เครื่องโทรศัพท์ส่งอีเมลล์

ELECTRONIC MAIL BASED ON TELEPHONE

โดย

นายธนบัตร วรธนารัตน์

นายธนาพร ถาวรวัตร์

นางสาวธิภาพร ฤกษนันทน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เครื่องโทรศัพท์ส่งอีเมลล์

ELECTRONIC MAIL BASED ON TELEPHONE

โดย

นายธนบัตร วรธนารัตน์ 41014166

นายธนาพร ถาวรวัตร์ 41014175

นางสาวธิภาพร ฤกษนันทน์ 41014186

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

เรื่อง **เครื่องโทรศัพท์ส่งอีเมลล์**

ELECTRONIC MAIL BASED ON TELEPHONE

ผู้จัดทำ

1. นายธนบัตร วราธนารัตน์ 41014166

2. นายธนาพร ถาวรวัตร์ 41014175

3. นางสาวธิภาพร ฤกษนันทน์ 41014186

.....อาจารย์ที่ปรึกษา

(รศ.ดร. กอบชัย เดชหาญ)

เครื่องโทรศัพท์ส่งอีเมลล์

Electronic Mail Based on Telephone

โดย

นาย ธนบัตร วรธนารัตน์ 41014166

นาย ธนาพร ถาวรวัตร์ 41014175

นางสาว ธิภาพร ฤกษนันทน์ 41014186

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

บทคัดย่อ

เนื่องด้วยในปัจจุบันระบบอินเทอร์เน็ทได้มีการขยายตัวและเริ่มเข้ามามีบทบาทในชีวิตประจำวันของเราเป็นอย่างมาก โดยเฉพาะอย่างยิ่งการติดต่อสื่อสารผ่านทางจดหมายอิเล็กทรอนิกส์ (อีเมลล์) ก็ได้มีผู้นิยมหันมาใช้กันอย่างแพร่หลาย เนื่องจากสามารถส่งข้อมูลได้อย่างรวดเร็ว สะดวกและมีราคาถูก แต่สำหรับผู้ที่ต้องการใช้อีเมลล์นั้น จำเป็นต้องใช้เครื่องคอมพิวเตอร์ซึ่งต่อกับระบบอินเทอร์เน็ท

โครงงาน "เครื่องโทรศัพท์ส่งอีเมลล์" นี้ได้ถูกจัดทำขึ้นมา เนื่องจากเล็งเห็นประโยชน์ของการส่งข้อมูลผ่านทางอีเมลล์ โดยทำการส่งอีเมลล์ผ่านทางอุปกรณ์ของโครงงานนี้ ซึ่งไม่ต้องใช้เครื่องคอมพิวเตอร์ ในโครงงานนี้จะใช้เครื่องโทรศัพท์เชื่อมต่อกับตัวไมโครคอนโทรลเลอร์ และนำข้อมูลส่งผ่านโมเด็มเพื่อเข้าสู่ระบบอินเทอร์เน็ท

Abstract

Nowadays, the growth of Internet system has become influence in our day life, especially, using electronic mail (e-mail) which is widespread used because of the fast speed of transmission, convenience and low cost, but it is necessary to have computer to check e-mail.

This project presents the advantage of sending information by e-mail via this equipment without using the computer. In this project, telephone is connected to microcontroller and the data are transmitted to modem into Internet system.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

ในการติดต่อสื่อสารข้อมูลนั้น เป็นที่แพร่หลายอย่างมากและกลายเป็นสิ่งสำคัญและจำเป็นที่ขาดไม่ได้ในปัจจุบัน เพราะโลกในยุคปัจจุบันได้กลายเป็น โลกของการติดต่อข้อมูลข่าวสารไปแล้วและด้วยการพัฒนาที่ไม่หยุดยั้งในวงการอิเล็กทรอนิกส์ คอมพิวเตอร์และการสื่อสารโทรคมนาคม จึงทำให้เกิดเทคโนโลยีและอุปกรณ์ใหม่ๆ ขึ้นเพื่อตอบสนองต่อความต้องการของมนุษย์ในการสื่อสารถึงกัน แต่สิ่งที่ตามมากับเทคโนโลยีที่สมัยใหม่นั้น ก็คือราคาของอุปกรณ์ที่ใช้กับเทคโนโลยีสมัยใหม่นั้น มักจะมีราคาค่อนข้างสูง อย่างเช่น การติดต่อกันโดยผ่านจดหมายอิเล็กทรอนิกส์ (e-mail) ถึงแม้จะใช้เทคโนโลยีของระบบอินเตอร์เน็ตที่มีค่าใช้จ่ายค่อนข้างถูก แต่ก็จำเป็นต้องใช้อุปกรณ์ในการส่งอีเมลล์ที่มีราคาค่อนข้างสูง นั่นก็คือเครื่องคอมพิวเตอร์สักเครื่อง หรือในเทคโนโลยีปัจจุบันเราอาจใช้โทรศัพท์มือถือในการส่งอีเมลล์ได้แล้ว แต่ราคาของอุปกรณ์และเทคโนโลยีก็ค่อนข้างแพง

ด้วยเล็งเห็นถึงความสำคัญและความจำเป็นของการติดต่อข้อมูลข่าวสารดังที่ได้กล่าวมา และต้องการที่จะให้อุปกรณ์ส่งข้อมูลข่าวสารมีราคาค่อนข้างต่ำ ปริญญานิพนธ์ฉบับนี้จึงได้ทำการศึกษาถึงโปรโตคอลและระบบที่ใช้ในการส่งอีเมลล์บนเครือข่ายอินเตอร์เน็ตโดยผ่านโมเด็ม และรับค่าอินพุตจากแป้นคีย์บอร์ดที่ต่อเพิ่ม ส่วนการแสดงผลนั้นจะทำการแสดงผลเอาท์พุทที่จอแอลซีดี (LCD MODULE)

บทที่ 2
ทฤษฎีและหลักการ

2.1 การใช้งานไมโครคอนโทรลเลอร์ MCS-51

ทฤษฎี MCS-8051

โดยปกติแล้ว เอ็มซีเอส-8051 ก็คือระบบคอมพิวเตอร์ซึ่งมีข้อมูลจากภายนอก และนำมาประมวลผล พร้อมทั้งสามารถส่งสัญญาณเพื่อการควบคุมการทำงานของอุปกรณ์ต่างๆ ได้อย่างดี แล้วในส่วนของการติดต่อสื่อสารข้อมูล (Data Communication) กับระบบภายนอกอื่นๆ ก็สามารถกระทำโดยผ่านพอร์ตอนุกรม (Serial Port) ซึ่งพอร์ตอนุกรมนี้เป็นส่วนที่เหมาะสมในการรับหรือส่งข้อมูลในระยะทางไกลได้ดีกว่าพอร์ตขนาน ซึ่งในการทำโครงงานนี้เน้นเฉพาะการใช้โปรแกรมใช้งานพอร์ตอนุกรมของเอ็มซีเอส-8051

2.1.1 การใช้งานพอร์ตอนุกรม

รีจิสเตอร์ต่างๆ ที่เกี่ยวข้องในการใช้งานพอร์ตอนุกรม

1. รีจิสเตอร์ควบคุมไทม์เมอร์ เนื่องจากการใช้งานพอร์ตอนุกรมนั้นจะมีสิ่งที่ต้องคำนึงถึง คือ อัตราการรับส่งข้อมูล หรือที่เรียกว่าอัตราบอด (Baud Rate) จริงๆ แล้วคือจังหวะการเลื่อนเข้าหรือออกของข้อมูลจากเอ็มซีเอส-8051 นั่นเอง โดยอัตราบอดนี้จะสามารถสร้างได้ภายในของเอ็มซีเอส-8051 ได้จากไทม์เมอร์แชนแนล 1 โดยทำงานในโหมด 2 คือโหลดข้อมูลกลับอัตโนมัติ ดังนั้นรีจีสเตอร์ที่จะต้องทำการโปรแกรมจะมีดังนี้

- TMOD ตำแหน่ง 89 H ทำหน้าที่เลือกโหมดของไทม์เมอร์
- TCON ตำแหน่ง 88 H ทำหน้าที่เริ่มต้นการสร้างอัตราบอด
- TH1 ตำแหน่ง 8C H ทำหน้าที่ใส่ข้อมูลการนับของไทม์เมอร์ 1 เพื่อสร้างอัตราบอดตามต้องการ

2. รีจิสเตอร์ควบคุมการลดกำลัง เนื่องจากการสร้างอัตราบอดนั้นจะต้องนำบิทในรีจีสเตอร์ PCON มาใช้ในการคำนวณข้อมูลของ TH1 ดังนั้น รีจีสเตอร์ที่ใช้คือ

7	6	5	4	3	2	1	0
SMOD	-	-	-	GF1	GF0	PD	IDL

บิท	สัญลักษณ์	รายละเอียด
7	SMOD	บิทที่สำหรับเปลี่ยนแปลง แก้ไขอัตราการส่งข้อมูลแบบอนุกรม โดยใช้โปรแกรมเซ็ตให้เป็น 1 จะเป็นการเพิ่มอัตราการส่งขึ้นเท่าตัวเมื่อใช้ไทม์เมอร์ 1 เข้าช่วย ถ้าเลือกใช้การรับส่งข้อมูลในโหมด 1, 2 , และ 3 แต่ถ้าเคลียร์เป็น 0 จะใช้ไทม์เมอร์ 1 ตามปกติที่เป็นอยู่ตามการกำหนดในช่วงเริ่มต้นของโปรแกรม
6-4	-	ไม่ได้ใช้งาน

3	GF1	แฟล็กใช้งานทั่วไป กำหนดโดยผู้ใช้ ไม่เกี่ยวข้องกับการควบคุมใดๆทั้งสิ้น เซ็ตหรือเคลียร์โดยโปรแกรม
2	GF0	แฟล็กใช้งานทั่วไป กำหนดโดยผู้ใช้เช่นกัน
1	PD	บิทที่แสดงการลดลงของกำลังไฟ (Power Down) เซ็ตเป็น 1 โดยโปรแกรมเพื่อเข้าสู่โหมดของการลดลงของกำลังไฟ ใช้ในไมโครโปรเซสเซอร์ที่เป็นแบบซีมอสเท่านั้น
0	IDL	บิตที่แสดงโหมดไอเดิล (Idle) เซ็ตเป็น 1 โดยโปรแกรมเพื่อเข้าไอเดิลในไมโครโปรเซสเซอร์ที่เป็นแบบซีมอสเท่านั้น

3. รีจีสเตอร์ควบคุมอินเทอร์รัพต์ เนื่องจากว่าเอ็มซีเอส-8051 สามารถใช้งานพอร์ตอนุกรมในลักษณะของการอินเทอร์รัพต์ได้ จึงมีรีจีสเตอร์ที่เกี่ยวข้องดังนี้

- IE ตำแหน่ง A8H ทำหน้าที่ยอมให้เกิดการอินเทอร์รัพต์จากพอร์ตอนุกรมได้หรือไม่ ซึ่งมีรายละเอียดของบิทต่างๆ ดังนี้

7	6	5	4	3	2	1	0
$\overline{EA}$	X	ET2	ES	ET1	EX1	ETO	EXO

บิท	สัญลักษณ์	รายละเอียด
7	$\overline{EA}$	บิทที่ควบคุมทั้ง 6 บิท ถ้าข้อมูลในบิทนี้เป็น 0 จะเป็นการดิสเอเบิลทุกบิททำให้ไม่เกิดการขัดจังหวะ ถ้าบิทนี้เป็น 1 การอีนาเบิล หรือดิสเอเบิล ใน 6 บิทจะขึ้นกับข้อมูลในแต่ละบิทนั้น
6	-	บิทที่ผู้ใช้กำหนดการใช้งานเอง
5	ET2	บิทนี้จะใช้อีนาเบิล หรือดิสเอเบิล สัญญาณขัดจังหวะที่มาจากวงจรไทม์เมอร์ 2 บิทนี้จะใช้อีนาเบิล หรือดิสเอเบิลสัญญาณขัดจังหวะที่มาจากพอร์ตอนุกรม
4	ES	เมื่อมีข้อมูลเข้ามายัง SBUF หรือ ข้อมูลจาก SBUF ได้ถูกส่งออกไปทาง พอร์ตอนุกรม หมดแล้ว
3	ET1	บิทนี้จะใช้อีนาเบิล หรือดิสเอเบิล สัญญาณขัดจังหวะที่มาจากวงจรไทม์เมอร์ 1
2	EX1	บิทนี้ใช้สำหรับการอีนาเบิลสัญญาณที่เข้ามาทางขา INT1 ให้เกิดการขัดจังหวะ
1	ET0	บิทนี้จะใช้อีนาเบิลหรือดิสเอเบิล สัญญาณขัดจังหวะที่มาจากวงจรไทม์เมอร์ 0
0	EX0	บิทนี้ใช้สำหรับการอีนาเบิลสัญญาณที่เข้ามาทางขา INT0 ให้เกิดการขัดจังหวะ

- IP ตำแหน่ง B8 H ทำหน้าที่จัดลำดับความสำคัญของการอินเทอร์รัพต์ ซึ่งมีรายละเอียดของบิทต่างๆ ดังนี้

7	6	5	4	3	2	1	0
X	X	PT2	PS	PT1	PX1	PT0	PX0

บิท	สัญลักษณ์	รายละเอียด
7	PCT	PCT = 1 จะมีเพียงหนึ่งระดับ
6	-	บิทที่ผู้ใช้กำหนดการใช้งานเอง
5	PT2	กำหนดระดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ของไทม์เมอร์ 2
4	PS	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ของพอร์ตอนุกรม
3	PT1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ของไทม์เมอร์ 1
2	PX1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ภายนอกที่ 1
1	PT0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ของไทม์เมอร์ 2
0	PX0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพต์ภายนอกที่ 0

4. รีจิสเตอร์ควบคุมพอร์ตอนุกรม การใช้งานพอร์ตอนุกรมนี้จะขึ้นอยู่กับรีจีสเตอร์นี้โดยตรง คือ

-SBUF ตำแหน่ง 99 H ทำหน้าที่เป็นบัฟเฟอร์การรับหรือการส่ง

-SCON ตำแหน่ง 98 H ทำหน้าที่ควบคุมและกำหนดโหมดการใช้งานพอร์ตอนุกรมทั้งหมดซึ่งมีรายละเอียดของบิทต่างๆ ดังนี้

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

บิท	สัญลักษณ์	รายละเอียด
7	SM0	โหมดการพอร์ตอนุกรมบิท 0 ทำการเซ็ทหรือเคลียร์โดยใช้โปรแกรมสั่งงาน
6	SM1	โหมดการพอร์ตอนุกรมบิท 1 ทำการเซ็ทหรือเคลียร์โดยใช้โปรแกรมสั่งงานเช่นกัน รายละเอียดดังตาราง 2.1
5	SM2	ใช้เป็นการติดต่อระหว่างไมโครโปรเซสเซอร์หลายตัวด้วย (Multiprocessor Communication) เซ็ทและเคลียร์ โดยการใช้โปรแกรมควบคุมในกรณีนี้ควรจะใช้เฉพาะโหมด 2 และ 3 เมื่อบิทนี้ถูกเซ็ตเป็น 1 การอินเทอร์รัพต์จะเกิดขึ้น ถ้าข้อมูลที่รับเข้ามาเป็นบิท 9 เป็น 1 ในทำนองกลับกัน ถ้าข้อมูลที่จะทำการรับเข้ามาเป็น 0 การอินเทอร์รัพต์จะเกิดขึ้นเมื่อได้รับบิตหยุดที่ถูกต้องแล้วเท่านั้น แต่ถ้าใช้ในโหมด 0 บิตนี้จะถูกเคลียร์ให้เป็น 0
4	REN	บิทอีนาเบิล การรับบิทนี้จะถูกเซ็ทเป็น 1 เมื่อต้องการรับสัญญาณอนุกรมและจะต้องถูกเคลียร์ให้เป็น 0 เมื่อไม่ต้องการรับสัญญาณอนุกรม
3	TB8	ใช้เลือกว่า จะให้ส่ง 8 บิทหรือไม่ และจะถูกเซ็ทหรือเคลียร์โดยโปรแกรมโหมด 2 หรือ 3
2	RB8	ใช้เลือกว่าจะรับบิท 8 หรือไม่ ใช้สำหรับกรณีรับข้อมูลโหมด 2 หรือ 3 หยุดในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้งานบิทนี้
1	TI	แฟลกอินเทอร์รัพต์เมื่อส่งข้อมูลในโหมด 0 จะถูกเซ็ทให้เป็น 1 หลังจากที่ส่งบิท 7 ออกไปแล้วในโหมดอื่นๆ จะถูกเซ็ทให้เป็น 1 เมื่อบิทหยุดเริ่มส่งออกไป การเคลียร์บิทนี้ จะทำได้โดยใช้โปรแกรม
0	RI	แฟลกอินเตอร์รัพต์เมื่อรับข้อมูลในโหมด 0 จะถูกเซ็ทให้เป็น 1 หลักจากรับบิท 7 เรียบร้อยแล้วในโหมดอื่นๆ จะถูกเซ็ทเป็น 1 เมื่อรับบิทหยุดได้ครึ่งหนึ่ง การเคลียร์บิทนี้ จะทำได้โดยการใช้โปรแกรม

การอ้างอิงแบบบิทแอดเดรสของรีจิสเตอร์นี้คือ SCON.0 ถึง SCON.7

SM0	SM1	โหมด	รายละเอียด
0	0	0	รีจิสเตอร์แบบเลื่อนบิท อัตราการส่ง = f / 12
0	1	1	UART ชนิด 8 บิท อัตราการส่ง = เปลี่ยนแปลงได้
1	0	2	UART ชนิด 9 บิท อัตราการส่ง = f / 32 หรือ f / 64
1	1	3	UART ชนิด 9 บิท อัตราการส่ง = เปลี่ยนแปลงได้

ตารางที่ 2.1 แสดงโหมดการทำงานของพอร์ตอนุกรม

การคำนวณอัตราการบอดโหมด 1

ไทม์เมอร์ 1 จะถูกใช้เป็นตัวสร้างอัตราการบอด เมื่อกำหนดการรับส่งข้อมูลแบบอนุกรมในโหมด 1 โดยไทม์เมอร์จะถูกใช้ให้ทำงานในโหมด 2 ซึ่งจะโหลดค่าเข้าไปโดยอัตโนมัติ การคำนวณจะเป็นดังนี้

$$f_{buad} = \frac{2^{SMOD} \text{x Oscilator frequency}}{32 \text{ x } 12\text{x}[256\text{d} - (\text{TH1})]} = \text{ความถี่ของออสซิลเลเตอร์}$$

เมื่อ SMOD จะเป็นบิตควบคุมอยู่ในรีจีสเตอร์ PCON ซึ่งจะเป็น 0 หรือ 1 ก็ได้ ถ้าเป็น 0 จะเป็นความถี่ปกติ ถ้าเป็น 1 ความถี่จะเพิ่มขึ้นเป็น 2 เท่า ดังได้กล่าวมาแล้วในรีจีสเตอร์ PCON

ถ้าไทม์เมอร์ 1 ไม่ถูกใช้งานในโหมด 2 ค่าอัตราบอดจะเป็น

$$f_{buad} = \frac{2^{SMOD} \text{x (timer 1 overflow frequency)}}{32}$$

ถ้าเลือกใช้อัตราบอดมาตรฐานควรจะเลือกความถี่ของคริสตอลให้ถูกต้องคือ 11.0592 เมกกะเฮิรตซ์

การกำหนดเริ่มต้นและการใช้งานรีจีสเตอร์ของพอร์ทอนุกรม

บิทต่างๆ ของ SCON

- บิท SM0 (บิตที่7) และบิต SM1 (บิตที่6) เป็นการเลือกโหมด การทำงานพอร์ท อนุกรมสามารถเลือกได้ 4 โหมด

- บิท SM2 (บิตที่5) บิตนี้ถ้าเซตเป็น " 1 " จะใช้งานในโหมดโปรเซสเซอร์หลายตัว (Multiprocessor) ดังนั้นการใช้งานโหมด 1 มิใช้เป็นการติดต่อโปรเซสเซอร์หลายตัว จึงให้บิทนี้เป็น " 0 "

- บิท REN (บิตที่ 4)

ถ้าเซตเป็น " 1 " หมายถึงเอ็มซีเอส-51 สามารถรับข้อมูลจากพอร์ทอนุกรมได้

ถ้าเคลียร์เป็น " 0 " หมายถึงเอ็มซีเอส-51 ไม่สามารถรับข้อมูลจากพอร์ทอนุกรมได้

- บิท TB8 (บิตที่3) จะใช้งานในการติดต่อข้อมูลโหมด 2 และโหมด 3

- บิท RB8 (บิตที่2) จะใช้งานในการติดต่อข้อมูลโหมด 2 และโหมด 3

- บิท TI (บิตที่1) จะเซตเมื่อทุกบิทของข้อมูลส่งออกไปแล้ว เพื่อเป็นการแสดงว่าบัฟเฟอร์ของการส่งออกข้อมูลว่าง ถ้าซอฟต์แวร์ต้องการส่งข้อมูลออกไปยังอุปกรณ์ที่ต่ออยู่กับพอร์ทอนุกรม โปรแกรมจะต้องตรวจสอบว่าพอร์ทอนุกรมว่างหรือไม่ (ซึ่งก็คือตรวจสอบว่าตัวอักษรที่แล้วส่งออกไปแล้วหรือยัง) ถ้าว่างก็ส่งข้อมูลต่อไป

- บิต RI (บิตที่ 0) จะเซ็ตเมื่อเสร็จสิ้นการรับข้อมูลแล้ว เพื่อเป็นการแสดงว่าบัฟเฟอร์ของการรับข้อมูลเต็มข้อกำหนดนี้จะตรวจสอบด้วยซอฟต์แวร์ หรือโปรแกรมให้เกิดการอินเตอรัพท์ก็ได้

2.1.2 การใช้งานไทม์เมอร์ / เคาน์เตอร์

การใช้งานไทม์เมอร์ / เคาน์เตอร์ภายในของเอ็มซีเอส-8051 จะต้องมีการกำหนดรูปแบบการใช้งานต่างๆ เสียก่อนจึงจะทำงานได้ถูกต้อง การกำหนดค่าเริ่มต้นและการอ่านค่าจากไทม์เมอร์รีจีสเตอร์ เพราะฉะนั้นการจะเริ่มต้นใช้งานไทม์เมอร์ได้จะต้องมีการกำหนดค่าในรีจีสเตอร์ ซึ่งรีจีสเตอร์ที่จะกำหนดจะมี ดังนี้

1. รีจีสเตอร์ TMOD เป็นรีจีสเตอร์ขนาด 8 บิต อยู่ตำแหน่งที่ 89H ไม่สามารถอ้างตำแหน่งแบบบิตได้ รายละเอียดของการกำหนดค่าบิตต่างๆ จะเป็นดังนี้

7	6	5	4	3	2	1	0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Timer 1				Time 0			

บิท	สัญลักษณ์	รายละเอียด
7/3	GATE	การอีนาเบิล OR เกท เพื่อควบคุมการทำงานของไทม์เมอร์ GATE = 1 ไทม์เมอร์จะทำงาน ถ้าสัญญาณขา INTn เป็นลอจิก 1 และบิต Trn ในรีจีสเตอร์ TCON เป็น 1 ด้วย
6/2	C/T	ถ้าบิต C/T = 0 เป็นการเลือกการทำงานแบบนับพัลส์จากภายในตัวเอ็มซีเอส-51 ถ้าบิต C/T = 0 เป็นการเลือกการทำงานแบบนับพัลส์จากภายนอกที่ป้อนเข้าทางขา P3.5 (T1) หรือ P3.4 (T0) (COUNTER)
5/1	M1	ใช้ในการเลือกโหมดการทำงานดังตารางที่ 2.2
4/0	M0	ใช้ในการเลือกโหมดการทำงานดังตารางที่ 2.2

M1	M0	โหมดที่ใช้งาน	ลักษณะการทำงาน
0	0	0	ไทม์เมอร์ขนาด 13 บิต
0	1	1	ไทม์เมอร์ขนาด 16 บิต
1	0	2	ไทม์เมอร์ขนาด 8 บิต แบบโหลดค่ากลับอัตโนมัติ
1	1	3	ไทม์เมอร์แบบใช้งานอิสระ

ตารางที่ 2.2 แสดงโหมดการทำงานของไทม์เมอร์

2. รีจิสเตอร์ TH0 , TL0 และTH1 , TL1 รีจีสเตอรเหล่านี้เป็นรีจีสเตอร์ขนาด 8 บิต ใช้กำหนดค่าของไทม์เมอร์ ตำแหน่งของ TH0 , TL0 จะอยู่ที่ 8CH , 8AH และตำแหน่ง TH1 , TL1 จะอยู่ที่ 8DH , 8BH และเนื่องจากว่า ไทม์เมอร์ภายในนี้เป็นไทม์เมอร์แบบนับขึ้นจนถึงค่าสูงสุด คือ FFH (ในกรณีที่เลือกการทำงานแบบ 8 บิต) , FFFFH (ในกรณีที่เลือกการทำงานแบบ 16 บิต) แล้วจะเกิดโอเวอร์โฟลว์ (เปลี่ยนจาก FFH เป็น

FFFFH เป็นค่า 0) ดังนั้นการกำหนดค่าการนับจะต้องนำค่าสูงสุดของโหมดการทำงานนั้นๆ มาลบออกด้วยค่าที่ต้องการ

3. รีจีสเตอร์ TCON เป็นรีจีสเตอร์ขนาด 8 บิท อยู่ที่ตำแหน่งที่ 88H สามารถอ้างอิงตำแหน่งแบบบิทได้ ใช้งานในการควบคุมการทำงานของไทม์เมอร์ และ ควบคุมการทำงานอินเตอร์รัพต์ด้วยรายละเอียดที่เกี่ยวกับบิทต่าง ๆ เป็นดังนี้

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

บิท	สัญลักษณ์	รายละเอียด
7	TF1	บิทโอเวอร์โฟลว์ของไทม์เมอร์ 1 เมื่อไทม์เมอร์ 1 นับค่าจนเปลี่ยนจากค่าสูงสุด (FFH , FFFFH) เป็น 0 บิทนี้จะถูกเซ็ตค่าเป็น 1 และบิทนี้จะเป็น 0 เมื่อกลับจากการทำงานในโปรแกรมบริการอินเทอร์รัพต์ (Interrupt Service Routine) หรือ จะใช้การเคลียร์ค่าจากซอฟท์แวร์
6	TR1	บิทควบคุมการเริ่มทำงาน หรือ หยุดการทำงาน ไทม์เมอร์ 1 กล่าวคือ ถ้าบิท TR1 = 1 หมายถึง เริ่มต้นการทำงานไทม์เมอร์ 1 TR1 = 0 หมายถึง หยุดการทำงานไทม์เมอร์ 1 หมายเหตุ:การหยุดการทำงานนี้ไม่ได้หมายความว่าค่าภายในจะถูกเคลียร์
5	TF0	ความหมาย และ การใช้งานเหมือนกับ TF1 แต่เป็นไทม์เมอร์ 0
4	TR0	ความหมาย และ การใช้งานเหมือนกับ TR1 แต่เป็นไทม์เมอร์ 0
3	IE1	บิทของแฟล็กจากการอินเทอร์รัพต์ภายนอกจากขา P3.3($\overline{INT1}$)
2	IT1	บิทนี้ใช้สำหรับ เลือกวิธีการอินเทอร์รัพต์ภายนอกจาก ขา P3.3 ($\overline{INT1}$) ถ้าบิท IT1 = 0 เป็นการเลือกการเกิดอินเทอร์รัพต์จากระดับลอจิก 0 IT1 = 1 เป็นการเลือกการเกิดอินเทอร์รัพต์จากขอบขาลง
1	IE0	การใช้งานเหมือนกับ IE1 แต่เป็นการอินเทอร์รัพต์ภายนอกจากขา P3.2 ($\overline{INT0}$)
0	IT0	การใช้งานเหมือนกับ IT1 แต่เป็นการอินเทอร์รัพต์ภายนอกจากขา P3.2 ($\overline{INT0}$)

2.1.3 การจัดวางหน่วยความจำ

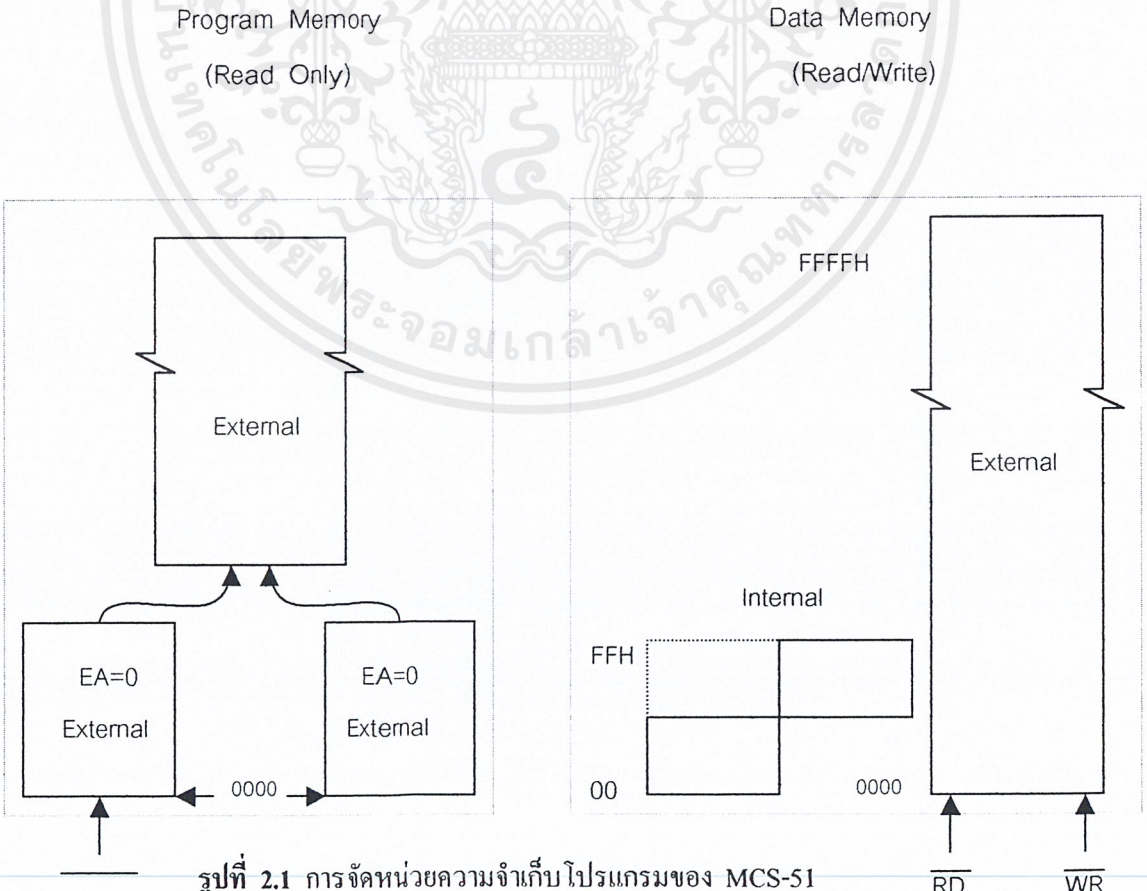
หน่วยความจำภายในของMCS-51 ประกอบด้วยหน่วยความจำเก็บข้อมูลที่เป็นรอม(8051/8052)หรืออีพรอม(ในกรณีของ 8751/8752)และหน่วยความจำแรม สำหรับหน่วยความจำแรมภายในถูกนำมาใช้หลายหน้าที่ด้วยกันได้แก่ใช้เก็บข้อมูลทั่วไป(ไบต์)เก็บข้อมูลในลักษณะของบิต ทำเป็นรีจิสเตอร์แบงก์ และใช้เป็นรีจิสเตอร์ที่ทำหน้าที่พิเศษเฉพาะงาน โดยมีข้อควรจำอยู่สองประการคือ

1.รีจิสเตอร์และอินพุต/เอาต์พุตพอร์ต ใช้แมปในลักษณะเดียวเดียวกับหน่วยความจำ(memory-mapped)ดังนั้นการเรียกใช้งานอาจใช้การเรียกเช่นเดียวกับตำแหน่งของหน่วยความจำก็ได้

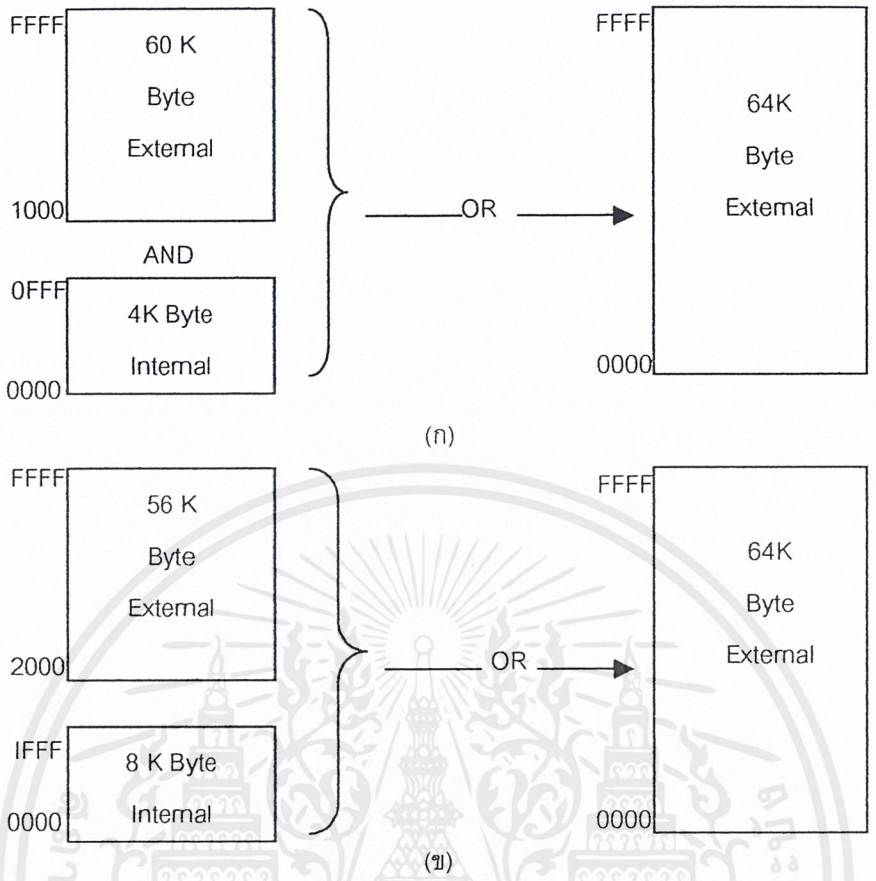
2.รีจิสเตอร์จะเป็นส่วนหนึ่งของหน่วยความจำแรม ดังนั้นการใช้งานบริเวณนี้จะใช้ในลักษณะรีจิสเตอร์หรือหน่วยความจำแรมก็ได้

- หน่วยความจำเก็บโปรแกรม (program memory)

ในกรณีที่ใช้เบอร์ที่มีหน่วยความจำเก็บโปรแกรมภายในแต่ออกแบบให้มีหน่วยความจำภายนอกอยู่ด้วย ผู้ใช้อาจจะเลือกใช้หน่วยความจำภายในร่วมกับภายนอกหรือเลือกไม่ใช้ส่วนภายในเลยก็ได้ โดยการควบคุมผ่านขา EA ของไมโครคอนโทรลเลอร์ ซึ่งการเลือกใช้งานจะเริ่มเมื่อมีการรีเซ็ทไมโครคอนโทรลเลอร์ ซีพียูภายในจะอ่านสถานะของขา EA ว่าเป็นอะไร ถ้าเป็นลอจิก “1” การเริ่มต้นเฟทช์คำสั่งจะเริ่มที่ตำแหน่ง 0000 ของหน่วยความจำเก็บโปรแกรมภายใน แต่ถ้าเป็นลอจิก “0” ก็จะเริ่มจากตำแหน่ง 0000 ของหน่วยความจำภายนอกเลย



รูปที่ 2.1 การจัดหน่วยความจำเก็บโปรแกรมของ MCS-51

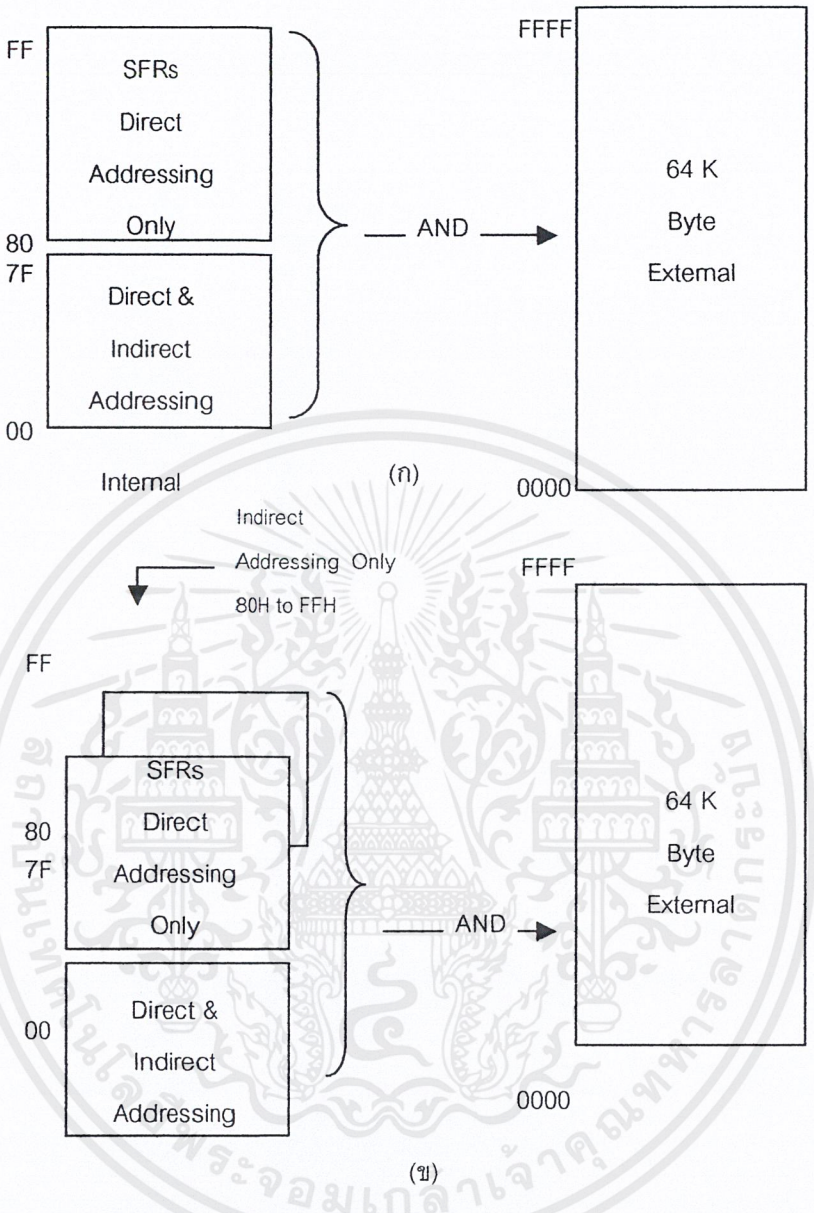


รูปที่ 2.2 การจัดหน่วยความจำเก็บโปรแกรมของ MCS-51

สิ่งที่ควรคำนึงในการออกแบบวงจรไมโครคอนโทรลเลอร์คือควรจะให้ใช้เฉพาะหน่วยความจำภายในทั้งที่เป็นหน่วยความจำเก็บโปรแกรมและหน่วยความจำเก็บข้อมูล เพราะเมื่อออกแบบให้ใช้หน่วยความจำภายนอกแล้วไมโครคอนโทรลเลอร์จะต้องสูญเสียพอร์ตไปเป็นจำนวนมาก เพื่อที่จะใช้ในการติดต่อระหว่างซีพียูกับหน่วยความจำอย่างในตระกูล MCS-51 จะต้องใช้พอร์ต 0 และพอร์ต 2 ไปรวมทั้งหมด 16 เส้น และถ้าเป็นการใช้หน่วยความจำเก็บข้อมูลภายนอกด้วยแล้วจะต้องเสียพอร์ต 3.6 และพอร์ต 3.7 ไปเป็นสายสัญญาณ WR และ RD อีก 2 เส้น

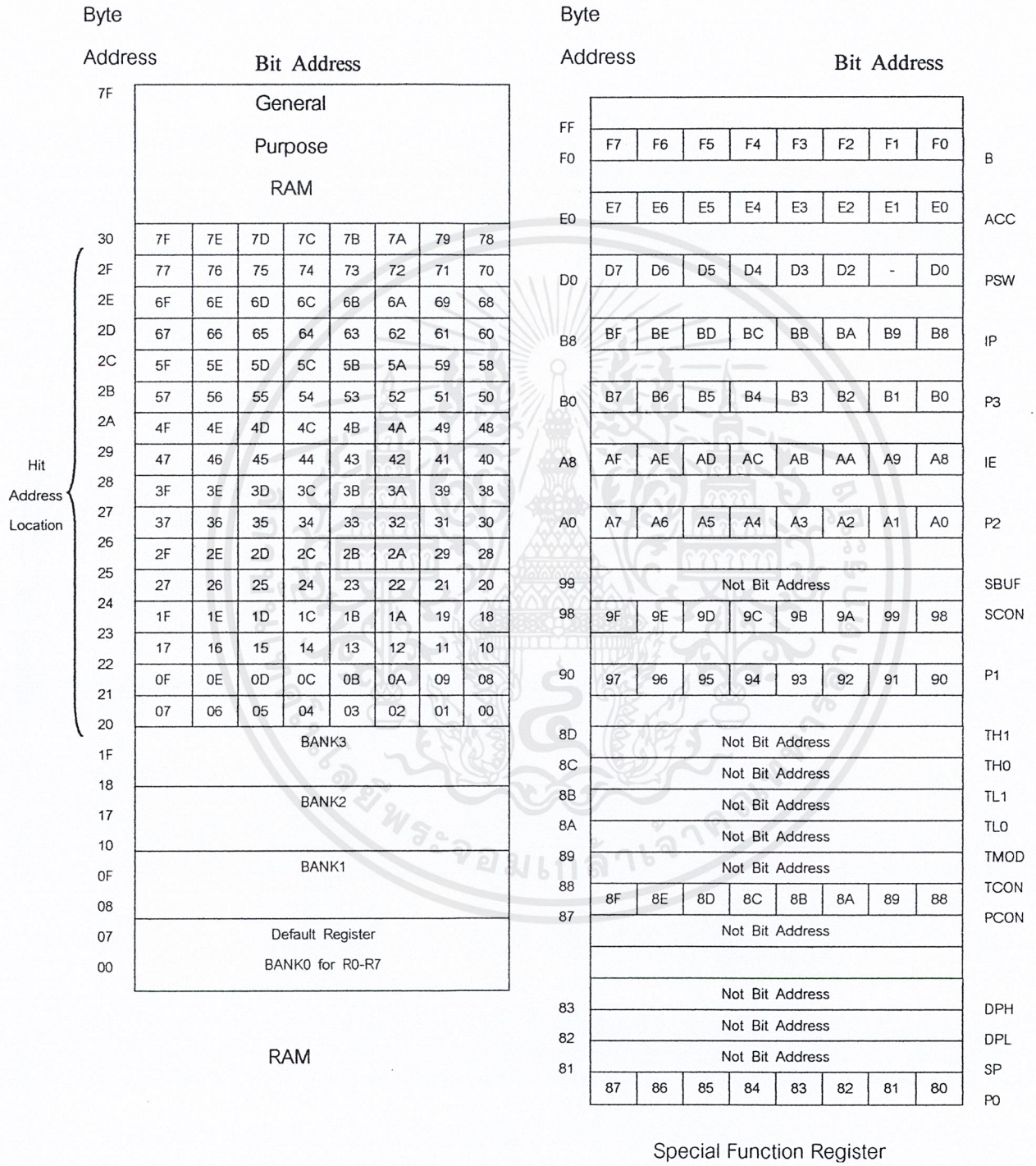
- หน่วยความจำเก็บข้อมูลภายใน(internal data memory)

หน่วยความจำเก็บข้อมูลของMCS-51นี้แบ่งแยกระหว่างภายในกับภายนอกโดยเด็ดขาด คำสั่งที่ใช้เข้าถึงข้อมูลก็ใช้ต่างกันจากรูปที่ 2.7 สำหรับเบอร์ 8051 มีหน่วยความจำภายในที่ใช้เก็บข้อมูลทั่วๆไป ซึ่งต่อจากนี้ไปจะเรียกว่าหน่วยความจำแรมภายใน มีอยู่ 128 ไบต์รวมกับส่วนที่เป็นรีจิสเตอร์ที่ใช้งานในหน้าที่พิเศษ (special function register,SFR)ซึ่งต่อไปจะเรียกว่า SFR อยู่อีก 128 ไบต์ รวมทั้งหมดเป็น 256 ไบต์ เฉพาะหน่วยความจำแรมภายในสามารถเข้าถึงด้วยการอ้างตำแหน่งได้ทั้งแบบโดยตรง (direct)และโดยอ้อม(indirect) ส่วน SFR สามารถเข้าถึงด้วยการอ้างตำแหน่งแบบโดยตรงเท่านั้น



รูปที่ 2.3 หน่วยความจำแรมภายใน MCS-51

ต่อไปนี้จะกล่าวเฉพาะหน่วยความจำแรมภายในตั้งแต่ตำแหน่ง 00H ถึง FFH รวมเป็น 256 ไบต์ ซึ่งเป็นหน่วยความจำพื้นฐานที่ทุกเบอร์มีเหมือนกัน ซึ่งแยกการใช้งานออกเป็น 3 ส่วนดังรูปที่ 2.4



รูปที่ 2.4 หน่วยความจำแรมภายในตั้งแต่ตำแหน่ง 00H ถึง FFH

1.รีจิสเตอร์แบงก์ แรมภายในจำนวน 32 ไบต์จากตำแหน่ง 00H ถึง 1FH ถูกนำมาเป็นรีจิสเตอร์ใช้งานทั่วไปโดยแยกออกเป็น 4 แบงก์ๆละ 8 รีจิสเตอร์ ในเวลาหนึ่งจะมีเฉพาะแบงก์เดียวเท่านั้นที่ถูกเลือกใช้งาน รีจิสเตอร์แต่ละแบงค์มีหมายกำกับเป็นแบงก์ 0 ถึงแบงก์ 3 และรีจิสเตอร์ภายในแต่ละแบงก์จะถูกกำหนดชื่อไว้เหมือนกันหมด คือ R0 ถึง R7 ดังนั้น รีจิสเตอร์แต่ละตัวจะอ้างตำแหน่งโดยใช้ชื่อ (เมื่อแบงก์ใดแบงก์หนึ่งถูกเลือกใช้) หรืออ้างเป็นตำแหน่งของหน่วยความจำแรมเลยก็ได้

การเลือกรีจิสเตอร์แบงก์ใดทำงาน สามารถกำหนดโดยเซ็ตหรือเคลียร์บิต RS0 และ RS1 ในรีจิสเตอร์ PSW โดยใช้คำสั่งทางซอฟท์แวร์ และรีจิสเตอร์แบงก์ใดที่ไม่ถูกเลือกใช้งานสามารถใช้เป็นหน่วยความจำแรมแบบปกติได้ หลังจากทำการรีเซ็ตไมโครคอนโทรลเลอร์ รีจิสเตอร์แบงก์ 0 จะถูกเลือกใช้งานโดยอัตโนมัติ

2.หน่วยความจำแรมที่สามารถอ้างอิงตำแหน่งแบบบิต หน่วยความจำที่อยู่เหนือจากรีจิสเตอร์แบงก์ขึ้นไปมีทั้งหมด 16 ไบต์ คือ ตั้งแต่ตำแหน่ง 20H ถึง 2FH (หนึ่งตำแหน่งคือหนึ่งไบต์) สามารถอ้างตำแหน่งได้ 2 แบบ แบบแรกที่อ้างแบบไบต์จะอ้างได้ 16 ตำแหน่ง แบบที่สองอ้างแบบบิตจะได้ทั้งหมด 128 ตำแหน่ง คือตำแหน่ง 00H ถึง 7FH (8 บิต x 16 ไบต์ = 128 บิท)

การอ้างตำแหน่งแบบบิตอาศัยแนวความคิดจากความคล่องตัวและประสิทธิภาพการใช้งานของไมโครคอนโทรลเลอร์ โดยที่สามารถเซ็ตหรือเคลียร์บิตใดบิตหนึ่งหรือทำการออร์บิตใดบิตหนึ่งได้ ภายในหนึ่งคำสั่ง ยิ่งไปกว่านั้นอินพุต / เอาต์พุตพอร์ตของ MCS – 51 สามารอ้างแบบบิตได้ จึงทำให้การเขียนซอฟต์แวร์ทำได้ง่ายขึ้น การควบคุมและตรวจสอบพอร์ต (ซึ่งก็คือการควบคุมและตรวจสอบอุปกรณ์ ภายนอกตัว CPU นั่นเอง) จะสามารถทำได้อย่างรวดเร็ว

3. หน่วยความจำแรมที่ใช้งานทั่วไป ส่วนนี้จะอยู่เหนือบริเวณหน่วยความจำแรมที่อ้างอิงแบบบิตขึ้นไปโดยเริ่มตั้งแต่ตำแหน่ง 30H จนกระทั่งถึง 7FH รวมทั้งสิ้น 80 ไบต์ ตำแหน่งเหล่านี้ถูกอ้างอิงแบบไบต์เท่านั้นการอ้างอิงจะใช้แบบกำหนดตำแหน่งโดยตรง

- รีจิสเตอร์ที่ใช้งานในหน้าที่พิเศษ (Special Function Registers ; SFR)

รีจิสเตอร์ภายในของ MCS-51 ถือเป็นส่วนหนึ่งของหน่วยความจำแรมภายในชิป ดังนั้นแต่ละรีจิสเตอร์จะมีตำแหน่งที่แน่นอนดังเช่น รีจิสเตอร์ R0-R7 ที่กล่าวผ่านมาแล้วข้างต้น ใน MCS-51 จะมีรีจิสเตอร์สำหรับทำหน้าที่พิเศษ (SFR) อยู่ส่วนบนสุดของหน่วยความจำแรมภายใน (รูปที่ 2.8) โดยมีตำแหน่งจาก 80H ถึง FFH แต่ทั้ง 128 ตำแหน่งนี้ไม่ได้ใช้ทั้งหมด แต่ใช้เพียง 21 ตำแหน่งเท่านั้น ในส่วนนี้เว้นว่างจะใช้เป็นหน่วยความจำแรมไม่ได้ (ข้อมูลที่อ่านออกมาจะไม่ยืนยัน ว่าถูกต้องเหมือนกับที่เขียนลงไป)

2.2 การสื่อสารแบบซิงโครนัสและอะซิงโครนัส

เมื่อข้อมูลถูกแปลงให้เป็นแบบอนุกรมแล้ว มีวิธีในการส่งข้อมูลอยู่สองแบบคือ ซิงโครนัส (Synchronous) และอะซิงโครนัส (Asynchronous)

เมื่อข้อมูลถูกส่งมาจากการพิมพ์ที่แป้นพิมพ์ การส่งและรับจะเป็นแบบอะซิงโครนัส คือคนที่พิมพ์ไม่สามารถที่จะพิมพ์ได้อย่างต่อเนื่อง ดังนั้นเมื่อคอมพิวเตอร์รับตัวอักษรแต่ละตัวจะมีช่องว่างระหว่างตัวอักษรที่ไม่สม่ำเสมอ ทำให้อุปกรณ์ฝ่ายรับไม่อาจคาดหมายได้ว่า ตัวอักษรต่อไปจะมาถึงเมื่อใด จากการขาดความต่อเนื่องนี้จึงมีความจำเป็นต้องใส่บิทพิเศษก่อนและหลังตัวอักษรแต่ละตัวเพื่อบ่งบอกจุดเริ่มต้นและสิ้นสุดของตัวอักษรบิทพิเศษนี้เรียกว่า บิทเริ่มต้น (Start bit) นอกจากนี้ยังมีอีกบิทหนึ่งคือ บิทพาริตี้ (Parity bit) ที่มักจะถูกใส่เพิ่มเข้าไปเพื่อใช้ตรวจสอบความผิดพลาด วิธีนี้เรียกว่า การสื่อสารแบบอะซิงโครนัส (Asynchronous communication)

เมื่อตัวอักษรถูกส่งไปเป็นกลุ่มตามความเร็วของเครื่อง ช่วงห่างระหว่างกันก็จะสม่ำเสมอจึงไม่มีความจำเป็นต้องมีบิทเริ่มต้นและบิทจบสำหรับตัวอักษรแต่ละตัวเพราะว่าเมื่อตัวอักษรแรกถูกรับไป อุปกรณ์ฝ่ายรับสามารถคาดหมายการมาถึงของตัวอักษรถัดไปได้ กล่าวอีกนัยหนึ่งคือ มันสามารถเข้าจังหวะตัวมันเองกับคอมพิวเตอร์ฝ่ายส่งได้ วิธีนี้เรียกว่า การสื่อสารแบบซิงโครนัส (Synchronous communication)

เนื่องจากการสื่อสารแบบอะซิงโครนัสต้องการบิทเริ่มต้นและบิทจบเพิ่มเข้าไปในแต่ละตัวอักษร จึงมีความยาวในการส่งไฟล์มากกว่าการสื่อสารแบบซิงโครนัส ประมาณ 20 เปอร์เซ็นต์ ความแตกต่างนี้อาจสังเกตไม่เห็นเมื่อแหล่งข้อมูลที่ส่งมาจากการพิมพ์ที่เทอร์มินอล

นอกจากในโลกของไอบีเอ็มแมนเฟรมซึ่งเทอร์มินอลแบบซิงโครนัสเป็นอุปกรณ์สามัญ การสื่อสารแบบอนุกรมส่วนใหญ่เกิดขึ้นในแบบอะซิงโครนัสซึ่งประยุกต์เข้ากับการสื่อสารเกือบทั้งหมดระหว่างไมโครคอมพิวเตอร์เทอร์มินอลและระบบยูนิกซ์ด้วยเหตุนี้ส่วนที่เหลือของบทนี้จะให้ความสนใจที่การสื่อสารแบบอะซิงโครนัสเป็นหลัก

2.3 คำสั่งควบคุมการทำงานของโมเด็ม

สำหรับโมเด็มทั่ว ๆ ไปโดยใช้มาตรฐานชุดคำสั่ง AT นี้ โดยจะรวมทั้ง การตอบสนองคำสั่งของโมเด็ม (Result Code) การกำหนดค่ารีจิสเตอร์ s ต่าง ๆ (S-register) และการทำงานของรีจิสเตอร์เหล่านี้ ทั้งนี้จะอ้างอิงจากโมเด็มของ Hayes รุ่นที่มีความเร็ว 9600 BPS เป็นหลัก ซึ่งคำสั่งส่วนใหญ่จะใช้ได้กับโมเด็มของ Hayes รุ่นที่มีความเร็วต่ำกว่าได้ อย่างเช่น รุ่นความเร็ว 2400, 1200 และ 300 BPS และยังสามารถใช้ได้กับโมเด็มของบริษัทอื่นที่มีความสามารถเทียบเท่ากับโมเด็มของ Hayes อีกด้วย

2.3.1 ขั้นตอนการโทรศัพท์ออก

เมื่อผู้ใช้คำสั่งให้โมเด็มเริ่มทำการโทรศัพท์ไปยังโมเด็มปลายทาง กระบวนการต่างๆ ระหว่างโปรแกรมสื่อสารและโมเด็มก็จะเริ่มขึ้น ถ้าหากผู้ใช้เคยใช้งานโมเด็มมาก่อนแล้ว ก็คงจะคุ้นเคยกับเสียงต่างๆ ที่ดังออกมาจากลำโพงของโมเด็ม เช่น เสียงไดอัลโทน (Dial Tone) ตามด้วยเสียง DTMF หรือเสียง pulse ในขณะที่ทำการหมุนหมายเลขโทรศัพท์ปลายทาง เสียง Ringing Tone และเสียงสัญญาณพาหะโต้ตอบกัน

ระหว่างโมเด็มต้นทางและปลายทาง หลังจากนั้นโมเด็มก็จะเงียบลงพร้อมกับเข้าสู่การเชื่อมต่อ (Connection) ซึ่งกระบวนการดังกล่าวนี้นับได้ว่าเป็นกระบวนการทำงานปกติที่ผู้ใช้คุ้นเคยเป็นอย่างดี ตารางที่2.3 จะแสดงแผนผังลำดับขั้นตอนการทำงาน โดยเริ่มตั้งแต่ขั้นตอนที่คำสั่งให้โมเด็มหมุนหมายเลขโทรศัพท์จนกระทั่งถึงการวางหูโทรศัพท์

ขั้นตอน	ผู้ใช้	ซอฟต์แวร์	โมเด็มต้นทาง	โมเด็มปลายทาง
1	เลือกคำสั่ง "Dial" จากซอฟต์แวร์	เปิดสัญญาณ DTR เพื่อส่งคำสั่งหมุนหายเลขไปยังโมเด็ม โดยใช้คำสั่ง : ATDT 555-1234	เปิดลำโพงยกหูโทรศัพท์ รอสัญญาณให้หมุนหมายเลข (Dial Tone) จากนั้นก็หมุนโทรศัพท์ตามหมายเลขดังกล่าว	
2		รอ result codes จากโมเด็ม	รอการตอบรับจากปลายทาง ทั้งนี้ระยะเวลาการรอขึ้นกับการกำหนดค่ารีจีสเตอร์	
3				เสียงโทรศัพท์ดัง
4				ตอบรับสัญญาณ
5			รับสัญญาณตอบรับและทำการส่งสัญญาณ Originate Carrier	
6			รับทราบวิธีการดีมอดูเลชันและความเร็วของแต่ละฝ่าย	รับทราบวิธีการมอดูเลชันและความเร็วของแต่ละฝ่าย
7			โมเด็มตกลงรับรู้โปรโตคอลการควบคุมความผิดพลาดและการบีบอัดข้อมูลของแต่ละฝ่าย	
8			ส่ง result code "CONNECT" ไปยัง PC ปิดลำโพงและเปิดสัญญาณ CD	

9		รับรู้ result code และสัญญาณ CD รายงานให้ผู้ใช้ทราบว่าการติดต่อได้เกิดขึ้นแล้ว		
10	เริ่มการติดต่อกับโฮสคอมพิวเตอร์	ดำเนินการสื่อสารและคอยดูสัญญาณที่ขาดหายไปจากจอสัญญาณ CD	ส่งและรับข้อมูล	ส่งและรับข้อมูล
11	การสื่อสารเสร็จสมบูรณ์ เลือกคำสั่ง "Disconnect" หรือ Hang Up	ปิดสัญญาณ DTR หรือส่ง +++ ตามด้วยคำสั่ง ATH		
12			วางสายโทรศัพท์	ยกเลิกสัญญาณ วางสายโทรศัพท์

ตารางที่ 2.3 กระบวนการทำงานของโมเด็มต้นทางและปลายทางตั้งแต่ขั้นตอนที่ 1 ถึงขั้นตอนที่ 12

จากตารางที่ 2.3 ผู้ใช้สามารถสังเกตได้ว่าสิ่งที่ทำมากที่สุดในกระบวนการสื่อสารก็คือโมเด็ม ส่วนโปรแกรมสื่อสารจะมีหน้าที่เพียงส่งชุดคำสั่ง AT ไปให้กับโมเด็มเท่านั้น การกำหนดเวลาต่างๆ ในกระบวนการนี้จะขึ้นอยู่กับข้อมูลที่อยู่ในรีจีสเตอร์ S ด้วย ตัวอย่างรีจีสเตอร์ S7 จะเก็บค่าของเวลาที่โมเด็มจะรอเสียงตอบมาจากโมเด็มปลายทางเป็นต้น

2.3.2 การใช้งานชุดคำสั่ง AT

ขั้นตอนการเชื่อมต่อระหว่างโมเด็มต้นทาง และปลายทาง ที่ได้กล่าวไปแล้วนั้น โมเด็มทั้งคู่จำเป็นจะต้องมีพื้นฐานที่เหมือนกันบางประการ เช่น ความเร็วรูปแบบของข้อมูล และโปรโตคอลในการรับส่งข้อมูล โดยการเชื่อมต่อโมเด็มในแต่ละครั้ง อย่างน้อยที่สุด ผู้ที่ใช้โมเด็มต้นทางจำเป็นต้องทราบว่าจะให้โมเด็มโทรศัพท์ออกด้วยหมายเลขใด ควรจะตั้งเวลาในการรอเสียงตอบจากโมเด็มปลายทางนานเท่าไร ระบบโทรศัพท์ของตนนั้นใช้ระบบ Tone หรือ Pulse และในขณะเดียวกันผู้ที่ใช้โมเด็มปลายทางก็จะต้อง กำหนดให้โมเด็มรับทราบด้วยว่าจะให้โมเด็มรับสายโทรศัพท์เข้าหลังจากที่มีเสียง Ringing กี่ครั้ง และจะต้องรอสัญญาณโต้ตอบนานเท่าไรเป็นต้น ซึ่งการดำเนินงานทั้งหมดนี้จะสามารถควบคุมได้โดยใช้ชุดคำสั่ง AT

<u>ประเภทของชุดคำสั่ง AT</u>

โดยพื้นฐานแล้ว เราสามารถจะแบ่งชุดคำสั่ง AT ออกเป็น 2 ประเภทใหญ่ๆ ได้ดังนี้ ประเภทแรกคือ ชุดคำสั่งที่ใช้ในการปฏิบัติงานอย่างเช่น ATD (คำสั่งให้หมุนหมายเลขโทรศัพท์) หรือ ATH (คำสั่งให้วางสายโทรศัพท์) เป็นต้น ส่วนประเภทที่สองคือ ชุดคำสั่งที่ใช้กำหนดค่า หรือเปลี่ยนแปลงค่าต่างๆ อย่างเช่น

ATS7 = 90 เป็นการกำหนดค่าให้รีจีสเตอร์ S7 มีค่า 90 ซึ่งก็คือให้โมเด็มรอการตอบรับจากโมเด็มปลายทางเป็นเวลา 90 วินาที หลังจากนั้นหากยังไม่มีสัญญาณใดๆ ตอบกลับมาโมเด็มก็จะวางสายทันที ตามปกติ แล้วโปรแกรมสื่อสารมักจะตั้งค่าของรีจีสเตอร์ S ต่างๆ ที่จำเป็นเอาไว้ตั้งแต่ตอนที่ผู้เรียกใช้โปรแกรมนั้นๆ ขึ้นมา ซึ่งผู้ใช้สามารถจะควบคุมและเปลี่ยนแปลงค่าของรีจีสเตอร์ S เหล่านี้ได้ในภายหลัง

<u>การออนไลน์และออฟไลน์</u>

สภาวะออฟไลน์สามารถจะเรียกได้อีกอย่างหนึ่งว่าสภาวะคำสั่ง (Command state) หมายถึง สภาวะที่ผู้ใช้สามารถจะส่งคำสั่งต่างๆ ไปยังโมเด็มได้ หรือพูดอีกนัยหนึ่งก็คือ สภาวะที่โมเด็มจะแปลความหมายของข้อมูลที่ได้รับมาจาก PC ให้เป็นคำสั่งเท่านั้น ซึ่งสภาวะนี้โมเด็มไม่ได้รับคำสั่งข้อมูลกับโมเด็มปลายทางแต่จะสื่อสารกันกับ PC เท่านั้น ส่วนสภาวะออนไลน์ หมายถึง สภาวะที่โมเด็มได้เชื่อมต่อกับโมเด็มปลายทางเป็นที่เรียบร้อยแล้ว ข้อมูลที่ส่งออกมาจาก PC ก็จะผ่านจากโมเด็มต้นทางไปยังโมเด็มปลายทางเสมอ ในกรณีนี้หากผู้ใช้ต้องการที่จะส่งคำสั่งให้กับโมเด็มหรือต้องการจะให้โมเด็มกลับมาอยู่ในสภาพออฟไลน์ เพื่อรับคำสั่งจาก PC ก็จะสามารถทำได้ โดยมีวิธีการอยู่ 2 วิธีคือ หนึ่งให้โมเด็มวางสายแล้วกลับมาอยู่ในสภาพออนไลน์ใหม่ และวิธีที่สองคือ ส่งชุดอักขระ Escape Sequence ถ้าเข้าไปยังโมเด็ม ในขณะที่ออนไลน์ ซึ่งวิธีหลังจะมีความเหมาะสมกว่า เพราะการเชื่อมต่อระหว่างโมเด็มจะยังคงดำเนินไปอยู่ และหลังจากที่ได้ส่งคำสั่งต่างๆ ให้กับโมเด็มเป็นที่เรียบร้อยแล้ว ก็จะสามารถกลับเข้าไปอยู่ในสภาวะออนไลน์ได้เช่นเดิม วิธีการส่งชุดอักขระ Escape Sequence สำหรับโมเด็มที่เข้ากันได้กับโมเด็มของ Hayes คือให้รอ 1 วินาที (เรียกว่า Guard Time) แล้วกดปุ่ม + ติดต่อกัน 3 ครั้ง (+++) การที่ต้องรอ 1 วินาทีก่อนที่จะกดปุ่มเครื่องหมาย + เป็นสิ่งจำเป็นเนื่องจากว่าโมเด็มได้รู้ว่าอักษร + นั้น เป็นส่วนของอักขระ Escape Sequence ไม่ใช่ส่วนของข้อมูลที่จะต้องส่งไปยังโมเด็มปลายทาง

<u>รูปแบบของชุดคำสั่ง AT</u>

การใช้ชุดคำสั่ง AT จำเป็นต้องเริ่มต้นด้วยตัวอักษร AT เสมอ และจะต้องจบด้วยการกด Enter หรือ Cariage return ยกเว้นคำสั่ง A/ (ไม่ต้องกดปุ่ม Enter) ซึ่งหมายถึงให้โมเด็มกลับไปทำคำสั่งล่าสุดซ้ำอีกครั้งหนึ่ง การที่ Hayes ได้กำหนดคำสั่งต่างๆ ขึ้นต้นด้วยอักษร AT ก็เพราะต้องการให้โมเด็มรับรู้ถึงความเร็วและรูปแบบของอักษรคำสั่ง ที่ถูกส่งออกมาจากพอร์ตสื่อสารอนุกรมของ PC ซึ่งอย่าสับสนระหว่างความเร็วของพอร์ตสื่อสารอนุกรม (ความเร็วของ DTE) และความเร็วของโมเด็ม (ความเร็วของ DCE) ในกรณีนี้จะหมายถึงความเร็วของ DTE ตัวอย่างการใช้ชุดคำสั่ง AT ก็ได้แก่คำสั่ง ATH หมายถึงคำสั่งที่ให้โมเด็มวางสายโทรศัพท์เป็นต้น แต่บางคำสั่งต้องการพารามิเตอร์เพิ่มเติมอย่างเช่น คำสั่ง ATDT 3197707 หมายถึงให้โมเด็มหมุนหมายเลข 3197707 โดยหมุนแบบระบบ Tone (T) แต่ถ้าต้องการให้โมเด็มหมุนแบบระบบ Pulse ต้องใช้คำสั่ง ATDP 3197707 เป็นต้น

<u>การสนองคำสั่งของโมเด็ม</u>

เมื่อผู้ใช้ส่งคำสั่งต่างๆ ไปให้กับโมเด็มแล้ว โมเด็มก็จะสนองคำสั่งโดยจะส่งข้อความที่เรียกว่า Result code กลับมายังเครื่อง PC และจะปรากฏขึ้นบนจอภาพขณะที่รันโปรแกรมสื่อสารไว้ Result code เหล่านี้จะเป็นภาษาอังกฤษ อย่างเช่น OK, ERROR, CONNECT 2400 หรืออาจจะเป็นตัวเลขอื่นๆ ซึ่งคำสั่งที่เกี่ยวข้องกับการส่ง Result code ก็จะมีดังนี้

ATV0 เป็นคำสั่งที่กำหนดให้โมเด็มส่ง Result code ที่จำเป็นออกมาเท่านั้น (Non-verbose)

ATV1 เป็นคำสั่งที่กำหนดให้โมเด็มส่ง Result code ออกมาทุกๆครั้ง หลังจากที่ได้รับคำสั่งต่าง ๆ

ATQ1 เป็นคำสั่งที่กำหนดให้โมเด็มไม่ส่ง Result code ออกมา (Quiet mode)

ATQ0 เป็นคำสั่งที่ใช้ Toggle ให้โมเด็มสามารถไปส่ง Result code ได้ดังเดิม

ชุดคำสั่งเพิ่มเติม

ชุดคำสั่งรุ่นแรกๆ ของ Hayes นั้นจะใช้ตัวอักษรตั้งแต่ A จนถึง Z เช่น ATA จนถึง ATZ ซึ่งก็หมายความว่าคำสั่งจะมีให้ใช้ได้ไม่เกิน 26 คำสั่งเท่านั้น แต่โมเด็มในปัจจุบันก็ได้มีความสามารถต่างๆ ที่พิเศษเพิ่มขึ้นมา ดังนั้น Hayes จึงได้กำหนดชุดคำสั่งเพิ่มเติมขึ้นมาโดยใช้อักษร & เป็นตัวเข้ามาประกอบดังตัวอย่าง AT&F หมายถึงให้โมเด็มเรียกค่า Default setting ขึ้นมาใช้งาน ซึ่งโมเด็มของผู้ผลิตรายอื่นๆ ก็อาจจะเพิ่มเติมชุดคำสั่งที่ใช้ตัวอักษรอื่นๆ ประกอบ อย่างเช่น % หรือ * เป็นต้น

การตั้งค่าในรีจีสเตอร์ S

ดังกล่าวมาแล้วว่า รีจีสเตอร์ S มีหน้าที่เก็บข้อมูลที่ใช้ควบคุมสถานะการทำงานของโมเด็มเอาไว้ ซึ่งผู้ใช้สามารถตั้งค่า หรือตรวจดูค่าที่เก็บเอาไว้ ในรีจีสเตอร์ S ต่างๆ ได้โดยใช้ชุดคำสั่งที่ขึ้นต้นด้วย ATS ในกรณีที่ต้องการตั้งค่าให้แก่รีจีสเตอร์ S ผู้ใช้จะต้องใส่ตัวเลขที่ระบุว่าเป็นรีจีสเตอร์ S ตัวที่เท่าใด ตามด้วยเครื่องหมาย = และค่าที่ต้องการจะใส่ลงไป เช่น คำสั่ง ATS0 = 9

ส่วนในกรณีที่ต้องการตรวจดูค่าของรีจีสเตอร์ S ต่างๆ ก็จะต้องพิมพ์คำสั่งที่ขึ้นต้นด้วย ATS ตามด้วยตัวเลขที่ระบุว่าเป็นรีจีสเตอร์ S ที่เท่าไร และเครื่องหมาย ? เช่น คำสั่ง ATS11?

ในกรณีที่ต้องการส่งคำสั่งไปให้โมเด็มมากกว่าหนึ่งคำสั่ง ผู้ใช้อาจจะส่งคำสั่งทั้งหมดไปในครั้งเดียวได้ ซึ่งโมเด็มจะสามารถรับคำสั่งได้ไม่เกิน 40 ตัวอักษร ทั้งนี้จะนับตัวอักษร AT และCariage Return ด้วย ผู้ใช้อาจจะพิมพ์คำสั่งทั้งหมดให้ตัวอักษรอยู่ติดกันหรือเว้นช่องว่างระหว่างแต่ละคำสั่งได้

เช่น ATS7 = 90V1X4DT13055551234 หรือ ATS7 = 90V1X4DT1-305-555-1234

การเก็บค่าต่างๆ ที่เซตเอาไว้ใน Profile

จะเห็นได้ว่า ผู้ใช้สามารถตั้งค่ารีจีสเตอร์ S ต่างๆ และใช้คำ AT เพื่อควบคุมการทำงานของโมเด็มในลักษณะต่างๆ ที่ต้องการ ซึ่ง Hayes ก็ได้ออกแบบหน่วยความจำภายในโมเด็มขึ้นมาเพื่อใช้สำหรับเก็บค่าต่างๆ ที่ตั้งเอาไว้ และมีให้เลือกใช้มากกว่า 4 หน่วยความจำ โดยที่หน่วยความจำชุดแรก เรียกว่า แอกทีฟคอนฟิกเกอร์เรชั่น (Active Configuration) ใช้เก็บข้อมูลการตั้งค่าต่างๆ ที่เซตเอาไว้ใช้งานในปัจจุบัน เมื่อผู้ใช้มีการเปลี่ยนแปลงค่าของรีจีสเตอร์ S ต่างๆ และได้ตั้งคำสั่ง AT บางคำสั่ง ข้อมูลการตั้งค่าเหล่านี้ก็จะสูญหายไป นอกจากนี้หน่วยความจำที่เหลือจะมีอยู่สามชุดโดย 2 ชุดจะเป็น Nonvolatile RAM (NVRAM) และอีก 1 ชุดที่เหลือเป็น Read Only Memory (ROM) ซึ่งใช้เก็บข้อมูลที่เป็นค่า Default Setting ของโมเด็มทุกๆ ตัว มีชื่อเรียกว่า Factory Configuration หรือ Factory Setting ซึ่งอ่านออกมาได้อย่างเดียวไม่สามารถเขียนข้อมูลลงไปได้ ผู้ใช้สามารถจะเรียกเอาค่า Factory Setting ออกมาได้โดยคำสั่ง AT&F และนำเก็บไปไว้ที่ แอกทีฟคอนฟิกูเรชั่น หลังจากที่ได้ติดตั้งและเปิดเครื่องโมเด็มเป็นครั้งแรกโมเด็มก็จะถูกกำหนดให้เรียกเอา Factory Setting ออกมาใช้งานเสมอ ซึ่งจะมีประโยชน์มากในกรณีที่ผู้ใช้ประสบกับปัญหาการเซตค่าต่างๆ ไม่ถูกต้องก็สามารถใช้คำสั่ง AT&F เพื่อเรียกค่า ดีฟอลต์ (Default) ต่างๆ กลับมาใช้งานได้ใหม่ได้ และข้อมูลที่เป็นค่าดีฟอลต์เหล่านี้

จะอยู่ในคู่มือการใช้โมเด็มที่แถมมาด้วยอยู่แล้ว ส่วนหน่วยความจำที่เป็น NVRAM ทั้งสองชุดจะถูกเรียกว่า User Profiles มีไว้สำหรับให้ผู้ใช้สามารถสำรองเก็บค่าต่างๆ ที่เซตเอาไว้ได้ตามที่ต้องการ ซึ่งหน่วยความจำแบบ NVRAM นี้จะไม่สูญหายเมื่อเปิดเครื่อง และทุกครั้งหลังจากที่ได้เปิดเครื่องแล้วโมเด็มก็จะนำข้อมูลจากโพรไฟล์ที่ 0 มาเก็บไว้ที่แอกทีฟคอนฟิกเกอร์เรชั่น โดยอัตโนมัติ แต่ถ้าต้องการให้โมเด็มเรียกโพรไฟล์ที่ 1 ขึ้นมาทุกๆ ครั้งที่เปิดเครื่อง ก็จะต้องใช้คำสั่ง AT&Y1 ดังกล่าวในหัวข้อสรุปชุดคำสั่ง AT ต่อไป

<u>การสร้างการเรียกใช้ User Profile</u>

การสร้าง User Profile จะมีขั้นตอนดังต่อไปนี้

1. ให้โปรแกรมสื่อสารอยู่ในสภาวะคำสั่ง (Command State หรือเรียกว่า Local Mode)

2. ใช้คำสั่ง AT เพื่อตั้งค่าพารามิเตอร์ต่างๆ และตั้งค่ารีจีสเตอร์ S ที่ต้องการ

3. ใช้คำสั่ง AT&W0 หรือ AT&W1 เพื่อเก็บข้อมูลการเซตต่างๆ เอาไว้ในโพรไฟล์ที่0 หรือโพร ไฟล์ที่ 1

หลังจากนั้นผู้ใช้สามารถเรียกใช้ User Profile ต่างๆ ที่จัดเก็บเอาไว้ขึ้นมาได้โดยใช้คำสั่ง ATZ0 หรือ ATZ1 เพื่อรีเซตโมเด็ม และเรียกข้อมูลการเซตต่างๆ ในโพรไฟล์ที่ 0 หรือโพรไฟล์ที่ 1 ขึ้นมาใช้ในแอกทีฟคอนฟิกเกอร์เรชั่นตามลำดับ ในกรณีที่ได้ปิดเครื่องและเปิดเครื่องขึ้นมาใหม่อีกครั้งโมเด็มก็จะดึงเอาโพรไฟล์ที่ 0 ขึ้นมาเก็บไว้ที่คอนฟิกเกอร์เรชั่นโดยอัตโนมัติ ซึ่งผู้ใช้สามารถกำหนดให้โมเด็มอ่านโพรไฟล์ที่ 1 ขึ้นมาแทนโดยใช้คำสั่ง AT&Y1 ก่อนที่จะปิดเครื่องได้

<u>การเรียกดูข้อมูลที่ได้ตั้งไว้ทั้งหมด</u>

หลังจากที่ผู้ใช้ได้ตั้งพารามิเตอร์ต่างๆ ไว้เรียบร้อยแล้ว ผู้ใช้จะสามารถเรียกดูข้อมูลที่ตั้งเอาไว้ทั้งหมดได้โดยใช้คำสั่ง AT&V ดังรูปที่ 2.5

```
AT&V
ACTIVE PROFILE:
B16 B1 B41 B60 E1 L2 M1 N1 P Q0 V1 W0 X4 Y0 &A0 &C1 &D2 &G0 &J0 &K3 &Q5 &R0 &S0
&T4 &U0 &X0 &Y0
S00:000 S01:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:050 S08:002 S09:006
S10:014 S11:095 S12:050 S18:000 S25:005 S26:001 S36:007 S37:000 S38:020 S44:003
S46:002 S48:007 S49:008 S50:016 S97:030

STORED PROFILE 0:
B16 B1 B41 B60 E1 L2 M1 N1 P Q0 V1 W0 X4 Y0 &A0 &C1 &D2 &G0 &J0 &K3 &Q5 &R0 &S0
&T4 &U0 &X0
S00:000 S02:043 S06:002 S07:050 S08:002 S09:006 S10:014 S11:095 S12:050 S18:000
S25:005 S26:001 S36:007 S37:000 S38:020 S44:003 S46:002 S48:007 S49:008 S50:016
S97:030

STORED PROFILE 1:
B16 B1 B41 B60 E1 L2 M1 N1 P Q0 V1 W0 X4 Y0 &A0 &C1 &D2 &G0 &J0 &K3 &Q5 &R0 &S0
&T4 &U0 &X0
S00:000 S02:043 S06:002 S07:050 S08:002 S09:006 S10:014 S11:095 S12:050 S18:000
S25:005 S26:001 S36:007 S37:000 S38:020 S44:003 S46:002 S48:007 S49:008 S50:016
S97:030

TELEPHONE NUMBERS:
0=                              1=
2=                              3=

OK
```

รูปที่ 2.5 โพรไฟล์ซึ่งแสดงข้อมูลที่เก็บไว้ เรียกโดยใช้คำสั่ง AT&V

การเก็บหมายเลขโทรศัพท์ไว้ในโมเด็ม

จากรูปที่ 2.5 จะเห็นคำว่า TELEPHON NUMBERS บริเวณบรรทัดล่างได้ ซึ่งหมายความว่าผู้ใช้สามารถเก็บหมายเลขโทรศัพท์เอาไว้ใน NVRAM ได้ 4 หมายเลข รวมทั้งสามารถเก็บตัวอักษรที่จำเป็นในการหมุนโทรศัพท์ได้ ดังเช่น T (Tone), P (Pulse), W (wait) โดยใช้คำสั่ง

AT&Zn = xxxxxx

และสามารถสั่งให้โมเด็มหมุนหมายเลขที่เก็บไว้ได้โดยใช้คำสั่ง

ATDS = n

ตัวอย่างเช่น ถ้าต้องการเก็บหมายเลข 3197707 เอาไว้ในที่เก็บตำแหน่งที่ 2 และเมื่อเรียกใช้หมายเลขนี้จะกำหนดให้โมเด็มหมุนแบบ Tone ให้ใช้คำสั่งดังนี้

AT&Z2 = T3197707

และเมื่อต้องการให้โมเด็มหมุนหมายเลขโทรศัพท์ที่เก็บไว้นี้ ให้ใช้คำสั่ง

ATDS = 2

โดยทั่วไปแล้ว โปรแกรมสื่อสารก็มักจะมีฟังก์ชันการเก็บและหมุนหมายเลขไว้ในตัวเช่นกัน ซึ่งมักจะเรียกว่า Phone Book หรือ Dialing Directory แต่สำหรับในกรณีการเก็บหมายเลขโทรศัพท์เอาไว้ใน NVRAM เช่นนี้ มักจะใช้เกี่ยวกับกระบวนการที่เรียกว่า Dial Backup Number ซึ่งใช้ในระบบสื่อสารที่ค่อนข้างอัตโนมัติ และต้องการให้โมเด็มติดต่อสื่อสารอยู่ตลอดเวลา กระบวนการ Dial Backup Number นี้จะมีขั้นตอนการทำงาน คือ ขณะที่โมเด็มต้นทางติดต่อสื่อสารกับโมเด็มปลายทางแล้วเกิดปัญหาสายขาด หรือสายหลุดไม่สามารถจะดำเนินการสื่อสารได้ต่อไปได้ โมเด็มต้นทางก็จะสามารถหมุนเบอร์โทรศัพท์ที่เก็บเอาไว้ใน NVRAM ได้อัตโนมัติ เพื่อทำการโทรศัพท์กลับไปเชื่อมต่อกลับปลายทางอีกครั้งหรือโทรศัพท์ไปยังโมเด็มตัวอื่นๆ ตามที่ผู้ใช้กำหนดเอาไว้

2.3.3 ชุดคำสั่ง AT ที่สำคัญ

ในหัวข้อนี้เราจะกล่าวถึงคำสั่ง AT ที่สำคัญและมักจะถูกใช้งานอยู่เสมอ ซึ่งเป็นคำสั่งทั่วๆ ไปที่สามารถใช้งานได้กับโมเด็มที่ใช้มาตรฐานของ Hayes ซึ่งโมเด็มดังกล่าวอาจจะมีชุดคำสั่งเพิ่มเติมขึ้นมา จากชุดคำสั่งที่จะกล่าวถึงเพื่อสนับสนุนความสามารถพิเศษของโมเด็มนั้นๆ ดังนั้นผู้ใช้สามารถศึกษาเพิ่มเติมจากคู่มือของโมเด็มดังกล่าวนั่นเอง

ATA (รับโทรศัพท์)

คำสั่งนี้จะทำให้โมเด็มสามารถรับสายโทรเข้าได้

ATD (หมุนหมายเลขโทรศัพท์)

คำสั่ง ATD จะทำให้โมเด็มยกหูโทรศัพท์ขึ้น (Off-hook) และหมุนหมายเลขโทรศัพท์ออกด้วย

ATE เปิดและปิดการแสดงอักษร (Echo)

ATE0 หรือ ATE : ECHO OFF

ATE1 : ECHO ON (Default)

ATH ยกหูและวางหูโทรศัพท์

ATH0 หรือ ATH : ให้โมเด็มวางหูโทรศัพท์

ATH1 : ให้โมเด็มยกหูโทรศัพท์

ATL ควบคุมความดังของเสียง

ATL0 หรือ ATL : ให้มีเสียงเบาที่สุด

ATL1 : ให้มีเสียงดังปานกลาง

ATL2 : ให้มีเสียงดังที่สุด

ATM ควบคุมการเกิดเสียง

ATM0 หรือ ATM : ให้โมเด็มปิดเสียงทั้งหมด

ATM1 : ให้โมเด็มส่งเสียงได้ในขณะที่หมุนหมายเลขโทรศัพท์ และปิดเสียง เมื่อเชื่อมต่อกับโมเด็มปลายทางเรียบร้อยแล้ว

ATM2 : อนุญาตให้โมเด็มส่งเสียงออกจากลำโพงตลอดเวลา

ATM3 : ให้โมเด็มปิดเสียง ในขณะที่หมุนหมายเลขโทรศัพท์ และหลังจากนั้นก็ทำการปิดเสียงจนกระทั่งได้รับสัญญาณพาหะ จากโมเด็มปลายทางจึงจะปิดเสียงอีกครั้งหนึ่ง

ATO กลับสู่ภาวะออนไลน์

การใช้คำสั่งนี้ในสภาวะคำสั่งจะทำให้โมเด็มกลับเข้าไปสู่สภาวะออนไลน์อีกครั้ง

ATS คำสั่งตั้งค่าให้แก่รีจีสเตอร์ S

รีจิสเตอร์	ย่าน/หน่วย	ความหมาย
S0	0-255 ครั้ง	จำนวนเสียงกริ่งก่อนที่จะรับโทรศัพท์
S6	2-255 วินาที	เวลาที่รอคอยสัญญาณให้หมุน
S7	1-60 วินาที	เวลาที่รอคอยสัญญาณ
S13	บิทแมป (Bitmapped)	รีจิสเตอร์สถานะของ UART

ตารางที่ 2.4 รีจิสเตอร์ของโมเด็ม Hayes ที่ใช้บ่อย

ATV กำหนด Verbose Mode

ATV0 หรือ ATV : ให้โมเด็มส่ง Result code เป็นตัวเลข

ATV1 : ให้โมเด็มส่ง Result code เป็นตัวอักษรโดยมีดีฟอลต์เป็น V1

ATWn กำหนด Result code เพิ่มเติม

ATW0 หรือ ATW : ไม่อนุญาตให้โมเด็มส่ง Negotiation Progress Codes ให้แก่ PC

ATW1 : อนุญาตให้โมเด็มส่ง Negotiation Progress Codes ให้แก่ PC

ATX กำหนดชนิดของ Result code

คำสั่ง	รูปแบบ
ATX0	ให้ Result code ให้ข้อความคือ OK, CONNECT, RING, NO CARRIER และ ERROR เมื่อเลือกตัวเลือกนี้ สัญญาณหมุนหมายเลขและสัญญาณสายไม่ว่างจะไม่ทำงาน
ATX1	เหมือนกับคำสั่ง ATX0 แต่เพิ่ม Result code ที่เกี่ยวกับความเร็วเข้าไป เช่น CONNECT 2400
ATX2	ทุกตัวเลือกของคำสั่ง ATX1 แต่เพิ่ม Result code ข้อความ " No Dial Tone " เข้าไป
ATX3	ทุกตัวเลือกของคำสั่ง ATX1 แต่เพิ่ม Result code ข้อความ " Busy " เข้าไป
ATX4	เลือก Result code ได้ถูกตัว

ตารางที่ 2.5 คำสั่ง ATX

ATZ รีเซตโมเด็ม

ATZ0 หรือATZ : โหลดข้อมูลการเซตจากโพรไฟล์ที่ 0 กลับมาเก็บไว้ในแอกทีฟโพรไฟล์

ATZ1 : โหลดข้อมูลการเซตจากโพรไฟล์ที่ 1 กลับมาเก็บไว้ในแอกทีฟโพรไฟล์

AT&C ควบคุมสัญญาณ CD

คำสั่ง	รูปแบบ
AT&C0	สัญญาณ CD จะอยู่ในสถานะ " ON " เสมอไม่ว่าโมเด็มจะออนไลน์อยู่หรือไม่ก็ตาม
AT&C1	สัญญาณ CD จะอยู่ในสถานะ " ON " เมื่อโมเด็มออนไลน์ และมีการต่อเชื่อมกับโมเด็มตัวอื่นๆ เท่านั้น
AT&C2	สัญญาณ CD จะอยู่ในสถานะ " ON " เมื่ออยู่ในสภาวะคำสั่ง (Command State) หลังจากที่โมเด็มทำการเชื่อมต่อเรียบร้อยแล้ว สัญญาณ CD จะมีสถานะตามความเป็นจริง
AT&C3	สัญญาณ CD จะอยู่ในสถานะ " ON " เสมอไม่ว่าโมเด็มจะออนไลน์อยู่หรือไม่

ตารางที่ 2.6 คำสั่ง AT&C

AT&D ควบคุมการรับรู้สัญญาณ DTR

คำสั่ง	รูปแบบ
AT&D0	โมเด็มไม่รับรู้สัญญาณ DTR
AT&D1	โมเด็มรับรู้สัญญาณ DTR และจะไม่ตอบรับถ้าสัญญาณอยู่ในสถานะ " off " ถ้า DTR เปลี่ยนจาก " on " เป็น " off " ขณะออนไลน์โมเด็มจะยังคงยกหูโทรศัพท์ไว้ แต่จะเข้าสู่สภาวะคำสั่ง ซึ่งสามารถรับกลับสู่สถานะออนไลน์ได้โดยใช้คำสั่ง ATO
AT&D2	โมเด็มรับรู้สัญญาณ DTR และจะไม่ตอบรับถ้าสัญญาณอยู่ในสถานะ " off " ถ้า DTR เปลี่ยนจาก " on " เป็น " off " ขณะที่ออนไลน์โมเด็มจะหยุดการติดต่อ และจะกลับสู่สภาวะคำสั่ง
AT&D3	โมเด็มรับรู้สัญญาณ DTR จะไม่ตอบรับถ้า DTR อยู่ในสถานะ " off " ถ้าDTR เปลี่ยนจาก " on " เป็น " off " ขณะที่ออนไลน์โมเด็มจะหยุดการติดต่อและรีเซตตัวเอง

ตารางที่ 2.7 AT&D

AT&K เลือกชนิดของการควบคุมการไหลของข้อมูล

คำสั่ง	รูปแบบ
AT&K0	ไม่มีการใช้การควบคุมการไหลของข้อมูล ตัวอักษร XON และ XOFF จะสามารถส่งผ่านได้ ส่วนสัญญาณ RTS มีสถานะ " ON " ตลอดเวลา และไม่รับรู้สัญญาณ CTSใช้การควบคุมการไหลข้อมูลแบบ RTS / CTS
AT&K3	ใช้การควบคุมการไหลข้อมูลแบบ XON / XOFF
AT&K4	ใช้การควบคุมการไหลแบบ " Transparent " วิธีนี้ไม่ค่อยใช้กันมากและจะถูกเลือกภายใต้ซอฟต์แวร์คอนโทรลเท่านั้น
AT&K5	

ตารางที่ 2.8 คำสั่ง AT&K

AT&W เก็บข้อมูลการเซตไว้ใน NVRAM

AT&W0 หรือ AT&W : ให้เก็บข้อมูลการเซตจาก Active Profile ลงใน User Profile ชุดที่ 0

AT&W1 : ให้เก็บข้อมูลการเซตจาก Active Profile ลงใน User Profile ชุดที่ 1

AT&Y โหลดข้อมูลการเซตจาก NVRAM เมื่อเปิดเครื่อง

AT&Y0 หรือAT&Y : ให้เรียก User Profile ตำแหน่งที่ 0 มาเก็บเอาไว้ที่แอกทีฟ โพรไฟล์ หลังจากเปิดเครื่อง

AT&Y : ให้โหลด User Profile ตำแหน่งที่ 1 มาเก็บเอาไว้ที่แอกทีฟโพรไฟล์

2.4 การอินเตอร์เฟสแบบอนุกรม RS-232

เทคนิคที่สำคัญอีกชนิดหนึ่งของการเชื่อมต่อ คือ การอินเตอร์เฟสแบบอนุกรม RS-232 โดยมีมาตรฐานที่ปรับปรุงใหม่เป็น RS-232

<u>การจัดวางตัวอักขระในรูปแบบอนุกรม</u>

ตัวอักขระจะถูกเก็บในคอมพิวเตอร์ในลักษณะของรูปแบบขนานขนาด 8 บิท ถูกกำหนดโดยแต่ละด้าน แต่ในการส่งแบบอนุกรมมันจะถูกแบ่งเป็น 8 ส่วนที่แยกจากกัน (ส่วนมากมักจะเป็น 7 บิท) ซึ่งแต่ละตัวจะส่งแยกจากกัน วงจรพิเศษทางอิเล็กทรอนิกส์ที่เรียกว่า UART (ใช้สำหรับการส่งและรับแบบอะซิงโครนัส) ใช้สำหรับการทำงานนี้ UART จะถูกสร้างในคอมพิวเตอร์หรือรวมเข้ากับส่วนของการอินเตอร์เฟสแบบอนุกรม UART รุ่นใหม่จะประกอบไปด้วยไอซีตัวเดียวและสามารถทำงานในหน้าที่ซับซ้อน UART ของคอมพิวเตอร์จะรับข้อมูลอักขระ 8 บิท นำมาแยกส่วนและส่งแต่ละบิทออกด้วยอัตราที่กำหนดไว้ก่อน ในการส่งข้อมูล 7 บิท UART จะส่ง 8 บิท ซึ่งมีพาริตี้บิท (Parity Bit) และอีก 2 บิทพิเศษคือ บิทเริ่มต้น (Start Bit) และบิทสิ้นสุด (Stop Bit)

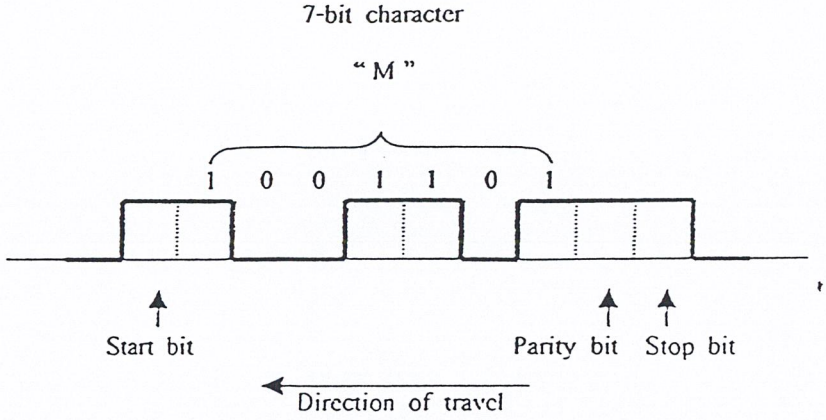
<u>พาริตี้บิท</u>

บิทที่ 8 ที่ส่งโดย UART ตามหลังข้อมูล 7บิทที่ถูกสร้างจากตัวอักขระ โดยปกติมักจะเป็นพาริตี้บิทซึ่งคือการตรวจสอบความผิดพลาด โดยจะมีการใช้บิทพาริตี้ 2 ทางคือ พาริตี้คู่ หรือ พาริตี้คี่ สมมุติว่าพาริตี้คี่ จะมีขบวนการทำงานดังนี้ ถ้ามีจำนวนบิท “1” เป็นจำนวนคู่ตัวในข้อมูลอักขระ 7 บิท ดังนั้นบิทพาริตี้จะต้องเป็น “1” ซึ่งจะมีผลทำให้มีจำนวนบิท “1” เป็นคี่ตัว ถ้ามีจำนวนบิท “1” เป็นจำนวนคี่ตัว ดังนั้นพาริตี้บิทจะถูกกำหนดเป็น “0” ซึ่งจะมีผลทำให้มีจำนวนบิทที่เป็น “1” เป็นคี่ตัว ดังนั้นข้อมูลที่ถูกส่งไปจะมีจำนวนบิท ”1” เป็นจำนวนคี่ตัว UART จะตรวจดูข้อมูลอักขระแต่ละตัวจะประกอบด้วยบิท “1” จำนวนคี่ตัว ถ้าในระหว่างการส่งมีบิทใดบิทหนึ่งในรูปแบบของตัวอักษรถูกเปลี่ยนแปลง บิทพาริตี้จะไม่ตรงกับข้อมูล และ UART จะลดความผิดพลาดที่จะเกิดขึ้น ส่วนระบบของพาริตี้คู่ก็เหมือนกัน เพียงแต่บิทนี้จะทำการส่งบิท “1” เป็นจำนวนคู่ตัว

<u>บิทเริ่มต้นและบิทสิ้นสุด</u>

UART จะต้องมีเวลาในการเตรียมตัวสำหรับการรับข้อมูล โดยเมื่อบิทเริ่มต้นถูกส่งมาก่อนที่จะส่งข้อมูลอักขระจริงมาตามสายสัญญาณ เมื่อตัวอักขระถูกส่งหมดแล้วบิทสิ้นสุดจะถูกส่งมา ซึ่งจะทำให้ UART สามารถแน่ใจได้ว่าการส่งสิ้นสุดลงแล้ว การใช้บิทเริ่มต้นและบิทสิ้นสุดนี้มีความจำเป็นมาก เพราะการส่งข้อมูลเป็นแบบอะซิงโครนัสไม่ใช่แบบซิงโครนัส ซึ่งคอมพิวเตอร์จะรู้แน่นอนว่าเมื่อใดข้อมูลอักขระจึงจะมาถึง

รูปที่ 2.6 แสดงขบวนของพัลซ์ที่แทนตัวอักษร “M” ซึ่งถูกส่งผ่านสาย RS-232 แบบอนุกรม



รูปที่ 2.6 ตัวอักษร "M" ในรูปแบบของ RS-232

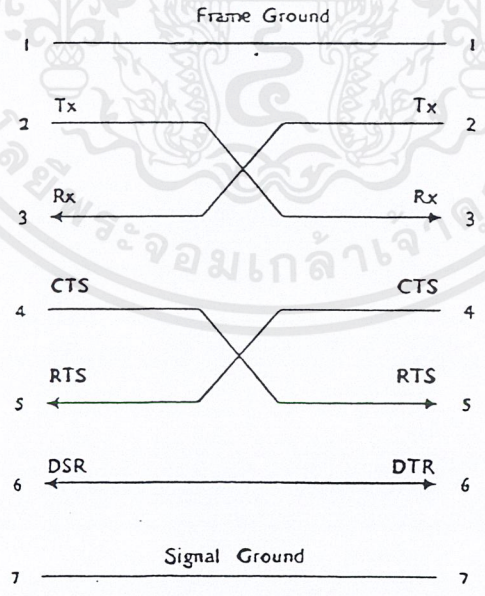
<u>สายสัญญาณต่างๆ</u>

1. สายส่งข้อมูล

จากการอธิบายบิทข้อมูลต่างๆ ข้อมูล 7 บิท พาริตี้บิท บิทเริ่มต้นและบิทสิ้นสุด จะถูกส่งผ่านสาย 1 สายที่เรียกว่า สายส่งข้อมูลปกติ (TX DATA) โดยที่ปลายสายด้านคอมพิวเตอร์จะเรียกว่า สายส่งข้อมูล ในขณะที่ปลายสายด้านโมเด็มจะเรียกว่า สายรับข้อมูล (RX DATA) โดยปกติขา 3 ของคอนเนคเตอร์จะใช้สำหรับทั้งคอมพิวเตอร์และโมเด็มสำหรับสายนี้ โดยการอินเตอร์เฟสแบบอนุกรมนั้นจะคล้ายกับการเชื่อมต่อแบบขนาน โดยจะต้องมีสายสัญญาณอื่น นอกจากสัญญาณข้อมูล รูปที่ 2.3 แสดงสายสัญญาณสำหรับมาตรฐาน RS-232

2. สายสัญญาณกราวด์

สายนี้เป็นหนึ่งในมาตรฐาน RS-232 ซึ่งโดยปกติจะต่อกับขา 7 ทั้งโมเด็มและคอมพิวเตอร์โดยปกติจะเรียกว่า เซอร์กิตคอมมอน (Circuit Common) เหมือนกับการอินเตอร์เฟสแบบขนาน กราวด์ชนิดต่างๆจะถูกใช้เป็นกราวด์เพื่อป้องกัน (Protective Ground) ซึ่งจะมีจุดประสงค์เดียวกันกับแชสสิสกราวนด์



รูปที่ 2.7 แสดงการอินเตอร์เฟสแบบ RS-232

3. ข้อมูลพร้อม(Data Set Ready Line)

สายนี้จะถูกเรียกว่า "Data Set Ready"หรือ "DSR"ที่เครื่องคอมพิวเตอร์ และเรียกว่า "Terminal Ready"หรือ "DTR"ที่ด้านโมเด็ม โดยปกติสายนี้มักจะเป็นสายของการแฮนด์เชค เทียบได้กับสัญญาณไม่

ว่างในการเชื่อมต่อแบบขนาน ดังนั้นเมื่อบัฟเฟอร์เต็ม โมเด็มจะใช้สายนี้เพื่อส่งสัญญาณไปที่คอมพิวเตอร์เพื่อไม่ให้ส่งข้อมูลมาอีก โดยจะใช้ขาที่ 6 จากคอมพิวเตอร์และขาที่ 4 จากโมเด็ม

4. สายรับข้อมูล (Receive Data Line)

สายนี้ถูกเรียกว่า "สายรับข้อมูล" ที่ด้านคอมพิวเตอร์ และ "สายส่งข้อมูล" ที่ด้านโมเด็ม สายรับข้อมูลนี้ไว้ใช้เพื่อรับข้อมูลทางด้านโมเด็ม

5. เคลียร์ทูเซ็นด์(Clear to Send)และดาต้าเทอร์มินอลเร้ดดี้(Data Terminal Ready)

สายเหล่านี้มีความจำเป็นในการอินเตอร์เฟสบางอย่าง บางครั้งมันจะมีข้อมูลเหมือนกับการแฮนด์เชค ซึ่งโดยปกติจะอยู่บนสายดาต้าเซ็นด์เร้ดดี้ ในการอินเตอร์เฟสอื่นๆ UART ที่โมเด็ม หรือคอมพิวเตอร์ จะต้องทำการตรวจสอบว่าสายนี้ได้ถูกเชื่อมต่อแล้ว แต่มันไม่ได้ถูกใช้สำหรับการแฮนด์เชค แต่สายนี้ต้องถูกต่อเพื่อให้การอินเตอร์เฟสทำงาน

เคลียร์ทูเซ็นด์(CTS)โดยปกติจะมาจากขา 4 ของคอมพิวเตอร์ต่อเข้ากับขา 5 ของโมเด็ม(ซึ่งจะเรียกว่า RTS)RTSโดยปกติจะมาจากขา 5 ของคอมพิวเตอร์และต่อกับขา 4 ของโมเด็ม(ในที่นี้จะเรียกว่า CTS)สายนี้จะกลับซึ่งกันและกัน

<u>ความเร็วในระบบ RS-232</u>

บอดเรท คือ จำนวนของบิทต่อวินาทีที่จะถูกส่งโดย UART เพราะว่าแต่ละตัวอักขระจะต้องการประมาณ 10 บิท(7 บิทข้อมูลและอย่างละ 1 บิทสำหรับพาริตี้บิท บิทเริ่มต้น และบิทสิ้นสุด)การหารบอดเรทด้วย 10 จะได้จำนวนของตัวอักขระใน 1 วินาที ดังนั้น 300 บอดเรท จะสามารถส่งตัวอักขระได้ 30 ตัวต่อวินาที โดยปกติบอดเรทที่ถูกใช้ระหว่างคอมพิวเตอร์และโมเด็มจะเป็น 300,1200,2400,4800 และ 9600 โดยดิฟสวิทซ์จะถูกใช้เพื่อกำหนดอัตราที่โมเด็ม ขณะที่ซอฟท์แวร์ (Software) จะกำหนดอัตราที่คอมพิวเตอร์ บางครั้งบอดเรทอาจถูกเปลี่ยนจากระบบการทำงาน ในบางกรณีถ้ามีปัญหาในการทำงานที่เชื่อมต่อแบบอนุกรม ก็ให้ตรวจสอบดูที่บอดเรทเพื่อกำหนดให้เหมาะสมกัน บางครั้งการเชื่อมต่อแบบอนุกรมโพรซีเยอร์(Procedure) ของการแฮนด์เชคจะไม่ทำงานด้วยเหตุผลบางประการ

2.5 แอลซีดีแบบกราฟฟิก (GRAPHIC LCD DV-12864)

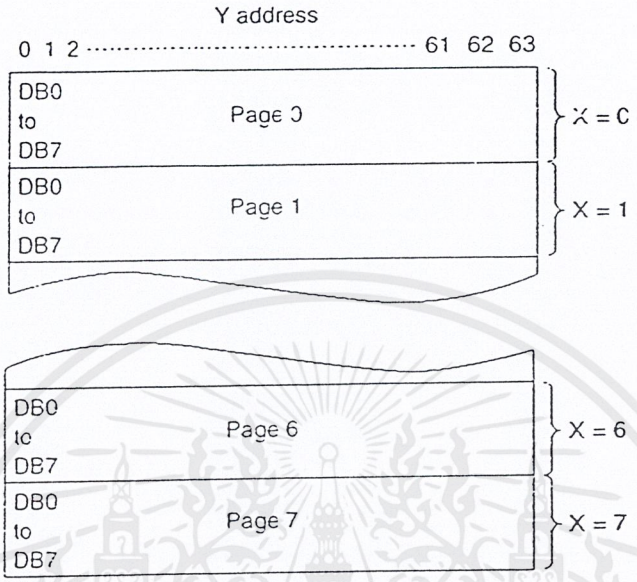
- **แอลซีดี(LCD DV-12864)**

  DV-12864 เป็น LCD Graphic ขนาด 128 X 64 Dot ซึ่งมี Controller ภายใน (HD61202, HD61203) โดยการทำงานของ Controller จะมีลักษณะการควบคุมดังนี้

Line คือการอ้างอิงบรรทัดของข้อมูลภายในจะแบ่งเป็น 64 แถว (com1 – com64)

Page (X-address) เป็นการอ้างอิงถึงหน้าต่างของการแสดงผลภายในหนึ่ง page จะประกอบไปด้วย 8 line ซึ่งจะเป็นการอ้างอิงถึงข้อมูลด้วย data-bus โดยตรง ภายในของ LCD จะประกอบไปด้วย 8 page ซึ่งถูกชี้โดย X-register โดยเมื่อต้องการให้ LCD แสดงผลที่หน้าต่างใดของหน้าจอ เราจะต้องตั้งค่า X ให้กับ LCD ซึ่งเมื่อตั้งค่า X ให้กับ LCD แล้ว ค่า X นั้นจะไม่มีการเปลี่ยนแปลงจนกระทั่งจะมีการตั้งค่าใหม่ให้กับ LCD

Segment(Y-address) เป็นค่าพอยท์เตอร์ในการชี้ที่อยู่ของข้อมูลซึ่งภายใน LCD จะถูกควบคุมการชี้ของข้อมูลโดย HD61202 ซึ่ง HD61202 จะสามารถชี้ที่อยู่ของข้อมูลได้ 64 segment ซึ่ง HD61202 ทั้งสองตัวก็จะสามารถทำการอ้าง segment ได้ถึง 128 segment

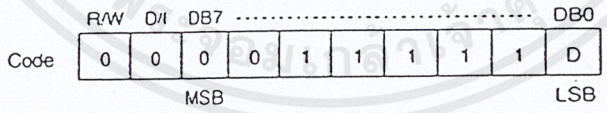


รูปที่ 2.8 แสดงตำแหน่งแอดเดรส (Address)

โดยการใช้งานเมื่อทำการตั้งค่า Y แล้ว ค่าจะถูกเพิ่มขึ้นหนึ่งค่าขึ้นเสมอ เมื่อมีการอ่านหรือเขียนข้อมูลบน LCD ***** ข้อควรระวัง** เมื่อค่า Y ถูกเพิ่มขึ้นมากกว่า 63 แล้ว ค่า Y จะยังไม่เป็นการอ้างถึงข้อมูล segment ที่ 64 **(Segment 0 ของ CS2)** ดังนั้นตัวโปรแกรมจะต้องช่วยจัดการในส่วนนี้

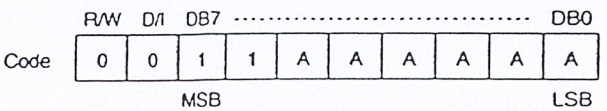
คำสั่งควบคุมการแสดงของ LCD

1. Display ON/OFF



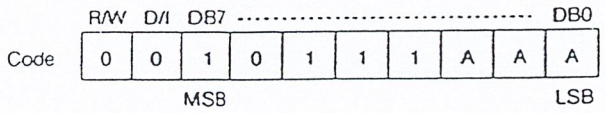
เป็นคำสั่งควบคุมการแสดงผล โดยการแสดงผลจะขึ้นอยู่กับค่า D (DBO) เมื่อค่า D เป็น 1 LCD จะทำการแสดงผลและเมื่อค่า D เป็น 0 LCD จะไม่ทำการแสดงผล ข้อมูลภายใน LCD จะไม่มีการเปลี่ยนแปลงเนื่องจากคำสั่งนี้

2. Display Start Line



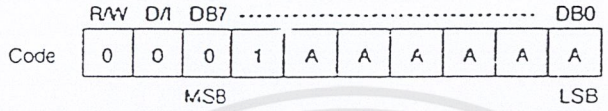
ค่า AAAAAA จะเป็นค่าหมายเลขบรรทัด ที่จะให้ LCD แสดงผลเป็นบรรทัดแรกของจอภาพในรูปจะเป็นตัวอย่างของการเลือกค่า Line จาก 0-3 ซึ่งจะทำให้การแสดงผลแตกต่างกันออกไป

3. Set PAGE (X-Address)



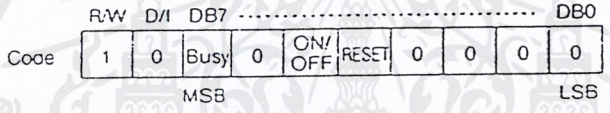
ค่า AAA ของคำสั่ง จะเป็นการตั้งค่า X-Address ซึ่งหลังจากทำคำสั่งนี้แล้วข้อมูลจาก DB0-DB7 จะเป็นการติดต่อกับ RAM ที่ PAGE นี้จนกว่าจะมีการตั้งค่าใหม่ให้กับ LCD

4. Set Y-Address



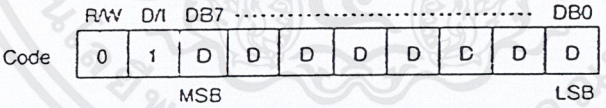
ค่า AAAAAA จะเป็นการตั้งค่าของ Y-Address (ค่า Y จะมีค่าอยู่ระหว่าง 0-63) และค่า Y จะเพิ่มขึ้นครั้งละ 1 เมื่อมีการอ่านหรือเขียนข้อมูลจาก CPU

5. Status Read



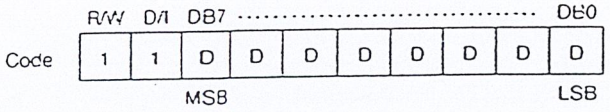
เป็นการอ่านค่าสถานะของ LCD โดยถ้าค่า Busy เป็น 1 LCD จะทำงานส่วนภายในซึ่งจะทำให้ไม่สามารถทำการควบคุม LCD ในขณะนี้ได้ เพราะฉะนั้นเพื่อให้แน่ใจในการควบคุมครั้งต่อไปจะต้องตรวจ ค่า Busy ให้ได้ค่าเป็น 0 เสียก่อน

6. Write Display Data

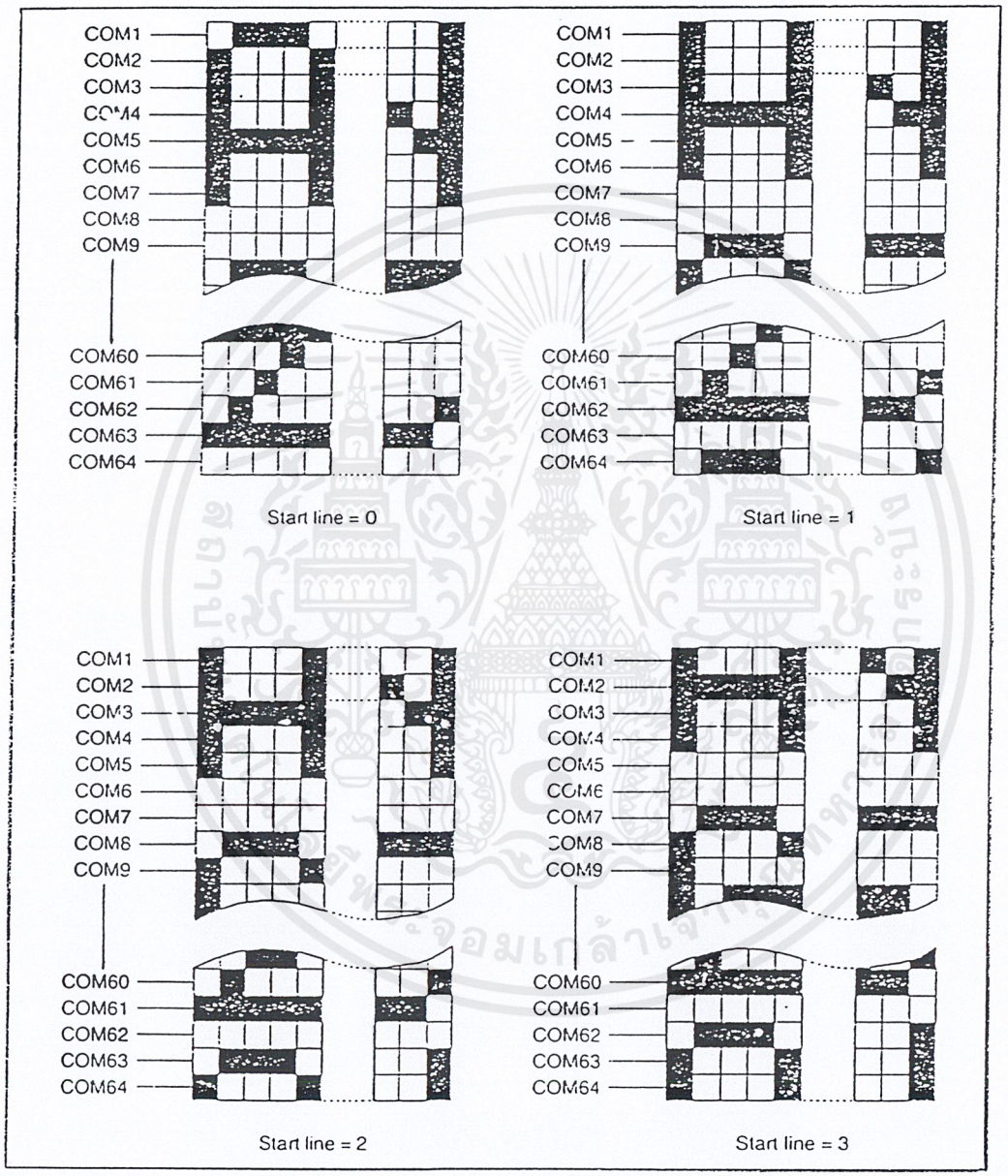


การเขียนข้อมูลเข้าไปใน LCD ซึ่งข้อมูล DDDDDDDD จะถูกเก็บใน LCD RAM และค่า Y จะถูกเพิ่มขึ้น 1

7. Read Display Data

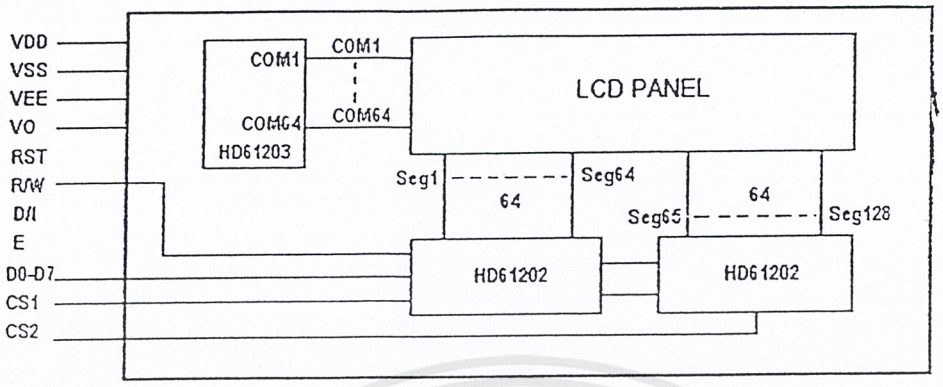


เป็นการอ่านข้อมูลที่แสดงผลโดย LCD จะให้ค่าของข้อมูลออกมาที่ Data bus ค่า Y จะถูกเพิ่มขึ้น 1 เช่นเดียวกับการเขียนข้อมูล



รูปที่2.9 แสดงตัวอย่างการตั้งค่า Display Start Line

- **อินเตอร์เฟสแอลซีดี (INTERFACE LCD DV-12864)**



รูปที่ 2.10 แสดงโครงสร้างภายในและขาควบคุม

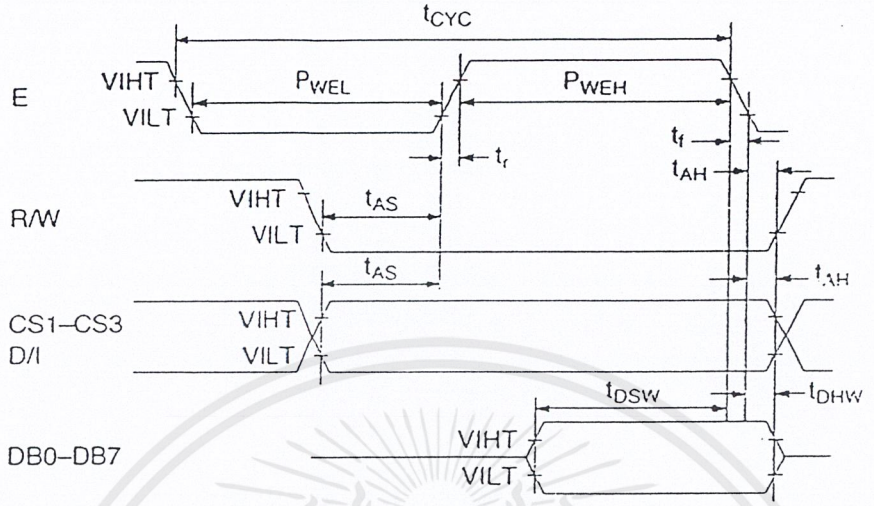
โครงสร้างภายในของ LCD จะประกอบด้วยส่วนของ Controller โดย HD61203 จะควบคุมการอ้างถึง Page ของข้อมูลและ HD61202 จะควบคุมในการอ้างของ Segment ซึ่งในการใช้งานเราจะต้องควบคุมส่วนต่าง ๆเหล่านี้โดยการส่งรหัสควบคุมไปที่ขาของ LCD ดังนี้

ขา RST	เป็นขาที่ใช้ Reset การทำงานของ LCD
ขา E	เป็นขา Enable การรับส่งข้อมูลจะทำงานที่ Logic HIGH และขอบขาลง
ขา R/W	เป็นขาที่ใช้กำหนดการอ่านหรือเขียนข้อมูล
ขา D/I	ใช้บอกถึงข้อมูลใน Data-bus ว่าเป็นรหัสควบคุมหรือเป็นส่วนของข้อมูล
ขา CS1	Chip Select ของ HD61201 ตัวแรก** 1
ขา CS2	Chip Select ของ HD61201 ตัวที่สอง
ขา Data-Bus	เป็นขาใช้ส่งข้อมูลหรือรหัสควบคุม

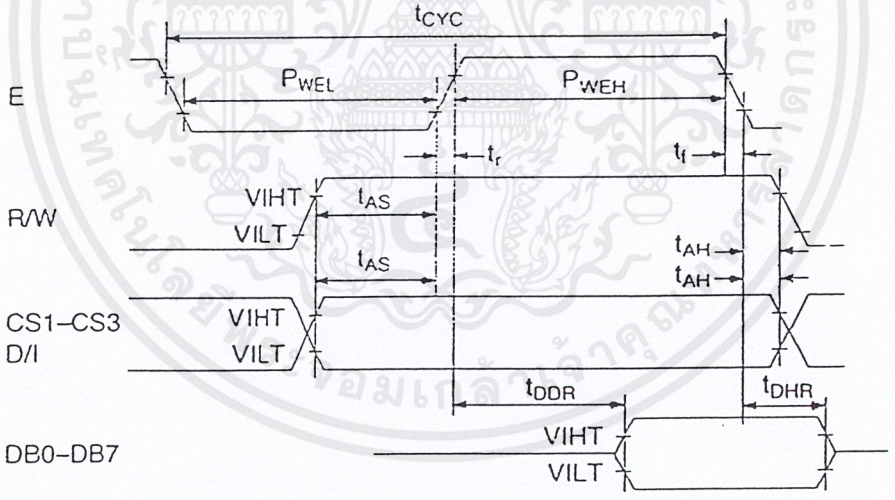
หมายเหตุ 1. เมื่อ CS1 เป็น HIGH และ CS2 เป็น LOW จะเป็นการอ้างอิงถึง Segment ที่ 0-63 และเมื่อ CS1 เป็น LOW CS2 เป็น HIGH จะเป็นการอ้างอิงถึง Segment ที่ 64-127

2. สภาวะการทำงานของขาควบคุมสามารถดูได้จากรูปที่ 2.11

1. Write Operation



2. Read Operation



รูปที่ 2.11 แสดงสภาวะการทำงานของขาควบคุม

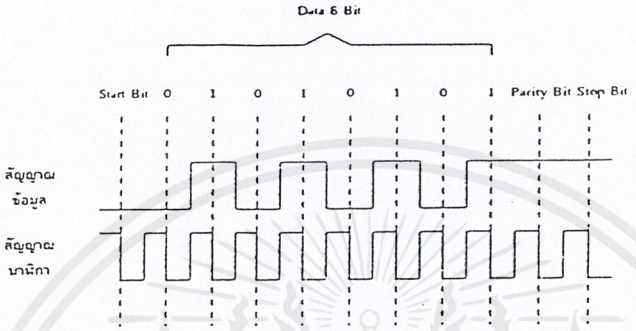
นอกจากขาควบคุมต่าง ๆแล้ว ยังมีขาของแหล่งจ่ายไฟ คือ

ขา VSS	Ground
ขา VDD	แรงดันไฟเลี้ยงวงจร Logic
ขา V0	แรงดันไฟเลี้ยง LCD
ขา VEE	ขาจ่ายแรงดันไฟลบ โดยเมื่อต่อ VDD ให้วงจรขา VEE จะจ่ายแรงดันไฟลบออกมา (ใช้นำไปขับ LCD ที่ขา V0)

2.6 การเชื่อมต่อคีย์บอร์ด (IBM PCAT)

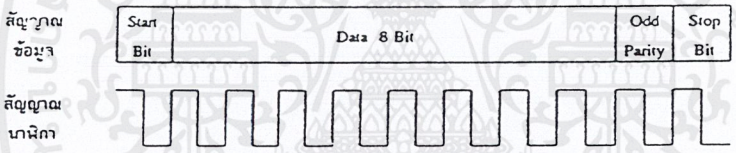
ทฤษฎีเกี่ยวกับคีย์บอร์ด (Keyboard)

การรับ-ส่ง ข้อมูลของคีย์บอร์ดเป็นแบบ Synchronous ซึ่งจะประกอบด้วยสัญญาณข้อมูลและสัญญาณนาฬิกา โดยสัญญาณข้อมูล 1 ชุด จะมีขนาด 11 บิต จำแนกได้เป็น บิตแรกเป็นบิตเริ่มต้น (Start Bit) มีสภาวะเป็นลอจิก "0" ตามด้วยบิตข้อมูลขนาด 8 บิต, 1 บิตพาริตี้ (Parity Bit) โดยใช้เป็นพาริตี้คี่ (Odd Parity) และบิตสุดท้ายจะเป็นบิตสิ้นสุด (Stop Bit) ซึ่งจะมีสภาวะเป็นลอจิก "1" การรับ-ส่งข้อมูล จะอาศัยสัญญาณนาฬิกาเพื่อสร้างความสอดคล้องกับอุปกรณ์ที่นำมาเชื่อมต่อ โดยสามารถแสดงได้ดังรูป 2.12



รูปที่ 2.12 แสดงการรับส่งข้อมูลแบบซิงโครนัส

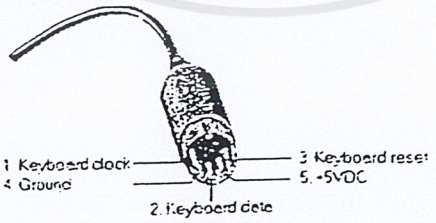
ตัวอย่างเช่น ข้อมูลที่ส่งออกมาจากคีย์บอร์ดเป็น 0AAH สัญญาณที่วัดได้จะเป็น



รูปที่ 2.13 ตัวอย่างสัญญาณที่วัดได้

จากรูป 2.13 ข้อมูลบิตแรก (ทางซ้ายมือ) เป็นบิตเริ่มต้น มีสภาวะเป็นลอจิก "0" ตามด้วยข้อมูล 8 บิต (10101010B) โดยลักษณะของการส่งข้อมูลจะส่งบิต LSB ก่อน ในที่นี้คือ "0" แล้วตามด้วย MSB จากนั้นจะเป็นพาริตี้บิต จากข้อมูล 8 บิต สังเกตว่ามีข้อมูลที่เป็นลอจิก "1" อยู่ 4 บิต ดังนั้นพาริตี้บิตจะเป็นลอจิก "1" เพื่อให้บิตที่มีสภาวะเป็นลอจิก "1" เป็นเลขคี่ และบิตสุดท้าย บิตสิ้นสุดซึ่งมีสภาวะลอจิก "1"

ตำแหน่งขาสัญญาณของคีย์บอร์ด



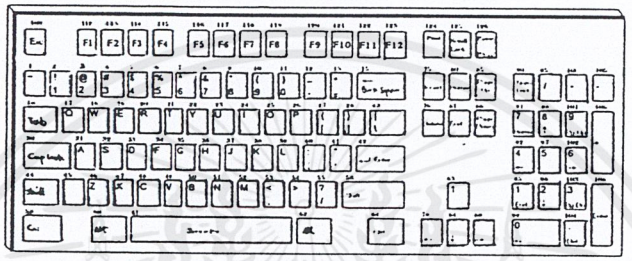
รูปที่ 2.14 คอนเน็คเตอร์แสดงขาสัญญาณของคีย์บอร์ด

จากรูป 2.14 เป็นคอนเน็คเตอร์ (Connector) แสดงขาสัญญาณของคีย์บอร์ด โดยมีรายละเอียดดังนี้

1. สัญญาณนาฬิกา (Keyboard Clock) เป็นขาสัญญาณที่สร้างความสอดคล้องระหว่างคีย์บอร์ดกับอุปกรณ์ที่นำมาเชื่อมต่อ ปกติจะมีสภาวะเป็นลอจิก "1" จะเป็นสัญญาณนาฬิกาเมื่อมีการใช้งานคีย์บอร์ด

2. สัญญาณข้อมูล (Keyboard Data) สภาวะปกติเป็นลอจิก “1” จะเป็นสัญญาณข้อมูลเมื่อมีการใช้งานคีย์บอร์ด เมื่อมีการกดคีย์บอร์ดข้อมูลที่ออกมาที่ขาสัญญาณนี้จะขึ้นอยู่กับคีย์ที่กด (รายละเอียดดังตาราง 2.9)
3. สัญญาณรีเซ็ต (Keyboard Reset) เป็นสัญญาณอินพุต ในการใช้งานปกติไม่ต้องต่อสัญญาณใดๆ
4. กราวน์ (Ground)
5. ไฟเลี้ยง 5 โวลต์ (Vcc)

การจัดวางตำแหน่งคีย์ของคีย์บอร์ด



รูปที่ 2.15 การจัดวางตำแหน่งของคีย์บอร์ด

จากรูปที่ 2.15 เป็นการจัดวางตำแหน่งคีย์ต่างๆ ของคีย์บอร์ด ซึ่งประกอบด้วยส่วนคีย์ที่เป็นตัวอักษรภาษาไทย ภาษาอังกฤษ และฟังก์ชันคีย์ โดยรายละเอียดของสัญญาณที่ได้เมื่อมีการกดคีย์ดังแสดงในตาราง 2.9

Key number	Make code	Break code	Key number	Make code	Break code
1	0EH	F0H 0EH	13	55H	F0H 55H
2	16H	F0H 16H	14	---	--- ---
3	1EH	F0H 1EH	15	66H	F0H 66H
4	26H	F0H 26H	16	0DH	F0H 0DH
5	25H	F0H 25H	17	15H	F0H 15H
6	2EH	F0H 2EH	18	1DH	F0H 1DH
7	36H	F0H 36H	19	24H	F0H 24H
8	3DH	F0H 3DH	20	2DH	F0H 2DH
9	3EH	F0H 3EH	21	2CH	F0H 2CH
10	46H	F0H 46H	22	35H	F0H 35H
11	45H	F0H 45H	23	3CH	F0H 3CH
12	4EH	F0H 4EH	24	43H	F0H 43H

Key number	Make code	Break code	Key number	Make code	Break code
25	44H	F0H 44H	57	59H	F0H 59H
26	4DH	F0H 4DH	58	14H	F0H 14H
27	54H	F0H 54H	59	---	--- ---
28	5BH	F0H 5BH	60	11H	F0H 11H
29	5DH	F0H 5DH	61	29H	F0H 29H
30	58H	F0H 58H	62	E0H 11H F0H	E0H 11H
31	1CH	F0H 1CH	63	--- --- ---	--- ---
32	1BH	F0H 1BH	64	E0H 14H F0H	E0H 14H
33	23H	F0H 23H	65	---	---
34	2BH	F0H 2BH	66	---	---
35	34H	F0H 34H	67	---	---
36	33H	F0H 33H	68	---	---
37	3BH	F0H 3BH	69	---	---
38	42H	F0H 42H	70	---	---
39	4BH	F0H 4BH	71	---	---
40	4CH	F0H 4CH	72	---	---
41	52H	F0H 52H	73	---	---
42	5DH	F0H 5DH	74	---	---
43	5AH	F0H 5AH	75	E0H 70H E0H	F0H 70H
44	12H	F0H 12H	76	E0H 71H E0H	F0H 71H
45	61H	F0H 61H	77	---	---
46	1AH	F0H 1AH	78	---	---
47	22H	F0H 22H	79	E0H 6BH	E0H F0H 6BH
48	21H	F0H 21H	80	E0H 6CH	E0H F0H 6CH
49	2AH	F0H 2AH	81	E0H 69H	E0H F0H 69H
50	32H	F0H 32H	82	--- ---	--- --- ---
51	31H	F0H 31H	83	E0H 75H	E0H F0H 75H
52	3AH	F0H 3AH	84	E0H 72H	E0H F0H 72H
53	41H	F0H 41H	85	E0H 7DH	E0H F0H 7DH
54	49H	F0H 49H	86	--- ---	--- --- ---
55	4AH	F0H 4AH	87	--- ---	--- --- ---
56	---	--- ---	88	E0H 7AH	E0H F0H 7AH

Key number	Make code	Break code	Key number	Make code	Break code
89	E0H 74H	E0H F0H 74H	109	---	--- ---
90	77H	F0H 77H	110	76H	F0H 76H
91	6CH	F0H 6CH	111	---	--- ---
92	6BH	F0H 6BH	112	05H	F0H 05H
93	69H	F0H 69H	113	06H	F0H 06H
94	--- ---	--- --- ---	114	04H	F0H 04H
95	E0H 95H	E0H F0H 95H	115	0CH	F0H 0CH
96	75H	F0H 75H	116	03H	F0H 03H
97	73H	F0H 73H	117	0BH	F0H 0BH
98	72H	F0H 72H	118	83H	F0H 83H
99	70H	F0H 70H	119	0AH	F0H 0AH
100	7CH	F0H 7CH	120	01H	F0H 01H
101	7DH	F0H 7DH	121	09H	F0H 09H
102	74H	F0H 74H	122	78H	F0H 78H
103	7AH	F0H 7AH	123	07H	F0H 07H
104	71H	F0H 71H	124	E0H 12H E0H 7CH	E0H F0H 7CH E0H F0H 12H
105	7BH	F0H 7BH	125	7EH	F0H 7EH
106	79H	F0H 79H	126	---	--- ---
107	---	--- ---			
108	E0H 5AH	F0H E0H 5AH			

ตาราง 2.9 แสดงข้อมูลที่ได้จากการกดคีย์บอร์ด

จากตาราง 2.9 แสดงข้อมูลที่ได้จากการกดคีย์ โดยข้อมูลจะขึ้นกับคีย์ที่กด โดยมีเงื่อนไขดังนี้

1. กดคีย์ 1 ครั้ง แล้วปล่อยคีย์จะ ได้ข้อมูลการกดคีย์ (Make Code) และข้อมูลการปล่อยคีย์ (Break Code)
2. กดคีย์ค้างไว้แล้วปล่อยคีย์ จะได้ข้อมูลของการกดคีย์ออกมาอย่างต่อเนื่องจนกว่าจะมีการปล่อยคีย์
3. กรณีที่มีการกดคีย์บอร์ด มากกว่า 1 คีย์ เช่น
   3.1 กดคีย์ Shift ค้างไว้ ตามด้วยคีย์ตัวอักษร “A” แล้วปล่อยคีย์ จะได้ข้อมูลของการกดคีย์ Shift ออกมาอย่างต่อเนื่องจนกว่าจะมีการกดคีย์ตัวอักษร “A” หลังจากนั้นข้อมูลการกดคีย์ Shift ก็จะหายไป โดยข้อมูลที่ได้จะเป็นข้อมูลการกดคีย์ “A”

3.2 เมื่อปล่อยคีย์ Shift หรือ "A" จะได้ข้อมูลของการปล่อยคีย์

3.3 กดคีย์ Shift และคีย์ของตัวอักษร "A" ค้างไว้ จะได้มีข้อมูล Shift ออกมาอย่างต่อเนื่องจนกว่าจะมีการกดคีย์ตัวอักษร "A" หลังจากนั้นข้อมูลการกดคีย์ Shift ก็จะหายไป โดยข้อมูลที่ได้จะเป็นข้อมูลของคีย์ตัวอักษร "A" ออกมาอย่างต่อเนื่องเช่นเดียวกัน เมื่อปล่อยคีย์ Shift หรือ "A" จะได้ข้อมูลของการปล่อยคีย์

2.7 คอนโทรล MSComm

สำหรับการใช้งาน Visual BASIC ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual BASIC จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยใน Visual BASIC เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCOMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCOMM32.OCX สำหรับการกับระบบปฏิบัติการ 32 บิต สำหรับใน Visual BASIC เวอร์ชัน 5 จะมีเพียง MSCOMM32.OCX เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทาง เพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communication) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชันต่างๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์อนุกรมและแอดเดรสของการเกิดอินเตอร์รัพต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติ (property) มากมาย แต่สามารถทำความเข้าใจได้ไม่ยาก ดังนี้

- CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1,COM2,COM3,COM4)

<u>รูปแบบการใช้งาน</u>

object.CommPort [=value]

โดย Value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึง อุปกรณ์

ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

- Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด, พาริตี้ , จำนวนของบิตข้อมูล, จำนวนของบิตปิดท้าย

รูปแบบการใช้งาน

object.Settings [=value]

ค่า Value มีชนิดข้อมูลเป็นแบบ String มีรูปแบบเป็น “BBBB,P,D,S” โดย BBBB เป็นค่าอัตราบอด, P เป็นค่าพาริตี้, D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600,N,8,1”

ค่าบอดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที

300 บิตต่อวินาที

600 บิตต่อวินาที

1200 บิตต่อวินาที

2400 บิตต่อวินาที

9600 บิตต่อวินาที (ค่าปกติ)

14,400 บิตต่อวินาที

19,200 บิตต่อวินาที

28,800 บิตต่อวินาที

38,400 บิตต่อวินาที (สงวน)

56,000 บิตต่อวินาที (สงวน)

128,000 บิตต่อวินาที (สงวน)

256,000 บิตต่อวินาที (สงวน)

สำหรับค่ามาตราฐานในการกำหนดค่าพาริตี้มีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่ (Even)
M	ลอจิก “1” (MARK)
N	ไม่ใช้ (ค่าปกติ)
O	พาริตี้คี่ (Odd)
S	ลอจิก “0” (Space)

ค่าที่ใช้กำหนดจำนวนบิตมี 5 ค่าคือ 4,5,6,7 และ 8 (เป็นค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่า คือ 1 (เป็นค่าปกติ), 1.5 และ 2

ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSComm1.Setting = “9600,N,8,1”

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด “” เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

- PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

object.PortOpen [=value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnable หรือ RTSE nable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

- Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

- InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

object.InBufferCount [=value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับเพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

หมายเหตุ อย่าสับสนระหว่างคำสั่ง InBufferSize และ InBufferCount คำสั่ง InBufferSize นั้นใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ

- InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งาน

object.InBufferSize [=value]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดภาคบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้ หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่นๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่า บัฟเฟอร์ภาครับที่น้อยเกินไปจะทำให้เกิดการโอเวอร์โฟลว์หรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลว์แล้วจึงค่อยปรับเพิ่มค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

- InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

object.InputLen [=value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ “0” การกำหนดค่าเท่ากับ “0” จะทำให้คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่าง (“”) กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InputBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลไว้แล้ว

- InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งาน

object.InputMode [=value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น “0” และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่นๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

- Output

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

object.Output [=value]

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte

- OutputBufferCount

คืนค่าของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งาน

object.OutBufferCount [=value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ “0”

หมายเหตุ ระวังการสับสนระหว่างคุณสมบัติ OutBufferCount กับ OutBufferSize ซึ่ง OutBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาคส่ง

- OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์

รูปแบบการใช้งาน

object.OutBufferSize [=value]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่งโดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้ มีหน่วยความจำเหลือให้ใช้งานน้อย แต่อย่างไรก็ตามถ้ากำหนดค่าน้อยเกินไป จะทำให้เกิดข้อมูลล้นบัฟเฟอร์ขึ้นได้ ยกเว้นจะมีการใช้ แฮนด์เช็ค วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองใช้ค่าเริ่มต้นคือค่า 512 ไบต์ดูก่อน ถ้าโปรแกรมทำงานแล้วเกิดการล้นของข้อมูลค่อยเพิ่มค่าของ OutBufferSize ให้มากขึ้น

- ParityReplace

กำหนดและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้

รูปแบบการใช้งาน

object.ParityReplace [=value]

บิตพาริตี้เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSComm จะทำการบวกบิตทุกบิตที่มีค่าลอจิก “1” ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก “1” เป็นเลขคู่หรือคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง (“”) จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อมูลผิดพลาด

ParityReplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใดๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0-255

- DTREnable

ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Ready (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน

<u>รูปแบบการใช้งาน</u>

object.DTREnable [=value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิกของขา DTR ให้เป็น “0” หรือ “1” โดย

True หมายถึง ให้ขา DTR มีลอจิก “1”

False หมายถึง ให้ขา DTR มีลอจิก “0” (เป็นค่าปกติ)

หมายเหตุ เมื่อขา DTR ถูกกำหนดสถานะให้เป็น True ที่ขา DTR จะมีสถานะลอจิก “1” เมื่อทำการเปิดพอร์ตและจะมีสถานะเป็น “0” เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น False ที่ขา DTR จะมีสถานะลอจิก “0” ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

- RTSEnable

ใช้เพื่อกำหนดสถานะลอจิกให้ขา Request To Send (RTS) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

<u>รูปแบบการใช้งาน</u>

object. RTSEnable [=value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิก “0” หรือ “1” ให้ขา RTS โดย

True หมายถึง ให้ขา RTS มีลอจิก “1”

False หมายถึง ให้ขา RTS มีลอจิก “0” (เป็นค่าปกติ)

หมายเหตุ เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1” เมื่อเปิดพอร์ตและมีสถานะลอจิก “0” เมื่อปิดพอร์ต

- EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file : EOF) ระหว่างการรับอินพุตเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF

<u>รูปแบบการใช้งาน</u>

object.EOFEnable [=value]

โดย value เป็นค่าสถานะ True หรือ False เพื่ออีนาเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

- CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีสถานะลอจิก "0" หรือ "1" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น "1" ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น "0"

รูปแบบการใช้งาน

object.CTSHolding

เมื่อขา CTS เป็นลอจิก "0" (CTSHolding=False) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCTSTO (Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

- CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิกเป็น "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก "1" ถ้า CDHolding เป็น False ขา DCD จะมีสถานะลอจิก "0"

รูปแบบการใช้งาน

object.CDHolding

เมื่อขา DCD มีลอจิก "1" (CDHolding=True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCDTO (Carrier Detect Timeout Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

- DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR ได้ว่ามีสถานะลอจิก "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิก "1" ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก "0"

รูปแบบการใช้งาน

object.DSRHolding

เมื่อขา DSR เป็นลอจิก "1" (DSRHolding=True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventDSRTO (Data Set Ready Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

- Handshaking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบการใช้งาน

object.Handshaking [=value]

ค่าตัวแปร Value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกันคือ

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก (เป็นค่าเริ่มต้น)
2. comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้ใช้แฮนด์เช็กแบบ XON/XOFF
3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request To Send/Clear To Send)
4. comRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send และ XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายใน ระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับเพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับโปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ Handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว์ (overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

- Break

ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean

<u>รูปแบบการใช้งาน</u>

object.Break [=value]

โดย Value เป็นค่าบูลีน ถ้า Value=True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า Value=False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False

- เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้น

2.8 ฐานข้อมูล

ฐานข้อมูล คือ การจัดเก็บข้อมูลที่มีความสัมพันธ์กันอย่างเป็นระเบียบ ซึ่งเราสามารถนำไปใช้ประโยชน์ได้ เช่น การเพิ่มข้อมูล การสร้างรายงานเกี่ยวกับข้อมูล และการแสดงผลข้อมูล เป็นต้น สำหรับระบบฐานข้อมูลนั้นมีตั้งแต่ระบบแบบเครื่องเดียวไปจนถึงระบบที่ทำงานกับเครือข่ายคอมพิวเตอร์

ใน VB6 สามารถเข้าถึงข้อมูลของโปรแกรม Microsoft Access ได้โดยตรง(รายละเอียดของการใช้งาน โปรแกรม Microsoft Access นั้นสามารถอ่านได้จากคู่มือเกี่ยวกับการใช้งานโปรแกรม Microsoft access) นอกจาก Microsoft Aceess แล้ว VB6 ยังสามารถเข้าถึงฐานข้อมูลที่สร้างด้วยโปรแกรมอื่นผ่านทาง ODVB (Open Database Connectivity) ซึ่งเป็นมาตรฐานในการติดต่อกับฐานข้อมูลรูปแบบต่างๆ

<u>คำศัพท์พื้นฐานที่ใช้ในทางฐานข้อมูล</u>

- SQL

ย่อมาจาก Structure Query Language เป็นชุดคำสั่งมาตรฐานที่ใช้ในการติดต่อกับข้อมูลในระบบฐานข้อมูล เช่น การเพิ่มข้อมูล การลบข้อมูล การเปลี่ยนแปลงข้อมูล เป็นต้น

- ตารางฐานข้อมูล

เป็นที่เก็บค่าฟิลด์ต่างๆ ของข้อมูลซึ่งจะอยู่ในรูปคอลัมน์ของตารางข้อมูลและแต่ละแถวจะเรียกเป็นเรคอร์ด

- Primary key

ใช้เป็นตัวแทนของเรคอร์ดนั้นๆ ซึ่งค่าของ primary key จะเป็นค่าที่มีค่าเดียวไม่ซ้ำกับเรคอร์ดอื่นๆ ในตารางฐานข้อมูลนั้นๆ

- คิวรี(Query)

เป็นการสอบถามข้อมูลด้วยคำสั่ง SQL ที่ เราต้องการจากฐานข้อมูลซึ่งจะได้ผลลัพธ์เป็นเรคอร์ดที่ตรงตามเงื่อนไขที่เราต้องการ

- เรคอร์ดเซต

เป็นกลุ่มของเรคอร์ดที่ได้จากการคิวรี คือจะเป็นเซตย่อยของข้อมูลทั้งหมดในฐานข้อมูล ซึ่งเราสามารถนำไปประมวลผลข้อมูลต่อไปได้

<u>การจัดการฐานข้อมูลด้วย Data Control</u>

เราจะมาเริ่มต้นสร้างโปรแกรมที่ติดต่อกับฐานข้อมูล ซึ่งในไฟล์ฐานข้อมูลนี้จะมีตารางข้อมูลชื่อต่างๆ ในความสัมพันธ์แบบ One-to-Many

โดยที่ใน VB6 จะมี Data Control ซึ่งจะช่วยเราในการติดต่อกับข้อมูลจากไฟล์ฐานข้อมูลโดยแทบจะไม่ต้องเขียนคำสั่งของโปรแกรมเลย Data Control จะอยู่ในทูลบ็อกซ์

<u>การเชื่อมกับไฟล์ฐานข้อมูลด้วย Data Control</u>

ขั้นตอนแรกในการสร้างโปรแกรมฐานข้อมูล เราต้องเชื่อมต่อกับฐานข้อมูลนั้นก่อน ซึ่งใน Data Control จะเชื่อมต่อผ่านทางคุณสมบัติต่างๆดังนี้

คุณสมบัติ	รายละเอียด
Connect	กำหนดชนิดไฟล์ฐานข้อมูลของโปรแกรมตัวใดที่เรากำลังเชื่อมต่อ เช่น ในโปรแกรมตัวอย่างที่เราเขียน จะทำงานกับไฟล์ฐานข้อมูลที่สร้างด้วยโปรแกรม Microsoft Access ดังนั้นเราจะต้องกำหนดคุณสมบัติ Connect เป็น Access นอกจากนี้ยังมีโปรแกรมฐานข้อมูลตัวอื่นๆอีก เช่น Dbase,Foxpro และ Excel ที่เราเลือกติดต่อได้
Database Name	กำหนดไฟล์ฐานข้อมูลที่จะใช้ โดยต้องระบุครบทั้ง Path ที่เก็บไฟล์นั้นด้วย เช่น "C:\ProgremFiles\DevStudio\Vb\Biblio.MDB" เป็นต้น หรืออาจจะเลือกจากไดอะล็อกซ์บ็อกซ์ก็ได้

การกำหนด RecordSet ที่ใช้ใน Data Control

คุณสมบัติ RecordSource กำหนดตารางข้อมูลในฐานข้อมูลที่จะนำมาใช้ ซึ่งอาจอยู่ในรูปคำสั่ง SQL หรือคิวรีก็ได้

การกำหนดชนิดของ RecordSet

เราสามารถกำหนดชนิดของ RecordSet ได้ เพื่อทำให้โปรแกรมของเราทำงานได้เร็วขึ้นด้วยคุณสมบัติ RecordSetType ซึ่งมีตัวเลือกอยู่ 3 ชนิดคือ

ชนิด Table	เป็นชนิดของ RecordSet ที่แทนตาราง ซึ่ง RecordSet ชนิดนี้จะมีความเร็วในการค้นหาข้อมูลดีที่สุดเนื่องจากใช้ Index เป็นฟิลด์พิเศษช่วยในการค้นหาข้อมูล
ชนิด Dynaset	เป็นชนิดของ RecordSet ที่สามารถสร้างจากตารางเดียว หรือหลายตารางก็ได้และเราสามารถแก้ไขค่าของข้อมูลได้
ชนิด Snapshot	จะเหมือนกับ Dynaset แต่แตกต่างจาก Dynaset ตรงที่ RecordSet ชนิดนี้จะทำงานได้ดีกว่า แต่ไม่สามารถแก้ไขค่าของข้อมูลได้

การกำหนดให้ฐานข้อมูลไม่สามารถ Share ได้

เราสามารถล็อคไฟล์ฐานข้อมูลที่ Data Control ใช้อยู่ เพื่อไม่ให้ผู้อื่นใช้ได้ โดยกำหนดคุณสมบัติ Exclusive ให้เป็น true

กำหนดให้ฐานข้อมูลไม่สามารถแก้ไขได้

เราสามารถทำให้ไฟล์ฐานข้อมูลที่ Data Control ใช้อยู่อ่านได้เพียงอย่างเดียวไม่สามารถแก้ไขได้ โดยการกำหนดให้คุณสมบัติ ReadOnly ให้เป็น True

การตรวจสอบตำแหน่งเรคอร์ดปัจจุบัน

เราสามารถตรวจสอบตำแหน่งเรคอร์ดปัจจุบันในตารางข้อมูลได้จากคุณสมบัติ BOF ซึ่งจะมีค่าเป็น True เมื่อเราอยู่ที่เรคอร์ดเริ่มต้นของตาราง และคุณสมบัติ EOF ซึ่งจะมีค่าเป็น True เมื่อเราอยู่ที่เรคอร์ดสุดท้ายของตาราง

การแสดงผลจาก Data Control

Data Control จะทำหน้าที่ในการติดต่อกับฐานข้อมูล ซึ่งเราต้องใช้คอนโทรลต่างๆต่อไปนี้ร่วมกับ Data Control เพื่อแสดงข้อมูลในฐานข้อมูลบนฟอร์ม

- Image - Label - PictureBox

- TextBox - ListBox - CheckBox

- ComboBox

ทุกคอนโทรลที่กล่าวมาทั้งหมดนี้ จะมีคุณสมบัติที่สำคัญที่เราจำเป็นต้องทราบในการทำงานทางด้านฐานข้อมูล ดังต่อไปนี้

คุณสมบัติ	รายละเอียด
DataSource	ใช้กำหนดชื่อ Data Control ที่จะติดต่อด้วย
ComboBox	ใช้กำหนดว่าฟิลด์ใดใน RecordSet ของ Data Control ที่จะนำมาแสดงในคอนโทรลนั้น

2.9 โปรแกรมสร้างเวบเพจด้วย ASP

การเขียนเวบเพจด้วย ASP นั้นจะมีข้อดีมากกว่าการเขียนด้วยภาษา HTML ซึ่ง ASP นั้นเราสามารถเพิ่มความสามารถให้กับเวบเพจได้อย่างมากมาย โดยที่ HTMLนั้นทำไม่ได้ เวบเพจที่มี ASP อยู่นั้น จะมีลักษณะเป็นแบบไดนามิก คือ สามารถติดต่อกับผู้ใช้งานได้อย่างสะดวก สามารถใช้ติดต่อกับฐานข้อมูลต่างๆ ได้เป็นอย่างดี และยังทำให้เวบเพจนั้นทำงานได้โดยอัตโนมัติอีกด้วย

<u>การติดต่อกับฐานข้อมูล</u>

ก่อนที่เราจะติดต่อกับฐานข้อมูลได้นั้น เราต้องเซ็ต ODBC ที่เครื่องเสียก่อน หลังจากนั้นก็ใช้คำสั่งที่ติดต่อกับฐานข้อมูล เช่น

```
Set Conn = Sever.CreateObject("ADODB.Connection")
Conn.Open "DRIVER=Microsoft Access Driver (*.mdb);DBQ="_
& Sever.MapPath("/ชื่อ user/database/filedatabase.mdb")
```

จากคำสั่งข้างต้น จะเป็นการติดต่อกับฐานข้อมูล โปรแกรม Microsoft Access

หลังจากติดต่อกับฐานข้อมูลได้แล้วนั้น เราต้องเปิดชุดข้อมูล ซึ่งเรียกว่า RecortSet ขึ้นมาจากฐานข้อมูลนั้น เพื่อนำชุดข้อมูลนั้นมาใช้งาน โดยใช้คำสั่ง

```
rs.open " Select * from databasename",conn,1,3
```

และใช้คำสั่ง rs.close เพื่อปิดชุดข้อมูลนั้น

และนอกจากนี้ ยังมีคำสั่งที่สำคัญๆ เพื่อใช้ในการปรับปรุง แก้ไข และใช้งานข้อมูลของฐานข้อมูลอีกได้แก่

1. การใช้คำสั่ง INSERT

การเพิ่มเรคอร์ดใหม่ลงในตารางโดยใช้ชุดคำสั่ง SQL สามารถทำได้โดยใช้คำสั่ง INSERT เพื่อเพิ่มข้อมูลเพียงคอลัมน์เดียว หรือหลายคอลัมน์ก็ได้ โดยที่การวางตำแหน่งของคอลัมน์และค่าที่ต้องการใส่จะต้องอยู่ในลำดับเดียวกันภายในวงเล็บ ซึ่งมีรูปแบบคำสั่งในการใช้งานดังนี้

```
INSERT INTO TableName (column1, column2, column3)
VALUES (value1, value2, value3)
```

การใช้คำสั่ง INSERT เพื่อเพิ่มเรคอร์ดลงในตารางจะเป็นการเพิ่มครั้งละหนึ่งเรคอร์ดต่อการทำงานหนึ่งครั้ง ซึ่งสามารถนำลูปมาใช้เพื่อเพิ่มเรคอร์ดใหม่หลายๆ เรคอร์ดพร้อมกันได้

สำหรับคอลัมน์ที่ไม่มีการระบุหลังจากการใช้งานคำสั่ง INSERT จะมีค่าเริ่มต้นตามที่ได้กำหนดไว้ในฐานข้อมูล (default) หรือเป็น NULL ถ้าไม่มีการกำหนดไว้

2. การใช้คำสั่ง UPDATE

คำสั่ง UPDATE สามารถอัปเดตหลายเรคอร์ด และ/หรืออัปเดตหลายฟิลด์ในการทำงานครั้งเดียวได้ โดยการใช้ WHERE ในการระบุเงื่อนไข โดยมีรูปแบบของการใช้งานดังนี้

```
UPDATE TableName SET Column1 = Value1, Column2 = Value2
WHERE criteria
```

โดยที่ criteria พารามิเตอร์ของ WHERE จะเป็นค่าที่ใช้ในการจำกัดจำนวนของเรคอร์ดที่จะทำการอัปเดต เช่น WHERE Dept = 'Sales' AND Age BETWEEN 18 AND 30 เป็นต้น

3. การใช้คำสั่ง DELETE

การลบข้อมูลออกจากฐานข้อมูลสามารถทำได้หลายวิธี โดยอาจเป็นการลบเรคอร์ดเดียว หรือหลายเรคอร์ดออกจากตาราง หรืออาจเป็นการลบหลายเรคอร์ดออกจากหลายๆ ตาราง การ DELETE เรคอร์ดต้องใช้อย่างระมัดระวัง เนื่องจากไม่สามารถกู้เรคอร์ดที่ลบแล้วคืนได้ ซึ่งลักษณะการใช้งานของคำสั่ง DELETE จะเป็นดังนี้

```
DELETE Column*FROM TableName WHERE criteria
```

<u>ส่วนของคำสั่งชนิด Script</u>

ในคำสั่งส่วนนี้จะทำให้เวบเพจนั้นทำงานได้อย่างอัตโนมัติ และทำให้การเขียนโปรแกรมสั้นลงได้อย่างมาก การทำงานในส่วนนี้จะใช้การประมวลผลทางตรรกศาสตร์เป็นหลัก

คำสั่งที่สำคัญและใช้งานบ่อยๆ ได้แก่

Do…Loop

For…Next

If…then…Else

Select Case

เป็นต้น

บทที่ 3
การออกแบบและการสร้าง

ในโครงงานนี้จะแบ่งการทำงานออกเป็น 2 ส่วนหลักๆ คือ ส่วนของเครื่องไคลเอ็นต์ (Client) และส่วนของเครื่องเซิฟเวอร์ (Server) ซึ่งจะกล่าวถึง หน้าที่ การทำงานและอุปกรณ์ต่างๆ ของทั้ง 2 ส่วน ดังนี้

3.1 เครื่องไคลเอ็นต์ (Client)

หน้าที่หลักๆ ของเครื่องไคลเอ็นต์นี้ ก็คือ ทำการติดต่อกับผู้ใช้งาน และรับ – ส่ง ข้อมูลกับเครื่องเซิฟเวอร์ ดังแสดงในรูปที่ 3.1 ซึ่งในรูปนี้จะแสดงถึงบล็อคไดอะแกรมของเครื่องไคลเอ็นต์ทั้งหมด

ส่วนประกอบของเครื่องไคลเอ็นต์นี้ ได้แก่

1.ส่วนการติดต่อกับผู้ใช้งาน

หน้าที่ของส่วนนี้ คือ รับคำสั่งจากผู้ใช้งาน และแสดงผลการทำงานของเครื่องไคลเอ็นต์ให้ผู้ใช้งานได้ทราบ โดยการรับข้อมูลจากผู้ใช้งานจะใช้คีย์บอร์ดของเครื่องคอมพิวเตอร์ ซึ่งข้อดีของคีย์บอร์ดชนิดนี้ก็คือ สามารถคีย์ข้อมูลได้หลากหลาย และมีการสั่งงานได้หลายคำสั่ง เช่น ข้อมูลที่เป็นทั้งตัวอักษรภาษาอังกฤษและตัวอักษรภาษาไทย คำสั่งต่าง เช่น Esc Tab Backspace Enter เป็นต้น ส่วนอุปกรณ์ที่ใช้แสดงผลการทำงานของเครื่องไคลเอ็นต์ให้ผู้ใช้ได้ทราบ คือ จอแอลซีดี ซึ่งในโครงงานนี้จะใช้จอแอลซีดีประเภทจอกราฟฟิก สาเหตุที่ต้องใช้จอแอลซีดีประเภทนี้ก็เพราะว่า จอแอลซีดีประเภทตัวอักษรนั้นไม่สามารถแสดงผลเป็นตัวอักษรภาษาไทยได้นั่นเอง แต่จอแอลซีดีกราฟฟิกนั้น ก็มีข้อเสียอยู่มากคือ การสั่งงานและการควบคุมนั้นจะยากและซับซ้อนกว่าจอแอลซีดีประเภทตัวอักษรอยู่มากและมีราคาสูงกว่าจอแอลซีดีอยู่หลายเท่า

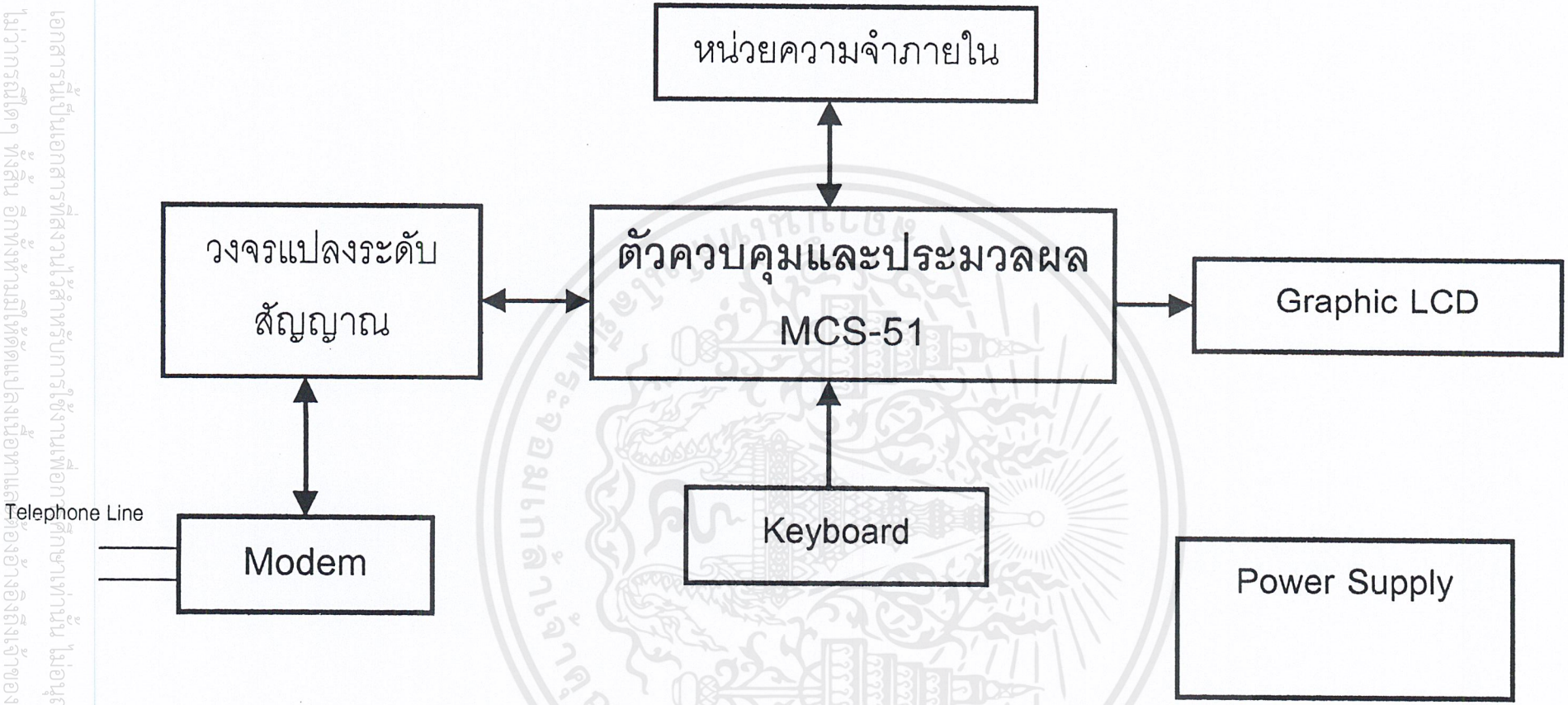
2.ส่วนรับ – ส่งข้อมูลกับเครื่องเซิฟเวอร์

อุปกรณ์ที่ใช้ในการรับ – ส่งข้อมูลในส่วนนี้นั้นเราจะใช้โมเด็มเป็นอุปกรณ์รับ – ส่งข้อมูลกับเครื่องเซิฟเวอร์ โดยผ่านโครงข่ายของระบบโทรศัพท์

3.ส่วนประมวลผลและควบคุมการทำงานของเครื่องไคลเอ็นต์

ในเครื่องไคลเอ็นต์นั้นจะใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นอุปกรณ์ควบคุมและประมวลผล ซึ่งไมโครคอนโทรลเลอร์ที่ใช้ในโครงงานนี้จะใช้ไมโครคอนโทรลเลอร์เบอร์ 89C52

เนื่องจากเราใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและตัวควบคุมการทำงานทั้งหมดของเครื่องไคลเอ็นต์ ดังนั้นไมโครคอนโทรลเลอร์นี้จะเชื่อมต่อกับอุปกรณ์ทั้งหมดภายในเครื่องไคลเอ็นต์ ได้แก่ จอแอลซีดี คีย์บอร์ด และโมเด็ม ซึ่งหลักการทำงานของส่วนประมวลผลนี้ คือ รับข้อมูลจากผู้ใช้งานทางคีย์บอร์ดแล้วทำการแปลงสัญญาณที่ส่งมาทางคีย์บอร์ด ทำการประมวลผลการส่งข้อมูลนั้นแล้วแสดงผลการกดคีย์นั้นออกทางจอแอลซีดี และทำการส่งข้อมูลของผู้ใช้งานไปยังเครื่องเซิฟเวอร์ โดยผ่านโมเด็ม



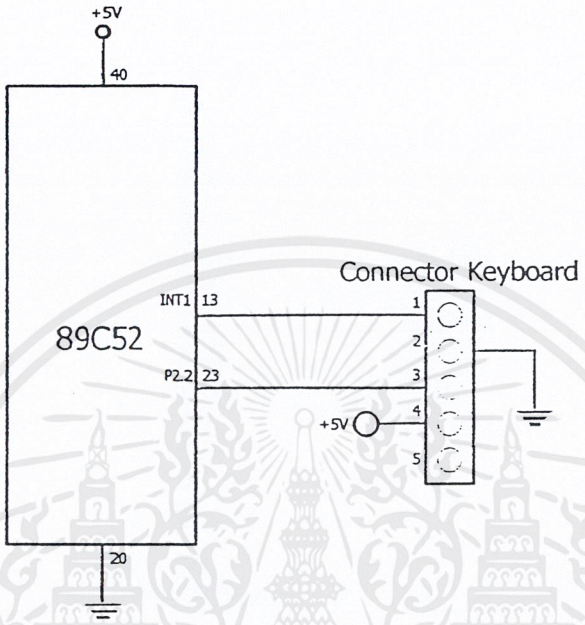
รูปที่3.1 บล็อคไดอะแกรมแสดงวงจรรวมที่ไคลเอนท์ (Client)

ส่วนออกแบบเครื่องไคลเอ็นต์นั้นจะแบ่งเป็น 2 ส่วน คือ

3.1.1 ส่วน Hardware

ในส่วนนี้จะมีวงจรต่างๆ ดังนี้

1. วงจรรับข้อมูลจากผู้ใช้งาน



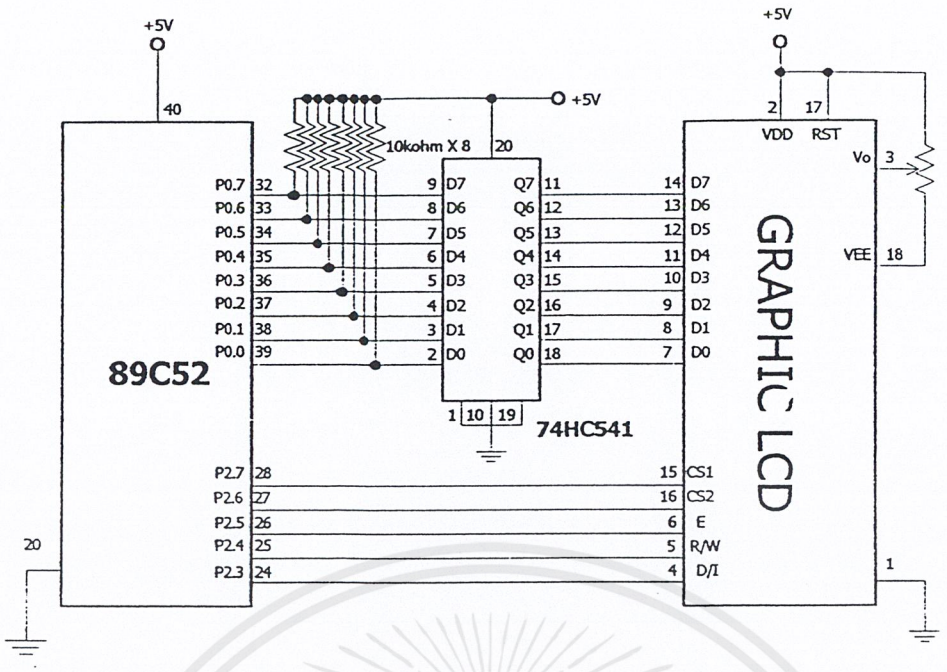
รูปที่ 3.2 แสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคีย์บอร์ด

จากรูปจะเห็นว่าคีย์บอร์ดจะเชื่อมต่อกับไมโครคอนโทรลเลอร์ได้โดยตรง โดยที่คีย์จะส่งข้อมูลผ่านขา data (ขา 3) เข้าสู่ไมโครคอนโทรลเลอร์ที่พอร์ต 2.2 (ขา 23) ซึ่งการส่งข้อมูลของคีย์บอร์ดนั้นจะเป็นการส่งข้อมูลแบบซิงโครนัส โดยที่คีย์บอร์ดจะส่งสัญญาณนาฬิกาออกมาที่ขา clk (ขา1) ด้วย และไมโครคอนโทรลเลอร์จะต้องเอาสัญญาณนาฬิกานี้เข้ามาประมวลผลด้วย เพื่อทำการดีเทคข้อมูลที่ส่งมาจากคีย์บอร์ด

2. วงจรแสดงผล

จากที่กล่าวมาแล้วในข้างต้น อุปกรณ์แสดงผลนั้นเราจะใช้จอแอลซีดีกราฟฟิกเป็นอุปกรณ์แสดงผล ซึ่งการเชื่อมต่อจอแอลซีดีกับไมโครคอนโทรลเลอร์นั้นจะมีลักษณะการต่อดังรูปที่ 3.3

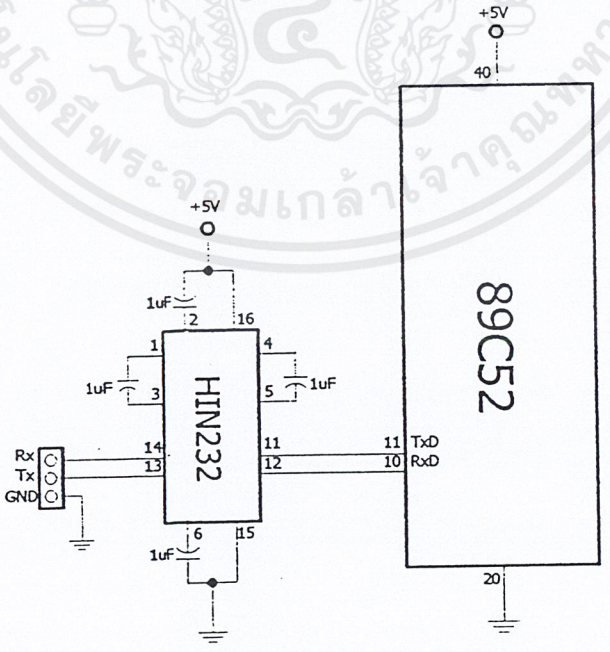
ซึ่งการสั่งงานของจอแอลซีดีนั้นจะมีสัญญาณอยู่ 2 ประเภท คือ สัญญาณข้อมูล และสัญญาณควบคุม สัญญาณข้อมูลของจอแอลซีดีนั้นจะถูกส่งผ่านพอร์ต 0 ของไมโครคอนโทรลเลอร์ ซึ่งการส่งข้อมูลนี้ จะเป็นการส่งแบบขนาน 8 บิต โดยผ่านตัวแลตช์ เพื่อให้ข้อมูลจากไมโครคอนโทรลเลอร์ไปถึงจอแอลซีดีพร้อมกัน ซึ่งสัญญาณข้อมูลนี้จะมีสัญญาณ 2 ประเภท คือ ข้อมูล และ แอดเดรส ส่วนสัญญาณควบคุมนั้น ก็จะมี สัญญาณอ่าน-เขียน สัญญาณข้อมูล-คำสั่ง สัญญาณอีนาเบิล ฯลฯ ซึ่งสัญญาณควบคุมนั้นจะต่อกับไมโครคอนโทรลเลอร์โดยตรง ดังแสดงในรูป 3.3



รูปที่ 3.3 แสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์และจอแอลซีดี

3. วงจรรับ – ส่งข้อมูลกับเครื่องเซิฟเวอร์

ในโครงงานนี้จะใช้โมเด็มเป็นอุปกรณ์รับ – ส่ง ข้อมูล ซึ่งการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับโมเด็มนั้น จำเป็นต้องมีวงจรแปลงระดับสัญญาณเป็นวงจรเชื่อมต่อเสียก่อน เพราะว่าไมโครคอนโทรลเลอร์กับโมเด็มนั้นมีระดับสัญญาณที่แตกต่างกัน ซึ่งวงจรแปลงระดับสัญญาณสำหรับการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับโมเด็มนั้นจะแสดงในรูปที่ 3.4



รูปที่ 3.4 แสดงการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์และโมเด็ม ผ่านวงจรแปลงระดับสัญญาณ

และสำหรับโมเด็มนั้นจะต้องต่อแบบ null modem เพื่อให้โมเด็มสามารถใช้งานได้ และการสั่งงานของโมเด็มโดยไมโครคอนโทรลเลอร์นั้นจะใช้รหัสแอสกี

3.1.2 ส่วน Software

ในการออกแบบส่วนของ Software นั้น เนื่องจากเราใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและควบคุม ดังนั้นการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์นั้นต้องใช้ภาษาแอสเซมบลี และจากที่กล่าวไปแล้วในส่วนของ Hardware นั้นไม่มีวงจรที่ซับซ้อนเลย การเชื่อมต่อของอุปกรณ์ต่างๆ นั้น ก็สามารถทำได้ง่าย และมีอุปกรณ์ไม่มาก ดังนั้นในส่วนของ Software นั้นจะมีความซับซ้อนและยุ่งยากมากขึ้น เพื่อทำให้อุปกรณ์ต่างๆ สามารถทำงานประกอบกันได้ อย่างถูกต้อง ซึ่งสาเหตุหลักๆ ที่ทำให้การเขียนโปรแกรมมีความซับซ้อนก็คือ อุปกรณ์ทั้ง 3 ชนิดที่ไมโครคอนโทรลเลอร์เชื่อมต่อซึ่งได้แก่ คีย์บอร์ด จอแอลซีดี และโมเด็ม นั้นมีการสั่งงานไม่เหมือนกัน ทำให้ต้องแปลงข้อมูลเพื่อเชื่อมต่ออุปกรณ์ทั้ง 3 อยู่ตลอด ทำให้โปรแกรมมีขนาดใหญ่และซับซ้อนนั่นเอง ซึ่งในการออกแบบโปรแกรมในส่วนนี้จะขอกล่าวแยกการเชื่อมต่อกับอุปกรณ์ต่างๆ ดังนี้

1. ส่วนรับข้อมูลจากผู้ใช้งานทางคีย์บอร์ด

ในการกดคีย์แต่ละครั้งนั้น คีย์บอร์ดจะส่งสัญญาณออกมา 2 สัญญาณคือ สัญญาณข้อมูลและสัญญาณนาฬิกา ซึ่งการส่งข้อมูลของคีย์บอร์ดจะเป็นการส่งข้อมูลแบบซิงโครนัส ดังนั้นการที่เราจะดีเทคสัญญาณข้อมูลที่ส่งมาได้นั้น เราต้องนำสัญญาณข้อมูลมาเทียบกับสัญญาณนาฬิกา เพื่อดีเทคสัญญาณข้อมูลออกมาได้ โดยการเชื่อมต่อระหว่างคีย์บอร์ดและไมโครคอนโทรลเลอร์จะมีลักษณะการเชื่อมต่อ ดังแสดงในรูปที่ 3.2 ซึ่งจากรูปเราจะนำสัญญาณนาฬิกาจากคีย์บอร์ดเข้าทางขา 13 ของไมโครคอนโทรลเลอร์ ซึ่งเป็นขาสัญญาณอินเทอร์รับท์จากภายนอกและนำสัญญาณข้อมูลเข้าทางขา 23 ของไมโครคอนโทรลเลอร์ และการส่งสัญญาณข้อมูลจากคีย์บอร์ดนั้นจะเป็นการส่งข้อมูลแบบ 9 บิต โดยที่บิตที่ 9 จะเป็นพาริตี้บิต ประเภทพาริตี้คี่ และเมื่อในขณะที่ไม่มีการส่งสัญญาณนั้น ระดับสัญญาณที่ออกจากขา data ของคีย์บอร์ดจะมีระดับสัญญาณเป็น 1 ตลอด ส่วนสัญญาณที่ได้จากสัญญาณนาฬิกานั้นสัญญาณที่ได้จะมีลักษณะเป็นสัญญาณสี่เหลี่ยมต่อเนื่องกันไป แต่ว่าสัญญาณที่ส่งออกมานั้นจะเกิดเมื่อมีการกดคีย์เท่านั้น และในขณะที่ไม่มีการกดคีย์ใดๆนั้น ระดับสัญญาณจากขา clk นั้นจะมีระดับสัญญาณเป็น 1 ตลอด ส่วนรหัสการกดคีย์นั้นจะเป็นรหัสเฉพาะสำหรับคียบอร์ดเท่านั้น ซึ่งการเขียนโปรแกรมเพื่อให้ไมโครคอนโทรลเลอร์รับทราบข้อมูลที่กดมานั้น เราต้องเขียนตารางรหัสคีย์บอร์ดขึ้นมาเพื่อให้ไมโครคอนโทรลเลอร์สามารถเข้าคีย์นั้นได้ เช่น เมื่อเรากดคีย์ A คีย์บอร์ดจะส่งสัญญาณข้อมูลมายังไมโครคอนโทรลเลอร์เป็นรหัส 1CH ออกมา ดังนั้นเราต้องมีตารางแสดงความสัมพันธ์ระหว่างรหัสคีย์บอร์ดกับรหัสแอสกี เพื่อให้ไมโครคอนโทรลเลอร์ทราบว่า รหัส1CH มีค่าเท่ากับ รหัส 41H ของรหัสแอสกีนั่นเอง

2. ส่วนแสดงผลการทำงาน

เนื่องจากเราใช้จอแอลซีดีประเภทกราฟฟิกเป็นตัวแสดงผลข้อมูล ดังนั้นการแสดงตัวอักษรขึ้นมาสักตัวเราจำเป็นต้องส่งข้อมูลจำนวนมากไปยังจอแอลซีดี เพราะว่าเราต้องขียนจุดจำนวนมากในจอแอลซีดี เพื่อให้แสดงเป็นตัวอักษร 1 ตัว เช่น สมมติว่า 1 บรรทัดมี 8 ช่อง และในแต่ละตัวอักษรนั้นต้องการความกว้างของตัวอักษรจำนวน 6 ช่อง ดังนั้นเราต้องส่งข้อมูลให้จอแอลซีดีถึง 48 บิต นั่นเอง ตัวอย่างเช่นการเขียนตัวอักษร " A " ขึ้นมาเราต้องส่งข้อมูล 00H,0FCH,22H,22H,22H,0FCH ออกไปที่จอแอลซีดี การเขียนโปรแกรมในส่วนนี้จะแสดงขั้นตอนการทำงานของเครื่องลูกให้ผู้ใช้ได้ทราบ เช่น แจ้งให้ผู้ใช้ใส่เบอร์โทรศัพท์ที่ต้องการหมุนลงไป

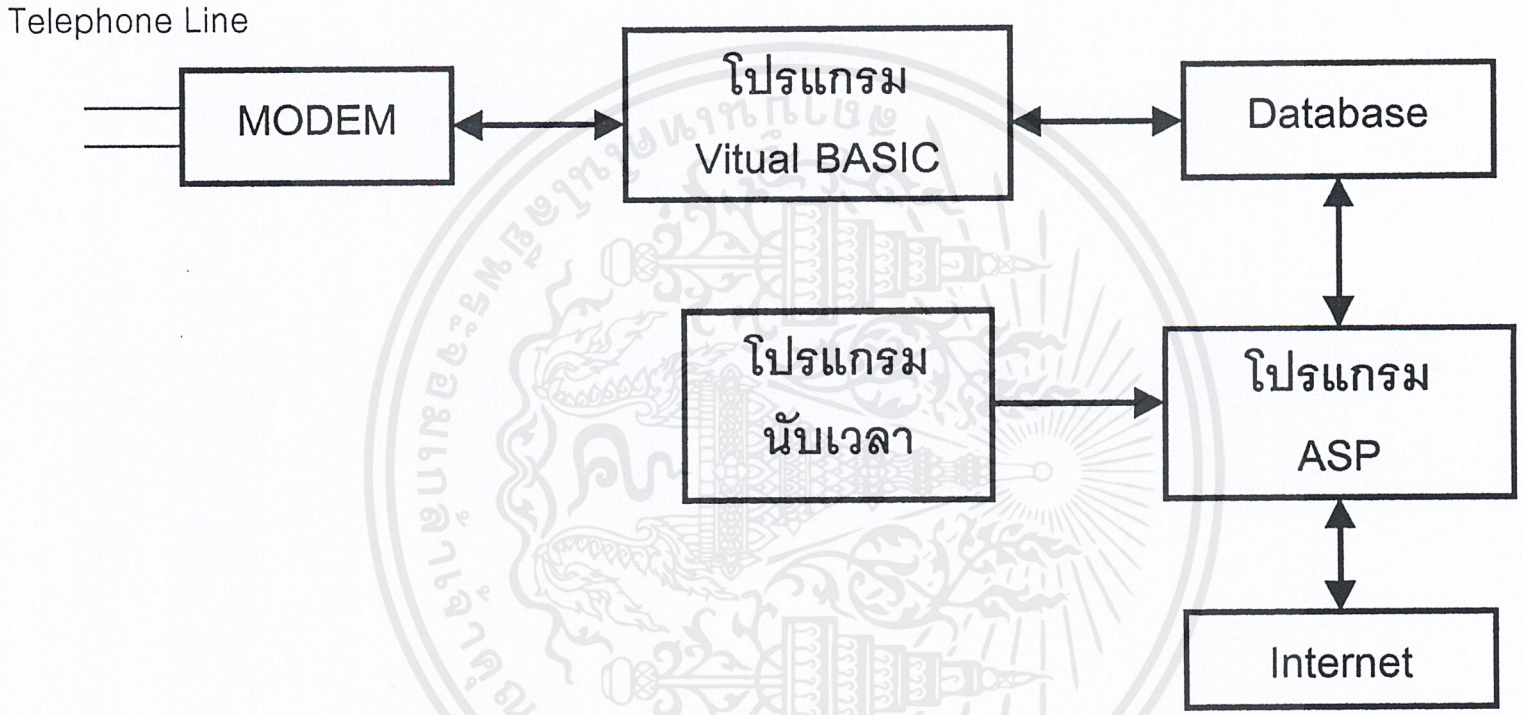
3. ส่วนรับ – ส่งข้อมูลกับเครื่องเซิฟเวอร์

ในการสั่งงานโมเด็มนั้นจะใช้รหัสแอสกีเป็นสื่อกลาง และการส่งคำสั่งให้โมเด็มนั้นต้องใช้ คำสั่ง AT (AT Command) ซึ่งจะดูรายละเอียดของคำสั่ง AT ได้จากบทที่ 2

และในการส่งข้อมูลจากผู้ใช้งานไปยังเครื่องเซิฟเวอร์นั่น เราจำเป็นต้องทำการใส่ส่วนของรหัสเริ่มต้นและรหัสปิดท้ายลงไปกับข้อมูลด้วย เพื่อให้เครื่องเซิฟเวอร์แยกประเภทของข้อมูลแต่ละชนิดได้ โดยที่รหัสเริ่มต้นของชนิดข้อมูล 'To' จะเป็นรหัส 128 (ฐานสิบของรหัสแอสกี) รหัสเริ่มต้นของชนิดข้อมูล 'From' จะเป็นรหัส 239 (ฐานสิบของรหัสแอสกี) รหัสเริ่มต้นของชนิดข้อมูล 'Subject' จะเป็นรหัส 250 (ฐานสิบของรหัสแอสกี) และรหัสเริ่มต้นของชนิดข้อมูล 'Message' จะเป็นรหัส 251 (ฐานสิบของรหัสแอสกี) ส่วนรหัสปิดท้ายของทุกชนิดข้อมูลนั้นจะใช้รหัสเดียวกันหมด คือใช้รหัส 149 (ฐานสิบของรหัสแอสกี) เป็นรหัสปิดท้าย

3.2 เครื่องเซิฟเวอร์ (Server)

หน้าที่หลักๆ ของเครื่องเซิฟเวอร์ คือรับข้อมูลจากเครื่องไคลเอ็นต์ และส่งข้อมูลให้อยู่ในรูปของอีเมลล์ไปยังปลายทาง แต่ในทางปฏิบัตินั้นเราไม่สามารถส่งอีเมลล์ไปยังปลายทางได้โดยตรง ซึ่งวิธีการที่ทำได้ คือ เมื่อเครื่องไคลเอ็นต์ต้องการส่งอีเมลล์ เครื่องไคลเอ็นต์ก็จะทำการติดต่อมายังเครื่องเซิฟเวอร์โดยผ่านโมเด็ม ซึ่งที่เครื่องไคลเอ็นต์ก็จะมีโมเด็ม 1 ตัวเพื่อเชื่อมต่อมายังเครื่องเซิฟเวอร์ ซึ่งที่เครื่องเซิฟเวอร์นั้นก็ต้องมีโมเด็มอีก 1 เครื่องเพื่อรอรับการติดต่อจากเครื่องไคลเอ็นต์ ซึ่งที่เครื่องเซิฟเวอร์นี้เราจะใช้เครื่องคอมพิวเตอร์เป็นตัวประมวลผลและควบคุมการทำงานที่เครื่องเซิฟเวอร์ โดยการรับข้อมูลจากเครื่องไคลเอ็นต์โดยผ่านโมเด็มนั้น ที่เครื่องเซิฟเวอร์จะใช้โปรแกรม VB เป็นตัวสั่งการโมเด็ม และนำข้อมูลที่ได้เก็บลงฐานข้อมูล ซึ่งในโครงงานนี้จะใช้โปรแกรม Microsoft Access เป็นฐานข้อมูล และใช้โปรแกรมภาษา ASP ในการดึงข้อมูลจากฐานข้อมูลขึ้นมาเพื่อส่งไปยังเมลล์เซิฟเวอร์เพื่อให้เมลล์เซิฟเวอร์ส่งอีเมลล์ฉบับนั้นไปยังปลายทาง ดังแสดงในรูปที่ 3.5 ซึ่งจะเห็นได้ว่าการทำงานที่เครื่องเซิฟเวอร์นั้นจะมีการทำงานอยู่หลายส่วนด้วยกัน ซึ่งจะขอกล่าวการทำงานอย่างละเอียดในแต่ละส่วน ดังนี้

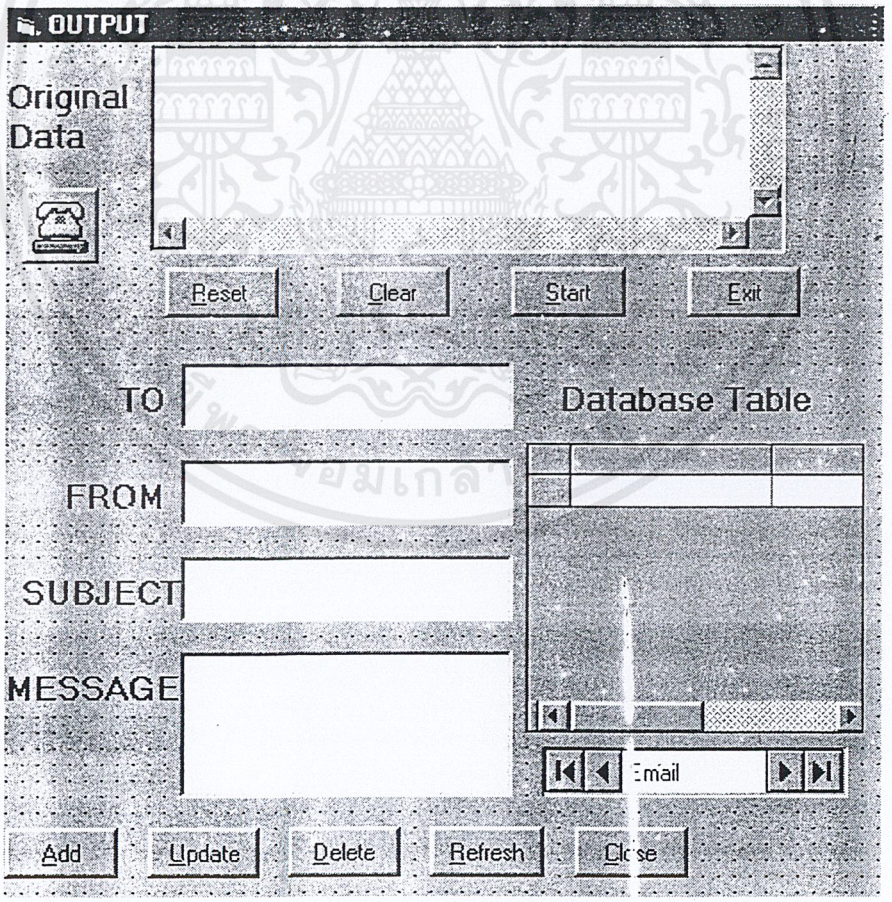


รูปที่3.5 แสดงบล็อคไดอะแกรมการทำงานของเครื่องเซิฟเวอร์ (Server)

3.2.1 ส่วนโปรแกรมภาษา Visual BASIC

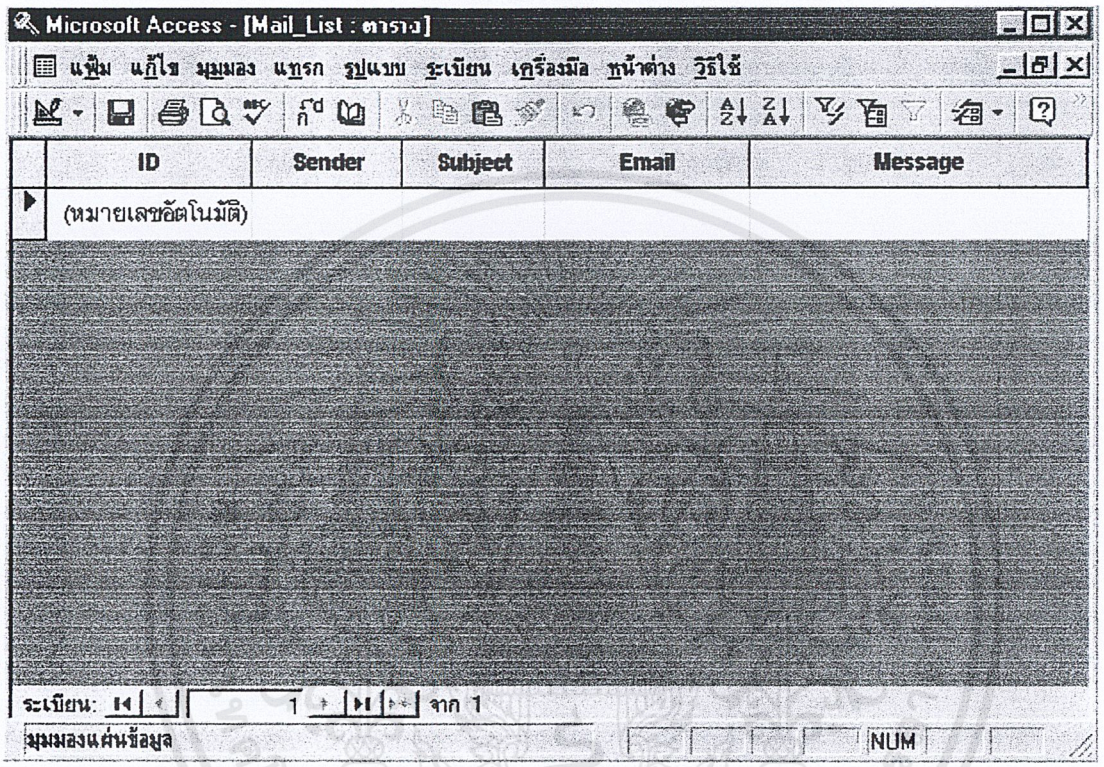
สำหรับในส่วนของการเขียนโปรแกรมเพื่อติดต่อกับโมเด็ม โดยผ่านทางพอร์ตอนุกรม (Serial Port) นั้น เราได้เลือกใช้โปรแกรม Visual BASIC ซึ่งจะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านพอร์ตอนุกรมของคอมพิวเตอร์มาให้ เพื่อความสะดวกในการสื่อสารข้อมูลเราใช้ MSComm ใน Visual BASIC สำหรับการสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communication) ซึ่งเป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสรองแบบทันทีทันใด เช่น เมื่อมีตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ OnComm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที

สำหรับในโครงงานชิ้นนี้ ในส่วนของโปรแกรม Visual BASIC เราได้แสดงบน Form ดังรูปที่ 3.6 ให้เห็นว่ามีการรับสัญญาณใดๆ เข้ามาในพอร์ตอนุกรมบ้าง และข้อมูลที่เข้ามาคืออะไร หลังจากนั้นโปรแกรมก็จะจับเอาเฉพาะข้อมูลเข้ามา โดยตัดในส่วนของรหัสเริ่มต้นและรหัสปิดท้ายออกไป แล้วนำข้อมูลที่ได้เข้าไปเก็บไว้ในฐานข้อมูล ในตารางที่ชื่อ Mail_List ซึ่งจะมีฟิลด์ (Field) ของข้อมูลต่างๆ คือ ID, Sender, Subject, Email และ Message ดังแสดงในรูปที่ 3.7



รูปที่ 3.6 แสดงหน้าจอรูปแบบฟอร์ม (Form) บนโปรแกรม Visual BASIC

สำหรับในรายละเอียดของโปรแกรมในส่วนนี้ การออกแบบขั้นแรกต้องทำการกำหนดค่าต่างๆที่จำเป็นต่อการติดต่อผ่านพอร์ตอนุกรม เช่น กำหนดพอร์ตอนุกรมที่ติดต่ออยู่, ค่าอัตราบอด, พาริตี้, จำนวนบิตข้อมูล, จำนวนบิตสุดท้าย และคำสั่งในการเปิด – ปิดพอร์ตอนุกรม หลังจากนั้นก็ทำการส่งสัญญาณข้อมูลไปยังโมเด็มเพื่อทำการติดต่อกับโมเด็ม และเหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลง เพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใด



รูปที่ 3.7 แสดงการจัดฟิลด์ข้อมูลต่างๆ ที่อยู่ในฐานข้อมูล

3.2.2 ส่วนฐานข้อมูล

การพัฒนาในเรื่องการเข้าถึงฐานข้อมูล (Database) ได้แสดงผลอย่างชัดเจนใน Visual BASIC ซึ่งมีประสิทธิภาพในการโปรแกรมฐานข้อมูล ในส่วนของฐานข้อมูลนี้เราใช้โปรแกรมไมโครซอฟต์ แอคเซส (Microsoft Access) เป็นฐานข้อมูล ดังที่กล่าวไปแล้วว่าในฐานข้อมูลของโครงงานชิ้นนี้เราใช้ตารางในการเก็บข้อมูลของอีเมลล์

และในการติดต่อกับฐานข้อมูลโดยใช้โปรแกรม Visual BASIC โครงงานนี้ได้ใช้ ADO (ActiveX Data Object) ที่มีแนวความคิดเช่นเดียวกับ DAO (Data Access Object) ซึ่ง ADO ใช้เทคโนโลยี OLE Database (OLE DB) เพื่อเข้าถึงข้อมูลจากแหล่งข้อมูลต่างๆ รวมทั้งเท็กซ์ไฟล์ (.txt)

3.2.3 ส่วนโปรแกรมภาษา ASP

หน้าที่หลักๆ ของส่วนนี้ ก็คือ ทำการดึงข้อมูลจากฐานข้อมูลขึ้นมาเพื่อส่งต่อไปยังเมลล์เซิฟเวอร์ ส่วนการทำงานในส่วนนี้จะมีการทำงานอยู่ทั้งหมด 3 ส่วน ดังนี้

1. ส่วนการนับเวลาเพื่อดึงข้อมูลจากฐานข้อมูล

โดยที่โปรแกรมการทำงานในส่วนนี้จะถูกเก็บไว้ในไฟล์ที่ชื่อว่า 'Countdown.asp' การทำงานของโปรแกรมนี้จะทำการนับเวลาถอยหลัง ในที่นี้ตั้งไว้ที่ 3,600 วินาที หรือเท่ากับ 1 ชั่วโมง เมื่อครบทุกๆ 1 ชั่วโมงโปรแกรมนี้จะทำการเรียกไฟล์ 'Mailform.asp' ขึ้นมา และโปรแกรมนี้ก็จะเริ่มทำการนับเวลาต่อไปเรื่อยๆ ตลอด

2. ส่วนการดึงข้อมูลขึ้นมาจากฐานข้อมูล

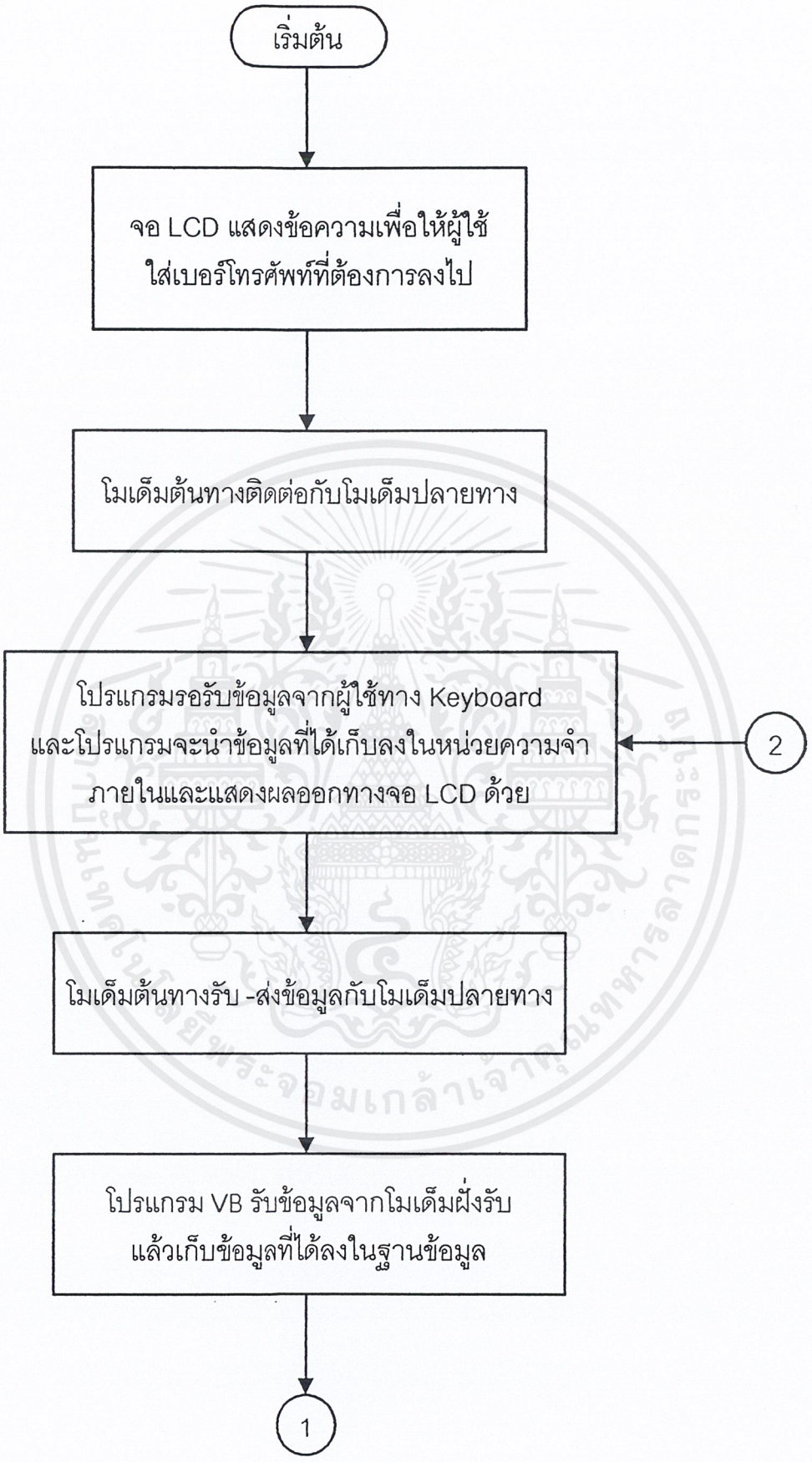
โปรแกรมการทำงานในส่วนนี้จะถูกเก็บไว้ในไฟล์ที่มีชื่อว่า ' Mailform.asp ' ซึ่งการทำงานของโปรแกรมนี้จะทำงานทุกๆ 1 ชั่วโมง โดยที่โปรแกรมนี้จะถูกเรียกว่า การนับเวลาถอยหลังของโปรแกรมในไฟล์ที่มีชื่อว่า 'Countdown.asp' เมื่อโปรแกรมนี้ถูกเรียกขึ้นมา มันจะทำการติดต่อกับฐานข้อมูลที่เก็บข้อมูลของอีเมลล์ที่ส่งมาจากเครื่องลูก โปรแกรมจะทำการตรวจสอบว่ามีข้อมูลอยู่ในฐานข้อมูลหรือไม่ ถ้ามีโปรแกรมก็จะทำการเรียกฐานข้อมูลขึ้นมาแสดงที่หน้าจอ เพื่อให้อีเมลล์ฉบับนั้นๆ ถูกส่งไปยังเมลล์เซิฟเวอร์ แต่การที่ข้อมูลจากโปรแกรมนี้จะถูกส่งไปยังเครื่องเมลล์เซิฟเวอร์ได้นั้น จำเป็นต้องมีการกดปุ่ม ' Summit ' เพื่อให้ข้อมูลนั้นถูกส่งไปยังเครื่องเมลล์เซิฟเวอร์ ดังนั้นในกระบวนการนี้ต้องมีเจ้าหน้าที่ประจำที่เครื่องเซิฟเวอร์เพื่อทำหน้าที่ส่งอีเมลล์ไปยังเครื่องเมลล์เซิฟเวอร์

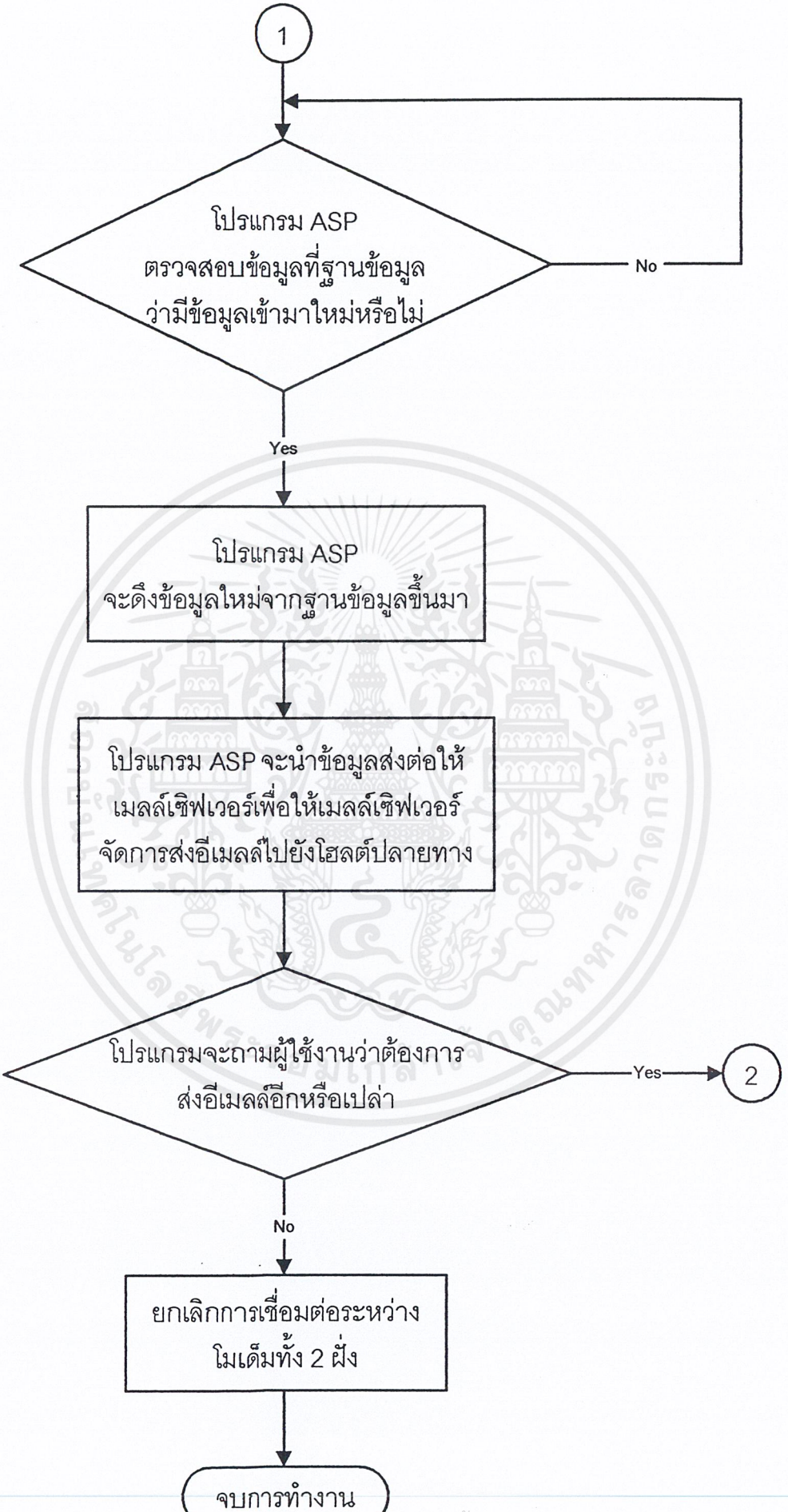
3. ส่วนส่งอีเมลล์ไปยังปลายทาง

โปรแกรมในส่วนนี้จะถูกเก็บไว้ที่เครื่องเมลล์เซิฟเวอร์ ซึ่งต้องทำการอัปโหลดขึ้นไปไว้ที่เมลล์เซิฟเวอร์เสียก่อน เมื่อโปรแกรมได้รับข้อมูลที่ส่งมาจากเครื่องเซิฟเวอร์แล้วมันจะทำการจัดเรียงข้อมูลให้อยู่ในรูปของอีเมลล์ แล้วทำการส่งไปยังปลายทางตามที่ระบุไว้ในอีเมลล์ฉบับนั้น และหลังจากที่โปรแกรมได้ทำการส่งอีเมลล์ฉบับนั้นไปแล้ว ข้อมูลของอีเมลล์ฉบับนั้นก็จะถูกลบออกจากฐานข้อมูล

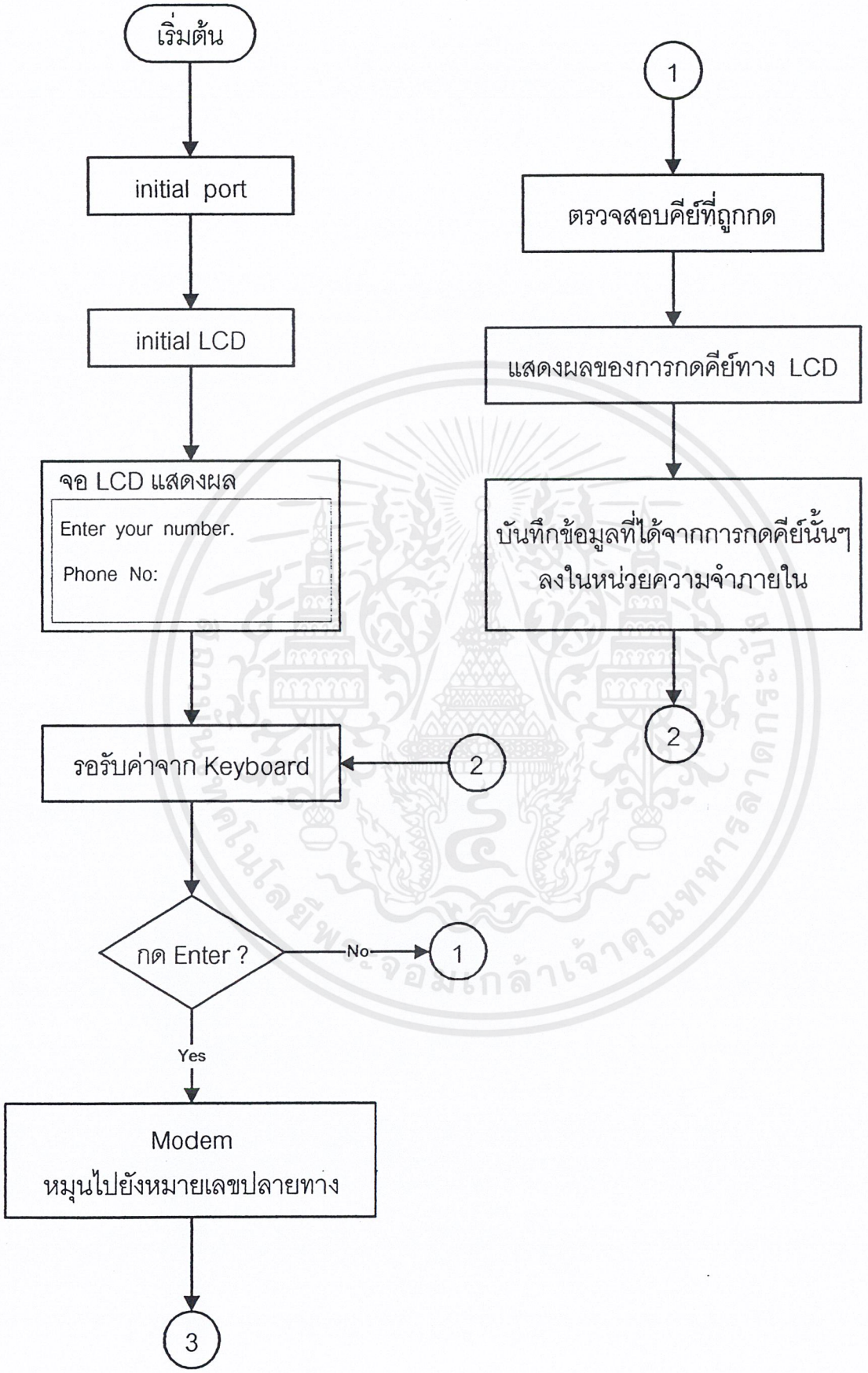
หมายเหตุ : ดังที่กล่าวไปแล้วนั้นจะเห็นได้ว่าการทำงานในส่วนนี้นั้น จำเป็นต้องใช้ฟังก์ชั่นการทำงานหลายอย่าง ซึ่งการเขียนโปรแกรมด้วยภาษา HTML นั้นไม่สามารถรองรับการทำงานได้เลย จำเป็นต้องเขียนโปรแกรมด้วยภาษา ASP แทรกลงในภาษา HTML ด้วย แต่ก็ยังไม่สามารถทำงานได้อย่างสมบูรณ์ในตัวเอง จึงได้มีการเพิ่ม Tag ของภาษา Script ลงไปอีก ซึ่งภาษา Script ที่ใช้นั้น จะมีการใช้งานทั้งภาษา VB Script และ Java Script แต่โปรแกรมก็ยังไม่สามารถทำงานได้อย่างสมบูรณ์ด้วยตัวเอง ดังจะเห็นว่าต้องมีเจ้าหน้าที่ที่คอยกดปุ่ม ' Summit ' ให้นั่นเอง

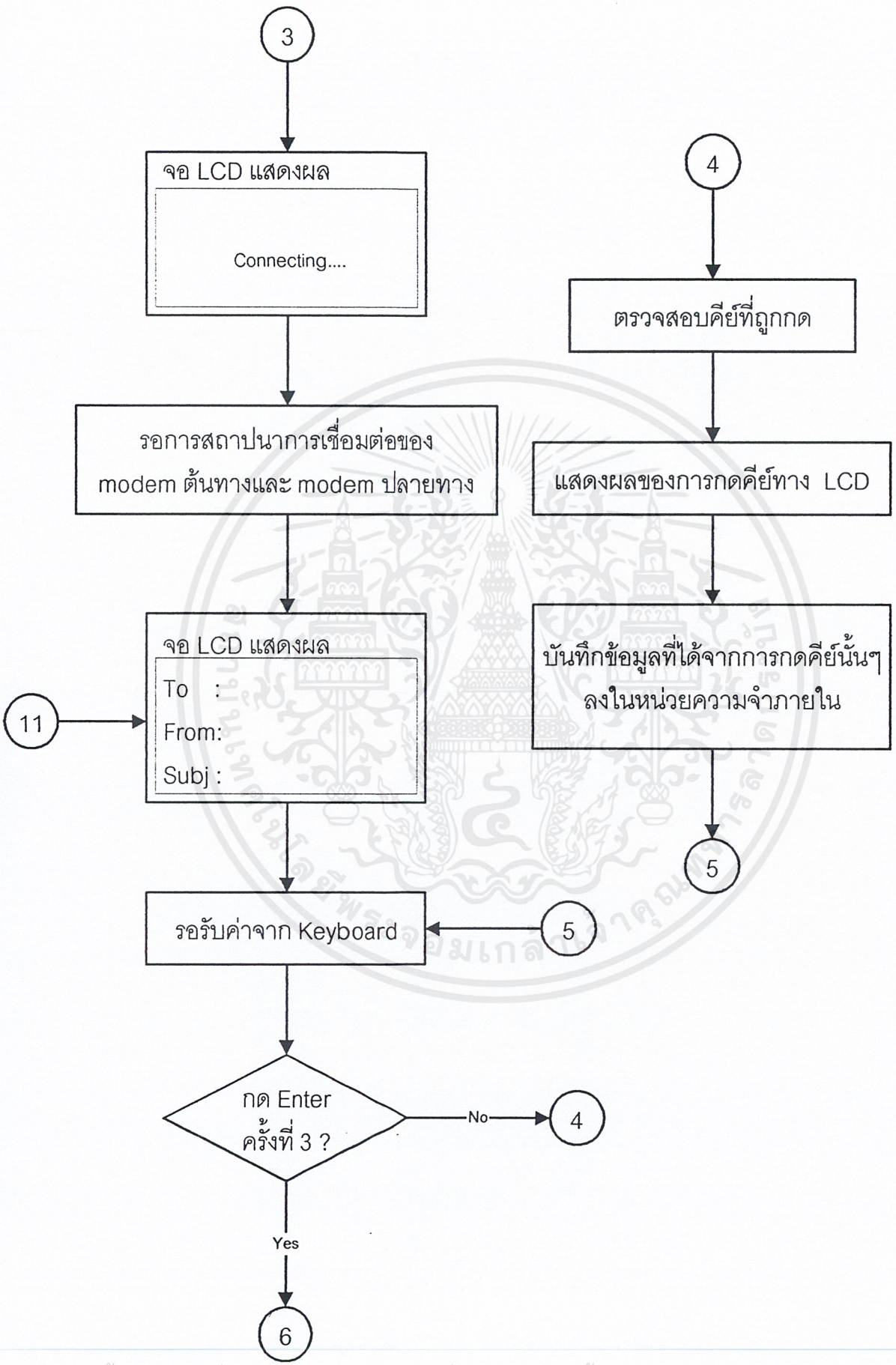
ในบทนี้ได้กล่าวถึงการออกแบบและการสร้างของส่วนต่างๆในโครงงานนี้ ไปแล้วนั้น จะขอสรุปส่วนที่เป็น Hardware โดยแสดงดังรูปในภาคผนวก ค ซึ่งจากรูปจะเป็นการแสดงถึงวงจรรวมทุกวงจรที่เครื่องลูก และส่วนที่เป็น Software และขั้นตอนการทำงานจะแสดงไว้ในรูปที่ 3.8 3.9 และ 3.10 ซึ่งทั้ง 3 รูปจะแสดงถึง Flowchart การทำงานของระบบโดยรวม การทำงานที่เครื่อง Client และการทำงานที่เครื่อง Server

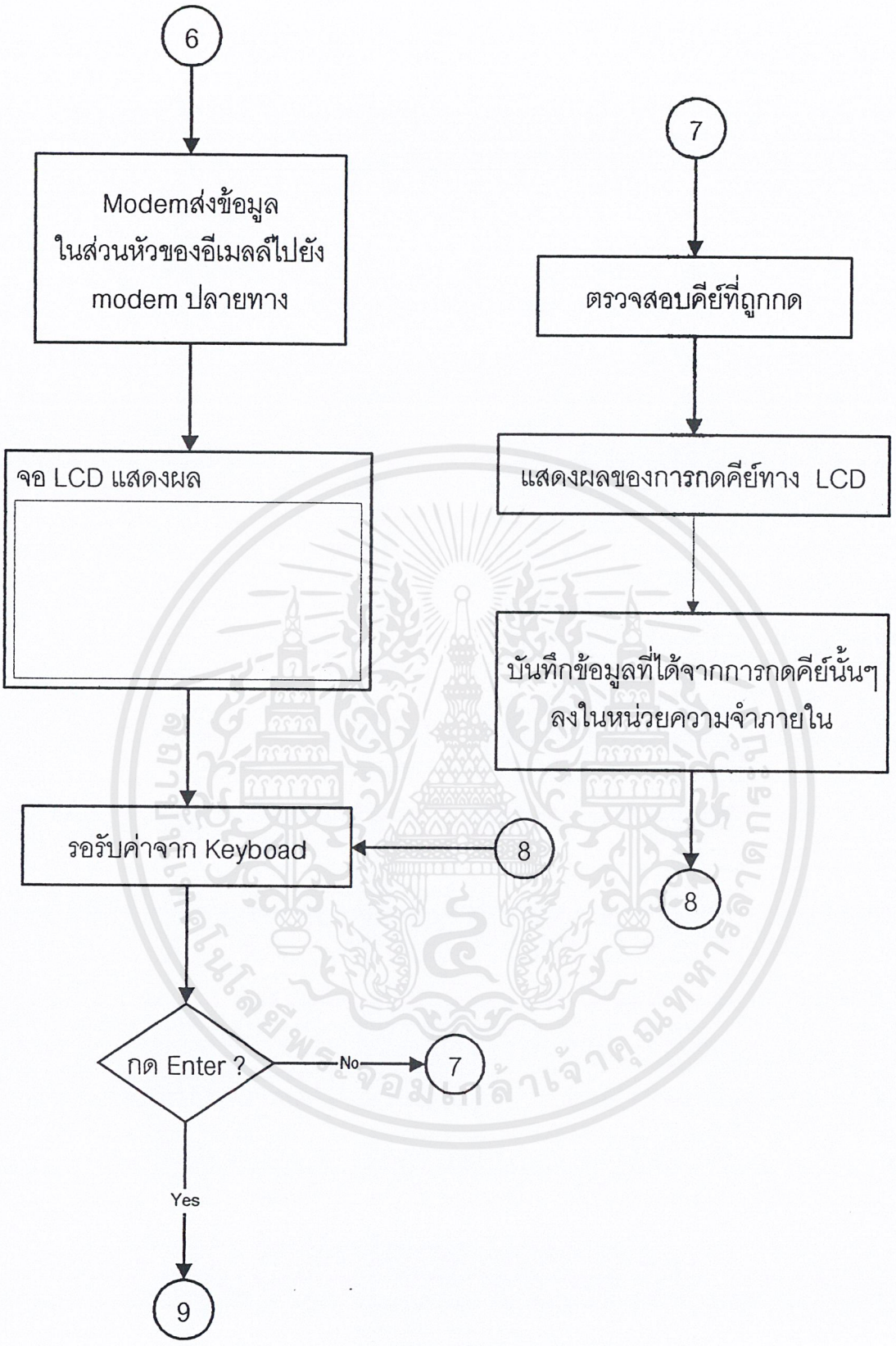


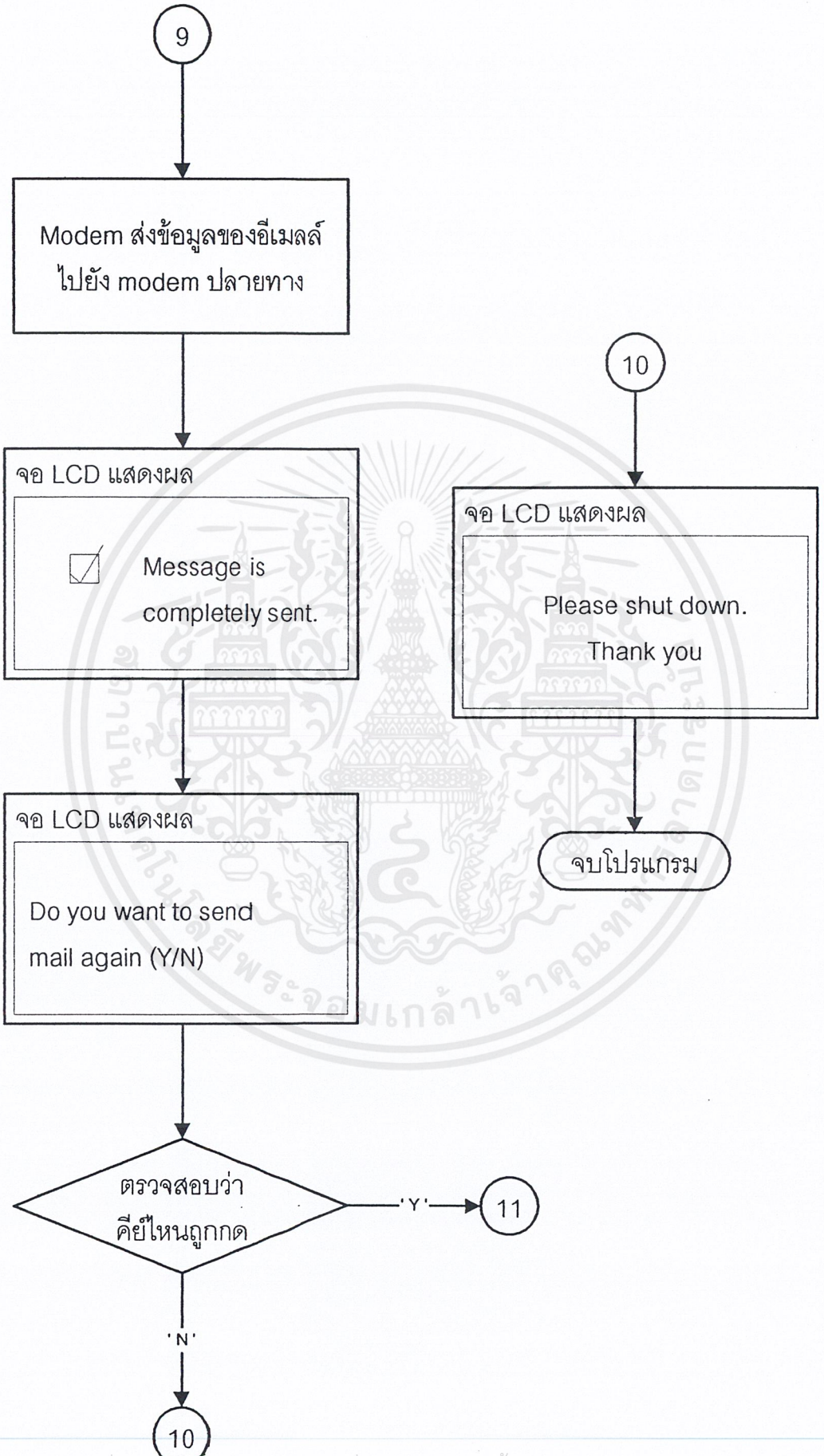


รูปที่3.8 ผังงานแสดงการทำงานโดยรวมของโครงงาน

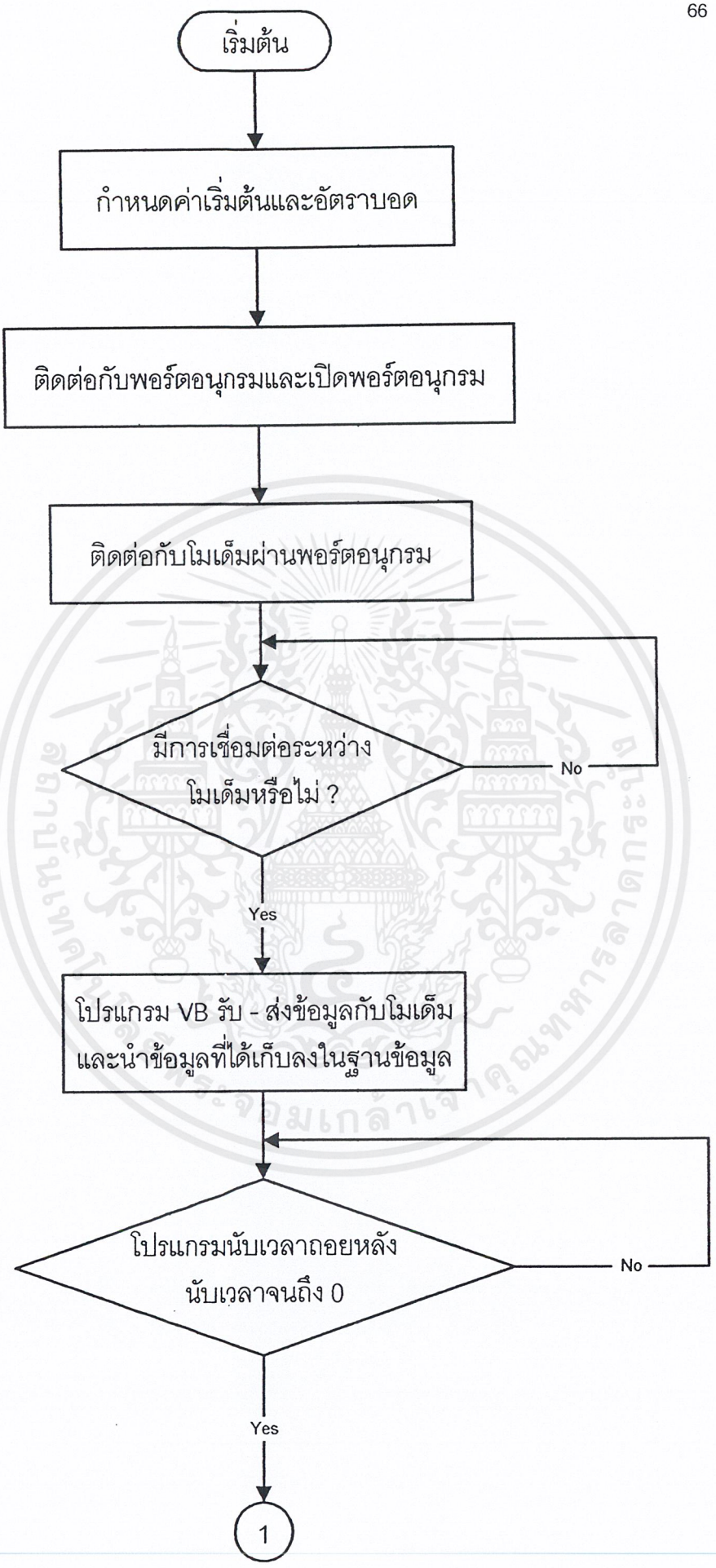


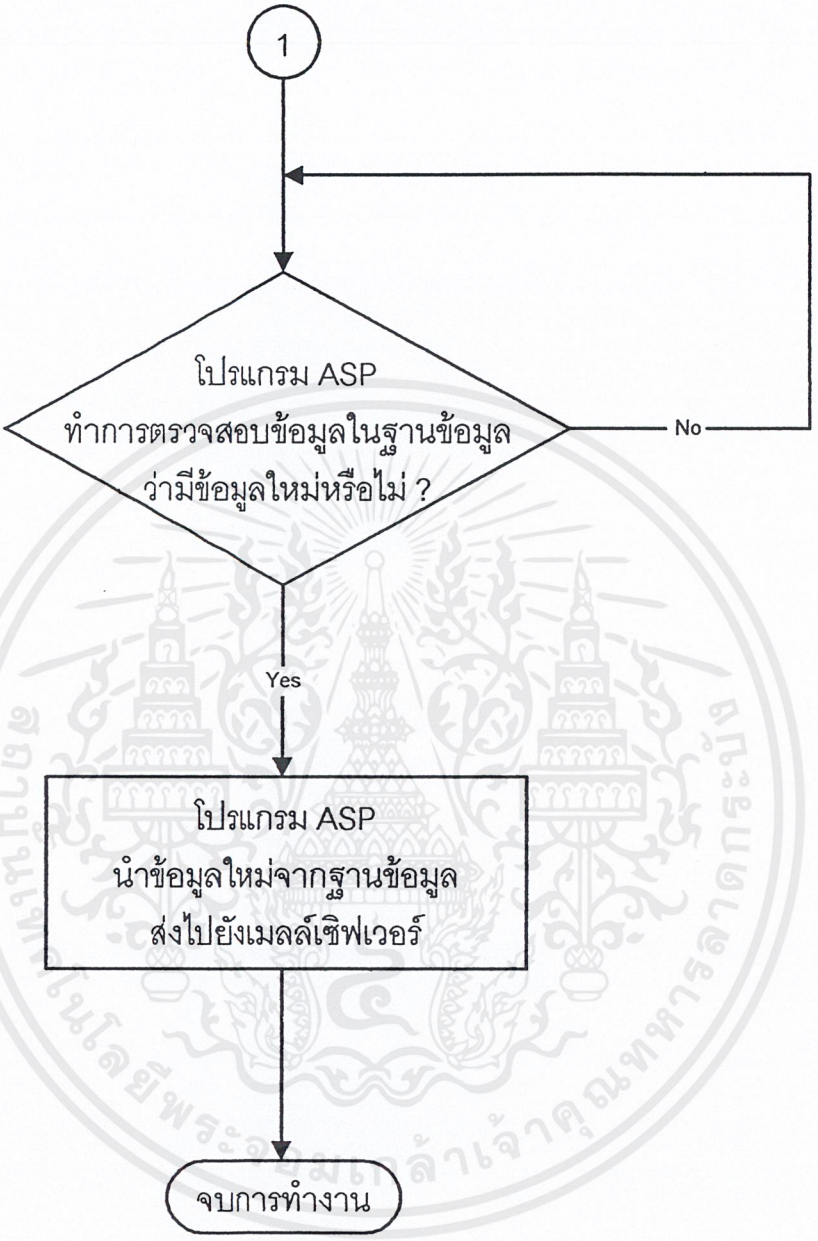






รูปที่3.9 ผังงานแสดงการทำงานของเครื่องไคลเอ็นต์





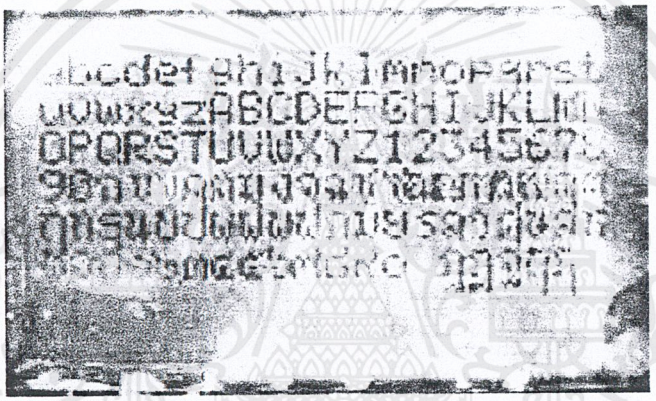
รูปที่3.10 ผังงานแสดงการทำงานของเครื่องเซิฟเวอร์

บทที่ 4
การทดลองและผลการทดลอง

4.1 ส่วนการติดต่อกับผู้ใช้งาน (Interface User)

การทดลองในส่วนนี้ จะเชื่อมต่อคีย์บอร์ด และจอแอลซีดี เพื่อให้ผู้ใช้คีย์ข้อมูลให้ปรากฏบนหน้าจอแอลซีดี โดยที่จะใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมและประมวลผล

เมื่อได้ทำการเชื่อมต่ออุปกรณ์ต่างๆ เข้าด้วยกันแล้ว แล้วทดลองคีย์ข้อมูลที่เป็นตัวอักษรจากคีย์บอร์ด ทั้งตัวอักษรภาษาอังกฤษ ตัวอักษรภาษาไทย และตัวเลขต่างๆ จะได้ผลที่หน้าจอแอลซีดี ดังในรูปที่ 4.1



รูปที่ 4.1 แสดงผลการทดลองการกดคีย์บอร์ดและแสดงผลออกทางหน้าจอแอลซีดี

ซึ่งจากรูป เราจะพบได้ว่าไมโครคอนโทรลเลอร์สามารถรับรู้ถึงการกดคีย์จากคีย์บอร์ดในแต่ละครั้งได้ แล้วก็สามารถแสดงผลการกดคีย์นั้นออกทางหน้าจอแอลซีดีได้อย่างถูกต้องอีกด้วย และโปรแกรมของไมโครคอนโทรลเลอร์สามารถขึ้นสั่งให้จอแอลซีดีแสดงการขึ้นบรรทัดใหม่เมื่อผู้ใช้พิมพ์ตัวอักษรถึงตัวสุดท้ายโดยที่ผู้ใช้ไม่ต้องกดคีย์ใดๆ เพิ่มเติมเลย

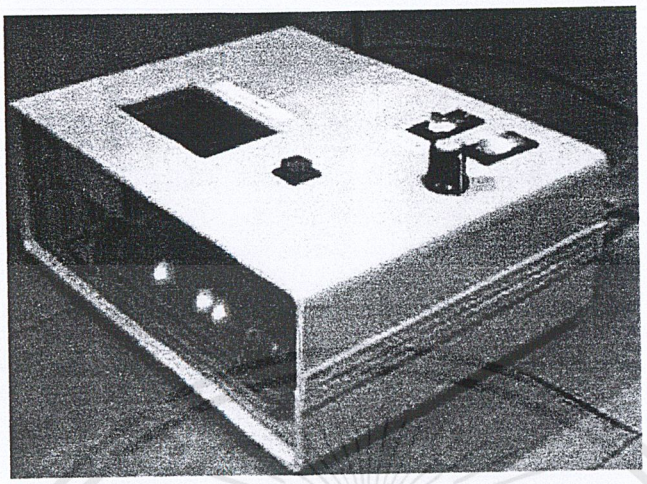
4.2 ส่วนการทำงานของเครื่องไคลเอ็นต์ (Client)

ดังที่กล่าวไปแล้วถึงการทำงานของเครื่องไคลเอ็นต์ในบทที่ 3 ดังนั้นในการทดลองนี้ เราจะลองส่งข้อมูลจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิฟเวอร์ (Server) ซึ่งเราจะต้องทำตามขั้นตอน ตามที่เครื่องได้ปรากฏข้อความขึ้นมา เพื่อให้เครื่องสามารถทำงานได้อย่างถูกต้อง โดยที่ขั้นตอนการทำงานของเครื่องลูกจะสามารถดูได้จาก flowchart การทำงานของเครื่องไคลเอ็นต์ที่แสดงไว้ในรูปที่

ขั้นตอนที่ 1

เมื่อเราทำการเปิดเครื่องโดยเปิดสวิตซ์ที่ปุ่ม Power จะสังเกตเห็นไฟที่ปุ่มสีแดงจะติดขึ้น และเราต้องกดเปิดสวิตซ์ที่ปุ่ม Modem ขึ้นด้วยเพื่อให้โมเด็มใช้งานได้ เมื่อเปิดแล้วก็จะมีไฟสีเขียว

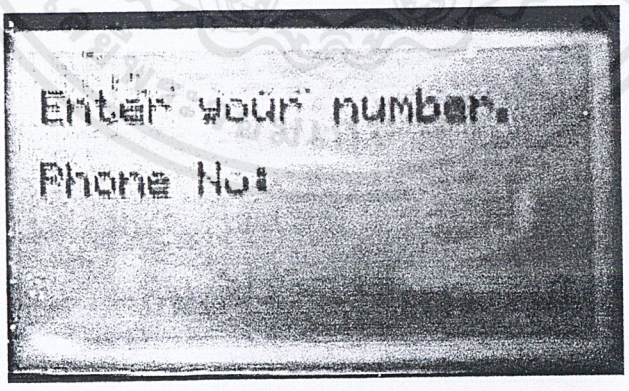
ติดขึ้นที่ปุ่ม Modem เช่นกัน ตามารูปที่ 4.2 แล้วหลังจากนั้นเราต้องกดปุ่ม Reset เพื่อให้โปรแกรมในไมโครคอนโทรลเลอร์ทำงานได้อย่างถูกต้อง



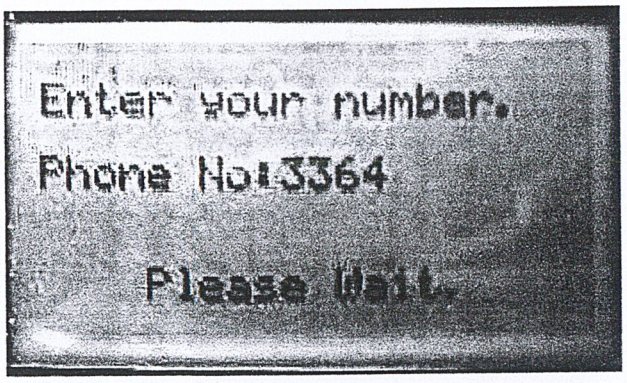
รูปที่ 4.2 แสดงสถานะเครื่องไคลเอ็นต์ที่พร้อมใช้งานแล้ว

ขั้นตอนที่ 2

หลังจากที่เราได้ทำตามขั้นตอนที่ 1 แล้วนั้น จอแอลซีดีจะปรากฏข้อความเพื่อให้ผู้ใช้ใส่เบอร์โทรศัพท์ ที่ให้โมเด็มที่เครื่องลูกติดต่อไปยังโมเด็มของเครื่องแม่ ซึ่งเบอร์โทรศัพท์นี้ทางผู้ใช้ต้องทราบเสียก่อนว่าโมเด้มของเครื่องแม่นั้นมีเบอร์โทรศัพท์เบอร์ใดต่ออยู่ การทำงานที่ขั้นตอนนี้จะแสดงในรูปที่ 4.3

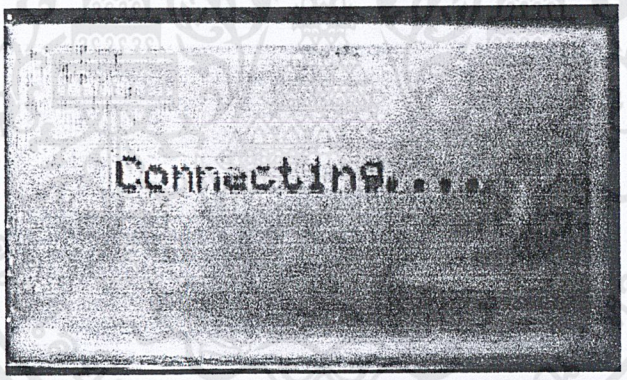


รูปที่ 4.3 จอแอลซีดีแสดงข้อความให้ผู้ใช้ใส่เบอร์โทรศัพท์ที่ต้องการติดต่อ



รูปที่ 4.4 แสดงเบอร์โทรศัพท์ที่ผู้ใช้ต้องการติดต่อ ขณะที่โมเด็มฝั่งไคลเอ็นต์กำลังติดต่อกับโมเด็มฝั่งเซิฟเวอร์

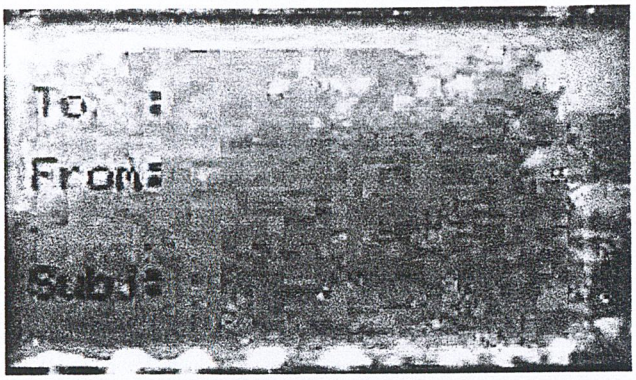
และเมื่อเราใส่เบอร์โทรศัพท์ที่ต้องการแล้วนั้น พอเรากด Enter ไมโครคอนโทรลเลอร์ก็จะสั่งให้โมเด็มของเครื่องลูกหมุนไปยังเบอร์ที่ผู้ใช้ใส่เข้ามาให้ จอแอลซีดีก็จะปรากฏข้อความตามารูปที่ 4.4 และรูปที่ 4.5 อีกทั้งเราก็จะสามารถสังเกตได้จากไฟที่ตัวโมเด็มและเสียงของโมเด็มได้อีกด้วย



รูปที่ 4.5 จอแอลซีดีแสดงข้อความ ขณะที่โมเด็มทั้ง 2 ฝั่งกำลังติดต่อกันอยู่

<u>ขั้นตอนที่ 3</u>

เมื่อโมเด็มทั้ง 2 ฝั่งสามารถติดต่อกันได้แล้ว โปรแกรมก็จะขึ้นความเพื่อให้ผู้ใช้ได้ใส่ข้อมูลที่ต้องการลงไป โดยที่ข้อมูลที่ต้องใส่ในขั้นตอนนี้ ได้แก่ 'To', 'From' และ 'Subj' ตามรูปที่ 4.6



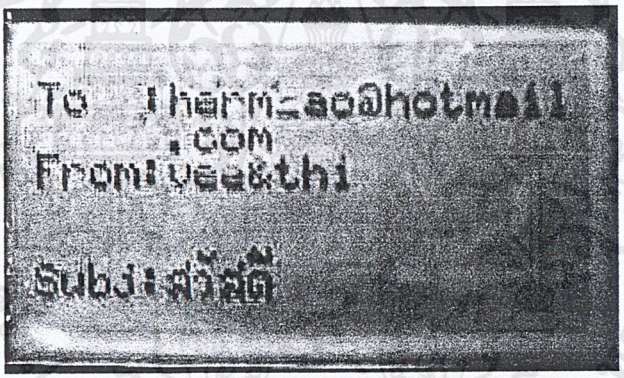
รูปที่ 4.6 แสดงข้อความเพื่อให้ผู้ใช้ใส่ข้อมูล หลังจากที่โมเด็มทั้ง 2 ฝั่งติดต่อกันได้แล้ว

โดยที่ ข้อมูล ‘ To ’ หมายถึง อีเมลล์ปลายทางของผู้รับ

ข้อมูล ‘ From ’ หมายถึง ชื่อของผู้ส่ง

ข้อมูล ‘ Subj ’ หมายถึง หัวข้อของอีเมลล์ที่ต้องการส่ง

ซึ่งในการทดลองในขั้นตอนนี้จะแสดงในรูปที่ 4.7



รูปที่ 4.7 แสดงตัวอย่างการใส่ข้อมูลลงในฟิลด์ต่างๆ

<u>ขั้นตอนที่ 4</u>

หลังจากที่เราได้ใส่ข้อมูลต่างๆ ในขั้นตอนที่ 3 แล้วนั้น ในขั้นตอนนี้ทางโปรแกรมจะให้เราใส่ข้อความที่เราต้องการส่งไปกับอีเมลล์เข้าไปในขั้นตอนนี้ ตามรูปที่ 4.8

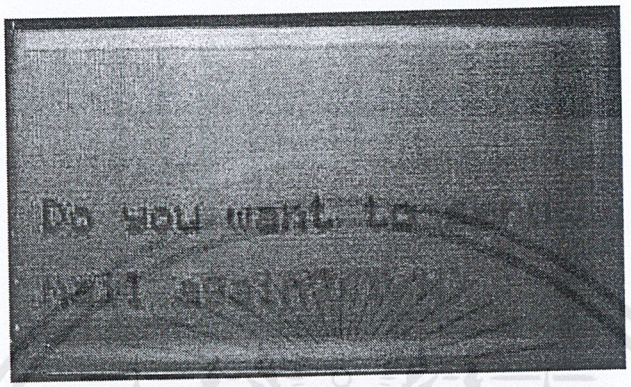
รูปที่ 4.8 แสดงตัวอย่างการใส่ข้อความที่ต้องการส่งไปยังอีเมลล์ปลายทาง

เมื่อเราใส่ข้อมูลเสร็จแล้ว เราต้องกดปุ่ม Enter เพื่อให้ข้อมูลที่ได้ใส่มาทั้งหมดส่งไปยังเครื่อง Server หน้าจอแอลซีดีก็จะแสดงข้อความตามรูปที่ 4.9 เพื่อแจ้งให้ผู้ใช้ทราบว่าข้อมูลได้ถูกจัดส่งไปเรียบร้อยแล้ว ตามรูปที่ 4.10

รูปที่ 4.9 จอแอลซีดีแสดงข้อความเพื่อให้ผู้ใช้รับทราบว่าขณะนี้กำลังส่งข้อมูลไปให้เครื่องเซิฟเวอร์

รูปที่ 4.10 แสดงการทำงานเมื่อเครื่องไคลเอ็นต์ส่งข้อมูลไปให้เครื่องเซิฟเวอร์ได้แล้ว

หลังจากนั้นโปรแกรมก็จะถามผู้ใช้งานว่าต้องการส่งอีเมลล์ฉบับใหม่อีกหรือไม่ ตามรูปที่ 4.11 ถ้าผู้ใช้ต้องส่งอีก ก็จะกดปุ่ม 'Y' โปรแกรมก็จะกลับไปทำตามขั้นตอนที่ 3 อีกครั้ง แต่ถ้าผู้ใช้กดปุ่ม 'N' แสดงว่าผู้ใช้ไม่ต้องการส่งอีเมลล์แล้ว จอแอลซีดีก็จะแสดงข้อความตามรูปที่ 4.12 เพื่อให้ผู้ใช้ยกเลิกการเชื่อมต่อระหว่างเครื่องแม่กับเครื่องลูก



รูปที่ 4.11 แสดงข้อความเพื่อให้ผู้ใช้เลือกว่าต้องการส่งอีเมลล์ฉบับใหม่อีกหรือไม่

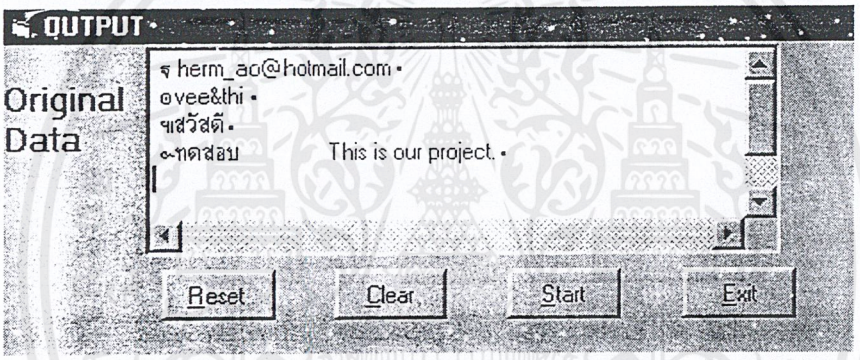
รูปที่ 4.12 แสดงข้อความเพื่อให้ผู้ใช้ยกเลิกการเชื่อมต่อ เมื่อผู้ใช้ไม่ต้องการส่งอีเมลล์ฉบับใหม่แล้ว

4.3 ส่วนการเชื่อมต่อกับโมเด็มของเครื่องลูกที่เครื่อง Server

โปรแกรมในส่วนนี้จะทำการติดต่อกับโมเด็มของเครื่องเซิฟเวอร์และรับข้อมูลที่ส่งมาจากโมเด็มฝั่งเครื่องไคล์เอ็นต์ หลังจากนั้นโปรแกรมก็จะทำการถอดเอารหัสเริ่มต้นและรหัสปิดท้ายออก เพื่อนำข้อมูลไปเก็บไว้ที่ฐานข้อมูล ซึ่งการทำงานในส่วนนี้จะแสดงได้ตามขั้นตอน ดังนี้

ขั้นตอนที่ 1

เมื่อมีการส่งข้อมูลมาจากโมเด็มฝั่งไคลเอ็นต์แล้ว โปรแกรมที่เครื่องเซิฟเวอร์ก็จะนำข้อมูลที่ได้มาแสดงที่หน้าจอส่วนแสดงผลของการส่งข้อมูล ดังในรูปที่ 4.13



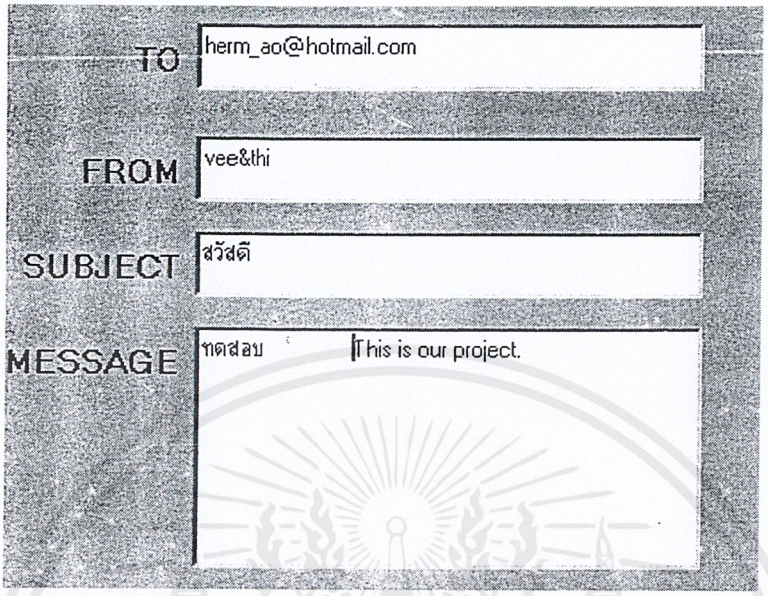
รูปที่ 4.13 แสดงหน้าจอรูปแบบข้อมูลที่ถูกส่งมาจากเครื่องไคลเอ็นต์

ขั้นตอนที่ 2

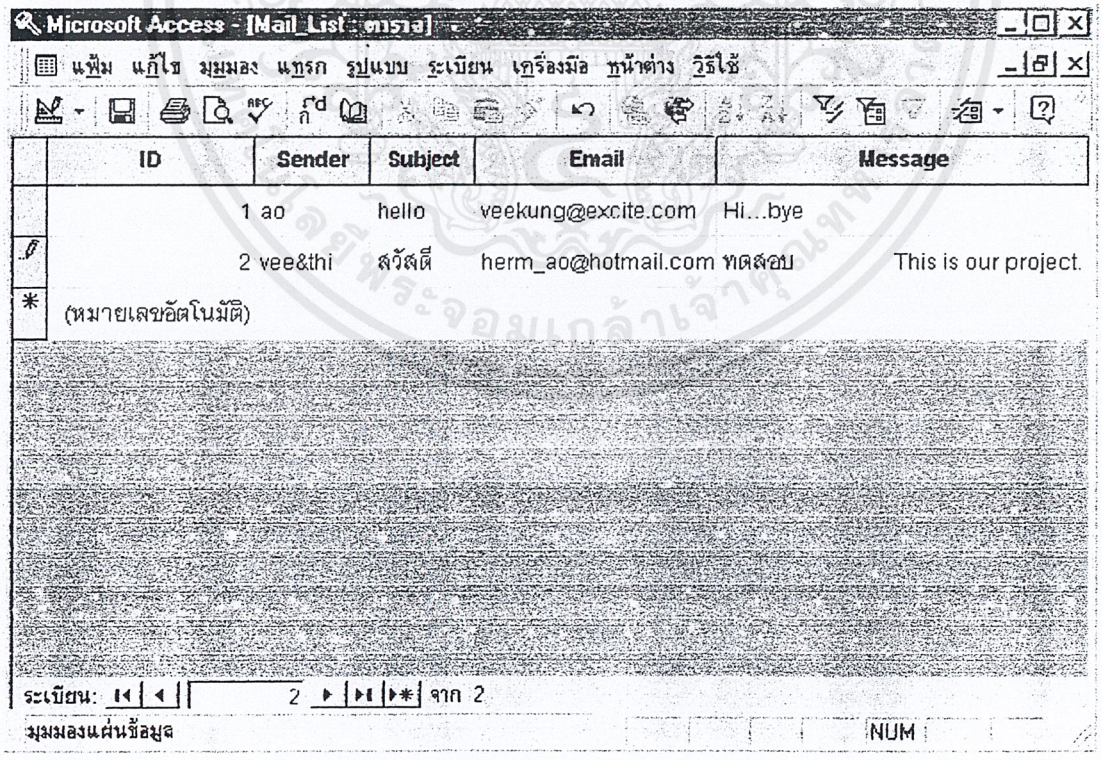
ข้อมูลที่รับได้จากเครื่องไคลเอ็นต์จะถูกถอดเอารหัสเริ่มต้น และรหัสปิดท้ายออก แล้วข้อมูลในแต่ละส่วนก็จะถูกจัดไว้ในฟิลด์ข้อมูลประเภทต่างๆ ดังรูปที่ 4.14

ขั้นตอนที่ 3

หลังจากข้อมูลถูกแยกไปตามฟิลด์ต่างๆ แล้ว ข้อมูลเหล่านั้นก็จะถูกนำไปเก็บไว้ที่ฐานข้อมูลเองโดยอัตโนมัติ ดังแสดงในรูปที่ 4.15



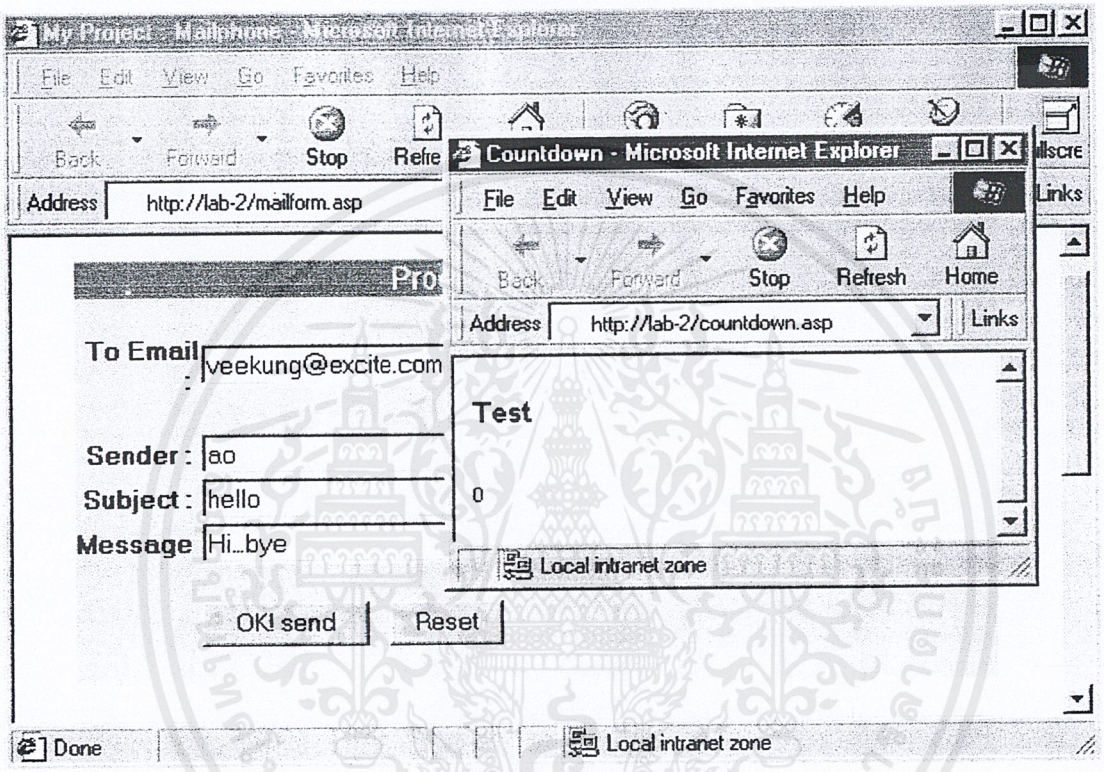
รูปที่ 4.14 แสดงหน้าจอที่ข้อมูลถูกถอดรหัสออกและแยกไปตามฟิลด์ข้อมูลประเภทนั้นๆ



รูปที่ 4.15 แสดงข้อมูลที่ถูกนำมาเก็บไว้ในฐานข้อมูล

4.4 ส่วนการเชื่อมต่อระหว่างฐานข้อมูลกับเมลล์เซิฟเวอร์

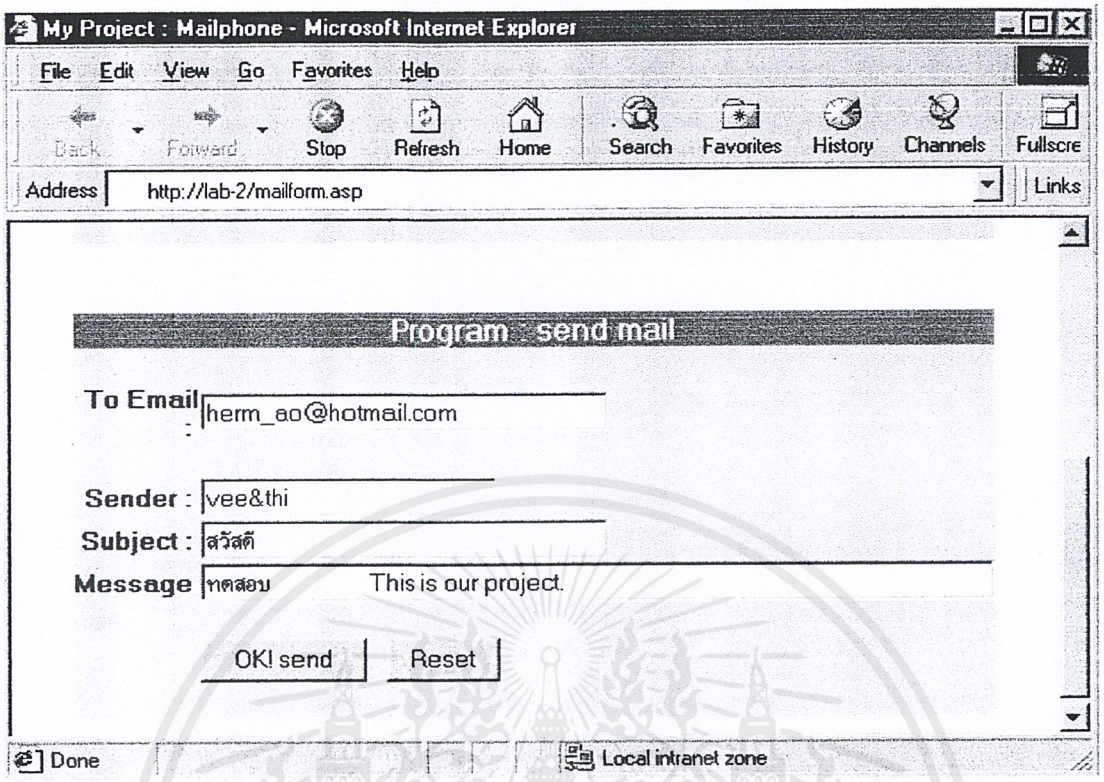
การทดลองในส่วนนี้ จะแสดงถึงการทำงานของโปรแกรมภาษา ASP โดยที่โปรแกรมนับเวลาถอยหลังเริ่มทำงาน โปรแกรมนี้ก็จะแสดงเวลานับถอยหลังเป็นวินาที ในที่นี้ตั้งไว้ที่ 3,600 วินาที และเมื่อโปรแกรมนับเวลาถึง 0 โปรแกรมนี้ก็จะเรียกโปรแกรมที่ติดต่อกับฐานข้อมูลขึ้นมา และโปรแกรมนับเวลาถอยหลังก็จะทำการนับเวลาถอยหลังอีกครั้งที่ 3,600วินาทีเช่นเดิม ดังแสดงในรูปที่ 4.16



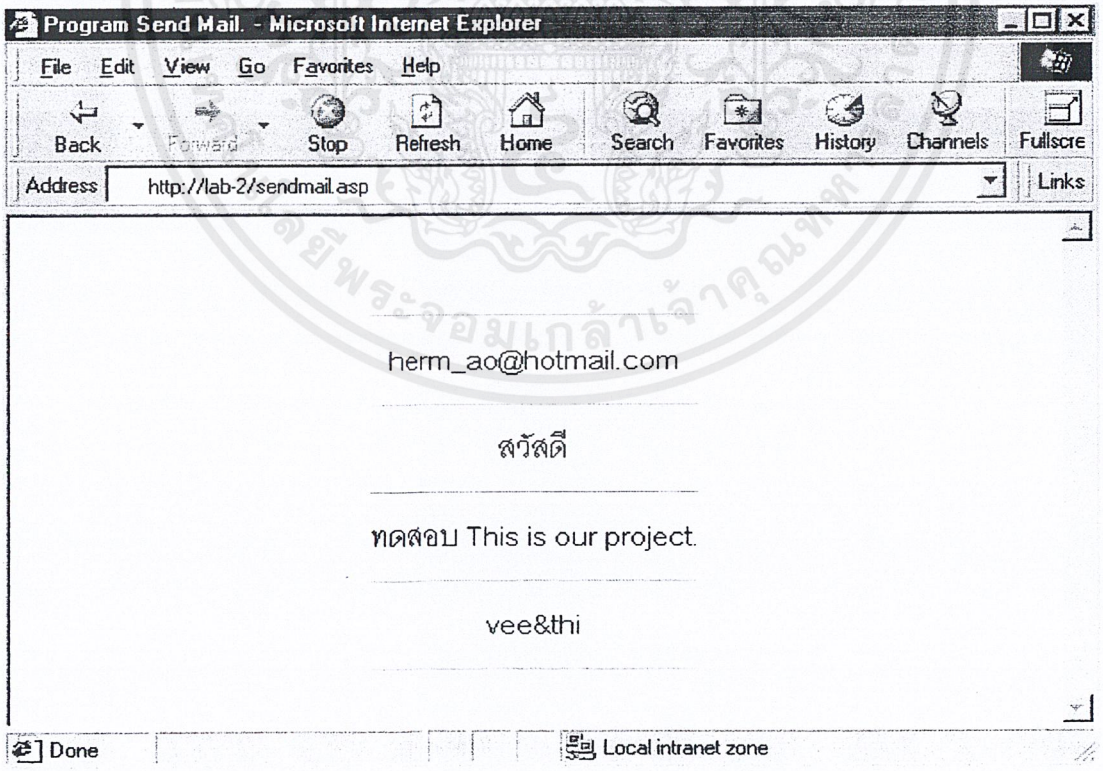
รูปที่ 4.16 แสดงการทำงานของโปรแกรมนับเวลาถอยหลังขณะที่เวลาถูกนับจนถึง 0

เมื่อโปรแกรมที่ใช้ดึงข้อมูลจากฐานข้อมูลขึ้นมา โปรแกรมนี้ก็จะแสดงข้อมูลที่ถูกเก็บไว้ในฐานข้อมูลขึ้นมาที่หน้าจอคอมพิวเตอร์ของเครื่อง Server ดังแสดงในรูปที่ 4.17

ซึ่งจากรูปเราจะเห็นว่ามีอีเมลล์ที่ต้องการส่งอยู่ 2 ฉบับ หลังจากนั้นเพื่อให้ข้อมูลถูกส่งไปยังเมลล์เซิฟเวอร์ ต้องมีการกดปุ่ม ' OK! Send ' เพื่อให้อีเมลล์ฉบับนั้นถูกส่งไปยังเมลล์เซิฟเวอร์ เพื่อให้เมลล์เซิฟเวอร์จัดการส่งอีเมลล์ไปยังปลายทางต่อไป ซึ่งในส่วนนี้เครื่องเมลล์เซิฟเวอร์จะถูกจำลองการใช้งานบนเครื่องเซิฟเวอร์ที่ลงโปรแกรม PWS ดังแสดงในรูป 4.18

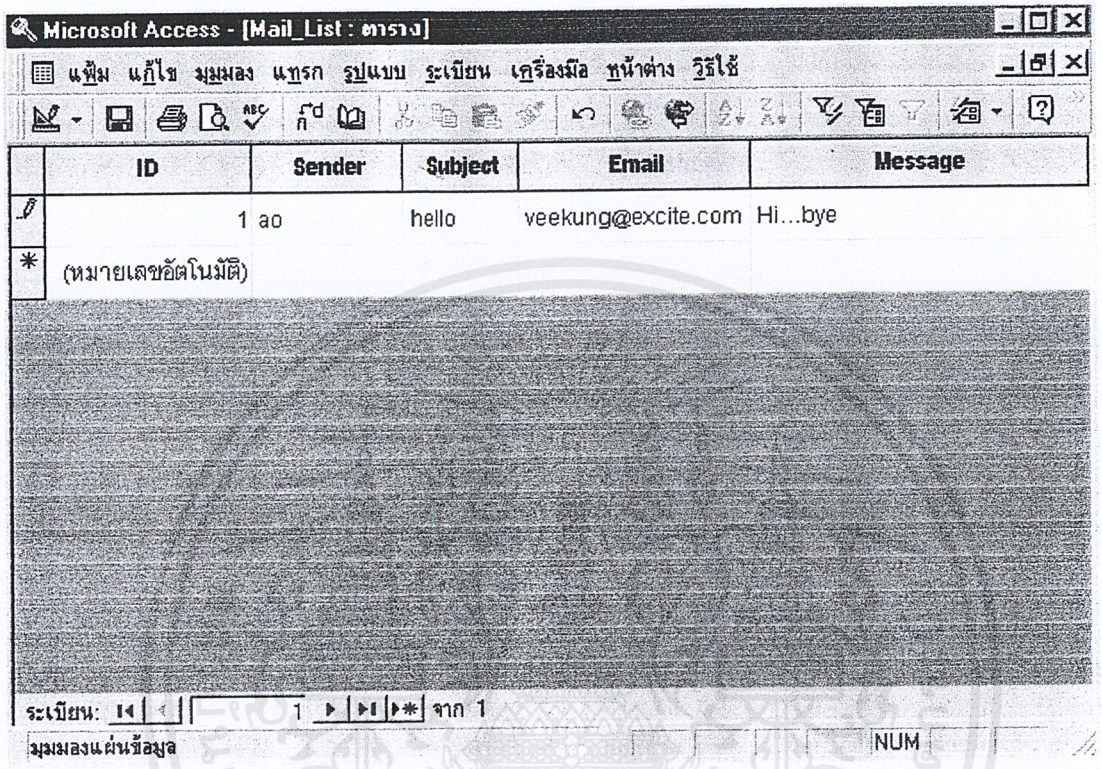


รูปที่ 4.17 แสดงหน้าจอของโปรแกรม ASP ขณะที่โปรแกรมทำการเรียกข้อมูลจากฐานข้อมูลขึ้นมา



รูปที่ 4.18 แสดงหน้าจอของโปรแกรมส่งเมลล์ เมื่อกดปุ่ม ' OK! Send ' ที่หน้าจอโปรแกรม mailform.asp

ซึ่งหลังจากที่เราได้ส่งอีเมลล์ไปแล้ว ข้อมูลของอีเมลล์ฉบับนั้นๆ ที่อยู่ในฐานข้อมูลจะถูกลบออกโดยอัตโนมัติ เพื่อที่โปรแกรมเรียกข้อมูลจากฐานข้อมูลจะได้ไม่เรียกข้อมูลที่ส่งไปแล้วขึ้นมาใหม่ อีเมลล์ที่ถูกส่งไปจะได้ไม่มีการส่งซ้ำกันไปนั่นเอง



รูปที่ 4.19 แสดงข้อมูลในฐานข้อมูลเมื่อได้ส่งอีเมลล์ที่ 2 ไปแล้ว

บทที่5
สรุปและวิจารณ์

5.1 สรุปผลการดำเนินงาน

ในการดำเนินงานที่ผ่านมา สามารถสรุปผลการทำงานได้ดังนี้

- ศึกษาการทำงานของจอแอลซีดีแบบกราฟฟิก
- ศึกษาการทำงานของคีย์บอร์ด และรหัสของคีย์บอร์ดที่ถูกส่งออกมาจากคีย์บอร์ด
- ศึกษาและออกแบบวงจรอินเตอร์รัปท์ เพื่อเชื่อมต่อกับไมโครคอนโทรลเลอร์
- ศึกษาการทำงานของไมโครลคอนโทรลเลอร์เบอร์ 89C52 ในด้านการควบคุมและรับ-ส่งข้อมูลผ่านพอร์ทอินพุท พอร์ทเอาท์พุท การอินเตอร์รัปท์ การรับ-ส่งข้อมูลแบบอนุกรม และการจัดวางตำแหน่งหน่วยความจำภายในของไมโครคอนโทรลเลอร์
- ศึกษาและออกแบบการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับจอแอลซีดีกราฟฟิก
- ศึกษาและออกแบบการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคีย์บอร์ด
- เขียนตารางการแปลงข้อมูลของรหัสคีย์บอร์ด รหัสแอสกี และข้อมูลที่ต้องส่งให้กับจอแอลซีดีแบบกราฟฟิก
- เขียนโปรแกรมในส่วนของการติดต่อกับผู้ใช้งาน โดยผ่านอุปกรณ์รับค่าอินพุท คือ คีย์บอร์ด และอุปกรณ์แสดงผลเอาทพุท คือ จอแอลซีดีแบบกรฟฟิก
- ศึกษาการทำงานของโมเด็ม และคำสั่งที่ควบคุมการทำงานของโมเด็ม
- ศึกษาการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับโมเด็ม โดยผ่านวงจรแปลงระดับสัญญาณ ระหว่าง TTL และ RS-232
- ศึกษาการเชื่อมต่อระหว่างโมเด็มและควบคุมการไหลของข้อมูลระหว่างโมเด็ม
- ศึกษาการใช้งานคอนโทรลเลอร์ MSComm ในโปรแกรม Visual BASIC
- ศึกษาการใช้โปรแกรม Visual BASIC เพื่อติดต่อกับฐานข้อมูล
- เรียนรู้การใช้งาน ADO (ActiveX Data Objects)
- ศึกษาการใช้งานฐานข้อมูล Microsoft Access
- ศึกษาและเขียนโปรแกรม ASP , VBScript และ JavaScript
- ศึกษาการเชื่อมโยงข้อมูลระหว่าง ASP และ ฐานข้อมูล
- ศึกษาการรับ – ส่งข้อมูลของเครื่องเซิฟเวอร์กับเมลล์เซิฟเวอร์
- ออกแบบและเขียนลายวงจรพิมพ์ของเครื่องไคลเอ็นต์

5.2 ปัญหาที่เกิดขึ้นและแนวทางแก้ไข

1.เนื่องจากในโครงงานนี้มีความจำเป็นที่ต้องใช้อุปกรณ์และโปรแกรมต่างๆมากมาย ทั้งในฝั่งของไคลเอ็นต์และฝั่งของเซิฟเวอร์ ซึ่งในส่วนของไคลเอ็นต์นั้นอุปกรณ์แต่ละตัวมีรูปแบบของข้อมูลที่แตกต่างกันทั้งหมด กล่าวคือ ข้อมูลจากคีย์บอร์ด ข้อมูลในไมโครคอนโทรลเลอร์ และข้อมูลที่ส่งไปยังจอแอลซีดีแบบกราฟฟิก ถึงแม้ข้อมูลจะเป็นข้อมูลเดียวกันแต่ข้อมูลที่อุปกรณ์ต่างๆ เหล่านี้สามารถนำไปใช้งานได้แตกต่างโดยสิ้นเชิง ทำให้ต้องมีการแปลงข้อมูลอยู่ตลอดเวลา

2. การเชื่อมต่อข้อมูลในโครงงานนี้ มีการเชื่อมโยงของข้อมูลอยู่หลายช่วงมาก ทำให้เกิดปัญหาระหว่างการเชื่อมโยงข้อมูลเป็นอย่างมาก ซึ่งจะเห็นได้ว่า ข้อมูลจากผู้ใช้งานจะถูกส่งออกมาจากคีย์บอร์ด แล้วไมโครคอนโทรลเลอร์ต้องแปลงข้อมูลแล้วส่งให้จอแอลซีดี และนำข้อมูลนั้นส่งผ่านออกไปยังโมเด็มฝั่งส่ง ที่เครื่องเซิฟเวอร์โปรแกรม Visual BASIC จะรับข้อมูลจากโมเด็มฝั่งส่งโดยใช้โมเด็มฝั่งรับ แล้วทำการถอดรหัสเริ่มต้นและรหัสปิดท้ายออก แล้วนำข้อมูลเหล่านั้นไปเก็บไว้ที่ฐานข้อมูล รอให้โปรแกรม ASP ดึงข้อมูลจากฐานข้อมูลส่งไปยังเมลล์เซิฟเวอร์ เพื่อให้เมลล์เซิฟเวอร์ส่งข้อมูลนั้นไปที่ปลายทางอีกที ซึ่งจะเห็นได้ว่า กระบวนการส่งข้อมูลหลายขั้นตอนมาก

3. ในส่วนของโปรแกรมการเชื่อมต่อกับเครือข่ายอินเตอร์เน็ตนั้น ต้องมีความรู้ในเรื่องการเขียนโปรแกรมเพื่อส่งอีเมลล์ โดยที่ต้องใช้ความรู้ในการเขียนโปรแกรมหลายภาษา เช่น ASP , VBScript และ JAVAScript รวมทั้งภาษา HTML ทำให้การเขียนโปรแกรมในส่วนนี้ไม่สมบูรณ์เท่าใดนัก

4. การใช้งานเมลล์เซิฟเวอร์นั้น เป็นให้เครื่องเซิฟเวอร์เชื่อมต่อกับเมลล์เซิฟเวอร์ได้นั้น จำเป็นต้องทำการ upload ไฟล์บางส่วนขึ้นไปอยู่ที่เครื่องเมลล์เซิฟเวอร์ และการส่งข้อมูลไปยังเมลล์เซิฟเวอร์แล้วสั่งให้เมลล์เซิฟเวอร์ส่งอีเมลล์ไปยังปลายทางได้นั้นทำได้ยากในทางปฏิบัติ ถึงแม้ในทางทฤษฎีนั้นกระบวนการนี้ยอมเป็นไปได้ แต่เมื่อเกิดข้อผิดพลาดขึ้น ก็ยากที่จะหาสาเหตุได้เจอ

5. การเขียนตัวอักษรภาษาไทยลงในจอแอลซีดีนั่นเป็นไปได้อย่างยากยิ่ง เนื่องจากตัวอักษรภาษาไทยนั้นมีระดับการจัดวางอยู่ถึง 4 ระดับ ทำให้เกิดปัญหาการแสดงผลที่จอแอลซีดีและทำให้เกิดข้อผิดพลาดมากในส่วนนี้

6. การใช้งานโมเด็มแต่ละรุ่น แต่ละยี่ห้อ และแต่ละครั้งในการผลิตนั้น คำสั่งและการใช้งานไม่เหมือนกัน ถึงแม้ถูกคำสั่งจะใช้ AT Command เหมือนกัน แต่ในรายละเอียดปลีกย่อยจะไม่เหมือนกัน โดยเฉพาะอย่างยิ่งในฝั่งไคลเอ็นต์ เมื่อนำไมโครคอนโทรลเลอร์ควบคุมการทำงานของโมเด็มด้วยแล้ว ถ้าเปลี่ยนชนิดของโมเด็มแล้ว ต้องเขียนโปรแกรมควบคุมโมเด็มใหม่ทันที

7. ในโปรแกรม Visual BASIC การรับ – ส่งข้อมูลกับไมโครคอนโทรลเลอร์ยังมีข้อผิดพลาดบ้างบางส่วน เนื่องจากในโปรแกรมมีการใช้รูปแบบข้อมูล เช่น บิตเริ่มต้น (Start Bit), บิตปิดท้าย (Stop Bit) ทำให้โปรแกรมในส่วนนี้ยังไม่มีประสิทธิภาพอย่างเพียงพอ

5.3 แนวทางการพัฒนา

เนื่องจากการทำงานของโครงงานนี้ยังไม่สมบูรณ์มากนัก ดังนั้นจึงสามารถพัฒนาโครงงานชิ้นนี้ได้อีกอย่างมาก ซึ่งอาจพัฒนาในส่วนต่างๆ ได้ ดังนี้

- พัฒนาการใช้งานภาษาไทยให้สามารถแสดงผลได้อย่างสมบูรณ์มากยิ่งขึ้น
- พัฒนาโปรแกรมเพื่อควบคุมการทำงานของโมเด็มเพื่อให้รองรับโมเด็มในแต่ละรุ่นได้
- พัฒนาในส่วนการตอบรับของข้อมูลเมื่อไปถึงปลายทางได้ดีกว่านี้
- เขียนโปรแกรมในส่วนของการติดต่อกับโมเด็ม โดยให้ไมโครคอนโทรลเลอร์และเครื่องคอมพิวเตอร์สามารถตรวจสอบการเปลี่ยนแปลงของขาสัญญาณต่างๆ ได้อย่างสมบูรณ์มากยิ่งขึ้น
- ออกแบบและสร้างฟังก์ชั่นเพิ่มเติมในส่วนของการติดต่อกับผู้ใช้งานให้มากขึ้นกว่าเดิม เนื่องจากในโครงงานนี้บางส่วนของคีย์บอร์ดยังไม่สามารถใช้งานได้ครบทุกส่วน
- ในส่วนของการส่งข้อมูลไปยังเมลล์เซิฟเวอร์นั้นในโครงงานนี้ ยังไม่สามารถทำงานได้อย่างเป็นอัตโนมัติ ทำให้การพัฒนาในส่วนนี้สามารถพัฒนาการทำงานให้เป็นไปอย่างอัตโนมัติได้
- สามารถออกแบบเครื่องโทรศัพท์ส่งอีเมลล์ให้มีขนาดเล็กและมีน้ำหนักเบามากยิ่งขึ้น

ภาคผนวก ก

-โปรแกรมภาษาแอสเซมบลี (Assembly)

-โปรแกรมวิชวลเบสิก (Visual Basic)

-โปรแกรมเอเอสพี (ASP)

```
; โปรแกรม Assembly
; โปรแกรมควบคุมการทำงานที่เครื่องไคลเอ็นต์
LCD_DI                  BIT   P2.3
LCD_RW          BIT   P2.4
LCD_EN          BIT   P2.5
LCD_CS2         BIT   P2.6
LCD_CS1         BIT   P2.7

REG5            EQU   02FH
REG6            EQU   02EH
REG7            EQU   02DH
DATA            EQU   02CH

                ORG   0000H
                AJMP  INITIAL
                ORG   0013H
                AJMP  INTERRUPT

INITIAL:        MOV   P0,#00000000B
                MOV   P1,#11111111B
                MOV   P2,#11111111B
                MOV   P3,#11111111B
                MOV   TMOD,#21H
                MOV   TH1,#0FDH
                MOV   TL1,#0FDH
                MOV   IE,#10000100B
                SETB  IT1
                MOV   SCON,#11000000B
                SETB  TR1

                CLR   LCD_RW
                CLR   LCD_DI
                ACALL INIT_LCD
                AJMP  MAIN

INIT_LCD:       ACALL DELAY_100ms
                MOV   P0,#00111111B
                ACALL LCD_CLK
                ACALL DELAY_10ms
                MOV   P0,#11000000B
                ACALL LCD_CLK
                RET
;-----------------------------------------------------------------
INTERRUPT:      MOV   C,P2.2
                RRC   A
                INC   R2
                RETI
;-----------------------------------------------------------------
GET_DATA:       MOV   A,#00H
                MOV   R2,#00H
WAIT_DATA:      CJNE  R2,#09H,$
                MOV   DATA,A
WAIT_LAST:      CJNE  R2,#11,$
                CJNE  A,#0F0H,NORMAL
                CJNE  A,#0E0H,NORMAL
                AJMP  GET_DATA
NORMAL:         RET

;-----------------------------------------------------------------
PAGE_L:         CLR   LCD_CS2
                SETB  LCD_CS1
```

```
                    CLR   00H
                    RET

PAGE_R:             SETB  LCD_CS2
                    CLR   LCD_CS1
                    SETB  00H
                    RET
;----------------------SET  ADDRESS---------------------------------
SET_X_ADDR:         CLR   LCD_DI
                    MOV   A,R6
                    MOV   P0,A
                    ACALL LCD_CLK
                    RET

SET_Y_ADDR:         CLR   LCD_DI
                    MOV   A,R7
                    MOV   P0,A
                    ACALL LCD_CLK
                    RET
;-------------------------------------------------------------------

LCD_CLR:            ACALL PAGE_L
                    MOV   R6,#10111000B
      AGAIN2:       ACALL SET_X_ADDR
                    MOV   R7,#01000000B
                    ACALL SET_Y_ADDR
                    SETB  LCD_DI
      AGAIN1:       MOV   P0,#00H
                    ACALL LCD_CLK
                    INC   R7
                    CJNE  R7,#10000000B,AGAIN1
                    MOV   C,00H
                    JC    INV1
                    ACALL PAGE_R
      AGAIN3:       CJNE  R6,#11000000B,AGAIN2
                    CLR   LCD_DI
                    RET
      INV1:         ACALL PAGE_L
                    INC   R6
                    AJMP  AGAIN3

LCD_CLK:            SETB  LCD_EN
                    ACALL LCD_DELAY
                    CLR   LCD_EN
                    ACALL LCD_DELAY
                    RET

;-----------------------------------DELAY---------------------------

LCD_DELAY:          MOV   REG7,#02H
LCD_DELAY_1:              MOV   REG6,#0E6H
LCD_DELAY_2:              NOP
                    NOP
                    DJNZ  REG6,LCD_DELAY_2
                    DJNZ  REG7,LCD_DELAY_1
                    RET
DELAY_10ms:         MOV   REG7,#0AH
DELAY_10ms_1:       MOV   REG6,#0E6H
DELAY_10ms_2:       NOP
                    NOP
                    DJNZ  REG6,DELAY_10ms_2
```

```
                DJNZ   REG7,DELAY_10ms_1
                RET
DELAY_100ms:          MOV    REG7,#100
DELAY_100ms_1:  MOV    REG6,#0E6H
DELAY_100ms_2:  NOP
                NOP
                DJNZ   REG6,DELAY_100ms_2
                DJNZ   REG7,DELAY_100Ms_1
                RET
DELAY_500ms:          MOV    REG5,#50
DELAY_500ms_1:  ACALL  DELAY_10ms
                DJNZ   REG5,DELAY_500ms_1
                RET
DELAY_1s:       MOV    REG5,#100
DELAY_1s_1:     ACALL  DELAY_10ms
                DJNZ   REG5,DELAY_1s_1
                RET
;-------------------------------------------------------------------
WR_CHAR:        MOV    R1,#00H
WR_CHAR_1:      SETB   LCD_DI
                CLR    A
                MOVC   A,@A+DPTR
                MOV    P0,A
                ACALL  LCD_CLK
                INC    DPTR
                INC    R1
                CJNE   R1,#06H,WR_CHAR_1
                CLR    LCD_DI
                RET

PRINT_DATA:     ACALL  SET_Y_ADDR
                ACALL  SET_X_ADDR
                ACALL  WR_CHAR
                MOV    A,#06H
                ADD    A,R7
                MOV    R7,A
                CJNE   R7,#01111100B,PRINT_NORM
                AJMP   INV3
PRINT_NORM:     CJNE   R7,#80H,PRINT_OUT
                MOV    R7,#40H
                MOV    C,00H
                JC     INV2
                ACALL  PAGE_R
PRINT_OUT:      RET
      INV2:     ACALL  PAGE_L
                INC    R6
                AJMP   PRINT_OUT
      INV3:     ACALL  PAGE_L
                MOV    C,05H
                JNC    PAGE_TO
                AJMP   PAGE_MESS
PAGE_TO:        MOV    R7,#01100010B
                INC    R6
                AJMP   PRINT_OUT
PAGE_MESS:      MOV    R7,#01000100B
                INC    R6
                INC    R6
                AJMP   PRINT_OUT

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

```
MAIN:           ACALL LCD_CLR
PAGE1:          MOV   R6,#10111001B          ; Line 1
                MOV   R7,#01000100B          ; Char 4
                MOV   DPTR,#CHAR_EBE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESR
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESY
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESR
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESM
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESB
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESR
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                ACALL PAGE_L
                MOV   R6,#10111011B          ; Line 3
                MOV   R7,#01000100B          ; Char 4
                MOV   DPTR,#CHAR_EBP
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESH
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBL31
                ACALL PRINT_DATA
```

```
                MOV   R0,#30H
                ACALL   GET_NUM                 ; Get Number Phone
                MOV   @R0,#149                  ; END

                ACALL PAGE_L
                MOV   R6,#10111110B             ; Line 7
                MOV   R7,#01011100B             ; Char 28
                MOV   DPTR,#CHAR_EBP
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBW
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                LCALL DIAL_MODEM

PAGE2:          ACALL LCD_CLR                   ; Connection ACK
                MOV   R6,#10111011B             ; Line 3
                MOV   R7,#01010110B             ; Char 22
                MOV   DPTR,#CHAR_EBC
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESC
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESG
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
```

```
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s

                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s

                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s
                ACALL DELAY_1s

PAGE4:          ACALL LCD_CLR                   ; Header Mail
                MOV   R6,#10111001B             ; Line 1
                MOV   R7,#01000100B             ; Char 4
                MOV   DPTR,#CHAR_EBT
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBL31
                ACALL PRINT_DATA

                MOV   R6,#10111011B             ; Line 3
                MOV   R7,#01000100B             ; Char 4
                MOV   DPTR,#CHAR_EBF
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESR
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESM
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBL31
                ACALL PRINT_DATA

                MOV   R6,#10111110B             ; Line 6
                MOV   R7,#01000100B             ; Char 4
                MOV   DPTR,#CHAR_EBS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESB
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESJ
                ACALL PRINT_DATA
```

```
        MOV   DPTR,#CHAR_EBL31
        ACALL PRINT_DATA

        ACALL   PAGE_L
        MOV   R6,#10111001B          ; Line 1
        MOV   R7,#01100010B          ; Char 34
        CLR   02H                    ; Set English char
        CLR   01H                    ; Set small char
        CLR   06H                    ; Block Thai

        MOV   R0,#30H
        ACALL GET_CHAR               ; Get Data
        MOV   @R0,#149               ; END
        LCALL SEND_TO                ; SEND_TO

        ACALL PAGE_L
        MOV   R6,#10111011B          ; Line 3
        MOV   R7,#01100010B          ; Char 34
        CLR   06H                    ; Block Thai
        MOV   R0,#30H
        ACALL GET_CHAR               ; Get Data
        MOV   @R0,#149               ; END
        LCALL SEND_FROM              ; SEND_FROM

        ACALL PAGE_L
        MOV   R6,#10111110B          ; Line 5
        MOV   R7,#01100010B          ; Char 34
        SETB  06H                    ; Open Thai
        MOV   R0,#30H
        ACALL GET_CHAR               ; Get Data
        MOV   @R0,#149               ; END
        LCALL SEND_SUBJ              ; SEND_SUBJ

        ACALL DELAY_1s
        ACALL DELAY_1s

PAGE5:  ACALL LCD_CLR                ; Message Mail
        MOV     R6,#10111010B        ; Line 2
        MOV   R7,#01000100B          ; Char 4
        CLR   02H                    ; Set English char
        CLR   01H                    ; Set small char
        SETB  05H
        SETB  06H                    ; Open Thai
        MOV   R0,#30H
        ACALL GET_CHAR               ; Get Data
        MOV   @R0,#149               ; END
        CLR   05H
        LCALL SEND_MESS

PAGE6:  ACALL LCD_CLR                ; Sendind Mail
        MOV   R6,#10111100B          ; Line 4
        MOV   R7,#01011100B          ; Char 28
        MOV   DPTR,#CHAR_EBP
        ACALL PRINT_DATA
        MOV   DPTR,#CHAR_ESL
        ACALL PRINT_DATA
        MOV   DPTR,#CHAR_ESE
        ACALL PRINT_DATA
        MOV   DPTR,#CHAR_ESA
        ACALL PRINT_DATA
        MOV   DPTR,#CHAR_ESS
```

```
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBW
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                ACALL PAGE_L
                MOV   R6,#10111110B          ; Line 6
                MOV   R7,#01010110B          ; Char 22
                MOV   DPTR,#CHAR_EBS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESD
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESG
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBM
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL
                ACALL PRINT_DATA


PAGE7:          AJMP  PAGE7_1
PAGE7_1:        ACALL LCD_CLR                ; Send OK
                MOV   R6,#10111001B          ; Line 1
                MOV   R7,#01001010B          ; Char 10
                MOV   DPTR,#PAGE7_1_11
                ACALL PRINT_DATA
                MOV   DPTR,#PAGE7_1_12
                ACALL PRINT_DATA
                MOV   DPTR,#PAGE7_1_13
                ACALL PRINT_DATA
                ACALL PAGE_L
```

```
MOV   R6,#10111010B          ; Line 2
MOV   R7,#01001010B          ; Char 10
MOV   DPTR,#PAGE7_1_21
ACALL PRINT_DATA
MOV   DPTR,#PAGE7_1_22
ACALL PRINT_DATA
MOV   DPTR,#SPACE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_EBM
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESS
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESS
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESA
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESG
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESE
ACALL PRINT_DATA
MOV   DPTR,#SPACE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESI
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESS
ACALL PRINT_DATA
ACALL PAGE_L
MOV   R6,#10111100B          ; Line 4
MOV   R7,#01011100B          ; Char 28
MOV   DPTR,#CHAR_ESC
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESO
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESM
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESP
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESL
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_EST
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESL
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESY
ACALL PRINT_DATA
MOV   DPTR,#SPACE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESS
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESE
ACALL PRINT_DATA
MOV   DPTR,#CHAR_ESN
ACALL PRINT_DATA
MOV   DPTR,#CHAR_EST
ACALL PRINT_DATA
```

```
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

PAGE8:          ACALL LCD_CLR                   ; Send again ?
                MOV   R6,#10111100B             ; Line 4
                MOV   R7,#01000100B             ; Char 4
                MOV   DPTR,#CHAR_EBD
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESY
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESW
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESD
                ACALL PRINT_DATA

                ACALL PAGE_L
                MOV   R6,#10111110B             ; Line 6
                MOV   R7,#01000100B             ; Char 4
                MOV   DPTR,#CHAR_ESM
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESG
```

```
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESI
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBL43
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EB9
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBY
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL43
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EBN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EB0
                ACALL PRINT_DATA

PAGE8_A:        LCALL GET_DATA
                MOV   A,DATA
PAGE8_Y:        CJNE  A,#35H,PAGE8_N
                AJMP  PAGE4
PAGE8_N:        CJNE  A,#31H,PAGE8_O
                AJMP  PAGE9
PAGE8_O:        AJMP  PAGE8_A

PAGE9:          ACALL LCD_CLR                 ; Shut down
                MOV   R6,#10111100B           ; Line 4
                MOV   R7,#01010000B           ; Char 16
                MOV   DPTR,#CHAR_EBP
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESE
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESS
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESH
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_EST
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESD
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESW
                ACALL PRINT_DATA
```

```
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                ACALL PAGE_L
                MOV   R6,#10111110B          ; Line 6
                MOV   R7,#01100010B          ; Char 34
                MOV   DPTR,#CHAR_EBT
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESH
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESA
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESN
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESK
                ACALL PRINT_DATA
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESY
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESO
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESU
                ACALL PRINT_DATA
                MOV   DPTR,#CHAR_ESL42
                ACALL PRINT_DATA

                SJMP  $

;-------------------------------------------------------------------
BACKSPACE:      ACALL DEL6_R7
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                ACALL DEL6_R7
                RET
DEL6_R7:        CLR   C
                MOV   A,R7
                CJNE  A,#40H,DEL_NORMAL
                ACALL PAGE_L
                MOV   R7,#80H
                ACALL SET_X_ADDR
                ACALL SET_Y_ADDR
DEL_NORMAL:     SUBB  A,#06H
                MOV   R7,A
                RET

BACKSPACE_1:    ACALL DEL6_R7
                DEC   R6
                ACALL SET_X_ADDR
                ACALL SET_Y_ADDR
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
                ACALL DEL6_R7
                INC   R6
                INC   R6
                ACALL SET_X_ADDR
                ACALL SET_Y_ADDR
                MOV   DPTR,#SPACE
                ACALL PRINT_DATA
```

```
                        DEC   R6
                        ACALL BACKSPACE
                        RET
;-------------------------------------------------------------------
REC_NUM:                MOV   @R0,A
                        INC   R0
                        AJMP  GET_NUM

REC_CHAR:               MOV   @R0,A
                        INC   R0
                        AJMP  GET_CHAR

;-------------------------------------------------------------------
GET_NUM:                ACALL DELAY_100ms
                        ACALL GET_DATA
                        MOV   A,DATA

            GET_1:      CJNE  A,#16H,GET_2
                        MOV   DPTR,#CHAR_ES1
                        ACALL PRINT_DATA
                        MOV   A,#'1'
                        AJMP  REC_NUM
            GET_2:      CJNE  A,#1EH,GET_3
                        MOV   DPTR,#CHAR_ES2
                        ACALL PRINT_DATA
                        MOV   A,#'2'
                        AJMP  REC_NUM
            GET_3:      CJNE  A,#26H,GET_4
                        MOV   DPTR,#CHAR_ES3
                        ACALL PRINT_DATA
                        MOV   A,#'3'
                        AJMP  REC_NUM
            GET_4:      CJNE  A,#25H,GET_5
                        MOV   DPTR,#CHAR_ES4
                        ACALL PRINT_DATA
                        MOV   A,#'4'
                        AJMP  REC_NUM
            GET_5:      CJNE  A,#2EH,GET_6
                        MOV   DPTR,#CHAR_ES5
                        ACALL PRINT_DATA
                        MOV   A,#'5'
                        AJMP  REC_NUM
            GET_6:      CJNE  A,#36H,GET_7
                        MOV   DPTR,#CHAR_ES6
                        ACALL PRINT_DATA
                        MOV   A,#'6'
                        AJMP  REC_NUM
            GET_7:      CJNE  A,#3DH,GET_8
                        MOV   DPTR,#CHAR_ES7
                        ACALL PRINT_DATA
                        MOV   A,#'7'
                        AJMP  REC_NUM
            GET_8:      CJNE  A,#3EH,GET_9
                        MOV   DPTR,#CHAR_ES8
                        ACALL PRINT_DATA
                        MOV   A,#'8'
                        AJMP  REC_NUM
            GET_9:      CJNE  A,#46H,GET_0
                        MOV   DPTR,#CHAR_ES9
                        ACALL PRINT_DATA
                        MOV   A,#'9'
```

```
                    AJMP   REC_NUM
          GET_0:    CJNE   A,#45H,GET_COM
                    MOV    DPTR,#CHAR_ES0
                    ACALL  PRINT_DATA
                    MOV    A,#'0'
                    AJMP   REC_NUM
          GET_COM:  CJNE   A,#41H,GET_ENT
                    MOV    DPTR,#CHAR_ESL41
                    ACALL  PRINT_DATA
                    MOV    A,#','
                    AJMP   GET_NUM
          GET_ENT:  CJNE   A,#59H,GET_BS
                    RET
          GET_BS:   CJNE   A,#66H,OUT_NUM
                    ACALL  BACKSPACE
                    DEC    R0
                    AJMP   GET_NUM
          OUT_NUM:  AJMP   GET_NUM
;-------------------------------------------------------------------

     GET_CHAR:      ACALL  DELAY_100ms
                    ACALL  GET_DATA
                    MOV    A,DATA

     CHAR_ENTER:    CJNE   A,#59H,CHAR_SPACE
                    RET
     CHAR_SPACE:    CJNE   A,#29H,CHAR_TAB
                    MOV    DPTR,#SPACE
                    ACALL  PRINT_DATA
                    MOV    A,#' '
                    LJMP   REC_CHAR
     CHAR_TAB:      CJNE   A,#0DH,JUMP_CHAR_ESC
                    AJMP   CHAR_TAB_1
     JUMP_CHAR_ESC: LJMP   CHAR_ESCA
     CHAR_TAB_1:    MOV    DPTR,#SPACE
                    LCALL  PRINT_DATA
                    MOV    DPTR,#SPACE
                    LCALL  PRINT_DATA
                    MOV    DPTR,#SPACE
                    LCALL  PRINT_DATA
                    MOV    DPTR,#SPACE
                    LCALL  PRINT_DATA
                    MOV    A,#09
                    LJMP   REC_CHAR
     CHAR_ESCA:     CJNE   A,#76H,CHAR_BS
                    MOV    C,05H
                    JC     ESC_1
                    LJMP   PAGE4
     ESC_1:         LJMP   PAGE5

     CHAR_BS:       CJNE   A,#66H,CHAR_CAP
                    MOV    C,06H
                    JC     CHAR_BS_1
                    LCALL  BACKSPACE
                    DEC    R0
                    LJMP   GET_CHAR
     CHAR_BS_1:     LCALL  BACKSPACE_1
                    DEC    R0
                    LJMP   GET_CHAR
     CHAR_CAP:      CJNE   A,#58H,CHAR_EN_TH
                    MOV    C,01H
```

```
                    JNC   CHANGE_BIG
                    CLR   01H
                    LJMP  GET_CHAR
CHANGE_BIG:         SETB  01H
                    LJMP  GET_CHAR

CHAR_EN_TH:         CJNE  A,#0EH,CHOOSE_CHAR
                    MOV   C,06H
                    JC    NON_BLOCK
                    LJMP  GET_CHAR
NON_BLOCK:          MOV   C,02H
                    JNC   CHANGE_THAI
                    CLR   02H
                    LJMP  GET_CHAR
CHANGE_THAI:        SETB  02H
                    LJMP  GET_CHAR

CHOOSE_CHAR:        MOV   C,02H
                    JNC   CHAR_ENG
                    LJMP  CHAR_THAI

CHAR_ENG:           MOV   C,01H
                    JNC   CHAR_ES_1
                    LJMP  CHAR_EB_1

CHAR_THAI:          MOV   C,01H
                    JNC   THAI_1
                    LJMP  CHAR_TB_1
THAI_1:             LJMP  CHAR_TS_1
;-------------------------------------------------------------------

CHAR_ES_1:          CJNE  A,#1CH,CHAR_ES_2
                    MOV   DPTR,#CHAR_ESA
                    LCALL PRINT_DATA
                    MOV   A,#'a'
                    LJMP  REC_CHAR
CHAR_ES_2:          CJNE  A,#32H,CHAR_ES_3
                    MOV   DPTR,#CHAR_ESB
                    LCALL PRINT_DATA
                    MOV   A,#'b'
                    LJMP  REC_CHAR
CHAR_ES_3:          CJNE  A,#21H,CHAR_ES_4
                    MOV   DPTR,#CHAR_ESC
                    LCALL PRINT_DATA
                    MOV   A,#'c'
                    LJMP  REC_CHAR
CHAR_ES_4:          CJNE  A,#23H,CHAR_ES_5
                    MOV   DPTR,#CHAR_ESD
                    LCALL PRINT_DATA
                    MOV   A,#'d'
                    LJMP  REC_CHAR
CHAR_ES_5:          CJNE  A,#24H,CHAR_ES_6
                    MOV   DPTR,#CHAR_ESE
                    LCALL PRINT_DATA
                    MOV   A,#'e'
                    LJMP  REC_CHAR
CHAR_ES_6:          CJNE  A,#2BH,CHAR_ES_7
                    MOV   DPTR,#CHAR_ESF
                    LCALL PRINT_DATA
                    MOV   A,#'f'
```

```
                LJMP   REC_CHAR
CHAR_ES_7:      CJNE   A,#34H,CHAR_ES_8
                MOV    DPTR,#CHAR_ESG
                LCALL  PRINT_DATA
                MOV    A,#'g'
                LJMP   REC_CHAR
CHAR_ES_8:      CJNE   A,#33H,CHAR_ES_9
                MOV    DPTR,#CHAR_ESH
                LCALL  PRINT_DATA
                MOV    A,#'h'
                LJMP   REC_CHAR
CHAR_ES_9:      CJNE   A,#43H,CHAR_ES_10
                MOV    DPTR,#CHAR_ESI
                LCALL  PRINT_DATA
                MOV    A,#'i'
                LJMP   REC_CHAR
CHAR_ES_10:     CJNE   A,#3BH,CHAR_ES_11
                MOV    DPTR,#CHAR_ESJ
                LCALL  PRINT_DATA
                MOV    A,#'j'
                LJMP   REC_CHAR
CHAR_ES_11:     CJNE   A,#42H,CHAR_ES_12
                MOV    DPTR,#CHAR_ESK
                LCALL  PRINT_DATA
                MOV    A,#'k'
                LJMP   REC_CHAR
CHAR_ES_12:     CJNE   A,#4BH,CHAR_ES_13
                MOV    DPTR,#CHAR_ESL
                LCALL  PRINT_DATA
                MOV    A,#'l'
                LJMP   REC_CHAR
CHAR_ES_13:     CJNE   A,#3AH,CHAR_ES_14
                MOV    DPTR,#CHAR_ESM
                LCALL  PRINT_DATA
                MOV    A,#'m'
                LJMP   REC_CHAR
CHAR_ES_14:     CJNE   A,#31H,CHAR_ES_15
                MOV    DPTR,#CHAR_ESN
                LCALL  PRINT_DATA
                MOV    A,#'n'
                LJMP   REC_CHAR
CHAR_ES_15:     CJNE   A,#44H,CHAR_ES_16
                MOV    DPTR,#CHAR_ESO
                LCALL  PRINT_DATA
                MOV    A,#'o'
                LJMP   REC_CHAR
CHAR_ES_16:     CJNE   A,#4DH,CHAR_ES_17
                MOV    DPTR,#CHAR_ESP
                LCALL  PRINT_DATA
                MOV    A,#'p'
                LJMP   REC_CHAR
CHAR_ES_17:     CJNE   A,#15H,CHAR_ES_18
                MOV    DPTR,#CHAR_ESQ
                LCALL  PRINT_DATA
                MOV    A,#'q'
                LJMP   REC_CHAR
CHAR_ES_18:     CJNE   A,#2DH,CHAR_ES_19
                MOV    DPTR,#CHAR_ESR
                LCALL  PRINT_DATA
                MOV    A,#'r'
                LJMP   REC_CHAR
```

```
CHAR_ES_19:     CJNE  A,#1BH,CHAR_ES_20
                MOV   DPTR,#CHAR_ESS
                LCALL PRINT_DATA
                MOV   A,#'s'
                LJMP  REC_CHAR
CHAR_ES_20:     CJNE  A,#2CH,CHAR_ES_21
                MOV   DPTR,#CHAR_EST
                LCALL PRINT_DATA
                MOV   A,#'t'
                LJMP  REC_CHAR
CHAR_ES_21:     CJNE  A,#3CH,CHAR_ES_22
                MOV   DPTR,#CHAR_ESU
                LCALL PRINT_DATA
                MOV   A,#'u'
                LJMP  REC_CHAR
CHAR_ES_22:     CJNE  A,#2AH,CHAR_ES_23
                MOV   DPTR,#CHAR_ESV
                LCALL PRINT_DATA
                MOV   A,#'v'
                LJMP  REC_CHAR
CHAR_ES_23:     CJNE  A,#1DH,CHAR_ES_24
                MOV   DPTR,#CHAR_ESW
                LCALL PRINT_DATA
                MOV   A,#'w'
                LJMP  REC_CHAR
CHAR_ES_24:     CJNE  A,#22H,CHAR_ES_25
                MOV   DPTR,#CHAR_ESX
                LCALL PRINT_DATA
                MOV   A,#'x'
                LJMP  REC_CHAR
CHAR_ES_25:     CJNE  A,#35H,CHAR_ES_26
                MOV   DPTR,#CHAR_ESY
                LCALL PRINT_DATA
                MOV   A,#'y'
                LJMP  REC_CHAR
CHAR_ES_26:     CJNE  A,#1AH,CHAR_ES_27
                MOV   DPTR,#CHAR_ESZ
                LCALL PRINT_DATA
                MOV   A,#'z'
                LJMP  REC_CHAR
CHAR_ES_27:     CJNE  A,#45H,CHAR_ES_28
                MOV   DPTR,#CHAR_ES0
                LCALL PRINT_DATA
                MOV   A,#'0'
                LJMP  REC_CHAR
CHAR_ES_28:     CJNE  A,#16H,CHAR_ES_29
                MOV   DPTR,#CHAR_ES1
                LCALL PRINT_DATA
                MOV   A,#'1'
                LJMP  REC_CHAR
CHAR_ES_29:     CJNE  A,#1EH,CHAR_ES_30
                MOV   DPTR,#CHAR_ES2
                LCALL PRINT_DATA
                MOV   A,#'2'
                LJMP  REC_CHAR
CHAR_ES_30:     CJNE  A,#26H,CHAR_ES_31
                MOV   DPTR,#CHAR_ES3
                LCALL PRINT_DATA
                MOV   A,#'3'
                LJMP  REC_CHAR
CHAR_ES_31:     CJNE  A,#25H,CHAR_ES_32
```

```
                MOV   DPTR,#CHAR_ES4
                LCALL PRINT_DATA
                MOV   A,#'4'
                LJMP  REC_CHAR
CHAR_ES_32:     CJNE  A,#2EH,CHAR_ES_33
                MOV   DPTR,#CHAR_ES5
                LCALL PRINT_DATA
                MOV   A,#'5'
                LJMP  REC_CHAR
CHAR_ES_33:     CJNE  A,#36H,CHAR_ES_34
                MOV   DPTR,#CHAR_ES6
                LCALL PRINT_DATA
                MOV   A,#'6'
                LJMP  REC_CHAR
CHAR_ES_34:     CJNE  A,#3DH,CHAR_ES_35
                MOV   DPTR,#CHAR_ES7
                LCALL PRINT_DATA
                MOV   A,#'7'
                LJMP  REC_CHAR
CHAR_ES_35:     CJNE  A,#3EH,CHAR_ES_36
                MOV   DPTR,#CHAR_ES8
                LCALL PRINT_DATA
                MOV   A,#'8'
                LJMP  REC_CHAR
CHAR_ES_36:     CJNE  A,#46H,CHAR_ES_37
                MOV   DPTR,#CHAR_ES9
                LCALL PRINT_DATA
                MOV   A,#'9'
                LJMP  REC_CHAR
CHAR_ES_37:     CJNE  A,#4EH,CHAR_ES_38
                MOV   DPTR,#CHAR_ESL11
                LCALL PRINT_DATA
                MOV   A,#'-'
                LJMP  REC_CHAR
CHAR_ES_38:     CJNE  A,#55H,CHAR_ES_39
                MOV   DPTR,#CHAR_ESL12
                LCALL PRINT_DATA
                MOV   A,#'='
                LJMP  REC_CHAR
CHAR_ES_39:     CJNE  A,#5DH,CHAR_ES_40
                MOV   DPTR,#CHAR_ESL13
                LCALL PRINT_DATA
                MOV   A,#'\'
                LJMP  REC_CHAR
CHAR_ES_40:     CJNE  A,#54H,CHAR_ES_41
                MOV   DPTR,#CHAR_ESL21
                LCALL PRINT_DATA
                MOV   A,#'['
                LJMP  REC_CHAR
CHAR_ES_41:     CJNE  A,#5BH,CHAR_ES_42
                MOV   DPTR,#CHAR_ESL22
                LCALL PRINT_DATA
                MOV   A,#']'
                LJMP  REC_CHAR
CHAR_ES_42:     CJNE  A,#4CH,CHAR_ES_43
                MOV   DPTR,#CHAR_ESL31
                LCALL PRINT_DATA
                MOV   A,#';'
                LJMP  REC_CHAR
CHAR_ES_43:     CJNE  A,#52H,CHAR_ES_44
                MOV   DPTR,#CHAR_ESL32
```

```
                        LCALL PRINT_DATA
                        MOV   A,#''''
                        LJMP  REC_CHAR
CHAR_ES_44:             CJNE  A,#41H,CHAR_ES_45
                        MOV   DPTR,#CHAR_ESL41
                        LCALL PRINT_DATA
                        MOV   A,#','
                        LJMP  REC_CHAR
CHAR_ES_45:             CJNE  A,#49H,CHAR_ES_46
                        MOV   DPTR,#CHAR_ESL42
                        LCALL PRINT_DATA
                        MOV   A,#'.'
                        LJMP  REC_CHAR
CHAR_ES_46:             CJNE  A,#4AH,CHAR_ES_47
                        MOV   DPTR,#CHAR_ESL43
                        LCALL PRINT_DATA
                        MOV   A,#'/'
                        LJMP  REC_CHAR
CHAR_ES_47:             CJNE  A,#35H,OUT_CHAR_ES
                        MOV   DPTR,#CHAR_ES9
                        LCALL PRINT_DATA
                        MOV   A,#'?'
                        LJMP  REC_CHAR
OUT_CHAR_ES:            LJMP  GET_CHAR
;-------------------------------------------------------------------

CHAR_EB_1:              CJNE  A,#1CH,CHAR_EB_2
                        MOV   DPTR,#CHAR_EBA
                        LCALL PRINT_DATA
                        MOV   A,#'A'
                        LJMP  REC_CHAR
CHAR_EB_2:              CJNE  A,#32H,CHAR_EB_3
                        MOV   DPTR,#CHAR_EBB
                        LCALL PRINT_DATA
                        MOV   A,#'B'
                        LJMP  REC_CHAR
CHAR_EB_3:              CJNE  A,#21H,CHAR_EB_4
                        MOV   DPTR,#CHAR_EBC
                        LCALL PRINT_DATA
                        MOV   A,#'C'
                        LJMP  REC_CHAR
CHAR_EB_4:              CJNE  A,#23H,CHAR_EB_5
                        MOV   DPTR,#CHAR_EBD
                        LCALL PRINT_DATA
                        MOV   A,#'D'
                        LJMP  REC_CHAR
CHAR_EB_5:              CJNE  A,#24H,CHAR_EB_6
                        MOV   DPTR,#CHAR_EBE
                        LCALL PRINT_DATA
                        MOV   A,#'E'
                        LJMP  REC_CHAR
CHAR_EB_6:              CJNE  A,#2BH,CHAR_EB_7
                        MOV   DPTR,#CHAR_EBF
                        LCALL PRINT_DATA
                        MOV   A,#'F'
                        LJMP  REC_CHAR
CHAR_EB_7:              CJNE  A,#34H,CHAR_EB_8
                        MOV   DPTR,#CHAR_EBG
                        LCALL PRINT_DATA
                        MOV   A,#'G'
                        LJMP  REC_CHAR
```

```
CHAR_EB_8:      CJNE  A,#33H,CHAR_EB_9
                MOV   DPTR,#CHAR_EBH
                LCALL PRINT_DATA
                MOV   A,#'H'
                LJMP  REC_CHAR
CHAR_EB_9:      CJNE  A,#43H,CHAR_EB_10
                MOV   DPTR,#CHAR_EBI
                LCALL PRINT_DATA
                MOV   A,#'I'
                LJMP  REC_CHAR
CHAR_EB_10:     CJNE  A,#3BH,CHAR_EB_11
                MOV   DPTR,#CHAR_EBJ
                LCALL PRINT_DATA
                MOV   A,#'J'
                LJMP  REC_CHAR
CHAR_EB_11:     CJNE  A,#42H,CHAR_EB_12
                MOV   DPTR,#CHAR_EBK
                LCALL PRINT_DATA
                MOV   A,#'K'
                LJMP  REC_CHAR
CHAR_EB_12:     CJNE  A,#4BH,CHAR_EB_13
                MOV   DPTR,#CHAR_EBL
                LCALL PRINT_DATA
                MOV   A,#'L'
                LJMP  REC_CHAR
CHAR_EB_13:     CJNE  A,#3AH,CHAR_EB_14
                MOV   DPTR,#CHAR_EBM
                LCALL PRINT_DATA
                MOV   A,#'M'
                LJMP  REC_CHAR
CHAR_EB_14:     CJNE  A,#31H,CHAR_EB_15
                MOV   DPTR,#CHAR_EBN
                LCALL PRINT_DATA
                MOV   A,#'N'
                LJMP  REC_CHAR
CHAR_EB_15:     CJNE  A,#44H,CHAR_EB_16
                MOV   DPTR,#CHAR_EBO
                LCALL PRINT_DATA
                MOV   A,#'O'
                LJMP  REC_CHAR
CHAR_EB_16:     CJNE  A,#4DH,CHAR_EB_17
                MOV   DPTR,#CHAR_EBP
                LCALL PRINT_DATA
                MOV   A,#'P'
                LJMP  REC_CHAR
CHAR_EB_17:     CJNE  A,#15H,CHAR_EB_18
                MOV   DPTR,#CHAR_EBQ
                LCALL PRINT_DATA
                MOV   A,#'Q'
                LJMP  REC_CHAR
CHAR_EB_18:     CJNE  A,#2DH,CHAR_EB_19
                MOV   DPTR,#CHAR_EBR
                LCALL PRINT_DATA
                MOV   A,#'R'
                LJMP  REC_CHAR
CHAR_EB_19:     CJNE  A,#1BH,CHAR_EB_20
                MOV   DPTR,#CHAR_EBS
                LCALL PRINT_DATA
                MOV   A,#'S'
                LJMP  REC_CHAR
CHAR_EB_20:     CJNE  A,#2CH,CHAR_EB_21
```

```
                    MOV   DPTR,#CHAR_EBT
                    LCALL PRINT_DATA
                    MOV   A,#'T'
                    LJMP  REC_CHAR
CHAR_EB_21:         CJNE  A,#3CH,CHAR_EB_22
                    MOV   DPTR,#CHAR_EBU
                    LCALL PRINT_DATA
                    MOV   A,#'U'
                    LJMP  REC_CHAR
CHAR_EB_22:         CJNE  A,#2AH,CHAR_EB_23
                    MOV   DPTR,#CHAR_EBV
                    LCALL PRINT_DATA
                    MOV   A,#'V'
                    LJMP  REC_CHAR
CHAR_EB_23:         CJNE  A,#1DH,CHAR_EB_24
                    MOV   DPTR,#CHAR_EBW
                    LCALL PRINT_DATA
                    MOV   A,#'W'
                    LJMP  REC_CHAR
CHAR_EB_24:         CJNE  A,#22H,CHAR_EB_25
                    MOV   DPTR,#CHAR_EBX
                    LCALL PRINT_DATA
                    MOV   A,#'X'
                    LJMP  REC_CHAR
CHAR_EB_25:         CJNE  A,#35H,CHAR_EB_26
                    MOV   DPTR,#CHAR_EBY
                    LCALL PRINT_DATA
                    MOV   A,#'Y'
                    LJMP  REC_CHAR
CHAR_EB_26:         CJNE  A,#1AH,CHAR_EB_27
                    MOV   DPTR,#CHAR_EBZ
                    LCALL PRINT_DATA
                    MOV   A,#'Z'
                    LJMP  REC_CHAR
CHAR_EB_27:         CJNE  A,#45H,CHAR_EB_28
                    MOV   DPTR,#CHAR_EB0
                    LCALL PRINT_DATA
                    MOV   A,#')'
                    LJMP  REC_CHAR
CHAR_EB_28:         CJNE  A,#16H,CHAR_EB_29
                    MOV   DPTR,#CHAR_EB1
                    LCALL PRINT_DATA
                    MOV   A,#'!'
                    LJMP  REC_CHAR
CHAR_EB_29:         CJNE  A,#1EH,CHAR_EB_30
                    MOV   DPTR,#CHAR_EB2
                    LCALL PRINT_DATA
                    MOV   A,#'@'
                    LJMP  REC_CHAR
CHAR_EB_30:         CJNE  A,#26H,CHAR_EB_31
                    MOV   DPTR,#CHAR_EB3
                    LCALL PRINT_DATA
                    MOV   A,#'#'
                    LJMP  REC_CHAR
CHAR_EB_31:         CJNE  A,#25H,CHAR_EB_32
                    MOV   DPTR,#CHAR_EB4
                    LCALL PRINT_DATA
                    MOV   A,#'$'
                    LJMP  REC_CHAR
CHAR_EB_32:         CJNE  A,#2EH,CHAR_EB_33
                    MOV   DPTR,#CHAR_EB5
```

```
                LCALL PRINT_DATA
                MOV   A,#'%'
                LJMP  REC_CHAR
CHAR_EB_33:     CJNE  A,#36H,CHAR_EB_34
                MOV   DPTR,#CHAR_EB6
                LCALL PRINT_DATA
                MOV   A,#'^'
                LJMP  REC_CHAR
CHAR_EB_34:     CJNE  A,#3DH,CHAR_EB_35
                MOV   DPTR,#CHAR_EB7
                LCALL PRINT_DATA
                MOV   A,#'&'
                LJMP  REC_CHAR
CHAR_EB_35:     CJNE  A,#3EH,CHAR_EB_36
                MOV   DPTR,#CHAR_EB8
                LCALL PRINT_DATA
                MOV   A,#'*'
                LJMP  REC_CHAR
CHAR_EB_36:     CJNE  A,#46H,CHAR_EB_37
                MOV   DPTR,#CHAR_EB9
                LCALL PRINT_DATA
                MOV   A,#'('
                LJMP  REC_CHAR
CHAR_EB_37:     CJNE  A,#4EH,CHAR_EB_38
                MOV   DPTR,#CHAR_EBL11
                LCALL PRINT_DATA
                MOV   A,#'_'
                LJMP  REC_CHAR
CHAR_EB_38:     CJNE  A,#55H,CHAR_EB_39
                MOV   DPTR,#CHAR_EBL12
                LCALL PRINT_DATA
                MOV   A,#'+'
                LJMP  REC_CHAR
CHAR_EB_39:     CJNE  A,#5DH,CHAR_EB_40
                MOV   DPTR,#CHAR_EBL13
                LCALL PRINT_DATA
                MOV   A,#'|'
                LJMP  REC_CHAR
CHAR_EB_40:     CJNE  A,#54H,CHAR_EB_41
                MOV   DPTR,#CHAR_EBL21
                LCALL PRINT_DATA
                MOV   A,#'{'
                LJMP  REC_CHAR
CHAR_EB_41:     CJNE  A,#5BH,CHAR_EB_42
                MOV   DPTR,#CHAR_EBL22
                LCALL PRINT_DATA
                MOV   A,#'}'
                LJMP  REC_CHAR
CHAR_EB_42:     CJNE  A,#4CH,CHAR_EB_43
                MOV   DPTR,#CHAR_EBL31
                LCALL PRINT_DATA
                MOV   A,#':'
                LJMP  REC_CHAR
CHAR_EB_43:     CJNE  A,#52H,CHAR_EB_44
                MOV   DPTR,#CHAR_EBL32
                LCALL PRINT_DATA
                MOV   A,#'"'
                LJMP  REC_CHAR
CHAR_EB_44:     CJNE  A,#41H,CHAR_EB_45
                MOV   DPTR,#CHAR_EBL41
                LCALL PRINT_DATA
```

```
                        MOV   A,#'<'
                        LJMP  REC_CHAR
CHAR_EB_45:             CJNE  A,#49H,CHAR_EB_46
                        MOV   DPTR,#CHAR_EBL42
                        LCALL PRINT_DATA
                        MOV   A,#'>'
                        LJMP  REC_CHAR
CHAR_EB_46:             CJNE  A,#4AH,CHAR_EB_47
                        MOV   DPTR,#CHAR_EBL43
                        LCALL PRINT_DATA
                        MOV   A,#'?'
                        LJMP  REC_CHAR
CHAR_EB_47:             CJNE  A,#35H,OUT_CHAR_EB
                        MOV   DPTR,#CHAR_EB9
                        LCALL PRINT_DATA
                        MOV   A,#'?'
                        LJMP  REC_CHAR
OUT_CHAR_EB:            LJMP  GET_CHAR
;-------------------------------------------------------------------
CHAR_TS_1:              CJNE  A,#1CH,CHAR_TS_2
                        MOV   DPTR,#CHAR_TSA
                        LCALL PRINT_DATA
                        MOV   A,#191
                        LJMP  REC_CHAR
CHAR_TS_2:              CJNE  A,#32H,CHAR_TS_3
                        MOV   DPTR,#CHAR_TSB
                        LCALL PRINT_SARA_UP
                        MOV   A,#212
                        LJMP  REC_CHAR
CHAR_TS_3:              CJNE  A,#21H,CHAR_TS_4
                        MOV   DPTR,#CHAR_TSC
                        LCALL PRINT_DATA
                        MOV   A,#225
                        LJMP  REC_CHAR
CHAR_TS_4:              CJNE  A,#23H,CHAR_TS_5
                        MOV   DPTR,#CHAR_TSD
                        LCALL PRINT_DATA
                        MOV   A,#161
                        LJMP  REC_CHAR
CHAR_TS_5:              CJNE  A,#24H,CHAR_TS_6
                        MOV   DPTR,#CHAR_TSE
                        LCALL PRINT_DATA
                        MOV   A,#211
                        LJMP  REC_CHAR
CHAR_TS_6:              CJNE  A,#2BH,CHAR_TS_7
                        MOV   DPTR,#CHAR_TSF
                        LCALL PRINT_DATA
                        MOV   A,#180
                        LJMP  REC_CHAR
CHAR_TS_7:              CJNE  A,#34H,CHAR_TS_8
                        MOV   DPTR,#CHAR_TSG
                        LCALL PRINT_DATA
                        MOV   A,#224
                        LJMP  REC_CHAR
CHAR_TS_8:              CJNE  A,#33H,CHAR_TS_9
                        MOV   DPTR,#CHAR_TSH
                        LCALL PRINT_SARA_UP
                        MOV   A,#233
                        LJMP  REC_CHAR
CHAR_TS_9:              CJNE  A,#43H,CHAR_TS_10
                        MOV   DPTR,#CHAR_TSI
```

```
                LCALL PRINT_DATA
                MOV   A,#195
                LJMP  REC_CHAR
CHAR_TS_10:     CJNE  A,#3BH,CHAR_TS_11
                MOV   DPTR,#CHAR_TSJ
                LCALL PRINT_SARA_UP
                MOV   A,#232
                LJMP  REC_CHAR
CHAR_TS_11:     CJNE  A,#42H,CHAR_TS_12
                MOV   DPTR,#CHAR_TSK
                LCALL PRINT_DATA
                MOV   A,#210
                LJMP  REC_CHAR
CHAR_TS_12:     CJNE  A,#4BH,CHAR_TS_13
                MOV   DPTR,#CHAR_TSL
                LCALL PRINT_DATA
                MOV   A,#202
                LJMP  REC_CHAR
CHAR_TS_13:     CJNE  A,#3AH,CHAR_TS_14
                MOV   DPTR,#CHAR_TSM
                LCALL PRINT_DATA
                MOV   A,#183
                LJMP  REC_CHAR
CHAR_TS_14:     CJNE  A,#31H,CHAR_TS_15
                MOV   DPTR,#CHAR_TSN
                LCALL PRINT_SARA_UP
                MOV   A,#215
                LJMP  REC_CHAR
CHAR_TS_15:     CJNE  A,#44H,CHAR_TS_16
                MOV   DPTR,#CHAR_TSO
                LCALL PRINT_DATA
                MOV   A,#185
                LJMP  REC_CHAR
CHAR_TS_16:     CJNE  A,#4DH,CHAR_TS_17
                MOV   DPTR,#CHAR_TSP
                LCALL PRINT_DATA
                MOV   A,#194
                LJMP  REC_CHAR
CHAR_TS_17:     CJNE  A,#15H,CHAR_TS_18
                MOV   DPTR,#CHAR_TSQ
                LCALL PRINT_DATA
                MOV   DPTR,#CHAR_TH_1
                LCALL PRINT_SARA_DOWN
                MOV   A,#230
                LJMP  REC_CHAR
CHAR_TS_18:     CJNE  A,#2DH,CHAR_TS_19
                MOV   DPTR,#CHAR_TSR
                LCALL PRINT_DATA
                MOV   A,#190
                LJMP  REC_CHAR
CHAR_TS_19:     CJNE  A,#1BH,CHAR_TS_20
                MOV   DPTR,#CHAR_TSS
                LCALL PRINT_DATA
                MOV   A,#203
                LJMP  REC_CHAR
CHAR_TS_20:     CJNE  A,#2CH,CHAR_TS_21
                MOV   DPTR,#CHAR_TST
                LCALL PRINT_DATA
                MOV   A,#208
                LJMP  REC_CHAR
CHAR_TS_21:     CJNE  A,#3CH,CHAR_TS_22
```

```
                    MOV   DPTR,#CHAR_TSU
                    LCALL PRINT_SARA_UP
                    MOV   A,#213
                    LJMP  REC_CHAR
CHAR_TS_22:         CJNE  A,#2AH,CHAR_TS_23
                    MOV   DPTR,#CHAR_TSV
                    LCALL PRINT_DATA
                    MOV   A,#205
                    LJMP  REC_CHAR
CHAR_TS_23:         CJNE  A,#1DH,CHAR_TS_24
                    MOV   DPTR,#CHAR_TSW
                    LCALL PRINT_DATA
                    MOV   A,#228
                    LJMP  REC_CHAR
CHAR_TS_24:         CJNE  A,#22H,CHAR_TS_25
                    MOV   DPTR,#CHAR_TSX
                    LCALL PRINT_DATA
                    MOV   A,#187
                    LJMP  REC_CHAR
CHAR_TS_25:         CJNE  A,#35H,CHAR_TS_26
                    MOV   DPTR,#CHAR_TSY
                    LCALL PRINT_SARA_UP
                    MOV   A,#209
                    LJMP  REC_CHAR
CHAR_TS_26:         CJNE  A,#1AH,CHAR_TS_27
                    MOV   DPTR,#CHAR_TSZ
                    LCALL PRINT_DATA
                    MOV   A,#188
                    LJMP  REC_CHAR
CHAR_TS_27:         CJNE  A,#45H,CHAR_TS_28
                    MOV   DPTR,#CHAR_TS0
                    LCALL PRINT_DATA
                    MOV   A,#168
                    LJMP  REC_CHAR
CHAR_TS_28:         CJNE  A,#16H,CHAR_TS_29
                    MOV   DPTR,#CHAR_TSK    ; TS1 = TSK + TH_1
                    LCALL PRINT_DATA
                    MOV   DPTR,#CHAR_TH_1
                    LCALL PRINT_SARA_DOWN
                    MOV   A,#229
                    LJMP  REC_CHAR
CHAR_TS_29:         CJNE  A,#1EH,CHAR_TS_30
                    MOV   DPTR,#CHAR_ESL43  ; TS2 = ESL43
                    LCALL PRINT_DATA
                    MOV   A,#'/'
                    LJMP  REC_CHAR
CHAR_TS_30:         CJNE  A,#26H,CHAR_TS_31
                    MOV   DPTR,#CHAR_ESL11  ; TS3 = ESL11
                    LCALL PRINT_DATA
                    MOV   A,#'_'
                    LJMP  REC_CHAR
CHAR_TS_31:         CJNE  A,#25H,CHAR_TS_32
                    MOV   DPTR,#CHAR_TS4
                    LCALL PRINT_DATA
                    MOV   A,#192
                    LJMP  REC_CHAR
CHAR_TS_32:         CJNE  A,#2EH,CHAR_TS_33
                    MOV   DPTR,#CHAR_TS5
                    LCALL PRINT_DATA
                    MOV   A,#182
                    LJMP  REC_CHAR
```

```
CHAR_TS_33:     CJNE  A,#36H,CHAR_TS_34
                MOV   DPTR,#CHAR_TS6
                LCALL PRINT_SARA_DOWN
                MOV   A,#216
                LJMP  REC_CHAR
CHAR_TS_34:     CJNE  A,#3DH,CHAR_TS_35
                MOV   DPTR,#CHAR_TS7
                LCALL PRINT_SARA_UP
                MOV   A,#214
                LJMP  REC_CHAR
CHAR_TS_35:     CJNE  A,#3EH,CHAR_TS_36
                MOV   DPTR,#CHAR_TS8
                LCALL PRINT_DATA
                MOV   A,#164
                LJMP  REC_CHAR
CHAR_TS_36:     CJNE  A,#46H,CHAR_TS_37
                MOV   DPTR,#CHAR_TS9
                LCALL PRINT_DATA
                MOV   A,#181
                LJMP  REC_CHAR
CHAR_TS_37:     CJNE  A,#4EH,CHAR_TS_38
                MOV   DPTR,#CHAR_TSL11
                LCALL PRINT_DATA
                MOV   A,#162
                LJMP  REC_CHAR
CHAR_TS_38:     CJNE  A,#55H,CHAR_TS_39
                MOV   DPTR,#CHAR_TSL12
                LCALL PRINT_DATA
                MOV   A,#170
                LJMP  REC_CHAR
CHAR_TS_39:     CJNE  A,#5DH,CHAR_TS_40
                MOV   DPTR,#CHAR_TSL13
                LCALL PRINT_DATA
                MOV   A,#163
                LJMP  REC_CHAR
CHAR_TS_40:     CJNE  A,#54H,CHAR_TS_41
                MOV   DPTR,#CHAR_TSL21
                LCALL PRINT_DATA
                MOV   A,#186
                LJMP  REC_CHAR
CHAR_TS_41:     CJNE  A,#5BH,CHAR_TS_42
                MOV   DPTR,#CHAR_TSL22
                LCALL PRINT_DATA
                MOV   A,#197
                LJMP  REC_CHAR
CHAR_TS_42:     CJNE  A,#4CH,CHAR_TS_43
                MOV   DPTR,#CHAR_TSL31
                LCALL PRINT_DATA
                MOV   A,#199
                LJMP  REC_CHAR
CHAR_TS_43:     CJNE  A,#52H,CHAR_TS_44
                MOV   DPTR,#CHAR_TSL32
                LCALL PRINT_DATA
                MOV   A,#167
                LJMP  REC_CHAR
CHAR_TS_44:     CJNE  A,#41H,CHAR_TS_45
                MOV   DPTR,#CHAR_TSL41
                LCALL PRINT_DATA
                MOV   A,#193
                LJMP  REC_CHAR
CHAR_TS_45:     CJNE  A,#49H,CHAR_TS_46
```

```
                        MOV   DPTR,#CHAR_TSL42
                        LCALL PRINT_DATA
                        MOV   A,#227
                        LJMP  REC_CHAR
CHAR_TS_46:             CJNE  A,#4AH,OUT_CHAR_TS
                        MOV   DPTR,#CHAR_TSL43
                        LCALL PRINT_DATA
                        MOV   A,#189
                        LJMP  REC_CHAR

OUT_CHAR_TS:            LJMP  GET_CHAR
;-------------------------------------------------------------------

PRINT_SARA_UP:          MOV   A,R7
                        CLR   C
                        SUBB  A,#06H
                        MOV   R7,A
                        DEC   R6
                        LCALL PRINT_DATA
                        INC   R6
                        RET
PRINT_SARA_DOWN:        MOV   A,R7
                        CLR   C
                        SUBB  A,#06H
                        MOV   R7,A
                        INC    R6
                        LCALL PRINT_DATA
                        DEC   R6
                        RET

;-------------------------------------------------------------------

CHAR_TB_1:              CJNE  A,#1CH,CHAR_TB_2
                        MOV   DPTR,#CHAR_TS5     ; TBA = TS5 + TH_1
                        LCALL PRINT_DATA
                        MOV   DPTR,#CHAR_TH_1
                        LCALL PRINT_SARA_DOWN
                        MOV   A,#196
                        LJMP  REC_CHAR
CHAR_TB_2:              CJNE  A,#32H,CHAR_TB_3
                        MOV   DPTR,#CHAR_TBB
                        LCALL PRINT_SARA_DOWN
                        MOV   A,#254
                        LJMP  REC_CHAR
CHAR_TB_3:              CJNE  A,#21H,CHAR_TB_4
                        MOV   DPTR,#CHAR_TBC
                        LCALL PRINT_DATA
                        MOV   A,#169
                        LJMP  REC_CHAR
CHAR_TB_4:              CJNE  A,#23H,CHAR_TB_5
                        MOV   DPTR,#CHAR_TS4     ; TBD = TS4 + TH_3
                        LCALL PRINT_DATA
                        MOV   DPTR,#CHAR_TH_3
                        LCALL PRINT_SARA_DOWN
                        MOV   A,#175
                        LJMP  REC_CHAR
CHAR_TB_5:              CJNE  A,#24H,CHAR_TB_6
                        MOV   DPTR,#CHAR_TS4     ; TBE = TS4 + TH_2
                        LCALL PRINT_DATA
                        MOV   DPTR,#CHAR_TH_2
                        LCALL PRINT_SARA_DOWN
```

```
                    MOV   A,#174
                    LJMP  REC_CHAR
CHAR_TB_6:          CJNE  A,#2BH,CHAR_TB_7
                    MOV   DPTR,#CHAR_TBF
                    LCALL PRINT_DATA
                    MOV   A,#226
                    LJMP  REC_CHAR
CHAR_TB_7:          CJNE  A,#34H,CHAR_TB_8
                    MOV   DPTR,#CHAR_TBG
                    LCALL PRINT_DATA
                    MOV   A,#172
                    LJMP  REC_CHAR
CHAR_TB_8:          CJNE  A,#33H,CHAR_TB_9
                    MOV   DPTR,#CHAR_TBH
                    LCALL PRINT_SARA_UP
                    MOV   A,#231
                    LJMP  REC_CHAR
CHAR_TB_9:          CJNE  A,#43H,CHAR_TB_10
                    MOV   DPTR,#CHAR_TBI
                    LCALL PRINT_DATA
                    MOV   A,#179
                    LJMP  REC_CHAR
CHAR_TB_10:         CJNE  A,#3BH,CHAR_TB_11
                    MOV   DPTR,#CHAR_TBJ
                    LCALL PRINT_SARA_UP
                    MOV   A,#235
                    LJMP  REC_CHAR
CHAR_TB_11:         CJNE  A,#42H,CHAR_TB_12
                    MOV   DPTR,#CHAR_TBK
                    LCALL PRINT_DATA
                    MOV   A,#201
                    LJMP  REC_CHAR
CHAR_TB_12:         CJNE  A,#4BH,CHAR_TB_13
                    MOV   DPTR,#CHAR_TBL
                    LCALL PRINT_DATA
                    MOV   A,#200
                    LJMP  REC_CHAR
CHAR_TB_13:         CJNE  A,#3AH,CHAR_TB_14
                    MOV   DPTR,#CHAR_EBL43  ; TBM = EBL43
                    LCALL PRINT_DATA
                    MOV   A,#'?'
                    LJMP  REC_CHAR
CHAR_TB_14:         CJNE  A,#31H,CHAR_TB_15
                    MOV   DPTR,#CHAR_TBN
                    LCALL PRINT_SARA_UP
                    MOV   A,#236
                    LJMP  REC_CHAR
CHAR_TB_15:         CJNE  A,#44H,CHAR_TB_16
                    MOV   DPTR,#CHAR_TBO
                    LCALL PRINT_DATA
                    MOV   A,#207
                    LJMP  REC_CHAR
CHAR_TB_16:         CJNE  A,#4DH,CHAR_TB_17
                    MOV   DPTR,#CHAR_TBP
                    LCALL PRINT_DATA
                    MOV   A,#173
                    LJMP  REC_CHAR
CHAR_TB_17:         CJNE  A,#15H,CHAR_TB_18
                    MOV   DPTR,#CHAR_TBQ    ; TBQ = TBQ + TH_1
                    LCALL PRINT_DATA
                    MOV   A,#240
```

```
                LJMP   REC_CHAR
CHAR_TB_18:     CJNE   A,#2DH,CHAR_TB_19
                MOV    DPTR,#CHAR_TBR
                LCALL  PRINT_DATA
                MOV    A,#177
                LJMP   REC_CHAR
CHAR_TB_19:     CJNE   A,#1BH,CHAR_TB_20
                MOV    DPTR,#CHAR_TBS
                LCALL  PRINT_DATA
                MOV    A,#166
                LJMP   REC_CHAR
CHAR_TB_20:     CJNE   A,#2CH,CHAR_TB_21
                MOV    DPTR,#CHAR_TBT
                LCALL  PRINT_DATA
                MOV    A,#184
                LJMP   REC_CHAR
CHAR_TB_21:     CJNE   A,#3CH,CHAR_TB_22
                MOV    DPTR,#CHAR_TBU
                LCALL  PRINT_SARA_UP
                MOV    A,#234
                LJMP   REC_CHAR
CHAR_TB_22:     CJNE   A,#2AH,CHAR_TB_23
                MOV    DPTR,#CHAR_TBV
                LCALL  PRINT_DATA
                MOV    A,#206
                LJMP   REC_CHAR
CHAR_TB_23:     CJNE   A,#1DH,CHAR_TB_24
                MOV    DPTR,#CHAR_EBL32   ; TBW = EBL32
                LCALL  PRINT_DATA
                MOV    A,#'"'
                LJMP   REC_CHAR
CHAR_TB_24:     CJNE   A,#22H,CHAR_TB_25
                MOV    DPTR,#CHAR_EB0     ; TBX = EB0
                LCALL  PRINT_DATA
                MOV    A,#')'
                LJMP   REC_CHAR
CHAR_TB_25:     CJNE   A,#35H,CHAR_TB_26
                MOV    DPTR,#CHAR_TBY
                LCALL  PRINT_SARA_UP
                MOV    A,#237
                LJMP   REC_CHAR
CHAR_TB_26:     CJNE   A,#1AH,CHAR_TB_27
                MOV    DPTR,#CHAR_EB9     ; TBZ = EB9
                LCALL  PRINT_DATA
                MOV    A,#'('
                LJMP   REC_CHAR
CHAR_TB_27:     CJNE   A,#45H,CHAR_TB_28
                MOV    DPTR,#CHAR_TB0
                LCALL  PRINT_DATA
                MOV    A,#247
                LJMP   REC_CHAR
CHAR_TB_28:     CJNE   A,#16H,CHAR_TB_29
                MOV    DPTR,#CHAR_EBL12   ; TB1 = ESL12
                LCALL  PRINT_DATA
                MOV    A,#'+'
                LJMP   REC_CHAR
CHAR_TB_29:     CJNE   A,#1EH,CHAR_TB_30
                MOV    DPTR,#CHAR_TB2
                LCALL  PRINT_DATA
                MOV    A,#241
                LJMP   REC_CHAR
```

```
CHAR_TB_30:     CJNE  A,#26H,CHAR_TB_31
                MOV   DPTR,#CHAR_TB3
                LCALL PRINT_DATA
                MOV   A,#242
                LJMP  REC_CHAR
CHAR_TB_31:     CJNE  A,#25H,CHAR_TB_32
                MOV   DPTR,#CHAR_TB4
                LCALL PRINT_DATA
                MOV   A,#243
                LJMP  REC_CHAR
CHAR_TB_32:     CJNE  A,#2EH,CHAR_TB_33
                MOV   DPTR,#CHAR_TB5
                LCALL PRINT_DATA
                MOV   A,#244
                LJMP  REC_CHAR
CHAR_TB_33:     CJNE  A,#36H,CHAR_TB_34
                MOV   DPTR,#CHAR_TB6
                LCALL PRINT_SARA_DOWN
                MOV   A,#217
                LJMP  REC_CHAR
CHAR_TB_34:     CJNE  A,#3DH,CHAR_TB_35
                MOV   DPTR,#CHAR_TB7
                LCALL PRINT_DATA
                MOV   A,#223
                LJMP  REC_CHAR
CHAR_TB_35:     CJNE  A,#3EH,CHAR_TB_36
                MOV   DPTR,#CHAR_TB8
                LCALL PRINT_DATA
                MOV   A,#245
                LJMP  REC_CHAR
CHAR_TB_36:     CJNE  A,#46H,CHAR_TB_37
                MOV   DPTR,#CHAR_TB9
                LCALL PRINT_DATA
                MOV   A,#246
                LJMP  REC_CHAR
CHAR_TB_37:     CJNE  A,#4EH,CHAR_TB_38
                MOV   DPTR,#CHAR_TBL11
                LCALL PRINT_DATA
                MOV   A,#248
                LJMP  REC_CHAR
CHAR_TB_38:     CJNE  A,#55H,CHAR_TB_39
                MOV   DPTR,#CHAR_TBL12
                LCALL PRINT_DATA
                MOV   A,#249
                LJMP  REC_CHAR
CHAR_TB_39:     CJNE  A,#5DH,CHAR_TB_40
                MOV   DPTR,#CHAR_TBL13
                LCALL PRINT_DATA
                MOV   A,#165
                LJMP  REC_CHAR
CHAR_TB_40:     CJNE  A,#54H,CHAR_TB_41
                MOV   DPTR,#CHAR_TBL21_1
                LCALL PRINT_DATA
                MOV   DPTR,#CHAR_TBL21_2
                LCALL PRINT_SARA_DOWN
                MOV   A,#176
                LJMP  REC_CHAR
CHAR_TB_41:     CJNE  A,#5BH,CHAR_TB_42
                MOV   DPTR,#CHAR_ESL41   ; TBL22 = ESL41
                LCALL PRINT_DATA
                MOV   A,#','
```

```
                        LJMP   REC_CHAR
CHAR_TB_42:             CJNE   A,#4CH,CHAR_TB_43
                        MOV    DPTR,#CHAR_TBL31
                        LCALL  PRINT_DATA
                        MOV    A,#171
                        LJMP   REC_CHAR
CHAR_TB_43:             CJNE   A,#52H,CHAR_TB_44
                        MOV    DPTR,#CHAR_ESL42  ; TBL32 = ESL42
                        LCALL  PRINT_SARA_DOWN
                        MOV    A,#218
                        LJMP   REC_CHAR
CHAR_TB_44:             CJNE   A,#41H,CHAR_TB_45
                        MOV    DPTR,#CHAR_TBL41
                        LCALL  PRINT_DATA
                        MOV    A,#178
                        LJMP   REC_CHAR
CHAR_TB_45:             CJNE   A,#49H,CHAR_TB_46
                        MOV    DPTR,#CHAR_TBL42
                        LCALL  PRINT_DATA
                        MOV    A,#204
                        LJMP   REC_CHAR
CHAR_TB_46:             CJNE   A,#4AH,OUT_CHAR_TB
                        MOV    DPTR,#CHAR_TS4    ; TBL43 = TS4 + TH_1
                        LCALL  PRINT_DATA
                        MOV    DPTR,#CHAR_TH_1
                        LCALL  PRINT_SARA_DOWN
                        MOV    A,#198
                        LJMP   REC_CHAR

OUT_CHAR_TB:            LJMP   GET_CHAR

;-------------------------------------------------------------------------

DIAL_MODEM:             MOV    A,#'A'
                        ACALL  TX_DATA
                        MOV    A,#'T'
                        ACALL  TX_DATA
                        MOV    A,#'H'
                        ACALL  TX_DATA
                        MOV    A,#'1'
                        ACALL  TX_DATA
                        ACALL  ENTER_DATA
                        LCALL  DELAY_1s
                        LCALL  DELAY_1s

                        MOV    A,#'A'
                        ACALL  TX_DATA
                        MOV    A,#'T'
                        ACALL  TX_DATA
                        MOV    A,#'D'
                        ACALL  TX_DATA
                        MOV    A,#'T'
                        ACALL  TX_DATA
                        MOV    R0,#30H
        DIAL1:          MOV    A,@R0
                        CJNE   A,#149,DIAL2
                        AJMP   DIAL3
        DIAL2:          ACALL  TX_DATA
                        INC    R0
                        AJMP   DIAL1
```

```
DIAL3:      ACALL ENTER_DATA
            RET
;-------------------------------------------------------------------

TX_DATA:    CLR   TI
            SETB  TB8
            MOV   SBUF,A
            JNB   TI,$
            RET
;-------------------------------------------------------------------

SEND_TO:    MOV   A,#128            ; Header  ' TO '
            ACALL TX_DATA
            MOV   R0,#30H
SEND_TO_1:  MOV   A,@R0
            CJNE  A,#149,SEND_TO_2
            AJMP  SEND_TO_3
SEND_TO_2:  ACALL TX_DATA
            INC   R0
            AJMP  SEND_TO_1
SEND_TO_3:  MOV   A,#149            ; Ender ' TO '
            ACALL TX_DATA
            MOV   A,#0AH
            ACALL TX_DATA
            MOV   A,#0DH
            ACALL TX_DATA
            RET

SEND_FROM:  MOV   A,#239            ; Header  ' FROM '
            ACALL TX_DATA
            MOV   R0,#30H
SEND_FROM_1: MOV  A,@R0
            CJNE  A,#149,SEND_FROM_2
            AJMP  SEND_FROM_3
SEND_FROM_2: ACALL TX_DATA
            INC   R0
            AJMP  SEND_TO_1
SEND_FROM_3: MOV  A,#149            ; Ender ' FROM '
            ACALL TX_DATA
            MOV   A,#0AH
            ACALL TX_DATA
            MOV   A,#0DH
            ACALL TX_DATA
            RET

SEND_SUBJ:  MOV   A,#250            ; Header  ' SUBJ '
            ACALL TX_DATA
            MOV   R0,#30H
SEND_SUBJ_1: MOV  A,@R0
            CJNE  A,#149,SEND_SUBJ_2
            AJMP  SEND_SUBJ_3
SEND_SUBJ_2: ACALL TX_DATA
            INC   R0
            AJMP  SEND_SUBJ_1
SEND_SUBJ_3: MOV  A,#149            ; Ender ' SUBJ '
            ACALL TX_DATA
            MOV   A,#0AH
            ACALL TX_DATA
```

```
                    MOV   A,#0DH
                    ACALL TX_DATA
                    RET

SEND_MESS:          MOV   A,#251            ; Header  ' MESS '
                    ACALL TX_DATA
                    MOV   R0,#30H
SEND_MESS_1:        MOV   A,@R0
                    CJNE  A,#149,SEND_MESS_2
                    AJMP  SEND_MESS_3
SEND_MESS_2:        ACALL TX_DATA
                    INC   R0
                    AJMP  SEND_MESS_1
SEND_MESS_3:        MOV   A,#149            ; Ender ' MESS '
                    ACALL TX_DATA
                    MOV   A,#0AH
                    ACALL TX_DATA
                    MOV   A,#0DH
                    ACALL TX_DATA
                    RET

;-------------------------------------------------------------------

ENTER_DATA:         CLR   TI
                    CLR   TB8
                    MOV   SBUF,#0AH
                    JNB   TI,$
                    CLR   TI
                    SETB  TB8
                    MOV   SBUF,#0DH
                    JNB   TI,$
                    RET

;--------------------------DATA------CHAR_ES-------------------------

CHAR_ESA:       DB      00H,40H,0A8H,0A8H,0A8H,0F0H
CHAR_ESB:       DB      00H,0FEH,90H,88H,88H,70H
CHAR_ESC:       DB      00H,70H,88H,88H,88H,40H
CHAR_ESD:       DB      00H,70H,88H,88H,90H,0FEH
CHAR_ESE:       DB      00H,70H,0A8H,0A8H,0A8H,0B0H
CHAR_ESF:       DB      00H,10H,0FCH,12H,02H,04H
CHAR_ESG:       DB      00H,18H,0A4H,0A4H,0A4H,07CH
CHAR_ESH:       DB      00H,0FEH,10H,08H,08H,0F0H
CHAR_ESI:       DB      00H,00H,88H,0FAH,80H,00H
CHAR_ESJ:       DB      00H,40H,80H,88H,7AH,00H
CHAR_ESK:       DB      00H,0FEH,20H,60H,88H,00H
CHAR_ESL:       DB      00H,00H,82H,0FEH,80H,00H
CHAR_ESM:       DB      00H,0F8H,08H,30H,08H,0F0H
CHAR_ESN:       DB      00H,0F8H,10H,08H,08H,0F0H
CHAR_ESO:       DB      00H,70H,88H,88H,88H,70H
CHAR_ESP:       DB      00H,0F8H,28H,28H,28H,10H
CHAR_ESQ:       DB      00H,10H,28H,28H,30H,0F8H
CHAR_ESR:       DB      00H,0F8H,10H,08H,08H,10H
CHAR_ESS:       DB      00H,90H,0A8H,0A8H,0A8H,40H
CHAR_EST:       DB      00H,08H,7EH,88H,80H,40H
CHAR_ESU:       DB      00H,78H,80H,80H,40H,0F8H
CHAR_ESV:       DB      00H,38H,40H,80H,40H,38H
CHAR_ESW:       DB      00H,78H,80H,70H,80H,78H
CHAR_ESX:       DB      00H,88H,50H,20H,50H,88H
CHAR_ESY:       DB      00H,18H,0A0H,0A0H,0A0H,78H
CHAR_ESZ:       DB      00H,88H,0C8H,0A8H,98H,88H
```

```
CHAR_ES1:       DB      00H,00H,84H,0FEH,80H,00H
CHAR_ES2:       DB      00H,84H,0C2H,0A2H,92H,8CH
CHAR_ES3:       DB      00H,42H,82H,8AH,96H,62H
CHAR_ES4:       DB      00H,30H,28H,24H,0FEH,20H
CHAR_ES5:       DB      00H,4EH,8AH,8AH,8AH,72H
CHAR_ES6:       DB      00H,78H,94H,92H,92H,60H
CHAR_ES7:       DB      00H,06H,02H,0E2H,12H,0EH
CHAR_ES8:       DB      00H,6CH,92H,92H,92H,6CH
CHAR_ES9:       DB      00H,0CH,92H,92H,52H,3CH
CHAR_ES0:       DB      00H,7CH,0A2H,92H,8AH,7CH
CHAR_ESL11:     DB      00H,00H,10H,10H,10H,00H
CHAR_ESL12:     DB      00H,28H,28H,28H,28H,28H
CHAR_ESL13:     DB      00H,04H,08H,10H,20H,40H
CHAR_ESL21:     DB      00H,0FEH,82H,82H,00H,00H
CHAR_ESL22:     DB      00H,00H,00H,82H,82H,0FEH
CHAR_ESL31:     DB      00H,00H,0ACH,6CH,00H,00H
CHAR_ESL32:     DB      00H,00H,0AH,06H,00H,00H
CHAR_ESL41:     DB      00H,0A0H,60H,00H,00H,00H
CHAR_ESL42:     DB      00H,0C0H,0C0H,00H,00H,00H
CHAR_ESL43:     DB      00H,40H,20H,10H,08H,04H

;---------------------------DATA------CHAR_EB-----------------------

CHAR_EBA:       DB      00H,0FCH,22H,22H,22H,0FCH
CHAR_EBB:       DB      00H,0FEH,92H,92H,92H,6CH
CHAR_EBC:       DB      00H,7CH,82H,82H,82H,44H
CHAR_EBD:       DB      00H,0FEH,82H,82H,44H,38H
CHAR_EBE:       DB      00H,0FEH,92H,92H,92H,82H
CHAR_EBF:       DB      00H,0FEH,12H,12H,02H,02H
CHAR_EBG:       DB      00H,7CH,82H,0B2H,92H,64H
CHAR_EBH:       DB      00H,0FEH,10H,10H,10H,0FEH
CHAR_EBI:       DB      00H,00H,82H,0FEH,82H,00H
CHAR_EBJ:       DB      00H,00H,40H,80H,80H,7EH
CHAR_EBK:       DB      00H,0FEH,10H,28H,44H,82H
CHAR_EBL:       DB      00H,0FEH,80H,80H,80H,80H
CHAR_EBM:       DB      00H,0FEH,04H,78H,04H,0FEH
CHAR_EBN:       DB      00H,0FEH,08H,10H,20H,0FEH
CHAR_EBO:       DB      00H,7CH,82H,82H,82H,7CH
CHAR_EBP:       DB      00H,0FEH,12H,12H,12H,0CH
CHAR_EBQ:       DB      00H,7CH,82H,0A2H,42H,0BCH
CHAR_EBR:       DB      00H,0FEH,12H,32H,52H,8CH
CHAR_EBS:       DB      00H,4CH,92H,92H,92H,64H
CHAR_EBT:       DB      00H,02H,02H,0FEH,02H,02H
CHAR_EBU:       DB      00H,7EH,80H,80H,80H,7EH
CHAR_EBV:       DB      00H,3EH,40H,80H,40H,3EH
CHAR_EBW:       DB      00H,7EH,80H,78H,80H,7EH
CHAR_EBX:       DB      00H,0C6H,28H,10H,28H,0C6H
CHAR_EBY:       DB      00H,06H,08H,0F0H,08H,06H
CHAR_EBZ:       DB      00H,0C2H,0A2H,92H,8AH,86H
CHAR_EB1:       DB      00H,00H,00H,9EH,00H,00H
CHAR_EB2:       DB      00H,64H,92H,0F2H,82H,7CH
CHAR_EB3:       DB      00H,28H,0FEH,28H,0FEH,28H
CHAR_EB4:       DB      00H,48H,54H,0FEH,54H,24H
CHAR_EB5:       DB      00H,46H,26H,10H,0C8H,0C4H
CHAR_EB6:       DB      00H,08H,04H,02H,04H,08H
CHAR_EB7:       DB      00H,6CH,92H,0AAH,44H,0A0H
CHAR_EB8:       DB      00H,00H,14H,08H,14H,00H
CHAR_EB9:       DB      00H,00H,38H,44H,82H,00H
CHAR_EB0:       DB      00H,00H,82H,44H,38H,00H
CHAR_EBL11:     DB      00H,80H,80H,80H,80H,80H
CHAR_EBL12:     DB      00H,10H,10H,7CH,10H,10H
```

```
CHAR_EBL13:       DB          00H,00H,00H,0FEH,00H,00H
CHAR_EBL21:       DB          00H,00H,10H,6CH,82H,00H
CHAR_EBL22:       DB          00H,00H,82H,6CH,10H,00H
CHAR_EBL31:       DB          00H,00H,6CH,6CH,00H,00H
CHAR_EBL32:       DB          00H,00H,0EH,00H,0EH,00H
CHAR_EBL41:       DB          00H,10H,28H,44H,82H,00H
CHAR_EBL42:       DB          00H,00H,82H,44H,28H,10H
CHAR_EBL43:       DB          00H,04H,02H,0A2H,12H,0CH

;---------------------------DATA------CHAR_TS-----------------------

CHAR_TSA:         DB          00H,74H,8CH,60H,80H,7FH
CHAR_TSB:         DB          00H,0C0H,0A0H,0A0H,0A0H,0C0H
CHAR_TSC:         DB          00H,7CH,80H,00H,7CH,80H
CHAR_TSD:         DB          00H,08H,0F4H,04H,04H,0F8H
CHAR_TSE:         DB          02H,05H,0AH,04H,04H,0F8H
CHAR_TSF:         DB          00H,78H,84H,64H,04H,0F8H
CHAR_TSG:         DB          00H,00H,00H,7CH,80H,00H
CHAR_TSH:         DB          00H,20H,0E0H,80H,40H,20H
CHAR_TSI:         DB          00H,00H,08H,94H,94H,64H
CHAR_TSJ:         DB          00H,00H,00H,00H,0E0H,00H
CHAR_TSK:         DB          00H,00H,08H,04H,04H,0F8H
CHAR_TSL:         DB          00H,48H,0A4H,24H,44H,0FBH
CHAR_TSM:         DB          00H,04H,0FCH,08H,04H,0F8H
CHAR_TSN:         DB          0C0H,0A0H,0C0H,0E0H,00H,0E0H
CHAR_TSO:         DB          00H,64H,9CH,40H,0FCH,40H
CHAR_TSP:         DB          00H,6CH,94H,90H,80H,7CH
CHAR_TSQ:         DB          00H,90H,0A0H,10H,0F0H
CHAR_TSR:         DB          00H,74H,8CH,60H,80H,7CH
CHAR_TSS:         DB          00H,04H,0FCH,08H,14H,0E4H
CHAR_TST:         DB          00H,48H,0B4H,90H,90H,48H
CHAR_TSU:         DB          00H,0C0H,0A0H,0C0H,80H,0E0H
CHAR_TSV:         DB          00H,68H,0A4H,84H,84H,78H
CHAR_TSW:         DB          00H,00H,02H,04H,7EH,80H
CHAR_TSX:         DB          00H,74H,8CH,80H,80H,7FH
CHAR_TSY:         DB          00H,00H,0C0H,80H,80H,40H
CHAR_TSZ:         DB          00H,7CH,84H,60H,80H,7CH
CHAR_TS4:         DB          00H,88H,0F4H,04H,04H,0F8H
CHAR_TS5:         DB          00H,74H,8CH,04H,04H,0F8H
CHAR_TS6:         DB          00H,00H,00H,00H,02H,0EH
CHAR_TS7:         DB          60H,50H,50H,60H,50H,60H
CHAR_TS8:         DB          00H,0F8H,24H,64H,04H,0F8H
CHAR_TS9:         DB          00H,78H,84H,68H,04H,0F8H
CHAR_TS0:         DB          00H,08H,24H,64H,84H,78H
CHAR_TSL11:       DB          00H,00H,04H,7CH,80H,7CH
CHAR_TSL12:       DB          00H,04H,7CH,88H,74H,02H
CHAR_TSL13:       DB          00H,04H,08H,74H,80H,7CH
CHAR_TSL21:       DB          00H,74H,8CH,80H,80H,7CH
CHAR_TSL22:       DB          00H,48H,0A4H,24H,44H,0F8H
CHAR_TSL31:       DB          00H,08H,04H,84H,78H,00H
CHAR_TSL32:       DB          00H,00H,30H,40H,84H,0FCH
CHAR_TSL41:       DB          00H,0C4H,7CH,40H,80H,7CH
CHAR_TSL42:       DB          00H,04H,0AH,02H,7CH,80H
CHAR_TSL43:       DB          00H,7CH,84H,60H,80H,7FH

;---------------------------DATA------CHAR_TB-----------------------

CHAR_TBB:         DB          00H,00H,00H,00H,02H,00H
CHAR_TBC:         DB          00H,28H,0E4H,44H,0F8H,40H
CHAR_TBF:         DB          00H,00H,04H,0AH,7AH,82H
CHAR_TBG:         DB          0ECH,94H,44H,0F8H,40H,0FCH
```

```
CHAR_TBH:       DB      00H,60H,90H,50H,90H,0C8H
CHAR_TBI:       DB      0F4H,88H,04H,0F8H,40H,0BCH
CHAR_TBJ:       DB      00H,00H,00H,40H,0E0H,40H
CHAR_TBK:       DB      00H,74H,8CH,0A0H,7CH,20H
CHAR_TBL:       DB      00H,0F8H,44H,64H,04H,0FBH
CHAR_TBN:       DB      00H,00H,00H,0C0H,0A0H,10H
CHAR_TBO:       DB      00H,08H,14H,10H,08H,0F4H
CHAR_TBP:       DB      0ECH,94H,04H,0B8H,0A0H,0FCH
CHAR_TBQ:       DB      00H,70H,88H,88H,88H,70H
CHAR_TBR:       DB      04H,08H,0F4H,08H,04H,0F8H
CHAR_TBS:       DB      00H,44H,0F8H,44H,80H,7CH
CHAR_TBT:       DB      00H,48H,94H,94H,94H,74H
CHAR_TBU:       DB      0E0H,0A0H,40H,20H,0C0H,70H
CHAR_TBV:       DB      00H,68H,0A4H,84H,84H,7BH
CHAR_TBY:       DB      00H,00H,00H,40H,0A0H,40H
CHAR_TB2:       DB      00H,30H,48H,28H,88H,70H
CHAR_TB3:       DB      04H,78H,80H,0B0H,90H,70H
CHAR_TB4:       DB      00H,0F0H,88H,30H,08H,0F0H
CHAR_TB5:       DB      00H,70H,88H,0E8H,0A8H,84H
CHAR_TB6:       DB      00H,00H,02H,0EH,08H,06H
CHAR_TB7:       DB      00H,7CH,54H,0FEH,54H,28H
CHAR_TB8:       DB      00H,70H,88H,0ECH,0A8H,84H
CHAR_TB9:       DB      00H,0EH,0A0H,88H,88H,70H
CHAR_TB0:       DB      0F0H,88H,30H,08H,10H,0FEH
CHAR_TBL11:     DB      00H,0F0H,88H,48H,0A8H,0E6H
CHAR_TBL12:     DB      00H,0F0H,90H,20H,0D0H,0CH
CHAR_TBL13:     DB      00H,0F8H,24H,68H,04H,0F8H
CHAR_TBL21_1:   DB      00H,08H,24H,64H,84H,7BH
CHAR_TBL21_2:   DB      00H,00H,02H,01H,02H,03H
CHAR_TBL31:     DB      00H,04H,08H,74H,88H,76H
CHAR_TBL41:     DB      7CH,88H,44H,0F8H,80H,7CH
CHAR_TBL42:     DB      00H,74H,8CH,60H,88H,77H
CHAR_TH_1:      DB      00H,00H,00H,00H,00H,0FH
CHAR_TH_2:      DB      00H,00H,04H,02H,04H,07H
CHAR_TH_3:      DB      00H,06H,06H,02H,04H,07H

SPACE:          DB      00H,00H,00H,00H,00H,00H
PAGE7_1_11:     DB      00H,00H,0C0H,40H,40H,40H
PAGE7_1_12:     DB      40H,40H,40H,40H,0C0H,0E0H
PAGE7_1_13:     DB      10H,08H,00H,00H,00H,00H
PAGE7_1_21:     DB      00H,00H,0FFH,80H,88H,0B0H
PAGE7_1_22:     DB      0E0H,0B0H,8CH,83H,80H,0FFH

                END
```

```
' โปรแกรม  Visual Basic
Public Temp1,Temp2,Temp4,Temp5,Temp9,Temp10,TempX As Variant
Public TempNum As Integer

Private Sub Form_Load()
MSComm1.Settings = "9600,n,8,1"
MSComm1.CommPort = 1
MSComm1.PortOpen = True
MSComm1.InputMode = comInputModeText
MSComm1.RThreshold = 1
MSComm1.InputLen = 0
MSComm1.Output = "ATS0=1" & Chr$(10) & Chr$(13)
TempNum = 1
End Sub

Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
Case comEvReceive
Dim buffRv As String
Dim source,Startb,Stopb1,Stopb2,Stopb3,Stopb4,Stopb5 As String
Dim num As Integer
Dim foundPos1,foundPos2,foundPos3,foundPos4,foundPos5 As Integer
bufferRv = MSComm1.Input
source = source & bufferRv
Startb = Startb & source
Stopb1 = Chr(128)
Stopb2 = Chr(239)
Stopb3 = Chr(250)
Stopb4 = Chr(251)
Stopb5 = Chr(149)
foundPos1 = InStr(Startb, Stopb1)
foundPos2 = InStr(Startb, Stopb2)
foundPos3 = InStr(Startb, Stopb3)
foundPos4 = InStr(Startb, Stopb4)
foundPos5 = InStr(Startb, Stopb5)

If (foundPos1 <> 0) Or (foundPos2 <> 0) Or (foundPos3 <> 0) Or
   (foundPos4 <> 0) Then

     If TempNum = 5 Then
     TempNum = 1
     End If

     If TempNum = 1 Then
     Email.Recordset.MoveLast
     Email.Recordset.AddNew
     End If

     If (foundPos1 <> 0) Then
     TempNum = TempNum + 1
     End If
     If (foundPos2 <> 0) Then
     TempNum = TempNum + 1
     End If
     If (foundPos3 <> 0) Then
     TempNum = TempNum + 1
     End If
     If (foundPos4 <> 0) Then
     TempNum = TempNum + 1
     End If
  End If
```

```
1:
    If (foundPos1 <> 0) Then
        If foundPos5 = 0 Then
        Temp2 = Right(bufferRv, Len(bufferRv) - 1)
        Temp1 = Stopb1
        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp2
        End If

        If foundPos5 <> 0 Then
        Temp2 = Mid(bufferRv, 2, Len(bufferRv) - 4)
        Temp10 = ""
        Data_Email
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp2 & vbCrLf
        End If

    ElseIf (Temp1 = Stopb1) Then
        If foundPos5 <> 0 Then
        Temp10 = Left(bufferRv, Len(bufferRv) - 3)
        Data_Email
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp10 & vbCrLf

    Else
        Temp4 = Temp4 & bufferRv

        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp4
        End If
    End If

2:
    If (foundPos2 <> 0) Then
        If foundPos5 = 0 Then
        Temp2 = Right(bufferRv, Len(bufferRv) - 1)
        Temp1 = Stopb2
        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp2
        End If


        If foundPos5 <> 0 Then
        Temp2 = Mid(bufferRv, 2, Len(bufferRv) - 4)
        Temp10 = ""
        Data_Sender
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp2 & vbCrLf
        End If

    ElseIf (Temp1 = Stopb2) Then
        If foundPos5 <> 0 Then
        Temp10 = Left(bufferRv, Len(bufferRv) - 3)
        Data_Sender
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp10 & vbCrLf

    Else
        Temp4 = Temp4 & bufferRv

        txtCommu2 = txtCommu2 & bufferRv
```

```
        txtcommu3 = txtcommu3 & Temp4
        End If
    End If

3:
    If (foundPos3 <> 0) Then
        If foundPos5 = 0 Then
        Temp2 = Right(bufferRv, Len(bufferRv) - 1)
        Temp1 = Stopb3
        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp2
        End If



        If foundPos5 <> 0 Then
        Temp2 = Mid(bufferRv, 2, Len(bufferRv) - 4)
        Temp10 = ""
        Data_Subject
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp2 & vbCrLf
        End If

    ElseIf (Temp1 = Stopb3) Then
        If foundPos5 <> 0 Then
        Temp10 = Left(bufferRv, Len(bufferRv) - 3)
        Data_Subject
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp10 & vbCrLf

    Else
        Temp4 = Temp4 & bufferRv

        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp4
        End If
    End If

4:
    If (foundPos4 <> 0) Then
        If foundPos5 = 0 Then
        Temp2 = Right(bufferRv, Len(bufferRv) - 1)
        Temp1 = Stopb4
        txtCommu2 = txtCommu2 & bufferRv
        txtcommu3 = txtcommu3 & Temp2
        End If



        If foundPos5 <> 0 Then
        Temp2 = Mid(bufferRv, 2, Len(bufferRv) - 4)
        Temp10 = ""
        Data_Message
        txtCommu2 = txtCommu2 & bufferRv & vbCrLf
        txtcommu3 = txtcommu3 & Temp2 & vbCrLf
        End If

    ElseIf (Temp1 = Stopb4) Then
        If foundPos5 <> 0 Then
        Temp10 = Left(bufferRv, Len(bufferRv) - 3)
        Data_Message
```

```
    txtCommu2 = txtCommu2 & bufferRv & vbCrLf
    txtcommu3 = txtcommu3 & Temp10 & vbCrLf

Else
    Temp4 = Temp4 & bufferRv
    txtCommu2 = txtCommu2 & bufferRv
    txtcommu3 = txtcommu3 & Temp4
    End If
End If
End Select
End Sub

Private Sub Data_Sender()
TempX = Temp2 & Temp4 & Temp10
With Email
.Recordset.Fields("Sender") = TempX
End With
Temp1 = ""
Temp2 = ""
Temp4 = ""
Temp10 = ""
End Sub

Private Sub Data_Subject()
TempX = Temp2 & Temp4 & Temp10
With Email
.Recordset.Fields("Subject") = TempX
End With
Temp1 = ""
Temp2 = ""
Temp4 = ""
Temp10 = ""
End Sub

Private Sub Data_Email()
TempX = Temp2 & Temp4 & Temp10
With Email
.Recordset.Fields("Email") = TempX
End With
Temp1 = ""
Temp2 = ""
Temp4 = ""
Temp10 = ""
End Sub

Private Sub Data_Message()
TempX = Temp2 & Temp4 & Temp10
With Email
.Recordset.Fields("Message") = TempX
End With
Temp1 = ""
Temp2 = ""
Temp4 = ""
Temp10 = ""
Email.Recordset.Update
End Sub

Private Sub Clear_Click()
    txtCommu2.Text = ""
    txtcommu3.Text = ""
End Sub
```

```
Private Sub Exit_Click()
End
End Sub

Private Sub Reset_Modem_Click()
MSComm1.Output = Chr$(13) & "ATZ" & Chr$(13)
MSComm1.Output = "ATH0" & Chr$(13)
MSComm1.DTREnable = False
MSComm1.RTSEnable = False
MSComm1.PortOpen = False
End Sub

Private Sub Start_Click()
MSComm1.CommPort = 1
MSComm1.DTREnable = True
MSComm1.RTSEnable = True
MSComm1.PortOpen = True
MSComm1.Settings = "9600,n,8,1"
MSComm1.InputMode = comInputModeText
MSComm1.InputLen = 0
MSComm1.Output = "ATS0=2" & Chr$(10) & Chr$(13)
End Sub

Private Sub AddCmnd_Click()
Email.Recordset.AddNew
Email_TO.SetFocus
End Sub

Private Sub UpdateCmnd_Click()
Email.Recordset.Update
End Sub

Private Sub DeleteCmnd_Click()
Email.Recordset.Delete
Email.Recordset.MoveLast
End Sub

Private Sub RefreshCmnd_Click()
Email.Refresh
End Sub
```

```
// โปรแกรมนับเวลาถอยหลัง 3600 วินาที

// โปรแกรมทำงานบนเครื่องเซิฟเวอร์

<html>

<head>
      <TITLE>Countdown</Title>

      <Script Language="VBScript">

         intLeft = 3600

         Sub leavePage()

           IF 0 = intLeft THEN

            Window.Navigate "http://lab-2/countdown.asp","Windows1"

            Window.Open "http://lab-2/mailform.asp","Windows2"

           Else

            intLeft = IntLeft - 1

            document.all.countdown.innerText = intLeft

            setTimeout "leavePage()",1000

           end IF

         End Sub

      </Script>

</head>

<body onload="setTimeout 'leavePage()', 1000">

      <H2> Test </H2>

      <SPAN ID="countdown">

      <Script Language="VbScript">

      document.write intLeft

      </Script>

</body>

</html>
```

```
// โปรแกรมบนเครื่องเซิฟเวอร์ ทำหน้าที่ดึงข้อมูลใหม่ๆ จากฐานข้อมูลขึ้นมาเพื่อส่งไปยังเมลล์เซิฟเวอร์
<% dmail = request.QueryString("dmail") %>
<html>
<head>
<title>My Project : Mailphone</title>
</head>
<%
Set ObjDB=Server.CreateObject("ADODB.Connection")
ObjDB.Open "project"
Set RS = Server.CreateObject("ADODB.RecordSet")
rs.open "Select * from mail_list  ",ObjDB,1,3
IF not rs.EOF then
numRecond = rs.recordcount
id = rs("ID")
For n1 = 1 To 100
      IF n1 = id then n2 = n1 end if
Next
NumLast = NumRecond + n2 -1
rs.close
For NUM = n2 To NumLast
rs.open "Select * from mail_list  Where ID = " & NUM , ObjDB,1,3
id = rs("ID")
 mail = rs("Email")
 send = rs("Sender")
 subj = rs("Subject")
 mess = rs("Message")

%>
<style type="text/css">
<!-- textarea {font-family : MS Sans Serif,Arial; font-size : 10 pt;
color : #000000} -->
A {
      Color: #6600FF;Text-Decoration: none
}


A:hover {
      Color: #CC33FF;Text-Decoration: underline
}
BODY  {
            font-family:  MS Sans Serif,CordiaUPC;
            }
TBODY       {
            font-family:  MS Sans Serif,CordiaUPC;
            }

</style>

<body>

<center>
<br>
<form action="sendmail.asp" method="post">
<table border=0 cellpadding=0 cellspacing=0 width="80%"
bgcolor=#CCFFCC>
      <tr bgcolor=#CC0033><td colspan=2 align=center><font
color=white size=+1>
            <b>Program : send mail</td>
      </tr>
<tr><td> </td><td> </td></tr>
```

```
<tr>
      <td align=right><font size=-1><strong>To Email : 
</strong></font></td>
      <td><input type="text" name="mail" maxlength=35 size="35"
value="<%=mail%>"></td>
</tr>
<tr><td> </td><td> </td></tr>
<tr>
      <td align=right><font size=-1><strong>Sender :  </strong>
</font></td>
      <td><input type="text" name="send" maxlength=25 size="25"
value="<%=send%>"></td>
</tr>
<tr>
      <td align=right><font size=-1><strong>Subject :  </strong>
</font></td>
      <td><input type="text" name="subj" maxlength=35 size="35"
value="<%=subj%>"></td>
</tr>
<tr>
    <td align="right"><font size=-1><strong>Message </strong>
</font></td>
    <td><input type="text" name="mess" maxlength=70 size="70"
value="<%=mess%>"></td>
 </tr>
  <tr>
      <td> </td><td><br><input type="submit" value=" OK! send ">
        <input type="reset" value=" Reset "><br><br></td>
</tr>
</table>
</form>
<%
rs.close
mail=""
send=""
mess=""
subj=""
Next
end if
%>
</body>
</html>
```

```
// โปรแกรมบนเครื่องเมลล์เซิฟเวอร์ ทำหน้าที่ส่งอีเมลล์ไปยังเครื่องปลายทาง
// แต่โปรแกรมจำลองการทำงานบนเครื่องเซิฟเวอร์ด้วยโปรแกรม PWS
<html>
<head>
<title>Program Send Mail.</title>
</head>
<body bgcolor="#FFFFFF"><center>
<%
subj = request.form("subj")
mail = request.form("mail")
send = request.form("send")
mess = request.form("mess")
Set ObjDB=Server.CreateObject("ADODB.Connection")
ObjDB.Open "project"
Set RS = Server.CreateObject("ADODB.RecordSet")
rs.open "Select * from mail_list",ObjDB,1,3
rs.close
%>
<table border=0 cellpadding="0" cellspacing="0">
<tr>
<td align="center"><font size=+2><br><%=ErrMsg%><hr></font></td>
<tr>
</tr>
<tr><td align="center"><font size=+2>To : <%=mail%><hr></font></td>
</tr>
<tr><td align="center"><font size=+2>Subj : <%=subj%><hr></font></td>
</tr>
<tr><td align="center"><font size=+2><%=mess%><hr></font></td></tr>
<tr><td align="center"><font size=+2>From : <%=send%><hr></font></td>
</tr>
<%
```

```
rs.open "Select * from mail_list " & id,ObjDB,1,3
rs.delete
ObjDB.close %>
</table>
</body>
</html>
```

ภาคผนวก ข

ตารางแสดงเหตุการณ์ใน MSComm

MSComm Control Table

ตาราง CommEvent เกี่ยวกับการเกิดสถานะเมื่อเกิดการผิดพลาดในการสื่อสาร

ชื่อ Property	คำอธิบาย
ComEventBreak	การได้รับสัญญาณ
ComEventCDTO	เมื่อเกิดไทม์เอาต์ ขณะที่กำลังคอยสัญญาณ(Carrier Detect)
ComEventCTSTO	เมื่อเกิดไทม์เอาต์ ขณะที่กำลังคอยสัญญาณCTS(Carrier To Send)
ComEventDSRTO	เมื่อเกิดไทม์เอาต์ ขณะที่กำลังคอยสัญญาณDSR(Data Set Ready)
ComEventFrame	การที่เกิดความผิดพลาดทางเฟรม เป็นลักษณะที่ไม่พบบิตจบตามที่ควรจะเป็น
ComEventOverrun	การที่เกิดความผิดพลาดโอเวอร์รัน เป็นลักษณะที่รับข้อมูลไม่ทันในการประมวลผล
ComEventRxOver	บัฟเฟอร์ที่รับข้อมูลเกิดโอเวอร์โฟวล์ ก็คือรับตัวอักษรหลังจากการรับ EOF Char
ComEventRxParity	การที่เกิดความผิดพลาดทางพาริตี้ เป็นลักษณะที่ตัวอักษรที่รับได้มีพาริตี้ไม่ถูกต้อง
ComEventTxFull	ตัวบัฟเฟอร์ที่ส่งข้อมูลเต็ม
ComEventDCB	การที่เกิดความผิดพลาดขึ้นโดยไม่ได้คาดถึง

ตารางCommEvent เกี่ยวกับเกิดสถานะการเมื่อมีการสื่อสาร

ชื่อ Property	คำอธิบาย
ComEvCD	CD(Carrirt Detect)เมื่อเปลี่ยนซึ่งคือสายของสัญญาณReceive Line Signal Detect (RLSD)
ComEvCTS	RCTS(Carrier To Send)เมื่อมีการเปลี่ยนสถานะเกิดขึ้น
ComEvDSR	DSR(Data Set Ready)เมื่อมีการเปลี่ยนสถานะเกิดขึ้น
ComEvRing	เมื่อตรวจจับสัญญาณ Ring Indicator ได้
ComEvReceive	เมื่อได้รับข้อมูลเก็บลงใน Input Buffer
ComEvSend	เมื่อส่งข้อมูลออกจาก Output Buffer
ComEvEof	เมื่อพบอักขระ EOF(End Of File)

ตาราง Handshake Property

ชื่อ Property	คำอธิบาย
ComNone	ไม่ใช้ให้ตรวจสอบแฮนเช็ก
ComXonZXoff	ให้มีการตรวจสอบแฮนเช็ก ในแบบ Xon/Xoff
ComRTS	ให้มีการตรวจสอบแฮนเช็ก ผ่านขา RTS และ CTS
ComRTSXOnXoff	กำหนดให้มีการตรวจทั้ง 2 แบบคือ RTS-CTS และ Xon/Xoff

ตาราง Input Mode Property

ชื่อ Properties	คำอธิบาย
ComInputMode Text	คุณสมบัติในการรับข้อมูลมาเป็นแบบข้อความ ปกติจะเป็นค่านี้อยู่แล้ว
ComInputMode Binary	คุณสมบัติในการรับข้อมูลมาเป็นแบบไบนารี่หรือเลขฐานสองนั้นเอง

ตาราง MSComm Control Property

ชื่อ Property	คำอธิบาย
Break	ในการที่เรากำหนดหรือเคลียร์สัญญาณเบรก
CDHolding	ตรวจสอบสัญญาณ Carrier Detect(CD)ดูว่ายังคงมีสถานะอยู่หรือเปล่า
CDTimeout	การกำหนดค่าหรือว่าให้ค่าของเวลา(หน่วย mmSec)ที่รอสัญญาณ Carrier Detect
CommEvent	จะให้ผลของการเกิด Event ของ Communication
CommID	จะให้ผลของการเฮดเดิลของ Communication ที่เปิดใช้อยู่
CommPort	การกำหนดหรือว่าอ้างอิงของหมายเลขคอมพอร์ตที่เปิดใช้อยู่ เช่น Com1=1,Com2=2
CTSHolding	เป็นการตรวจสอบสัญญาณของ Clear To Send ว่ายังคงมีสถานะอยู่หรือเปล่า
CTSTimeout	การกำหนดค่าหรือว่าให้ค่าของเวลา(หน่วย mmSec)ที่รอสัญญาณ Data Set Ready
DSRHolding	เป็นการตรวจสอบสัญญาณของ Data Set Ready ว่ายังคงมีสถานะอยู่หรือเปล่า
DSRTimeout	การกำหนดค่าหรือว่าให้ค่าของเวลา(หน่วย mmSec)ที่รอสัญญาณ Clear To Send
DTREnable	ให้อีนาเบิลสายของสัญญาณ Data Terminal Ready(DTR)
Handshaking	กำหนดการแฮนด์เช็คทางฮาร์ดแวร์ เพื่อที่คอยตรวจสอบการรับส่งข้อมูล
InBufferCount	ให้ค่าของจำนวนข้อมูลที่อยู่ภายในบัฟเฟอร์รับข้อมูล
InBufferSize	กำหนดหรือว่าให้ค่าของขนาดในบัฟเฟอร์รับข้อมูล
Input	เป็นการให้ค่าหรือว่าเคลื่อนย้ายข้อมูลจากบัฟเฟอร์รับข้อมูล
InputLen	การกำหนดหรือว่าให้ของจำนวนข้อมูลที่นำมาจากบัฟเฟอร์รับข้อมูล
Interval	เป็นการกำหนดอัตราความเร็วของการใช้งานในโหมดโพลลิ่ง
NullDiscard	เป็นการกำหนดให้มีการรับ Null Character เก็บลงในบัฟเฟอร์รับข้อมูล
OutBufferCount	เป็นจำนวนข้อมูลที่คอยอยู่ในบัฟเฟอร์ส่งข้อมูล
OutBufferSize	การกำหนดหรือว่าให้ค่าขนาดของบัฟเฟอร์ส่งข้อมูล
Output	เป็นการส่งข้อมูลให้กับบัฟเฟอร์ส่งข้อมูลเพื่อทำการส่งข้อมูลออก
ParityReplace	เป็นการกำหนดให้ส่งอักขระที่กำหนดนี้แทนหากเกิดการผิดพลาดในข้อมูล
PortOpen	เป็นการกำหนดหรือว่าให้ค่าของสถานะพอร์ตว่าเปิดหรือปิดอยู่
Rthreshold	การกำหนดหรือว่าให้ค่าของจำนวนข้อมูลที่เก็บลงในบัฟเฟอร์รับข้อมูลก่อนการเกิด CommEvent ในการรับข้อมูล
RTSEnable	ให้อีนาเบิลสัญญาณ Request To Send(RTS)
Settings	เป็นการกำหนดอัตราบอด พาริตี้ ข้อมูล บิตหยุด
Sthreshold	ผลของข้อมูลที่เก็บลงในของบัฟเฟอร์ส่งข้อมูลก่อนเกิด CommEvent ในการส่งข้อมูล

ตาราง รายละเอียดที่บ่งชี้ถึงความผิดพลาดในการใช้ MS Comm Control

ชื่อ Property	ค่าตัวเลข	คำอธิบาย
ComInvalidPrppertyValue	380	ตั้งค่าไม่ถูกต้อง
ComSetNotSupported	383	กำหนดค่าที่ตั้งไว้สามารถอ่านได้อย่างเดียว เขียนหรือเปลี่ยนข้อมูลไม่ได้
ComGetNotSupported	394	กำหนดค่าที่รับไว้สามารถอ่านได้อย่างเดียว เขียนหรือเปลี่ยนข้อมูลไม่ได้
ComPortOpen	8000	จะอ่านค่าไม่ได้ในขณะที่ Port นั้นยังถูกเปิดใช้อยู่
ComPortOpen	8001	ค่าของเวลาที่หาออกมาได้ต้องมีค่ามากกว่าศูนย์
ComPortOpen	8002	กำหนดหมายเลข Port ไม่ถูกต้อง
ComPortOpen	8003	ผลลัพธ์ของข้อมูลจะเกิดในขณะที่มีการทำงาน
ComPortOpen	8004	Port นั้นจะสามารถอ่านค่าได้ในขณะที่มีการทำงานเท่านั้น
ComPortAlreadyOpen	8005	Port ได้ถูกเปิดไว้เรียบร้อยแล้ว
ComPortAlreadyOpen	8006	อุปกรณ์เกิดความผิดพลาดหรือไม่สามารถรองรับค่าได้
ComPortAlreadyOpen	8007	อุปกรณ์ไม่ยอมรับค่าที่ Baud Rate ถูกตั้งเอาไว้
ComPortAlreadyOpen	8008	ขนาดของข้อมูลผิดพลาด
ComPortAlreadyOpen	8009	ค่าของตัวแปรที่แสดงอยู่ผิดพลาด
ComPortAlreadyOpen	8010	อุปกรณ์ภายนอก(Hardware)ยังไม่พร้อมที่จะทำงาน
ComPortAlreadyOpen	8011	ฟังก์ชั่นไม่สามารถกำหนดแถวข้อมูลได้
ComNoOpen	8012	Com Port ยังไม่พร้อมที่จะถูกเปิดใช้งาน
ComNoOpen	8013	Com Portพร้อมที่จะถูกเปิดใช้งาน
ComNoOpen	8014	Com Portไม่สามารถทำงานได้
ComSetCommStateFailed	8015	ไม่สามารถตั้งค่าสถานะของ Port ได้
ComSetCommStateFailed	8016	ไม่สามารถ Set Port ตามเหตุการณ์ที่กำหนดให้ได้
ComPortNotOpen	8018	จะสามารถหาผลลัพธ์ของข้อมูลได้ก็ต่อเมื่อ Port มีการทำงานแล้วเท่านั้น
ComPortNotOpen	8019	Port ไม่มีที่ว่างมีข้อมูลเต็มใน Port
ComReadError	8020	เกิดความผิดพลาดขึ้นขณะที่อ่าน
ComDCBError	8021	เกิดความผิดพลาดภายในต้องไปแก้ไขที่ตัวควบคุม Port

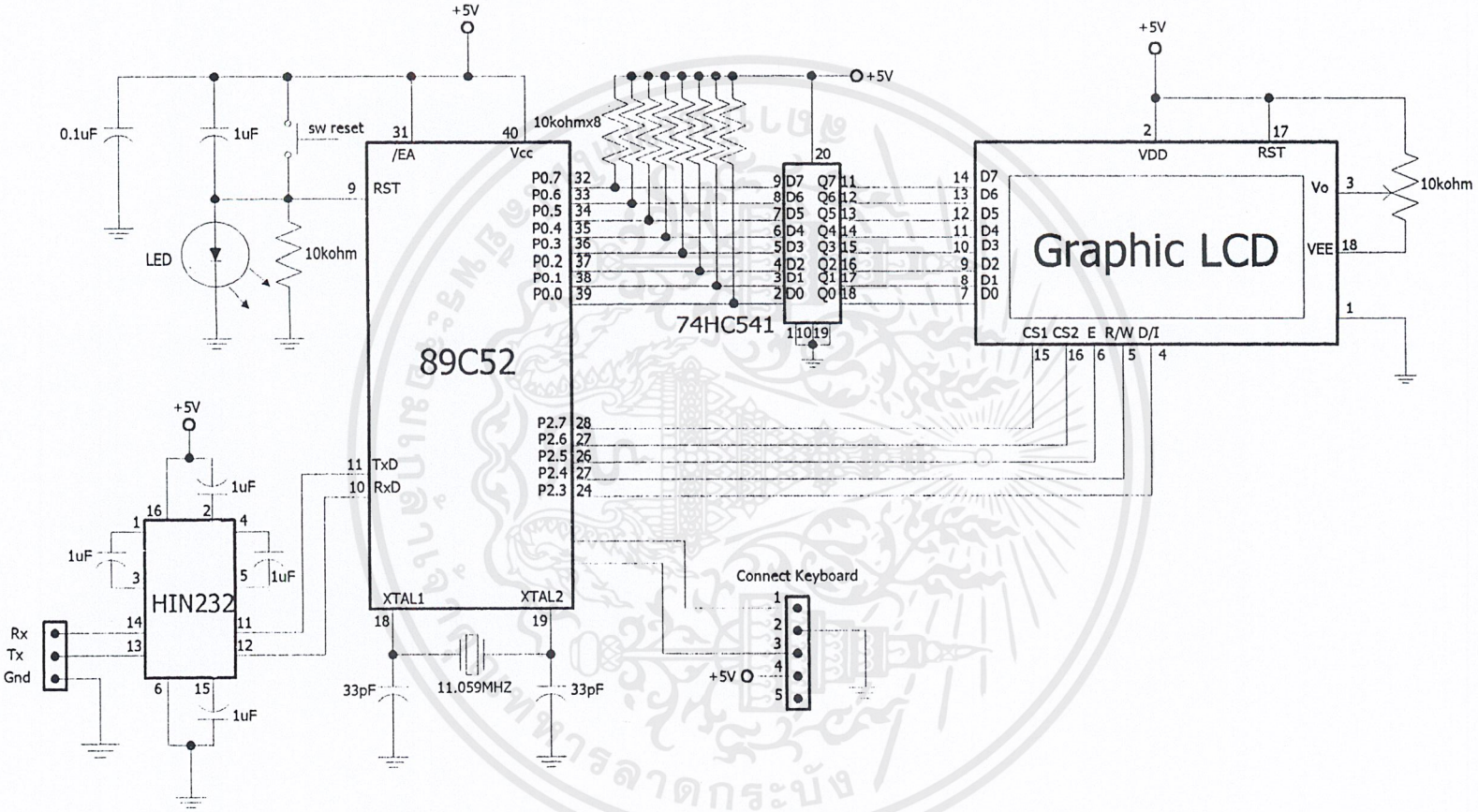
ภาคผนวก ค
ตารางแสดงรหัสแอสกี

ASCII Code	Keyboard	ASCII Code	Keyboard	ASCII Code	Keyboard	ASCII Code	Keyboard
8	ปุ่มลบ	60	<	91	[	124	|
9	Tab	61	=	92	\	125	}
10	Shift	62	>	93	]	126	~
13	Enter	63	?	94	^	128	
27	Esc	64	@	97	a	129	
33	!	65	A	98	b	130	
35	#	66	B	99	c	131	
36	$	67	C	100	d	132	
37	%	68	D	101	e	133	...
38	&	69	E	102	f	134	
39		70	F	103	g	135	
40	(	71	G	104	h	136	
41	)	72	H	105	i	137	
42	*	73	I	106	j	138	
43	+	74	J	107	k	139	
44	,	75	K	108	l	140	
45	-	76	L	109	m	141	
46	.	77	M	110	n	142	
47	/	78	N	111	o	143	
48	0	79	O	112	p	144	
49	1	80	P	113	q	145	‘
50	2	81	Q	114	r	146	’
51	3	82	R	115	s	147	“
52	4	83	S	116	t	148	”
53	5	84	T	117	u	149	
54	6	85	U	118	v	152	
55	7	86	V	119	w	153	
56	8	87	W	120	x	154	
57	9	88	X	121	y	155	
58	:	89	Y	122	z	156	
59	;	90	Z	123	{	157	

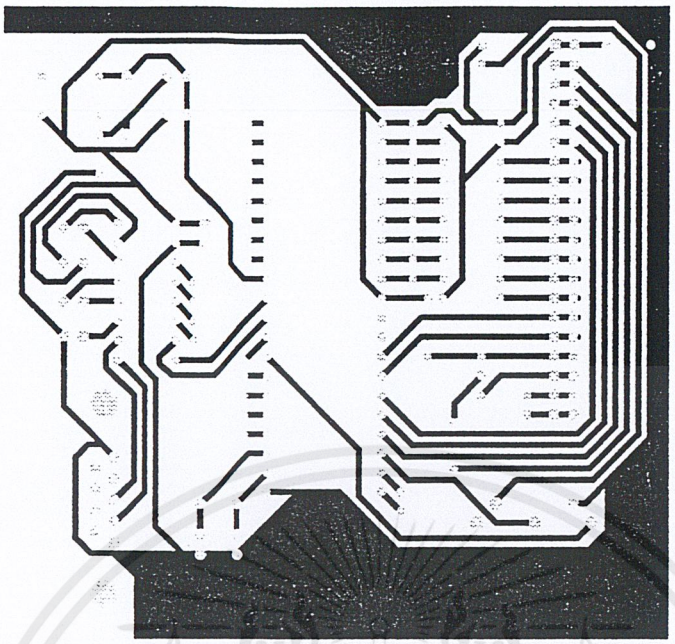
ASCII Code	Keyboard	ASCII Code	Keyboard	ASCII Code	Keyboard	ASCII Code	Keyboard
158		189	ฝ	219			
159		190	พ	220			
161	ก	191	ฟ	221			
162	ข	192	ภ	226	โ		
163	ฃ	193	ม	227	ใ		
164	ค	194	ย	228	ไ		
165	ฅ	195	ร	229			
166	ฆ	196	ฤ	230	ๆ		
167	ง	197	ล	231			
168	จ	198	ฦ	232			
169	ฉ	199	ว	233			
170	ช	200	ศ	234			
171	ซ	201	ษ	235			
172	ฌ	202	ส	236			
173	ญ	203	ห	240	0		
174	ฎ	204	ฬ	241	1		
175	ฏ	205	อ	242	2		
176	ฐ	206	ฮ	243	3		
177	ฑ	207	ฯ	244	4		
178	ฒ	208	ะ	245	5		
179	ณ	209		246	6		
180	ด	210	า	247	7		
181	ต	211		248	8		
182	ถ	212		249	9		
183	ท	213					
184	ธ	214					
185	น	215					
186	บ	216					
187	ป	217					
188	ผ	218					

ภาคผนวก ง

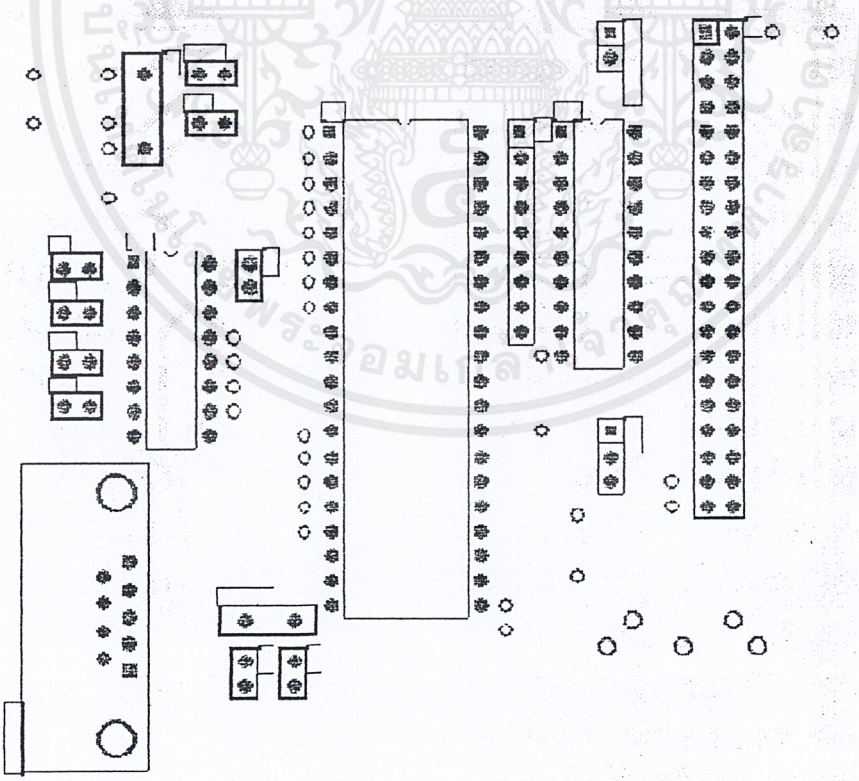
วงจรรวม ลายวงจรพิมพ์ และรูปภาพของโครงงาน



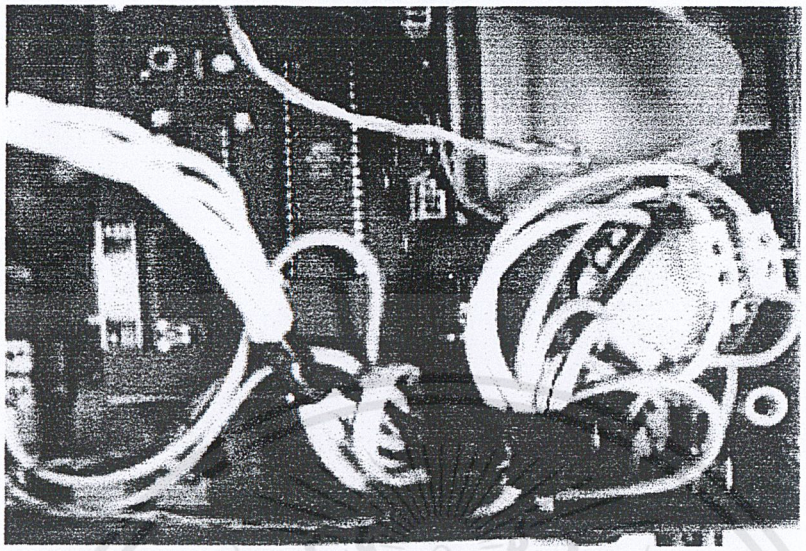
รูป แสดงวงจรรวมที่ ไคลเอ็นท์ (Client)



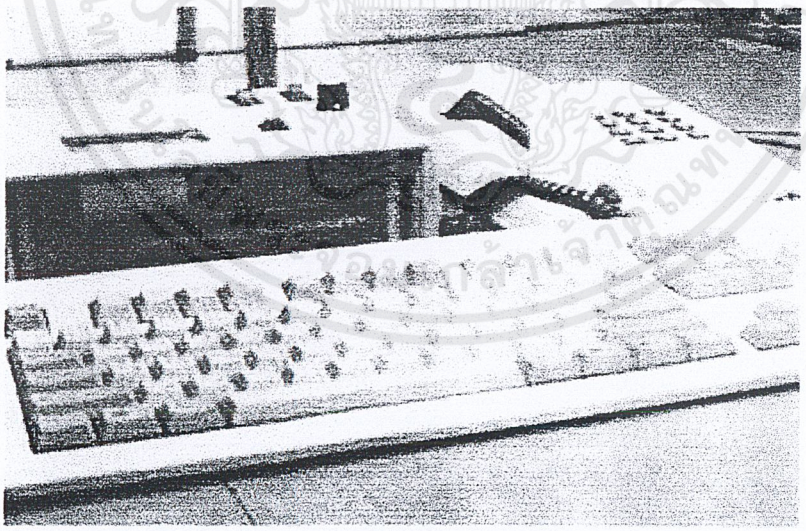
รูปแสดงลายวงจรพิมพ์ของเครื่องไคลเอ็นต์



รูปแสดงการจัดเรียงอุปกรณ์ต่างๆ ลงในลายวงจรพิมพ์



รูปแสดงอุปกรณ์ต่างๆ ภายในเครื่องไคลเอ็นต์



รูปแสดงอุปกรณ์ต่างๆ ที่เชื่อมต่อกับเครื่องไคลเอ็นต์

ภาคผนวก จ
Datasheet ของไอซีที่ใช้ในโครงงาน

8-Bit Microcontroller with 4 Kbytes Flash

Features

- Compatible with MCS-51™ Products
- 4 Kbytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

Description

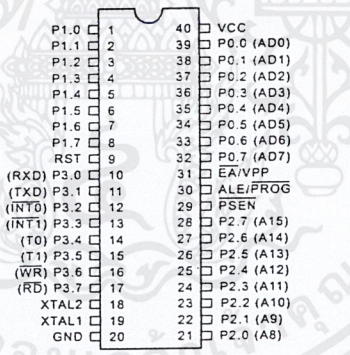
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4 Kbytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C51 provides the following standard features: 4 Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is

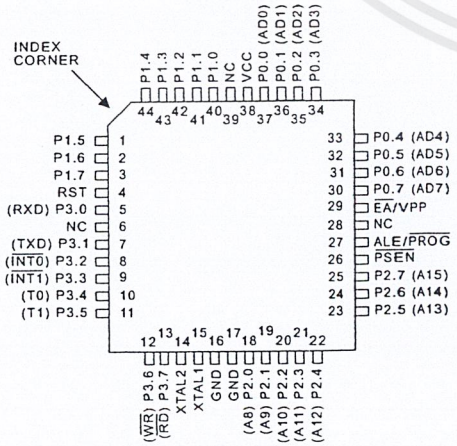
(continued)

Pin Configurations

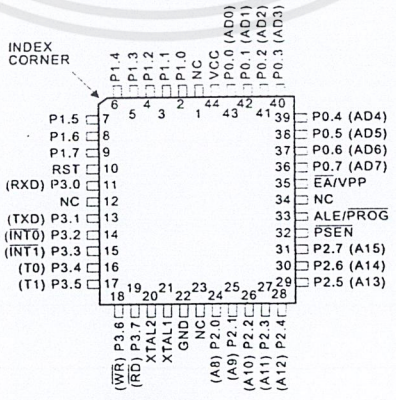
PDIP/Cerdip



PQFP/TQFP

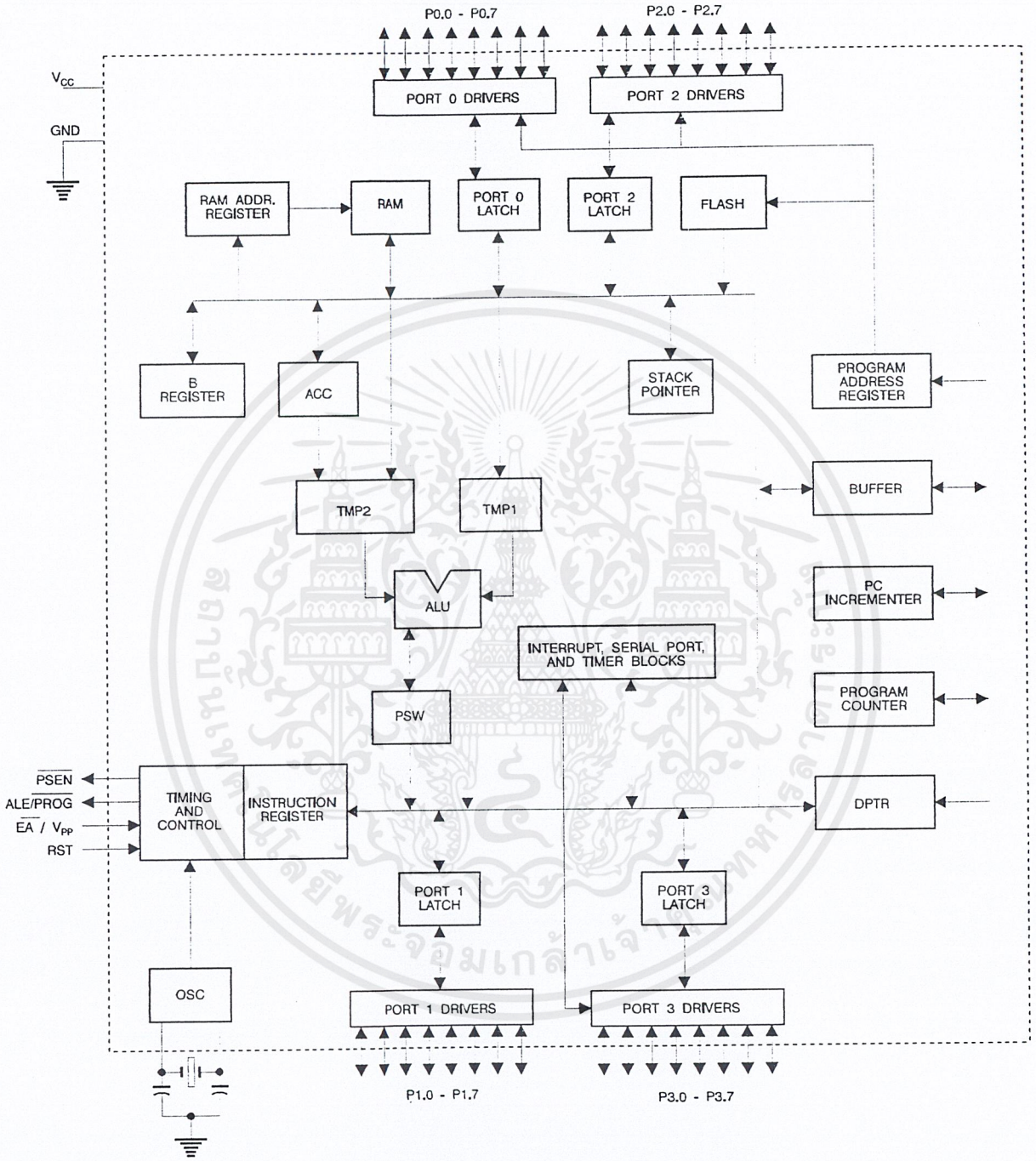


PLCC/LCC



0265E

Block Diagram



HIN231, HIN232, HIN236, HIN237, HIN238, HIN239, HIN240, HIN241

Data Sheet August 2001 File Number 3138.10

+5V Powered RS-232 Transmitters/Receivers

The HIN231-HIN241 family of RS-232 transmitters/receivers interface circuits meet all EIA RS-232E and V.28 specifications, and are particularly suited for those applications where ±12V is not available. They require a single +5V power supply (except HIN231 and HIN239) and feature onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply. The family of devices offer a wide variety of RS-232 transmitter/receiver combinations to accommodate various applications (see Selection Table).

The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to ±30V, and have a 3kΩ to 7kΩ input impedance. The receivers also feature hysteresis to greatly improve noise rejection.

Features

- Meets All RS-232E and V.28 Specifications
- Requires Only Single +5V Power Supply
  - (+5V and +12V - HIN231 and HIN239)
- High Data Rate. 120kbps
- HIN233 and HIN235 Require No External Capacitors
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- Low Power Shutdown Function
- Three-State TTL/CMOS Receiver Outputs
- Multiple Drivers
  - ±10V Output Swing for 5V Input
  - 300Ω Power-Off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/µs Maximum Slew Rate
- Multiple Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection

Applications

- Any System Requiring RS-232 Communication Ports
  - Computer - Portable, Mainframe, Laptop
  - Peripheral - Printers and Terminals
  - Instrumentation
  - Modems

Selection Table

PART NUMBER	POWER SUPPLY VOLTAGE	NUMBER OF RS-232 DRIVERS	NUMBER OF RS-232 RECEIVERS	EXTERNAL COMPONENTS	LOW POWER SHUTDOWN/TTL THREE-STATE	NUMBER OF LEADS
HIN231	+5V and +7.5V to 13.2V	2	2	2 Capacitors	NO/NO	16
HIN232	+5V	2	2	4 Capacitors	NO/NO	16
HIN236	+5V	4	3	4 Capacitors	YES/YES	24
HIN237	+5V	5	3	4 Capacitors	NO/NO	24
HIN238	+5V	4	4	4 Capacitors	NO/NO	24
HIN239	+5V and +7.5V to 13.2V	3	5	2 Capacitors	NO/YES	24
HIN240	+5V	5	5	4 Capacitors	YES/YES	44
HIN241	+5V	4	5	4 Capacitors	YES/YES	28

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
HIN231IB	-40 to 85	16 Ld SOIC	M16.3
HIN232CP	0 to 70	16 Ld PDIP	E16.3
HIN232CB	0 to 70	16 Ld SOIC	M16.3
HIN232IP	-40 to 85	16 Ld PDIP	E16.3
HIN232IB	-40 to 85	16 Ld SOIC	M16.3
HIN236CP	0 to 70	24 Ld PDIP	E24.3
HIN236CB	0 to 70	24 Ld SOIC	M24.3
HIN236IB	-40 to 85	24 Ld SOIC	M24.3
HIN237CB	0 to 70	24 Ld SOIC	M24.3
HIN238CP	0 to 70	24 Ld PDIP	E24.3

Ordering Information (Continued)

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
HIN238CB	0 to 70	24 Ld SOIC	M24.3
HIN238IB	-40 to 85	24 Ld SOIC	M24.3
HIN239CB	0 to 70	24 Ld SOIC	M24.3
HIN239CP	0 to 70	24 Ld PDIP	E24.3
HIN239IB	-40 to 85	24 Ld SOIC	M24.3
HIN240CN	0 to 70	44 Ld MQFP	Q44.10X10
HIN241CB	0 to 70	28 Ld SOIC	M28.3
HIN241IB	-40 to 85	28 Ld SOIC	M28.3
HIN241CA	0 to 70	28 Ld SSOP	M28.209

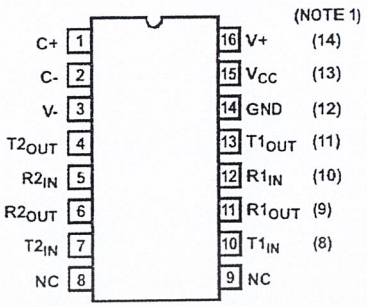
NOTE: Many of the surface mount devices are available on tape and reel; add -T to suffix.

Pin Descriptions

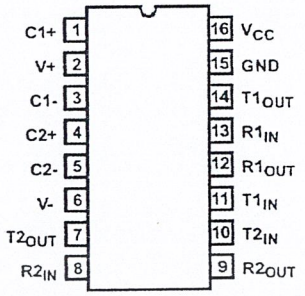
PIN	FUNCTION
V_{CC}	Power Supply Input 5V ±10%.
V+	Internally generated positive supply (+10V nominal). HIN231 and HIN239 require +7.5V to +13.2V.
V-	Internally generated negative supply (-10V nominal).
GND	Ground lead. Connect to 0V.
C1+	External capacitor (+ terminal) is connected to this lead.
C1-	External capacitor (- terminal) is connected to this lead.
C2+	External capacitor (+ terminal) is connected to this lead.
C2-	External capacitor (- terminal) is connected to this lead.
T_{IN}	Transmitter Inputs. These leads accept TTL/CMOS levels. An internal 400kΩ pull-up resistor to V_{CC} is connected to each lead.
T_{OUT}	Transmitter Outputs. These are RS-232 levels (nominally ±10V).
R_{IN}	Receiver Inputs. These inputs accept RS-232 input levels. An internal 5kΩ pull-down resistor to GND is connected to each input.
R_{OUT}	Receiver Outputs. These are TTL/CMOS levels.
$\overline{EN}$	Enable input. This is an active low input which enables the receiver outputs. With $\overline{EN}$ = 5V, the outputs are placed in a high impedance state.
SHUTDOWN	Shutdown Input. With SHUTDOWN = 5V, the charge pump is disabled, the receiver outputs are in a high impedance state and the transmitters are shut off.
NC	No Connect. No connections are made to these leads.

Pinouts

HIN231 (SOIC)
TOP VIEW

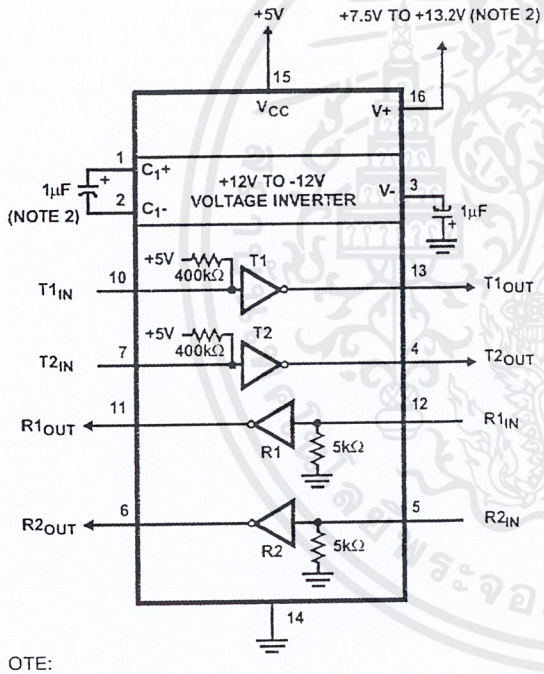


HIN232 (PDIP, SOIC)
TOP VIEW



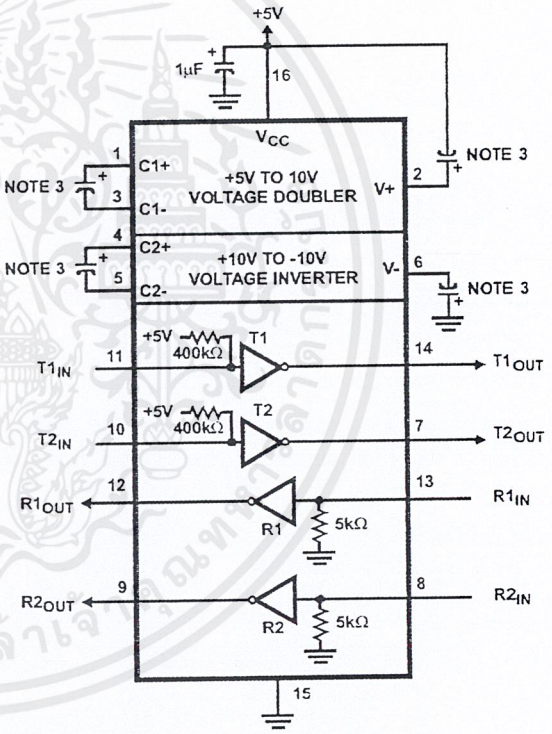
NOTE:

1. Pin numbers in parentheses are for PDIP Package.



OTE:

2. For V+ > 11V, use $C_1 \leq 0.1\mu F$.



NOTE:

3. Either 0.1µF or 1µF capacitors may be used. The V+ capacitor may be terminated to V_{CC} or to GND.

MC54/74HC541A

Octal 3-State Non-Inverting Buffer/Line Driver/ Line Receiver

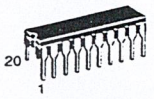
High–Performance Silicon–Gate CMOS

The MC54/74HC541A is identical in pinout to the LS541. The device inputs are compatible with Standard CMOS outputs. External pullup resistors make them compatible with LSTTL outputs.

The HC541A is an octal non–inverting buffer/line driver/line receiver designed to be used with 3–state memory address drivers, clock drivers, and other bus–oriented systems. This device features inputs and outputs on opposite sides of the package and two ANDed active–low output enables.

The HC541A is similar in function to the HC540A, which has inverting outputs.

- Output Drive Capability: 15 LSTTL Loads
- Outputs Directly Interface to CMOS, NMOS and TTL
- Operating Voltage Range: 2 to 6V
- Low Input Current: 1μA
- High Noise Immunity Characteristic of CMOS Devices
- In Compliance With the JEDEC Standard No. 7A Requirements
- Chip Complexity: 134 FETs or 33.5 Equivalent Gates



J SUFFIX
CERAMIC PACKAGE
CASE 732–03

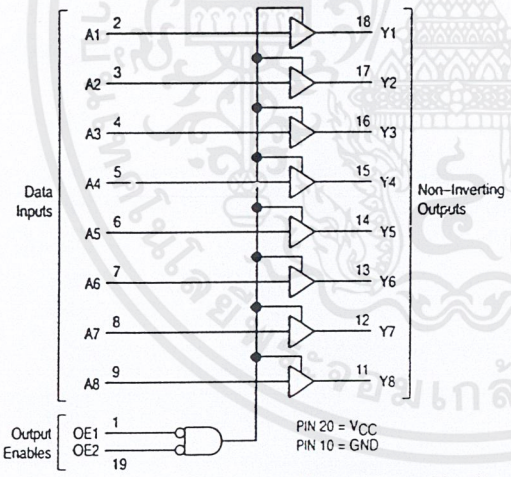
N SUFFIX
PLASTIC PACKAGE
CASE 738–03

DW SUFFIX
SOIC PACKAGE
CASE 751D–04

ORDERING INFORMATION

MC54HCXXXAJ	Ceramic
MC74HCXXXAN	Plastic
MC74HCXXXADW	SOIC

LOGIC DIAGRAM

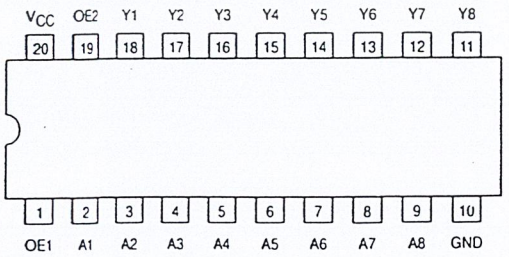


FUNCTION TABLE

Inputs			Output Y
OE1	OE2	A	
L	L	L	L
L	L	H	H
H	X	X	Z
X	H	X	Z

Z = High Impedance
X = Don't Care

Pinout: 20–Lead Packages (Top View)



MOTOROLA

MAXIMUM RATINGS*

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	– 0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	– 0.5 to V_{CC} + 0.5	V
V_{out}	DC Output Voltage (Referenced to GND)	– 0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 35	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 75	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP†	750	mW
	SOIC Package†	500	
T_{stg}	Storage Temperature Range	– 65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		°C
	Plastic DIP or SOIC Package	260	
	Ceramic DIP)	300	

This device contains protection circuitry to guard against damage due to high static voltages or electric fields. However, precautions must be taken to avoid applications of any voltage higher than maximum rated voltages to this high–impedance circuit. For proper operation, V_{in} and V_{out} should be constrained to the range GND ≤ (V_{in} or V_{out}) ≤ V_{CC}.

Unused inputs must always be tied to an appropriate logic voltage level (e.g., either GND or V_{CC}). Unused outputs must be left open.

* Maximum Ratings are those values beyond which damage to the device may occur. Functional operation should be restricted to the Recommended Operating Conditions.

†Derating — Plastic DIP: – 10 mW/°C from 65° to 125°C
Ceramic DIP: – 10 mW/°C from 100° to 125°C
SOIC Package: – 7 mW/°C from 65° to 125°C

For high frequency or heavy load considerations, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)		2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)		0	V_{CC}	V
T_A	Operating Temperature Range, All Package Types		– 55	+ 125	°C
t_r, t_f	Input Rise/Fall Time (Figure 1)	V_{CC} = 2.0 V	0	1000	ns
		V_{CC} = 4.5 V	0	500	
		V_{CC} = 6.0 V	0	400	

DC CHARACTERISTICS (Voltages Referenced to GND)

Symbol	Parameter	Condition	V_{CC} V	Guaranteed Limit –55 to 25°C	Guaranteed Limit ≤85°C	Guaranteed Limit ≤125°C	Unit
V_{IH}	Minimum High–Level Input Voltage	V_{out} = 0.1V, $\|I_{out}\| \le 20\mu A$	2.0	1.50	1.50	1.50	V
			3.0	2.10	2.10	2.10	
			4.5	3.15	3.15	3.15	
			6.0	4.20	4.20	4.20	
V_{IL}	Maximum Low–Level Input Voltage	V_{out} = V_{CC} – 0.1V, $\|I_{out}\| \le 20\mu A$	2.0	0.50	0.50	0.50	V
			3.0	0.90	0.90	0.90	
			4.5	1.35	1.35	1.35	
			6.0	1.80	1.80	1.80	
V_{OH}	Minimum High–Level Output Voltage	V_{in} = V_{IL}, $\|I_{out}\| \le 20\mu A$	2.0	1.9	1.9	1.9	V
			4.5	4.4	4.4	4.4	
			6.0	5.9	5.9	5.9	
		V_{in} = V_{IL}, $\|I_{out}\| \le 3.6mA$	3.0	2.48	2.34	2.20	
		$\|I_{out}\| \le 6.0mA$	4.5	3.98	3.84	3.70	
		$\|I_{out}\| \le 7.8mA$	6.0	5.48	5.34	5.20	
V_{OL}	Maximum Low–Level Output Voltage	V_{in} = V_{IH}, $\|I_{out}\| \le 20\mu A$	2.0	0.1	0.1	0.1	V
			4.5	0.1	0.1	0.1	
			6.0	0.1	0.1	0.1	
		V_{in} = V_{IH}, $\|I_{out}\| \le 3.6mA$	3.0	0.26	0.33	0.40	
		$\|I_{out}\| \le 6.0mA$	4.5	0.26	0.33	0.40	
		$\|I_{out}\| \le 7.8mA$	6.0	0.26	0.33	0.40	

DC CHARACTERISTICS (Voltages Referenced to GND)

Symbol	Parameter	Condition	V_{CC} V	Guaranteed Limit			Unit
				–55 to 25°C	≤85°C	≤125°C	
I_{in}	Maximum Input Leakage Current	V_{in} = V_{CC} or GND	6.0	±0.1	±1.0	±1.0	μA
I_{OZ}	Maximum Three–State Leakage Current	Output in High Impedance State V_{in} = V_{IL} or V_{IH} V_{out} = V_{CC} or GND	6.0	±0.5	±5.0	±10.0	μA
I_{CC}	Maximum Quiescent Supply Current (per Package)	V_{in} = V_{CC} or GND I_{out} = 0μA	6.0	4	40	160	μA

NOTE: Information on typical parametric values can be found in Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

AC CHARACTERISTICS (C_L = 50 pF, Input t_r = t_f = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit			Unit
			–55 to 25°C	≤85°C	≤125°C	
t_{PLH}, t_{PHL}	Maximum Propagation Delay, Input A to Output Y (Figures 1 and 3)	2.0 3.0 4.5 6.0	80 30 18 15	100 40 23 20	120 55 28 25	ns
t_{PLZ}, t_{PHZ}	Maximum Propagation Delay, Output Enable to Output Y (Figures 2 and 4)	2.0 3.0 4.5 6.0	110 45 25 21	140 60 31 26	165 75 38 31	ns
t_{PZL}, t_{PZH}	Maximum Propagation Delay, Output Enable to Output Y (Figures 2 and 4)	2.0 3.0 4.5 6.0	110 45 25 21	140 60 31 26	165 75 38 31	ns
t_{TLH}, t_{THL}	Maximum Output Transition Time, Any Output (Figures 1 and 3)	2.0 3.0 4.5 6.0	60 22 12 10	75 28 15 13	90 34 18 15	ns
C_{in}	Maximum Input Capacitance		10	10	10	pF
C_{out}	Maximum Three–State Output Capacitance (Output in High Impedance State)		15	15	15	pF

NOTE: For propagation delays with loads other than 50 pF, and information on typical parametric values, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

		Typical @ 25°C, V_{CC} = 5.0 V, V_{EE} = 0 V	
C_{PD}	Power Dissipation Capacitance (Per Buffer)*	35	pF

* Used to determine the no–load dynamic power consumption: $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$. For load considerations, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

SWITCHING WAVEFORMS

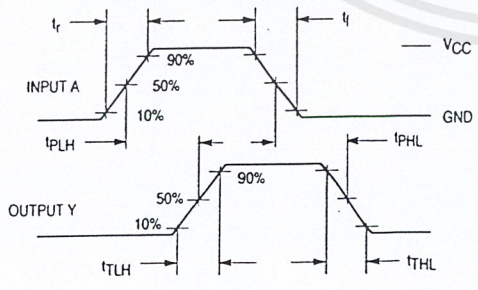


Figure 1.

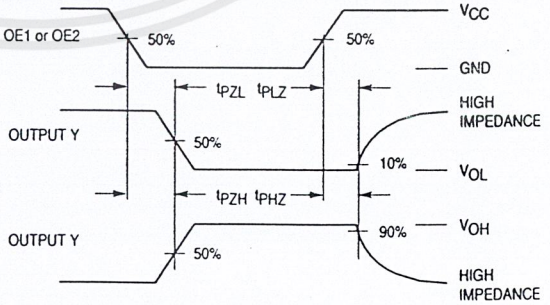
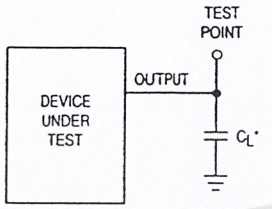


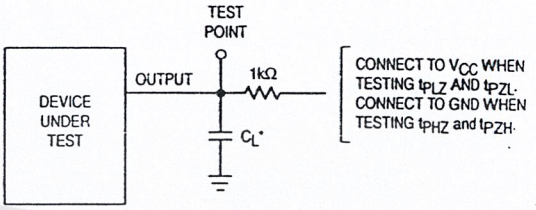
Figure 2.

TEST CIRCUITS



*Includes all probe and jig capacitance

Figure 3.



*Includes all probe and jig capacitance

Figure 4.

PIN DESCRIPTIONS

INPUTS

A1, A2, A3, A4, A5, A6, A7, A8 (PINS 2, 3, 4, 5, 6, 7, 8, 9) — Data input pins. Data on these pins appear in non–inverted form on the corresponding Y outputs, when the outputs are enabled.

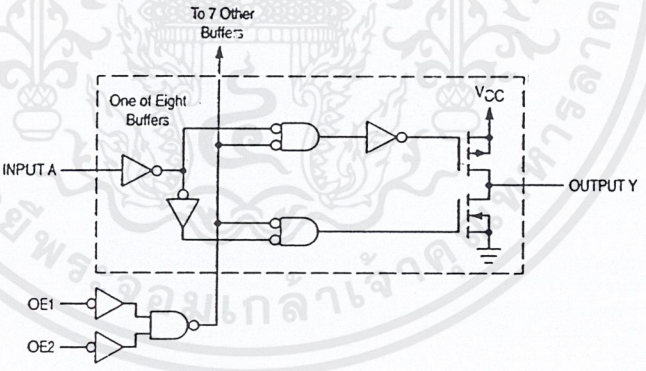
CONTROLS

OE1, OE2 (PINS 1, 19) — Output enables (active–low). When a low voltage is applied to both of these pins, the outputs are enabled and the device functions as an non–inverting buffer. When a high voltage is applied to either input, the outputs assume the high impedance state.

OUTPUTS

Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8 (PINS 18, 17, 16, 15, 14, 13, 12, 11) — Device outputs. Depending upon the state of the output enable pins, these outputs are either non–inverting outputs or high–impedance outputs.

LOGIC DETAIL



หนังสืออ้างอิง

1.Andrew S.Tanenbaum, " **Computer Networks** ",Prentice-Hall,Inc.,1996.

2.ชัยวัฒน์ ลิ้มพรจิตรวิไล,วรพจน์ กรแก้ววัฒนกุล," **เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51** ", กรุงเทพฯ.

3.สมยศ จุณณะปิยะ, " **ไมโครคอนโทรลเลอร์ MCS-51** " ,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,2543.

4.จิรศักดิ์ เหลืองอุไร, " **การสื่อสารอนุกรมบน PC** " ,บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน),กรุงเทพฯ.

5.สุรศักดิ์ สงวนพงษ์, " **สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี** " ,บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน), กรุงเทพฯ.

6.พิพัฒน์ หิรัณย์วนิชชากร, " **ระบบการสื่อสารข้อมูล และเครือข่ายคอมพิวเตอร์** " ,บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน),กรุงเทพฯ.

7.กอบเกียรติ สระอุบล, " **สร้างเวบเพจเหนือคำบรรยาย กับ ASP** " ,สำนักพิมพ์ บี อี แอนด์ ซี,กรุงเทพฯ.

8.สิทธิศักดิ์ คล่องดี, " **การพัฒนา Web Application ด้วย ASP** ",บริษัทสำนักพิมพ์ ข้าวฟ่าง จำกัด, กรุงเทพฯ.

9.สุปราณี ธีรไกรศรี, " **HTML4 Visual Guide** " ,บริษัท โปรวิชั่น จำกัด,กรุงเทพฯ.

10.กิตติ ภักดีวัฒนกุล, " **ASP ฉบับฐานข้อมูล** ",บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด,กรุงเทพฯ.

11.ชัชวาล ศุภเกษม, " **Visual Basic 6.0 ภาคปฏิบัติ** ",บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน),กรุงเทพฯ.

12.กิตติ ภักดีวัฒนกุล, " **Visual Basic 6 ฉบับโปรแกรมเมอร์** ",บริษัท เคทีพี คอมพ์ แอนด์ คอน ซัลท์ จำกัด กรุงเทพฯ.