังแสดงในรูปที่ 11



รูปที่ 11 รูปแบบของข้อมูลที่ ACS รับจาก CPU

จากรูปที่ 11 ข้อมูล (ADDRESS และ COMMAND) จะถูกส่งด้วยความเร็ว 4800 bit/sec โดยที่ A_0-A_6 คือ ADDRESS ของ ACS และ A_7 เป็นตัวบอกให้ทราบว่า WORD นี้เป็น ADDRESS WORD (A_7=1) ส่วน C_0-C_6 เป็น COMMAND WORD C_7 เป็นตัวบอก ให้ทราบว่า WORD นี้เป็น COMMAND WORD (C_7=0)

ในทำนองเดียวกันรูปแบบข้อมูลที่ ACS ส่งหรือตอบรับไปยัง CPU แสดงในรูปที่ 12 โดยที่ A_0-A_6 คือ ADDRESS ของ ACS S_0-S_7 เป็น STATUS WORD โดยมี S_0-S_6 เป็นตัวบอกสภาวะของ ACS ว่าอยู่ในสภาวะอะไร S_7 เป็นตัวบอกให้ทราบว่าเป็น STATUS WORD



รูปที่ 4 ACS ส่งข้อมูลตอบ CPU

หลักการทำงานของ AUTOMATRIC CHECK SERVICE

แบ่งการทำงานออกเป็น 2 ภาคคือ

1.ภาครับสัญญาณจากช่องบรรจุอาหารจากรูปที่ D_0_D_7 เป็นดิพสวิทช์(Dip switch)ในที่นี้สมมติให้เป็นสภาวะของช่องใส่อาหารประเภทเครื่องดื่มโดยเมื่ออยู่ในสภาวะปกติคือฝากล่องปิดสนิท จะให้ดิพสวิทช์อยู่ในสถานะ "ON" ในสภาวะเช่นนี้จะเกิดค่าโวลต์เตจตกคร่อม ประมาณ 0 โวลต์ สัญญาณ

จะอยู่ในสภาวะ "LOW"แต่ถ้าหากทำการดึงเครื่องดื่มออก ทำให้ฝากล่องเปิดออก จะให้ดิพสวิทช์อยู่ในสภาวะ "OFF"ในสภาวะเช่นนี้จะมีโวลต์เตจตกคร่อมประมาณ 5 โวลต์ สัญญาณจะอยู่ในสถานะ "HIGH" สถานะของดิพสวิทช์ต่างๆ นี้จะส่งเป็นอินพุต ซึ่งเสมือนกับ แสดงสถานะของการ ปิด-เปิด เครื่องดื่ม โดยจะส่งเป็นอินพุตให้กับ CD40175B 2 ตัวซึ่งเป็น D-FLIP FLOP โดยจะ LATCH ข้อมูลมาเก็บไว้เพื่อส่งเป็นข้อมูลให้กับขาสเตตัสของ MC14469 เมื่อช่องใดช่องหนึ่ง OFF หรือฝากล่องเปิดออกก็จะทำให้ มีข้อมูลมารอไว้ที่ขา S_0-S_7 และเมื่อมีการ ส่งข้อมูลมาจากคอมพิวเตอร์เพื่อเช็คสถานะของสเตตัส ก็จะสามารถทราบได้จากข้อมูลที่ LATCH มาไว้

สถานะของข้อมูลที่ถูกส่งนำมาเก็บไว้โดยเป็นอินพุทของ MC14469 จะมีสถานะดังนี้

	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
เมื่อช่องที่ 1 เปิด	0	0	0	0	0	0	0	1
เมื่อช่องที่ 2 เปิด	0	0	0	0	0	0	1	0
เมื่อช่องที่ 3 เปิด	0	0	0	0	0	1	0	0
เมื่อช่องที่ 4 เปิด	0	0	0	0	1	0	0	0
เมื่อช่องที่ 5 เปิด	0	0	0	1	0	0	0	0
เมื่อช่องที่ 6 เปิด	0	0	1	0	0	0	0	0
เมื่อช่องที่ 7 เปิด	0	1	0	0	0	0	0	0
เมื่อช่องที่ 8 เปิด	1	0	0	0	0	0	0	0
สภาวะที่ไม่มีการเปิดช่องใดๆ	0	0	0	0	0	0	0	0

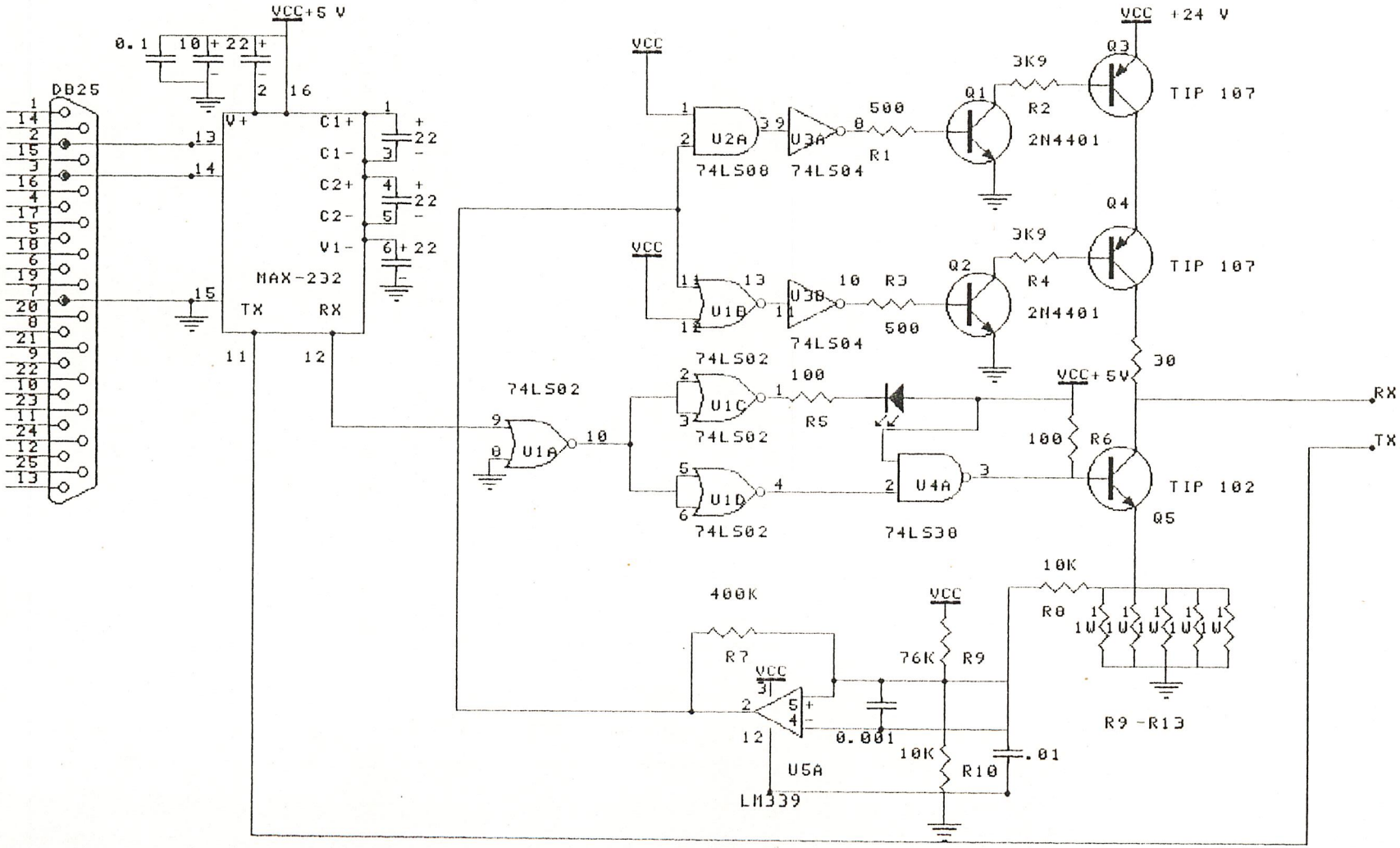
2.ภาคการรับส่งสัญญาณกับ CPU

จากรูปที่ 13 ใช้ MC14469 เป็น ADDRESSABLE ASYNCHORNOUS RECEIVER/TRANSMITTER มีความเร็วในการรับส่งข้อมูล 4800 บิทต่อวินาทีในส่วนนี้ จะมีการทำงานดังนี้
เมื่อ CPU ส่ง ADDRESS WORD ซึ่งข้อมูล ADDRESS ของ ACS มาทาง PORT RS-232(COM 1) ด้วยขนาดสัญญาณ(+/-)12 โวลต์ เข้าขา T_x ของ MAX232 ซึ่งจะแปลงสัญญาณลงเหลือ 0-5โวลต์ ผ่านมายังส่วนคอนโทรล ขนาดของสัญญาณ จะเพิ่มขึ้นเป็น 0-24 โวลต์ ACS ที่ต่อขนานอยู่กับสาย R_x จะรับ ADDRESS WORD ได้พร้อมกันหมด R_{14} และ R_{15} ทำหน้าที่เป็น VOLTANGE DIVIDER เพื่อแบ่งสัญญาณลงเหลือ +5 โวลต์ ป้อนเข้าขา R_x (ขา 19) ของ MC14469 ADDRESS WORD ที่ MC14469 รับไว้จะถูกนำไปเปรียบเทียบกับ ADDRESS ที่ตั้งไว้ด้วย DIP SWITCH ที่ขา A_0-A_8 ของ MC14469 ถ้า ADDRESS เหมือนกันก็จะเกิดพัลส์แคบๆ ที่ขา 31(VAP) กระตุ้นขาที่ขา 30 (SEND) MC14469 จะLATCH DATA ที่ S_0-S_7 เข้าไปเก็บไว้ในบัฟเฟอร์ พร้อมที่จะส่งสถานะนั้นออกไป เมื่อขา SEND ได้รับสัญญาณจากขา VAP MC14469 ก็จะส่ง ADDRESS และ STATUS WORD ของตัวเองตอบไปยังขา T_x สัญญาณจากขา T_x จะถูกอินเวิร์สด้วย Q6 และส่งผ่านไปยังขา R_x ของ MAX232 แปลงสัญญาณโวลต์เตจจาก 0-5 โวลต์ ไปเป็น(+/-)12 โวลต์ เข้าพอร์ต RS-232 หรือ COM 1 CPU ก็ทราบสภาวะของ ACS ได้ โดยมี LED เป็น MONITOR ให้ทราบว่าทุกครั้งที่ CPU และ ACS ส่งข้อมูลติดต่อกัน LED จะติดอยู่ตลอดเวลา

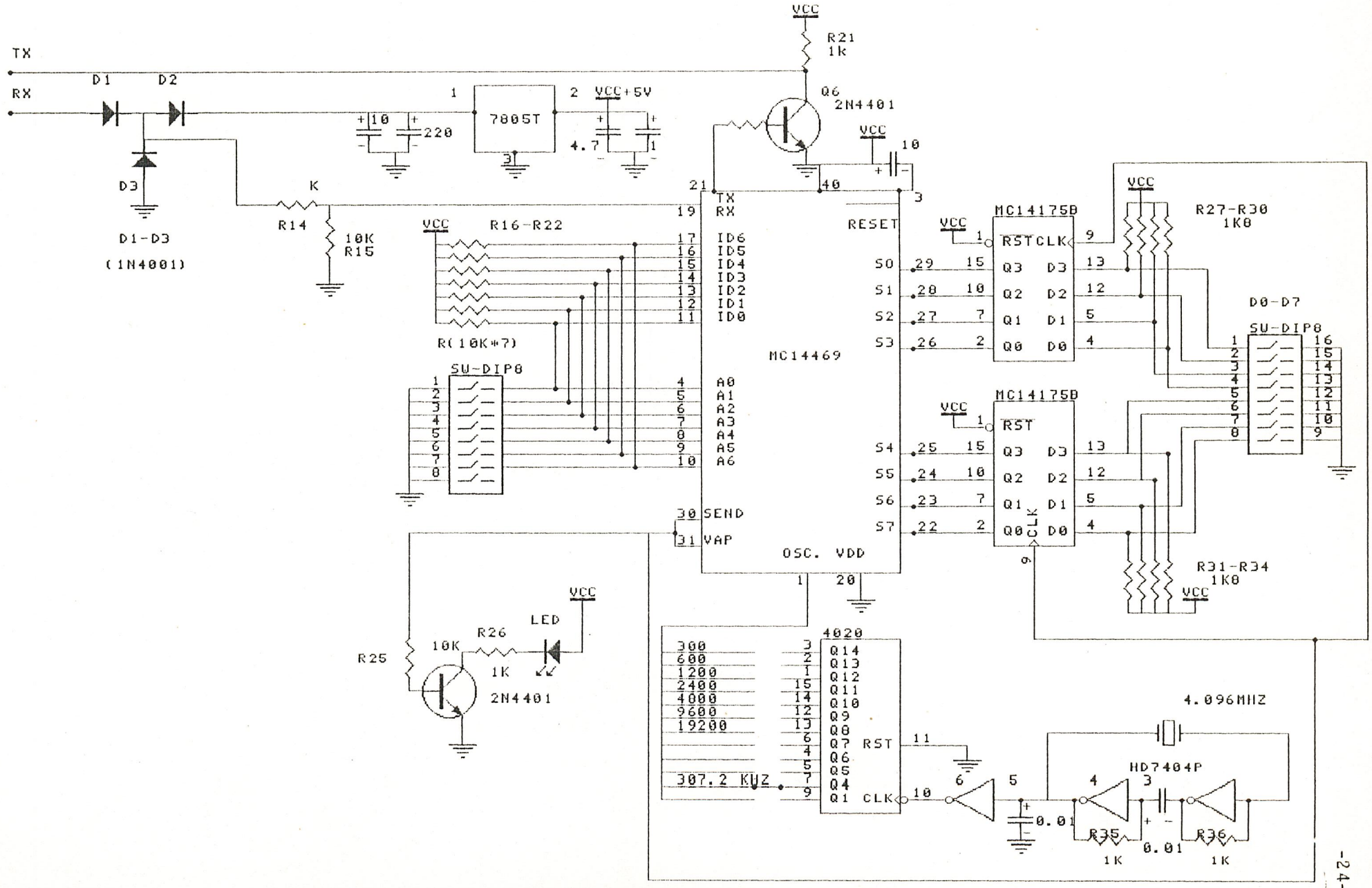
หลักการทำงานส่วนคอนโทรล

ส่วนของคอนโทรล เป็นส่วนที่ทำหน้าที่แปลงสัญญาณจากระดับ 0-5 โวลต์ไปเป็นระดับ 0-24 โวลต์ สำหรับส่งข้อมูลระหว่างไอบีเอ็มพีซีกับแอดเดรสเช็คเซอร์วิส หรือ ACS(Address Check Service :ACS) โดยผ่านทางซีเรียลพอร์ต คอม 1(serial port:com1)คอม 1 จะส่งข้อมูลด้วยความเร็ว 4800 บิทต่อวินาที ผ่าน U1A,U1D,U4A ไปยังขา B ของ Q5 เพื่อเตรียมพร้อมในการติดต่อกับ ACS ส่วน U5A เป็นโวลต์เตจคอมพาราเตอร์ ทำหน้าที่มอนิเตอร์สายส่งสัญญาณลัดวงจรหรือไม่ ในสภาวะปกติเอ้าท์พุทเป็นลอจิกโลว์(LOW) (ถ้าสายส่งลัดวงจรเอ้าท์พุทเป็นไฮท์"HIGH") สมมติว่า ในสภาวะปกติเอ้าท์พุตของ U5A เป็น "0" เอ้าต์พุท U3A เป็น "1" Q1 และ Q3 จะทำงาน(ON) ทำนองเดียวกันเอ้าต์พุทของ U1B เป็น "0" เอ้าท์พุต U3B เป็น "1" Q2 และ Q4 จะทำงาน ไฟบวก 24

โวลต์ จะถูกจ่าย มาที่คอนเลคเตอร์ของ Q5 ซึ่งจะเป็นการยกระดับสัญญาณที่ส่งมาจากขา Rx ซึ่งจะยกระดับสัญญาณจาก 0-5 โวลต์เป็น 0-24 โวลต์ สัญญาณจะถูกส่งไป ในสาย ผ่าน D1,D3 ซึ่ง D3 จะเป็นตัวทำหน้าที่เรคติฟาย(rectifier)ข้อมูลที่ส่งมาจากซีพียู ผ่าน 7805T ทำหน้าที่ เป็นโวลเตจเรคคูเรเตอร์ (voltage regulator)เป็นไฟตรง 5 โวลต์จ่ายแก่วงจร ACS ทั้งหมด เมื่อ ACS ได้รับแอดเดรสที่ถูกต้องจะเกิดพัลส์แคบๆ ที่ขา 31(VAP) จะถูกป้อนกลับมายังขา 30 (SEND)จะเกิดเอคโค่แอดเดรสเวิร์ดไปทาง TRO ผ่าน Q6 ซึ่งจะอินเวอร์สสัญญาณไปยังคอม 1ไอบีเอ็มพีซีก็จะทราบสถานะของ ACS ได้ในกรณีที่สายส่งเกิดลัดวงจรขาเอ้าท์พุตของ U5A จะเป็น "1"เอ้าท์พุตของ U2A เป็น "1" เอ้าท์พุตของ U3A เป็น "0"Q1 และ Q3 จะเลิกทำงานเพื่อตัดไฟตรง 24 โวลต์ออกจากวงจร ขณะเดียวกัน เอ้าท์พุตของ U1B เป็น "1" ทำให้เอ้าท์พุทของ U3B เป็น "0" Q2 และ Q4 ก็จะเลิกทำงานตัดวงจรอีกครั้งหนึ่ง



รูปที่ 13 แสดงวงจรส่วนควบคุม (CONTROL)



รูปที่ 14 แสดงวงจรของส่วนเอซีเอส (AUTOMATIC CHECK SERVICE)

ผลการทดลอง

ด้านฮาร์ดแวร์ ต่อสายส่วนที่เป็นส่วนของวงจร ACS (ADDRESS CHECK SERVICE) เข้ากับส่วนของคอนโทรล(CONTROL) ซึ่งทำหน้าที่ทั้งจ่ายไฟเลี้ยง และส่งสัญญาณไปยัง ACS ต่อ คอม 1 ของคอมพิวเตอร์เข้ากับส่วนของคอนโทรล และทำการตั้งค่า แอดเดรสของ ACS ไว้ที่ 128(80H) ส่งแอดเดรส 128 ออกที่คอม1 ผ่านคอนโทรลไปยัง ACS ปรากฏว่า ACS สามารถโต้ตอบกลับมา คือ ส่งค่าแอดเดรส 128 พร้อมกลับสเตตัสที่บ้อนสถานะโดย ใช้ดิพสวิทช์เป็นตัวกำหนด ได้อย่างถูกต้อง แต่เมื่อลองเปลี่ยนค่าแอดเดรสเป็นค่าอื่นแล้วส่งออกไป ซึ่งปรากฏว่า ACS ไม่สามารถตอบกลับมาได้ แสดงว่าแอดเดรสไม่ตรงกัน จึงไม่มีการส่งค่าแอดเดรสและสเตตัสกลับมา เป็นไปตามจุดประสงค์ที่ตั้งไว้

ด้านซอฟแวร์ เมื่อฮาร์ดแวร์เป็นไปดังจุดประสงค์จึงเขียนโปรแกรมเกี่ยวกับ การบริการภายในโรงแรม โดยลักษณะของโปรแกรมจะเป็นการรวบรวมและประมวลผลทุกวันในการประมวลผลทุกวันเพื่อให้ผู้ดูแลนำเครื่องดื่มไปใส่ทดแทนของที่แขกได้บริโภคไปและจะคิดรวมยอดให้ในวันที่แขกเช็คเอาต์ พร้อมทั้งเช็คในทันทีทันใดที่แขกมาขอเช็คเอาต์ได้เลย

สรุปผลการทดลอง

โครงงานนี้ได้แสดงถึงวิธีการสื่อสารข้อมูลระหว่าง CPU กับ ACS ซึ่งต่อขนานกันหลายๆ ตัวบนคู่สายเดียวพร้อมกับบังคับให้ ACS สื่อสารกับ PCU ได้ครั้งละ 1 ตัวเพื่อป้องกันการผิดพลาดรับข้อมูลตัวอื่น และได้พัฒนาเป็นโปรแกรมออกใบเสร็จโดยอัตโนมัติดังที่กล่าวมาแล้ว และการนำเอาสัญญาณในสายส่งมาใช้ประโยชน์ เป็นไฟเลี้ยง ACS ทั้งหมดซึ่งสามารถนำไปประยุกต์กับงานหลายๆด้าน

ภาคผนวก

```
##include "stdio.h"
#include "stdlib.h"
#include "ctype.h"
#include "conio.h"
#include "dos.h"
#include "time.h"

#define COK 50
#define SOD 40
#define BEE 80
#define PEP 50
#define JA 600
#define FAN 50
#define GRE 50
#define PAS 700

int a,b,c;

struct s {
            unsigned one:1;
            unsigned two:1;
            unsigned three:1;
            unsigned four:1;
            unsigned five:1;
            unsigned six:1;
            unsigned seven:1;
            unsigned eight:1;
            char room[4];
            int stat;
            int total;
            struct tm *time;
            char name[20];

            };

union {
          int x;
          struct s bit;
      }var;

void init(void);
void show(void);
void menu(void);
void frame(int x1,int y1,int x2,int y2);
void head(int x1,int y1,int x2,int y2);
int hight_light(void);
void checkout(void);
void menu_out(void);
void checkin(void);
void small_menu(void);
void hitil(void);
void win(void);
void hot(void);
```

```
                main()
{
  window(1,1,80,25);
  textbackground(BLACK);
  textcolor(WHITE);
  show();

  do{
     menu();
    }while(1);
}

void init(void)
{
  outportb(0x3fb,0x80);
  outportb(0x3f8,0x18);
  outportb(0x3f9,0x00);
  outportb(0x3fb,0x03);
}

/* Function for check out room */
void checkout(void)
{
    char cha;
    struct s s;
    time_t t;
    int count,Y;
   FILE *fp;
   t=time(NULL);

       do{
            hot();
            window(25,7,55,9);
            clrscr();
            frame(1,1,29,3);
            gotoxy(3,2);
            cprintf("ROOM NUMBER");
            gets(s.room);
            window(15,12,65,14);
            gotoxy(1,2);
            a=atoi(s.room);
            cprintf("ARE YOU SURE TO CHECK OUT ROOM NUMBER(Y/N):%d",a);
            cha=getch();
           }  while(cha=='N'||cha=='n');


           window(32,16,45,18);
           cprintf("Please wait!");

           if((fp=fopen(s.room,"ab"))==NULL)
                {
                printf("error in open file\n");
                exit(1);
                }
```

```
                a= atoi(s.room);
init();
outportb(0x3f8,a);
delay(10);
b = inportb(0x3f8);
delay(10);
c = inportb(0x3f8);

if(a==b)
  {
    s.stat = c;
    fwrite(&s,sizeof s,1,fp);
    if(ferror(fp))
      {
         printf("error in write file\n");

         exit(1);
      }

       }

else
   {
       window(15,20,75,21);
       gotoxy(1,1);
       cprintf("CAN NOT TRANSFER DATA TO ROOM NUMBER:%d",a);
       printf("\007");

   }
fclose(fp);

   if((fp=fopen(s.room,"rb"))==NULL)
         {
         printf("error in open file\n");
         exit(1);
         }
while(fread(&s,sizeof(s),1,fp)==1)
{
   menu_out();

   gotoxy(17,2);
   cprintf("%s",s.name);

   gotoxy(50,3);
   cprintf("%.2d-%.2d-%.2d",s.time[0].tm_mday,s.time[0].tm_mon+1,
                            s.time[0].tm_year);

   gotoxy(17,3);
   cprintf("%s",s.room);

   s.time = localtime(&t);
   gotoxy(50,2);
   cprintf("%.2d-%.2d-%.2d",s.time[0].tm_mday,s.time[0].tm_mon+1,
                            s.time[0].tm_year);
```

```
                                                        Y=10;
    count = 0;
      var.x = s.stat;
       if(var.bit.one==1)
                {
                if(count > 12)
                 {
                   count = 0;
                   Y = 10;
                   menu_out();
                 }
                gotoxy(32,Y);
                cprintf("COKE");
                gotoxy(68,Y);
                cprintf("%d",COK);
                s.total=s.total+COK;
                ++Y;
                ++count;
                }
if(var.bit.two==1)
                {
          if(count > 12)
                 {
                   count = 0;
                   Y = 10;
                   menu_out();
                 }
               gotoxy(32,Y);
               cprintf("SODA");
                gotoxy(68,Y);
               cprintf("%d",SOD);
               s.total=s.total+SOD;
               ++Y;
               ++count;
               }
if(var.bit.three==1)
                {
           if(count > 12)
                {
                  count = 0;
                  Y = 10;
                  menu_out();
                }
                gotoxy(32,Y);
                cprintf("BEER");
                 gotoxy(68,Y);
              cprintf("%d",BEE);
               s.total=s.total+BEE;
               ++Y;
               ++count;
               }
```

```
                if(var.bit.four==1)
                {
            if(count > 12)
               {
                 count = 0;
                 Y = 10;
                 menu_out();
               }
                gotoxy(32,Y);
                cprintf("PEPSI");
                 gotoxy(68,Y);
              cprintf("%d",PEP);
                s.total=s.total+PEP;
                ++Y;
                ++count;
                }
if(var.bit.five==1)
                {
            if(count > 12)
               {
                 count = 0;
                 Y = 10;
                 menu_out();
               }
                gotoxy(32,Y);
                cprintf("J&B");
                 gotoxy(68,Y);
              cprintf("%d",JA);
                s.total=s.total+JA;
                ++Y;
                ++count;
                }
if(var.bit.six==1)
                {
           if(count > 12)
               {
                 count = 0;
                 Y = 10;
                 menu_out();
               }
               gotoxy(32,Y);
               cprintf("FANTA");
                gotoxy(68,Y);
             cprintf("%d",FAN);
              s.total=s.total+FAN;
              ++Y;
              ++count;
              }
if(var.bit.seven==1)
                {
            if(count > 12)
               {
                 count = 0;
                 Y = 10;
                 menu_out();
               }
```

```
                    gotoxy(32,Y);
                   cprintf("GREENSPORT");
                    gotoxy(68,Y);
                 cprintf("%d",GRE);
                   s.total=s.total+GRE;
                   ++Y;
                   ++count;
                   }
    if(var.bit.eight==1)
                    {
              if(count > 12)
                    {
                    count = 0;
                    Y = 10;
                    menu_out();
                    }
                    gotoxy(32,Y);
                    cprintf("PASSPROT");
                     gotoxy(68,Y);
                 cprintf("%d",PAS);
                    s.total=s.total+PAS;
                    ++Y;
                    ++count;
                     }
  gotoxy(67,23);
  cprintf("%d",s.total);
  getch();
  }
  fclose(fp); getch();
}



   void hot(void)
        {
        window(1,1,79,24);
        clrscr();
        frame(1,1,79,24);
        gotoxy(32,3);
        cprintf("CHECK OUT");
        head(1,5,79,5);
        }
```

```
/* Function check everyday room */
void check_everyday(void)
{
    char ch,cr;
    struct s s;
    int Y;
    time_t t;
    FILE *fp;

    t=time(NULL);
       do{
          window(3,6,75,23);
          clrscr();
         do{
           hitil();
           window(25,7,55,9);
           clrscr();
           frame(1,1,29,3);
           gotoxy(7,2);
           cprintf("ROOM NUMBER : ");
           gets(s.room);
           window(15,12,65,14);
           gotoxy(1,2);
           a = atoi(s.room);
           cprintf("ARE YOU SURE TO CHECK ROOM NUMBER (Y/N):%d",a);
           cr=getch();
         } while(cr=='N'||cr=='n');

           textcolor(WHITE|BLINK);
           window(32,16,45,18);
           cprintf("Please wait!");
           if((fp=fopen(s.room,"ab"))==NULL)
                {
                printf("error in open file\n");
                exit(1);
                }
            init();
            a=atoi(s.room);
            outportb(0x3f8,a);
            delay(10);
            b=inport(0x3f8);
            delay(10);
            c=inportb(0x3f8);
            textcolor(WHITE);
            clrscr();

      if(a==b)
        {
          window(1,1,79,24);
          clrscr();
          frame(1,1,79,24);
          gotoxy(26,2);
          cprintf("ROOM NUMBER:%x",a);
          gotoxy(19,3);
/*        cprintf("DATE:%.2d-%.2d-%.2d",s.time->tm_mday,s.time->tm_mon+1
                                        s.time->tm_year);
*/        cprintf("DATE:%.2d-%.2d-%.2d",s.time[0].tm_mday,s.time[0].tm_m
                                        s.time[0].tm_year);
```

```
                                                        gotoxy(40,3);
 cprintf("TIME:%.2d:%.2d:%.2d",s.time[0].tm_hour,s.time[0].tm_min
                                  s.time[0].tm_sec);

Y=5;

 var.x=c;

if(var.bit.one==1)
     {
      gotoxy(30,Y);
      printf("COKE");
      s.total=s.total+COK;
      ++Y;
      }
if(var.bit.two==1)
      {
      gotoxy(30,Y);
      printf("SODA");
      s.total=s.total+SOD;
      ++Y;
      }
 if(var.bit.three==1)
       {
       gotoxy(30,Y);
       printf("BEER");
       s.total=s.total+BEE;
       ++Y;
         }
 if(var.bit.four==1)
        {
        gotoxy(30,Y);
        printf("PEPSI");
        s.total=s.total+PEP;
        ++Y;
        }
 if(var.bit.five==1)
        {
        gotoxy(30,Y);
        printf("J&B");
        s.total=s.total+JA;
        ++Y;
        }
  if(var.bit.six==1)
       {
       gotoxy(30,Y);
       printf("FANTA");
       s.total=s.total+FAN;
       ++Y;
       }
```

```
                    if(var.bit.seven)
                  {
                  gotoxy(30,Y);
                  printf("GREENSPORT");
                  s.total=s.total+GRE;
                  ++Y;
                  }
              if(var.bit.eight==1)
                   {
                   gotoxy(30,Y);
                   printf("PASSPROT");
                   s.total=s.total+PAS;
                   ++Y;
                   }
                s.time = localtime(&t);
                s.stat=c;
                fwrite(&s,sizeof s,1,fp);
                if(ferror(fp))
                     {
                     printf("error in write file\n");
                     exit(1);
                     }
               }
           else
               {
               window(15,20,75,21);
               gotoxy(1,1);
               cprintf("CAN NOT TRANSFER DATA TO ROOM NUMBER:%d",a);
               printf("\007");
               }
           window(20,22,60,23);
           printf("Q=Quit, Any other key= Continue");
           ch=getch();
           fclose(fp);
       }while(toupper(ch)!='Q');
}


void hitil(void)
       {


        window(1,1,79,24);
        clrscr();
        frame(1,1,79,24);
        gotoxy(32,3);
        cprintf("CHECK EVERYDAY");
        head(1,5,79,5);

        }

/* Function for check in */
void checkin(void)
{
char ch,cr;

struct s s;
FILE *fp;
time_t t;
```

```
t=time(NULL);
win();
do{
   window(2,6,75,23);
   clrscr();
   do{
      clrscr();
    window(25,7,55,9);
      clrscr();
      frame(1,1,29,3);
      gotoxy(7,2);
      cprintf("ROOM NUMBER:");
      gets(s.room);
      a = atoi(s.room);
      window(15,20,65,22);
      gotoxy(1,1);
      cprintf("ARE YOU SURE TO CHECK IN ROOM NUMBER:%d   (Y/N)",a);
      ch=getch();
   } while(ch=='N'||ch=='n');

      clrscr();
      window(3,10,75,18);
      if((fp=fopen(s.room,"wb"))==NULL)
         {
         gotoxy(20,10);
         cprintf("error in open file\n\r");
         exit(1);
         }
       gotoxy(15,2);
       cprintf("NAME: ");
       gets(s.name);

       s.time=localtime(&t);
       gotoxy(15,4);
       cprintf("ARRIVAL:%.2d-%.2d-%.2d",s.time[0].tm_mday,
                          s.time[0].tm_mon+1,s.time[0].tm_year);
       gotoxy(15,7);
       cprintf("TIME: %.2d:%.2d:%.2d",s.time[0].tm_hour,s.time[0].tm_min,
                          s.time[0].tm_sec);
       fwrite(&s,sizeof s,1,fp);
       if(ferror(fp))
              {
              printf("error in file\n");
              exit(1);
              }
        window(15,20,65,22);
        clrscr();
        frame(1,1,50,3);
        gotoxy(12,2);
        cprintf("Q=Quit, Any other key=continue");
        cr=getch();
   }while(cr!='Q'&& cr!= 'q');
   fclose(fp);

}
```

```
       /******win*******/
void win(void)
{
     window(1,1,79,24);
     clrscr();
     frame(1,1,79,24);
     gotoxy(32,3);
     cprintf("CHECK IN");
     head(1,5,79,5);

}

/* Function for run menu on editor */
void menu(void)
{
  window(1,1,80,24);
  clrscr();
  frame(1,1,79,24);
  gotoxy(25,3);
  cprintf("▒▒▒▒ AUTOMATIC BILLING ▒▒▒▒");
  gotoxy(35,5);
  cprintf("SELECT");
  small_menu();
}

void small_menu(void)
{
  int select;
  char cha;

  select = hight_light();
  switch(select){
    case 1: checkin();
            menu();
            break;
    case 2: checkout();
            break;
    case 3: check_everyday();
            break;
    case 4: window(20,6,60,20);
            clrscr();
            frame(1,4,39,6);
            gotoxy(3,5);
            cprintf("ARE YOU SURE TO QUIT PROGRAM (Y/N)");
            cha = getch();
            if(cha == 'Y' || cha == 'y')
                exit(0);
            else if(cha == 'N' || cha == 'n')
                  {
                    clrscr();
                    small_menu();
                  }
            break;
     }
}
```

```
 void frame(int x1,int y1,int x2,int y2)
{
  int count;

  gotoxy(x1,y1);
  putch(201);
  gotoxy(x2-1,y1);
  putch(187);
  gotoxy(x1,y2);
  putch(200);
  gotoxy(x2-1,y2);
  putch(188);

  for(count = x1+1 ; count < x2-1 ; ++count)
   {
    gotoxy(count,y1);
    putch(205);
   }
  for(count = x1+1 ; count < x2-1 ; ++count)
   {
    gotoxy(count,y2);
    putch(205);
   }
  for(count = y1+1 ; count < y2 ; ++count)
   {
    gotoxy(x1,count);
    putch(186);
   }
  for(count = y1+1 ; count < y2 ; ++count)
   {
    gotoxy(x2-1,count);
    putch(186);
   }

}

void head(int x1,int y1,int x2,int y2)
{
  int count;

  gotoxy(x1,y1);
  putch(204);
  gotoxy(x2-1,y2);
  putch(185);
  for(count = 2;count<x2-1;++count)
   {
     gotoxy(count,y2);
     putch(205);
   }
}
```

```
void show(void)

 clrscr();
 cprintf("\n\r\n\r\n\r");
 cprintf("                                                    \
 cprintf("                                                    \
 cprintf("                                                    \
 cprintf("                                                    \
 cprintf("                                                    \
 cprintf("\n\r\n\r\n\r");
 cprintf("                                                    \n\r")
 cprintf("                                                    \n\r")
 cprintf("                                                    \n\r")
 cprintf("                                                    \n\r")
 cprintf("                                                    \n\r")
 gotoxy(23,20);
 cprintf("TELECOMMUNICATION ENGINEERING");
 gotoxy(15,21);
 cprintf("KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG");
 gotoxy(27,22);
 cprintf("BY: SIRIPORN  PAWAVET");
 gotoxy(27,23);
 cprintf("    SOMNEOK  JUNPOO");
 getch();


nt hight_light(void)

 int select;
 char key;

 key = '2';
 select = 1;
 window(20,8,60,21);
 frame(1,1,39,6);
 window(21,9,60,9);
 cprintf("              CHECK IN             ");
 window(21,10,60,10);
 cprintf("              CHECK OUT            ");
 window(21,11,60,11);
 cprintf("            CHECK EVERYDAY         ");
 window(21,12,60,12);
 cprintf("                 QUIT              ");
```

AUTOMATIC BILLING

```
do{
 if(select ==1)
  {
    window(21,9,60,9);
    textbackground(WHITE);
    textcolor(BLUE);
    gotoxy(3,1);
    cprintf("            CHECK IN            ");
    key = getch();
    textbackground(BLACK);
    textcolor(WHITE);
    gotoxy(3,1);
    cprintf("            CHECK IN             ");
    if(key == 'P')
      select = 2;
    else if(key == 'H')
           select = 4;
  }
 else if(select == 2)
        {
          window(21,10,58,10);
          textbackground(WHITE);
          gotoxy(3,1);
          textcolor(BLUE);
          cprintf("            CHECK OUT           ");
          key = getch();
          textbackground(BLACK);
          textcolor(WHITE);
          gotoxy(3,1);
          cprintf("            CHECK OUT            ");
          if(key == 'P')
            select = 3;
          else if(key == 'H')
            select = 1;
        }
      else if(select == 3)
             {
                window(21,11,58,11);
                textbackground(WHITE);
                gotoxy(3,1);
                textcolor(BLUE);
                cprintf("          CHECK EVERYDAY        ");
                key = getch();
                textbackground(BLACK);
                textcolor(WHITE);
                gotoxy(3,1);
                cprintf("          CHECK EVERYDAY         ");
                if(key == 'P')
                  select = 4;
                else if(key == 'H')
                       select = 2;
             }
```

```
                    else if(select == 4)
                  {
                     window(21,12,58,12);
                     textbackground(WHITE);
                     gotoxy(3,1);
                     textcolor(BLUE);
                     cprintf("                    QUIT                ");
                     key = getch();
                     textbackground(BLACK);
                     textbackground(BLACK);
                     textcolor(WHITE);
                     gotoxy(3,1);
                     cprintf("                    QUIT                ");
                     if(key == 'P')
                         select = 1;
                     else if(key == 'H')
                               select = 3;
                   }
    }while(key != '\r');
  return select;


/* Function for write bin for check out */
void menu_out(void)

 int count;

 window(1,1,80,25);
 clrscr();
 gotoxy(15,1);
 cprintf("KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG");
 gotoxy(2,2);
 cprintf("NAME         :");
 gotoxy(2,3);
 cprintf("ROOM NUMBER :");
 gotoxy(2,4);
 cprintf("ROOM RATE   :");
 gotoxy(40,2);
 cprintf("DEPART  :");
 gotoxy(40,3);
 cprintf("ARRIVAL :");
 for(count = 1 ; count <=80 ; ++count)
  {
   gotoxy(count,5);
   putch(205);
  }
 gotoxy(1,7);
 cprintf("ITEM");
 gotoxy(7,7);
 cprintf("DATE");
 gotoxy(30,7);
```

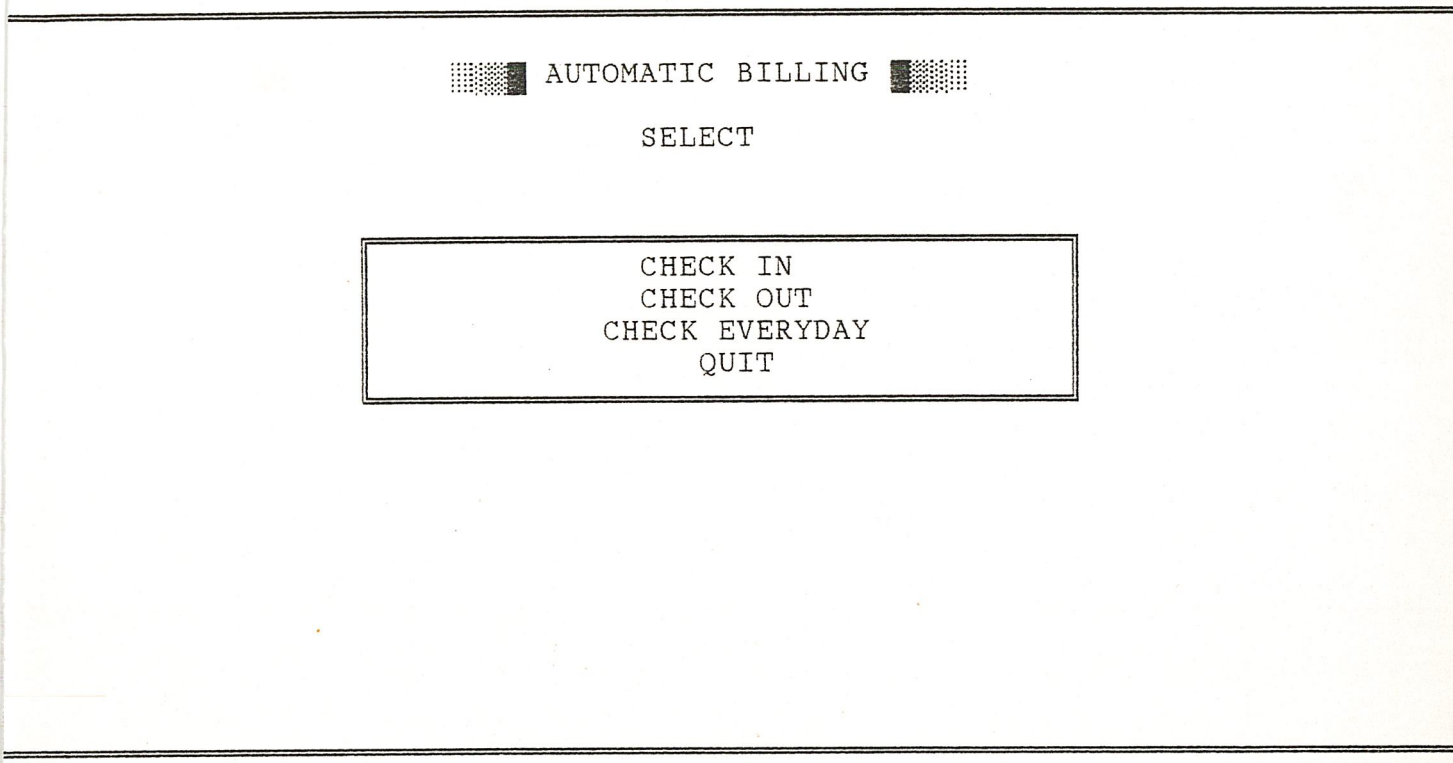
```
cprintf("DESCRIPTION");
gotoxy(70,6);
cprintf("AMOUNT");
gotoxy(70,8);
cprintf("Baht");
gotoxy(78,8);
cprintf("Stg");

for(count=1;count<=80;++count)
 {
   gotoxy(count,9);
   putch(196);
 }
for(count=1;count<=80;++count)
 {
   gotoxy(count,22);
   putch(196);
 }

gotoxy(5,5);
putch(209);
gotoxy(16,5);
putch(209);
gotoxy(65,5);
putch(209);
for(count=6;count<=21;++count)
 {
   gotoxy(5,count);
   putch(179);
 }
for(count=6;count<=21;++count)
 {
   gotoxy(16,count);
   putch(179);
 }
for(count=6;count<=21;++count)
 {
   gotoxy(65,count);
   putch(179);
 }
for(count=7;count<=21;++count)
 {
   gotoxy(76,count);
   putch(179);
 }

gotoxy(5,9);
putch(197);
gotoxy(16,9);
putch(197);
gotoxy(65,9);
putch(197);
gotoxy(76,9);
putch(197);
```

```
gotoxy(5,22);
putch(193);
gotoxy(16,22);
putch(193);
gotoxy(76 ,22);
putch(193);
gotoxy(65,22);
putch(193);
gotoxy(76 ,22);
putch(193);

for(count = 66;count<=80;++count)
 {
   gotoxy(count,7);
   putch(196);
 }
gotoxy(76,7);
putch(194);
gotoxy(65,7);
putch(195);
gotoxy(50,23);
cprintf("TOTAL (Baht)");
for(count = 66;count<=80;++count)
 {
   gotoxy(count,24);
   putch(205);
 }
gotoxy(65,24);
putch(212);
gotoxy(65,23);
putch(179);
gotoxy(65,22);
putch(197);
/*  window(1,23,80,24);
 clrscr();
*/  window(1,1,80,25);

 while(kbhit()) getch();
 }
```

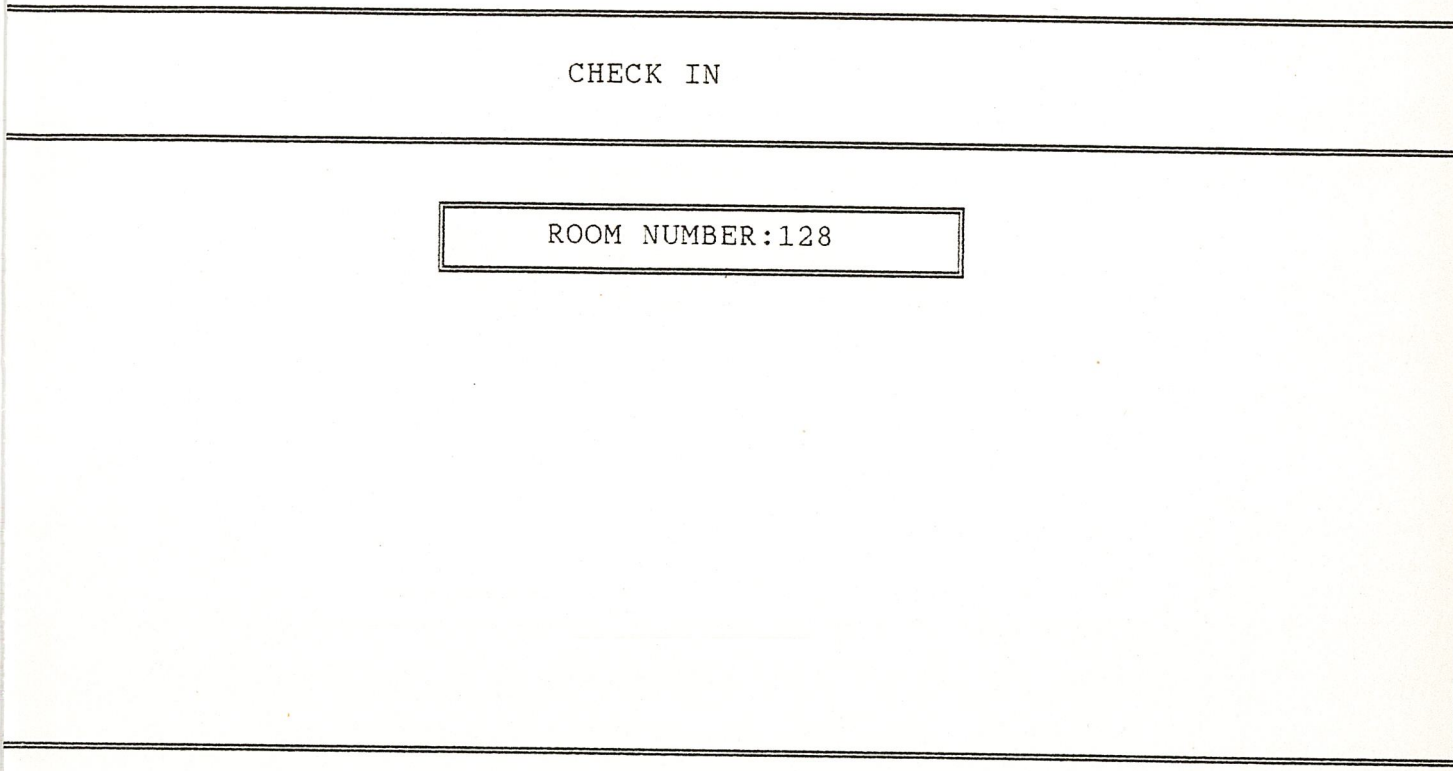
AUTOMATIC BILLING

TELECOMMUNICATION ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
BY: SIRIPORN PAWAVET
SOMNEOK JUNPOO

รูป ผ-1 แสดงเดโม



รูป ผ-2 แสดงเมนูเพื่อเลือกฟังก์ชันการทำงานต่างๆ



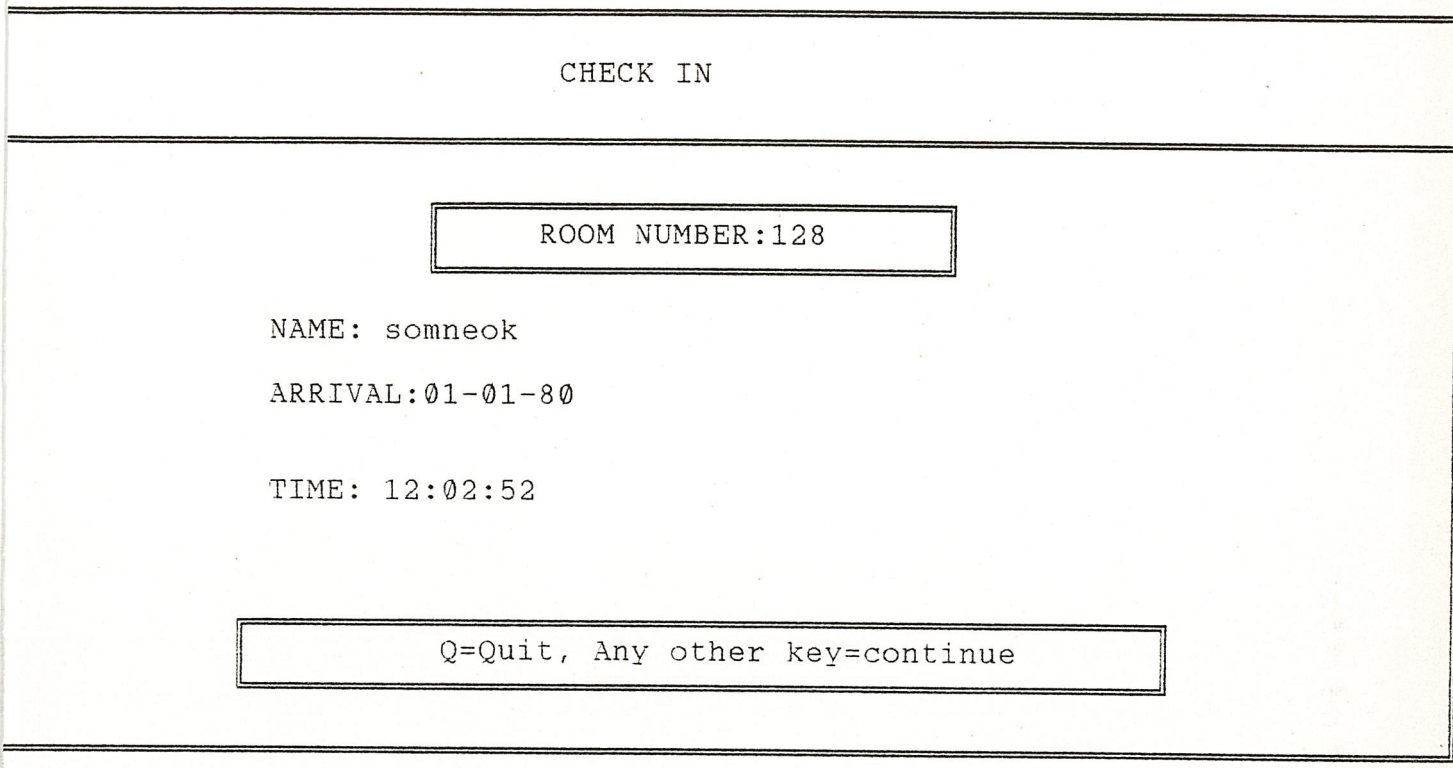
รูป ผ-3 แสดงการเลือกฟังก์ชันเช็คอิน (Check in) และการใส่ชื่อห้อง

CHECK IN

ROOM NUMBER:128

ARE YOU SURE TO CHECK IN ROOM NUMBER:128 (Y/N)

รูป ผ-4 แสดงเมนูเพื่อถามถึงเพื่อความแน่ใจ ในการเช็คอิน(Check in)



รูป ผ-5 แสดงการมาใส่ชื่อผู้มาพัก และจะแสดงวันที่ และเวลาที่เข้ามาพัก

CHECK EVERYDAY

ROOM NUMBER : 128

รูป ผ-6 แสดงการเลือกฟังก์ชัน เช็คทุกๆวัน(Check every day)

CHECK EVERYDAY

ROOM NUMBER : 128

ARE YOU SURE TO CHECK ROOM NUMBER (Y/N):128

รูป ผ-7 แสดงเมนูเพื่อถามถึงความแน่ใจ ในการเลือกเช็คทุกๆวัน(Check every day)

CHECK EVERYDAY

ROOM NUMBER : 128

ARE YOU SURE TO CHECK ROOM NUMBER (Y/N):128

CAN NOT TRANSFER DATA TO ROOM NUMBER:128

Q=Quit, Any other key= Continue

รูป ผ-8 แสดงเมนู เมื่อไม่สามารถจะส่งข้อมูลไปยังห้องนั้นๆ ได้

CHECK OUT

ROOM NUMBER128

รูป ผ-9 แสดงการเลือกฟังก์ชัน การเช็คเอาท์ (Check out)

CHECK OUT

ROOM NUMBER128

ARE YOU SURE TO CHECK OUT ROOM NUMBER(Y/N):128

รูป ผ-10 แสดงเมนูเพื่อถามถึงความแน่ใจในการเช็คเอาท์ (Check out)

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ME : somneok DEPART : 01-01-80
OM NUMBER : 128 ARRIVAL : 01-01-80
OM RATE :

M	DATE	DESCRIPTION	AMOUNT	
			Baht	Stg
		COKE SODA BEER J&B		
		TOTAL (Baht)	26933	

รูป ผ-11 แสดงรายการต่างๆ ที่ขายไป

```
/****program transfer data to ACS*****/
    #include "stdio.h"
    #include "dos.h"
    #include "conio.h"
    #include "ctype.h"
struct byte {
      unsigned one: 1;
      unsigned two: 1;
      unsigned three: 1;
      unsigned four: 1;
      unsigned five: 1;
      unsigned six: 1;
      unsigned seven: 1;
      unsigned eight :1;
            };
      union type_u{
      int x;
      struct byte bit;
            } var;
        int a=0x80;
        int b ,c;
      main ()
          {
       outportb(0x3fb,0x80);
       outportb(0x3f8,0x18);
       outportb(0x3f9,0x00);
       outportb(0x3fb,0x03);
       clrscr();
      outportb(0x3f8,a);
      delay(10);
      b= inportb(0x3f8);
      delay(10);
      c=inportb(0x3f8);
      if(b==0x80) {
                  printf("Addres is :%x",b);
                  printf("\n");
                  var.x = c;
                  if(var.bit.one == 1)
                  printf("coke\n");
                  if(var.bit.two ==1)
                  printf("soda\n");
                  if(var.bit.three ==1)
                  printf( "beer\n");
                 if(var.bit.four==1)
                 printf("pepsi\n");
                 if(var.bit.five==1)
                 printf("J&B\n");
                 if(var.bit.six==1)
                 printf("fanta\n");
                 if(var.bit.seven==1)
                 printf("green spot\n");
                 if(var.bit.eight==1)
                 printf("passport\n");
        }
          else
            printf( "address mismatch \n");
       }
```

โปรแกรมในการรับส่งข้อมูล ระหว่าง ACS กับ คอมพิวเตอร์

```
address is: 80
coke
soda
beer
pepsi
J&B
fanta
green spot

B:\>
```

รูป ผ-12 แสดงรายการรับแอดเดรสและสถานะ

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Addressable Asynchronous Receiver/Transmitter
CMOS

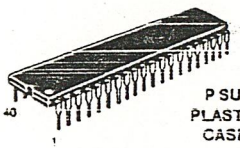
MC14469

The MC14469 receives one or two eleven-bit words in a serial data stream. One of the incoming words contains the address and when the address matches, the MC14469 then transmits information in two eleven-bit-word data streams. Each of the transmitted words contains eight data bits, an even parity bit, and start and stop bits.

The received word contains seven address bits with the address of the MC14469 set on seven pins. Thus 2^7 or 128 units can be interconnected in simplex or full duplex data transmission. In addition to the address received, seven command bits may be received for general-purpose data or control use.

The MC14469 finds application in transmitting data from remote A-to-D converters, remote MPUs, or remote digital transducers to the master computer or MPU.

- Supply Voltage Range: 4.5 V to 18 V
- Low Quiescent Current: 75 µA Maximum @ 5 V, 25°C
- Guaranteed Data Rates to 4800 Baud @ 5 V, to 9600 Baud @ 12 V
- Receive — Serial to Parallel
  Transmit — Parallel to Serial
- Transmit and Receive Simultaneously in Full Duplex
- Crystal or Resonator Operation for On-Chip Oscillator
- See Application Note AN-806A
- Chip Complexity: 1200 FETs or 300 Equivalent Gates



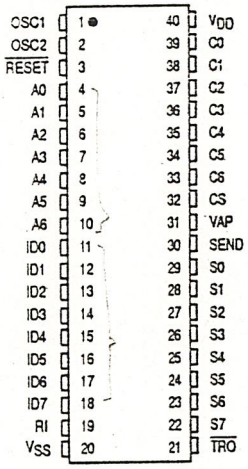
P SUFFIX
PLASTIC DIP
CASE 711

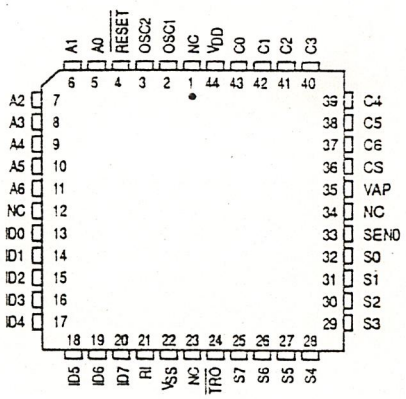
FN SUFFIX
PLCC
CASE 777

ORDERING INFORMATION

MC14469P	Plastic DIP
MC14469FN	PLCC

PIN ASSIGNMENTS

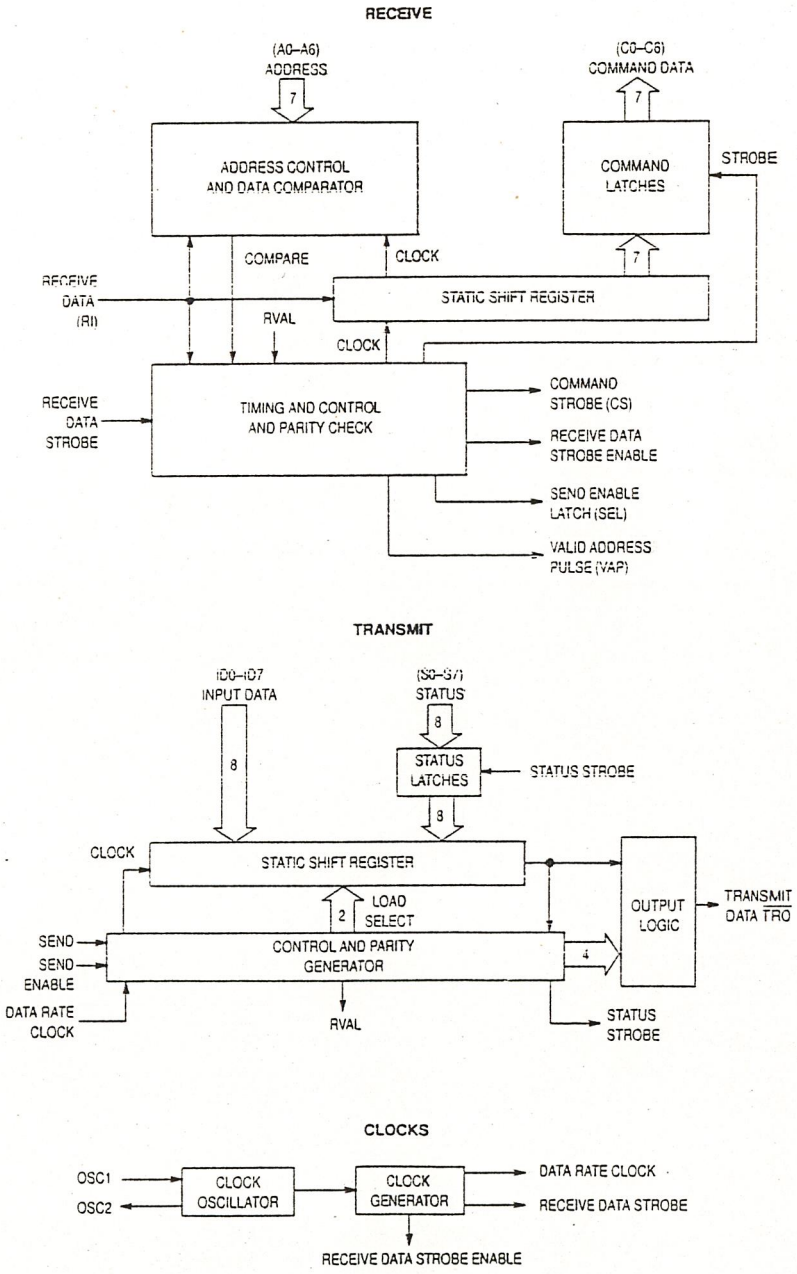




NC = NO CONNECTION

MC14469

BLOCK DIAGRAM



MAXIMUM RATINGS (Voltages referenced to V_{SS})

Parameter	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	−0.5 to +18	V
Input Voltage, All Inputs	V_{in}	−0.5 to V_{DD}+0.5	V
DC Current Drain per Pin	I	10	mA
Operating Temperature Range	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C

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

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

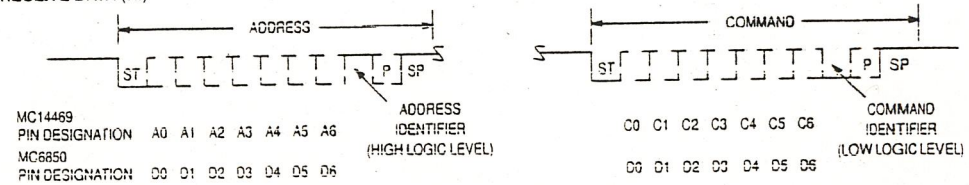
ELECTRICAL CHARACTERISTICS (Voltages Referenced to V_{SS})

Characteristic	Symbol	V_{DD} V	−40°C Min	−40°C Max	25°C Min	25°C Max	+85°C Min	+85°C Max	Unit
Output Voltage "0" Level	V_{OL}								V
V_{in} = V_{DD} or 0		5.0	—	0.05	—	0.05	—	0.05	
		10	—	0.05	—	0.05	—	0.05	
		15	—	0.05	—	0.05	—	0.05	
"1" Level	V_{OH}								V
V_{in} = 0 or V_{DD}		5.0	4.95	—	4.95	—	4.95	—	
		10	9.95	—	9.95	—	9.95	—	
		15	14.95	—	14.95	—	14.95	—	
Input Voltage (Except OSC1) "0" Level	V_{IL}								V
(V_O = 4.5 or 0.5 V)		5.0	—	1.5	—	1.5	—	1.5	
(V_O = 9.0 or 1.0 V)		10	—	3.0	—	3.0	—	3.0	
(V_O = 13.5 or 1.5 V)		15	—	4.0	—	4.0	—	4.0	
"1" Level	V_{IH}								V
(V_O = 0.5 or 4.5 V)		5.0	3.5	—	3.5	—	3.5	—	
(V_O = 1.0 or 9.0 V)		10	7.0	—	7.0	—	7.0	—	
(V_O = 1.5 or 13.5 V)		15	11	—	11	—	11	—	
Output Drive Current (Except OSC2)	I_{OH}								mA
(V_{OH} = 2.5 V) Source		5.0	−1.0	—	−0.8	—	−0.6	—	
(V_{OH} = 4.6 V)		5.0	−0.2	—	−0.16	—	−0.12	—	
(V_{OH} = 9.5 V)		10	−0.5	—	−0.4	—	−0.3	—	
(V_{OH} = 13.5 V)		15	−1.4	—	−1.2	—	−1.0	—	
(V_{OL} = 0.4 V) Sink	I_{OL}	5.0	0.52	—	0.44	—	0.36	—	mA
(V_{OL} = 0.5 V)		10	1.3	—	1.1	—	0.9	—	
(V_{OL} = 1.5 V)		15	3.6	—	3.0	—	2.4	—	
Output Drive Current (OSC2 Only)	I_{OH}								mA
(V_{OH} = 2.5 V) Source		5.0	−0.19	—	−0.16	—	−0.13	—	
(V_{OH} = 4.6 V)		5.0	−0.04	—	−0.035	—	−0.03	—	
(V_{OH} = 9.5 V)		10	−0.09	—	−0.08	—	−0.06	—	
(V_{OH} = 13.5 V)		15	−0.29	—	−0.27	—	−0.2	—	
(V_{OL} = 0.4 V) Sink	I_{OL}	5.0	0.1	—	0.085	—	0.07	—	mA
(V_{OL} = 0.5 V)		10	0.17	—	0.14	—	0.1	—	
(V_{OL} = 1.5 V)		15	0.5	—	0.42	—	0.3	—	
OSC Frequency	f_{OSC}	4.5	0	400	0	365	0	310	kHz
		12	0	800	0	730	0	620	
Input Current	I_{in}	15	—	±0.3	—	±0.3	—	±1.0	µA
Pull-Up Current (A0–A6, ID0–ID7)	I_{UP}	15	12	120	10	100	8.0	85	µA
Input Capacitance (V_{in} = 0)	C_{in}	—	—	—	—	7.5	—	—	pF
Quiescent Current (Per Package)	I_{DD}	5.0	—	75	—	75	—	565	µA
		10	—	150	—	150	—	1125	
		15	—	300	—	300	—	2250	
Supply Voltage	V_{DD}	—	+4.5	+18	+4.5	+18	+4.5	+18	V

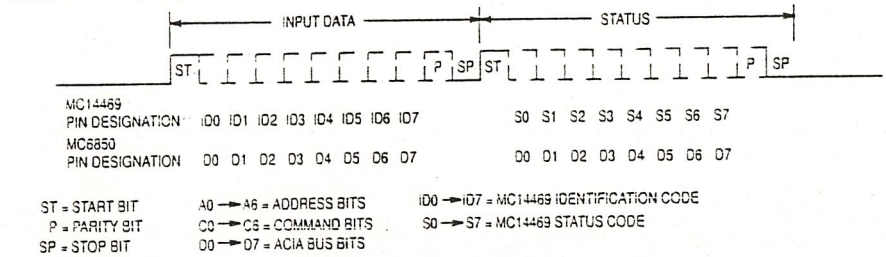
MC14469

DATA FORMAT AND CORRESPONDING DATA POSITION AND PINS FOR MC14469 AND MC6850

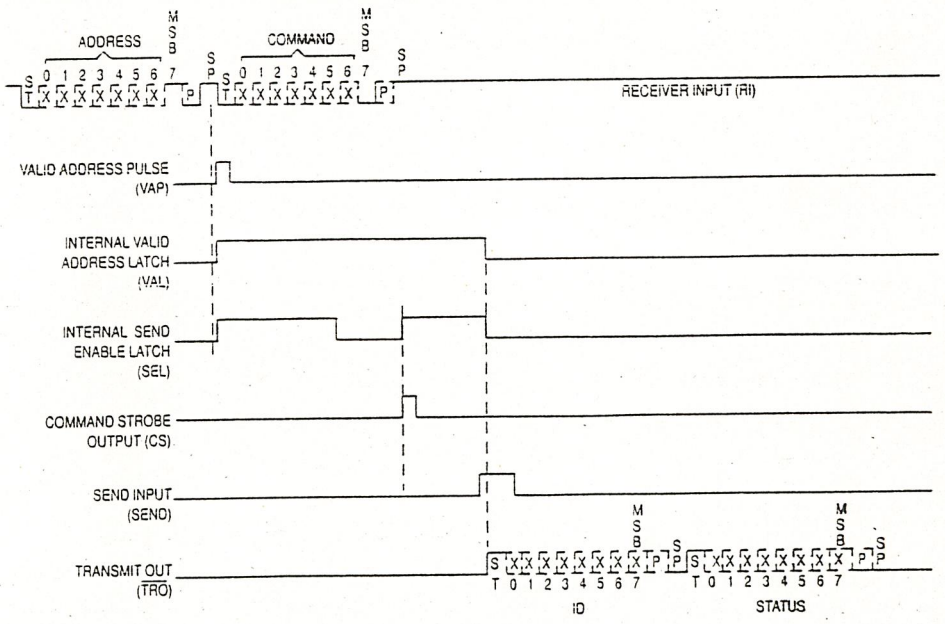
RECEIVE DATA (RI)



TRANSMIT DATA ($\overline{TRO}$)



TYPICAL RECEIVE/SEND CYCLE



PIN DESCRIPTIONS

OSCILLATOR (OSC1, OSC2)

These pins are the oscillator input and output. (See Figure 1.)

RESET ($\overline{\text{RESET}}$)

When this pin is pulled low for a minimum of 700 ns, the circuit is reset and ready for operation.

ADDRESS (A0–A6)

These inputs are the address setting pins which contain the address match for the received signal. Pins A0 through A6 have on-chip pullup resistors.

INPUT DATA (ID0–ID7)

These pins contain the input data for the first eight bits of data to be transmitted. Pins ID0–ID7 have on-chip pullup resistors.

RECEIVE INPUT (RI)

This is the receive input pin.

NEGATIVE POWER SUPPLY (VSS)

This pin is the negative power supply connection. Normally this pin is system ground.

TRANSMIT REGISTER OUTPUT SIGNAL ($\overline{\text{TRO}}$)

This pin transmits the outgoing signal. Note that it is inverted from the incoming signal. It must go through one stage of inversion if it is to drive another MC14469.

SECOND OR STATUS INPUT DATA (S0–S7)

These pins contain the input data for the second eight bits of data to be transmitted.

SEND (SEND)

This pin accepts the send command after receipt of an address.

VALID ADDRESS PULSE (VAP)

This is the output for the valid address pulse upon receipt of a matched incoming address.

COMMAND STROBE (CS)

This is the output for the command strobe signifying a valid set of command data (C0 through C6).

COMMAND WORD (C0–C6)

These pins are the readout of the general-purpose command word which is the second word of the received signal.

POSITIVE POWER SUPPLY (VDD)

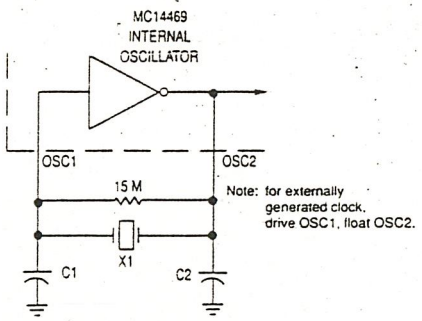
This pin is the package positive power supply pin.

OPERATING CHARACTERISTICS

The receipt of a start bit on the receive input (RI) line causes the receive clock to start at a frequency equal to that of the oscillator divided by 64. All received data is strobed in at the center of a receive clock period. The start bit is followed by eight data bits. Seven of the bits are compared against states of the address of the particular circuit (A0–A6). Address is latched 31 clock cycles after the end of the start bit of the incoming address. The eighth bit signifies an address word "1" or a command word "0". Next, a parity bit is received and checked by the internal logic for even parity. Finally a stop bit is received. At the completion of the cycle if the address matches, a valid address pulse (VAP) occurs. Immediately following the address word, a command word is received. It also contains a start bit, eight data bits, even parity bit, and a stop bit. The eight data bits are composed of a seven-bit command, and a "0" which indicates a command word. At the end of the command word a command strobe pulse (CS) occurs.

A positive transition on the send input initiates the transmit sequence. Send must occur within 7 bit times of CS. Again the transmitted data is made up of two eleven-bit words, i.e., address and command words. The data portion of the first word is made up from input data inputs (ID0–ID7), and the data for the second word from second input data (S0–S7) inputs. The data on inputs ID0–ID7 is latched one clock before the falling edge of the start bit. The data on inputs S0–S7 is latched on the rising edge of the start bit. The transmitted signal is the inversion of the received signal, which allows the use of an inverting amplifier to drive the lines. $\overline{\text{TRO}}$ begins either 1/2 or 1-1/2 bit times after send, depending where send occurs.

The oscillator can be crystal controlled or ceramic resonator controlled for required accuracy. OSC1 can be driven from an external oscillator. See Figure 1.



X1 = Ceramic Resonator: 307.2 kHz ± 1 kHz for 4800 baud rate.
C1 and C2 are sized per the ceramic resonator supplier's recommendation.

Ceramic Resonator Suppliers:*
1. Morgan Matroc, Inc., Bedford, OH, 216/232–8600
2. Radio Materials Co., Attica, IN, 317/762–2491

* Motorola cannot recommend one supplier over another and in no way suggests that this is a complete listing of ceramic resonator suppliers.

Figure 1. Oscillator Circuit

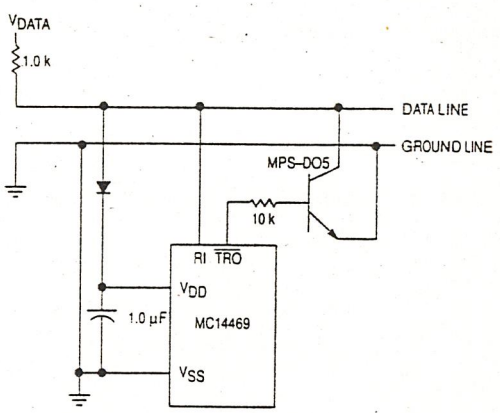


Figure 2. Rectified Power from Data Lines Circuit

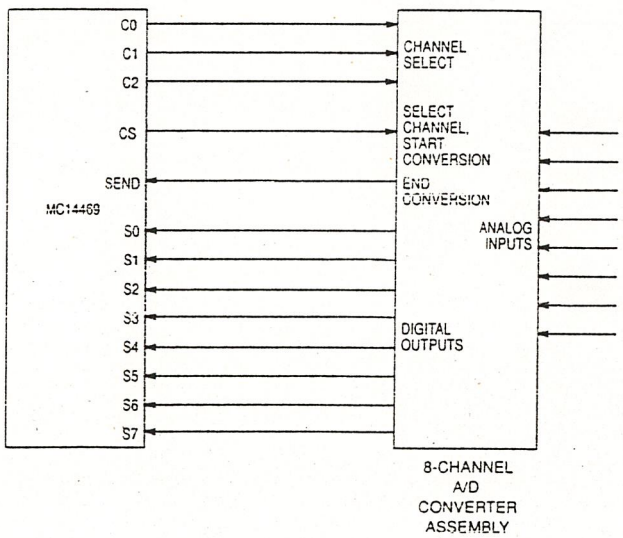


Figure 3. A-D Converter Interface

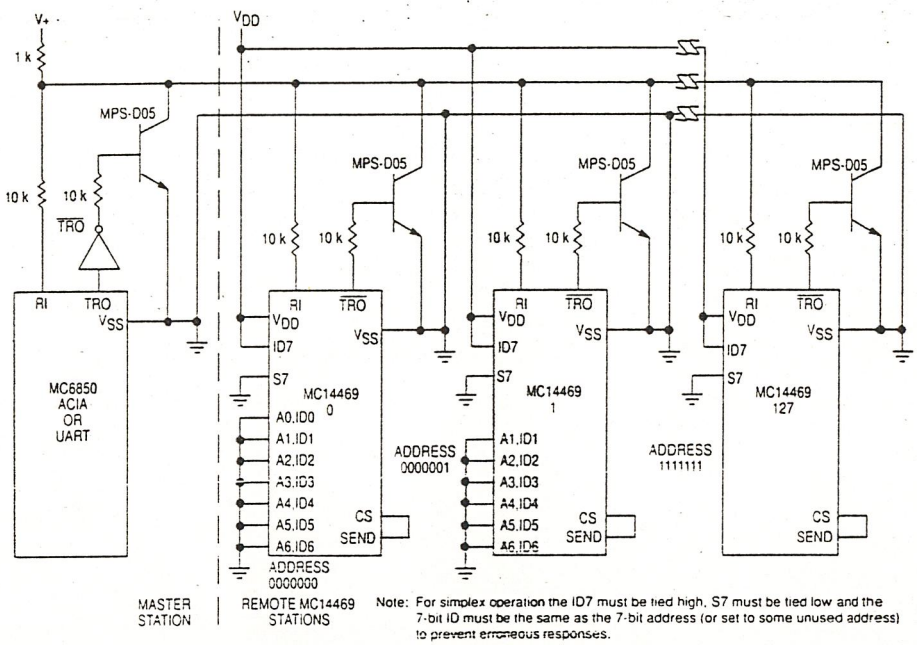


Figure 4. Single Line, Simplex Data Transmission

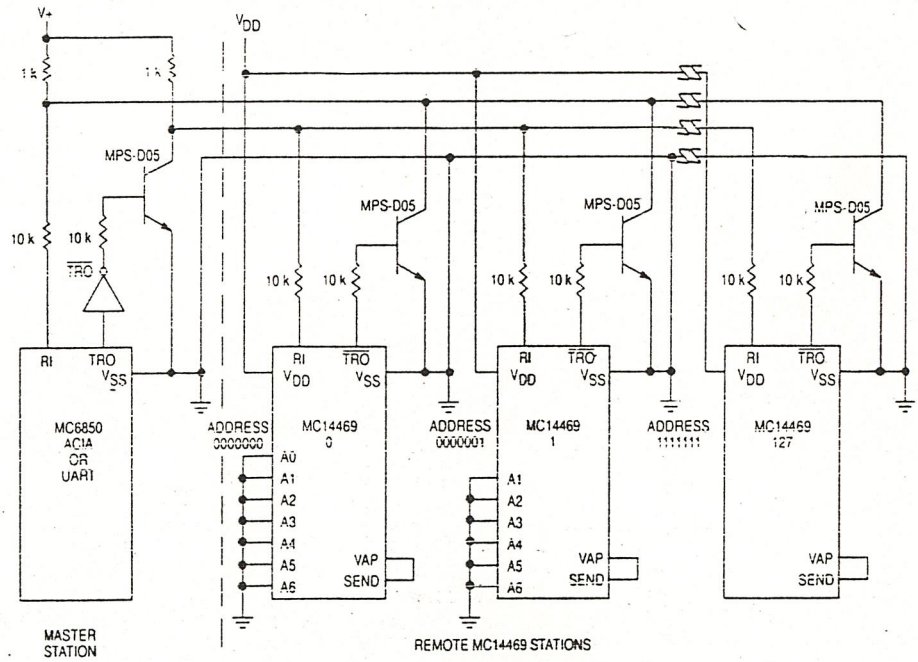


Figure 5. Double Line, Full Duplex Data Transmission

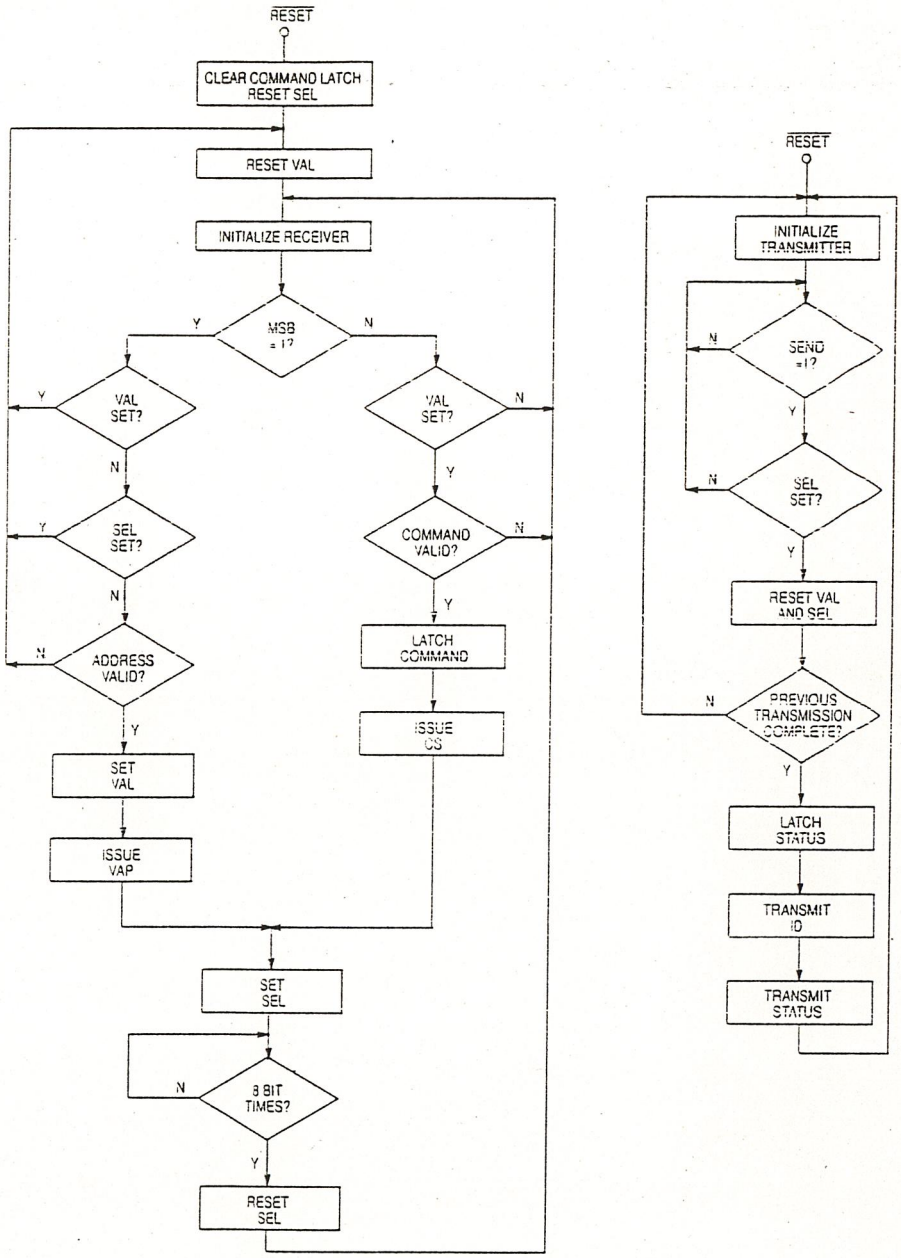


Figure 6. Flow Chart of MC14469 Operation

เอกสารอ้างอิง

[1] ชูชัย ธนสารดังเจริญ และ ทินกร ดุ๊ก, "การสื่อสารข้อมูล" สำนักพิมพ์ฟิสิกส์เซนเตอร์, พิมพ์ครั้งที่ 2.

[2] บริษัทซีเอ็ดยูเคชั่น จำกัด, "คู่มือไอซี CMOS 4000 SERIES" ห้างหุ้นส่วนจำกัดเอ็น-เอชการพิมพ์,2535.

[3] ยืน ภู่วรวรรณ, "เทคโนโลยีฮาร์ดแวร์ IBM PC", บริษัทซีเอ็ดยูเคชั่นจำกัด 2521.

[4] MOTOROLA, "CMOS LOGIC DATA", MOTOROLA, Third Print,1991.

[5] MOTOROLA, "CMOS APPLICATION-SPECIFIC STANDARD IC", MOTOROLA INC,1991.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้เนื่องจากได้รับคำแนะนำจากท่านอาจารย์หลาย ๆ ท่านจึงขอขอบพระคุณท่านอาจารย์สมยศ จุณณะปิยะที่ให้คำแนะนำเป็นที่ปรึกษาในการแก้ปัญหา พร้อมทั้งขอขอบพระคุณท่านอาจารย์ท่านอื่นรวมทั้งเพื่อน ๆ รุ่นพี่และรุ่นน้องที่ได้ให้กำลังใจ

จึงขอขอบคุณมา ณ ที่นี้

ผู้จัดทำ