ปีการศึกษา 2532

ข่ายข้อมูลผ่านระบบวิทยุ

(DATA LINK BY RADIO)

โดย

นาย เสรี อัศวาวัทย์

นาย สุรชัย อัศดามงคล

อาจารย์ที่ปรึกษา

ผศ.ณรงค์ เหมมาร์ณ

ปริญญานิพนธ์ปีการศึกษา 2532

ภาควิชา วิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ข่ายข้อมูลผ่านระบบวิทยุ

ผู้จัดทำ

1. นายเสรี อัศวารักษ์ 29-1264
2. นายสุรชัย อัศดามงคล 29-1273

..... อาจารย์ที่ปรึกษา

(ผศ.ณรงค์ เหมกรณ์)

ข่ายข้อมูลผ่านระบบวิทยุ

เสรี อัศวารักษ์
สุรชัย อัศดามงคล
ผศ.แรงค์ เหมกรณ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2532

บทคัดย่อ

การติดต่อสื่อสารด้วยคอมพิวเตอร์ผ่านระบบวิทยุนี้ ใช้ประโยชน์ได้ดียิ่งในพื้นที่ไกล ๆ ที่ไม่สามารถวางข่ายสายไปถึงได้ โดยใช้คลื่นวิทยุที่ตั้งไว้ที่ความถี่เดียวกัน และสามารถนำไปประยุกต์ใช้ร่วมกับอุปกรณ์ต่าง ๆ ให้มีประสิทธิผลมากขึ้นอีกด้วย

ส่วนที่ได้จัดทำขึ้นในโครงงานนี้ คือ โปรแกรมที่ใช้ในการส่งผ่านข้อมูลที่มีฟังก์ชันการทำงานในการติดต่อสื่อสารและรูปแบบของข้อมูลเพื่อการจัดเก็บ พร้อมทั้งอุปกรณ์บันทึกข้อมูลที่ติดต่อระหว่างกัน ซึ่งสามารถนำมาใช้ในการวิเคราะห์ลักษณะของข้อมูลและความผิดพลาดที่อาจเกิดขึ้นได้ต่อไป

DATA LINK BY RADIO

Seree Assavarak

Surachai Assadamongkol

Asst. Prof. Narong Haemakorn Advisor

1989

Abstract

Data link by radio with computer is very useful for longdistance areas that cannot use the cables. It is used by setting the same frequency from both sides of the terminals and can use with other devices for more efficiencies.

This study consists of the programes and data-recorder which are used for detection and analysing the errors from the data link.

สารบัญ

บทที่ 1
บทนำ

ในปัจจุบันคอมพิวเตอร์ (computer) ได้เข้ามาเป็นส่วนหนึ่งในชีวิตประจำวันโดยได้นำไปใช้งานในด้านต่าง ๆ หลายแขนง ในระบบโทรคมนาคมยุคใหม่ได้มีการปรับปรุงให้มีประสิทธิภาพสูงขึ้นเพื่อมารองรับการใช้งานในด้านการติดต่อสื่อสารที่เพิ่มขึ้นอย่างมากมาย

การสื่อสารข้อมูลระหว่างคอมพิวเตอร์สามารถติดต่อ (link) กันได้ใน 2 ลักษณะ คือ

1) การสื่อสารข้อมูลระหว่างกันโดยใช้สายเชื่อมต่อระหว่างกันซึ่งความสามารถในการส่งข้อมูลขึ้นอยู่กับคุณสมบัติของสายส่ง จึงไม่สะดวกในการบำรุงรักษาหรือตรวจสภาพ

2) การสื่อสารข้อมูลโดยผ่านคลื่นวิทยุ ซึ่งสามารถทำการติดต่อกันได้ในระยะทางไกลหรือที่ ๆ ไม่สามารถเดินสายส่งเข้าไปถึงได้ ทำให้ไม่มีปัญหาในเรื่องของสายส่ง

หลักการโดยทั่วไป เพื่อให้สะดวกในการกำหนดโครงสร้างและรูปแบบในการส่งข้อมูลให้มีลักษณะเดียวกัน จึงกำหนดให้คอมพิวเตอร์เป็น *อุปกรณ์ปลายทาง (Data Terminal Equipment) ส่วนโมเด็มและอุปกรณ์เชื่อมต่อกับโมเด็มเพื่อทำการส่งรวมเรียกว่า *อุปกรณ์สื่อสารข้อมูล (Data Circuit Terminating Equipment) ซึ่งสัญญาณข้อมูลจากอุปกรณ์ปลายทาง (DTE) จะต้องผ่านอุปกรณ์อินเตอร์เฟส เพื่อให้ได้สัญญาณที่เป็นไปตามข้อกำหนดที่เกี่ยวข้องกับการเชื่อมต่อทางกล (mechanical connection) เช่น คุณสมบัติทางไฟฟ้าและหน้าที่ของสัญญาณก่อนทำการส่งโดยอุปกรณ์สื่อสารข้อมูล (DATA Communication equipment) ต่อไป เพื่อให้เป็นมาตรฐาน สามารถใช้ได้กับอุปกรณ์ของบริษัทแต่ละแห่งได้ โดยที่อุปกรณ์อินเตอร์เฟสที่ใช้ในโครงงานนี้คือ EIA RS-232-C

เมื่ออุปกรณ์หรือ terminal เชื่อมต่อกันตามข้อกำหนดคุณสมบัติทางกายภาพ (physical layer) โดยผ่านตัวกลางใด ๆ สามารถที่จะส่งผ่านข้อมูลหรือติดต่อระหว่างกันได้แล้วนั้น ฟังก์ชั่นในการทำงานหรือในการสื่อสารระหว่างกันเหล่านั้นจำเป็นจะต้องมีโปรแกรมสนับสนุน เพื่อให้การทำงานเป็นไปตามขั้นตอนที่ถูกต้องสามารถติดต่อกับอุปกรณ์ภายนอก หรือ โปรเซสเซอร์ (processor) อื่น ๆ ซึ่งโปรแกรมที่เขียนขึ้นโดยทั่วไปผู้ใช้ไม่จำเป็นต้องมีความรู้ทางด้านฮาร์ดแวร์ของระบบอย่างลึกซึ้งก็สามารถที่จะใช้งานได้โดยสะดวก

โครงงานนี้ได้ทำการทดลองเชื่อมต่ออุปกรณ์คอมพิวเตอร์เข้าด้วยกันโดยใช้การส่ง

ผ่านข้อมูลทางระบบวิทยุ เพื่อหาข้อมูลที่เกี่ยวกับความผิดพลาดที่อาจเกิดขึ้นได้ในระบบ รวมทั้งทดลองใช้โปรแกรมที่สร้างขึ้นสนับสนุนการใช้งานในหลายลักษณะ เพื่อสังเกตความสัมพันธ์ระหว่างอัตราของการส่งผ่านข้อมูล การจัดรูปแบบของข้อมูลก่อนทำการส่ง และอุปกรณ์คอมพิวเตอร์ซึ่งมีการเชื่อมต่อกับอุปกรณ์ภายนอก เช่น เครื่องพิมพ์ ดิสด์ไดว์ (disk drive) และอุปกรณ์เก็บข้อมูล เป็นต้น

บทที่ 2

ทฤษฎีและหลักการ

2.1 การสื่อสารข้อมูล

การสื่อสารข้อมูล เป็นการแลกเปลี่ยนข่าวสารระหว่างอุปกรณ์ โดยผ่านตัวกลางที่มีหลายแบบ ข่าวสารที่ใช้แลกเปลี่ยนระหว่างกันนี้ จะเป็นข้อมูลดิจิตอล ซึ่งมีรูปแบบ หรือมาตรฐานที่สามารถเข้าใจได้ทั้งทางด้านส่งและทางด้านรับ มาตรฐานในการเข้ารหัสข้อมูลดิจิตอลที่มีใช้โดยทั่วไป คือ รหัส EBCDIC (EXTEND BINARY CODED DECIMAL INTERCHANGE CODE) และรหัสแอสกี้ ซึ่งรหัส EBCDIC นี้ใช้ในการสื่อสารข้อมูลเพียงบางแห่งเท่านั้น เนื่องจากระหัสชนิดนี้ ใน 1 ไบต์จะประกอบด้วยข้อมูล 8 บิท ซึ่งไม่มีที่ว่างเหลือสำหรับบิทพาริตี้

รหัสที่ใช้มากในปัจจุบัน คือ รหัสแอสกี้ ซึ่งในหนึ่งไบต์ประกอบด้วย 7 บิท โดยบิทที่ 8 เป็นบิทพาริตี้ (ดูรายละเอียดของรหัส EBCDIC และรหัสแอสกี้ในภาคผนวก ข.)

2.2 การส่งผ่านข้อมูล

สามารถส่งได้ 2 แบบคือ

2.2.1 การส่งข้อมูลแบบขนาน

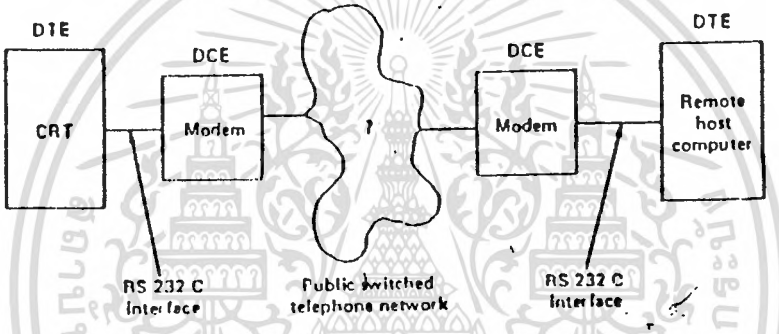
ข้อมูลทุก ๆ บิทจะถูกส่งออกไปพร้อม ๆ กันในครั้งเดียว เช่น ถ้ามีข้อมูล 1 ไบต์ (8 บิท) จะถูกส่งพร้อมกันทีเดียวทั้ง 8 บิท ทำให้ต้องใช้สายส่งข้อมูล (data bus) ถึง 8 เส้น ในระบบไมโครคอมพิวเตอร์ โดยทั่วไปการส่งผ่านข้อมูลระหว่างอุปกรณ์ต่าง ๆ ที่อยู่บนบอร์ด (board) เดียวกันหรือบอร์ดที่ใช้ร่วมภายในระบบจะใช้การส่งชนิดนี้

2.2.2 การส่งข้อมูลแบบอนุกรม

การส่งแบบนี้ ข้อมูลจะถูกส่งออกมาทีละบิทจาก อุปกรณ์ส่ง ไปยัง อุปกรณ์รับทำให้สามารถใช้ช่องสัญญาณเพียง 1 ช่อง และใช้สายส่งข้อมูลเพียง 1 เส้นซึ่งประหยัดกว่าการส่งแบบขนาน แต่อัตราการส่งข้อมูลจะช้ากว่าการส่งข้อมูลแบบขนานมาก และต้องมีการตรวจสอบข้อมูลระหว่างอุปกรณ์ส่งกับอุปกรณ์รับอยู่ตลอดเวลา

2.3 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน EIA RS-232-C

เราใช้มาตรฐาน RS-232-C ในการสื่อสารข้อมูลแบบอนุกรมระหว่าง อุปกรณ์ปลายทางข้อมูล (DATA TERMINAL EQUIPMENT) กับ อุปกรณ์สื่อสารข้อมูล (DATA COMMUNICATION EQUIPMENT) โดยอัตราการส่งข้อมูลจะถูกกำหนดให้อยู่ระหว่าง 0-20,000 บิทต่อวินาที (bps) ในการประยุกต์ใช้งาน RS-232-C อัตราเร็วสูงสุดที่ใช้ควรจะมีค่าไม่เกิน 19.2 kbps และไม่ควรใช้มาตรฐานการอินเตอร์เฟสแบบนี้ ในกรณีที่มีการแบ่งแยกทางไฟฟ้า (Electrical Isolation) ระหว่างอุปกรณ์ทั้ง 2 ด้านของการอินเตอร์เฟส และความยาวสายเคเบิ้ลที่ใช้ในการสื่อสารข้อมูลไม่ควรเกิน 50 ฟุต (เงื่อนไขนี้ขึ้นอยู่กับสภาพความเป็นตัวเก็บประจุที่จุดอินเตอร์เฟส)



รูปที่ 2.1 การเชื่อมต่อของระบบ

เนื่องจากความต้องการในการสื่อสารข้อมูลมีมากขึ้น จึงจำเป็นต้องมีการกำหนดมาตรฐานแก่อุปกรณ์ที่ถูกผลิตจากบริษัทต่าง ๆ คุณสมบัติพื้นฐานที่ได้กำหนดขึ้นคือ โครงสร้างทางกายภาพของการอินเตอร์เฟส (Physical Layer Communication Protocals) ซึ่งเป็นข้อกำหนดที่เกี่ยวกับคุณสมบัติทางสัญญาณไฟฟ้า และคุณสมบัติที่เกี่ยวกับหน้าที่ของสัญญาณ เหล่านี้ขนาดและรูปร่างของคอนเน็คเตอร์ (connector) ซึ่งมาตราฐานดังกล่าวที่ใช้ในโครงงานนี้คือ RS-232-C

2.3.1 โครงสร้างทางกลของการอินเตอร์เฟส

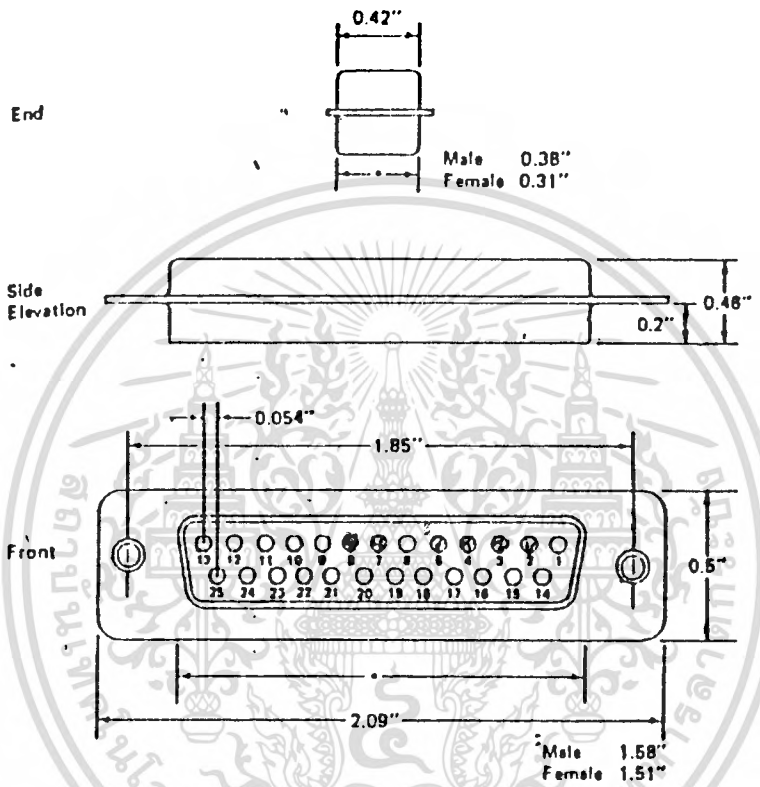
อินเตอร์เฟส RS-232-C ได้แสดงรายละเอียดของขาต่าง ๆ บนคอนเน็คเตอร์ (connector) ดังแสดงในตารางที่ 2.1 โดยที่ไม่ได้กล่าวถึงลักษณะของคอนเน็คเตอร์ที่ใช้ไว้เลย แต่โดยทั่วไปมักจะใช้คอนเน็คเตอร์แบบ DB-25 (D-Type 25 pin connector)

ซึ่งมีคุณสมบัติเทียบเท่าแบบ ISO 2113 กำหนดโดย International Standard Organization

ตารางที่ 2.1 รายละเอียดของขาต่าง ๆ

ขา	เซอร์กิต	ความหมายของเซอร์กิต
1	AA	Protective Ground
2	BA	Transmitted Data
3	BB	Received Data
4	CA	Request to Send
5	CB	Clear to Send
6	CC	Data Set Ready
7	AB	Signal Ground
8	CF	Received Line Signal Detector
9/10	-	(Reserved for Data Set Testing)
11	-	Unassigned
12	SCF	Secondary Received Line Signal Detector
13	SCB	Secondary Clear to Send
14	SBA	Secondary Transmitted Data
15	DB	Transmit Signal Element Timing (DCE Source)
16	SBB	Secondary Received Data
17	DD	Receive Signal Element Timing
18	-	Unassigned
19	SCA	Secondary Request to Send
20	CD	Data Terminal Ready
21	CG	Signal Quality Detector
22	CE	Ring Indicator
23	CH/CI	Data Signal Rate Select (DTE/DCE Source)
24	DA	Transmit Signal Element Timing (DTE Source)
25	-	Unassigned

คอนเน็คเตอร์แบบ DB-25 นี้จะแบ่งออกได้เป็น 2 ส่วนคือ คอนเน็คเตอร์ตัวผู้ (DB-25-P) และคอนเน็คเตอร์ตัวเมีย (DB-25-B) โดยทั่วไปคอนเน็คเตอร์ตัวผู้มักจะติดตั้งอยู่กับอุปกรณ์ปลายทางข้อมูล (DTE) ส่วนคอนเน็คเตอร์ตัวเมียจะอยู่กับอุปกรณ์สื่อสารข้อมูล (DCE) ดังรูปที่ 2.2

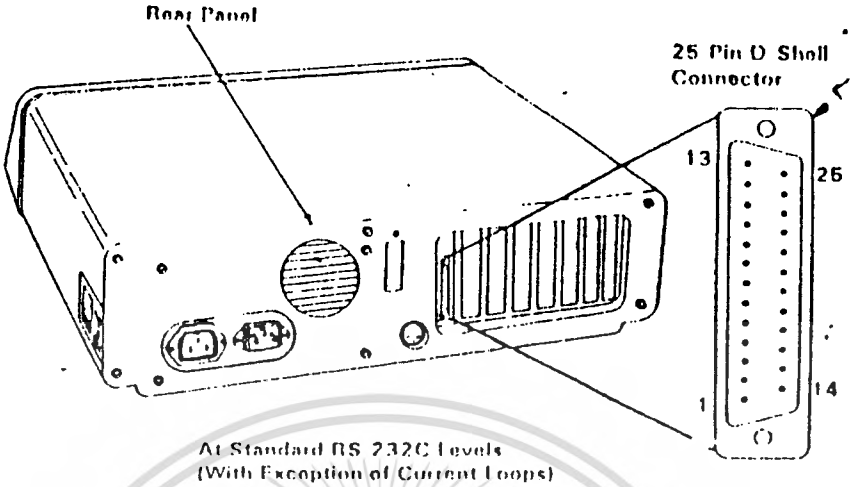


รูปที่ 2.2 ลักษณะทางกลของคอนเนคเตอร์แบบ DB-25

หน้าที่และระดับสัญญาณของขาสัญญาณต่าง ๆ ของอินเตอร์เฟส RS-232-C ดังแสดงในรูปที่ 2.3 มีดังนี้

ขาที่ 2: Transmitted Data (TD)

สัญญาณของเซอร์กิตนี้จะถูกส่งจากอุปกรณ์ปลายทางข้อมูลไปยังอุปกรณ์สื่อสารข้อมูล มีสถานะลอจิก "1" (Mark) ตลอดเวลาที่ไม่มีการส่งข้อมูล และจะไม่ทำการส่งข้อมูลนอกจากว่า ขาสัญญาณต่อไปนี้มีสถานะลอจิก "0" (Space)



At Standard RS 232C Levels
(With Exception of Current Loops)

	Description	Pin	
	NC	1	
◄	Transmitted Data	2	
►	Received Data	3	
◄	Request to Send	4	
►	Clear to Send	5	
►	Data Set Ready	6	
►	Signal Ground	7	
►	Received Line Signal Detector	8	
◄	(Transmit Current Loop Data	9	
	NC	10	
◄	Transmit Current Loop Data	11	
	NC	12	Asynchronous Communications Adapter (RS-232C)
External Device	NC	13	
	NC	14	
	NC	15	
	NC	16	
	NC	17	
►	(Receive Current Loop Data	18	
	NC	19	
◄	Data Terminal Ready	20	
	NC	21	
►	Ring Indicator	22	
	NC	23	
	NC	24	
►	Receive Current Loop Return	25	

Note: To avoid inducing voltage surges on interchange circuits, signals from * interchange circuits shall not be used to drive inductive devices, such as relay coils

รูปที่ 2.3 รูปแบบของคอนเนคเตอร์

- ขาที่ 4 (RTS) Request to send
- ขาที่ 5 (CTS) Clear to send
- ขาที่ 6 (DSR) Data set ready
- ขาที่ 20 (DTR) Data terminal ready

ขาที่ 3: Receive Data (RD)

สัญญาณของขานี้จะถูกส่งจาก DCE ไปยัง DTE เมื่อไม่มีสัญญาณเข้ามาจะมีสถานะเป็น "1" และในการส่งข้อมูลแบบ Half duplex เมื่อขาสัญญาณ Request to send อยู่ในสภาวะ ON ("0") ขาสัญญาณ Receive DATA เป็น OFF ("1")

ขาที่ 4: Request to send (RTS)

สัญญาณนี้จะถูกส่งจาก DTE ไปยัง DCE ในกรณีของ Half-Duplex เมื่อ Request to send อยู่ในสภาวะ ON จะทำให้ DCE อยู่ในโหมดการส่งข้อมูล แต่ถ้า Request to send อยู่ในสภาวะ OFF ทำให้ DCE จะอยู่ในโหมดการรับข้อมูล

ขาที่ 5: Clear to send (CTS)

สัญญาณนี้เป็นสัญญาณควบคุมซึ่งถูกส่งจาก DCE ไปยัง DTE เพื่อบอกให้ DTE ทราบว่า DCE พร้อมที่จะรับข้อมูลที่จะส่งมาจาก DTE บนสาย Transmitted Data แล้ว เมื่อสัญญาณ Clear to send อยู่ในสภาวะ ON รวมทั้งสัญญาณ Request to send และ Data Set Ready หรือ Data Terminal Ready อยู่ในสภาวะเป็น ON ด้วย การ ON ของสัญญาณเหล่านี้จะบอกให้ DTE ทราบว่า ข้อมูลที่ส่งไปยัง DCE จะถูก DCE รับไว้และส่งต่อไปยัง Communication Channel และเมื่อสัญญาณ Clear to send อยู่ในสภาวะ OFF จะบอกให้ DTE ทราบว่า DCE ไม่พร้อมที่จะรับข้อมูล ดังนั้น DTE จะยังไม่ส่งข้อมูลออกมาจนกว่าจะเริ่มต้นขบวนการใหม่โดยการส่ง Request to send ใหม่

ขาที่ 6: Data set ready (DSR)

สัญญาณนี้เป็นสัญญาณควบคุมที่ส่งจาก DCE ไปยัง DTE ในกรณีที่ Data set ready อยู่ในสภาวะ ON ซึ่งแสดงว่า DCE ได้ถูกต่อกับช่องสัญญาณของโทรศัพท์เรียบร้อยแล้ว และระบบเริ่มเข้าสู่โหมดการส่งข้อมูล

ขาที่ 7: Signal ground (SG)

ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุก ๆ วงจรของสายสัญญาณทุกสายยกเว้น Protective ground

ขาที่ 20: Data terminal ready (DTR)

สัญญาณควบคุมนี้จะถูกส่งจาก DTE ไปยัง DCE ซึ่งจะมีสภาวะ ON ก่อนที่ DCE จะ ON สัญญาณ Data set ready ซึ่งเป็นการบอกให้อุปกรณ์ทราบว่า DCE ถูกเชื่อมต่อเข้ากับช่องทางการสื่อสารแล้ว และสามารถส่งข้อมูลได้ขณะใดก็ตามที่ DCE ต่อกับช่องทางการสื่อสารแล้ว Data terminal ready จะเปลี่ยนสภาวะจาก ON เป็น OFF

ขาที่ 8: Receive line signal detector (DCD)

สัญญาณนี้จะถูกส่งจาก DCE ไปยัง DTE เมื่อ DCE ได้รับสัญญาณพาหะ (carrier) ที่ส่งมาจาก DCE อีกด้านหนึ่ง สัญญาณ Receive line signal detector จะมีสภาวะเป็น ON แสดงว่า DCE จับสัญญาณในช่องสัญญาณซึ่งจะถูกนำไป Demodulate ได้แล้ว

ขาที่ 22: Ring indicator (RI)

เป็นขาที่ DCE จะให้สถานะเป็น ON ออกมาเมื่อ DCE ได้รับสัญญาณ Ringing แต่ขานี้จะไม่ทำงาน ถ้าสัญญาณ Data terminal ready มีสถานะเป็น OFF

ขาที่ 21: Signal quality detector (SQ)

สัญญาณที่ขานี้เป็นขาที่ DCE ใช้แสดงว่าสัญญาณข้อมูลที่ได้รับมามีโอกาสเกิดการผิดพลาดสูงหรือไม่ ถ้าเป็นสถานะ ON แสดงว่าโอกาสเกิดการผิดพลาดจะต่ำ แต่ถ้ามีสถานะ OFF แสดงว่ามีโอกาสสูงที่จะรับข้อมูลผิดพลาดเพราะ DCE จะตรวจสอบกับสัญญาณคลื่นพาหะว่ามีคุณภาพดีหรือไม่ มีสัญญาณรบกวนมากหรือไม่

ขาที่ 23: Data signal rate selector

เป็นขาสัญญาณที่ใช้ควบคุมอัตราเร็วในการส่งข้อมูล ทำให้สามารถส่งข้อมูลด้วยอัตราเร็ว 2 อัตราได้ เมื่ออยู่ในสถานะ "ON" อัตราเร็วในการส่งข้อมูลที่สูงกว่าจะถูกเลือกใช้ ขาสัญญาณนี้สามารถควบคุมได้จาก DTE หรือ DCE ขึ้นอยู่กับการนำไปใช้งาน

ขาที่ 24: Transmitter signal element timing

เป็นขาที่ DTE ใช้ในการส่งสัญญาณ เพื่อไปควบคุมจังหวะการทำงาน (Clocking signal) ของ DCE ในการส่งข้อมูล

ขาที่ 25: Transmitter signal element timing

เป็นขาที่ DCE ใช้ส่งสัญญาณ เพื่อไปควบคุมจังหวะการทำงานของ DTE ในการส่งข้อมูล สัญญาณควบคุมจังหวะการทำงานนี้จะมีใช้เฉพาะใน ซิงโครนัสโมเด็ม (Synchronus Modem) เท่านั้น

ขาที่ 17: Receive signal element timing

เป็นขาที่ DCE ใช้ส่งสัญญาณ เพื่อไปควบคุมจังหวะการทำงานของ DTE ในการรับข้อมูล

ขาที่ 14: Secondary Transmitted Data

ขาที่ 16: Secondary Received Data

ขาที่ 19: Secondary Request to Send

ขาที่ 13: Secondary Clear to Send

ขาที่ 12: Secondary Received Line Signal Detector

ขาสัญญาณ Secondary ทั้ง 5 นี้ จะใช้สำหรับช่องสัญญาณช่องที่ 2 (Secondary Channel) ในลักษณะเช่นเดียวกับช่องสัญญาณที่ 1 แต่ช่องสัญญาณที่ 2 นี้จะมีทิศทางตรงกันข้ามกับช่องสัญญาณที่ 1 และช่องสัญญาณที่ 2 นี้จะอยู่ภายใต้การควบคุมของขาสัญญาณ DTR และ DSR ด้วย

2.3.2 คุณสมบัติทางไฟฟ้าของการอินเตอร์เฟส

อินเตอร์เฟส RS-232-C ใช้ลอจิกลบ (negative logic) แทนระดับแรงดันต่าง ๆ (ลอจิคลบ คือวิธีการเปรียบเทียบแรงดันแบบหนึ่ง ถ้าระดับแรงดันหนึ่งมีค่าเป็นลบมากกว่าอีกระดับแรงดันหนึ่ง ระดับแรงดันนั้นจะมีลอจิคเป็น "สูง" ดังนี้คือ ถ้าระดับแรงดันมีค่า -V = "1" มีความหมาย OFF และ +V = "0" มีความหมาย ON) โดยระดับแรงดันของสัญญาณดังกล่าวจะถูกวัดเทียบกับ Signal ground ดังตารางที่ 2.2

ตารางที่ 2.2

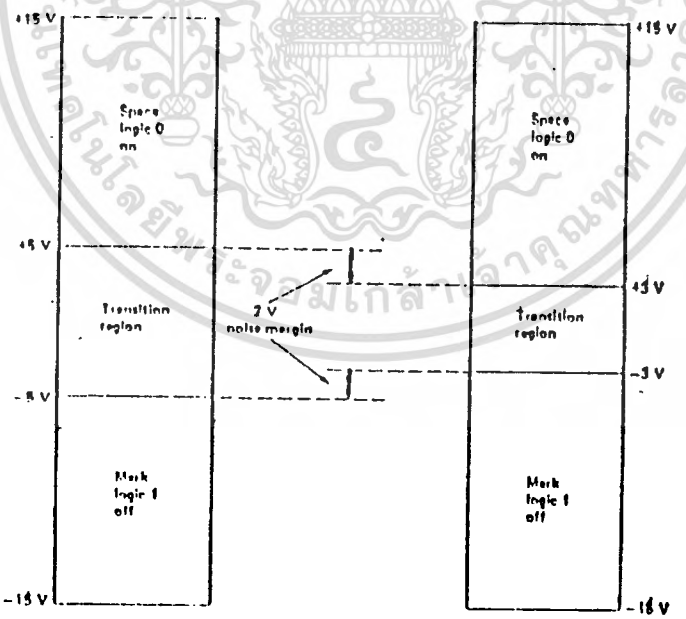
	Signal Voltage	
Status	$-25V < V_1 < -3V$	$3V < V_1 < 25V$
Binary logic state	1	0
Signal condition	MARK	SPACE
Function	OFF	ON

ในการแทนลอจิค "1" หรือ Mark ตัวขับสัญญาณ Driver ต้องจ่ายระดับแรง

ดันระหว่าง -5 ถึง -15 โวลต์ ส่วนในการแทนลอจิก "0" หรือ Space ตัวขับสัญญาณต้องจ่ายแรงดันระหว่าง +5 ถึง +15 โวลต์ ซึ่งความสัมพันธ์ระหว่างระดับแรงดันและสถานะของสัญญาณจะเป็นดังรูปที่ 2.4 ซึ่งจากรูปจะพบว่าตัวกำเนิดสัญญาณต้องการส่งลอจิก "0" ตัวขับสัญญาณต้องจ่ายแรงดันระหว่าง +5 ถึง +15 โวลต์ ส่วนตัวรับสัญญาณปลายทาง (line receiver) จะถือว่าแรงดันที่อยู่ภายในช่วง +3 ถึง +15 โวลต์ แทนลอจิค "0" จะเห็นว่า RS-232-C ยอมให้มี Noise margine ได้ไม่เกิน 2 โวลต์

มาตรฐาน RS-232-C ยังได้กำหนดเพิ่มอีกว่า ตัวเก็บประจุที่ต่อขนานกับอุปกรณ์รับข้อมูลปลายทางจะต้องมีค่าไม่เกิน 2,500 PF โดยค่านี้ไม่รวมค่าความจุไฟฟ้าของสายเคเบิ้ลด้วย รวมทั้งแรงดันขณะเปิดวงจรหรือขณะที่ไม่มีโหลด แต่ต้องไม่เกิน 25 โวลต์ ซึ่งก็คือแรงดันใด ๆ ในเซอร์กิตของการอินเตอร์เฟสแบบ RS-232-C ต้องไม่เกิน 25 โวลต์

วงจรรับสัญญาณที่ใช้กับ RS-232-C ต้องสามารถทนต่อการลัดวงจรที่เกิดขึ้นได้โดยไม่ทำให้เกิดความเสียหายต่อตัวมันเองหรืออุปกรณ์ที่เกี่ยวข้องด้วย เช่น โมเด็ม พอร์ท I/O และอุปกรณ์ต่าง ๆ ที่ต่อเข้ากับเคเบิ้ลที่ใช้ในการอินเตอร์เฟสแบบ RS-232-C



รูปที่ 2.4 คุณสมบัติทางไฟฟ้าของการอินเตอร์เฟสแบบ RS-232-C

2.4 การใช้โปรแกรมควบคุมการส่งผ่านข้อมูล

เนื่องจากลักษณะของการส่งผ่านข้อมูลดังที่ได้กล่าวมาในบทก่อน ๆ ในระหว่างการส่ง ข้อผิดพลาดที่อาจเกิดขึ้นได้มีหลายสาเหตุ ดังเช่น

ก. เครื่องส่งทำการส่งข้อมูลทั้ง ๆ ที่เครื่องรับยังไม่พร้อมที่จะรับข้อมูล

ข. เครื่องรับสุ่มรับข้อมูลในช่วงจังหวะที่ไม่สอดคล้องกับทางด้านส่งทำให้รับข้อมูลที่คลาดเคลื่อนเข้ามา

ค. เครื่องส่งหลาย ๆ เครื่องส่งข้อมูลเข้ามายังเครื่องรับเครื่องเดียวในเวลาเดียวกัน

ง. ประสิทธิภาพของอุปกรณ์ลดลงเมื่อมีการใช้งาน

ข้อบกพร่องต่าง ๆ เหล่านี้สามารถแก้ไขได้ โดยการปรับปรุงโครงสร้างทางฮาร์แวร์ของระบบ ซึ่งต้องเสียค่าใช้จ่ายมากขึ้น อีกวิธีหนึ่งที่ใช้ค่าใช้จ่ายน้อยกว่าคือ ใช้โปรแกรมควบคุมผ่านซีพียู (CPU)

ซึ่งข้อบกพร่องเหล่านี้แก้ไขได้ ใน 2 ลักษณะ คือ

- ปรับปรุงโครงสร้าง ทางฮาร์ดแวร์ของระบบ
- สร้างซอฟท์แวร์สนับสนุนการทำงาน

วิธีแรกเป็นวิธีที่ดีที่สุดเพราะสามารถแก้ไขได้ตั้งแต่ระดับต่ำสุด คือการเชื่อมต่อทางกลจนถึงระดับที่ใช้งานจริง ทำให้ประสิทธิภาพของการส่งข้อมูลเพิ่มขึ้น รวมทั้งสามารถเพิ่มอัตราเร่งการส่งข้อมูลได้ในเวลาเดียวกัน แต่ต้องเสียค่าใช้จ่ายเพิ่มมากขึ้นด้วย ในกรณีที่ใช้วิธีสองจะเสียค่าใช้จ่ายน้อยกว่ามาก และผลที่ได้จะใกล้เคียงกัน ซึ่งในโครงงานนี้ได้ทดลองสร้างซอฟท์แวร์ที่ช่วยลดความผิดพลาดของข้อมูล (error) สามารถสื่อสารและใช้งานร่วมกับอุปกรณ์รอบนอกอื่น ๆ รวมทั้งทดลองสร้างโปรแกรมสนับสนุนการทำงานในส่วนของโปรแกรมสื่อสารข้อมูลด้วย

ในการสร้างโปรแกรมส่งผ่านข้อมูลที่ใช้กับเครื่องไมโครคอมพิวเตอร์ 16 บิท IBM PC/XT จำเป็นต้องมีการติดต่อกับพอร์ทสื่อสาร (Com-port) ในตัวเครื่อง ซึ่งผู้สร้างได้ออกแบบให้เป็นไปตามมาตรฐานการเชื่อมต่อแบบอนุกรม ที่เรียกว่า RS-232-C ซึ่งมีโครงสร้างที่สามารถโปรแกรมด้วยการส่งรหัสคำสั่ง ไปควบคุมการทำงานของไอซีหลัก คือ ชิพ 8250 (โปรดดูภาคผนวก ค)

IC 8250 นี้ เป็น IC ขนาด 40 ขา มีขีดความสามารถพิเศษ ดังนี้

- มีบัฟเฟอร์ (buffer) ในตัว ทำให้ไม่จำเป็นต้องมีการซิงโคร์ไนซ์ของการรับส่ง
- ใช้สัญญาณนาฬิกาอิสระต่างหาก ไม่ขึ้นกับสัญญาณนาฬิกาของระบบ
- มีสัญญาณตอบโต้และควบคุมโมเด็ม
- สามารถตรวจสอบบิทเริ่มต้น (Start bit) ที่ผิดพลาด
- สามารถตรวจสอบและสร้างสัญญาณสายขาด (line break) เพื่อใช้ในการตรวจสอบการทำงานของระบบ

IC 8250 จะเป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือ สล๊อตขนาด 62 ขา ของระบบนั่นเอง และซีพียูจะติดต่อกับ IC 8250 ในลักษณะที่เป็นพอร์ต อินพุท เอ้าท์พุท การจัดพอร์ตนี้กำหนดหมายเลขพอร์ตอย่างเจาะจง ซึ่งในเครื่องไมโครคอมพิวเตอร์ IBM PC/XT, IBM PC/AT จะมีพอร์ตสื่อสาร 2 พอร์ต คือ คอม 1 (COM 1) และ คอม 2 (COM 2) โดยที่คอม 1 จะมีหมายเลขอินพุท เอ้าพุทพอร์ตคือ 3FX (3F8 - 3FE) และคอม 2 จะมีหมายเลขอินพุท เอ้าท์พุทพอร์ตคือ 2FX (2F8-2FE) (โปรดดูภาคผนวก ค)

จากคุณสมบัติของไอซี 8250 ที่กล่าวมา จะพบว่ามีการตรวจสอบข้อผิดพลาดต่างๆ ที่เกิดขึ้น แต่เมื่อทดลองทำการรับส่งจะเกิดความผิดพลาดขึ้นจากหลายสาเหตุ เพราะฉะนั้น การใช้การตรวจจับความผิดพลาดของตัวไอซีไม่เพียงพอ จำเป็นต้องมีการแก้ไขและตรวจสอบจากหลายสาเหตุนอกเหนือจากนี้ ดังจะได้กล่าวต่อไปในบทที่ 3 (การออกแบบและการสร้าง)

โดยทั่วไปมีการเขียนโปรแกรมที่เกี่ยวกับการสื่อสารข้อมูล มักจะคำนึงถึงความเร็วของภาษาที่ใช้ ความสะดวกในการสร้างรูปแบบการทำงาน รวมทั้งความสะดวกในการเขียน ซึ่งในปัจจุบันที่นิยมใช้มากที่สุดคือ ภาษาปาสคาล ภาษาซี ภาษาแอสเซ็มบลี รูปแบบของภาษาปาสคาลและภาษาซีมีความใกล้เคียงกันมาก แต่ภาษาซีมีข้อดีกว่าคือ มีความเร็วมากกว่า ในโครงงานนี้ได้ทดลองใช้ภาษาปาสคาลสร้างโปรแกรมส่งผ่านข้อมูล โดยแบ่งฟังก์ชั่นการทำงานออกตามลักษณะการใช้งาน 3 แบบคือ

1. ส่งผ่านข้อมูลในลักษณะของข่าวสาร ไฟล์ กราฟฟิค โดยควบคุมการทำงานจากต้นทาง
2. การใช้โปรแกรมสนับสนุน (Utility)
3. ใช้งานร่วมกับอุปกรณ์ภายนอก (อุปกรณ์เก็บข้อมูลตามช่วงเวลา)

ในฟังก์ชั่นการทำงานแบบที่ 1 นั้น จุดประสงค์ก็คือ ทดลองส่งข้อมูลที่จะนำไปใช้

ในหลายลักษณะ เนื่องจากมีการติดต่อกับอุปกรณ์ภายนอกที่ต่างกัน ทดลองหาความผิดพลาดที่อาจจะเกิดขึ้นในระบบ ความเร็วของการทำงานในลักษณะต่าง ๆ เพื่อนำไปสร้างรูปแบบของโปรแกรมที่มีความเหมาะสมต่อไป

ฟังก์ชั่นการทำงานแบบที่ 2 นั้น มีจุดประสงค์เพื่อเพิ่มประสิทธิภาพของการใช้งานอุปกรณ์สื่อสารข้อมูลโดยทดลองสร้างโปรแกรมสนับสนุน ใช้หลักการของการทำงานคล้ายกับโปรแกรมไซด์คิก (Side Kick) คือสร้างโปรแกรมในส่วนของการสื่อสารข้อมูลให้ฝังตัวอยู่ในหน่วยความจำของเครื่อง PC ที่ใช้ (Program Residence) สามารถเรียกใช้ได้โดยกด key ตั้งแต่ 1 ถึง 3 key บน key board ในขณะที่กำลังทำงานกับโปรแกรมอื่นอยู่ได้ ซึ่งหมายความว่า ผู้ใช้สามารถที่จะรอการติดต่อและส่งข้อมูลในขณะเดียวกันก็สามารถทำงานกับโปรแกรมอื่น ซึ่งถ้ามีข้อมูลหรือการติดต่อเข้ามาก็สามารถหยุดการทำงานของโปรแกรมอื่นชั่วคราวและเรียกใช้โปรแกรมสื่อสารข้อมูลได้ โดยที่ข้อมูลที่รับเข้ามาไม่มีผลต่อข้อมูลเดิมของโปรแกรมเดิม เมื่อสิ้นสุดการติดต่อสามารถที่จะกลับไปยังโปรแกรมเดิมเพื่อทำงานต่อไปได้

ในการสร้างโปรแกรมสนับสนุนนั้นจะต้องมีการติดต่อกับหน่วยความจำของเครื่อง PC และพารามิเตอร์ต่าง ๆ ที่มีการใช้ในภาษาปาสคาล เช่น ส่วน Data transfer area (DTA) , Memory control block (MCB), Programme segment prefix (PSP), พารามิเตอร์ที่ใช้เรียกเข้าหาระบบ ซึ่งความสัมพันธ์ของหน่วยความจำและพารามิเตอร์ต่าง ๆ มีหลักการดังนี้

เมื่อเริ่มเปิดเครื่องซอฟท์แวร์ใน ROM จะทำการตรวจสอบความพร้อมของอุปกรณ์และตรวจสอบอุปกรณ์ที่ติดตั้งเพิ่มขึ้น (self test) เมื่อ self test ตรวจสอบไม่มีปัญหาซอฟท์แวร์ใน ROM จะเริ่มต้นการทำงานของ DOS (Disk Operation System) โดยรูทีนบูทสแตรป โหลดเดอร์ (Routine Bootsrap Loader) จะอ่านบูทเรคคอร์ด (Boot Record) จาก Sector แรกในแผ่นดิส (ถ้าไม่พบ Boot Record เครื่องจะกระโดดไปทำงานในพื้นที่ของ ROM - Basic แทน

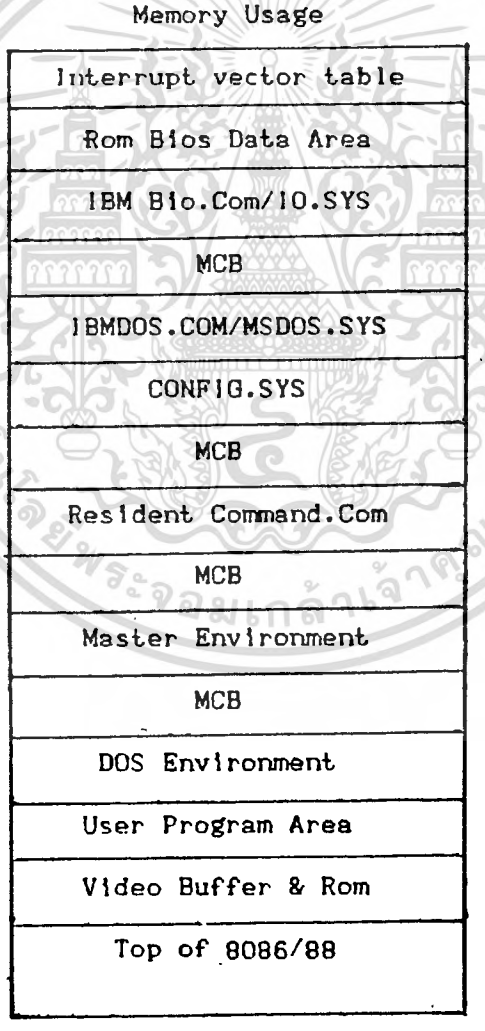
หลังจากนั้นโปรแกรมในส่วน Boot จะทำการโหลดไฟล์ที่เกี่ยวกับช้องกับการทำงานของระบบในหน่วยความจำ RAM ของเครื่อง ดังรูปที่ 2.5

จะพบว่า หลังจากดอสได้โหลดไฟล์ IBM Bio.Com/IO.SYS ลงในหน่วยความจำแล้ว (ส่วนนี้จะเป็นส่วนที่จัดการกับอุปกรณ์อินพุท เอ้าพุท และขยายการทำงานของ รอม. ไบออส ในเวอร์ชั่นที่ต่างกัน ซึ่งโปรแกรมโดยทั่วไปจะมีการติดต่อกับส่วนนี้น้อยมาก) ไฟล์ถัด

มาจะต้องมี Memory Control Block (MCB) ที่ตำแหน่งเริ่มต้นของพื้นที่หน่วยความจำนั้น ซึ่ง MCB จะมีโครงสร้างดังรูปข้างล่าง

Format MCB	ID	Chain Segment	Block size	No use
Byte	0	1-2	3-4	5-15

ID: ถ้าไบท์นี้เป็นตัวอักษร "M" แสดงว่ายังมีบล๊อกต่อจากนี้ไป แต่ถ้าเป็นตัว "Z" แสดงว่าสิ้นสุดบล๊อก



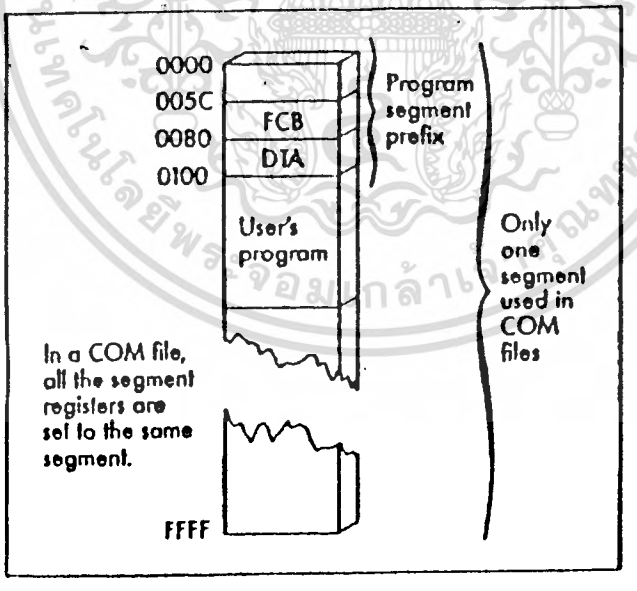
รูปที่ 2.5 การจัดหน่วยความจำของ PC (PC Memory Map)

Chain Segment:: ปกติจะชี้ไปที่ PSP ของโปรแกรมว่าหน่วยความจำบล็อกนี้มีโปรแกรมชื่ออะไรเป็นเจ้าของ (อาจยกเว้นในบางกรณี)

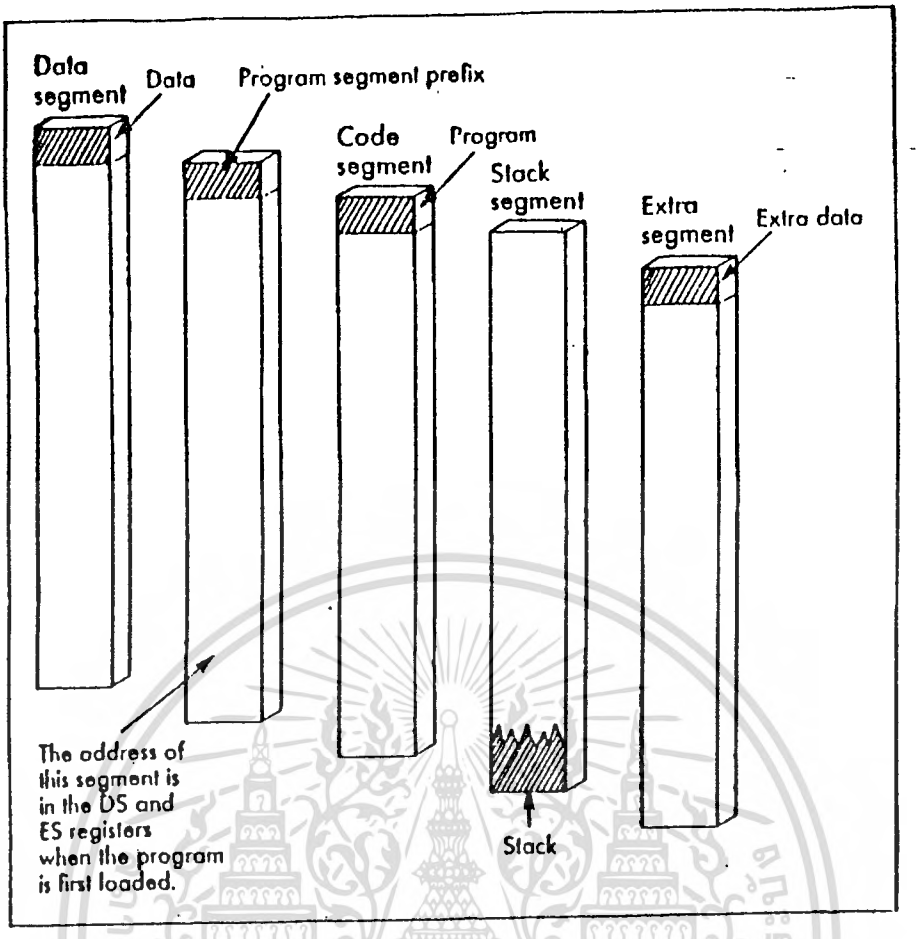
Block Size: จะบอกว่าหน่วยความจำบล็อกนี้ครอบครองพื้นที่หน่วยความจำไว้กี่ paragraph

หน้าที่ของ MCB นั้นจะทำการเชื่อมโยงในลักษณะลูกโซ่ (Chain) ของบล็อกหน่วยความจำของไฟล์ต่าง ๆ ทำให้สามารถรู้ได้ว่า ไฟล์หนึ่ง ๆ อยู่ ณ. พารากราฟไหนของหน่วยความจำบ้าง ขนาดของพารากราฟแต่ละพารากราฟ แสดงตำแหน่งพารามิเตอร์ของโปรแกรม เช่นส่วน Environment, Program Segment Prefix (PSP) ซึ่งจะพบว่า MCB มีความจำเป็นต่อระบบจัดการของ DOS มาก

จากรูปที่ 2.5 ในส่วนพื้นที่โปรแกรมของผู้ใช้ (User program area) เป็นพื้นที่หน่วยความจำที่ใช้งานเมื่อมีการโหลดโปรแกรมเข้ามา ซึ่งโปรแกรมที่โหลดเข้ามานี้ส่วนแรกจะเป็นเนื้อที่ที่ DOS จัดสรรไว้เป็นที่เก็บข้อมูลในการช่วยดำเนินโปรแกรม เช่นการจัดสภาพแวดล้อม (Environment) การควบคุมแฟ้ม ประกอบไปด้วยส่วนต่าง ๆ ดังรูปที่ 2.6 a และ 2.6 b



รูปที่ 2.6 a Programme Segment Prefix ของไฟล์ .com



รูปที่ 2.6 b Programme Segment Prefix ของไฟล์ .exe

*DTA: Data Transfer Area มีขนาด 128 ไบท์ ใช้เป็นเนื้อที่ส่งผ่าน parameter ระหว่างดิสค์กับโปรแกรม ระหว่างการอ่านและเขียนไฟล์ (ใน DOS version ใหม่จะใช้บัฟเฟอร์ แทน DTA แต่ลักษณะการใช้งานจะคล้ายคลึงกัน)

ในฟังก์ชั่นการทำงานแบบที่ 3 นี้มีจุดประสงค์เพื่อ ทดลองทำการเชื่อมต่อ (interface) อุปกรณ์สื่อสารข้อมูลกับอุปกรณ์ภายนอกโดยให้มีการรับส่งข้อมูลในช่วงเวลาที่คงที่ค่าหนึ่ง

2.5 การใช้งานฮาร์ดแวร์อินเตอร์รัพ (Hardware interrupt)

เมื่อทำการอินเตอร์เฟสอุปกรณ์ภายนอก เช่น โมเด็มหรือคีย์บอร์ดเข้ากับไมโครโปรเซสเซอร์ เราสามารถกำหนดให้เกิดการอินเตอร์รัพจากอุปกรณ์ภายนอกเหล่านั้นได้ซึ่งขบวนการอินเตอร์รัพนี้ มีขั้นตอนโดยย่อดังนี้

1) วงจรอินเตอร์เฟส (อุปกรณ์ภายนอก) ส่งสัญญาณเพื่อขออินเตอร์รัพ

2) 8259 ได้รับสัญญาณอินเตอร์รัพและเซ็ทอินเตอร์รัพรีจิสเตอร์ (interrupt register) ที่ตรงกับช่องสัญญาณ (channel) ที่ขออินเตอร์รัพนั้น

3) 8259 ส่งสัญญาณ int ให้กับ 8088 เพื่อทำการขออินเตอร์รัพ

4) 8088 ทำการตอบรับการขออินเตอร์รัพโดยการส่งพัลส์ (pulse) ลูกแรกของสัญญาณ INTA ให้กับ 8259 ซึ่งจะทำให้รีจิสเตอร์แสดงการตอบสนองอินเตอร์รัพถูกเซ็ท และ interrupt register ของแชนแนลนั้นถูกรีเซ็ทด้วย

5) 8088 ทำการส่งพัลส์ลูกที่สองของสัญญาณ INTA ให้กับ 8259 จากนั้นชิพ 8259 ก็จะส่งเวคเตอร์ของการอินเตอร์รัพออกมาบนบัสข้อมูล 8088 จะใช้ค่าเวคเตอร์นี้ไปเปิดตารางเวคเตอร์ เพื่อหาตำแหน่งที่อยู่เริ่มต้น (ค่าที่อยู่นี้จะถูกโหลดให้กับรีจิสเตอร์ cs:ip) ของโปรแกรมที่ตอบสนองต่อการขออินเตอร์รัพที่เกิดขึ้น

6) 8088 จะทำการ disable การขออินเตอร์รัพ (แฟล็ก i จะถูก reset โดยอัตโนมัติ) เก็บข้อมูลในรีจิสเตอร์แฟล็ก (คือสถานะการทำงานของ 8088 จะกลับมาทำหลังจากเสร็จจากการทำงานในโปรแกรมตอบสนองการอินเตอร์รัพแล้ว และทำการโหลดค่า cs และ ip ที่อยู่ในตำแหน่งที่กำหนดโดยอินเตอร์รัพเวคเตอร์ (interrupt vector) จาก vector table ซึ่งจะเป็นผลให้ชิพ 8088 กระโดดไปทำงานในตำแหน่งที่กำหนดโดยค่าของ cs และ ip ที่โหลดเข้ามาใหม่นี้

7) โปรแกรมตอบสนองการอินเตอร์รัพสั่งให้ 8088 ทำการเก็บค่าของรีจิสเตอร์ต่าง ๆ ที่ถูกใช้ในโปรแกรมตอบสนองอินเตอร์รัพนี้ไว้บนสแต็ก (คำสั่ง push) เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้

8) กรณีที่ต้องการให้มีอินเตอร์รัพซ้อนขึ้นได้โปรแกรมการตอบสนองอินเตอร์รัพจะส่งคำสั่ง EOI ให้กับ 8259 และเซ็ทแฟล็ก i ของ 8088

9) 8088 เข้าสู่การทำงานในส่วนของโปรแกรม ที่ตอบสนองต่ออุปกรณ์ที่ขออินเตอร์รัพ

10) หลังเสร็จจากการทำงานในโปรแกรมตอบสนองอินเตอร์รัพแล้ว ต้องใส่คำสั่ง pop เพื่อดึงเอาค่าของรีจิสเตอร์ต่าง ๆ ที่เก็บไว้บนสแต็ก (ในขั้นตอนที่ 7) ออกมาให้กับรีจิสเตอร์ต่าง ๆ เหล่านั้นตามเดิม

11) ภายในโปรแกรมตอบสนองการอินเตอร์รัพจะสิ้นสุดด้วยคำสั่ง iret

(return from interrupt) ซึ่งจะทำให้ 8088 ทำการอีนาเบิลการขออินเตอร์รัพและนำเอาค่าของรีจิสเตอร์ cs, ip และ flag ที่เก็บไว้บนสแต็กมาคืนให้กับรีจิสเตอร์ทั้งสามตามเดิม ซึ่งจะทำให้ 8088 กลับเข้าสู่การทำงานในโปรแกรมเดิมก่อนอินเตอร์รัพได้ตามปกติ

บทที่ 3

การออกแบบและการสร้าง

3.1 วัตถุประสงค์

โครงงานนี้เป็นการทดลองสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์ (IBM PC/XT) 2 ตัว ผ่านอุปกรณ์โมเด็ม ความเร็ว 1,200 baud ทั้ง 2 ด้าน (โปรดดูภาคผนวก ง) มอดูเลท (Modulate) และส่งออกด้วยความถี่ 144 ถึง 146 MHz โดยใช้โปรแกรมควบคุมการทำงานในลักษณะของการสุ่มรับข้อมูล (Polling) และการใช้ฮาร์ดแวร์อินเทอร์รัพท์ (Hardware Interrupt) รวมทั้งทดลองส่งข้อมูลด้วยค่าความถี่คงที่ค่าหนึ่ง จากอุปกรณ์อินเตอร์เฟสภายนอก

ในการใช้โปรแกรมภาษา Turbo Pascal สร้างโปรแกรมที่ใช้สำหรับการสื่อสารข้อมูลจะต้องอาศัยการติดต่อกับ i/o port โดยตรง (i/o port ตำแหน่งที่ 03F8H - 03FEH) ในการอ่านและเขียน และติดต่อส่งผ่านข้อมูลกับ Register ของ 8250 ซึ่งเป็นอะแดปเตอร์ (adapter) สื่อสารข้อมูลแบบอะซิงโครนัส

หน้าที่โดยทั่วไปของโปรแกรมสื่อสารข้อมูล คือ จัดเก็บข้อมูลเป็นไฟล์ จัดเก็บข้อมูลลงดิสก์ ลดค่าความผิดพลาดที่อาจจะเกิดขึ้นให้น้อยที่สุด ลดความยุ่งยากในการใช้งานลง เพิ่มประสิทธิภาพในการใช้อุปกรณ์

ในโครงงานนี้ได้ทดลองสร้างโปรแกรมสำหรับสื่อสารข้อมูลโดยแบ่งลักษณะการใช้งาน ดังจะได้กล่าวต่อไป

3.2 การออกแบบอุปกรณ์เก็บข้อมูล

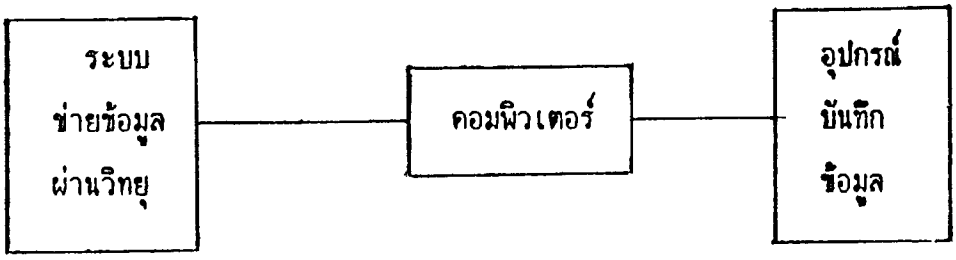
อุปกรณ์เก็บข้อมูลที่ใช้ร่วมในโครงงานนี้ จะต้องสามารถทำหน้าที่เก็บบันทึกข้อมูลต่าง ๆ เพื่อที่จะได้นำข้อมูลเหล่านั้นมาใช้ในการส่งผ่านระบบวิทยุ จะต้องมีชุดการทำงานตามหน้าที่ต่าง ๆ ดังต่อไปนี้คือ

ก. มีการเก็บข้อมูลเป็นระยะเวลาที่คงที่

ข. สามารถต่อเชื่อมเข้ากับระบบข่ายข้อมูลได้

ค. มีความแม่นยำในการเก็บข้อมูล

ง. มีการเก็บข้อมูลไว้ในหน่วยความจำสำรองซึ่งสามารถดึงเอาข้อมูลที่มีอยู่มาวิเคราะห์ได้



รูปที่ 3.1 บล๊อกไดอะแกรมการเชื่อมโยงของอุปกรณ์เก็บข้อมูล
กับข่ายข้อมูลผ่านระบบวิทยุ

3.3 การสร้างอุปกรณ์บันทึกข้อมูล

ในการสร้างอุปกรณ์เก็บข้อมูลนี้จะต้องอาศัยการทำงานของวงจรหลายชนิด ได้แก่ วงจรผลิตความถี่ วงจรหารความถี่ วงจรนับ มีการเปลี่ยนสัญญาณจากอนาลอกเป็นดิจิตอล มีตัวแรมเก็บข้อมูล ซึ่งวงจรต่าง ๆ เหล่านี้ก็อยู่ในไอซีที่จะเลือกมาใช้ดังรายละเอียดข้างล่าง

3.3.1 การผลิตความถี่เพื่อนำมาเป็นสัญญาณนาฬิกาของอุปกรณ์เก็บข้อมูล ได้นำเอาไอซี 555 มาใช้งาน ซึ่งไอซี 555 นี้สามารถผลิตความถี่ตามต้องการได้โดยการทำเป็นวงจรที่ให้กำเนิดสัญญาณพัลส์ได้อย่างต่อเนื่องหรือที่เรียกว่า อะสเตเบิ้ลมัลติไวเบรเตอร์ ซึ่งวงจรพื้นฐานแสดงดังรูป 3.2

ช่วงเวลาของพัลส์จะถูกกำหนดด้วยเวลาที่ใช้ในการประจุด้วยตัวเก็บประจุจาก 0 ถึง 2/3 ของแรงดันไฟเลี้ยง (แรงดันเทรสโฮลด์)

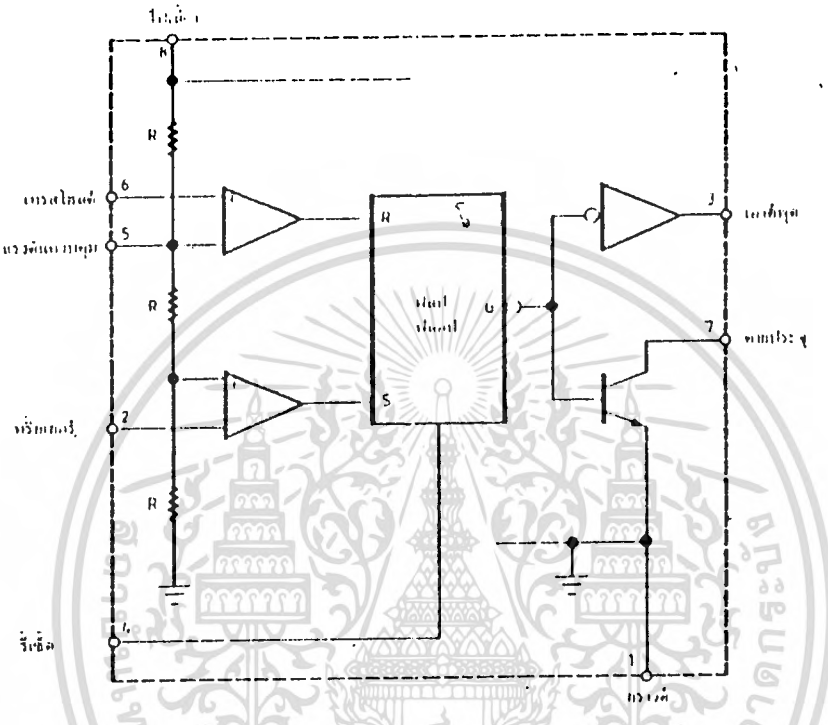
โดยทั่วไปแล้วแรงดันประจุ C (Vc) จะถูกประจุผ่าน R จากแหล่งจ่ายไฟ Vcc จนมีแรงดันเท่ากับ 2/3 Vcc

เมื่อ $V_c(t) = V_{cc}(1-e^{-T/RC})$

โดยที่ $T = (-\log e^{1/3})RC$

$= 1.1\ RC$

แรงดันที่ถูกประจุจะเป็นตัวกำหนดช่วงเวลาเดี่ยว (monotime) และพัลส์ที่มากระตุ้นก็จะสิ้นกว่าเวลาเดี่ยว ถ้าพัลส์ที่มากระตุ้นยิ่งกว้างก็จะส่งผลให้ช่วงเวลาเดี่ยวกว้างขึ้นด้วย แต่สามารถแก้ไขได้ โดยการใช้วิธีการส่งผ่านแบบสัญญาณไฟสลับ (โดยมีการเพิ่ม R2 และ C3) และให้ R2 * C3 < R1 * C1



รูปที่ 3.2 โครงสร้างพื้นฐานของไอซี 555

ในที่นี้เราต้องการใช้วงจรอะสเตเบิ้ลจึงใช้วิธีการทำให้ไอซีกระตุ้นตัวเองได้ ตัวเก็บประจุ C1 จะถูกประจุโดย R1 และ R2 จนมีแรงดันเท่ากับ 2/3 Vcc ในช่วงเวลา t1

$$t_1 = (-\log e^{1/3})(R1+R2)C - (-\log e^{2/3})(R1+R2)C$$

$$= 0.694\ (R1+R2)C$$

เมื่อถึงช่วงเวลาของการคายประจุจะเท่ากับ

$$t_2 = 0.694\ (R2*C)$$

ซึ่งก็หมายความว่าแรงดันบนตัวเก็บประจุจะเปลี่ยนไปมาระหว่าง 1/3 Vcc ถึง 2/3 Vcc ช่วงเวลาทั้งหมดคำนวณได้จาก

$$T = t_1 + t_2$$

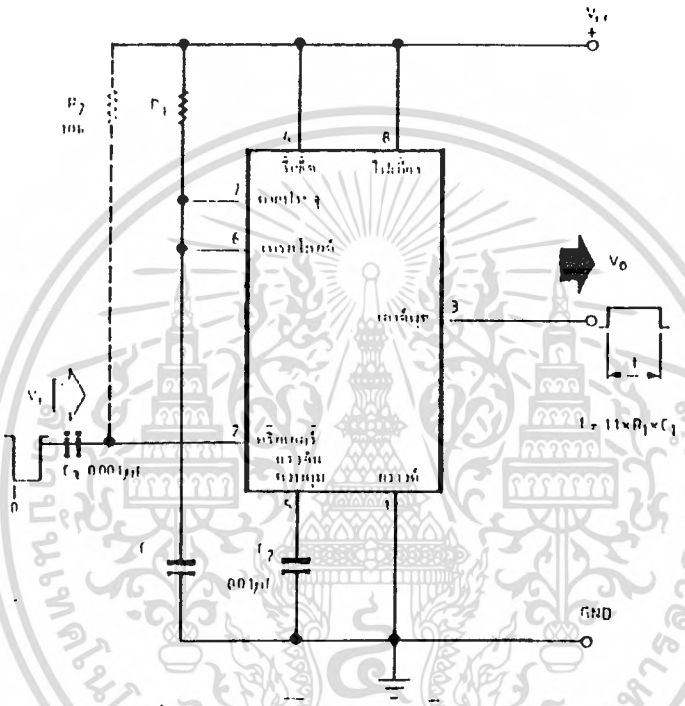
$$= 0.694\ (R_1 + 2R_2) * C_1$$

และค่าความถี่เอ้าพุท (fo) จะเท่ากับ

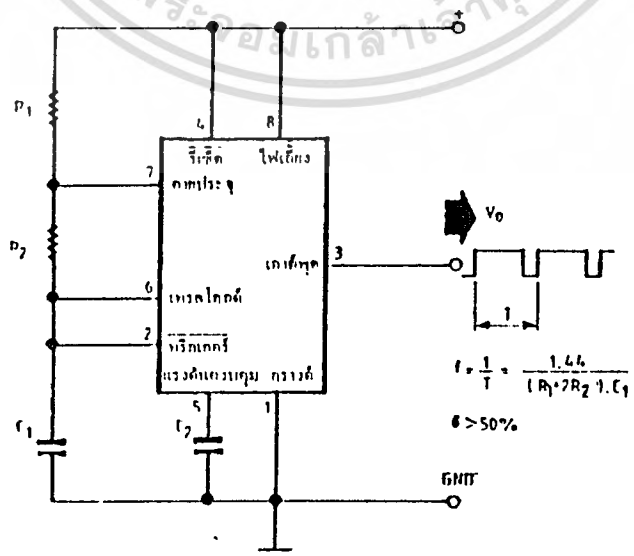
$$f_O = 1/T$$

$$= 1.44\ (R_1 + 2R_2) * C_1$$

เมื่อเราจ่ายไฟให้แก่วงจรครั้งแรก หรือเมื่อให้อินพุทที่รีเซ็ทเป็นระดับสูง C จะเริ่มประจุจาก 0 โวลต์ ดังนั้นช่วงเวลาของพัลส์ลูกแรกจะเท่ากับ 1.1 R1*C1 ดังรูป 3.3 และ 3.4



รูปที่ 3.3 วงจรโมโนสเตเบิลโดยทั่วไป

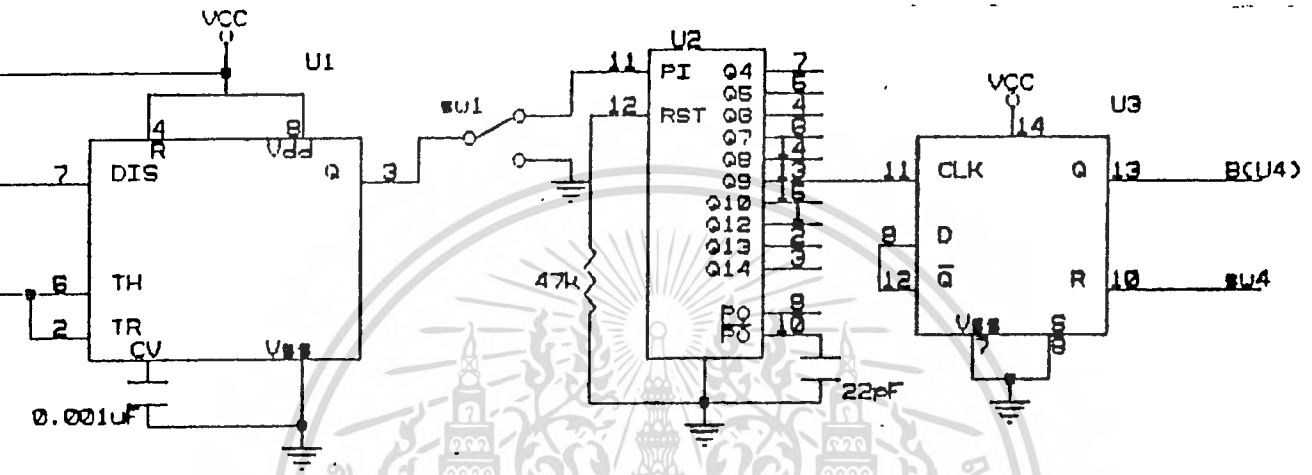


รูป 3.4 วงจรอะสเตเบิลของไอซี 555

จุดเด่นที่ใช้ไอซีเบอร์นี้เพื่อทำเป็นวงจรอะสเตเบิ้ล ก็คือ ช่วงเวลาของพัลส์จะไม่ขึ้นอยู่กับแรงดันของแหล่งจ่ายไฟเลี้ยง Vcc

3.3.2 วงจรหารความถี่

ในวงจรที่ผลิดความถี่เราต้องการนำความถี่ต่ำ ๆ มาใช้งาน เพื่อให้การนับของชุดวงจรนับทำงานตามสัญญาณพัลส์ที่ผลิดออกมา ในชุดของวงจรผลิดความถี่ดังรูป 3.5



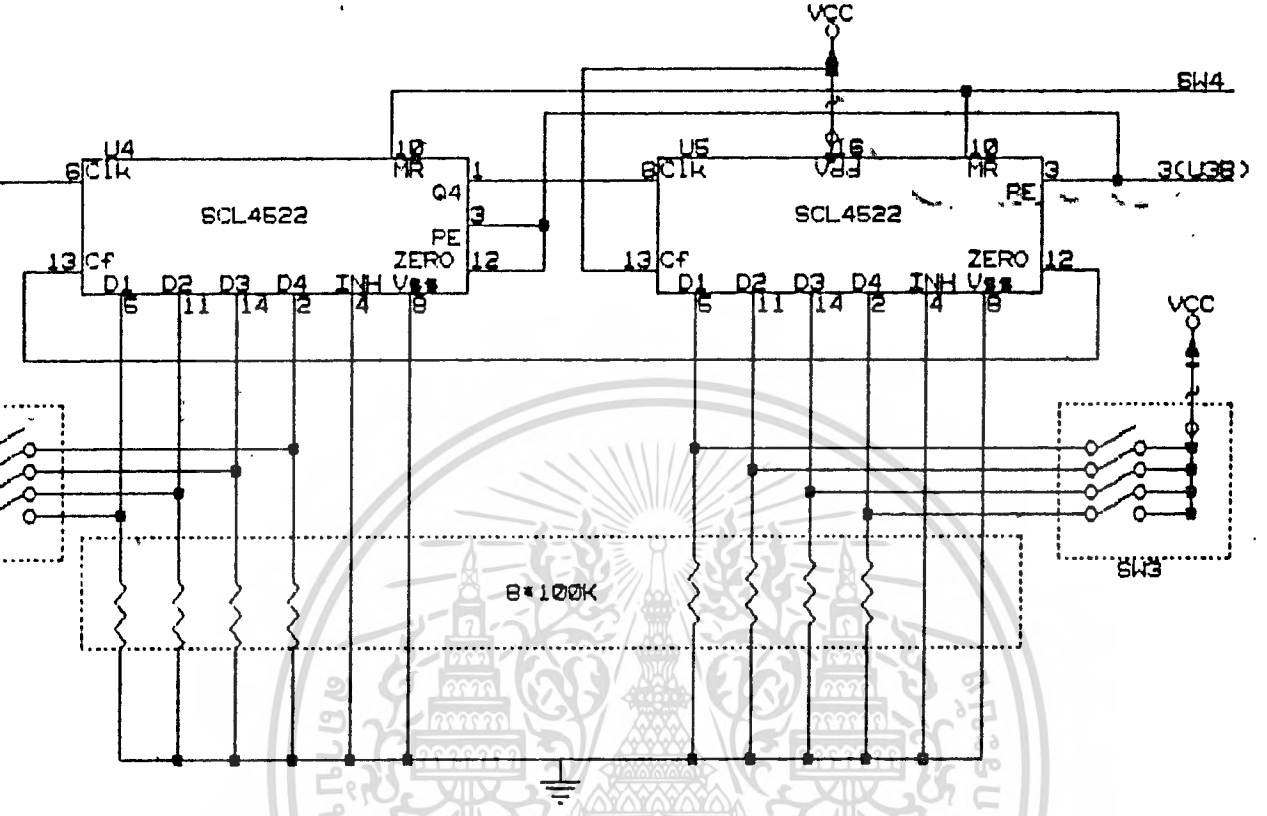
รูป 3.5 วงจรหารความถี่

เราต้องการความถี่เอาต์พุทที่ 1Hz เพื่อนำไปเป็นสัญญาณนาฬิกาให้แก่วงจรนับไอซี 555 นี้จะผลิดความถี่ออกมาที่ 1024Hz จากการที่เราใช้ความถี่ประมาณ 1K นี้ เนื่องจากเราต้องการให้มีความเที่ยงตรงมากหน่อย เพราะ 555 จะทำงานได้ดีที่ความถี่สูงปานกลาง จากนั้น เราจะใช้ไอซี 4060 เป็นตัวหารความถี่นี้ให้เหลือ 1Hz เพราะ 4060 เป็นวงจรหารด้วยเลขหาร 2 และยังสามารถผลิดความถี่ได้ด้วย

จากรูป 3.5 มีการสร้างความถี่จากไอซี 555 ที่ 1024Hz ผ่านไอซี 4060 ซึ่งเป็นวงจรหารความถี่ที่ 2 ได้ความถี่เอาท์พุท 2Hz แล้วหารด้วยวงจรหาร 2 อีกที (ไอซี 4013) ได้ความถี่ที่จะนำไปป้อนเป็นสัญญาณนาฬิกาให้แก่วงจรนับ โดยเอาท์พุทออกที่ขา 13 ของ 4013

3.3.3 ส่วนวงจรนับ (ไอซี 4522) จากรูป 3.6 ใช้ไอซี 4522 2 ตัวต่อกันในลักษณะของวงจรริปเปิ้ลเคาน์เตอร์ธรรมดาคือ ใช้ Qc ของชุดแรกต่อเข้ากับขาคล็อกของชุด

ที่ 2 และใช้ขา Zero สำหรับหารด้วย n เมื่อต่อใช้งานหลาย ๆ ชุด โดยจะต่อเข้ากับขาคาสเคดนีดแบ็ก (CF)



รูปที่ 3.6 การต่อขาของวงจรนับลง

ขา Zero ปกติจะมีสถานะเป็น "0" ตลอดเวลาของการนับและจะมีสถานะเป็น "1" เมื่อวงจรนับลงจนเป็น "0"

เมื่อขาอีนาเบิ้ลเป็น "1" มันจะโหลดค่าที่ต้องการจะนับ ซึ่งกำหนดโดยสวิตช์รหัส BCD สวิตช์มาเก็บไว้ และขา Zero ของวงจรนับชุดที่ 2 จะต่อกับขา CF ของวงจรนับชุดแรก

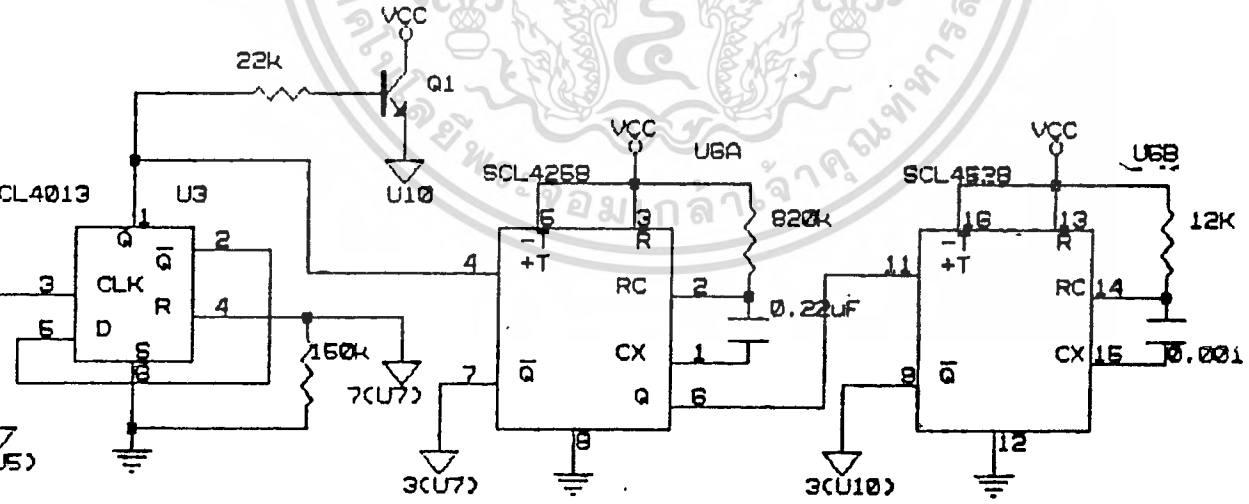
เมื่อวงจรนับชุดแรกเป็น "1" และชุดที่ 2 เป็น "0" ซึ่งมี CF เป็น "1" (ต่อกับ Vdd) จะทำให้เอาต์พุทที่ขา Zero เป็น "1" สัญญาณ "1" นี้ใช้เป็นอินพุทให้กับ CF ของชุดแรก ถ้าขณะนี้มีสัญญาณนาฬิกาเกิดขึ้น 1 ลูกวงจรนับชุดแรกก็จะกลายเป็น "0" และเนื่องจากขา CF เป็น "1" ดังนั้นขา Zero จึงเปลี่ยนสถานะจาก "0" ไปเป็น "1" และทำการ

ตั้งค่า (preset) ที่จะนับทั้งหมดแล้วโหลดค่าที่จะนับจากสวิตช์รหัส BCD เข้ามาใหม่ (เช่น สมมติเป็น 15) เพื่อกำหนดจำนวนที่จะนับลงและเพราะที่วงจรนับชุดแรกเป็น "0" อยู่ไม่นาน (เนื่องจากรับค่าเข้ามาใหม่) ดังนั้น ขา Zero ก็จะกลับมาเป็น "0" อีกครั้ง

ทางด้านการทำงานของสัญญาณนาฬิกาจะทำการลดค่าของวงจรนับชุดแรก จาก 5 จนกระทั่งเป็น 0 แต่คราวนี้ CF ของวงจรนับชุดแรกจะมีสถานะเป็น "0" เพราะวงจรนับชุดที่สองมีค่าเป็น "1" (และที่ขา Zero เป็น "0") ดังนั้น ขา Zero ของวงจรนับชุดแรกก็จะมีสถานะเป็น "0" และจะไม่ทำการตั้งค่าที่จะนับในรอบนี้ ดังนั้นสัญญาณนาฬิกาก็จะทำการลดค่าของวงจรนับชุดแรกจาก 9 จนถึง 0 และจะมีสัญญาณริปเปิ้ลไปยังวงจรนับชุดที่สอง ทำให้วงจรนับมีค่าเป็น "0" หลังจาก 9 ลูกคลื่นผ่านไป วงจรนับชุดแรกก็จะมาเป็น "0" อีกครั้ง และขณะนี้วงจรนับทั้งหมดมีค่าเป็น "0" จะทำให้วงจรนับทำการตั้งค่าแล้วโหลดค่าจากสวิตช์รหัส BCD มาใหม่อีกครั้ง จึงเห็นได้ว่า การใช้วงจรนับสองชุดสามารถกำหนดการหารความถี่ได้ตั้งแต่ 1 พัลส์/วินาที ถึง 1 พัลส์/99 วินาที โดยการตั้งสวิตช์รหัส BCD

เอาต์พุตของ 4522 เป็นสัญญาณที่ใช้กระตุ้นอุปกรณ์อื่น ๆ เพื่อทำให้เกิดการเก็บข้อมูลโดยป้อนที่ขาคล็อก (ขา 3) ของ IC 4013 D ฟลิปฟลอป ซึ่งทำงานที่ขอบขาขึ้นของสัญญาณนาฬิกา

3.3.4 ส่วนของโมโนสเตเบิล (4528)



รูปที่ 3.7 ชุดไอซีโมโนสเตเบิล

เป็นมัลติไวเบรเตอร์อีกแบบหนึ่งที่นิยมใช้กันอย่างแพร่หลาย แบบนี้จะไม่กำเนิดพัลส์ต่อเนื่อง แต่จะให้พัลส์ออกมาลูกเดียวเมื่อมีสัญญาณมากระตุ้น ข้อสำคัญคือช่วงเวลาของพัลส์สามารถกำหนดให้สั้นยาวได้ โดยการเลือกใช้ค่า R และ C วงจรชนิดนี้ใช้ในวงจรควบคุมดิจิตอลมากเพราะสามารถใช้เป็นไทเมอร์ (timer) หรือตัวตั้งเวลา เมื่อมีคำสั่งให้ทำงานจะส่งเอาท์พุตออกไปควบคุมให้เครื่องจักรเครื่องใดเครื่องหนึ่งทำงานในช่วงเวลาที่กำหนดแล้วหยุดทำงานเองโดยอัตโนมัติ

การทำงานของไอซีชุดนี้ แสดงในรูป 3.7 ในสภาพปกติ ขณะที่ไม่มีอินพุทเลยนั้น เอาต์พุท Q="0" และ $\overline{Q}$="1" อยู่ตลอดเวลา ต่อเมื่อมีสัญญาณเข้ามาทางอินพุทขา 4 หรือขา 11 จะทำให้ Q เป็น "1" (และ $\overline{Q}$= "1") นานช่วงเวลาหนึ่ง ในที่นี้ประมาณ 2 ไมโครวินาที จากนั้นจึงตกกลับเป็น "0" ตามเดิม ช่วงเวลานี้กำหนดโดย R และ C ที่ต่ออยู่ที่ขา1 และ ขา 2

จากรูป 3.7 เอาต์พุทที่ขา 1 จาก 4013 ให้พัลส์ 2 ไมโครวินาที ทริกที่ขอบขาขึ้นเข้าขา 4 ของ 4528 ทำให้ได้เอาต์พุทที่ขา 6 เป็นการกลับสัญญาณพัลส์ของอินพุท เพื่อนำเอาต์พุทที่ได้ไปป้อนให้กับ ไอซี 4528 อีกตัว โดยจะเป็นตัวใช้สัญญาณพัลส์ไปควบคุมไอซี ADC 0804 ซึ่งเป็นตัวแปลงสัญญาณจากอนาลอกเป็นดิจิตอลอีกที และยังทำการควบคุมไอซี 40132 (U 3/B) ด้วย

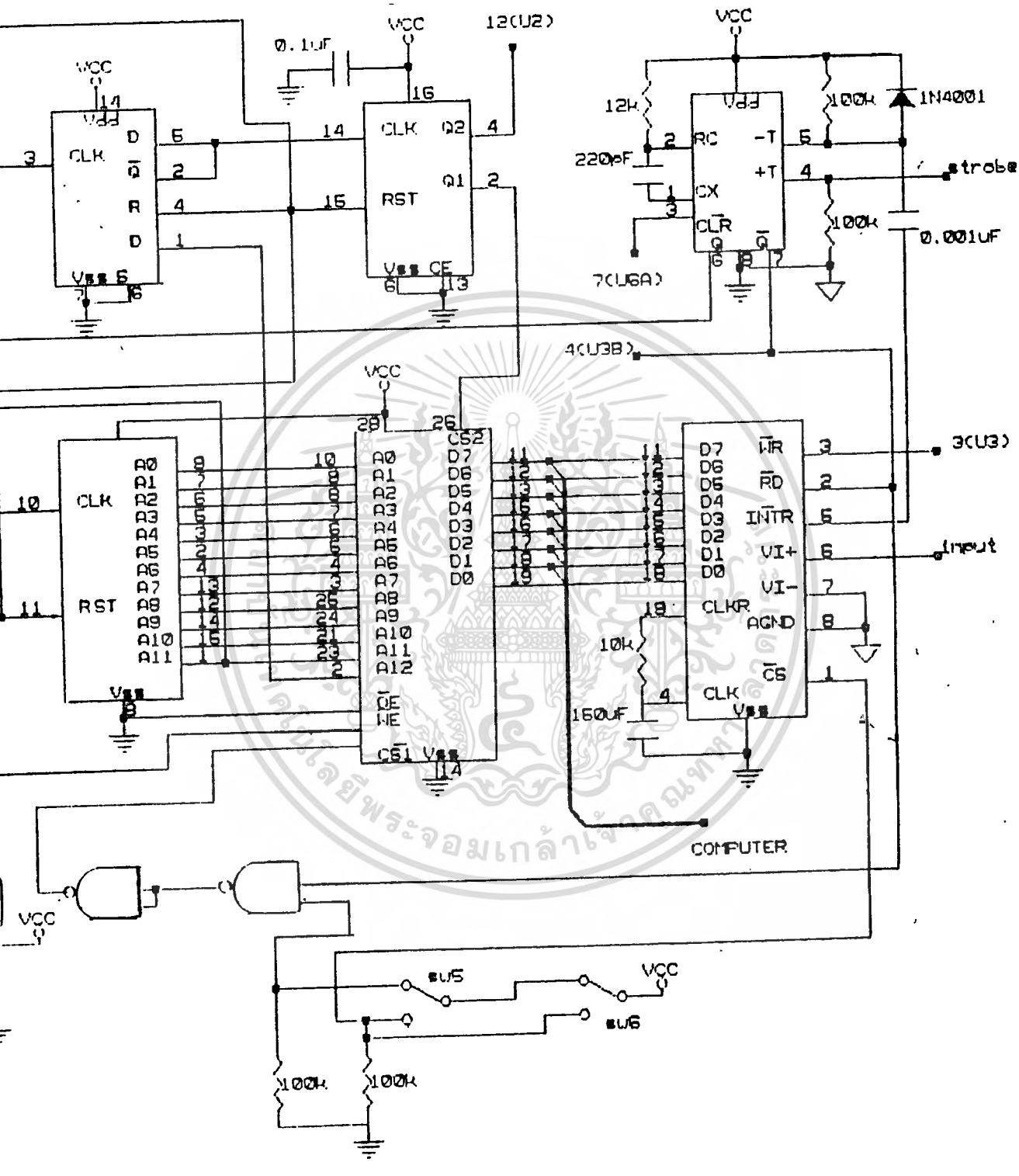
ส่วนทรานซิสเตอร์จะเป็นตัวควบคุมการทำงานของ ADC 0804 ให้เป็นไปตามช่วงเวลาที่กำหนด

3.3.5 ส่วนแปลงสัญญาณอนาลอกเป็นดิจิตอล

ส่วนนี้จะทำหน้าที่แปลงสัญญาณอนาลอกเป็นดิจิตอล อุปกรณ์ที่ใช้ในการแปลงสัญญาณคือ ไอซีซึ่งเป็นชิพขนาด 8 บิท โดยในโครงงานนี้ใช้ไอซีเบอร์ ADC 0804

รูป 3.8 ไอซี ADC 0804 เป็นตัวรับสัญญาณอนาลอกจากภายนอกเข้ามา ใช้ C และ R เป็นตัวกำหนดค่าเวลาความถี่ของวงจรผลิตความถี่

การแปลงสัญญาณแบบประมาณค่าอย่างต่อเนื่องนั้นมีการทำงานดังนี้คือ ให้ขา $\overline{\text{STRT}}$ เป็นลอจิก "0" เข้ารีจิสเตอร์เก็บค่าประมาณอย่างต่อเนื่อง (SAR) จะได้ขาเอาต์พุท D7 (MSB) เป็น "1" นอกนั้นเป็น "0" หมด ดังนั้นภาคแปลงสัญญาณดิจิตอลเป็นอนาลอก (D/A) ก็จะแปลงสัญญาณดิจิตอลที่ได้ไปเปรียบเทียบกับแรงดันอินพุทที่ส่งเข้ามา ถ้า D/A สูงกว่าแรงดันอินพุท SAR จะเปลี่ยน D7 ให้เป็น "0" แต่ถ้าเอาต์พุทจาก D/A ต่ำกว่าแรงดันอินพุท SAR



รูปที่ 3.8 ชุดหน่วยความจำและแปลงสัญญาณ

ก็จะให้ D7 เป็น "1" ต่อไป

ต่อมาก็จะให้บิตถัดไป (D6) เป็น "1" แล้วแปลงสัญญาณดิจิตอลที่ได้เป็นอนาลอกไปเปรียบเทียบกับอินพุทอีกว่า สูงกว่าแรงดันอินพุทหรือเปล่า ถ้าสูงกว่าก็ให้ D6 เป็น "0" ถ้าต่ำกว่าก็ให้เป็น "1" การทำงานจะซ้ำไปอย่างนี้จนกระทั่งครบทั้ง 8 บิต แล้ว SAR ก็จะให้ขา DR เป็น "0" เพื่อแสดงผลของข้อมูลออกมาจากเอาต์พุทรีจิสเตอร์เป็นสัญญาณดิจิตอล 8 บิททางขา Q "0" ถึง Q "7"

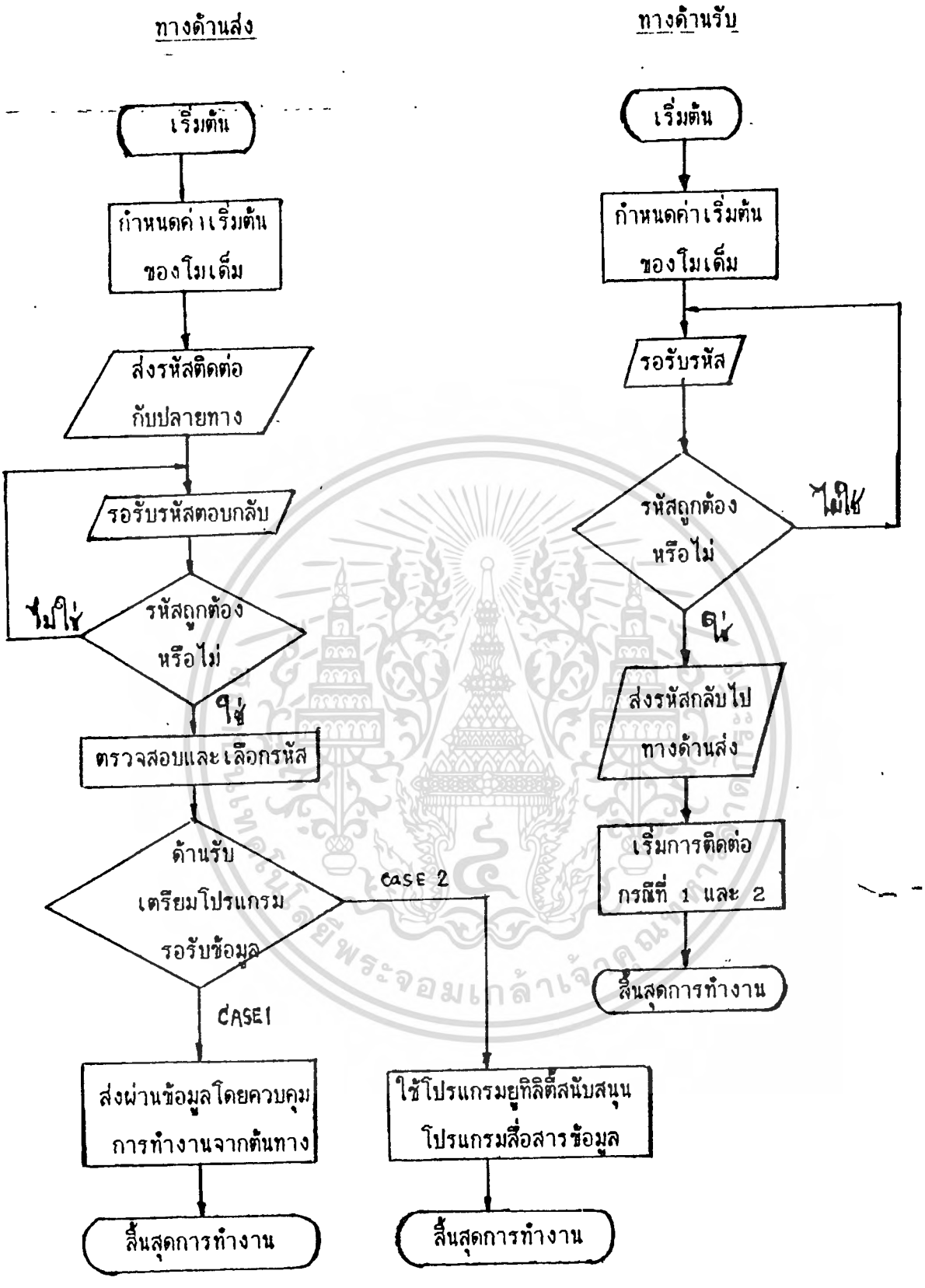
เมื่อเริ่มต้นการแปลงสัญญาณ จำเป็นจะต้องมีสัญญาณที่ขา CS และ WR เป็น "0" ซึ่ง CS จะถูกรักษาไว้ที่ "0" ด้วยตัวต้านทาน R14 และเมื่อ WR ถูกทำให้เป็น "0" นาน 10 ไมโครวินาที โดยพัลส์ "0" จากโมโนสเตเบิ้ล IC 4528 ขบวนการแปลงสัญญาณก็จะเกิดขึ้น และเมื่อแปลงสัญญาณเสร็จ A/D จะทำให้ขา INT เปลี่ยนจากลอจิก "1" มาเป็น "0" จะทำให้เกิดการกระตุ้นโมโนสเตเบิ้ลอีกตัวขึ้น ซึ่งโมโนสเตเบิ้ลตัวนี้จะถูกกำหนดคาบเวลาโดย C6 และ R7 สัญญาณพัลส์จากขา Q จะนำไปขับขา RD ของ A/D ให้เป็น "0" เพื่อให้ A/D ส่งข้อมูลออกไปยังบัสข้อมูล

ที่ช่วงเวลาในการเกิดสัญญาณดังที่กล่าวมาแล้วที่ขา Q ของ IC13 จะเป็น "1" เป็นเวลา 2 ไมโครวินาที และสัญญาณนี้จะนำไปขับขา 6 ของ IC5/2 ให้เป็น "1" ซึ่งขาอื่นของเกตตัวนี้จะถูกรักษาไว้ที่ "1" ผ่าน R15 แต่ที่เอาต์พุทของเกตที่ขา 4 จะเป็น "0" เป็นเวลา 2 ไมโครวินาที ดังนั้น จะทำให้เกิด write พัลส์ที่ขา WE ของ RAM ซึ่งในช่วงเวลานี้ข้อมูลของ A/D จะถูกเก็บเข้า RAM

ที่ขา Q (ขา6) ของโมโนสเตเบิ้ล IC13 จะต่ออยู่กับขา CLK (ขา 10) ของ IC4 เบอร์ 4040 ซึ่งทำหน้าที่เป็นตัวนับแอดเดรส และเพราะตัวนับจะเพิ่มค่าขึ้นในขอบขาขึ้นของ write พัลส์ ดังนั้นจึงเป็นการกำหนดแอดเดรสของ RAM ในตำแหน่งถัดไป และสัญญาณที่ขา Q นี้จะต่อเข้ากับขารีเซ็ต (ขา 4) ของ IC4013 (U3B) ด้วยโดยผ่าน C3 ดังนั้นที่ขอบขาขึ้นของสัญญาณพัลส์จะเกิดการรีเซ็ตเป็นเวลา 2 ไมโครวินาที ทำให้ขา Q ของ IC U3B เป็น "0" และหยุดจ่ายไฟให้ A/D

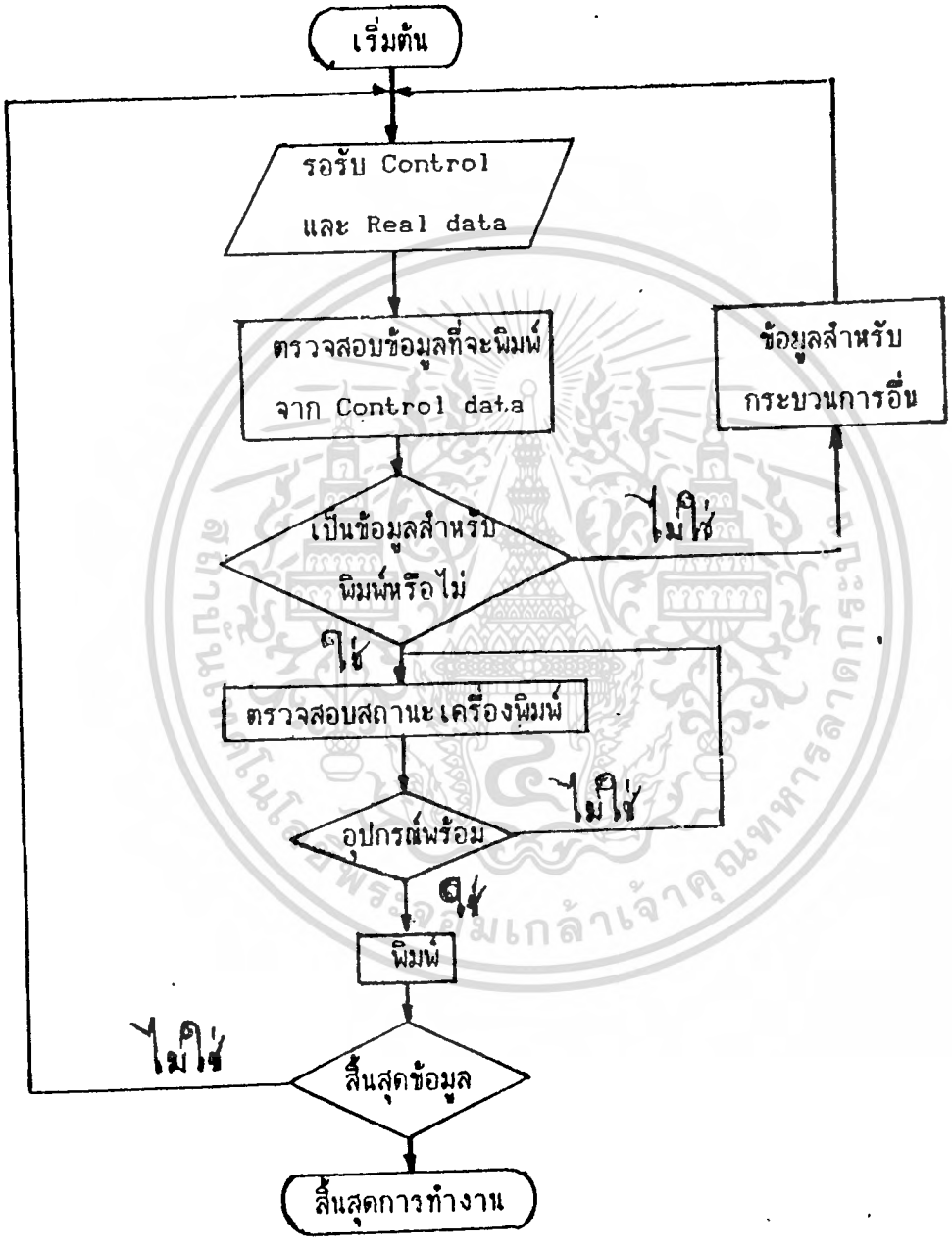
3. 4 โครงสร้างของโปรแกรมสำหรับการสื่อสารข้อมูล

ดูรูปที่ 3.9



รูปที่ 3.9 โฟล์วชาร์ตของโปรแกรมสำหรับการสื่อสารข้อมูล

ทางด้านรับ เมื่อผู้ใช้ได้ออนเครื่องทิ้งไว้เพื่อรอรับข้อมูล ทางด้านส่งจำเป็นต้องส่งข้อมูลในรูปแบบที่ใช้สำหรับ ควบคุมอุปกรณ์ภายนอกเพิ่มขึ้นจากข้อมูลที่ต้องการติดต่อจริง ๆ เช่นการติดต่อกับเครื่องพิมพ์ จอภาพ ซึ่งมีโครงสร้างดังรูปที่ 3.10



รูปที่ 3.10a โครงสร้างการติดต่อกับอุปกรณ์ภายนอก

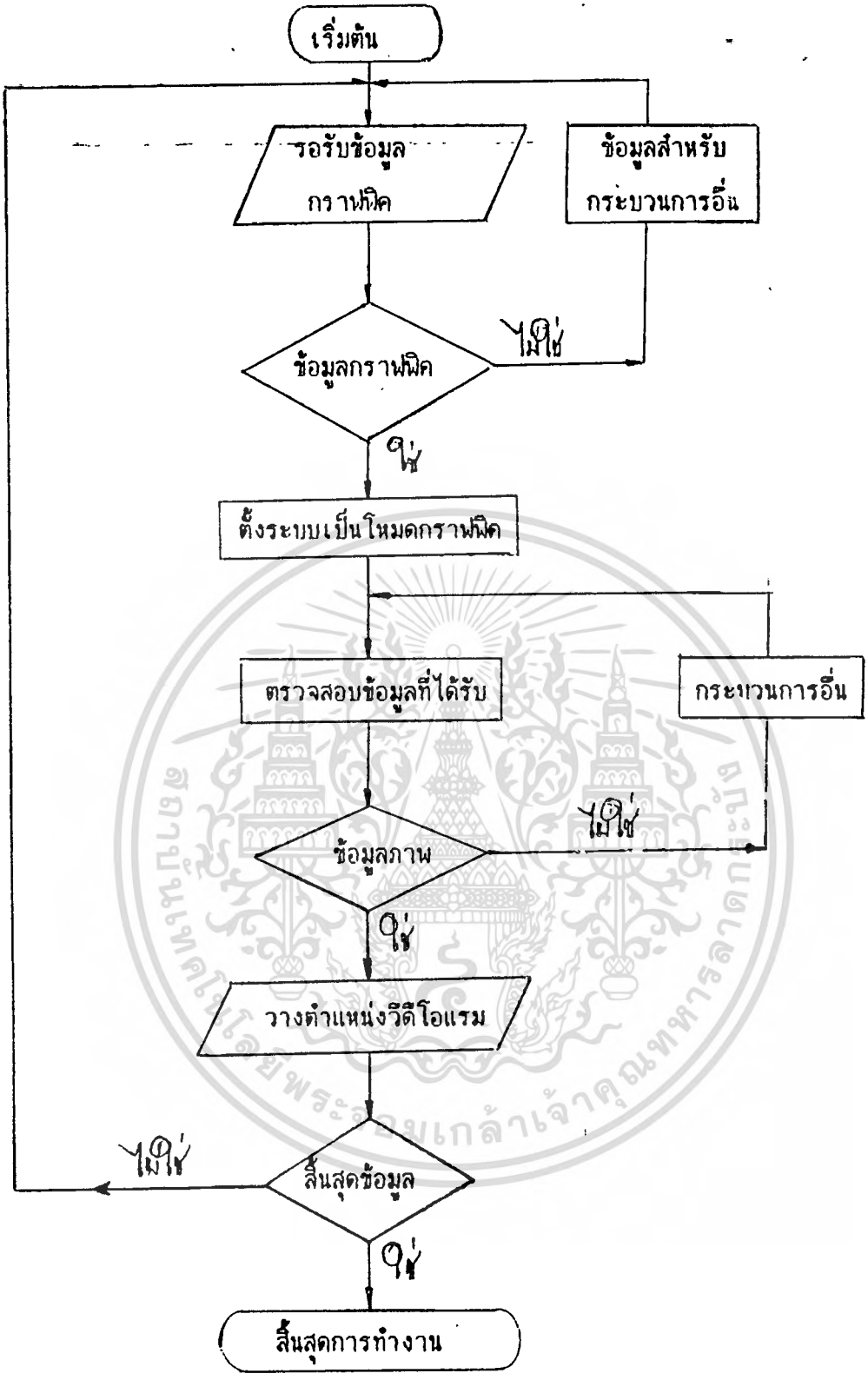
จากรูปที่ 3.10a control device data นี้เป็นข้อมูล 1 ไบท์ ซึ่งจะส่งเป็นส่วนหัวของทุก ๆ บล๊อกข้อมูล เพื่อให้ทางด้านรับทราบว่า ข้อมูลนี้ต้องการให้พิมพ์ออกทางเครื่องพิมพ์หรือนำไปใช้ใน process อื่น

ในกรณีที่มีการติดต่อกับจอภาพ (monitor) ของ PC นั้น ถ้าไมโครคอมพิวเตอร์มีจอภาพชนิดเดียวกันทั้งทางด้านรับและส่ง ข้อมูลที่จะทำการส่งจะได้จากการอ่านข้อมูลจากตำแหน่งของ Video Ram แต่ละตำแหน่งที่สอดคล้องกับตำแหน่งจุดแต่ละจุดบนจอ (pixel) ดังนั้น จำเป็นจะต้องมีข้อมูลที่ใช้เป็นข่าวสารกำกับที่ส่วนหัวและส่วนท้ายของ block ข้อมูล ดังรูป 3.10b

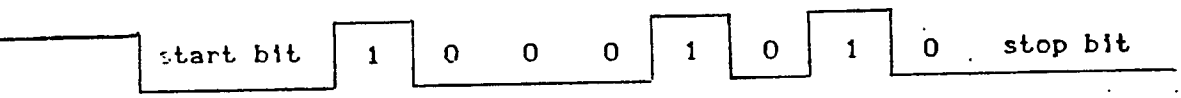
จากการส่งข้อมูลในรูปแบบต่าง ๆ ที่กล่าวมา สิ่งที่หลีกเลี่ยงไม่ได้คือ ความผิดพลาด (error) ที่จะเกิดขึ้นซึ่ง 8250 มีคุณสมบัติในการตรวจสอบความผิดพลาดได้ โดยใช้การตรวจสอบ bit parity แต่ถ้าข้อมูลที่การตรวจสอบ bit parity ไม่สามารถตรวจจับได้ดังรูปที่ 3.11 ซึ่งจะมีเปอร์เซ็นต์ในการเกิดน้อยในระหว่างการส่งสายข้อมูลที่ติดต่อกัน (จากการทดลองส่งพบว่าจะเกิด 1 ตัวหลังจากการส่งข้อมูลติดต่อกันไปแล้ว 36330 ตัว) และเปอร์เซ็นต์ของการเกิดจะเพิ่มมากขึ้นเมื่อหยุดการรับส่งสายข้อมูล แต่ยังมีการเชื่อมต่อกันโดยใช้เพียงคลื่นพาหะ (carrier)

ในการลดความผิดพลาดที่เกิดขึ้นนี้สามารถทำได้ 2 แบบคือ ปรับปรุงทางฮาร์ดแวร์และซอร์ฟแวร์ให้มีการตรวจสอบบิทแต่ละบิทในรูปแบบอื่นเพิ่มขึ้น เช่น การตรวจสอบในแนวดิ่ง (vertical redundancy check) การตรวจสอบแบบเศษส่วนข้อมูล (cyclic redundancy check) เป็นต้น

อีกวิธีหนึ่งคือ ตรวจสอบในระดับไบท์ข้อมูล ซึ่งบิทแต่ละบิทในหนึ่งไบท์จะให้ข่าวสารที่แตกต่างกันเพื่อบอกให้ทางด้านรับทราบว่า ข้อมูลที่ตามมาเป็นข้อมูลที่แท้จริงหรือไม่ การตรวจสอบแบบนี้จะมีโครงสร้างดังรูปที่ 3.12

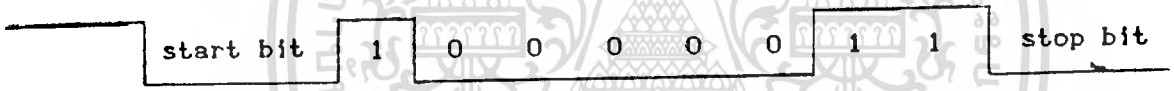


รูปที่ 3.10b โฟล์วชาร์ตแสดงการติดต่อกับอุปกรณ์ภายนอก



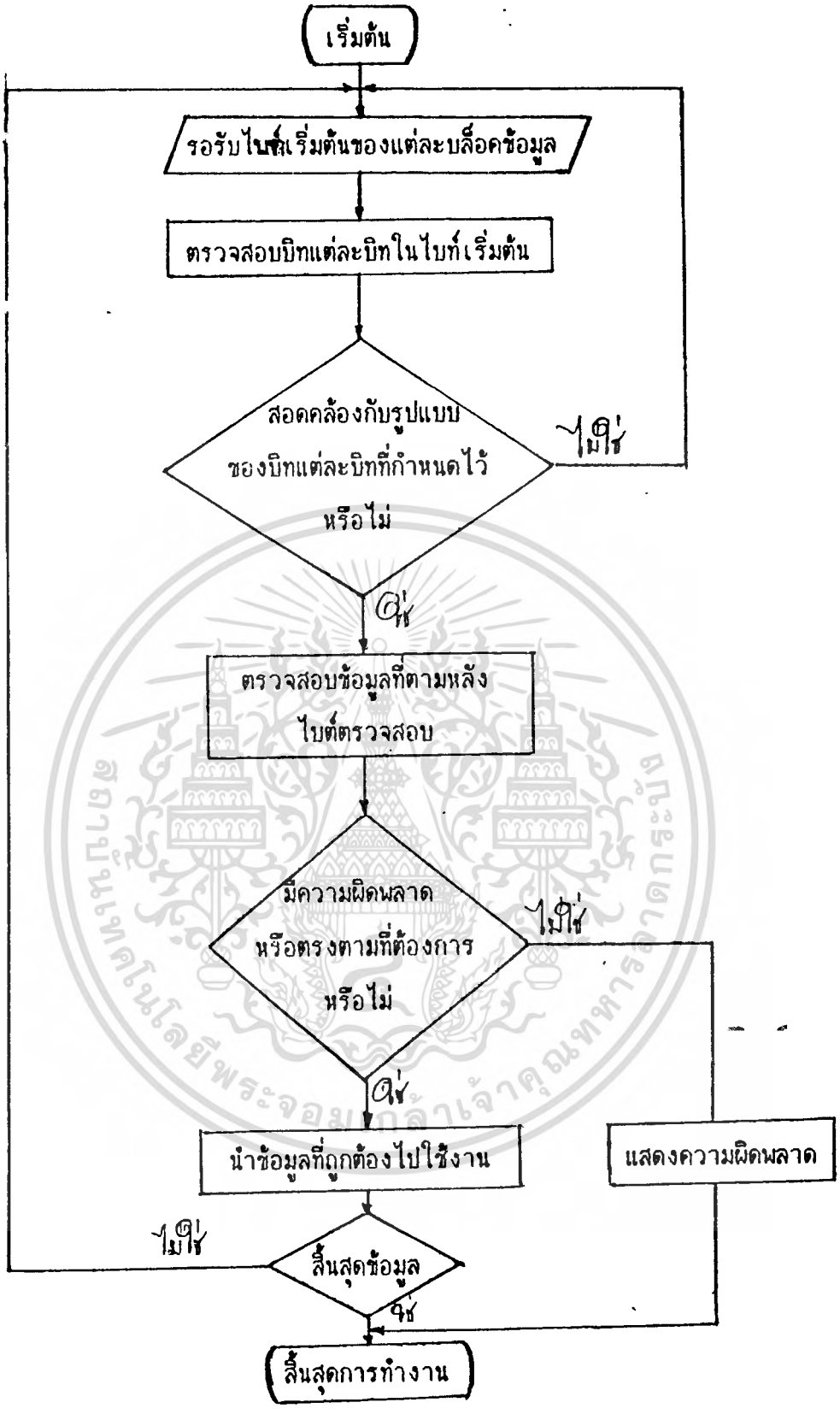
a) ข้อมูลที่ถูกต้องอักขระ "E" (รหัสแอสกี้ 8A และ odd parity)

b) บิทข้อมูลผิดพลาด 2 บิท



c) บิทข้อมูลผิดพลาด 1 บิท และบิทพาริตี้ผิดพลาด

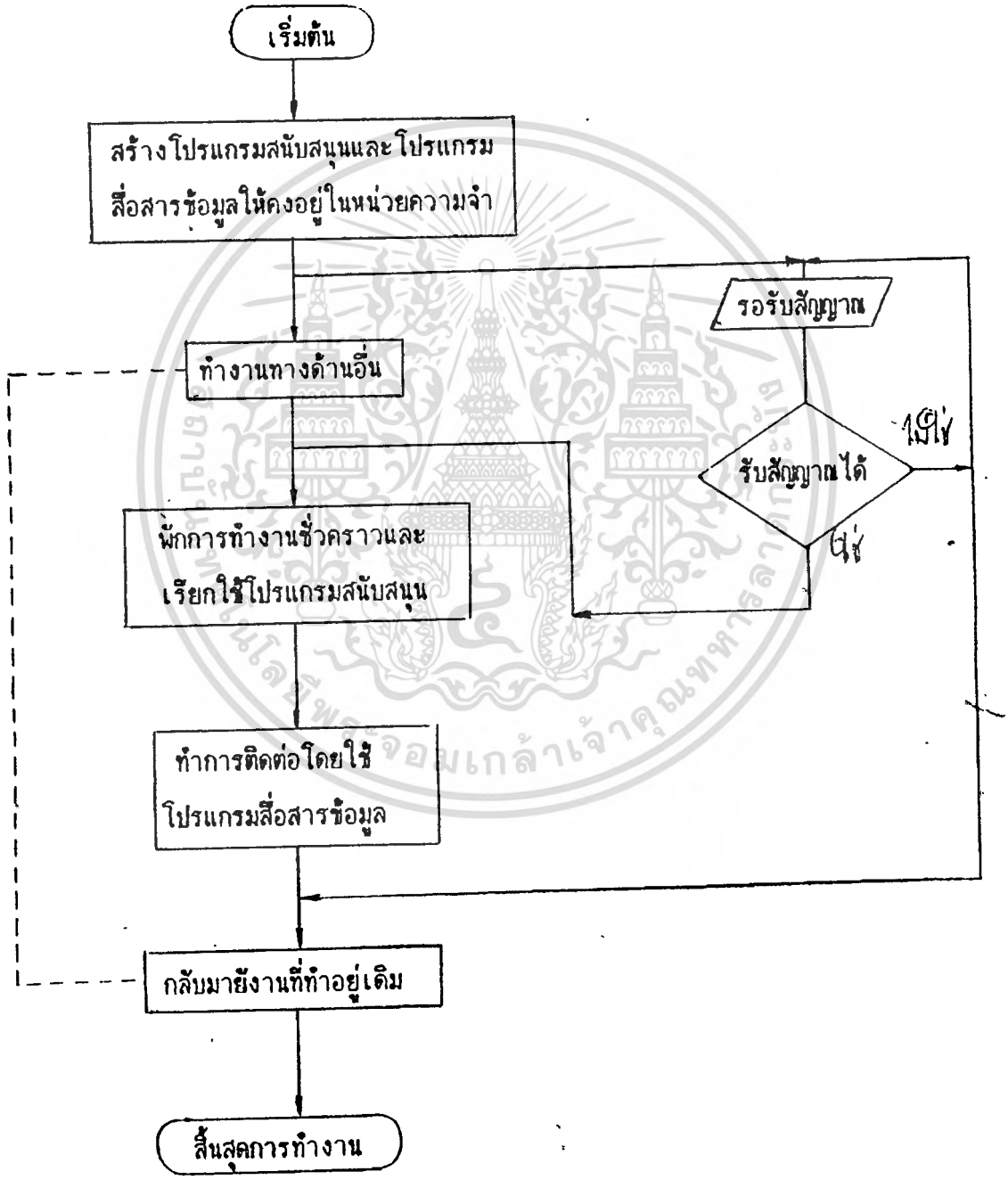
รูปที่ 3.11 ตัวอย่างข้อมูลที่ผิดพลาด



รูปที่ 3.12 การลดค่าความผิดพลาด

3.4.1 การสร้างส่วนสนับสนุนโปรแกรมการสื่อสารข้อมูล

ในการใช้อุปกรณ์ไมโครคอมพิวเตอร์สำหรับการสื่อสารข้อมูลนั้น วิธีหนึ่งที่จะช่วยให้อุปกรณ์มีประสิทธิภาพที่สูงขึ้นคือ ใช้โปรแกรมช่วยงานทางด้านอื่นมาสนับสนุน เพื่อให้ผู้ใช้สามารถทำงานหลายด้านในเวลาเดียวกัน โดยไม่รบกวนกันและกัน ในโครงงานนี้ได้ทดลองสร้างโปรแกรมสนับสนุนการส่งผ่านข้อมูลโดยใช้การอินเตอร์รัพจาก คีย์บอร์ด (pop-up) มีหลักการทำงานคร่าว ๆ ดังรูป



การสร้างโปรแกรมสนับสนุนการทำงานในลักษณะนี้ คุณสมบัติที่สำคัญอย่างหนึ่งคือ เมื่อเรียกใช้โปรแกรมสื่อสารข้อมูล โดยผ่านโปรแกรมสนับสนุนนี้ ข้อมูลของโปรแกรมเดิมกับโปรแกรมใหม่ จะต้องไม่รบกวนกัน ไม่ทำให้การทำงานของดอสผิดพลาดและไม่รบกวนการทำงานของโปรแกรมเดิม ซึ่งเรียกการทำงานในลักษณะดังกล่าวว่า มัลติทาสค์กิ้ง (Multi Tasking)

มัลติทาสกิง คือ ผู้ใช้สามารถเรียกฟังก์ชันของโปรแกรมสนับสนุน (เช่น โปรแกรมไซด์คิก หรือพีซีทูล) ขณะรันโปรแกรมอื่นอยู่ แต่ดอสยังมีข้อจำกัดในส่วนนี้อยู่คือ ไม่สามารถถูกอินเตอร์รัพขณะทำงานแบบ critical อยู่ เช่น การเขียนข้อมูลลงดิสค์ ดังนั้นโปรแกรมสนับสนุนที่ดีควรรู้ว่าเมื่อใดที่สามารถจะทำการอินเตอร์รัพดอสได้ โดยจะตรวจสอบได้จากแฟลกที่ใช้ตรวจเช็ค Critical error กับ Control break (แฟลก INDOS)

แฟลก INDOS นี้เป็นแฟลกขนาด 1 เวิร์ดที่ไม่เปิดเผย โดยที่แฟลกนี้จะถูกเพิ่มค่าขึ้น เมื่อดอสนี้ทำงานที่เป็น critical ดังนั้นการขออินเตอร์รันดอสบางครั้งให้มีความปลอดภัยจะต้องมีการตรวจสอบแฟลกนี้ว่าเป็น 0 หรือไม่ ซึ่งตำแหน่งของแฟลกตัวนี้สามารถขอดูได้ผ่านทางอินเตอร์รันดอสฟังก์ชัน 34h

Function 34h : Get INDOS Flag Address

Interrupt : 21h

Function : 34H

โดยให้ค่ารีจิสเตอร์ทาง AX = 34h

และส่งค่ากลับทาง ES : Segment INDOS Flag

BX : Offset INDOS Flag

โดยการใช้ ES : BX ชี้ตำแหน่งของ INDOS Flag ทำให้สามารถทราบได้ว่าจะปลอดภัยสำหรับการอินเตอร์รันหรือไม่ ซึ่งไม่เพียงแต่จะมีประโยชน์เฉพาะโปรแกรมสนับสนุนเท่านั้น แต่ยังเป็นประโยชน์สำหรับโปรแกรมประเภทมัลติทาสกิงอีกด้วย

แต่มีสิ่งที่แปลกอยู่อีกอย่างของแฟลก INDOS คือเมื่อดอสรอรับคำสั่งอยู่ที่ดอส prompt เฉย ๆ แฟลกนี้จะถูกเซ็ตเป็น 1 ทั้ง ๆ ที่จริง ๆ แล้ว ขณะนั้นดอสไม่ได้ทำงานอะไรที่สำคัญเลย เพราะฉะนั้นจึงต้องมีการตรวจสอบในกรณีนี้อีกอย่าง คือ การตรวจสอบอินเตอร์รัพ 28h (Dos Idle) ซึ่งเราสามารถนำคุณสมบัติดังกล่าวมาสร้างโปรแกรมที่ต้องตรวจสอบการทำงานของคีย์บอร์ด มีการตรวจสอบอินเตอร์รันคีย์บอร์ด (Int 9h) พร้อม ๆ กับตรวจสอบ

อินเตอร์รัพ 28h ไปด้วย (แสดงว่าดอสไม่ได้ทำอะไรที่สำคัญอยู่) จะเลิกสนใจแฟลก INDOS ทันทีและรอคีย์ที่ต้องการถูกกดเท่านั้น

ในขณะที่โปรแกรมสนับสนุนเริ่มทำงานจะต้องมีการเปลี่ยนข้อมูลบางอย่าง เพื่อไม่ให้โปรแกรมสนับสนุนไปใช้พารามิเตอร์ของโปรแกรมที่ทำงานอยู่ก่อนหน้าที่โปรแกรมสนับสนุนจะแอคทีฟ ซึ่งพารามิเตอร์ที่สำคัญตัวนี้คือตัวโปรแกรมเซกเมนต์พรีฟิกซ์ (Program Segment Prefix หรือ PSP) โดยที่ดอสจะถือว่าโปรแกรมที่ถูกโหลดลงมาหลังสุดจะเป็นปัจจุบัน (Current Program) ซึ่งดอสก็จะใช้ข้อมูลของโปรแกรมนี้เป็นหลัก เช่น ในการเปิดไฟล์ก็จะใช้ตารางแฮนเดิลของ PSP ของโปรแกรมนั้น และเมื่อโปรแกรมสนับสนุนเริ่มทำงานจึงจำเป็นต้องเก็บ PSP เก่า (ของโปรแกรมที่กำลังรันก่อนหน้าโปรแกรมสนับสนุนจะรัน) ไว้ก่อน จากนั้นจึงบอกดอสให้มาใช้ PSP ของโปรแกรมสนับสนุน ไม่เช่นนั้น PSP เดิมอาจเสียหายได้โดยที่มีฟังก์ชันของดอสที่ช่วยในการทำงานด้านนี้อยู่ คือ

Function 50h : Set Current PSP

Interrupt : 21h

Function : 50h

โดยให้ค่ารีจิสเตอร์ดังนี้

AX = 50h

BX = Segment of PSP ที่ต้องการให้เป็น PSP ปัจจุบันและไม่มีค่าส่งกลับ

Function 51h ,62h : Get PSP Segment

Interrupt : 21h

Function : 51h (ไม่เปิดเผยในดอสเวอร์ชัน 2.x เป็นต้นไป)

Function : 62h (เปิดเผยในเวอร์ชัน 3.x เป็นต้นไป)

โดยให้ค่ารีจิสเตอร์ดังนี้

AX = 51h ,62h

และมีค่าที่ส่งกลับคือ

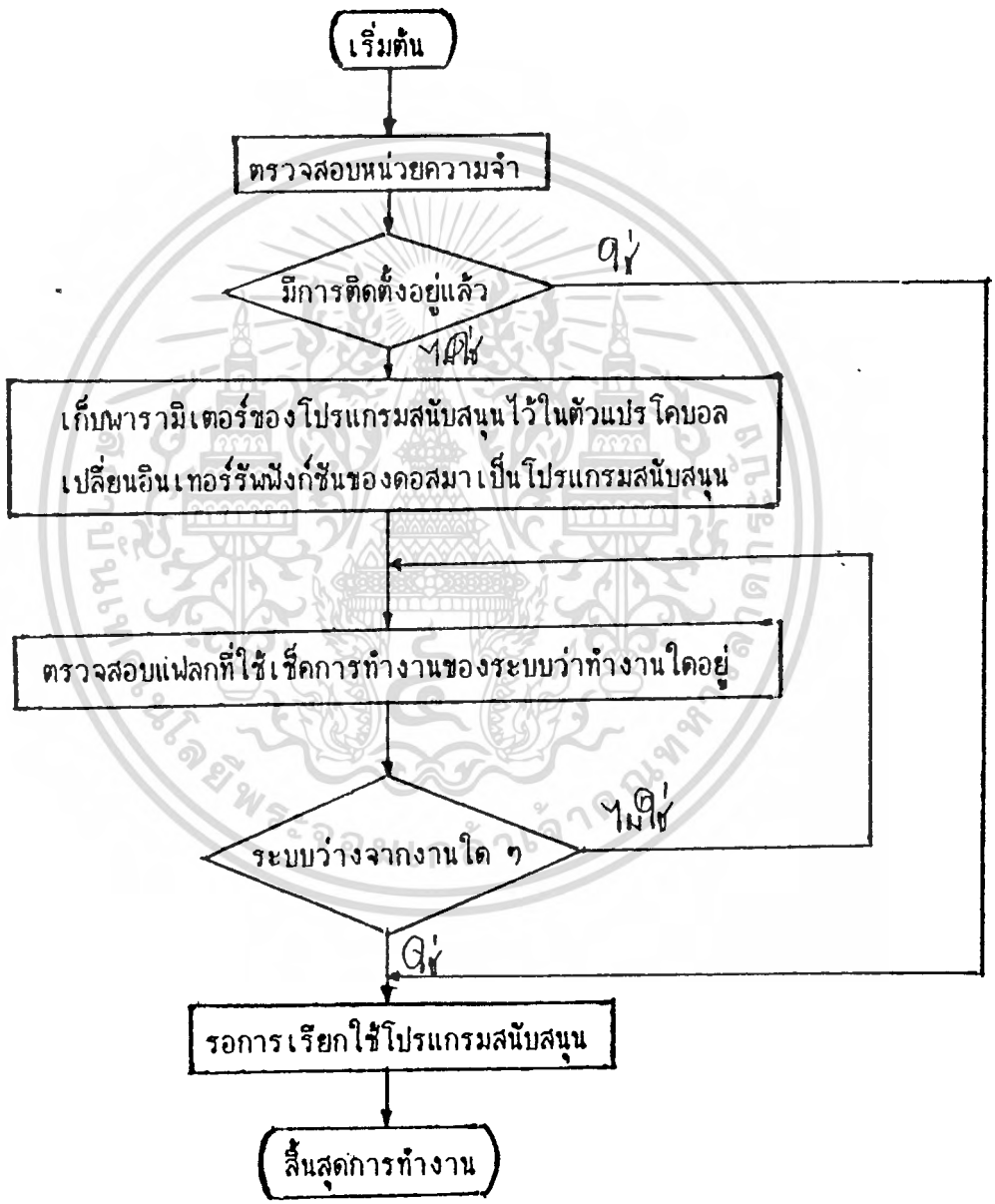
BX = Current PSP Segment

ดังนั้นอย่างแรกที่โปรแกรมสนับสนุนต้องทำ คือ การเซฟ PSP เก่า และเซ็ตตำแหน่งของ PSP ใหม่ แต่ในคอมไพเลอร์บางตัวจะมีตัวแปร Global ที่จะถูกทำการให้ค่าเซ็กเมนต์ PSP โดยอัตโนมัติ เช่น ในเทอร์โบปาสคาลจะมีตัวแปรที่ใช้เก็บตำแหน่งเซกเมนต์

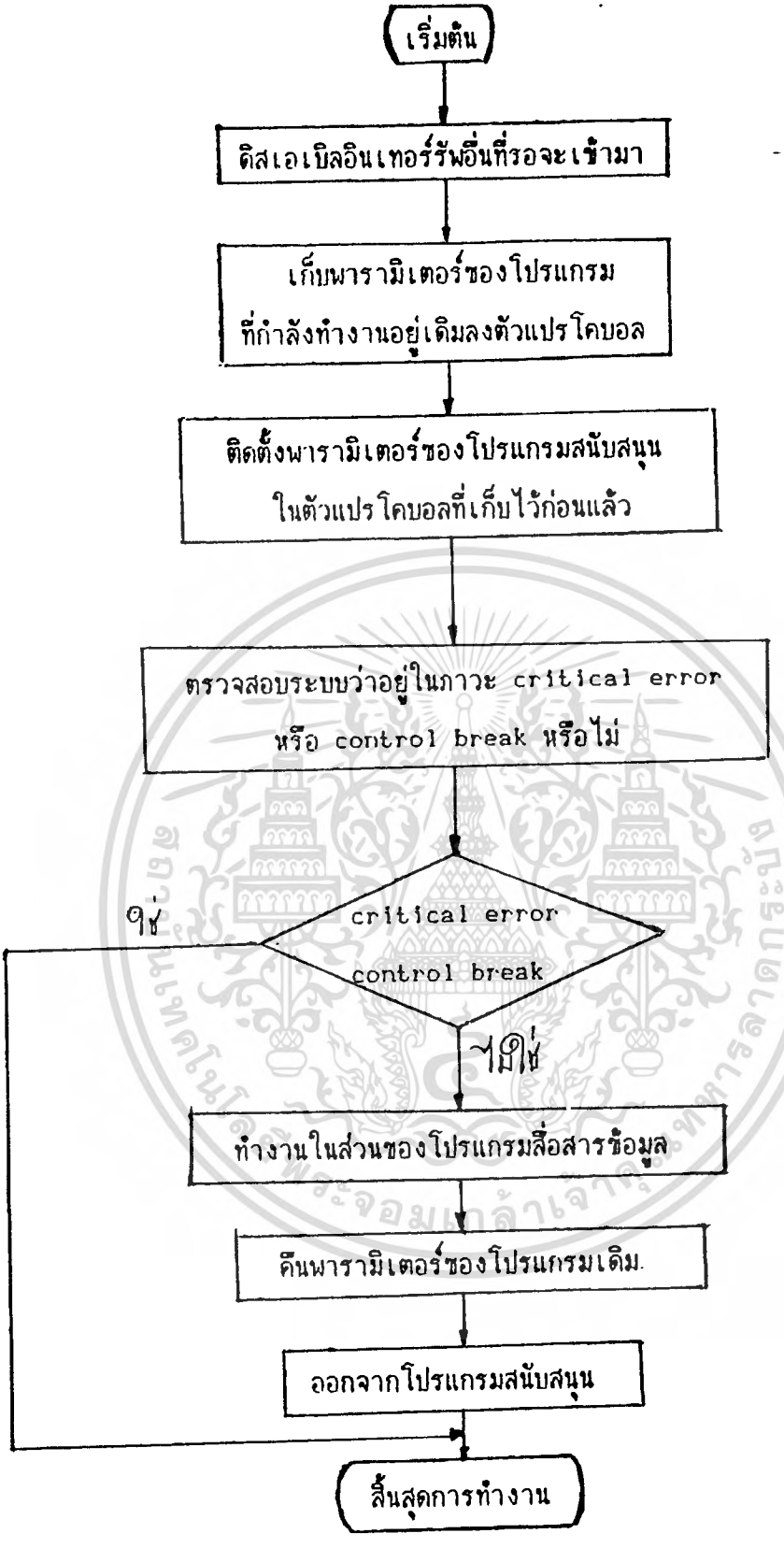
ของ PSP คือตัวแปรคงที่ที่ชื่อ Prefixseg ซึ่งเซกเมนต์ของ PSP นี้ควรได้ก่อนโปรแกรมสนับสนุนจะฝังลงในหน่วยความจำ

ความแตกต่างของอินเตอร์รัพดอสฟังก์ชัน 51h และ 62h คือ ฟังก์ชัน 51h เป็นฟังก์ชันในดอสเวอร์ชัน 2.x ซึ่งจะมีบั๊กมากเมื่อใช้กับอินเตอร์รัพที่ 28h (เกี่ยวกับการจัดการสแต๊ก) และอินเตอร์รัพใหม่ที่ไม่มีบั๊กในดอส 3.x คือ อินเตอร์รัพ 62h

(โปรดดูโครงสร้างของโปรแกรมสนับสนุนในรูปที่ 3.13)



รูปที่ 3.13a โครงสร้างของโปรแกรมสนับสนุน



รูปที่ 3.13b เมื่อมีการเรียกใช้โปรแกรมสนับสนุนโดยการกดคีย์บอร์ด

บทที่ 4

การทดลองและผลการทดลอง

โครงงานนี้ได้ทดลองสื่อสารข้อมูลโดยแบ่งการทำงานออกเป็นรูปแบบต่าง ๆ กัน ซึ่งมีโปรแกรมและฮาร์ดแวร์ควบคุมในส่วนต่าง ๆ เหล่านี้ ผลการทดลองในส่วนต่าง ๆ มีดังต่อไปนี้

4.1 การทดลองวัดค่าระดับสัญญาณที่ส่วนต่าง ๆ ในส่วนของชุดเครื่องส่ง

โครงงานนี้ได้ใช้โมเด็มที่มีอัตราเร็ว 1200 บอด ซึ่งมีการมอดูเลชั่นแบบฟรีเควนซีชิฟคีย์อิง (Frequency Shift Keying) ซึ่งควบคุมการทำงานด้วยไอซี MSM 6927 RS ดูวงจรในภาคผนวก ก.

4.1.1 ระดับสัญญาณที่ส่วนอินพุทของวงจรมอดูเลเตอร์

สัญญาณเมื่อผ่านไลน์ไดเวอร์ (line driver) และไลน์รีซีฟเวอร์ (line receiver) จะได้สัญญาณพัลส์ข้อมูลที่มีระดับสัญญาณ 10-12 โวลต์ เข้าสู่คอมพิวเตอร์ และสัญญาณพัลส์ข้อมูลที่มีระดับสัญญาณ 5 โวลต์ เข้าสู่ภาคมอดูเลเตอร์

4.1.2 การวัดระดับสัญญาณเมื่อทำการรับส่งข้อมูล

ระดับของสัญญาณที่ทำให้เครื่องรับตอบสนอง("on") เท่ากับ 8.5 mV_{p-p}

ระดับของสัญญาณที่ทำให้เครื่องรับไม่ตอบสนองการทำงาน ("off") เท่ากับ 5.6 mV_{p-p}

4.1.3 การวัดระดับสัญญาณที่ใช้ในการรับส่งข้อมูล

จากการทดลองส่งในระยะทางต่าง ๆ โดยใช้ดัมมีโหลด (Dummy Load) ระดับสัญญาณสูงสุดจะมีค่าประมาณ 13.8 dBm

โครงงานนี้ได้ทดลองรับส่งข้อมูล โดยใช้ระยะห่างประมาณ 10 เมตรจะได้ระดับสัญญาณ ดังนี้ คือ

เมื่อใช้ดัมมีโหลดทั้งด้านส่งและรับเท่ากับ -41.5 dBm

เมื่อใช้สายอากาศที่มี VSWR = 3.1 ได้ระดับสัญญาณเท่ากับ -18.5 dBm

4.2 การวัดความถี่ของสัญญาณที่เอาต์พุทของวงจรมอดูเลเตอร์

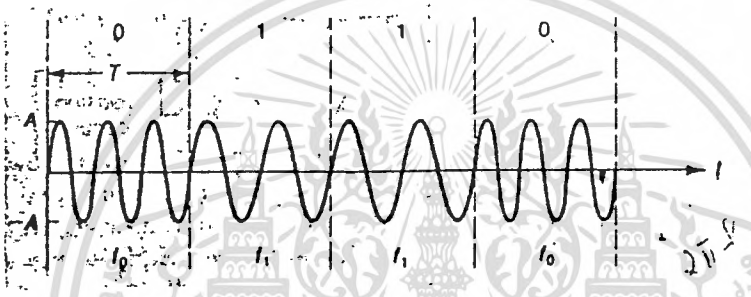
เมื่อสัญญาณข้อมูลเป็นไบนารี่ มีค่า "1" สัญญาณเอาต์พุทจากวงจรมอดูเลเตอร์จะมีความถี่เท่ากับ 1298.5 Hz

เมื่อสัญญาณข้อมูลเป็นไบนารี่ มีค่า "0" สัญญาณเอาต์พุทจากวงจรมอดูเลเตอร์จะมีความถี่เท่ากับ 2097.6 Hz

จากค่าดังกล่าวทำให้สามารถแสดงสมการของสัญญาณมอดูเลทได้ ดังนี้

สัญญาณมอดูเลท $S1(t) = A \cos(2 \quad 1298.5)\ t$ สำหรับไบนารี่ 1

สัญญาณมอดูเลท $S2(t) = A \cos(2 \quad 2097.6)\ t$ สำหรับไบนารี่ 0



รูปที่ 4.1 รูปคลื่นของฟรีเควนซีชิฟคีย์อิง

4.3 การทดลองส่งข้อมูลที่มีรูปแบบแตกต่างกันผ่านระบบวิทยุ

4.3.1 ทำการทดลองส่งข้อมูลเป็นข้อความ (message) จำนวน 30 บรรทัด บรรทัดละ 40 ตัวอักขระ (ในขณะทำการติดต่อจะมีสัญญาณพาหะเชื่อมโยงการติดต่ออยู่ตลอดเวลา) โดยที่ไม่ใช้โปรแกรมในการแก้ไขข้อผิดพลาด พบว่าจะเกิดอักขระที่ไม่ต้องการขึ้นที่ส่วนหัวของบล็อคข้อความในรูปแบบที่ไม่แน่นอน อักขระดังกล่าวมีลักษณะที่ซ้ำ ๆ กัน เช่น

(0CH), Blank (FFH), @ (040H), ? (3FH)

และเกิด Overrun error ขึ้น 1 ครั้งในระหว่างการทดลอง ดูรูปที่ 4.2

4.3.2 ทำการส่งข้อมูลเป็นไฟล์ข้อมูล (Text File) ซึ่งมีขนาดต่าง ๆ กัน และกราฟนิคไฟล์ (31326 ไบต์)

โดยที่ไม่ใช้โปรแกรมในการแก้ไขข้อผิดพลาดจะพบว่า ข้อผิดพลาดที่เกิดขึ้นส่วนใหญ่จะเป็นการ Overrun error และจะมีกรณีของ Parity error เกิดขึ้นน้อยมาก รวมทั้ง

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
gggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
llllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
```

รูปที่ 4.2 ผลการส่งข้อมูลเป็นข้อความหลังจากแก้ข้อผิดพลาดแล้ว

ความผิดพลาดที่เกิดขึ้นจาก ความล่าช้าในการโอนย้ายข้อมูล กับเครื่องพิมพ์ (ในกรณีที่ต้องการพิมพ์ออกทางเครื่องพิมพ์ในระหว่างการส่ง) ซึ่งความผิดพลาดชนิดนี้จะเพิ่มมากขึ้น เมื่อบรรทัดแต่ละบรรทัดมีขนาดสั้น ๆ ดังรูปที่ 4.3

ส่วนในการทดลองส่งกราฟฟิคไฟล์ (ซึ่งใช้การอ่านข้อมูลของมอนิเตอร์จากหน่วยความจำตำแหน่งวีดีโอแรม แล้วจึงส่งข้อมูลออกอากาศ) จะพบว่าเมื่อทดลองส่งจำนวน 5 ไฟล์ และในแต่ละไฟล์จะมีจำนวน 31326 ไบท์ ซึ่งเป็นขนาดของหน่วยความจำของจอโมโนโครม (720x360 pixel) ซึ่งใช้เวลาในการส่งแต่ละไฟล์เท่ากับ จะเกิดความผิดพลาดชนิด Overrun ขึ้นสองครั้ง และ Parity error หนึ่งครั้ง รวมทั้งอักขระที่ผิดพลาด ที่ไม่ทราบสาเหตุอีก 2 ครั้ง ดูรูปที่ 4.4

4.4 การทดลองวัดค่าต่าง ๆ ในวงจรของอุปกรณ์เก็บข้อมูล

4.4.1 ในวงจรผลิตความถี่

- วัดความถี่เอาท์พุตที่ขา 3 ของไอซี 555 ได้ประมาณ 1 KHz

```
begin to send file after corrected some of errors

  s_file
  name of file : ppt.pas
  program comm1;
  uses crt;
  var status:byte;
     y,i,a:integer;
  procedure set_commparameter;
  begin
  port[$3fb] := $83;
  port[$3f9]:=$00;
  port[$3f8]:=$60;
  port[$3fb]:=$03;
  end;
  procedure rts_cts;
  begin
  port[$3fc] :=$02;
  status :=$00;
  while (status =$00) do
  begin
  status := port[$3fe];
  status := status and $10;
  end;
  end;
  procedure transmit;
  begin
  for y:= 1 to 2 do
  begin
  for i:= 0 to 150 do
  begin
  status := $00;
  while (status =$00) do
  begin
  status :=port[$3fd];
  status :=status and $20;
  end;
  a := 3;
  port[$3f8] :=a;
  writeln(a:5);
  end;
  end;
  writeln('ok');
  end;
  begin
  set_commparameter;
  rts_cts;
  writeln('begin of transmission');
  transmit;
  delay(100);
  writeln('end of transmission');
```

รูปที่ 4.3 ผลการส่งไฟล์ข้อมูลหลังจากการแก้ไขความผิดพลาดแล้ว

```
begin to send graphic file after corrected some type of error
s_graphic
name of file =saml.pic
```

รูปที่ 4.4 ผลการส่งกราฟฟิคไฟล์หลังการแก้ไขข้อผิดพลาดแล้ว

โดยความถี่ที่ได้สามารถเปลี่ยนแปลงได้ขึ้นอยู่กับค่า R และ C ในที่นี้เราใช้ค่าความจุของตัวเก็บประจุคงที่ จึงใช้ค่าความต้านทานแบบปรับค่าได้

4.4.2 วงจรหารความถี่

- วัดความถี่เอาท์พุตของไอซี 4060 ที่ขา 13 ซึ่งทำการหารค่าความถี่ที่ได้จากไอซี 555 ด้วยค่า 2^{5} ได้ค่าความถี่เท่ากับ 2 Hz (ไอซีตัวนี้สามารถหารค่าได้ถึง 2^{16} ซึ่งในที่นี้เราต้องการให้ได้ค่าความถี่ที่ 2 Hz จึงเลือกค่าที่หารด้วย 2^{5}) การที่จะเลือกหารด้วยค่าใดขึ้นอยู่กับค่าความถี่ที่ป้อนเข้ามาจากไอซี 555

- วัดค่าที่ได้จากขา Q (13) ของไอซี 4013 คือ สัญญาณพัลส์ความถี่ 1 Hz เพื่อใช้เป็นตัวกำหนดคาบเวลาสัญญาณนาฬิกาให้กับวงจรอื่น

4.4.3 การทดลองตรวจสอบวงจรนับ

ทำการทดลองเช็คการทำงานของวงจรนับโดยการตั้งสวิทช์รหัส BCD เพื่อกำหนดคาบเวลาในการสุ่มแต่ละครั้ง

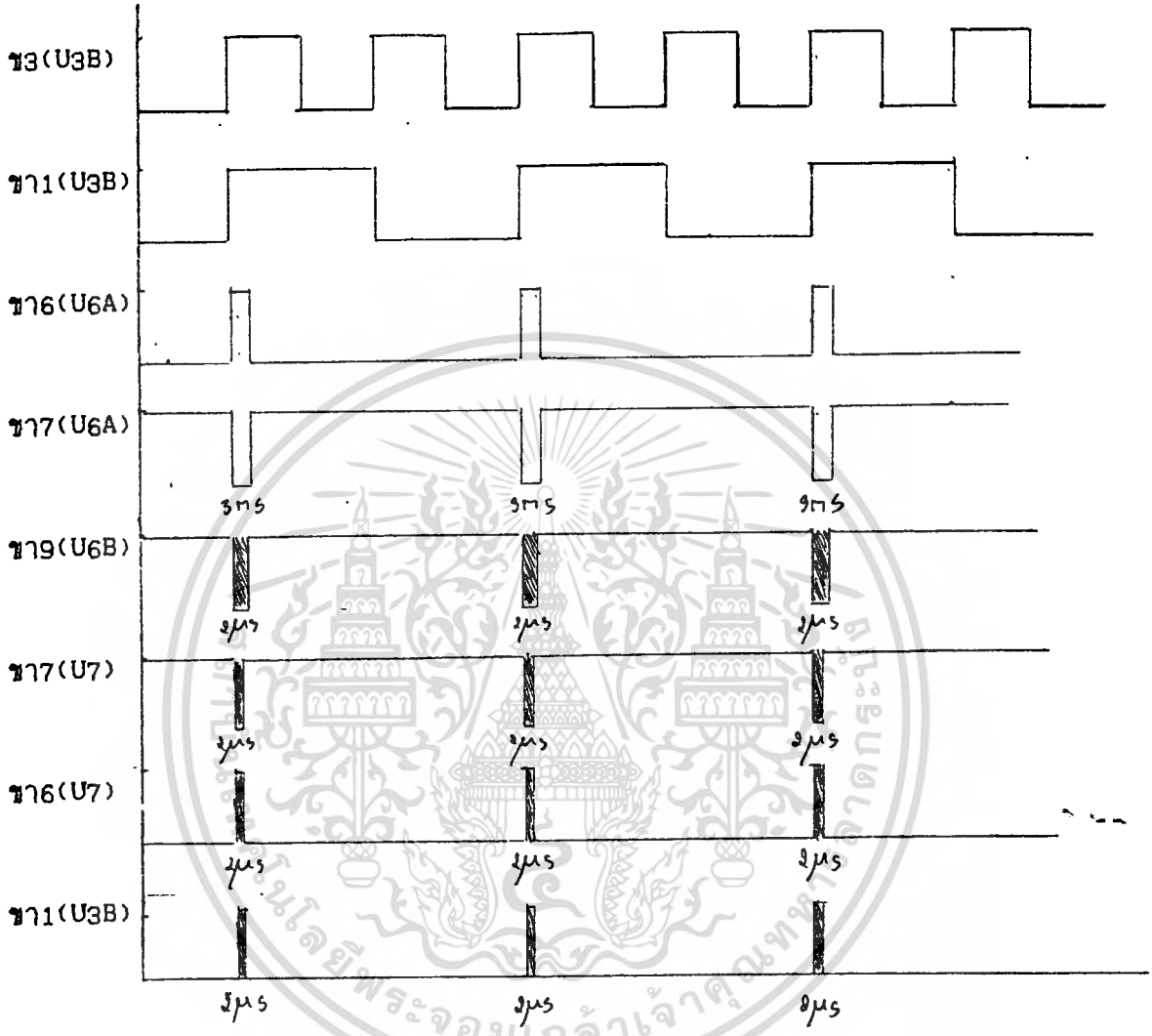
- ทำการวัดที่ขา 1 ของไอซี 4013 จะได้ค่าเป็น +5 V หรือลอจิก "1" ทุก ๆ คาบเวลา

วัดการทำงานของสวิทช์รหัส BCD โดยดูจากการที่ตั้งเวลาที่จะเก็บข้อมูลไว้ แล้วทำการตรวจสอบดูว่ามีการทำงานทุก ๆ ช่วงเวลาที่ตั้งไว้

- จากการทดลองวัดดูได้ค่าใกล้เคียงมากหรืออาจถือได้ว่าไม่ผิดพลาดก็ได้

ผลที่ได้จากขาเอาท์พุตของไอซี 4013 จะใช้เป็นพัลส์เริ่มต้นการทำงานและไว้ตรวจสอบไทมิ่งที่จุดต่าง ๆ

- ทดลองวัดไทมิ่งที่ขาต่าง ๆ ได้ ดังรูปที่ 4.5



รูป 4.5 ไทม์มิ่งของอุปกรณ์เก็บข้อมูล

บทที่ 5
บทสรุปและวิจารณ์

ในการติดต่อสื่อสารข้อมูลผ่านระบบวิทยุนั้น สิ่งที่ต้องคำนึงถึงอย่างมาก คือ ความเร็วในการส่งข้อมูล และข้อผิดพลาด (error) เกิดขึ้นในระหว่างการส่งข้อมูล รวมทั้งประสิทธิภาพของอุปกรณ์ในการใช้งานด้วยเช่นกัน ซึ่งความเร็วในการส่งข้อมูลนั้นขึ้นอยู่กับคุณสมบัติของฮาร์ดแวร์ที่ใช้และความสามารถของโปรแกรมในการจัดรูปแบบข้อมูล

ข้อผิดพลาด (error) ที่อาจจะเกิดขึ้นนั้น เกิดจากสาเหตุหลายประการ เช่น Overrun error ซึ่งเกิดจากข้อมูลใหม่ซ้อนทับข้อมูลเก่าก่อนที่จะถูกอ่านออกไปใช้งาน วิธีแก้ไข Overrun error วิธีหนึ่งคือ ควบคุมความเร็วในการส่งข้อมูลแต่ละชุดให้สอดคล้องกันทั้งด้านรับและด้านส่ง โดยคำนึงถึงเวลาที่ใช้ในการติดต่อกับอุปกรณ์ภายนอกอื่น ๆ เช่น เครื่องพิมพ์ เป็นต้น ส่วน Parity error ซึ่งเกิดจากการตรวจสอบบิทพาริตี้ไม่ถูกต้องสอดคล้องกับข้อมูล ถ้าเกิดความผิดพลาดชนิดนี้ขึ้น ควรร้องขอให้ส่งข้อมูลนั้นมาใหม่หรือหยุดการทำงาน ความผิดพลาดที่เกิดจากความล่าช้าในการโอนย้ายข้อมูลกับอุปกรณ์ภายนอกไม่เร็วพอกับความเร็วของการส่งข้อมูล ความผิดพลาดชนิดนี้จะเพิ่มมากขึ้นเมื่อบรรทัดแต่ละบรรทัดมีขนาดสั้น ๆ ติดต่อกันหลาย ๆ บรรทัด วิธีแก้ไขวิธีหนึ่ง คือ หน่วงความเร็วทางด้านส่งลงในขณะที่มีการติดต่อกับอุปกรณ์ภายนอกเหล่านั้น ยกตัวอย่างเช่น เครื่องพิมพ์มีการทำงานขึ้นบรรทัดใหม่ (line feed) ทุก ๆ ครั้งที่มี Carriage return ทั้งข้อผิดพลาดที่เกิดจากอุปกรณ์เองซึ่งสามารถตรวจสอบได้โดยส่งข้อมูลติดต่อกันเป็นเวลานานๆในคาบเวลาที่คงที่ค่าหนึ่งเป็นต้น.

ในการเพิ่มประสิทธิภาพของอุปกรณ์ในการใช้งาน เพื่อให้ผู้ใช้สามารถใช้อุปกรณ์ได้อย่างเต็มที่และหลาย ๆ ลักษณะในเวลาเดียวกัน

ในโครงงานนี้ได้ใช้โปรแกรมเป็นส่วนใหญ่เพื่อตอบสนองคุณลักษณะดังกล่าว เช่น ลดความผิดพลาดของข้อมูล ตรวจสอบความผิดพลาดของอุปกรณ์ เมื่อทำการส่งติดต่อกัน รวมทั้งการเพิ่มประสิทธิภาพของอุปกรณ์โดยการใช้คีย์บอร์ดอินเตอร์รัพ

หวังว่าที่กล่าวมาทั้งหมดนี้คงจะเป็นประโยชน์ต่อผู้ที่สนใจไม่มากก็น้อย

โมเด็มสำหรับการสื่อสารข้อมูลผ่านระบบวิทยุ จะต้องสามารถทำหน้าที่ต่าง ๆ ดังต่อไปนี้คือ

1. การอินเตอร์เฟสเข้ากับอุปกรณ์ปลายทางข้อมูลหรือคอมพิวเตอร์

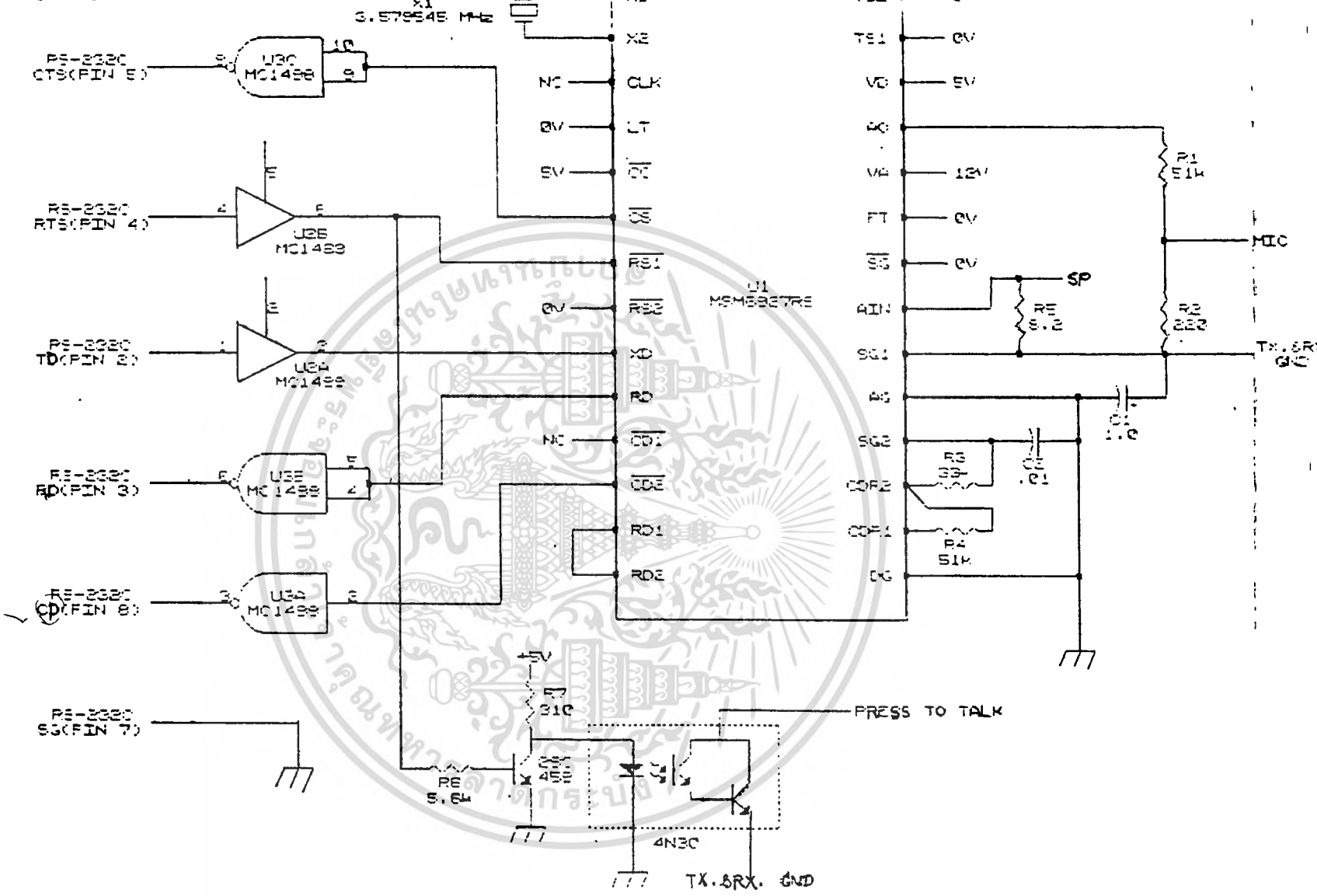
2. การโมดูเลชัน และการดีโมดูเลชัน

3. การตรวจสอบการออกอากาศจากเครื่องส่งวิทยุอื่น

4. การควบคุมการออกอากาศของเครื่องส่งวิทยุ

5. การส่งสัญญาณเสียงพูดไปให้กับเครื่องส่งวิทยุและการรับสัญญาณเสียงพูดจากเครื่องรับวิทยุ

จากหน้าที่ต่าง ๆ ของโมเด็มสำหรับการสื่อสารข้อมูลผ่านระบบวิทยุ จึงได้ใช้โมเด็ม 1200 บิทต่อวินาที โดยนำเอาไอซี MSM 6927 RS มาใช้งานดังรูปหน้าถัดไป

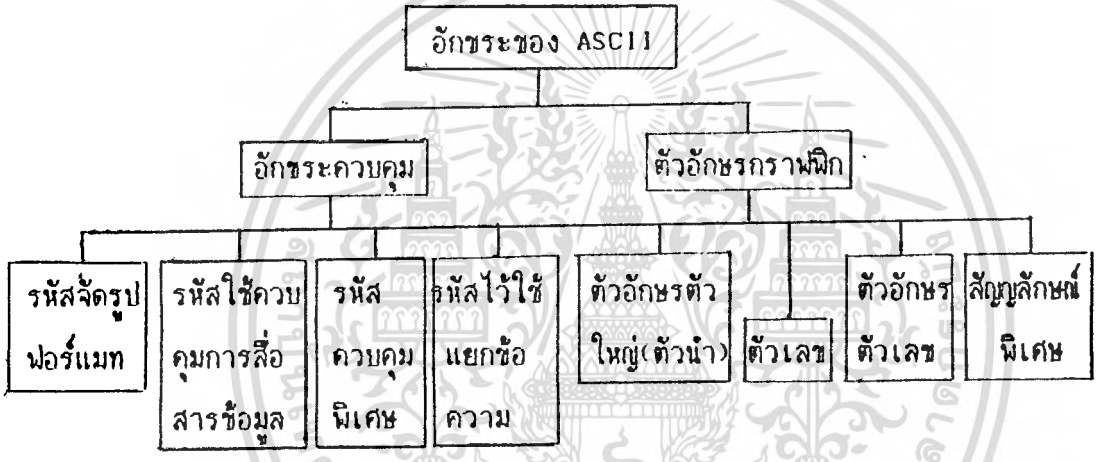


รูปแสดงวงจรของโมเด็ม 1200 บิทต่อวินาทีสำหรับการสื่อสารข้อมูลผ่านระบบวิทยุ

ANSI (American National Standard Institue) ได้กำหนดมาตรฐานของรหัส ASCII ออกเป็นสองพวกใหญ่ ๆ ตามลักษณะการใช้งานคือ อักขระที่ทำให้เกิดข้อความที่อ่านเข้าใจได้เรียกว่า ตัวอักขระกราฟิค (Graphic character) และอักขระที่ใช้ทำให้เกิดการควบคุม เรียกว่าอักขระควบคุม (Control characters)

อักขระควบคุมของ ASCII แบ่งออกเป็นกลุ่มย่อยตามการใช้งาน คือ

- รหัสที่ใช้ควบคุมการสื่อสารข้อมูล(communication control)
- รหัสจัดรูปฟอร์แมต(format effectors)
- รหัสไว้ใช้แยกข้อความ(information separator)
- รหัสควบคุมพิเศษ(special control characters)



รูปที่ 1 แผนภูมิของอักขระ ASCII

รูปที่1 แสดงถึงแผนภูมิของอักขระในรหัสของ ASCII กลุ่มของรหัสที่ใช้ในการควบคุมทั้งหมด บางทีก็เรียกว่า รหัสที่ไม่สามารถพิมพ์ออกมาดูได้(nonprinting control characters) แต่ในไมโครคอมพิวเตอร์บางเครื่องใช้อักขระเหล่านี้ สำหรับการเข้ารหัสรูปกราฟิก ดังตัวอย่างของ IBM PC แสดงไว้ในตารางที่ 1

รหัสที่ใช้ในการสื่อสารข้อมูลนั้น แสดงในตารางที่ 1 ซึ่งเป็นตารางที่แสดงถึงรหัส 128 ตัวแรกของ ASCII โดยประกอบด้วยอักขระภาษาอังกฤษ 26 ตัวใหญ่ 26 ตัวเล็ก เลขโรมัน 10 ตัว และอักขระพิเศษที่มีอยู่ตามเครื่องพิมพ์ดีดทั่วไป รหัสการสื่อ

สารถูกรวมเข้าไปใช้อยู่ในที่นี้ด้วย

ตารางที่ 1 ตารางรหัสแอสกี้

ASCII Value	Hex Value	Character	Control Character
000	00	(null)	NUL
001	01	☺	SOH
002	02	☻	STX
003	03	♥	ETX
004	04	♦	EOT
005	05	♣	ENO
006	06	♠	ACK
007	07	•	BEL
008	08	◘	(backspace) BS
009	09	○	(tab) HT
010	OA	◙	(line feed) LF
011	OB	♂	(home) VT
012	OC	♀	(form feed) FF
013	OD	♪	(carriage return) CR
014	OE	♫	SO
015	OF	☼	SI
016	10	►	DLE
017	11	◄	DCI
018	12	↕	DC2
019	13	‼	DC3
020	14	¶	DC4
021	15	§	NAK
022	16	▬	SYN
023	17	↨	ETB
024	18	↑	CAN
025	19	↓	EM
026	1A	→	SUB
027	1B	←	ESC
028	1C	∟	(cursor right) FS
029	1D	↔	(cursor left) GS
030	1E	▲	(cursor up) RS
031	1F	▼	(cursor down) US
032	20	(space)	
033	21	!	
034	22	"	
035	23	#	
036	24	$	
037	25	%	
038	26	&	
039	27	'	

ASCII Value	Hex Value	Character
040	28	(
041	29	)
042	2A	*
043	2B	+
044	2C	,
045	2D	—
046	2E	.
047	2F	/
048	30	0
049	31	1
050	32	2
051	33	3
052	34	4
053	35	5
054	36	6
055	37	7
056	38	8
057	39	9
058	3A	:
059	3B	;
060	3C	<
061	3D	=
062	3E	>
063	3F	?
064	40	@
065	41	A
066	42	B
067	43	C
068	44	D
069	45	E
070	46	F
071	47	G
072	48	H
073	49	I
074	4A	J
075	4B	K
076	4C	L
077	4D	M
078	4E	N
079	4F	O
080	50	P
081	51	Q
082	52	R
083	53	S
084	54	T
085	55	U
086	56	V
087	57	W
088	58	X
089	59	Y
090	5A	Z
091	5B	[
092	5C	\
093	5D	]
094	5E	^
095	5F	—
096	60	`
097	61	a
098	62	b
099	63	c

ASCII Value	Hex Value	Character
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒

ASCII Value	Hex Value	Character
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	▒
220	DC	▒
221	DD	▒
222	DE	▒
223	DF	▒
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	ø
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	n
253	FD	2
254	FE	■
255	FF	(Blank 'FF')

ตารางรหัส EBCDIC

ASCII Binary Code	EBCDIC Binary Code	Character
1000000	01111100	@
1000001	11000001	A
1000010	11000010	B
1000011	11000011	C
1000100	11000100	D
1000101	11000101	E
1000110	11000110	F
1000111	11000111	G
1001000	11001000	H
1001001	11001001	I
1001010	11010001	J
1001011	11010010	K
1001100	11010011	L
1001101	11010100	M
1001110	11010101	N
1001111	11010110	O
1010000	11011000	P
1010001	11011000	Q
1010010	11011001	R
1100101	11100010	S
1010100	11100011	T
1010101	11100100	U
1010110	11100101	V
1010111	11100110	W
1011000	11100111	X
1011001	11101000	Y
1011010	11011001	Z
1011011		[(left bracket)
1011100		/ (left slash)
1011101		] (right bracket)
1011110		> (caret or up arrow)
1011111	01101101	_
1100000	01111101	`
1100001	10000001	a
1100010	10000010	b
1100011	10000011	c
1100100	10000100	d
1100101	10000101	e
1100110	10000110	f
1100111	10000111	g
1101000	10001000	h
1101001	10001001	i
1101010	10001001	j
1101011	10010010	k
1101100	10010011	l
1101101	10010100	m
1101110	10010101	n
1101111	10010110	o
1110000	10010111	p
1110001	10011000	q
1110011	10011001	r
1110011	10100010	s
1110100	10100011	t
1110101	10100100	u
1110110	10100101	v
1110111	10100110	w
1111000	10100111	x
1111001	10101000	y
1111010	10101001	z
1111011		{
1111100	01101010	:
1111101		}
1111110		~
1111111		DEL

ASCII Binary Code	EBCDIC Binary Code	Character
0000000	00000000	NUL
0000001	00000001	SOH
0000010	00000010	STX
0000011	00000011	ETX
0000100		EOT
0000101	00101101	ENQ
0000110		ACK
0000111		BEL
0001000		BS
0001001		HT
0001010		LF
0001011		VT
0001100	00001100	FF
0001101		CR
0001110		SO
0001111		SI
0010000		DLE
0010001		DC1
0010010		DC2
0010011		DC3
0010100		DC4
0010101		NAK
0010110	00110010	SYN
0010111	00100110	ETB
0011000		CAN
0011001	00011001	EM
0011001	00111111	SUB
0011011	00010111	ESC
0011100		FS
0011101		GS
0011110		RS
0011111		US
0100000	01000000	SP
0100001	01011010	!
0100010	01111111	"
0100011		#
0100100	01011011	$
0100101	01101100	%
0100110	01010000	&
0100111	01111101	'
0101000	01001101	(
0101001	01011101	)
0101010		*
0101011	01001110	+
0101100	01101011	,
0101101	01100000	-
0101110	01001001	.
0101111	01100001	/
0110000	11110000	0
0110001	11110001	1
0110010	11110010	2
0111100	11110011	3
0110100	11110100	4
0110101	11110101	5
0110110	11110110	6
0110111	11110111	7
0111000	11111000	8
0111001	11111001	9
0111010	01111010	:
0111011	01011110	;
0111100	01001100	<
0111101	01111011	=
0111110	01101110	>
0111111	01101111	?

ภาคผนวก ค.

รีจิสเตอร์ควบคุมสายสื่อสาร (line control register)

ในการควบคุมรูปแบบของข้อมูลแบบอะซิงโครนัสนั้น ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร รีจิสเตอร์ตัวนี้มี 8 บิท โดยแต่ละบิทมีความหมายดังนี้

บิท 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิท 0	บิท 1	
0	0	หมายถึงข้อมูลขนาด 5 บิท
0	1	หมายถึงข้อมูลขนาด 6 บิท
1	0	หมายถึงข้อมูลขนาด 7 บิท
1	1	หมายถึงข้อมูลขนาด 8 บิท

บิท 2 เป็นบิทที่ใช้ในการกำหนดจำนวนสต๊อปบิท ถ้าเป็น "0" หมายถึงใช้สต๊อปบิท 1 บิท แต่ถ้าบิท 2 เป็น "1" ในกรณีส่งแบบ 5 บิทจะมีความยาวของสต๊อปบิทเป็น 1.5 บิท แต่ถ้าส่งแบบ 6,7 หรือ 8 บิทความยาวของสต๊อปบิทจะเป็น 2

บิท 3 เป็นบิทแสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้ โดยถ้าบิทนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้

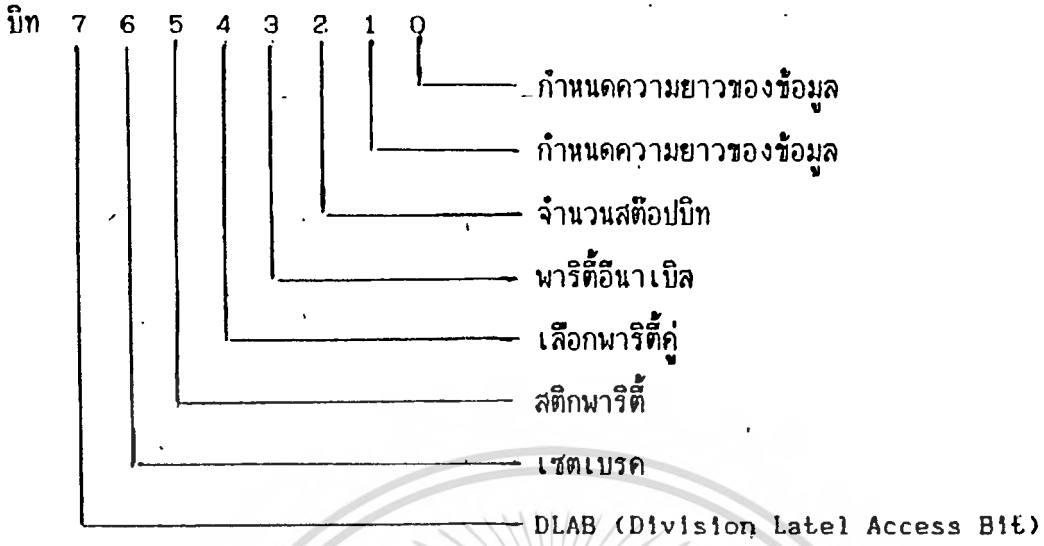
บิท 4 มีค่าเป็น "0" และบิท 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิทนี้มีค่าเป็น "1" จะเป็นพาริตี้คู่

บิท 5 เมื่อบิท 3 มีค่าเป็น "1" และบิท 5 มีค่าเป็น "1" และบิท 4 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี้ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิท 4 มีค่าเป็น "0" บิท 3 มีค่าเป็น "1" และบิท 5 มีค่าเป็น "1" จะมีการกำหนดบิทพาริตี้เป็น "1"

บิท 6 ควบคุมการเบรค เมื่อบิท 6 มีค่าเป็น "1" ส่วนของ SOUTจะได้รับการกำหนดให้เป็น "0" ตลอด

บิท 7 ทำหน้าที่เป็น DLAB บิทที่จะมีผลต่อการแลตช์ตัวหารดังที่กล่าวมาแล้วในตารางที่ 1 และ 2

ตารางที่ 2 พอร์ตแอดเดรสหมายเลข 3FB



การโปรแกรมอัตราบอด (baud rate generator)

อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz และสามารถโปรแกรมตัวหารได้ตั้งแต่ $1-(2^{16}-1)$ ค่าความถี่เอาท์พุตของตัวกำหนดอัตราบอดมีค่าเท่ากับ 16×อัตราบอด ดังนั้นตัวหาร = ความถี่สัญญาณนาฬิกา/(อัตราบอด×16) การกำหนดอัตราบอดด้วยการกำหนดตัวหารนี้จึงเป็นค่าที่กำหนดในรีจิสเตอร์สองตัว ตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ DLAB=1 แล้วให้ลดลงมาในรีจิสเตอร์ 3F8 ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F9 เมื่อ DLSB =1 จะเป็นค่าของตัวหาร MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตารางที่ 4

รีจิสเตอร์แสดงสถานะสายสื่อสาร (line status register)

รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบองบิทต่าง ๆ ในรีจิสเตอร์จะเป็นดังนี้

บิท 0 เป็นบิทที่บอกสถานะการรับข้อมูล ถ้าเป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิทแล้ว บิทนี้จะได้รับการรีเซ็ทให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว หรือจะให้ซีพียูเขียนข้อมูลกลับมายังรีจิสเตอร์นี้ก็ได้

บิท 1 ถ้ามีค่าเป็น "1" แสดงว่าเกิด overrun error กล่าวคือขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไปปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับบนบัฟเฟอร์นี้ บิทนี้จะรีเซต

ตารางที่ 4 ค่าตัวหารสำหรับกำหนดอัตราบอด

อัตราบอด	ตัวหารฐานสิบ	ตัวหารฐานสิบหก	ค่าผิดพลาด
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	0.69
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

โดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิท 2 มีค่าเป็น "1" แสดงว่าเกิด parity error คือถ้ามีการตรวจสอบบิทพาริตี้แล้วไม่เป็นไปตามที่กำหนดไว้ บิทนี้จะได้รับการรีเซตโดยซีพียู เมื่อซีพียูอ่านค่าจากรีจิสเตอร์ไปแล้ว

บิท 3 ถ้ามีค่าเป็น "1" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่น ตรวจสอบจำนวนบิทโดยดูที่พาริตี้และสต๊อปบิทไม่เป็นไปตามที่กำหนด

บิท 4 เรียกว่า break Interrupt (BI) บิทนี้จะได้รับการเซตให้มีค่าเป็น "1" ถ้าหากว่ารับข้อมูลอินพุทเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิท 5 เป็นบิทที่บอกว่า 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสาร บิทนี้จะได้รับการเซตให้มีค่าเป็น "1" บิทนี้ยังคงสร้างสัญญาณอินเตอร์รัพเมื่อส่งไปบอกซีพียูด้วย บิทนี้จะ

มีสถานะเซต เมื่อมีการส่งถ่ายข้อมูลจากโฮล์ดรีจิสเตอร์ไปยังชิฟรีจิสเตอร์เพื่อพร้อมที่จะส่ง

บิท 6 เป็นบิทที่จะบอกว่าชิฟรีจิสเตอร์ว่างเปล่า บิทนี้จะได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิท 7 จะเป็น "0" ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพ (IRR-Interrupt Identification Register)

ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพภายในชิพ เพื่อว่าให้การทำงานระหว่าง 8250 กับซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และเพื่อให้เขียนซอฟต์แวร์ได้ง่ายและสั้นลงได้มาก 8250 กำหนดความสำคัญของอินเตอร์รัพไว้ 4 ระดับ คือ

ระดับแรก สถานะการรับข้อมูลจากสายสื่อสาร

ระดับสอง การพร้อมรับข้อมูล

ระดับสาม ขณะรีจิสเตอร์โฮลดิ้งสำหรับส่งข่าว

ระดับสี่ สัญญาณสถานะโมเด็ม

ในขณะที่มีความต้องการอินเตอร์รัพหลายระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่ารอไว้ก่อนโดยเก็บสถานะการอินเตอร์รัพนี้ไว้ใน IIR

ความหมายของรีจิสเตอร์มีดังนี้

บิท 0 เป็นบิทที่ใช้แสดงว่ามีอินเตอร์รัพเกิดขึ้นหรือไม่ ซึ่งสามารถให้ซีพียูตรวจสอบ ดูด้วยวิธีการโพลลิ่งได้ ถ้าบิทนี้เป็น "1" หมายถึง ไม่มีอินเตอร์รัพเกิดขึ้น

บิท 1-2 เป็นบิทที่แสดงความหมายว่าการอินเตอร์รัพที่เกิดขึ้นนั้นมาจากการอินเตอร์รัพตามฟังก์ชันใด

บิท 3-7 มีค่าเป็น "0"

รีจิสเตอร์อีนาเบิลอินเตอร์รัพ (INTRPT-Interrupt Enable Register)

ใน COM1 เมื่อให้ DLAB = 0 พอร์ต 3F9 จะเป็นรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์ ผู้ใช้สามารถกำหนดให้เกิดอินเตอร์รัพหรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจิสเตอร์นี้จากที่กล่าวแล้วว่าการอินเตอร์รัพของ 8250 นี้มี 4 แบบ ดังนั้นจึงต้องกำหนดการอีนาเบิลได้ทั้ง 4 แบบ ดังนั้นจึงต้องกำหนดการอีนาเบิลได้ทั้ง 4 แบบ โดยการใช้ข้อมูลแต่ละบิทของรีจิสเตอร์นี้เพื่อการกำหนดการอีนาเบิล ข้อมูลที่อยู่ในรีจิสเตอร์นี้มีความหมายดังนี้

บิท 0 ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพการพร้อมรับข้อมูล

บิท 1 ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพโฮลดิ้งรีจิสเตอร์ว่าง

บิท 2 ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพจากสถานะการรับข้อมูลจากสายสื่อสาร

บิท 3 ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิลอินเตอร์รัพจากสถานะโมเด็ม

บิท 4-7 ได้รับการกำหนดให้เป็น 0 เสมอ

รีจิสเตอร์ควบคุมโมเดม (modem control register)

รีจิสเตอร์ตัวนี้มีไว้ให้ซีพียูส่งผ่านข้อมูลมาเก็บเพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเดม การกำหนดพอร์ตของรีจิสเตอร์ตัวนี้คือ 3FC ข้อมูลต่าง ๆ ที่มีในรีจิสเตอร์ตัวนี้มีความหมายดังนี้

บิท 0 มีความหมายดังถึงการควบคุมสัญญาณ DTR เมื่อบิทนี้มีค่าเป็น "1" เอาท์พุตที่ DTR จะได้รับการกำหนดให้เป็น "0" และถ้าบิทนี้มีค่าเป็น "0" เอาท์พุตที่ DTR จะได้รับการกำหนดให้เป็น "1"

บิท 1 มีความหมายถึงสัญญาณ RTS ซึ่งจะมีผลเหมือนกับบิท 0 ในกรณีของ DTR

บิท 2 ใช้ควบคุมขาเอาท์พุต 1 (OUT1) ซึ่งจะมีผลเหมือนบิท 0

บิท 3 ใช้ควบคุมเอาท์พุต 2 (OUT2) ซึ่งจะมีผลเหมือนบิท 0

บิท 4 ใช้สำหรับการกำหนดวงรอบสำหรับการตรวจสอบ 8250 เมื่อบิท 4 นี้ได้รับการเซตเป็น "1" สิ่งที่จะเกิดขึ้นเป็นดังนี้ ข้อมูลที่ SOUT จะได้รับการเซตให้เป็นลอจิก "1" ขาข้อมูลอินพุต SIN จะได้รับการแยกตัวออก ข้อมูลของเอาท์พุตชิพรีจิสเตอร์จะได้รับการป้อนกลับมายังรีจิสเตอร์ข้อมูลอินพุต ส่วนสัญญาณ CTS, DSR, RLSD และ RI จะได้รับการแยกออกจากระบบแต่สัญญาณควบคุมโมเด็มคือ DTR, RTS, OUT1 และ OUT2 จะต่อเข้ากับสัญญาณทั้งสี่ที่เป็นอินพุต ดังนั้นจึงตรวจสอบระบบการทำงานได้

รีจิสเตอร์แสดงสถานะโมเด็ม

รีจิสเตอร์ตัวนี้จะเป็นตัวที่รับสถานะจากโมเด็มมาเก็บไว้ เพื่อให้ซีพียูสามารถอ่านตรวจสอบดูได้ สถานะของข้อมูลจะแอคตีฟเมื่อมีข้อมูลเป็น "1" และจะได้รับการรีเซตเมื่อซีพียูอ่านข้อมูลในรีจิสเตอร์นี้ไป พอร์ตที่ใช้กำหนดเป็นพอร์ตหมายเลข 3FF ข้อมูลภายใน

รีจิสเตอร์นี้เป็นดังนี้

บิท 0 บิทนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ CTS กล่าวคือ เมื่อขา CTS ของ 8250 ได้เปลี่ยนสถานะหลังจากที่ซีพียูได้อ่านสถานะนี้ไปแล้ว บิทนี้ก็จะได้รับการรีเซต ("0") และจะได้รับการเซต ("1") เมื่อมีการเปลี่ยนสถานะที่ขา CTS

บิท 1 เหมือนบิท 0 แต่เป็นบิทที่แสดงสถานะการเปลี่ยนแปลงของขา DSR

บิท 2 เป็นบิทที่แสดงว่าสัญญาณ RI ซึ่งเป็นอินพุตของ 8250 ได้รับการเปลี่ยนจากออน ("1") มาเป็นออฟ ("0")

บิท 3 เหมือนบิท 0 แต่เป็นบิทแสดงสถานะการเปลี่ยนแปลงของ line signal detector ซึ่งเป็นขาอินพุท RLSD

บิท 4 เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา CTS

บิท 5 เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา DSR

บิท 6 เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RI

บิท 7 เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RLSD

ถ้าบิท 4 ของ MCR ได้รับการเซตหรือให้ทำลูปตรวจสอบข้อมูลในบิท 4 จะเหมือนกับ RTS ใน MCR ข้อมูลในบิท 5 จะเหมือนกับ DTR ใน MCR ข้อมูลในบิท 6 จะเหมือนกับ OUT1 ใน MCR ข้อมูลในบิท 7 จะเหมือนกับ OUT2 ใน MCR

บัฟเฟอร์รีจิสเตอร์สำหรับตัวรับข้อมูล (receive buffer register)

เป็นรีจิสเตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนดคือหมายเลขแอดเดรส 3F8 ขณะที่ DLAB = 0 หากซีพียูอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

รีจิสเตอร์โฮลดิ้งสำหรับตัวส่งข้อมูล (transmitter holding register)

เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจิสเตอร์นี้จะรับข้อมูลจากซีพียู โดยที่กำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ DLAB = 0 ข้อมูลของซีพียูที่เอาท์พุตมาที่พอร์ตนี้ก็เพื่อจะส่งต่อไปยังสายสื่อสารข้อมูล

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงมาได้ด้วยความอนุเคราะห์จากบุคคลหลาย ๆ ท่าน ที่ได้ให้ความช่วยเหลืออย่างดียิ่งทั้งในด้านกำลังใจและกำลังกาย

ขอขอบคุณ อาจารย์ทุกท่านที่ให้คำปรึกษาและช่วยเหลืออย่างเต็มที่ พี่ ๆ เพื่อน ๆ รวมทั้งผู้ที่เกี่ยวข้องทุก ๆ คน

ส่วนดีของปริญญานิพนธ์ฉบับนี้ขอมอบให้บิดามารดา ครูบาอาจารย์และผู้มีพระคุณทุกๆ ท่าน ส่วนข้อบกพร่องใด ๆ ผู้จัดทำขอรับผิดไว้แต่ทั้งหมด

ผู้จัดทำ

หนังสืออ้างอิง

1. ธานินท์ ถาวรศาสนวงศ์ และทินกร ดุ๊ก, "การอินเตอร์เฟส IBM PC ", พิสิกส์เซนเตอร์ การพิมพ์, 2531.

2. เจเอสเคกรุป, "แอสเซมบลี้ 8086, 8088", พิสิกส์เซนเตอร์การพิมพ์, 2530.

3. IBM CORP., "TECHNICAL REFERENCE", IBM PERSONAL COMPUTER HARDWARE REFERENCE LIBRARY, 1983.

4. PHILIP ECG, "SEMICONDUCTORS MASTER REPLACEMENT GUIDE", A NORTH AMERICA PHILIPS COMPANY, 1989.

5. WAYNE THOMAS, "ADVANCED ELECTRONIC COMMUNICATIONS SYSTEM", PRENTICE-HALL, 1987.

6. PAUL BATES,P.END , "PRACTICAL DIGITAL AND DATA COMMUNICATION",PRENTICE-HALL,1987.