การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

โดย

นาย อนิรุทธ์ บัญชา

นาย ฐนวัฒน์ คงสงค์

นาย สุรเชษฎ์ ขวัญทอง

ปริญญานิพนธ์นี้สำหรับปริญญาวิศวกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

โดย

นาย อนิรุทธ์ บัญชา รหัส 42015248

นาย ฐนวัฒน์ คงสงค์ รหัส 42515722

นาย สุรเชษฎ์ ขวัญทอง รหัส 42515751

อาจารย์ที่ปรึกษา

ศ.ดร.วัลลภ สุระกำพลธร

ผศ. สุชาติ คูณทวีเทพ

รายงานนี้สำหรับปริญญาวิศวกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

ผู้จัดทำ

นาย อนิรุทธ์ บัญชา รหัส 42015248

นาย ฐนวัฒน์ คงสงค์ รหัส 42515722

นาย สุรเชษฎ์ ขวัญทอง รหัส 42515751

โครงงานนี้ได้รับการตรวจสอบพร้อมที่ทำการสอบได้

_____อาจารย์ที่ปรึกษา

(ศ.ดร.วัลลภ สุระกำพลธร)

_____อาจารย์ที่ปรึกษา

(ผศ. สุชาติ คูณทวีเทพ)

13 มีนาคม 2545

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย**

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

ผู้จัดทำ

นาย อนิรุทธ์ บัญชา รหัส 42015248

นาย ฐนวัฒน์ คงสงค์ รหัส 42515722

นาย สุรเชษฏ์ ขวัญทอง รหัส 42515751

_____อาจารย์ที่ปรึกษา

(ศ.ดร.วัลลภ สุระกำพลธร)

_____อาจารย์ที่ปรึกษา

(ผศ. สุชาติ คูณทวีเทพ)

13 มีนาคม 2545

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

ผู้จัดทำ

นาย อนิรุทธ์	บัญชา	รหัส 42015248
นาย ฐนวัฒน์	คงสงค์	รหัส 42515722
นาย สุรเชษฎ์	ขวัญทอง	รหัส 42515751

_____อาจารย์ที่ปรึกษา

(ศ.ดร.วัลลภ สุระกำพลธร)

_____อาจารย์ที่ปรึกษา

(ผศ. สุชาติ คูณทวีเทพ)

13 มีนาคม 2545

การประยุกต์บัตรแม่เหล็กสำหรับงานระบบความปลอดภัย

ACCESS CONTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

ผู้จัดทำ

นาย อนิรุทธ์ บัญชา รหัส 42015248

นาย ฐนวัฒน์ คงสงค์ รหัส 42515722

นาย สุรเชษฏ์ ขวัญทอง รหัส 42515751

โครงงานนี้ได้รับการตรวจสอบพร้อมที่ทำการสอบได้

_____ อาจารย์ที่ปรึกษา

(ศ.ดร.วัลลภ สุระกำพลธร)

_____ อาจารย์ที่ปรึกษา

(ผศ. สุชาติ คูณทวีเทพ)

13 มีนาคม 2545

การประยุกต์บัตรแม่เหล็กสำหรับงานระบบรักษาความปลอดภัย

นาย อนิรุทธ์ บัญชา
นาย ฐนวัฒน์ คงสงค์
นาย สุรเชษฎ์ ขวัญทอง
ศ.ดร.วัลลภ สุระกำพลธร ที่ปรึกษา
ผศ. สุชาติ คูณทวีเทพ ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

โครงงานนี้เป็นโครงงานที่เกี่ยวข้องกับการใช้เครื่องอ่านบัตรแม่เหล็กมาประยุกต์ใช้งานในระบบรักษาความปลอดภัยในการเข้าภายในอาคาร โดยการออกแบบประกอบด้วย 2 ส่วน คือ ส่วนประมวลผลและส่วนวงจรควบคุม ซึ่งส่วนประมวลผลได้ใช้ ไมโครคอนโทรลเลอร์ และไมโครคอมพิวเตอร์ ส่วนวงจรได้แก่ วงจรอ่านข้อมูลจากบัตรแถบแม่เหล็ก วงจรควบคุมอุปกรณ์โซลินอยด์ วงจร keyboard สำหรับกดรหัสของบัตร โดยมีหลักการทำงานคือเมื่อนำบัตรแถบแม่เหล็ก เช่น บัตรเอทีเอ็ม (ATM : Automatic Teller Machine) บัตรประชาชน บัตรนักศึกษา มารูดที่เครื่องอ่านบัตร ระบบการทำงานของเครื่องจะทำการตรวจสอบรหัสบัตร โดยระบบจะถูกควบคุมโดยไมโครคอนโทรลเลอร์ และไมโครคอมพิวเตอร์ โดยใช้โปรแกรม Visual basic Version 6 เป็นตัวแสดงผลและควบคุมการทำงานของระบบ เพื่อบันทึกข้อมูลของเจ้าของบัตรและบันทึกภาพผู้เข้ามาในตัวอาคาร

ACCRSS COTROL FOR SECURITY SYSTEM BY USING MAGNETIC CARD

MR.Anirutt Bancha

MR.Tanawat Kongsong

MR.Surachet Kwantong

Assoc.Prof.Dr.Wanlop Surakumpontorn ADVISER

Asst.Suchart Khuntaweetep ADVISER

Academic year 2001

ABSTRACT

This project is applying magnetic card for security system which to consist of two parts. One is the hardware by using microcontroller , microcomputer ,camera , magnetic card reader and press code keyboard. The second parts is software to control the security system by using Visual Basic Version 6 and assembly program. In this system, it can capture picture and use with there magnetic card readers to access any I/O.

สารบัญ

หน้า

หน้า

สารบัญรูป

หน้า

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงการ

ในปัจจุบัน ความก้าวหน้าทางเทคโนโลยีการสื่อสาร คอมพิวเตอร์ การเก็บรวบรวมข้อมูลต่าง ๆ โดยเฉพาะการเก็บข้อมูลและการเชื่อมต่อข้อมูลต่าง ๆ ให้สามารถค้นหาได้สะดวกรวดเร็ว เพื่อความสะดวกและประหยัด เวลารวมทั้งบุคลากร โดยการใช้บัตรแม่เหล็ก (Magnetic Card) แทนการใช้สมุดบัญชีธนาคาร การประยุกต์ใช้บัตรแม่เหล็กไปเป็นบัตรเครดิตเพื่อใช้แทนเงินสด รวมทั้งมีการนำเอารหัสแท่ง (Bar Code) มาใช้ในระบบการจำหน่ายสินค้า อันจะช่วยอำนวยความสะดวกให้แก่ผู้ซื้อกละผู้ขาย แต่บัตรแม่เหล็กมีความเป็นมาตรฐาน และเป็นที่นิยมใช้กันอย่างแพร่หลาย อันจะเห็นได้จากการใช้บัตรเอทีเอ็ม (ATM : Automatic Teller Machine) ในการฝาก-ถอนเงินของธนาคารต่าง ๆ ซึ่งสามารถนำเอาข้อมูลที่บันทึกอยู่บนบัตรมาใช้ในการอ้างอิงข้อมูลของเจ้าบัตรได้

1.2 วัตถุประสงค์ของโครงการ

1. ศึกษาการอ่านข้อมูลของเครื่องอ่านบัตรแม่เหล็ก
2. ศึกษาการรับข้อมูลจากบัตรแม่เหล็กเพื่อเชื่อมต่อกับโปรแกรมควบคุมและบันทึก รายชื่อและเวลาการเข้าอาคาร
3. เขียนโปรแกรมควบคุมและบันทึกรายชื่อและเวลาการเข้าอาคาร
4. เก็บรวบรวมข้อมูลจากการทำโครงการลงในแผ่นบันทึก (diskette)

1.3 ขอบเขตของโครงการ

จะใช้บัตรแม่เหล็กร่วมกับโปรแกรมควบคุมและบันทึกรายชื่อและเวลาการเข้าอาคาร โดยใช้โปรแกรม Visual Basic Version 6 เป็นตัวเขียนโปรแกรม

มีหัวข้อในการทำงานดังรายการต่อไปนี้

1. งานเก็บข้อมูล รายชื่อผู้ที่ต้องการเข้าอาคาร
   1.1 ชื่อของผู้ที่ต้องเข้าอาคาร
   1.2 รหัสของผู้ที่ต้องเข้าอาคาร
2. งานควบคุมการเปิด-ปิดประตูเข้าอาคาร
3. งานบันทึกรายชื่อและเวลาของผู้ที่เข้าอาคาร

1.4 วิธีการดำเนินงาน

1. ศึกษาการอ่านข้อมูลเครื่องอ่านบัตรแม่เหล็ก
2. ศึกษาการเชื่อมโยง (Interface) ข้อมูลระหว่างเครื่องอ่านบัตรแม่เหล็กกับคอมพิวเตอร์
3. เขียนโปรแกรมควบคุมและบันทึก รายชื่อ และเวลาการเข้าอาคาร

1.5 ประโยชน์ที่ได้รับ

1. ช่วยลดขั้นตอนการลงเวลาการทำงาน
2. เพิ่มประสิทธิภาพในการรักษาความปลอดภัย

บทที่ 2
ทฤษฎี

2.1 คำนำ

ในการนำบัตรแม่เหล็กมาใช้งานเป็นบัตรตรวจสอบการเข้าออก โดยใช้ข้อมูลบนบัตรแม่เหล็กเป็นข้อมูลประจำตัวผู้ถือบัตร สำหรับการพัฒนาระบบในขั้นต้นนี้ ได้นำบัตรแม่เหล็กของธนาคารพาณิชย์ต่าง ๆ มาแระยุกต์ใช้งานกับระบบ ซึ่งบัตรแม่เหล็กดังกล่าวมีรูปแบบของข้อมูลที่บันทึกอยู่บนแถบแม่เหล็กและคุณลักษณะต่าง ๆ ของบัตรแม่เหล็ก เป็นไปตามมาตรฐานสากล (International Organization for Standardization : ISO) โดยรายละเอียดของบัตรแม่เหล็กจะนำเสนอในหัวข้อ 2.2 สำหรับรูปแบบการอ่านข้อมูลจากแถบแม่เหล็ก ได้นำเสนอการอ่านข้อมูลจากบัตรแม่เหล็กเพื่อเชื่อมต่อกับคอมพิวเตอร์ ซึ่งรายละเอียดจะนำเสนอในหัวข้อ 2.4

2.2 คุณสมบัติของบัตรแม่เหล็ก ATM [1]

แถบแม่เหล็กของบัตรแม่เหล็กธนาคารพาณิชย์ต่าง ๆ ทั่วโลก จะมีแทร็ก (Track) บันทึกข้อมูลจำนวนสามแทร็ก โดยแทร็กที่สองของแถบแม่เหล็กบันทึกตัวเลขที่อ้างอิงกับหมายเลขบัญชีของผู้ถือบัตร เพื่อใช้เป็นข้อมูลดิบ (Raw Date) สำหรับการคำนวณรหัส PIN (Personal Identification Number) ของบัตรแม่เหล็กใบนั้น ๆ โดยข้อมูลในแทร็กที่สองนี้ จะเป็นตัวเลขชุดเดียวกันกับตัวเลขที่ปั้มนูนอยู่บนบัตรแม่เหล็ก อาทิ บัตรแม่เหล็กของธนาคารกรุงเทพ จำกัด บัตรเครดิตและวีซ่า (Visa) เป็นต้น จากข้อมูลดังกล่าว บนบัตรจะบันทึกข้อมูลและรายละเอียดต่าง ๆ ของบัตรไว้ในรูปของเส้นแรงแม่เหล็ก (Flux) โดยแถบแม่เหล็กบนบัตรซึ่งเรียกว่าแทร็กนั้นจะถูกแบ่งออกเป็น 3 ส่วน ซึ่งแต่ละส่วนจะใช้เก็บข้อมูลซึ่งมีความหนาแน่น และลักษณะข้อมูลที่มีความแตกต่างกันแสดงดังนี้

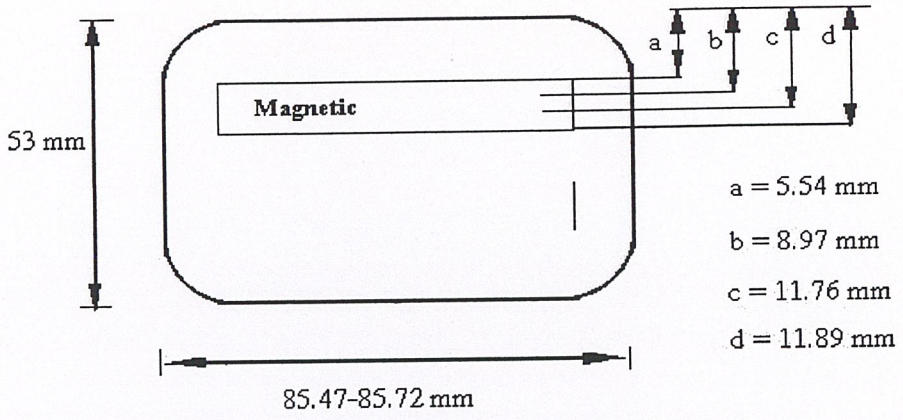
ตารางที่ 2.1 แสดงลักษณะของส่วนที่ใช้เก็บข้อมูลที่มีความแตกต่างกัน

แถบแม่เหล็ก	ความหนาแน่น	การเข้ารหัส	จำนวนตัวอักษร	ลักษณะของข้อมูลที่เก็บ
TRACK 1	210 BPI	ALPHA	79	ชื่อเจ้าของบัตรและหมายเลขเลขบัตร
TRACK 2	75 BPI	BCD	40	หมายเลขบัตรและวันหมดอายุ
TRACK 3	210 BPI	BCD	107	หมายเลขบัตรและรหัสพิเศษ

หมายเหตุ ความหนาแน่นของการบันทึกข้อมูลมีหน่วยเป็น BPI (Bit Per Inch)

2.2.1 ตำแหน่งแทร็กที่สองของบัตรแม่เหล็ก

ตำแหน่งแทร็กที่สองของบัตรแม่เหล็ก เป็นไปตามมาตรฐาน ISO 7811/5 ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 แสดงตำแหน่งแทร็กที่สองของบัตรแม่เหล็ก (ISO 7811/5)

จากภาพที่ 2.1 บริเวณของแทร็กที่สอง จะอยู่ระหว่างเส้นขนาน (b) ค่า 8.97 มิลลิเมตร (0.353 นิ้ว) (ค่าสูงสุด) หรือค่า 8.46 มิลลิเมตร (0.333 นิ้ว) (ค่าต่ำสุด) กับเส้นขนาน c ค่า 11.76 มิลลิเมตร (0.463 นิ้ว)

2.2.2 ชุดรหัสข้อมูลในแทร็กที่สอง

ข้อมูลที่บันทึกในแทร็กที่สองของบัตรแม่เหล็กเป็นตัวเลขอย่างเดียว โดยที่ตัวเลขหนึ่งตัวจะประกอบด้วยบิต (Bit) ข้อมูลแบบ BCD 4 บิตและบิตภาวะเสมอมูล (Parity Bit) 1 บิต ซึ่งใช้ในการตรวจสอบข้อมูลของแต่ละตัวเลข โดยตรวจสอบแบบภาวะเสมอมูลคี่ ISO ได้ระบุจำนวนข้อมูลสูงสุดที่สามารถบันทึกในแทร็กที่สองไว้ไม่เกิน 40 ตัว (รวมสัญลักษณ์เริ่มและสิ้นสุด) แสดงในตารางที่ 2.2 ส่วนชุดรหัสข้อมูลตัวเลขแต่ละตัวสำหรับแทร็กที่สอง แสดงในตารางที่ 2.3

2.2.3 รูปแบบของข้อมูลที่บันทึกในแทร็กที่สองบนบัตรแม่เหล็ก

ตารางที่ 2.2 รูปแบบข้อมูลที่บันทึกในแทร็กที่สองของบัตรแม่เหล็กของธนาคารพาณิชย์ทั่ว ๆ ไป

SYN	B1	ข้อมูล	B2	ข้อมูล	B3	ข้อมูล	B4	LRC	SYN

หมายเหตุ SYN : Synchronization characters LRC : Longitudinal redundancy check

ตารางที่ 2.3 แสดงรหัสข้อมูลตัวเลขสำหรับแทร็กที่สอง

P	b4	b3	b2	b1	รหัส
1	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
1	0	0	1	1	3
0	0	1	0	0	4
1	0	1	0	1	5
1	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8
1	1	0	0	1	9
1	1	0	1	0	A
0	1	0	1	1	B1
1	1	1	0	0	A
0	1	1	0	1	B2
0	1	1	1	0	A
1	1	1	1	1	B3

จากตารางที่ 2.3

A เป็นตำแหน่งของสัญลักษณ์ที่ใช้เฉพาะในระบบควบคุมทางฮาร์ดแวร์(Hard Ware)

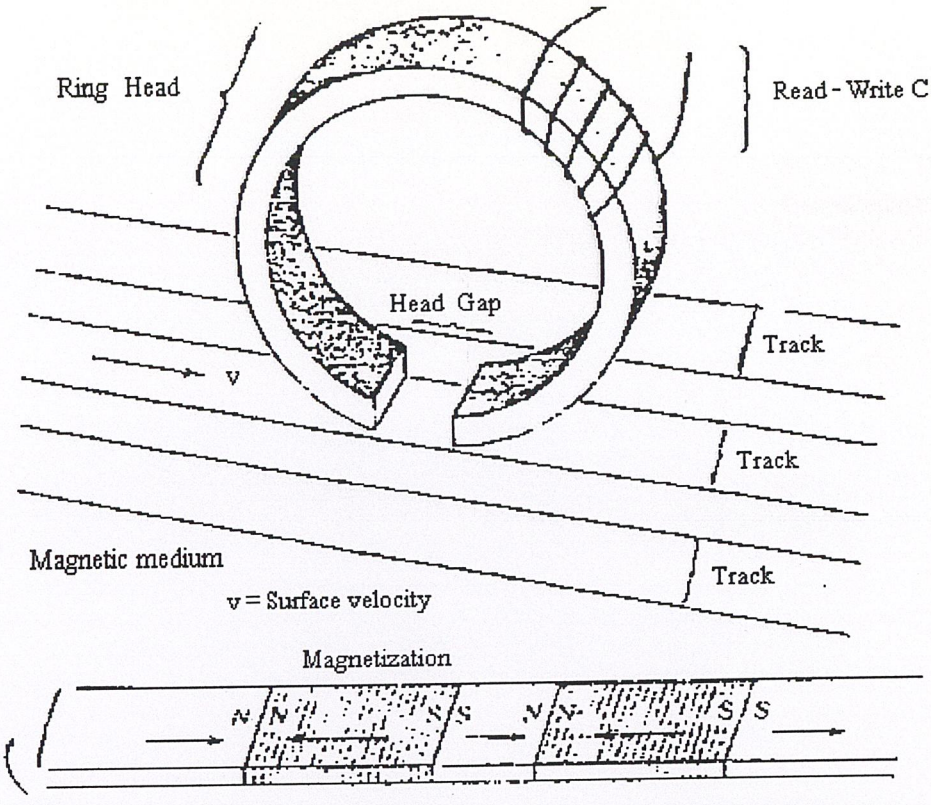
B1 เป็นสัญลักษณ์การเริ่มต้นของข้อมูล (Start Sentinel)

B2 เป็นสัญลักษณ์ตัวแยกข้อมูล (Separator)

B3 เป็นสัญลักษณ์การสิ้นสุดของข้อมูล (Stop Sentinel)

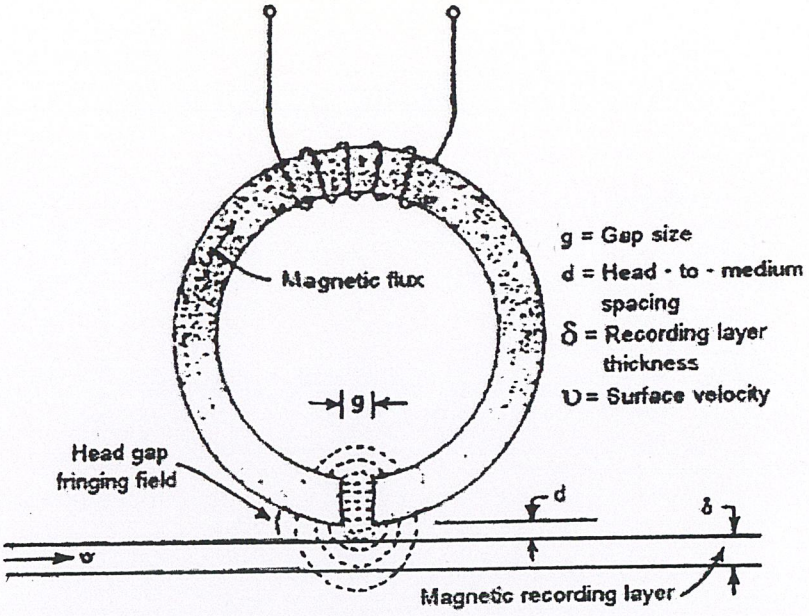
2.3 การบันทึกข้อมูลลงบนบัตรแม่เหล็ก

การบันทึกข้อมูลลงบนบัตรแม่เหล็กประกอบด้วย แถบแม่เหล็กที่ใช้เป็นตัวกลางและหัวแม่เหล็กที่ใช้เป็นหัวบันทึก ซึ่งเคลื่อนที่ด้วยความเร็วคงที่ ดังแสดงในรูปที่ 2.2



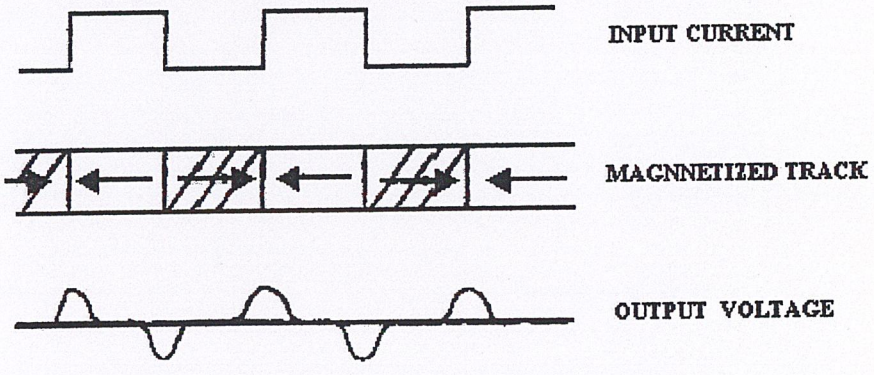
รูปที่ 2.2 แสดงหัวบันทึกและแถบแม่เหล็กในกระบวนการบันทึกข้อมูลลงบนบัตรแม่เหล็ก

จากรูปที่ 2.2 การบันทึกข้อมูลลงบนบัตรแม่เหล็ก จะขึ้นอยู่กับคุณสมบัติของสารแม่เหล็กในแถบแม่เหล็กและหัวแม่เหล็กที่นำมาใช้เป็นหัวบันทึก โดยที่แกนของหัวบันทึกทั่วไปจะเป็นแกนที่มีคุณสมบัติทางแม่เหล็กอย่างอ่อน ๆ พันด้วยขดลวด และที่แกนมีช่องว่าง (Gap) อยู่บริเวณของสารแม่เหล็กที่จะถูกบันทึกข้อมูล เรียกว่า แทร็ก โดยแต่ละแทร็กจะเรียงขนานกันบนแถบแม่เหล็ก สัญญาณเอาท์พุตจากขดลวดที่พันรอบแกน จะเป็นสัดส่วนกับอัตราการเปลี่ยนแปลงฟลักซ์ (Flux) ของหัวบันทึก ความกว้างของหัวบันทึก และความกว้างของแทร็ก การบันทึกข้อมูลลงบนบัตรแม่เหล็กจะใช้วิธีการป้อนกระแสพัลส์ (Pluse) ทั้งด้านบวกและด้านลบ พร้อมทั้งมีขนาดเพียงพอเข้าที่ขดลวดของหัวบันทึกที่วางอยู่ใกล้กับแถบแม่เหล็ก เมื่อป้อนกระแสพัลส์จะก่อให้เกิดสนามแม่เหล็กรอบบริเวณช่องว่างของแกนหัวบันทึกที่วางอยู่ใกล้กับแถบแม่เหล็กซึ่งสนามแม่เหล็กนี้จะใช้ในการบันทึกข้อมูล ดังแสดงในรูปที่ 2.3 ส่วนรูปแบบของกระแสพัลส์ แรงดันไฟฟ้า และขั้วแม่เหล็กในแถบแม่เหล็ก เมื่อมีการบันทึกแล้ว แสดงในรูปที่ 2.4



รูปที่ 2.3 แสดงสนามแม่เหล็กรอบช่องว่างของหัวบันทึก

จากรูปที่ 2.3 ค่าความกว้างของช่องว่าง ต้องมีค่าน้อยกว่าความกว้างของแทร็กข้อมูลเสมอ โดย ISO ได้ระบุความกว้างของช่องว่างหัวบันทึก ๆ ไว้มีค่าประมาณ 0.00625 มิลลิเมตร (0.00025 นิ้ว) หรือน้อยกว่า และค่าช่องว่างของหัวอ่านมีค่าประมาณ 0.025 มิลลิเมตร (0.001 นิ้ว) หรือน้อยกว่า สำหรับค่าความหนาแน่นของเนื้อแถบแม่เหล็ก (δ) ISO ระบุไว้มีค่าไม่เกิน 0.038 มิลลิเมตร



รูปที่ 2.4 แสดงกระแสพัลส์ รูปแบบของแทร็กที่ถูกบันทึก
ฟลักซ์แม่เหล็กที่พื้นผิวแทร็กข้อมูลและลักษณะแรงดันไฟฟ้าของการบันทึก

2.3.1 รูปแบบของการบันทึกข้อมูลบนบัตรแม่เหล็ก

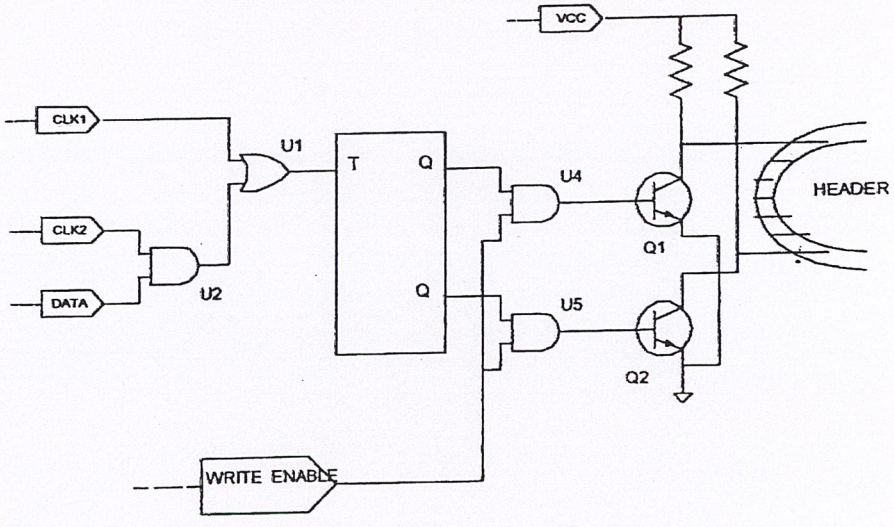
รูปแบบพื้นฐานสำหรับการบันทึกข้อมูลบนบัตรแม่เหล็ก จะใช้สัญญาณความถี่สองความถี่ซึ่งมีเฟส (Phase) ตรงกันมาใช้ในการบันทึกข้อ ซึ่งการบันทึกในลักษณะเช่นนี้จะบันทึกข้อมูลและสัญญาณนาฬิกาที่ใช้ในการบันทึก เข้าไว้ในแทร็กเดียวกัน ซึ่งคิดค้นโดย Aiken (1954)

สัญญาณบันทึกจะประกอบด้วยบิตของข้อมูลที่บันทึก และบิตของสัญญาณนาฬิกา (2F) สำหรับการเปลี่ยนแปลงของฟลักซ์แม่เหล็กของสัญญาณบันทึก จะเกิดขึ้นในระหว่างที่สัญญาณข้อมูลมีสภาวะ “1” ทางตรรก (Logic) หรือระหว่างสัญญาณบิตของข้อมูลติดกัน และจะไม่มีการเปลี่ยนแปลงของฟลักซ์แม่เหล็กในระหว่างที่สัญญาณข้อมูลมีสภาวะ “0” ทางตรรก สำหรับการบันทึกบิตข้อมูลของแต่ละตัวเลข หรือตัวอักษรบนบัตรแม่เหล็ก จะเป็นการบันทึกข้อมูลที่มีค่านัยสำคัญต่ำสุด (b1) ก่อนและบิตภาวะเสมอมูล (p) เป็นบิตสุดท้ายที่ถูกบันทึก ส่วนค่าความหนาแน่นในการบันทึกข้อมูลในแทร็กที่สอง ISO ได้กำหนดให้บันทึกด้วยความหนาแน่น 75 $\pm$ 20 บิตต่อนิ้ว

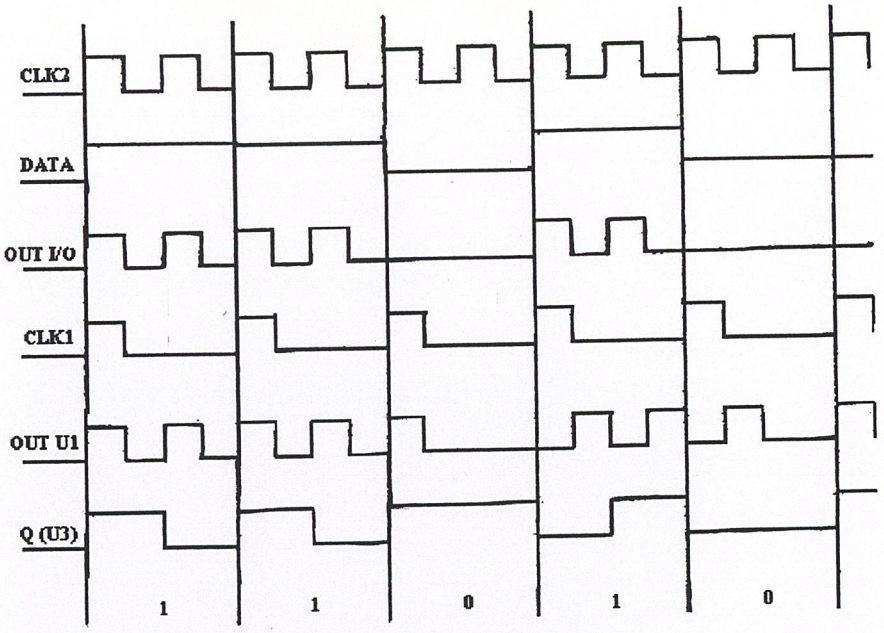
2.3.2 วงจรพื้นฐานสำหรับการบันทึกข้อมูลบนบัตรแม่เหล็ก

วงจรพื้นฐานสำหรับการบันทึกข้อมูลบนบัตรแม่เหล็ก แสดงในรูปที่ 2.5

รูปที่ 2.5 แสดงรูปวงจรพื้นฐาน สำหรับการบันทึกข้อมูลบนบัตรแม่เหล็ก

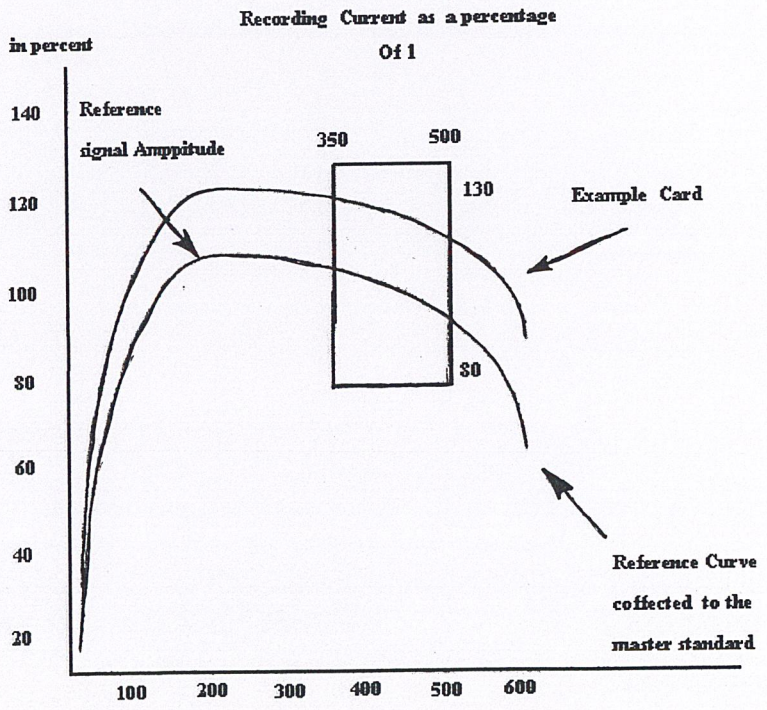


จากรูปที่ 2.5 ไอซี U1, U2, U3, U4 และ U5 จัดรูปวงจรมอดูเลต (Modulation Circuit) สัญญาณข้อมูลและสัญญาณนาฬิกา ให้อยู่ในรูปแบบของสัญญาณเอฟเอ็ม แล้วป้อนสัญญาณดังกล่าวให้กับวงจรจ่ายกระแสไฟฟ้า ให้กับขดลวดที่พันรอบแกนของหัวบันทึก เมื่อทำการบันทึกข้อมูลสัญญาณ White Enable มีตรรกเป็น “1” พร้อมกับทำให้ทรานซิสเตอร์ (Transistor) Q1 และ Q2 ทำงาน ซึ่งทรานซิสเตอร์ทั้งคู่จะสลับกันทำงานเมื่อป้อนกระแสพัลส์ด้านบวกและด้านลบให้แก่ขดลวดตามสัญญาณบันทึก โดยรูปสัญญาณของวงจรพื้นฐานนี้แสดงดังรูปที่ 2.6



รูปที่ 2.6 แสดงรูปสัญญาณการทำงานของวงจรพื้นฐานสำหรับ
การบันทึกข้อมูลบนบัตรแม่เหล็ก

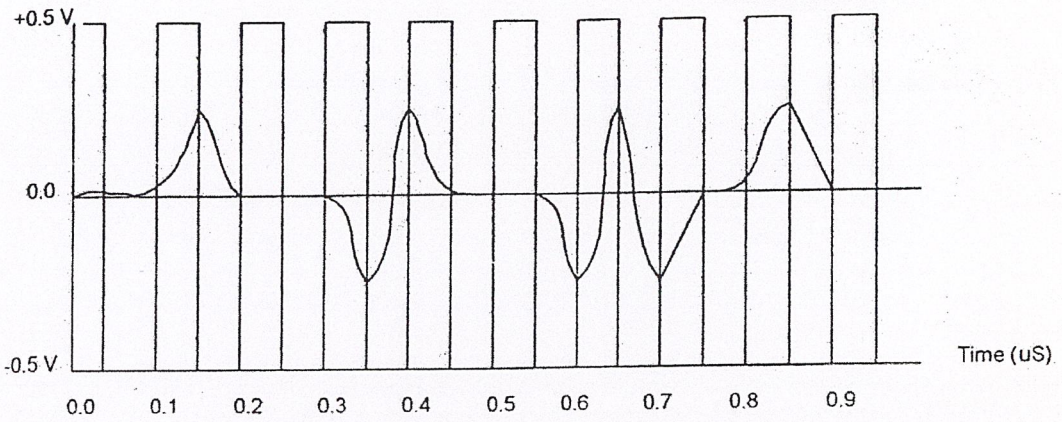
สำหรับค่าของกระแสพัลส์ที่ใช้ในการบันทึกข้อมูล สามารถหาได้จากกราฟแสดงคุณสมบัติการอิ่มตัวของแม่เหล็กบนบัตรแม่เหล็ก ดังแสดงในรูปที่ 2.8



รูปที่ 2.7 กราฟแสดงคุณสมบัติการอิ่มตัวของแถบแม่เหล็กบนบัตรแม่เหล็กมาตรฐาน

2.4 การอ่านข้อมูลจากบัตรแม่เหล็ก

การอ่านข้อมูลจากบัตรแม่เหล็กสามารถทำได้โดยให้แถบแม่เหล็กสัมผัสกับหัวอ่าน ซึ่งเคลื่อนที่ด้วยความเร็วคงที่ ดังแสดงในรูปที่ 2.2 ฟลักซ์แม่เหล็กที่เกิดขึ้นจะผ่านจากช่องว่างของแกนหัวอ่าน ไปยังขดลวดที่พันรอบแกนอยู่ ซึ่งการเปลี่ยนแปลงของฟลักซ์แม่เหล็กตามข้อมูลที่บันทึกจะทำให้เกิดแรงดันไฟฟ้าที่ขดลวดของหัวอ่านตามข้อมูลนั้น ดังแสดงในรูปที่ 2.8

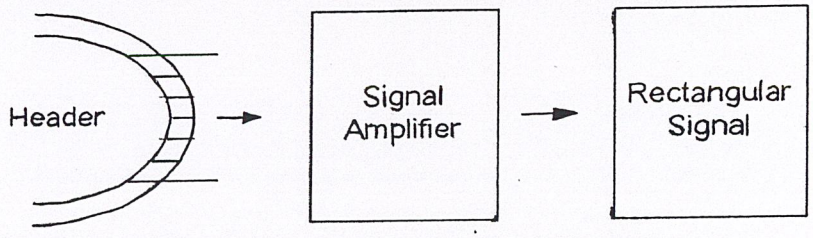


รูปที่ 2.8 แสดงสัญญาณแรงดันไฟฟ้าจากหัวอ่าน
และถูกขยายสัญญาณด้วยอัตราขยายประมาณ 100 เท่า

จากรูปที่ 2.8 ตำแหน่งสูงสุดของแรงดันไฟฟ้าที่อ่านได้จะตรงกับตำแหน่งที่สนามแม่เหล็กบนบัตรแม่เหล็ก มีการกลับทิศทาง ทำให้สามารถนำตำแหน่งนี้มาสร้างสัญญาณพัลส์ เพื่ออ่านข้อมูลที่บันทึกอยู่บนบัตรแม่เหล็กได้

2.4.1 วงจรพื้นฐานสำหรับสร้างสัญญาณรูปแบบ ISO จากบัตรแม่เหล็ก

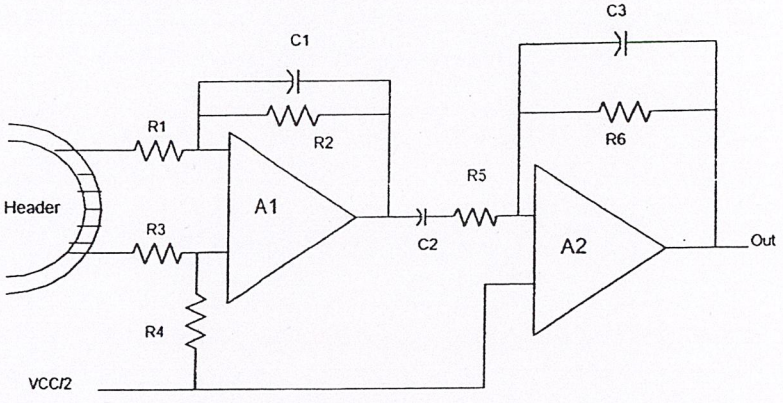
ส่วนประกอบของวงจรพื้นฐาน สำหรับสร้างสัญญาณรูปแบบ ISO จากบัตรแม่เหล็ก แสดงดังรูปที่ 2.9



รูปที่ 2.9 แสดงส่วนประกอบของวงจรพื้นฐานสร้างสัญญาณรูปแบบ ISO จากบัตรแม่เหล็ก

1) วงจรขยายสัญญาณแรงดันไฟฟ้า สัญญาณแรงดันไฟฟ้าจากหัวอ่านข้อมูลที่ค่าค่อนข้างต่ำ (5-10 มิลลิโวลท์) จึงจำเป็นที่ต้องขยายสัญญาณเพื่อสะดวกต่อการนำไปสร้างสัญญาณรูปเหลี่ยม โดยการใช้วงจรขยายสัญญาณสองชุดและกำหนดค่ากำลังขยายสัญญาณ รวมมีค่าประมาณ 1000 เท่า (ไม่ให้สัญญาณอิ่มตัว) ดังแสดงไว้ในรูปที่ 2.10

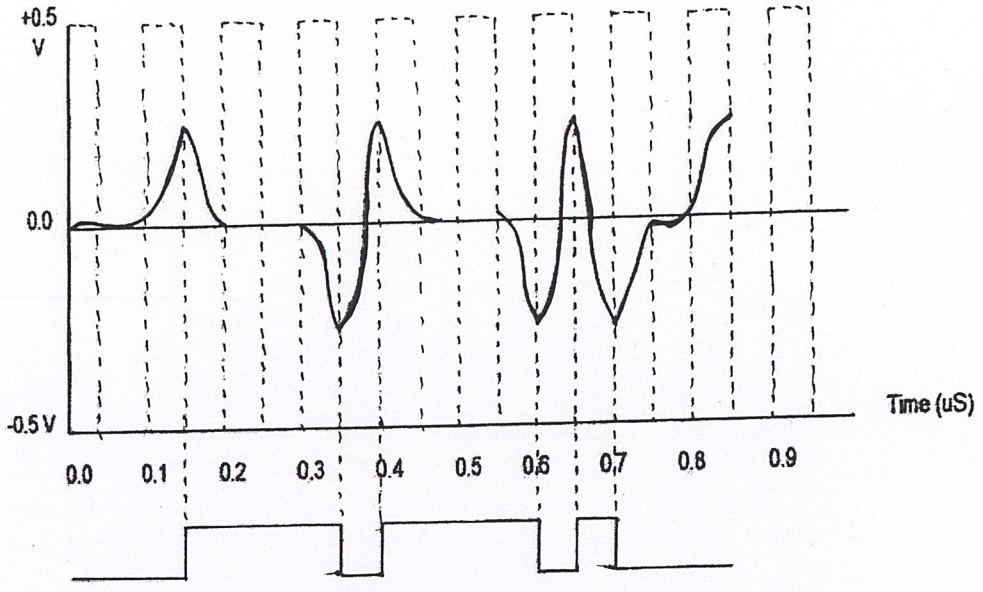
- วงจรขยายสัญญาณความแตกต่าง (Differential Amplifier) เป็นวงจรขยายสัญญาณชุดแรกที่รับสัญญาณข้อมูลจากขั้วทั้งสองของหัวอ่านข้อมูล ซึ่งมีค่าความต้านทานต่ำ และใช้ 1C A_1 ทำหน้าที่ขยายสัญญาณความแตกต่าง มีค่าอัตราการขยาย $AV = -R_2/R_1$
- วงจรขยายสัญญาณแบบกลับเฟส (Inverting Amplifier) เพื่อให้ได้สัญญาณข้อมูลที่มีค่าแรงดันไฟฟ้าสูงขึ้นพอที่จะป้อนให้กับวงจรสัญญาณรูปเหลี่ยมจึงนำสัญญาณข้อมูลมาผ่านวงจรแบบกลับเฟส ซึ่งอัตราขยายแรงดันไฟฟ้ามีค่า $AV = -R_6/R_5$



รูปที่ 2.10 แสดงวงจรขยายสัญญาณแรงดันไฟฟ้าสำหรับอ่านข้อมูลบัตรแม่เหล็ก

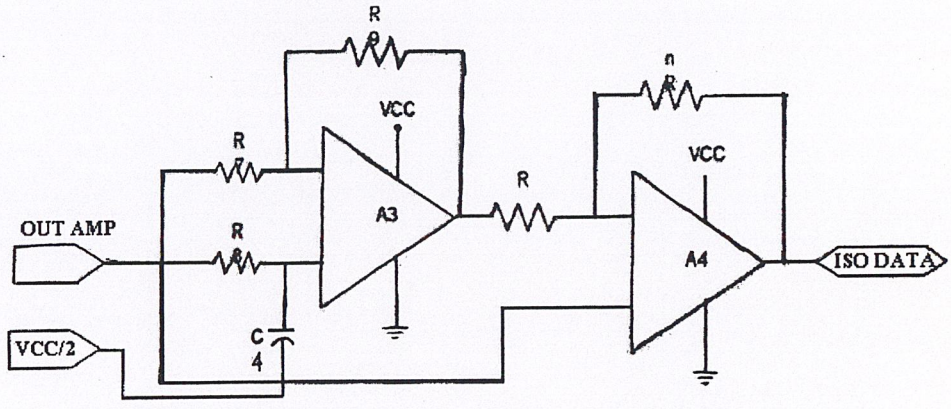
2) วงจรสร้างสัญญาณรูปเหลี่ยม (รูปแบบ ISO) จากสัญญาณข้อมูลที่ขยายแล้ว (จากรูปที่ 2.8) ทำให้สามารถพิจารณารูปแบบการสร้างสัญญาณรูปเหลี่ยมได้หลายรูปแบบ โดยเฉพาะการนำตำแหน่งจุดยอดของสัญญาณข้อมูลทั้งด้านบนและด้านล่างมาใช้ในการสร้างสัญญาณรูปเหลี่ยม จะทำให้ได้สัญญาณพัลส์ที่เหมือนกับสัญญาณที่ใช้ในการบันทึกข้อมูลบนบัตรแม่เหล็ก สำหรับการสร้างสัญญาณรูปเหลี่ยมสามารถสร้างได้ 2 วิธี

- วงจรเลื่อนเฟสและวงจรเปรียบเทียบแรงดันไฟฟ้า ใช้หลักการหน่วงสัญญาณข้อมูลในช่วงเวลาสั้นมาก ๆ แล้วนำสัญญาณข้อมูลจริง (เส้นทึบ) เปรียบเทียบกับสัญญาณข้อมูลที่ถูกหน่วง (เส้นประ) ซึ่งสัญญาณเอาท์พุทจากการเปรียบเทียบนี้แสดงไว้ในรูปที่ 2.11



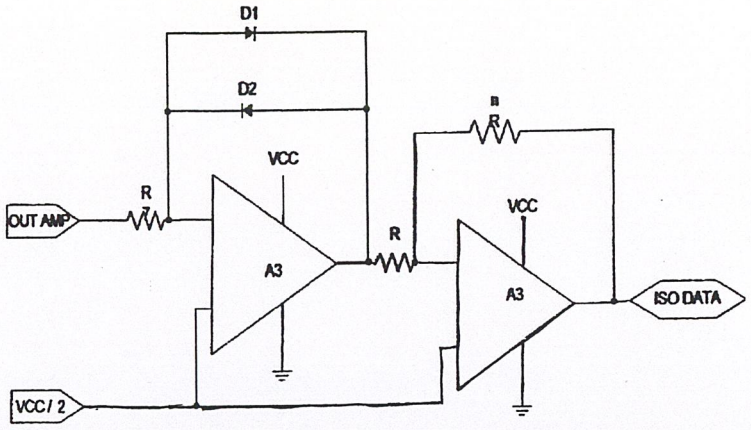
รูปที่ 2.11 แสดงการเปรียบเทียบสัญญาณ และลักษณะสัญญาณที่ต้องการ

จากรูปที่ 2.11 วงจรที่ใช้งานจะใช้ออปแอมป์ (Op Amp) สองตัว โดยที่ออปแอมป์ตัวแรกจัดรูปวงจรเป็นวงจรเลื่อนเฟส เพื่อให้ได้สัญญาณข้อมูลที่ถูกหน่วง (เอาท์พุตเป็นสัญญาณเส้นประ) และออปแอมป์ตัวที่สอง จัดรูปวงจรเป็นวงจรเปรียบเทียบแรงดันระหว่างสัญญาณข้อมูลกับสัญญาณที่ถูกหน่วง ดังแสดงไว้ในรูปที่ 2.12



รูปที่ 2.12 แสดงวงจรเลื่อนเฟส และวงจรเปรียบเทียบสัญญาณแรงดันไฟฟ้า

- วงจรดิฟเฟอร์เรนทิเอเตอร์และวงจรเปรียบเทียบแรงดันไฟฟ้า แนวทางที่สอง จะใช้วงจรดิฟเฟอร์เรนทิเอเตอร์ซึ่งมีไดโอด D_1 และ D_2 ประพฤติตัวเสมือนตัวต้านทานป้อนกลับด้านด้านลบของวงจรทำงานร่วมกับวงจรเปรียบเทียบแรงดันไฟฟ้า ดังแสดงในรูปที่ 2.13



รูปที่ 2.13 แสดงวงจรดิฟเฟอร์เรนทิเอเตอร์ และวงจรเปรียบเทียบแรงดันไฟฟ้า

จากรูปที่ 2.13 เมื่อพิจารณาสัญญาณข้อมูลจากหัวอ่านข้อมูล ซึ่งมีลักษณะของสัญญาณคล้ายรูปคลื่นสามเหลี่ยม เมื่อป้อนสัญญาณข้อมูลให้กับวงจรดิฟเฟอร์เรนทิเอเตอร์ สัญญาณเอาท์พุตที่ได้จะเป็นสัดส่วนกับค่าอนุพันธ์ของสัญญาณเอาท์พุต เมื่อสัญญาณข้อมูลเป็นรูปสามเหลี่ยมที่มีค่าสโลป (Slope) เป็นค่าบวกและค่าลบที่แกว่งบนแรงดันอ้างอิง (Vcc/2) สัญญาณเอาท์พุตของวงจรดิฟเฟอร์เรนทิเอเตอร์จะเป็นสัญญาณรูปคลื่นสี่เหลี่ยม มีค่าแรงดันไฟฟ้าเอาท์พุต ตามสมการที่ 2.1

$$V_o = -RCk_n \qquad \text{....(2.1)}$$

เมื่อค่า k_n เป็นค่าสโลปของสัญญาณอินพุต (โวลต์ต่อวินาที) และใช้วงจรเปรียบเทียบสัญญาณแรงดันไฟฟ้าเป็นส่วนสร้างสัญญาณรูปเหลี่ยมให้ดียิ่งขึ้น

สำหรับวงจรเปรียบเทียบแรงดันไฟฟ้าของทั้งสองแนวทาง จะใช้ตัวต้านทาน R และ nR จัดรูปวงจรแบบป้อนกลับทางด้านบวก เพื่อป้องกันให้วงจรเกิดการออสซิลเลท โดยมีค่าแรงดันขีดเริ่มเปลี่ยนด้านสูง (Upper Threshold Voltage : V_{UT}), ค่าแรงดันขีดเริ่มเปลี่ยนด้านต่ำ (Lowwer Threshold Voltage : V_{LT}) และค่าศักดาไฟฟ้าฮิสเตอร์ซิส (V_T) ตามสมการที่ 2.2, 2.3 และ 2.4 ตามลำดับ

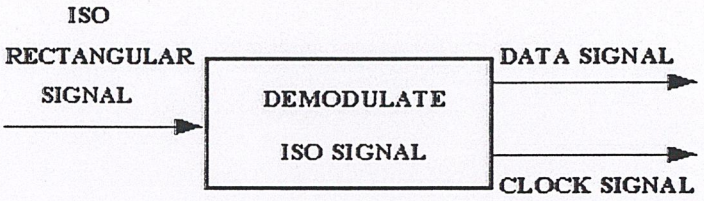
$$V_{UT} = Vref.[1+(1/n)] - (Vsat/n) \qquad \text{....(2.2)}$$

$$V_{LT} = Vref.[1+(1/n)] - (+Vsat/n) \qquad \text{....(2.3)}$$

$$V_T = [(+Vsat) - (Vsat)]/n \qquad \text{....(2.4)}$$

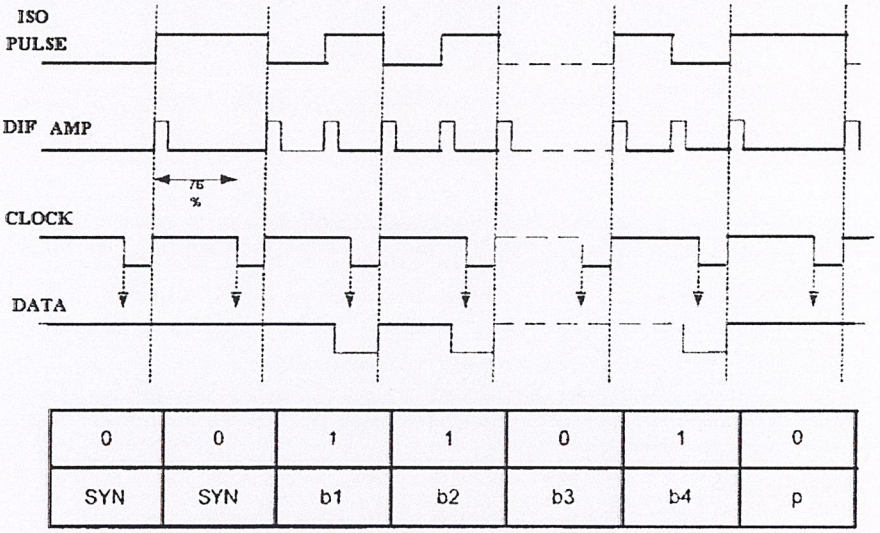
2.4.2 **วงจรอ่านข้อมูลบัตรแม่เหล็ก**

วงจรพื้นฐานในหัวข้อ 2.4.1 สามารถนำสัญญาณดังหล่าวมาประยุกต์ใช้งานต่อเนื่องได้ เพื่อให้อ่านข้อมูลจากบัตรแม่เหล็กได้สะดวกขึ้น โดยการดีมอดูเลต (Demodulate) สัญญาณรูปสามเหลี่ยม ISO เพื่อให้ได้สัญญาณนาฬิกาสำหรับการอ่านและสัญญาณข้อมูล ดังแสดงในภาพที่ 2.15



รูปที่ 2.14 แสดงแผนภาพของวงจรอ่านบัตรแม่เหล็ก

การแยกสัญญาณข้อมูลออกจากสัญญาณนาฬิกา โดยใช้หลักการดีมอดูเลตสัญญาณ ISO สามารถพิจารณาได้จากรูปคลื่นสัญญาณ ดังแสดงในรูปที่ 2.15



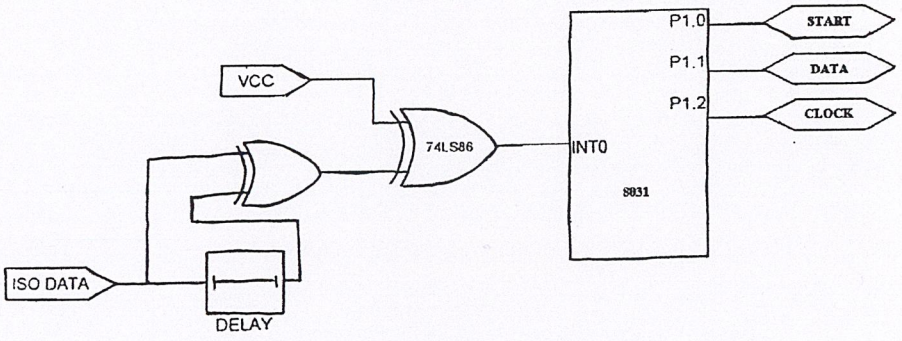
รูปที่ 2.15 แสดงรูปคลื่นสัญญาณของ ISO ที่ถูกดีมอดูเลต

จากรูปที่ 2.15 การแยกสัญญาณข้อมูลออกจากสัญญาณรูปเหลี่ยม ISO ได้นั้น จะต้องพิจารณาการกลับขั้วของสัญญาณรูปเหลี่ยม ISO ในช่วงเวลามาตรฐาน (Standard Time) หรือไม่ ถ้ามีการกลับขั้วของสัญญาณรูปเหลี่ยม ISO แสดงว่าข้อมูลมีค่าทางตรรกเป็น “1” และถ้าไม่มีการกลับขั้วของสัญญาณรูปเหลี่ยม ISO แสดงว่าข้อมูลมีค่าทางตรรกเป็น “0” โดยที่ค่าเวลามาตรฐาน มีค่าประมาณ 75% ของคาบเวลา 1 บิต ส่วนสัญญาณที่ได้จากการดีมอดูเลตจะมีสัญญาณข้อมูล และสัญญาณนาฬิกาสำหรับการอ่านข้อมูลโดยที่ทุก ๆ ขอบขาลงของสัญญาณนาฬิกา บิตข้อมูลจริงจะ

กลับตรรกกับสัญญาณข้อมูลที่อ่านได้ สำหรับการสร้างสัญญาณเวลาเพื่อใช้สำหรับแยกสัญญาณข้อมูลออกมานั้นมี 2 วิธี คือ แบบความยาวคงที่ และแบบความยาวเปลี่ยนแปลงได้

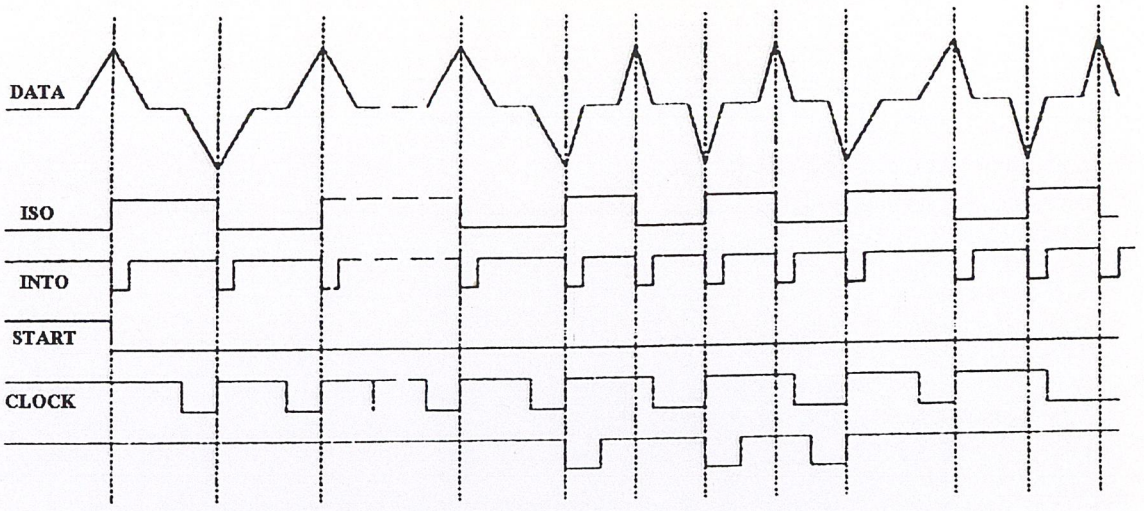
1) แบบความยาวคงที่ สามารถกำหนดเวลามาตรฐานให้มีค่าคงที่ และเปรียบเทียบกับทุกบิต วิธีนี้เหมาะกับชุดอ่านบัตรแบบที่ใช้มอเตอร์ดึงบัตรผ่านหัวอ่านด้วยความเร็วคงที่

2) แบบความยาวเปลี่ยนแปลงได้ ค่าเวลามาตรฐานมีค่าไม่คงที่ สัญญาณเวลามาตรฐานจะถูกสร้างจากความยาวของบิตก่อนหน้านี้ 1 บิต และสัญญาณเวลามาตรฐานลูกต่อไปจะสร้างจากความยาวของบิตปัจจุบัน โดยจะเปรียบเทียบกันไปอย่างนี้จนหมดข้อมูล ซึ่งวิธีนี้ใช้กับการอ่านบัตรแบบบัตรที่ใช้การรูดบัตรผ่านหัวอ่านด้วยมือ เนื่องจากความเร็วในการรูดมีค่าไม่แน่นอน และจากรูปคลื่นสัญญาณที่แสดงในรูปที่ 2.15 ได้นำไมโครคอนโทรลเลอร์ AT89S8252 มาประยุกต์ในการสร้างข้อมูลและสัญญาณเวลามาตรฐานทจากสัญญาณรูปเหลี่ยม ISO โดยจัดรูปวงจรแสดงในรูปที่ 2.16



รูปที่ 2.16 แสดงวงจรสร้างสัญญาณข้อมูล และสัญญาณนาฬิกาอ่านข้อมูล โดยใช้ AT89S8252

จากรูปที่ 2.16 หลักการทำงานจะใช้ ไอซี 74LS86 เป็นตัวสร้างสัญญาณ Difference Pulse แล้วป้อนให้กับ INTO การทำงานในส่วนของ AT89S8252 จะใช้ Time 0 ของ AT89S8252 เป็นตัวจับคาบเวลาแต่ละช่วงเวลาของ Difference Pulse ถ้าช่วงเวลามีค่ามากกว่าเวลามาตรฐาน (5% ของช่วงเวลาระหว่าง Difference Pulse ชุดก่อน) จะสร้างสัญญาณนาฬิกาทันที แต่จะไปสร้างสัญญาณนาฬิกา ณ ที่เวลา 75% ของช่วงเวลาชุดก่อนพร้อมกับสร้างสัญญาณข้อมูล “0” โดยใช้พอร์ต P1.0 เป็นขาสัญญาณการเริ่มต้นของการรูดบัตรพอร์ต P1.1 เป็นขาสัญญาณข้อมูลและพอร์ต P1.2 เป็นขาสัญญาณนาฬิกาสำหรับการอ่านข้อมูล โดยรูปสัญญาณเอาท์พุตทั้งสาม แสดงในรูปที่ 2.17



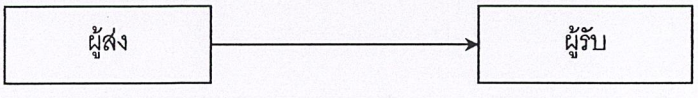
รูปที่ 2.17 แสดงรูปสัญญาณเอาท์พุตของ AT89S8252 สำหรับสร้างสัญญาณอ่านบัตรแม่เหล็ก

2.5 การส่งและรับข้อมูล (Data transmission)

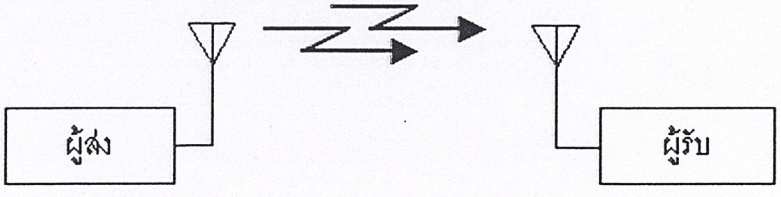
การส่ง-รับข้อมูลเพื่อโอนถ่ายหรือแลกเปลี่ยนข้อมูลกันระหว่างผู้ส่งและผู้รับ จะสำเร็จขึ้นได้ต้องประกอบด้วยปัจจัยสำคัญ 2 ประการ คือ คุณภาพของที่ส่งรับกัน และคุณลักษณะของสายสื่อสารสำหรับส่งผ่านข้อมูล

การส่งสัญญาณข้อมูล หมายถึง การส่ง (นำ) ข้อมูลหรือข่าวสารจากเครื่องส่งหรือผู้ส่ง ผ่านทางสื่อหรือตัวกลางไปยังเครื่องรับหรือผู้รับ ข้อมูลหรือข่าวสารที่ถูกส่งออกไปอาจจะอยู่ในรูปของสัญญาณเสียงสัญญาณคลื่นแม่เหล็กไฟฟ้าหรือแสงก็ได้ โดยที่สื่อหรือตัวกลางของสัญญาณสามารถแบ่งออกได้เป็น 2 แบบ คือ

1. แบบสามารถกำหนดเส้นทางสัญญาณได้ ได้แก่ สายเกลียวคู่ สายโคแอคเชียล และสายเบอร์ ออบติค
2. แบบไม่สามารถกำหนดเส้นทางสัญญาณได้ ได้แก่ ชั้นบรรยากาศ สูญญากาศ และน้ำ



รูปที่ 2.18 การกำหนดเส้นทางสัญญาณได้

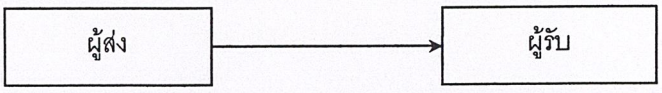


รูปที่ 2.19 การกำหนดเส้นทางสัญญาณไม่ได้

2.6 รูปแบบของการส่งสัญญาณข้อมูล

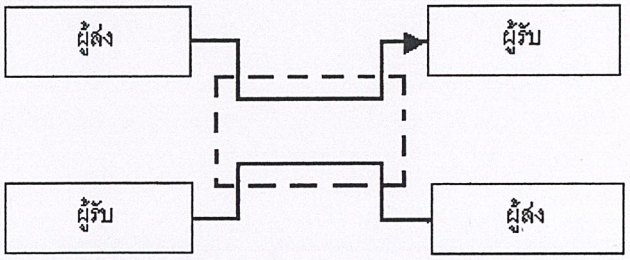
เราสามารถจัดแบ่งรูปแบบของการส่งสัญญาณข้อมูลได้เป็น 4 รูปแบบ ดังนี้

1. แบบทิศทางเดียวหรือซิมเพล็กซ์ (One-way หรือ Simplex) ในการส่งสัญญาณข้อมูลแบบ Simplex ข้อมูลจะถูกส่งไปในทางเดียวเท่านั้น เช่น การกระจายเสียงของสถานีวิทยุ หรือการแพร่ภาพทางโทรทัศน์



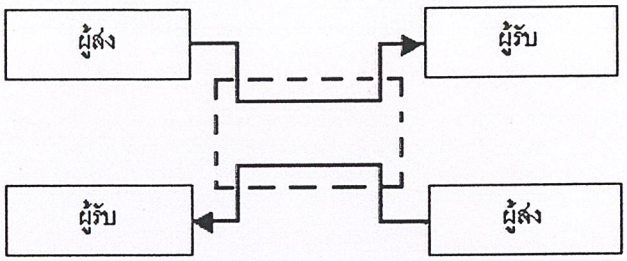
รูปที่ 2.20 แบบทิศทางเดียว

2. แบบกึ่งทางคู่หรือครึ่งดูเพล็กซ์ (Either-Way of Two Ways หรือ Half Duplex) การสื่อสารแบบครึ่งดูเพล็กซ์ เราสามารถส่งข้อมูลสวนทางกันได้ แต่ต้องสลับกันส่ง จะทำในเวลาเดียวกันไม่ได้ เช่น วิทยุสื่อสารของตำรวจ ซึ่งต้องอาศัยการสลับสวิตซ์เพื่อแสดงการเป็นผู้ส่งสัญญาณ และให้ทางอีกด้านหนึ่งเป็นผู้รับสัญญาณ คือ ต้องผลัดกันพูด



รูปที่ 2.21 แบบกึ่งทางคู่

3. แบบทางคู่หรือดูเพล็กซ์เต็ม (Both-Way หรือ Full Duplex) ในแบบนี้เราสามารถส่งข้อมูลได้ในเวลาเดียวกันทั้งสองทาง เช่น การพูดโทรศัพท์ สามารถพูดพร้อมกันได้ ประโยชน์การใช้งานของการส่งสัญญาณแบบดูเพล็กซ์เต็มย่อมให้ประโยชน์ใช้สอยได้ดีกว่า รวมทั้งลดเวลาในการส่งสัญญาณ อย่างไรก็ตามค่าใช้จ่ายในการติดตั้งและอุปกรณ์ของระบบการส่งสัญญาณแบบดูเพล็กซ์เต็มย่อมแพงกว่า และยุ่งยากกว่าเช่นนั้น



รูปที่ 2.22 แบบทางคู่

2.7 การสื่อสารระหว่างเครื่องอ่านบัตรแม่เหล็ก และคอมพิวเตอร์

การส่งข้อมูลจากเครื่องอ่านบัตรแม่เหล็กไปยังคอมพิวเตอร์ สามารถส่งข้อมูลผ่านพอร์ตต่าง ๆ ของคอมพิวเตอร์ แบ่งออกได้ดังนี้

1. การสื่อข้อมูลแบบขนาน
2. การสื่อข้อมูลแบบอนุกรม

2.8 การสื่อสารข้อมูลแบบขนาน

พอร์ตข้อมูลแบบขนานนั้นเป็นที่ใช้กันอย่างมาก สำหรับการควบคุมหรือติดต่อกับอุปกรณ์ภายนอกด้วยคอมพิวเตอร์ ซึ่งจะใช้อุปกรณ์ภายนอกต่อร่วมด้วยน้อยมากเมื่อเปรียบเทียบกับพอร์ตข้อมูล พอร์ตข้อมูลแบบขนานนี้สามารถใช้เป็นอินพุตได้ 9 บิต และใช้เป็นเอาท์พุตได้ 12 บิต โดยพอร์ตนี้จะประกอบด้วยสายควบคุม 4 เส้น สายแสดงสถานะ 5 เส้น และสายข้อมูลอีก 8 เส้น ซึ่งโดยปกติพอร์ตข้อมูลแบบขนานนี้จะเป็นแบบ D-type ตัวเมียขนาด 25 pin ติดอยู่ที่ด้านหลังของคอมพิวเตอร์ ส่วนอีกพอร์ตหนึ่งที่เป็น D-type ตัวผู้ขนาด 25 pin เช่นกันนั้นเป็นพอร์ตข้อมูลแบบอนุกรม ซึ่งแตกต่างจากพอร์ตข้อมูลแบบขนานโดยสิ้นเชิง

มาตราฐานของพอร์ตขนานได้มีการกำหนดขึ้นมาใหม่โดย IEEE 1284 ซึ่งพอร์ตข้อมูลแบบขนานจะต้องสามารถทำงานได้ตามโหมดต่าง ๆ ดังนี้

1. Compatibility Mode
2. Nibble Mode
3. Byte Mode
4. EPP Mode (Enhanced Parallel Port)
5. ECP Mode (Extended Capabilities Port)

โดยที่ไดร์เวอร์หรืออุปกรณ์จะต้องสนับสนุนต่อการทำงานในโหมดต่าง ๆ และยังต้องสนับสนุนต่อการใช้งานของพอร์ตมาตรฐานแบบขนาน (SPP : Standard Parallel) ด้วย การทำงาน

ที่ทำหน้าที่ติดต่อสื่อสารภายนอกร่วมด้วย การทำงานในสองโหมดหลังนี้จะทำให้มีความเร็วในการรับส่งสูงขึ้น และยังคงสนับสนุนกับการทำงานของพอร์ตมาตราฐานแบบขนานด้วย

ในโหมด Compatibility หรือเรียกอีกชื่อหนึ่งว่า Centronics Mode จะสามารถส่งข้อมูลได้ในทิศทางเดียวเท่านั้น ซึ่งปกติจะมีความเร็วประมาณ 50 KB/s ในการรับส่งจากพอร์ตขนานจะต้องเปลี่ยนมาทำงานในโหมด Nibble หรือ Byte ในโหมด Nibble จะสามารถรับข้อมูลได้ 4 บิต เช่น การรับข้อมูลจากอุปกรณ์ภายนอกมายังคอมพิวเตอร์ ส่วนการทำงานแบบ Byte Mode จะเป็นการทำงานในแบบสองทิศทาง (bi-direction) ซึ่งจะพบในการ์ดบางตัวเท่านั้น โดยจะสามารถรับส่งข้อมูลได้ครั้งละ 8 บิต

สำหรับการทำงานในโหมด EPP และ ECP นั้นจะต้องมีส่วนของอุปกรณ์ภายนอกร่วมด้วยเพื่อสร้างและจัดการกับสัญญาณที่ใช้สำหรับการแฮนด์เชค ในกาส่งข้อมูลไปยังเครื่องพิมพ์ (printer) จะใช้โหมด Compatibility โดยโปรแกรมที่ส่งข้อมูลจะมีลำดับขั้นการทำงาน ดังนี้

1. ส่งข้อมูลครั้งละ 8 บิต ไปยังพอร์ตข้อมูล
2. ตรวจสอบสัญญาณ busy ของเครื่องพิมพ์ กรณีที่สัญญาณ busy active ในช่วงเวลานี้ข้อมูลเหล่านั้นก็จะสูญหายไป
3. เปลี่ยนสัญญาณ Strobe เป็น “0” เพื่อบอกให้เครื่องพิมพ์รู้ว่าข้อมูลเป็นข้อมูลที่ถูกต้อง
4. เปลี่ยนสัญญาณ Strobe เป็น “1” อีกครั้งหลังจากที่ได้เปลี่ยนสัญญาณ Strobe เป็น “0” แล้วประมาณ 5 μS

ขั้นตอนต่าง ๆ เหล่านี้จะเป็นตัวกำหนดความเร็วในการรับส่งข้อมูลแต่ในโหมด EPP และ ECP จะใช้ฮาร์ดแวร์เป็นตัวตรวจสอบว่าเครื่องพิมพ์ว่างอยู่หรือไม่ และสร้างสัญญาณ Strobe และสัญญาณในการทำ handshaking จากภายนอก ดังนั้นการทำงานในสองโหมดนี้จึงสามารถรับส่งข้อมูลได้ด้วยความเร็วประมาณ 1-2 MB/s แต่ในโหมด ECP จะมีการใช้ DMA channel ร่วมกับบัฟเฟอร์แบบ FIFO ซึ่งทำให้มีความเร็วมากกว่าโหมด EPP

2.9 ลักษณะทางฮาร์ดแวร์

ลักษณะภายนอกของพอร์ตข้อมูลแบบขนานได้แสดงไว้ดังในรูปที่ 1 ซึ่งเป็นคอนเน็คเตอร์อบบ D-type ขนาด 25 pin รูป 1(ก) และคอนเน็คเตอร์แบบ Centronics ขนาด 32 pin รูป 1(ข) ส่วนหน้าที่ของขาสัญญาณต่าง ๆ ได้แสดงไว้ในตารางที่ 1 โดยทั่วไปแล้วคอนเน็คเตอร์แบบ D-type จะติดอยู่กับคอมพิวเตอร์ ส่วนคอนเน็คเตอร์แบบ Centronics จะอยู่ที่เครื่องพิมพ์ ตามมาตราฐาน IEEE 1284 ได้กำหนดคอนเน็คเตอร์ที่ใช้กับพอร์ตขนานไว้ 3 แบบคือ 1284 Type A เป็นคอนเน็คเตอร์แบบ D-type ขนาด 25 pin ซึ่งจะอยู่ที่คอมพิวเตอร์ แบบที่สองคือ 1284 Type B เป็นคอนเน็ตเตอร์แบบ Centronics ซึ่งจะอยู่ที่เครื่องพิมพ์เป็นส่วนใหญ่ ส่วนแบบที่สามคือ 1284 Type C ซึ่งมีขนาด

36 pin เช่นเดียวกันแต่มีขนาดเล็กกว่า และเพิ่มคลิปสำหรับล็อคสายไว้ด้วย ซึ่งจะเพิ่มสายสัญญาณอีก 2 เส้น เพื่อตรวจสอบว่ามีอุปกรณ์ต่ออยู่หรือไม่

ตารางที่ 2.4 การกำหนดสัญญาณของพอร์ตข้อมูลแบบขนาน

Pin No. (D-type 25)	Pin No. (Centronics)	SPP Signal	Direction In/Out	Register	Hardware Inverted
1	1	Strobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data	Out	Data	
5	5	Data	Out	Data	
6	6	Data	Out	Data	
7	7	Data	Out	Data	
8	8	Data	Out	Data	
9	9	Data	Out	Data	
10	10	Ack	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out / Paper End	In	Status	
13	13	Select	In	Status	
14	14	Auto-Linefeed	In/Out	Control	Yes
15	3	Error/Fault	In	Status	
16	31	Initialize	In/Out	Control	Yes
17	36	Select-Printer Select-In	In/Out	Control	
18-25	19-30	Ground	In/Out		

2.10 ตำแหน่งพอร์ต

พอร์ตข้อมูลแบบขนานจะใช้พอร์ต 3 พอร์ต ซึ่งแสดงไว้ในตารางที่ 2 พอร์ตหมายเลข 3BCh เป็นพอร์ตขนานที่อยู่บน Video Card ซึ่งใช้ในช่วงแรก ๆ ตำแหน่งของพอร์ตขนานเราสามารถตรวจสอบได้ โดยการอ่านค่าจาก BIOS (Basic Input/Output System) ของคอมพิวเตอร์

พอร์ตขนานพอร์ตแรกหรือ LTP ปกติจะกำหนดให้อยู่ที่หมายเลข 378h ขณะที่ LTP2 จะอยู่ที่พอร์ตหมายเลข 278h ซึ่งตำแหน่งทั้งสองนี้จะใช้กันมากสำหรับพอร์ตขนาน ตัวอักษร h แสดงว่าเป็นเลขฐานสิบหก ตำแหน่งของพอร์ตเหล่านี้อาจจะแตกต่างกันไปตามชนิดของคอมพิวเตอร์

เมื่อเปิดคอมพิวเตอร์ BIOS จะกำหนดหมายเลขของที่มีอยู่บนเครื่องและกำหนดชื่อเรียก LTP1, LTP2 และ LTP3 ให้กับพอร์ตเหล่านั้น ในครั้งแรก BIOS จะตรวจสอบที่ตำแหน่ง 3BCh หากพบพอร์ตขนานที่ตำแหน่งนี้จะกำหนดให้เป็น LTP1 หลังจากนั้นจึงตรวจสอบที่ตำแหน่ง 378h หากพบพอร์ตขนานอีกจะกำหนดด้วยชื่อเรียกของอุปกรณ์ตัวต่อไป ซึ่งอาจทำให้ผู้ใช้เกิดการสับสนขึ้นได้ ในบางการ์ดของพอร์ตขนานจะมีสายจัมพ์ ซึ่งผู้ใช้สามารถกำหนดพอร์ต LTP1, LTP2 และ LTP3 ได้ ซึ่งบางครั้งตำแหน่ง 378h อาจจะไม่ใช่ LTP1 ก็ได้

ตารางที่ 2.5 ตำแหน่งพอร์ต

Address	Notes :
3BCh	Used for parallel ports which were incorporated in to video card And now, commonly an option for port controlled by BIOS
378h-37Fh	Usual Address for LPT1
278h-27Fh	Usual Address for LPT2

โดยทั่วไปเราจะติดต่อผ่าน LTP1 แต่จะต้องหาตำแหน่งที่แน่นอนของพอร์ตให้ได้ ซึ่งสามารถทำได้โดยการตรวจสอบผ่าน BIOS เมื่อ BIOS กำหนดตำแหน่งเครื่องพิมพ์แล้วจะเก็บตำแหน่งไว้ในหน่วยความจำ ตารางที่ 3 แสดงตำแหน่งในหน่วยความจำที่ BIOS เก็บตำแหน่งของเครื่องพิมพ์ไว้

ตารางที่ 2.6 ตำแหน่ง LPT ที่เก็บไว้ใน BIOS

Start Address	Function
0000 : 0408	LPT1's Base Address
0000 : 040A	LPT2's Base Address
0000 : 040C	LPT3's Base Address
0000 : 040E	LPT4's Base Address

2.11 การสื่อสารข้อมูลแบบอนุกรม

การติดต่อสื่อสารผ่านพอร์ตอนุกรมนั้น มีข้อยุ่งยากมากกว่าการใช้พอร์ตขนาน สำหรับอุปกรณ์ทั่ว ๆ ไปแล้วจะต้องการข้อมูลในการประมวลแบบขนาน ดังนั้นจึงเพิ่มอุปกรณ์ที่ทำหน้าที่ในการแปลงข้อมูลแบบอนุกรมมาเป็นข้อมูลแบบขนาน และซอฟแวร์ที่ใช้ควบคุมการทำงานก็จะยุ่งยากกว่าการควบคุมผ่านพอร์ตมาตราฐานแบบขนาน แต่การสื่อสารโดยใช้พอร์ตอนุกรมนั้นก็มีข้อดีอยู่ไม่น้อยเช่นกัน อาทิเช่น

1. การติดต่อสื่อสารผ่านพอร์ตอนุกรมสามารถใช้สายได้ยาวกว่าการติดต่อสื่อสารผ่านพอร์ตขนาน โดยที่พอร์ตอนุกรมจะใช้ระดับแรงดันในช่วง –3 ถึง –25 V แทนลอจิก “1” และใช้แรงดันในช่วง +3 ถึง +25 V แทนลอจิก “0” ดังนั้นจะเป็นว่าช่วงการสวิงแรงดันของพอร์ตอนุกรมจะมีค่าประมาณ 50 V ส่วนพอร์ตขนานจะมีช่วงสวิง 5V เท่านั้น ซึ่งจะเห็นได้อย่างชัดเจนว่าหากมีการสูญเสียในสายแล้ว การสื่อสารผ่านพอร์ตอนุกรมจะสามารถส่งข้อมูลไปได้ไกลกว่าอย่างแน่นอน

2. ใช้จำนวนสายน้อยกว่าการส่งข้อมูลแบบขนาน ในกรณีที่อุปกรณ์อยู่ห่างจากคอมพิวเตอร์มาก ๆ ย่อมจะสะดวกและประหยัดกว่าหากจะเดินสายเพียง 3 เส้น ซึ่งเป็นลักษณะโครงสร้างของนัลโมเด็ม (Null Modem) เมื่อเทียบกับการเดินสายจำนวน 19 หรือ 25 เส้นในการใช้พอร์ตขนาน

3. ปัจจุบันอุปกรณ์ที่ใช้แสงอินฟาเรด ได้รับความนิยมมากขึ้น ซึ่งจะเห็นได้จากอุปกรณ์ประเภทสมุดบันทึกอิเล็กทรอนิกส์ คอมพิวเตอร์แบบโน้ตบุ๊ก ฯลฯ จะมีการติดต่อสื่อสารโดยใช้อินฟาเรดร่วมอยู่ด้วยและแน่นอนว่าการใช้อินฟาเรดก็จะต้องใช้การติดต่อสื่อสารแบบอนุกรม เนื่องจากความไม่สะดวกอย่างยิ่งในการที่จะส่งข้อมูลแบบขนานด้วยอินฟาเรด

2.12 RS-232 มาตรฐานเพื่อการสื่อสาร

มาตรฐาน RS-232 ประกาศและได้กำหนดการอินเตอร์เฟสระหว่างอุปกรณ์เทอร์มินัลของข้อมูล (Data Terminal Equipment, DTE) กับอุปกรณ์ส่วนปลายของวงจรข้อมูล (Data Circuit Terminating Equipment, DCT) อุปกรณ์ DTE ตามปกติแล้วจะเป็นอุปกรณ์เทอร์มินัลที่พูดไม่ได้ (dumb terminal) เป็นอุปกรณ์ที่ความฉลาด เช่น ไมโครคอนโทรลเลอร์หรือเครื่องคอมพิวเตอร์ส่วนบุคคลที่ความฉลาดในการสร้างบิตข้อมูลเป็นแบบอนุกรมได้ อุปกรณ์ DCE จะรับบิตข้อมูลที่อุปกรณ์ DTE ส่งออกมา ผ่านทางการอินเตอร์เฟสแบบ RS-232 และแปลงให้อยู่ในรูปที่เหมาะสมสำหรับถ่ายทอดผ่านตัวกลางการสื่อสารจากระยะไกล เช่น ผ่านทางสายโทรศัพท์

ถ้าคิดตามมาตรฐาน RS-232 ต่อไปจะพบว่าในทางกายภาพแล้วขั้วต่อพอร์ตของอุปกรณ์ DTE จะเป็นข้อต่อตัวผู้และขั้วต่อพอร์ตของอุปกรณ์ DCE จะเป็นข้อต่อตัวเมีย พอร์ตอนุกรมของคอมพิวเตอร์ส่วนบุคคลปกติแล้วจะเป็นอุปกรณ์ DTE และพอร์ตของโมเด็มก็มักจะมีโครงสร้างเป็นอุปกรณ์ DCE

พอร์ตอนุกรมของเครื่องพีซีจะเป็นตัวผู้ไม่ว่าจะเป็นขั้วต่อแบบ 9 ขาหรือ 25 ขา ส่วนข้อต่อของโมเด็มส่วนมากแล้วจะเป็นขั้วต่อแบบตัวเมีย 25 ขา แม้ว่าขั้วต่อแบบ 9 ขาจะไม่ได้เป็นมาตรฐาน RS-232 ของ EIA แต่ในปัจจุบันนี้ก็มีการใช้อยู่ทั่วไป ในการอินเตอร์เฟสแบบ RS-232 การอินเตอร์เฟสที่ใช้ขั้วต่อแบบ 9 ขา พบจำหน่ายเป็นครั้งแรกในเครื่องพีซีรุ่น AT ในตอนต้นทศวรรษ 1980

มาตรฐาน RS-232 ยังกำหนดคุณลักษณะสำหรับการสื่อสารผ่านทางสายส่งสัญญาณเสียงด้วยอัตราความเร็วสูงถึง 9600 bps ซึ่งช่วยให้สามารถมีการสื่อสารแบบอะซิงโครนัสจากจุดหนึ่งไปยังอีกจุดหนึ่งผ่านทางเครือข่ายโทรศัพท์สาธารณะในขณะนั้นได้

มาตรฐาน RS-232 กำหนดวงจร 21 วงจรในการอินเตอร์เฟสนี้ ตาราง 1 คือ สรุปคำบรรยายหน้าที่ของขาในมาตรฐาน RS-232 A, B และ C

ในระหว่างการส่งข้อมูล สภาวะมีข้อมูล (mark condition) บ่งชี้ด้วยสถานะ “1” ในระบบเลขฐานสอง และสภาวะช่องว่าง (space condition) บ่งชี้ด้วยสถานะ “0” ในระบบเลขฐานสอง

สำหรับวงจรกำหนดจังหวะเวลาและควบคุมการแลกเปลี่ยนข้อมูลแล้วการทำงานของมันจะเป็น “on” เมื่อแรงดันไฟฟ้าเป็นบวกมากกว่า +3 โวลท์ และจะเป็น “off” เมื่อแรงดันไฟฟ้าเป็นลบมากกว่า –3 โวลท์ เมื่อเทียบกับกราวน์ การทำงานนี้จะไม่สามารถกำหนดได้ถ้าแรงดันไฟฟ้าอยู่ในช่วงของการเปลี่ยนแปลงระหว่าง –3 ถึง +3 โวลท์ เมื่อเทียบกราวน์

คำว่า “mark” และ “space” จะใช้อยู่ทั่วไปในเอกสารกำหนดคุณลักษณะ เพื่อบรรยายสภาวะของสายนำข้อมูลหรือสายสัญญาณควบคุมในระบบ RS-232

ตารางที่ 2.7 แสดงรายการคุณลักษณะเฉพาะสำหรับการอินเตอร์เฟสแบบ RS-232

หมายเลขขาสัญญาณ	ชื่อของสายสัญญาณ	ทิศทางของสัญญาณ
1	Positive Ground	N.A.
2	Transmitted Data	To DCE
3	Received Data	To DCE
4	Request To Send	To DCE
5	Clear To Send	To DCE
6	Data Set Ready	To DCE
7	Signal Ground	N.A.
8	Received Line Signal Detector (RS-232C) Data Carrier Detect (RS-232)	To DCE
11	Select Standby	To DCE
12	Secondary Receive Line Signal Detector	To DTE

หมายเลขขาสัญญาณ	ชื่อของสายสัญญาณ	ทิศทางของสัญญาณ
13	Secondary Clear To Send	To DTE
14	Secondary Transmitted Data	To DCE
14	New Sync	To DCE
15	Transmitted Signal Element Timing	To DTE
16	Secondary Received Data	To DTE
17	Receiver Signal Element Timing	To DTE
18	Test	To DCE
19	Secondary Request To Send	To DCE
20	Data Terminal Ready	To DCE
21	Signal quality Detector	To DTE
22	Ring/Calling Indicator	To DTE
23	Data Signal Rate Selector	To DCE
23	Data Signal Rate Selector	To DTE
24	Transmitter Signal Element Timing	To DCE

ปัจจุบันนี้อุปกรณ์ส่วนใหญ่ที่มีวางจำหน่ายอยู่จะเป็นไปตามมาตรฐาน RS-232 หรือ RS-232D (มาตรฐาน CCITT V.24 และ V.28 ก็ยังคงมีใช้อยู่ทั่วไปอย่างกว้างขวาง) วงจรของ RS-232 ส่วนมากไม่ได้ใช้ในการติดต่อสื่อสารระหว่างเครื่องเทอร์มินัล 2 เครื่องหรือเครื่องคอมพิวเตอร์ 2 เครื่องโดยตรง

2.13 คำบรรยายลักษณะวงจร RS-232

สิ่งแรกที่คุณอาจสังเกตเห็นก็คือ ตาราง 3 จะแสดงสายสัญญาณเพียง 11 เส้น จาก 25 เส้นที่เป็นไปได้ของระบบ RS-232 ที่ต้องการใช้ในการทำการสื่อสารระหว่าง DTE ไปยัง DCE ให้สมบูรณ์ส่วนมากแล้วคุณสามารถจะทิ้งสายวงจรตัวตรวจจับอัตราสัญญาณข้อมูล (Data Signal Rate) และสายวงจรกราวด์ (Protective Ground) ออกไปได้ ทำให้เหลือสายสัญญาณที่ต้องต่อเพียง 9 เส้น

RS-232 เป็นข้อกำหนดการอินเตอร์เฟสมาตรฐาน และสามารถใช้เพื่อจุดประสงค์อื่น ๆ ต่าง ๆ กันไป เช่น การสื่อสารแบบซิงโครนัส (Synchronous Communication) และรูปแบบการสื่อสารสามารถทำให้มีการสนทนากันจาก DTE ไปยัง DCE โดยใช้สายสัญญาณเพียง 3 เส้น จากจำนวน 11 เส้นที่แสดงในตารางที่ 3 ถ้าอุปกรณ์ DTE และ DCE ใช้ซอฟต์แวร์ที่เขียนขึ้นตามความ

ต้องการของลูกค้า (Custom-Write Software) ก็จะใช้เพียงสาย TD, RD และสายกราวด์สัญญาณเท่านั้นในการย้ายข้อมูลไปตามสายตัวนำ 3 เส้นนี้

ตารางที่ 2.8 แสดงรายละเอียดสำหรับการอินเตอร์เฟส RS-232 โดยใช้คอนเน็กเตอร์แบบ DB-25

หมายเลขขาสัญญาณ	ชื่อของขาสัญญาณ
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request To Send
5	Clear To Send
6	Data Set Ready
7	Signal Common
8	Received line Signal Detect
9	Reserved For Testing
10	Reserved For Testing
11	Unassigned
12	Secondary Received Line Signal Detector
13	Secondary Clear To Send
14	Secondary Transmitted Data
15	Transmission Signal Element Timing
16	Secondary Received Data
17	Transmission Signal Element
18	Unassigned
19	Secondary Request to Send
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicator
23	Data Signal Rate
24	Transmitter Signal Element Timing
25	Unassigned

2.14 ข้อกำหนดของขาสัญญาณ 11 ขา

1. ขา 1 (Protective Ground Circuit, AA) ขานี้จะต่อเข้ากับตัวถังของอุปกรณ์ และสามารถต่อเข้ากับกราวด์ภายนอกถ้าอุปกรณ์อื่น ๆ ต้องใช้ขานี้

2. ขา 2 (Transmitted Data Circuit, TD) เป็นขาสัญญาณข้อมูลที่ออกมาจากอุปกรณ์ DTE กระแสบิตข้อมูลอนุกรมจากขานี้ คือ ข้อมูลที่จะถูกถอดรหัสโดยอุปกรณ์ DCE

3. ขา 3 (Received Data Circuit BB, RD) สัญญาณที่ขานี้จะถูกสร้างจากอุปกรณ์ DCE กระแสบิตข้อมูลอนุกรมนี้จะกำเนิดขึ้นที่อุปกรณ์ DTE ปลายทาง และเป็นผลผลิตของวงจรนับข้อมูลของอุปกรณ์ DCE สัญญาณนี้มักจะเป็นข้อมูลดิจิตอลที่ถูกสร้างขึ้นโดยอุปกรณ์ DCE ที่มีความฉลาดหรือจากวงจรถอดสัญญาณ (demodulation) ของโมเด็ม

4. ขา 4 (Repeat to Send Circuit CA, RTS) สัญญาณนี้จะเตรียมพร้อมอุปกรณ์ DCE สำหรับการทำการส่งข้อมูล เมื่อสัญญาณ RTS นี้อยู่ในสภาวะ "ON" จะทำให้อุปกรณ์ DCE อยู่ในโหมดส่งข้อมูล (transmit mode) ในขณะที่สัญญาณนี้อยู่ในสภาวะ "OFF" จะทำให้อุปกรณ์ DCE อยู่ในโหมดรับข้อมูล (receive mode) อุปกรณ์ DCE ควรจะตอบสนองต่อสัญญาณ RTS ON โดยการทำให้สัญญาณ Clear to Send (CTS) อยู่ในสภาวะ "ON" ด้วย เมื่อสัญญาณ RTS อยู่ในสภาวะ "OFF" สัญญาณไม่ควรจะ "ON" ขึ้นอีก จนกว่าสัญญาณ CTS จะอยู่ในสภาวะ "OFF" เสียก่อน สัญญาณนี้จะถูกใช้ร่วมกับสัญญาณ DTR, DSR และ DCD ขาสัญญาณ RTS จะถูกใช้อย่างมากในการควบคุมการไหลของข้อมูล

5. ขา 5 (Clear to Send Circuit CB, CTS) สัญญาณนี้จะตอบรับกลับไปยังอุปกรณ์ DTE เมื่อได้รับสัญญาณ RTS และข้อมูลสามารถส่งออกไปได้ ข้อมูลจะถูกส่งไปตามตัวกลางที่ใช้สื่อสารได้ก็ต่อเมื่อสัญญาณ CTS นี้อยู่ในสภาวะ "ON" เท่านั้น สัญญาณนี้จะใช้ร่วมกับขา DTR, DSR และ DCD ขาสัญญาณ CTS นี้จะใช้ร่วมกับขา RTS สำหรับควบคุมการไหลของข้อมูล

6. ขา 6 (Data Set Ready Circuit CC, DSR) สัญญาณจะบอกต่ออุปกรณ์ DTE ว่าอุปกรณ์ DCE ได้ต่อกับตัวกลางสื่อสารที่ถูกต้องแล้วและในบางกรณีจะบ่งชี้ว่าสายโทรศัพท์อยู่ในสภาวะ "OFF HOOK" สภาวะ "OFF HOOK" นี้จะเป็นตัวบ่งชี้ว่าอุปกรณ์ DCE กำลังอยู่ในโหมด dialing' หรือกำลังติดต่อกับอุปกรณ์ DCE อีกตัวหนึ่งอยู่ เมื่อสัญญาณ DSR นี้อยู่ในสภาวะ "OFF" อุปกรณ์ DTE ก็ควรจะถูกกำหนดให้ไม่สนใจสัญญาณอื่น ๆ ทั้งหมดจากอุปกรณ์ DCE ถ้าสัญญาณนี้ถูกทำให้อยู่ในสภาวะ "OFF" ก่อนอุปกรณ์ DTR แล้วอุปกรณ์ DTE ก็จะสรุปว่าการสื่อสารกันนั้นสิ้นสุดลง

7. ขา 7 (Signal Common Circuit, AB) สายตัวนี้จะให้สัญญาณอ้างอิงของกราวน์ร่วมกันสำหรับวงจรการแลกเปลี่ยนข้อมูลทั้งหมด ยกเว้นวงจร AA หรือ protective ground ข้อกำหนด RS-232 จะอนุญาตให้วงจรนี้ถูกทำให้อยู่ในสภาวะ "OFF" ก่อนอุปกรณ์ DTR แล้ว อุปกรณ์ DTE ก็จะสรุปว่าการสื่อสารกันนั้นสิ้นสุดลง

8. ขา 8 (Data Carrier Detect Circuit CF, DCD) ขานี้ยังรู้จักกันในนามของ Received Line Signal Detect (RLSD) หรือขา Carrier Detect (CD) สัญญาณนี้จะแอกทีฟเมื่อเกิดสัญญาณพาหะที่เหมาะสมระหว่างอุปกรณ์ DCE ที่สถานีหับที่อยู่ในระยะไกลเมื่อสัญญาณนี้อยู่ในสภาวะ "OFF" สัญญาณที่ขา RD ควรจะถูกทำให้ค้างอยู่ในสภาวะ "Mark" (สถานะ "1" ในเลขฐานสอง)

9. ขา 20 (Data Terminal Ready Circuit CD, DTR) สัญญาณ DTR ถูกควบในการสวิทซ์อุปกรณ์ DCE เข้ากับตัวกลางในการสื่อสารสัญญาณ DTR ON บ่งชี้ว่าอุปกรณ์ DCE ที่กำลังต่อเชื่อมกันอยู่ ก็ยังคงต่อเชื่อมกัน และถ้าไม่มีการต่อเชื่อมกันก็สามารถทำการต่อเชื่อมกับครั้งใหม่ได้ ปกติแล้วสัญญาณ DTR อยู่ในสภาวะ "OFF" เพื่อกระตุ้นให้เกิดสภาวะ ON HOOK (วางสาย) (hang up) อุปกรณ์ DCE โดยปกติแล้วจะตอบสนองต่อการกระตุ้นจากสัญญาณ DTR โดยการทำให้สัญญาณ DSR แอกทีฟ

10. ขา 22 (Ring Indicator Circuit CE, RI) สภาวะ "ON" ของสัญญาณนี้จะบ่งชี้ได้ว่ารับสัญญาณเรียกสายโทรศัพท์จากตัวกลางในการสื่อสาร (สายโทรศัพท์) ปกติแล้วจะขึ้นอยู่กับโปรแกรมควบคุมในการที่จะทำให้เกิดสัญญาณนี้ขึ้นหรือไม่

11. ขา 23 (Data Signal Rate Detector Circuit CH/CI, DSRD) วงจร CH เป็นส่วนประกอบของ DTE และวงจร CI ปีนส่วนประกอบของ DCE สัญญาณที่ขานี้ถูกใช้ในการเลือกค่าอัตราการส่งสัญญาณข้อมูลค่าใดค่าหนึ่งในสองค่าในกรณีที่ใช้โมเด็มที่มีอัตราการส่งข้อมูลได้ 2 ค่า (dual-rate modem) ถ้าสัญญาณที่ขานี้เป็น "ON" ก็จะเป็นการเลือกอัตราการส่งข้อมูลที่มีค่าสูงที่สุดใน 2 ค่านั้น

ตารางที่ 2.9 แสดงขาสัญญาณที่ต่อจาก DTE ไปยัง DCE โดยใช้คอนเน็กเตอร์แบบ DB-25

หมายเลขขาสัญญาณ	ชื่อของสายสัญญาณ
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request To Send
5	Clear To Send
6	Data Set Ready
7	Signal Common
8	Data Carrier Detect
20	Data Terminal Ready
22	Ring Indicator
23	Data Signal Rate Detector

ตารางที่ 2.10 การต่อแบบคอนเน็กเตอร์แบบ DB-9 ตามมาตรฐาน RS-232

หมายเลขขาสัญญาณ	ชื่อของสายสัญญาณ
1	Data Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Common
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator

2.15 ขั้นตอนการติดต่อระหว่างอุปกรณ์ DTE และ DCE

1. การจ่ายกำลังงานให้อุปกรณ์ DTE และอุปกรณ์นี้ส่งสัญญาณ DTR ออกมา
2. อุปกรณ์ DCE ถูกเปิดขึ้น และรับรู้ถึงสัญญาณ DTR ที่ส่งมาจากอุปกรณ์ DTE
3. อุปกรณ์ DCE ส่งสัญญาณ DSR ออกมา และโมเด็มก็กระทำกระบวนการ OFF HOOK
4. ถ้าสายสัญญาณโทรศัพท์อยู่ในสภาพดีและปลายทางอีกด้านหนึ่งก็พร้อมจะรับข้อมูลแล้วโดยจะตรวจจับพบสัญญาณพาหะแล้วอุปกรณ์ DCE จะส่งสัญญาณ DCD ออกมา
5. อุปกรณ์ DTE ยกระดับสัญญาณ RTS ขึ้นสูง
6. อุปกรณ์ DCE จะสนองด้วยการส่งสัญญาณ CTS ออกมา
7. การติดต่อสื่อสารก็เริ่มขึ้น โปรแกรมควบคุมจะทำการส่งหรือรับข้อมูล ส่วนลำดับขั้นในการตอบรับก็จะเป็นในทำนองนี้

2.16 ลักษณะทางฮาร์ดแวร์

อุปกรณ์ที่ใช้การสื่อสารแบบอนุกรมสามารถแยกออกได้เป็น 2 ประเภท คือ DCE (Data Communication Equipment) อุปกรณ์เหล่านี้ได้แก่ โมเด็ม ฯลฯ และ DTE (Data Terminal Equipment) ซึ่งก็คือคอมพิวเตอร์นั่นเอง

ข้อกำหนดทางไฟฟ้าของพอร์ตอนุกรมได้ถูกกำหนดเป็นมาตรฐานโดย EIA (Electronics Industry Association) หรือ RS-232 ซึ่งประกอบไปด้วยสิ่งต่าง ๆ เหล่านี้

1. ช่วงไม่มีข้อมูล (space) หรือลอจิก “0” ต้องมีแรงดันอยู่ในช่วง +3 ถึง –25 V
2. ช่วงมีข้อมูล (mark) หรือลอจิก “1” ต้องมีแรงดันอยู่ในช่วง +3 ถึง +25 V

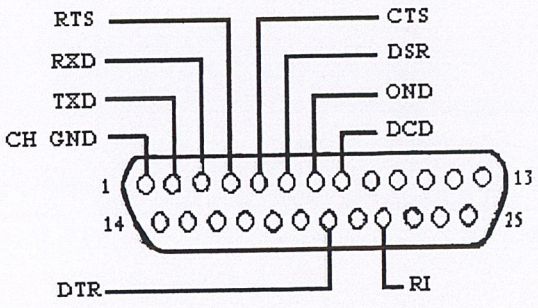
3. แรงดันในช่วง –3 ถึง +3 V ไม่มีการนิยามไว้
4. แรงดันในขณะเปิดวงจรต้องมีค่าไม่เกิน 25 V
5. กระแสขณะช็อตวงจรมีค่าไม่เกิน 500mA

ข้อมูลเหล่านี้ยังไม่ใช่ข้อกำหนดที่ครอบคลุมมาตรฐานของ RS-232 ทั้งหมดมาตรฐานของ RS-232 นั้น นอกจากจะมีคุณสมบัติดังที่ได้กล่าวมาแล้วยังจะต้องประกอบด้วยค่าคาปาซิเตนซ์ของสาย อัตราบอดสูงสุด ฯลฯ ด้วย

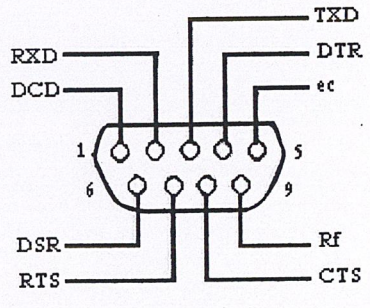
พอร์ตอนุกรมนี้จะมีอยู่ด้วยกัน 2 ขนาด คือ คอนเน็กเตอร์แบบ D-type ตัวผู้ขนาด 25 pin รูปที่ 2.23 (ก) และคอนเน็กเตอร์แบบ D-type ตัวผู้เช่นกันขนาน 9 pin รูปที่ 2.23 (ข) ซึ่งคอนเน็กเตอร์ทั้ง 2 แบบนี้จะติดอยู่ที่ด้านหลังของคอมพิวเตอร์ ตารางที่ 1 แสดงตำแหน่งขาสัญญาณของพอร์ตอนุกรม

ตารางที่ 2.11 ตำแหน่งขาสัญญาณของพอร์ตอนุกรม

D-type 25 Pin No.	D-type 9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request to Send
Pin 5	Pin 8	CTS	Clear to Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator



รูปที่ 2.23 (ก) คอนเน็คเตอร์แบบ DB-25



รูปที่ 2.23 (ข) คอนเน็คเตอร์ แบบ DB-9

2.17 ตำแหน่งพอร์ตอนุกรม

ตำแหน่งของพอร์ตและ IRQ ของพอร์ตอนุกรมนั้นมีลักษณะการกำหนดและการอ่านตำแหน่งจาก BIOS คล้ายกับของพอร์ตขนาน ในตารางที่ 3 เป็นการแสดงตำแหน่งมาตรฐานของพอร์ตอนุกรมและ IRQ ที่ใช้สำหรับตำแหน่งของพอร์ตแต่ละเบอร์

ตารางที่ 2.12 ตำแหน่งมาตรฐานของพอร์ตอนุกรม

Name	Address	IRQ
Com 1	3F8	4
Com 2	2F8	3
Com 3	3E8	4
Com 4	2E8	3

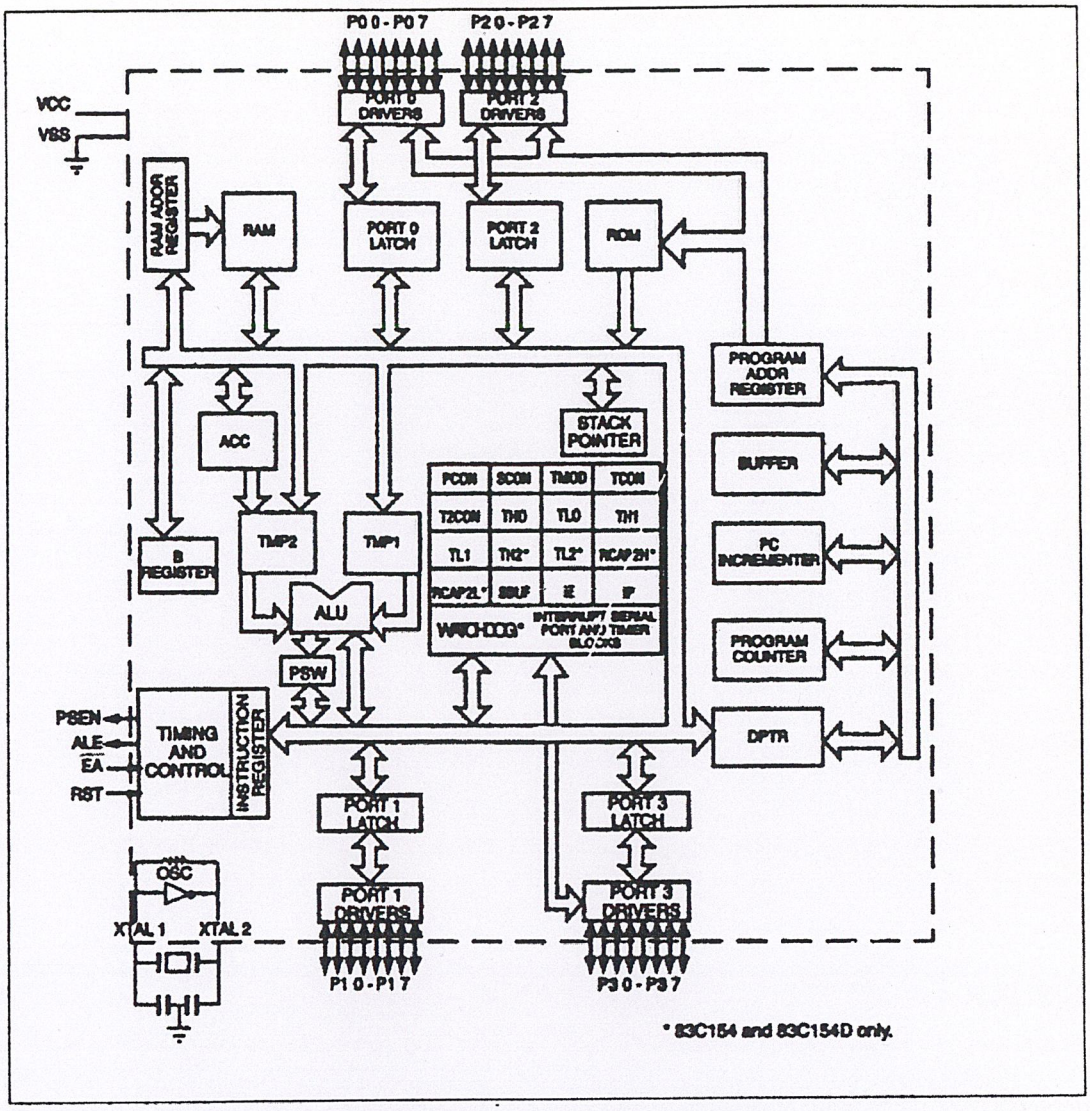
2.18 ไมโครคอนโทรลเลอร์ (MCS-51)

1. ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ MCS-51

ปัจจุบันมีไมโครคอนโทรลเลอร์ MCS-51 ซึ่งเป็นไมโครคอมพิวเตอร์แบบชิพเดี่ยว (ไม่ต้องต่อกับอุปกรณ์ภายนอกก็สามารถทำงานได้) มีความสะดวกในการใช้งานและเขียนโปรแกรมควบคุมด้วยภาษาเบสิกได้โดยไม่ต้องศึกษาการทำงานของวงจรเหมือนกับภาษาแอสเซมบลี้ หรือบางท่านที่ถนัดภาษาแอสเซมบลี้ก็ยังสามารถใช้ได้เช่นเดียวกัน

ไมโครคอนโทรลเลอร์ MCS-51 เป็นอุปกรณ์ที่ออกแบบเพื่อสนองความต้องการของผู้ใช้โดยมีสายอินพุทและเอ้าท์พุตภายในตัวเอง พอร์ทของอินพุทและเอ้าท์พุทบัฟเฟอร์อินเตอร์เฟสและสายควบคุมอื่น ๆ ใช้สำหรับแยกข้อมูลกับแอดเดรสและยังมีชุดคำสั่งเพิ่มขึ้นเป็นพิเศษ เพื่อจัดการข้อมูลแถมท้ายด้วยวงจรตั้งเวลากับวงจรนับด้วย

2. โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51



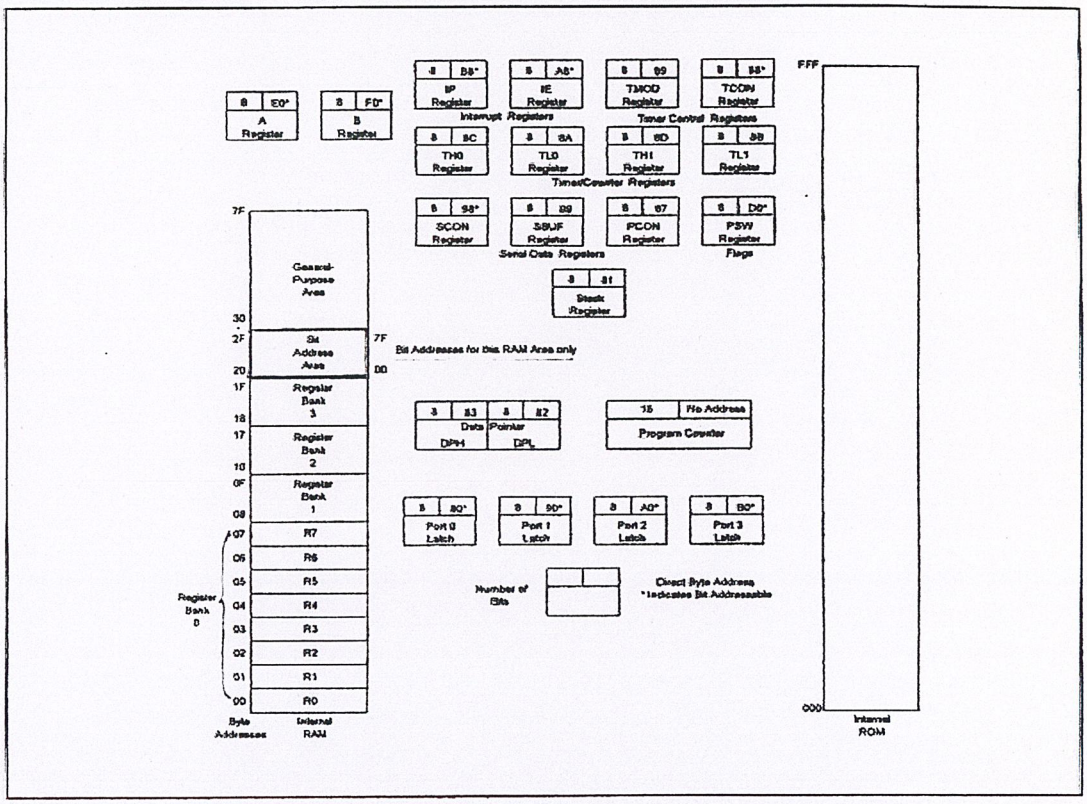
รูปที่ 2.24 โครงสร้างภายในชิพไมโครคอนโทรลเลอร์ MCS-51

3. การจัดการหน่วยความจำของไมโครคอนโทรลเลอร์ เบอร์ 8051

หน่วยความจำของ 8051 แบ่งออกไว้เป็น 2 แบบตามลักษณะของการใช้งาน คือ

3.1 **ความจำโปรแกรม (Program Menory)** เป็นหน่วยความจำที่ใช้งาน คือ ภาษาเครื่อง (Machine Language) ซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็จะอ่านข้อมูลที่เก็บคำสั่งในหน่วยความจำประเภทนี้เข้าไปถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่น ๆ ตามการทำงานของแต่ละคำสั่งนั้น หน่วยความจำแบบนี้จะต้องเป็นแบบหน่วยความจำอ่านได้เท่านั้น (Read Only Memory (ROM)) และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ (หน่วยความจำแบบหน่วยความจำแบบอ่านได้เท่านั้น ซึ่งเมื่อปิดไปแล้วข้อมูลก็ไม่มีการสูญหาย) ในระหว่างการทำงานของ 8051 ผู้ใช้จะไม่สามารถใช้คำสั่ง

ทำการเขียนข้อมูลลงในหน่วยความจำแบบนี้ได้ จำนวนตำแหน่งสูงสุดของหน่วยความจำแบบนี้ที่ 8051 จะใช้งานได้คือ 65536 ตำแหน่ง ตำแหน่ง 0000H ถึง 0FFFH จำนวน 4 กิโลไบท์ ถ้าต้องการให้ 8051 ทำงานตามคำสั่งที่เก็บไว้ใน ROM ภายใน 8051 ก็ให้ป้อนสัญญาณสภาวะลอจิกสูง (1) เข้าที่ขา EA ของ 8051 แต่ถ้าต้องการให้ทำงานในโปรแกรมที่เก็บไว้ใน ROM ภายนอก 8051 ก็ให้ต่อลอจิกต่ำ (0) เข้าที่ขา/EA ของ 8051 ส่วนหน่วยความจำที่ตำแหน่ง 1FFFH ถึง FFFFH จะต้องต่ออยู่ภายนอก 8051 เสมอดังแสดงในแผนภูมิหน่วยความจำ (Memory Map) หน่วยความจำภายใน (Internal Memory) หมายถึง หน่วยความจำนั้นอยู่ภายใน 8051 ส่วนหน่วยความจำภายนอก (External Memory) หมายถึง หน่วยความจำนั้นอยู่ภายนอก 8051



รูปที่ 2.27 ตำแหน่งต่าง ๆ ของรีจีสเตอร์และหน่วยความจำ

3.2 **หน่วยความจำของข้อมูล (Data Memory)** เป็นหน่วยความจำที่ไมโครคอนโทรลเลอร์เบอร์ 8051 จะใช้สำหรับพัก เก็บข้อมูล แล้วเรียกมาใช้ใหม่ในระหว่างการทำงานของไมโครคอนโทรลเลอร์เบอร์ 8051 การอ่านหรือเขียนจากหหน่วยความจำจะกระทำโดยคำสั่งที่เก็บไว้ในหน่วยความจำของโปรแกรม หน่วยความจำแบบนี้เป็นประเภท หน่วยความจำอ่านและเขียนได้ (Random Access Memory (RAM)) ถ้ามีไฟเลี้ยงอยู่ข้อมูลที่เก็บไว้จะไม่สูญหาย แต่ถ้าปิดเครื่องหรือไม่จ่ายไฟให้แก่หน่วยความจำอ่านและเขียนได้แล้ว ข้อมูลในหน่วยความจำอ่านและเขียนได้ก็จะสูญหายไป หน่วยความจำของข้อมูลของไมโครคอนโทรลเลอร์ 8051 จะอยู่ 2 ชุด ชุด

หนึ่งอยู่ภายใน 8051 จำนวน 128 ไบท์ที่ตำแหน่ง 00H ถึง 7FH และอีกชุดหนึ่งจะต้องต่ออยู่ภายนอกของวงจรรวม 8051 มีได้สูงสุด 65536 ไบท์ (64 กิโลไบท์) อยู่ที่ตำแหน่ง 0000H ถึง FFFFH หน่วยความจำของข้อมูลภายใน 8051 ที่ตำแหน่ง 08H ถึง FFH นั้นไม่ได้มีอยู่ทุกตำแหน่ง จะมีเฉพาะในบางตำแหน่ง ซึ่งเรียกหน่วยความจำบางตำแหน่งนี้ว่า รีจีสเตอร์พิเศษ (Special Function Register (SER)) เพราะจะใช้หน่วยความจำเหล่านี้สำหรับงานพิเศษเท่านั้น

4. รีจีสเตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 มีรีจีสเตอร์ที่อำนวยความสะดวกในการใช้งานตามคำสั่งต่าง ๆ ประกอบด้วยแอคคิวมูเลเตอร์ รีจีสเตอร์ B ที่ใช้ในการคูณและหาร รีจีสเตอร์สถานะ สแต็กพอยน์เตอร์ ข้อมูลพอยน์เตอร์ (2*8 บิต หรือ 1*16 บิต) พอร์ทหมายเลขศูนย์ถึงพอร์ทหมายเลขสาม รีจีสเตอร์แบบคู่ ซึ่งใช้ส่งและรับข้อมูลชนิดอนุกรม รีจีสเตอร์ 16 บิท ที่เป็นวงจรตั้งเวลาและวงจรนับที่ 3 รีจีสเตอร์ คำสั่งสำหรับหน้าที่พิเศษ (เช่น การอินเตอร์รัพท์ RTC : Read Time Clock) และอินพุท เอ้าท์พุทแบบอนุกรม

5. ไทม์เมอร์/ เคาน์เตอร์

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 มีรีจีสเตอร์พิเศษที่สามารถเลือกใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง (นับจำนวนแมชชีนไซเคิลหรือนับจำนวนพัลส์ที่เกิดขึ้นภายนอกชิพ) รีจีสเตอร์แระเภทนี้มีอยู่ด้วยกัน 2 ตัว แต่ละตัวมีขนาด 16 บิต โดยมีชื่อเรียกว่าไทม์เมอร์ 0 และไทม์เมอร์ 1 ตามลำดับ รีจีสเตอร์ที่ใช้เป็นไทม์เมอร์ 0 ประกอบขึ้นจากรีจีสเตอร์ใช้งานเฉพาะ TL0, TH0 ส่วนรีจีสเตอร์ที่ใช้เป็นไทม์เมอร์ 1 ประกอบขึ้นจากรีจีสเตอร์ใช้งานเฉพาะ TL1, TH1 รีจีสเตอร์ไทม์เมอร์ 0 และไทม์เมอร์ 1 สามารถกำหนดการใช้งานเป็นไทม์เมอร์หรือเคานเตอร์อย่างใดอย่างหนึ่งได้ ในการทำงานเป็นไทม์เมอร์หรือเคาน์เตอร์จะมีรายละเอียดที่แตกต่างกันออกไปดังนี้

5.1 ไทม์เมอร์

ค่าในรีจีสเตอร์ที่ใช้เป็นไทม์เมร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทุก ๆ แมชชีนไซเคิล ดังนั้นจึงสามารถคิดว่าในขณะทำงานเป็นไทม์เมอร์หมายถึงใช้รีจีสเตอร์เป็นตัวนับจำนวนแมชชีนไซเคิลได้และเนื่องจากใน 1 แมชชีนไซเคิลใด ๆ ของไมโครคอนโทรลเลอร์ MCS-51 ประกอบไปด้วย 12 คาบสัญญาณออสซิลเลเตอร์ (Oscillator period) ดังนั้นอัตราเร็วในการนับ (Count rate) จึงมีค่าเป็น 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

5.2 เคาน์เตอร์

ค่าในรีจีสเตอร์ที่ใช้เป็นเคาน์เตอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทีละหนึ่ง เมื่อมีการเปลี่ยนสถานะซึ่งตรวจจับได้จากขา T0, T1 หรือ T2 (ใน 8052) ขึ้นกับรีจีสเตอร์ที่ถูกเลือกใช้งานเป็นเคาน์เตอร์ในขณะนั้น การตรวจสอบการเปลี่ยนสถานะจะตรวจเฉพาะในขณะที่สัญญาณมีการเปลี่ยนค่าจาก 1 เป็น 0 โดยมีรายละเอียดในการตรวจสอบสัญญาณดังนี้

ไมโครคอนโทรลเลอร์ MCS-51 จะตรวจสอบสถานะสัญญาณที่ขา T0, T1 หรือ T2 (ใน 8052) โดยการตรวจสอบจะเกิดขึ้นในระหว่างสเตท 5 เฟส 2 ของแต่ละแมชชีนไซเคิลรายละเอียดในการตรวจสอบการเปลี่ยนสถานะสัญญาณที่แต่ละขา จะมีลักษณะที่เหมือนกัน ดังนี้ เมื่อขา T0, T1 หรือ T2 มีสถานะสัญญาณเป็น 1 ในขณะที่สเตท 5 เฟส 2 ของแมชชีนไซเคิลใด ๆ และสเตท 5 เฟส 2 ของแมชชีนไซเคิลถัดไป หากสัญญาณที่ขา T0, T1 หรือ T2 มีค่าเปลี่ยนเป็น 0 จะทำให้รีจีสเตอร์ที่ถูกเลือกใช้งานเป็นเคาน์เตอร์ถูกเพิ่มค่าขึ้นอีก 1 ในช่วงสตท 3 เฟส 1 ของแมชชีนไซเคิล ซึ่งตรวจพบการเปลี่ยนสถานะของสัญญาณ ดังนั้นไมโครคอนโทรลเลอร์ MCS-51 จำเป็นจะต้องใช้เวลา 2 แมชชีนไซเคิล (24 คาบสัญญาณออสซิลเลเตอร์) เพื่อตรวจสอบการเปลี่ยนสถานะสัญญาณจาก 1 เป็น 0 ที่ขา T0, T1 หรือ T2 จึงทำให้อัตราการนับสูงสุดของเคาน์เตอร์ในไมโครคอนโทรลเลอร์ MCS-51 ถูกจำกัดความถี่ของสัญญาณไว้ที่ 1/24 ของความถี่ออสซิลเลเตอร์ที่ใช้ โดยไม่มีข้อจำกัดในเรื่อง dutycycle (อัตราส่วนของช่วงเวลาที่สัญญาณมีค่าเป็น 1 ต่อคาบเวลาของสัญญาณ) ของสัญญาณ แต่เพื่อให้มั่นใจว่าสถานะสัญญาณจะถูกตรวจสอบเข้ามาอย่างน้อย 1 ครั้งก่อนที่จะเปลี่ยนระดับจึงสมควรให้สถานะสัญญาณคงค่า 0 หรือ 1 อย่างน้อย 1 แมชชีนไซเคิลเต็ม

นอกจากจะสามารถเลือกการทำงานของรีจีสเตอร์ให้เป็นไทม์เมอร์หรือเคาน์เตอร์ได้แล้ว ในแต่ละการทำงานเป็นไทม์เมอร์หรือเคาน์เตอร์ของไทม์เมอร์ 0 หรือ ไทม์เมอร์ 1 ก็ยังมีการทำงานที่แยกย่อยลงไปอีกถึง 4 แบบ (โหมด 0, 1, 2 และ 3) ตามความเหมาะสมของการใช้งาน (ไทม์เมอร์ 0 และไทม์เมอร์ 1 มีให้เลือก 4 แบบ แต่ไทม์เมอร์ 2 มีให้เลือกเพียง 3 แบบ) การทำงานย่อยทั้ง 4 ประเภทของไทม์เมอร์ 0 และไทม์เมอร์ 1 มีรายละเอียดดังต่อไปนี้

5.3 ไทม์เมอร์ 0 และไทม์เมอร์ 1

ไทม์เมอร์ 0 และไทม์เมอร์ 1 ทั้ง 2 ตัวมีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์ผู้ใช้สามารถเลือกการทำงานให้เป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง โดยการกำหนดค่าบิต C/T ในรีจีสเตอร์ใช้งานเฉพาะ TMOD ดังแสดงในรูปที่ (ไทม์เมอร์ 0 ใช้บิท 2 ส่วน ไทม์เมอร์ 1 ใช้บิต 6) โดยหากบิตนี้มีค่าเป็น 0 หมายถึงเลือกให้รีจีสเตอร์ทำงานเป็นไทม์เมอร์ (นับจำนวนแมชชีนไซเคิล) ถ้าบิตนี้มีค่าเป็น 1 หมายถึงเลือกให้ทำงานเป็นเคาน์เตอร์ (นับจำนวนการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T0 หรือ T1)

รีจีสเตอร์ใช้งานเฉพาะ TMOD (Timer/ Counter Mode Control Register) ไม่สามารถเข้าถึงข้อมูลในระดับบิตได้

บิต GATE

บิตเลือกการควบคุมให้รีจีสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ทำงานโดยควบคุมจากฮาร์ดแวร์หรือซอฟต์แวร์ ดังนี้

(1) เมื่อบิต TRx (TR0, TR1) และ GATE ถูกเซต ไทม์เมอร์หรือเคาน์เตอร์จำทำงานต่อเมื่อสถานะที่ขา INTx (INT0, INT1) มีค่าเป็น 1 (ควบคุมจากฮาร์ดแวร์)

(2) เมื่อบิต GATE ถูกเคลียร์ไทม์เมอร์หรือเคาน์เตอร์จะทำงานก็ต่อเมื่อบิต TRx ถูกเซตโดยไม่ขึ้นกับสถานะสัญญาณที่ขา INTx (ควบคุมจากซอฟต์แวร์)

บิต C/T

บิตเลือกการทำงานของรีจีสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ ดังนี้

0 หมายถึง ทำงานเป็นไทม์เมอร์ (นับจำนวนแมชชีนไซเคิล)

1 หมายถึง ทำงานเป็นเคาน์เตอร์ (นับจำนวนพัลส์ภายนอกที่ขา Tx)

บิต M0 และ M1

M1 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือไทม์เมอร์ 1

M0 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือไทม์เมอร์ 1

0 0 โหมด 0 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 13 บิต

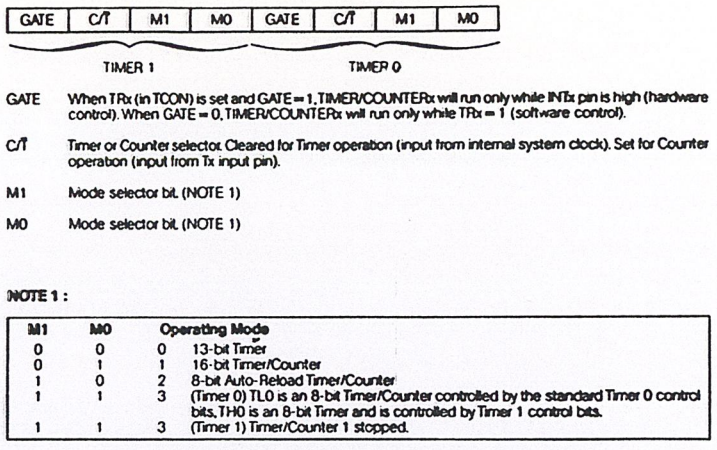
0 1 โหมด 1 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิต

1 0 โหมด 2 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 18 บิต ที่มีการโหมดค่าเองเมื่อเกิด โอเวอร์โฟลว์

1 1 โหมด 3 : ไทม์เมอร์ 0 รีจีสเตอร์ TL0 ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิตที่ควบคุมการทำงานได้จากบิตไทม์เมอร์ 0 รีจีสเตอร์ TH0 เป็นไทม์เมอร์ขนาด 8 บิต ที่ควบคุมการทำงานได้จากบิตของไทม์เมอร์ 1

1 1 โหมด 3 : ไทม์เมอร์ 1 หยุดทำงาน (หยุดนับ)

รีจีสเตอร์ที่ใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์หรือเคาน์เตอร์ทั้ง 2 ตัวสามารถทำงานแตกต่างกันออกไป 4 แบบคือ โหมด 0, 1, 2 และ 3 โดยการเปลี่ยนค่าบิต M0 และ M1 ในรีจีสเตอร์ใช้งานเฉพาะ TMOD เช่นเดียวกับบิต C/T การทำงานในโหมด 0, 1, 2 จะคล้าย ๆ กัน สำหรับไทม์เมอร์ 0 และไทม์เมอร์ 1 แต่ในโหมด 3 รีจีสเตอร์ที่ใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์ทั้งสองตัวจะมีการทำงานที่ต่างออกไปจาก 3 โหมดแรก



GATE	C/T	M1	M0	GATE	C/T	M1	M0

TIMER 1 TIMER 0

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/T Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit (NOTE 1)

M0 Mode selector bit (NOTE 1)

NOTE 1 :

M1	M0		Operating Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

รูปที่ 2.28 รีจีสเตอร์ใช้ในงานเฉพาะ TMOD

6. พอร์ทสื่อสารข้อมูลแบบอนุกรมไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 มีพอร์ตสำหรับสื่อสารข้อมูลแบบอนุกรม ที่สามารถรับและส่งข้อมูลแบบอนุกรม ได้โดยผู้ใช้ไม่จำเป็นต้องต่อชิพที่ทำหน้าที่รับหรือส่งข้อมูลแบบอนุกรมโดยเฉพาะเพิ่มเติมแต่อย่างใดเลย การนำไมโครคอนโทรลเลอร์ MCS-51 ไปประยุกต์ใช้งานที่ต้องมีการติดต่อสื่อสารข้อมูลแบบอนุกรมกับวงจรภายนอกอื่น ๆ จึงทำให้สะดวกแบบมีความคล่องตัวสูงมาก

พอร์ทสื่อสารข้อมูลแบบอนุกรมที่มีในไมโครคอนโทรลเลอร์ MCS-51 สามารถทำงานได้ในแบบ full duplex หมายความว่าไมโครคอนโทรลเลอร์ MCS-51 สามารถรับและส่งข้อมูลได้พร้อม ๆ กัน โดยในการรับข้อมูลจะมีการบัฟเฟอร์ข้อมูลให้ด้วย จึงทำให้ไมโครคอนโทรลเลอร์ MCS-51 สามารถกำหนดการรับข้อมูลไบท์ที่สองซึ่งถูกส่งตามเข้ามาก่อนที่ไบท์แรกที่ได้รับเข้ามาจะถูกอ่านจากรีจีสเตอร์ใช้งานเฉพาะที่ใช้สำหรับข้อมูล (receive register) เพื่อนำไปเก็บไว้ในหน่วยความจำต่อไป

พอร์ทสื่อสารข้อมูลแบบอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ประกอบด้วยรีจีสเตอร์ขนาด 8 บิต จำนวนสองตัว แต่ละตัวมีชื่อเรียกตามหน้าที่ ดังนี้

รีจีสเตอร์สำหรับข้อมูลใช้รับข้อมูลที่ส่งเข้ามาจากภายนอก

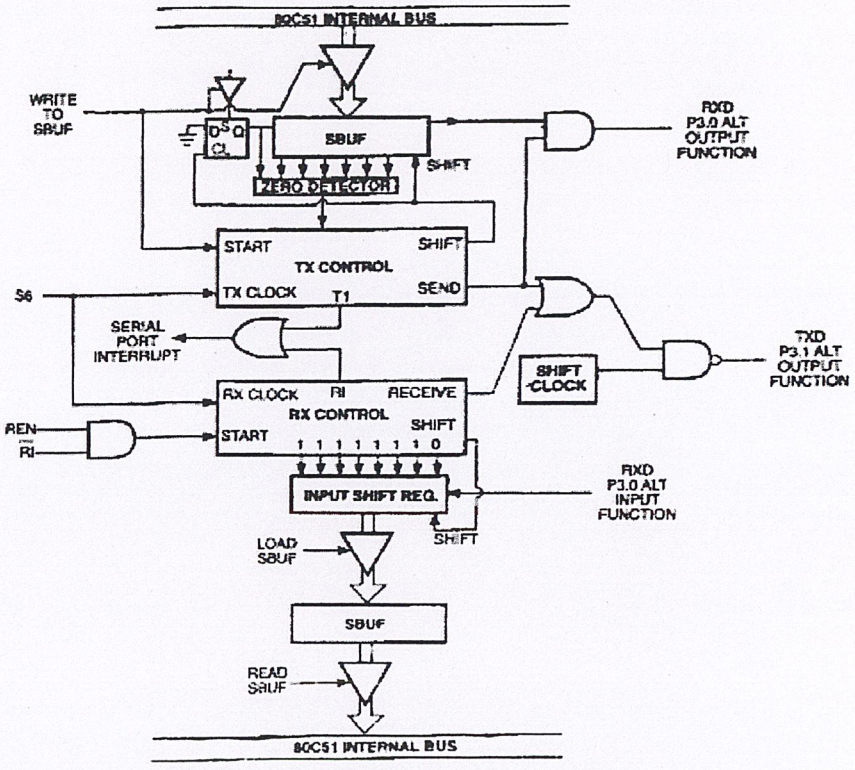
รีจีสเตอร์สำหรับส่งข้อมูล (transmit register) ใช้ส่งข้อมูลจากไมโครคอนโทรลเลอร์ MCS-51 ออกไปภายนอก

รีจีสเตอร์ทั้งสองตำแหน่งเดียวกันในรีจีสเตอร์ใช้งานเฉพาะ คือ ตรงกับตำแหน่งของรีจีสเตอร์ใช้งานเฉพาะ SBUF (ตำแหน่ง 99H) ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิพที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะ การเข้าถึงข้อมูลในรีจีสเตอร์แต่ละตัวไมโครคอนโทรลเลอร์ MCS-51 จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจีสเตอร์ตัวใดโดยตรวจสอบจากรหัสคำสั่ง ทั้งนี้เพราะในการ

เขียนข้อมูลไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF หมายถึงการโหลดข้อมูลไปที่รีจิสเตอร์สำหรับส่งข้อมูลออกไปภายนอก ส่วนการอ่านข้อมูลจากรีจิสเตอร์สำหรับข้อมูลมาใช้งาน

ผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยสามารถกำหนดได้จากบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON การใช้งานที่แตกต่างกัน 4 ประเภทนี้มีจุดประสงค์เพื่อความคล่องตัวในการรับส่งในการรับหรือส่งข้อมูลแบบแต่ละประเภทดังนี้

1. **โหมด 0** การทำงานของพอร์ทสื่อสารข้อมูลแบบอนุกรมในโหมด 0 ขา RXD จะใช้สำหรับรับและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ shift clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล (ข้อมูลจะถูกรับหรือส่งตามจังหวะของสัญญาณ shift clock) ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิต (บิตข้อมูล 8 บิต) โดยเริ่มรับและส่งบิตต่ำสุดก่อน (LSB first) อัตราการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดไว้ที่ 112 ของความถี่ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ทสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูล (start bit) และบิตสิ้นสุดของข้อมูล (stop bit) เพราะจังหวะการรับและส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock แล้ว

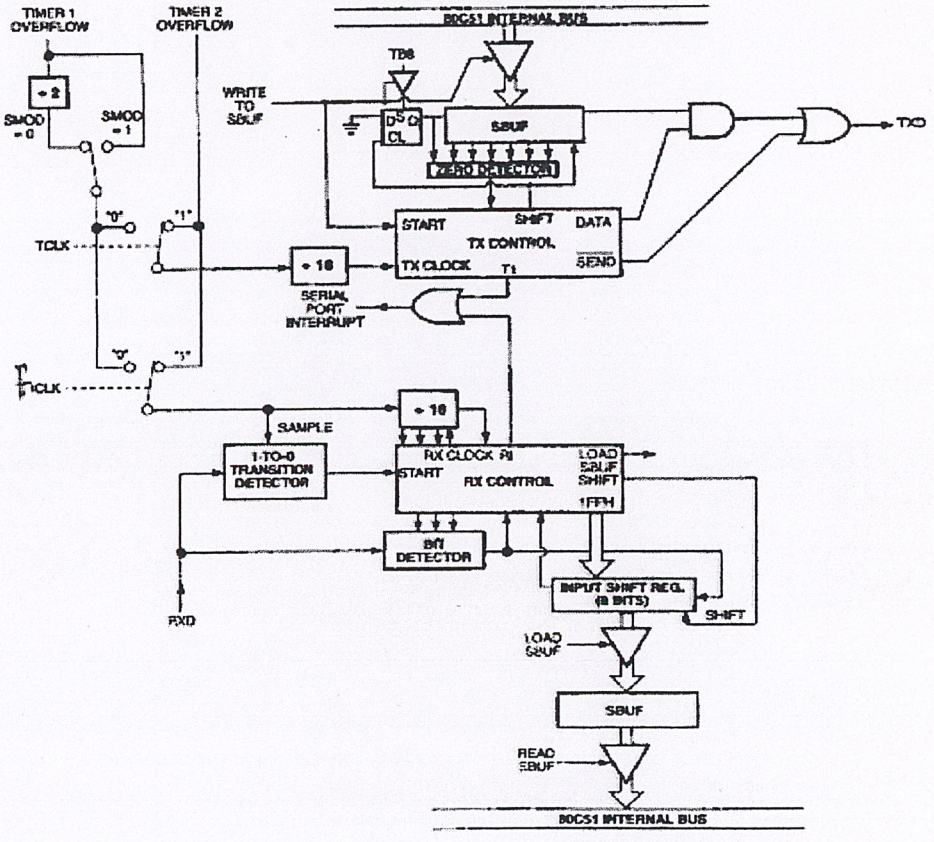


รูปที่ 2.27 การแสดงข้อมูลรับและส่งในการทำงานของพอร์ตสื่อสารอนุกรมโหมด 0

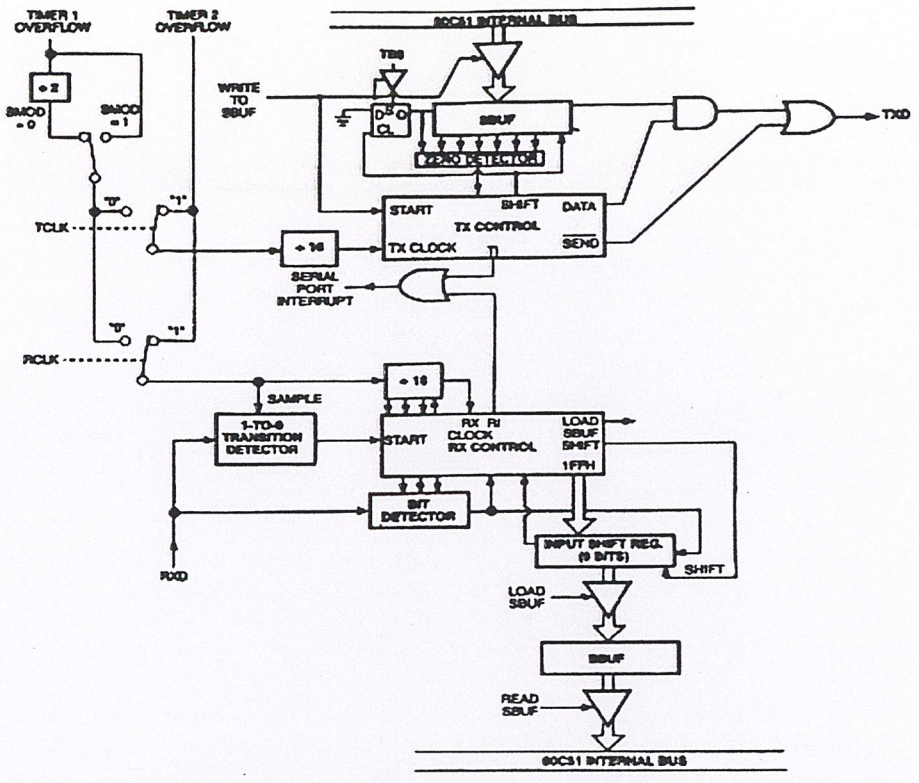
2. **โหมด 1** การทำงานแบบที่สองหรือการทำงานโหมด 1 นี้ มีการรับและส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งออกไปภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิต ประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับข้อมูลและส่ง

ข้อมูลบิตต่ำสุดก่อน) และบิตสิ้นสุดของข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าในบิตสิ้นสุดของข้อมูลที่รับได้จะไปอยู่ในบิต RB8 ของรีจีสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ทสื่อสารข้อมูลแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้

3. **โหมด 2** การทำงานจะมีการรับและส่งข้อมูลครั้งละ 11 บิต ข้อมูลจะถูกส่งออกภายนอกผ่านทางข้อมูล 1 บิต (มีค่า 0 เสมอ) บิตข้อมูล 8 บิต ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็นศูนย์หรือหนึ่งได้และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (มีค่าเป็น 1 เสมอ) ดังนั้นจำนวนบิตที่รับส่งทั้งหมด 11 บิตประกอบด้วยบิตต่าง ๆ ดังนี้ ในขณะทำการส่งข้อมูล บิตที่ 9 จะได้จากค่าในบิต TBB ของรีจีสเตอร์ใช้งานเฉพาะ SCON บิตนี้สามารถถูกกำหนดให้มีค่าเป็น 0 หรือ 1 อย่างไรก็ได้ส่วนใหญ่ในการใช้งานเฉพาะใช้บิตนี้สำหรับตรวจสอบความถูกต้องของข้อมูลที่รับหรือส่ง (parity bit) โดยจะนำบิต P (parity) ในรีจีสเตอร์ PSW ไปไว้ในบิต TB8 ส่วนในขณะรับข้อมูลบิตที่ 9 จะไปปรากฏอยู่ในบิต RB8 ของรีจีสเตอร์ SCON โดยไม่สนใจบิตสิ้นสุดของข้อมูลค่าอัตราในการรับหรือส่งข้อมูลโหมดนี้ถูกกำหนดไว้ที่ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้



รูปที่ 2.28 การแสดงข้อมูลรับปละส่งในการทำงานของพอร์ตสื่อสารอนุกรมโหมด 1



รูปที่ 2.31 การแสดงข้อมูลรับและส่งในการทำงานของพอร์ตสื่อสารอนุกรมโหมด 2, 3

4. **โหมด 3** การทำงานของพอร์ทสื่อสารข้อมูลแบบอนุกรมแบบสุดท้าย คือ การทำงานในโหมด 3 ในการทำงานโหมดนี้ข้อมูลจำนวน 11 บิตถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิตประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (เป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดค่าได้เหมือนในโหมด 2 (programmable 9 bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (เป็น 1 เสมอ) อัตราเร็วในการรับหรือส่งข้อมูลสามารถเปลี่ยนแปลงได้ดังจะศึกษาในรายละเอียดต่อไป ดังนั้นจะเห็นว่ารูปแบบการรับส่งข้อมูลในโหมด 2 ทุกอย่าง แต่ในโหมดนี้สามารถกำหนดค่าอัตราเร็วในการรับหรือส่งข้อมูลได้ตามความต้องการของผู้ใช้

7. อัตราการเร็วในการรับและส่งข้อมูล

baud rate หมายความถึง อัตราเร็วในการรับหรือส่งข้อมูล โดยในไมโครคอนโทรลเลอร์ MCS-51 ค่าอัตราเร็วในการรับและส่งข้อมูลจะมีค่าเท่าใด ก็ขึ้นอยู่กับการทำงานในแต่ละโหมดของพอร์ทสื่อสารข้อมูลแบบอนุกรมดังนี้

baud rate โหมด 0 = ความถี่ออสซิลเลเตอร์ที่ใช้ / 12

ในโหมด 2 ค่า baud rate ขึ้นอยู่กับค่าของบิต SMOD ที่อยู่ในรีจีสเตอร์ใช้งานเฉพาะ PCON โดย

บิต SMOD = 0 ค่า baud rate จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

บิต SMOD = 1 ค่า baud rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ที่ใช้

หลังจากการรีเซตไมโครคอนโทรลเลอร์ MCS-51 ค่าในบิต SMOD จะเป็น 0 เสมอ และเราสามารถเขียนสูตรสำหรับคำนวณค่า baud rate ได้ดังสมการนี้

baud rate โหมด 2 = [2(smod)*(ความถี่ออสซิลเลเตอร์) 1/64

เราสามารถที่จะสร้าง baud rate ค่าต่ำ ๆ ด้วยไทม์เมอร์ 1 ได้โดยปล่อยให้ไทม์เมอร์ 1 อินเตอร์รัพท์ซีพียูได้ และกำหนดการทำงานให้เป็นไทม์เมอร์ขนาด 16 บิต (โหมด 1) และไทม์เมอร์ 1 อินเตอร์รัพท์ซีพียูเพื่อโหลดค่าใหม่เองด้วยซอฟต์แวร์ขณะเกิดโอเวอร์โฟลว์ เนื่องจากในการทำงานในโหมด 1 ของไทม์เมอร์ 1 ไม่สามารถโหลดค่าใหม่เองด้วยฮาร์ดแวร์ได้

ตารางที่ 2.13 ค่าที่นำไปไว้ในรีจีสเตอร์ของไทม์เมอร์ 1 เมื่อใช้ baud rate ค่ามาตรฐานต่าง ๆ

Baud Rate	Fosc	SMOD	TIMER		
			C/T	MODE	Reload Value
(MODE) Max : 1 MHz	12 MHz	X	X	X	X
(MODE) Max : 375 KHz	12 MHz	1	X	X	X
(MODE) Min : 187.5 KHz	12 MHz	0	X	X	X
MODE 1, 3 : 62.5 K	12 MHz	1	0	2	FFH
19.2 K	11.059 MHz	0	0	2	FDH
9.6 K	11.059 MHz	0	0	2	FDH
4.8 K	11.059 MHz	0	0	2	FAH
2.4 K	11.059 MHz	0	0	2	F4H
1.2 K	11.059 MHz	0	0	2	E8H
137.5 K	11.059 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	2	FEEBH

จากตารางที่ 2.10 จะเห็นว่าในการทำงานของพอร์ทสื่อสารข้อมูลแบบอนุกรมโหมด 0 จะมีความเร็วในการส่งมากที่สุดเมื่อเปรียบเทียบกับโหมดอื่นที่ความถี่คริสตอลค่าเดียวกัน และจะเห็นว่าหากเลือกใช้คริสตอลความถี่ 11.059 เมกะเฮิรตซ์ จะสามารถตั้งค่า baud rate ในโหมด 1 และ 3 เป็นค่ามาตรฐานที่ใช้กันทั่วไปได้เช่น 1200, 2400, 4800, 9600, 19200 จึงเป็นเหตุผลสำคัญที่ใช้ในระบบควบคุมส่วนใหญ่เลือกใช้คริสตอลส่วนใหญ่เลือกใช้คริสตอลความถี่ 11.059 เมกะเฮิรตซ์มากกว่า 12 เมกะเฮิรตซ์

จากตารางที่ 2.10 นอกจาหจะแสดงค่า baud rate ค่าต่าง ๆ เปรียบเทียบให้เห็นแล้ว ตารางนี้ยังแสดงค่าที่ต้องโหลดไปไว้ในรีจีสเตอร์ใช้งานเฉพาะ TH1 ที่ค่า baud rate มาตรฐานต่าง ๆ ให้ทราบอีกด้วย สามารถเขียนโปรแกรมสามารถนำค่านี้ไปใช้ได้เลย แต่หากต้องการใช้ค่า baud rate อื่น ๆ ที่นอกเหนือไปจากตารางนี้ จะต้องคำนวณค่าที่ต้องโหลดให้รีจีสเตอร์ใช้งานเฉพาะ TH1 จากสมการที่แสดงไปแล้ว

8. โครงสร้างการอินเตอร์รัพท์ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 สามารถรับสัญญาณอินเตอร์รัพท์ที่เกิดขึ้นได้อย่างน้อย 5 ชนิดด้วยกัน (ไมโครคอนโทรลเลอร์ MCS-51 บางเบอร์ในตระกูลนี้สามารถรับสัญญาณอินเตอร์รัพท์ได้มากกว่า 5 ชนิด เช่น เบอร์ 8052 สามารถรับได้ 6 ชนิด) แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ทั้ง 5 ชนิดที่ไมโครคอนโทรลเลอร์ MCS-51 สามารถรับได้มีดังแสดงในตารางที่ 2.11

ตารางที่ 2.14 แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ทั้ง 5 ชนิด

Number	Name	Vector Address	Priority
1	INTO	0003H	Highest
2	TFO	000BH	↕
3	INT1	0013H	
4	TF1	001BH	
5	TL, RI	0023H	Lowest

อินเตอร์รัพท์แต่ละชนิดที่ไมโครคอนโทรลเลอร์ MCS-51 สามารถรับได้มีรายละเอียดดังต่อไปนี้

8.1 อินเตอร์รัพท์ที่เกิดจากภายนอก (External Interrupts) เป็นอินเตอร์รัพท์เกิดขึ้นจากภายนอกไมโครคอนโทรลเลอร์ MCS-51 มี 2 ชนิดด้วยกันคือ

- อินเตอร์รัพท์ภายนอกชนิด 0 รับได้จากขา INT0
- อินเตอร์รัพท์ภายนอกชนิด 1 รับได้จากขา INT1

ผู้ใช้สามารถกำหนดให้ไมโครคอนโทรลเลอร์ MCS-51 ตรวจสอบสัญญาณอินเตอร์รัพท์ทั้งสองชนิดที่เกิดขึ้นที่ขา INT0, INT1 2 แบบด้วยกัน คือ

- ตรวจสอบจากระดับสัญญาณ (level-activated)
- ตรวจสอบจากการเปลี่ยนสถานะสัญญาณ (transition-activated)

การตรวจสอบสถานะของสัญญาณอินเตอร์รัพท์ภายนอกที่ขาทั้งสอง สามารถเลือกได้เพียงอย่างใดอย่างหนึ่งขึ้นอยู่กับการกำหนดค่าบิต IT0, IT1 ในรีจีสเตอร์ใช้งานเฉพาะ TCON

เมื่อไมโครคอนโทรลเลอร์ MCS-51 ตรวจพบสัญญาณอินเตอร์รัพท์จากภายนอกจะมีผลทำให้บิต IEO (จากขา INT0) หรือ IE1 (จากขา INT1) ของรีจีสเตอร์ใช้งานเฉพาะ TCON ถูกเซต บิตทั้งสองจะเป็นตัวบอกสถานะของสัญญาณอินเตอร์รัพท์ที่เกิดจากภายนอก โดยถูกเซตเมื่อเกิดสัญญาณอินเตอร์รัพท์และจะถูกเคลียร์โดยฮาร์ดแวร์ภายใน MCS-51 เอง เมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัพท์ต่อเมื่อสัญญาณอินเตอร์รัพท์ภายนอกที่เกิดขึ้นเป็นชนิดที่ตรวจสอบได้ จากการเปลี่ยนสถานะสัญญาณแต่ถ้าสัญญาณอินเตอร์รัพท์ภายนอกที่เกิดขึ้นได้มาจากการตรวจสอบระดับสัญญาณ เมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัพท์ จะไม่มีการเคลียร์บิต IE0 หรือ IE1 ให้ ในกรณีนี้จะเป็นหน้าที่ของวงจรกำเนิดสัญญาณอินเตอร์รัพท์ภายนอกที่จะต้องทำหน้าที่ควบคุมสถานะของสัญญาณที่ขา INTx (INT0 หรือ INT1) ให้กลับสู่สภาพเดิมเอง มิฉะนั้นโปรแกรมหลักที่ทำงานอยู่จะถูกอินเตอร์รัพท์ไปเรื่อย ๆ จนกระทั่งสัญญาณอินเตอร์รัพท์กลับมีค่าเป็น 1 อีกครั้ง

8.2 อินเตอร์รัพท์ของไทม์เมอร์ 0 และไทม์เมอร์ 1 อินเตอร์รัพท์ของไทม์เมอร์ 0 หรือไทม์เมอร์ 1 ถูกทำให้เกิดขึ้นโดยบิต TE0 หรือ TE1 ซึ่งถูกเซต ยกเว้นไทม์เมอร์ 0 ในโหมด 3 ซึ่งหยุดการทำงานเมื่อมีอินเตอร์รัพท์จากไทม์เมอร์เกิดขึ้น บิท TF0 และ TF1 จะถูกเคลียร์โดยฮาร์ดแวร์ภายในไมโครคอนโทรลเลอร์ MCS-51 เอง เมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัพท์

8.3 อินเตอร์รัพท์ของพอร์ตสื่อสารอนุกรม (Serial Port Interrupt) พอร์ทสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถทำให้เกิดสัญญาณอินเตอร์รัพท์ได้ สัญญาณอินเตอร์รัพท์ที่เกิดขึ้นได้มาจากบิต T1 หรือ R1 ที่นำมาผ่านเกตเตอร์ และบิตที่ควบคุมการอินเตอร์รัพท์ทั้งสองนี้จะไม่ถูกเคลียร์โดยฮาร์ดแวร์ในไมโครคอนโทรลเลอร์ MCS-51 เมื่อซีพียูไปทำงานในโปรแกรมบริการอินเตอร์รัพท์เพราะการเกิดอินเตอร์รัพท์ของพอร์ทสื่อสารอนุกรมอาจจะเกิดจากบิท R1 หรือ T1 ก็ได้ ดังนั้นโปรแกรมในส่วนบริการอินเตอร์รัพท์จะต้องตรวจสอบเองว่าสัญญาณอินเตอร์รัพท์ที่เกิดขึ้นได้มาจากบิท R1 หรือ T1 และบิตทั้งสองจะถูกเคลียร์โดยซอฟต์แวร์เท่านั้น

8.4 อินเตอร์รัพท์ของไทม์เมอร์ 2 (Timer 2 Interrupt) ในเบอร์ 8052 จะมีรีจีสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์เพิ่มขึ้นมาอีก 1 ตัวคือ ไทม์เมอร์ 2 โดยสามารถใช้สร้างสัญญาณอิน

เตอร์รัพท์ได้เหมือน เช่น ในไทม์เมอร์ 0 และไทม์เมอร์ 1 แต่การเกิดอินเตอร์รัพท์ของไทม์เมอร์ 2 จะแตกต่างออกไปจากไทม์เมอร์ 0 และไทม์เมอร์ 1 ดังนี้

อินเตอร์รัพท์ของไทม์เมอร์ 2 เกิดขึ้นโดยการนำบิต TF2 และ EXF2 มาผ่านออร์เกต โดยบิต TF2 และ EXF2 ทั้งสองจะไม่ถูกเคลียร์โดยฮาร์ดแวร์เมื่อซีพียูไปทำงานในส่วนโปรแกรมบริการอินเตอร์รัพท์เหมือนในการเกิดอนเตอร์รัพท์ของพอร์ทสื่อสารอนุกรม ดังนั้นในโปรแกรมบริการอินเตอร์รัพท์จะต้องตรวจสอบเองว่าบิต TF2 และ EXF2 ที่เป็นสาเหตุทำให้เกิดการอินเตอร์รัพท์และบิตที่ทำให้เกิดอินเตอร์รัพท์จะต้องถูกเคลียร์โดยซอฟต์แวร์เองด้วย

อินเตอร์รัพท์แต่ละชนิดที่กล่าวไปแล้ว สามารถควบคุมให้สามารถอินเตอร์รัพท์ไมโครคอนโทรลเลอร์ MCS-51 ได้หรือไม่ โดยการควบคุมจากบิตต่าง ๆ ในรีจีสเตอร์ใช้งานเฉพาะ IE ดังต่อไปนี้

EA	X	ET2	ES	ET1	EX1	ET0	EX0

บิต	ชื่อบิต	
IE.7	EA	ใช้ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ทั้งหมด 0 : MCS-51 จะไม่ตอบสนองต่อสัญญาณอินเตอร์รัพท์ใด ๆ ทั้งสิ้น 1 : อินเตอร์รัพท์แต่ละชนิดจะถูกควบคุมการตอบสนองอย่างอิสระจากบิต รีจีสเตอร์นี้
IE.6		ไม่ถูกกำหนดการใช้งาน
IE.5	ET2	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของไทม์เมอร์ 2 เมื่อเกิดโอเวอร์โฟลว์
IE.4	ES	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของพอร์ทสื่อสารอนุกรม
IE.3	ET1	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของไทม์เมอร์ 1 เมื่อเกิดโอเวอร์โฟลว์
IE.2	EX	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ภายนอกชนิด 1
IE.1	ET0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของไทม์เมอร์ 0 เมื่อเกิดโอเวอร์โฟลว์
IE.0	EX0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์ภายนอกชนิด 0

การกำหนดให้บิตควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัพท์แต่ละชนิดมีค่าเป็น 0 หมายถึง ไม่ให้ MCS-51 ตอบสนองต่อสัญญาณอินเตอร์รัพท์ชนิดนั้น หากกำหนดให้บิตควบคุม

การตอบสนองต่อสัญญาณอินเตอร์รัพท์แต่ละชนิดมีค่าเป็น 12 หมายถึง ให้ไมโครคอนโทรลเลอร์ MCS-51 ตอบสนองต่อสัญญาณอินเตอร์รัพท์ชนิดนั้น (บิต EA ถูกเซตไว้ก่อนด้วย)

บิต EA ในรีจีสเตอร์ใช้งานเฉพาะ IE สามารถควบคุมการอินเตอร์รัพท์ในไมโครคอนโทรลเลอร์ MCS-51 ได้ทั้งหมดหากบิตนี้มีค่าเป็น 0 สัญญาณอินเตอร์รัพท์ทุกชนิดที่เกิดขึ้นจะไม่สามารถอินเตอร์รัพท์ไมโครคอนโทรลเลอร์ MCS-51 ได้ แต่หากบิตนี้มีค่าเป็น 1 สัญญาณอินเตอร์รัพท์แต่ละชนิดจะถูกควบคุมให้อินเตอร์รัพท์ไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างอิสระ (ควบคุมจากบิต IE.0-IE.5)

บิต IE.5-IE.6 ไม่ถูกใช้ใน 8051 เพราะถูกสงวนไว้ใยไมโครคอนโทรลเลอร์ MCS-51 เบอร์อื่น ๆ ที่สามารถรับอินเตอร์รัพท์ได้เพิ่มขึ้น ดังนั้นซอฟแวร์ของผู้ใช้ไม่ควรจะมีคำสั่งเขียนลงไปในบิทเหล่านี้ เพื่อให้โปรแกรมยังคงสามารถใช้กับชิพเบอร์ใหม่ ๆ ในตระกูลนี้ได้

9. โครงสร้างระดับความสำคัญในการบริการอินเตอร์รัพท์ (Priority Level Structure)

อินเตอร์รัพท์แต่ละชนิดสามารถถูกเลือกระดับความสำคัญในการบริการได้ 2 ระดับ โดยการเซตหรือเคลียร์บิทในรีจีสเตอร์ใช้งานเฉพาะ IP รีจีสเตอร์ใช้งานเฉพาะ IP (Interrupt Priority) เข้าถึงข้อมูลได้ในระดับบิต

PCT	X	PT2	PS	PT1	PX1	PT0	PX0

รูปที่ 2.33 รีจีสเตอร์ใช้งานเฉพาะ IP

บิต	ชื่อบิต	
IP.7	-	ไม่ถูกกำหนดการใช้งาน
IP.6	-	ไม่ถูกกำหนดการใช้งาน
IP.5	PT2	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพท์ไทม์เมอร์ 2
IP.4	PS	กำหนดลำดับความสำคัญในการตอบสนองสัญญาณอินเตอร์รัพท์ของพอร์ทสื่อสารอนุกรม
IP.3	PT1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของไทม์เมอร์ 1
IP.2	PX1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพท์ภายนอกชนิด 1
IP.1	PT0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพท์ของไทม์เมอร์ 0
IP.0	PX0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัพท์ภายนอกชนิด 0

การให้บิตกำหนดลำดับความสำคัญของอินเตอร์รัพท์เป็น 0 หมายถึง ให้อินเตอร์รัพท์ชนิดนั้นมีลำดับความสำคัญต่ำ ส่วนการให้บิตกำหนดลำดับความสำคัญของอินเตอร์รัพท์เป็น 1 หมายถึง ให้อินเตอร์รัพท์ชนิดนั้นมีลำดับความสำคัญสูง

ระดับความสำคัญในการบริการอินเตอร์รัพท์ที่เกิดขึ้นมีได้ 2 ระดับ ได้แก่

1. **อินเตอร์รัพท์ระดับความสำคัญต่ำ (Low Priority Interrupt)** อินเตอร์รัพท์ชนิดนี้สามารถถูกอินเตอร์รัพท์จากสัญญาณอินเตอร์รัพท์ระดับความสำคัญสูงได้ แต่จะไม่สามารถถูกอินเตอร์รัพท์โดยสัญญาณอินเตอร์รัพท์ระดับความสำคัญต่ำตัวอื่น ๆ ได้

2. **อินเตอร์รัพท์ระดับความสำคัญสูง (High Priority Interrupt)** อินเตอร์รัพท์ประเภทนี้ไม่สามารถถูกอินเตอร์รัพท์โดยสัญญาณอินเตอร์รัพท์ชนิดอื่น ๆ ได้เลย นั่นคือมีระดับความสำคัญสูงสุด

ถ้ามีสัญญาณอินเตอร์รัพท์เกิดขึ้นพร้อมกัน 2 ชนิด โดยมีระดับความสำคัญในการบริการอินเตอร์รัพท์ไม่เท่ากัน สัญญาณอินเตอร์รัพท์ที่มีระดับความสำคัญสูงกว่าจะได้รับการบริการก่อน แต่ถ้ามีการขออินเตอร์รัพท์พร้อมกัน 2 ชนิด ซึ่งมีวามสำคัญเท่าเทียมกัน ลำดับการบริการอินเตอร์รัพท์ภายในจะเป็นตัวกำหนดเองว่าอินเตอร์รัพท์ชนิดใดควรจะถูกบริการก่อน กังนั้นภายในระดับความสำคัญของการบริการอินเตอร์รัพท์หนึ่ง ๆ จะมีระดับความสำคัญในการบริการอินเตอร์รัพท์ย่อยลงไปอีกหนึ่งระดับ

การจัดระดับการกับสัญญาณอินเตอร์รัพท์ที่มีระดับความสำคัญเท่าเทียมกันที่เกิดขึ้นพร้อมกันถูกใช้เพียงเพื่อแก้ปัญหาสัญญาณอินเตอร์รัพท์ที่มีระดับความสำคัญเท่ากันที่เกิดขึ้นพร้อมกันเท่านั้น

รีจีสเตอร์ใช้งานเฉพาะ IP มีบิตที่ไม่ถูกใช้งานอยู่บางบิต คือ IP.7 และ IP.6 โดยไม่ถูกใช้ในไมโครคอนโทรลเลอร์ 8051 และ 8052 ในไมโครคอนโทรลเลอร์ 8051 จะมีบิตที่ว่าเพิ่มมาอีก 1 บิต คือ IP.5 ดังนั้นซอฟต์แวร์ของผู้ใช้ไม่ควรมีการเขียนค่า 1 ไปที่ตำแหน่งบิทเหล่านี้เพราะนอาจถูกนำไปใช้งานในไมโครคอนโทรลเลอร์ MCS-51 เบอร์ใหม่ ๆ ในอนาคตต่อไป

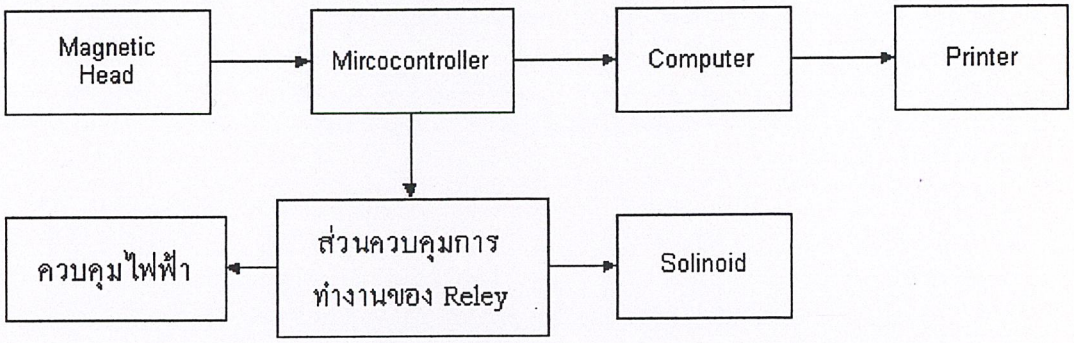
ไมโครคอนโทรลเลอร์ AT89S8252

AT89S8252 เป็นไมโครคอนโทรลเลอร์ที่ผลิตขึ้นโดย ATMEL ซึ่งมีโครงสร้างและชุดคำสั่งเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยบรรจุอยู่ในตัวถังแบบ DIP ขนาด 20 ขา นับว่าเป็นไมโครคอนโทรลเลอร์ขนาดเล็กอีกตัวหนึ่ง ซึ่งมีโครงสร้างที่ง่ายต่อการศึกษาและทำความเข้าใจและยังสามารถทำการพัฒนาโปรแกรมได้โดยง่ายไม่ยุ่งยากซับซ้อน มีพอร์ตหน่วยความจำและทรัพยากรที่สำคัญ ๆ อย่างครบถ้วน เหมาะแก่การนำไปประยุกต์ใช้งานในด้านต่าง ๆ ได้เป็นอย่างดี

บทที่ 3
การออกแบบ

3.1 กล่าวนำ

การออกแบบประกอบด้วย 2 ส่วน คือ ส่วนประมวลผลและส่วนวงจรควบคุม ซึ่งส่วนประมวลผลได้ใช้ไมโครโปรเซสเซอร์ และไมโครคอมพิวเตอร์ ส่วนวงจรได้แก่ วงจรอ่านข้อมูลจากบัตรแถบแม่เหล็ก วงจรควบคุมอุปกรณ์โซลินอยด์ โดยการทำงานทั้งหมดได้แสดงโดยโครงสร้างการทำงานของระบบควบคุมในรูปที่ 3.1 ซึ่งมีการทำงาน ดังนี้



รูปที่ 3.1 โครงสร้างการทำงานของระบบควบคุมการเข้าโดยบัตรแม่เหล็ก

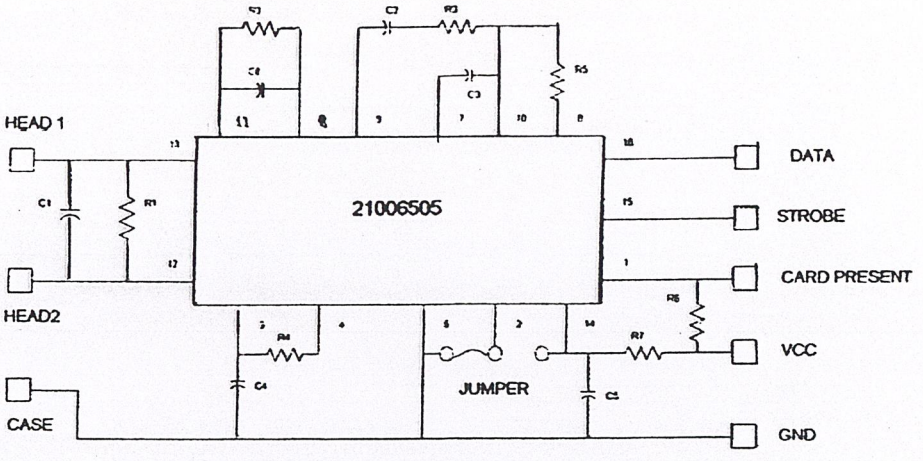
ส่วนของวงจรรับข้อมูลหัวอ่านแถบแม่เหล็ก จะต่ออยู่กับไมโครคอนโทรลเลอร์โดยใช้ IC AT89S8252 เป็นตัวรับข้อมูลโดยผ่านพอร์ตสื่อสารอนุกรม RS-232 และส่งสัญญาณข้อมูลไปยังไมโครคอมพิวเตอร์ทำการประมวลผลข้อมูล ซึ่งในบทนี้จะกล่าวถึงการออกแบบการสร้างและการทำงานของระบบ โดยแยกเป็น 2 ส่วนใหญ่ ๆ คือ

1. ส่วนวงจร
2. ส่วนโปรแกรม (Microcontroller)
3. ส่วนโปรแกรม (Microcomputer)

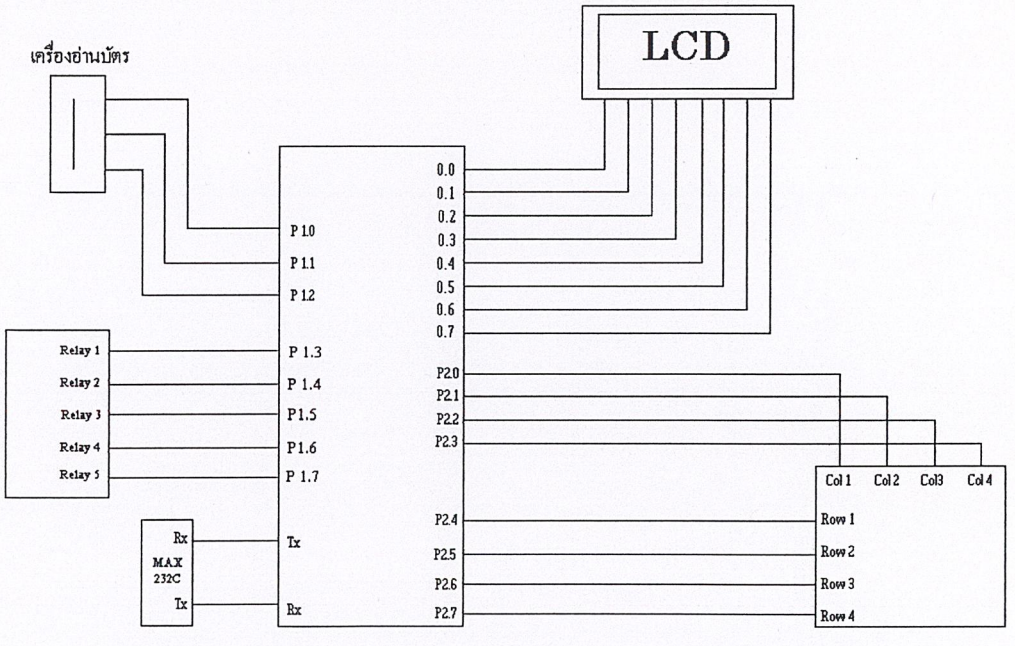
3.2 การออกแบบวงจร

วงจรรับข้อมูล ในวงจรรับข้อมูลจะใช้ IC 21006505 เป็นตัวทำการอ่านสัญญาณจากหัวเทปที่กำลังรูดผ่านบัตรแถบเหล็ก เมื่อสัญญาณจากหัวอ่านผ่านเข้าขา 12, 13 ของ IC โดยมี C1 และ R1 เป็น impedance matching ซึ่งใน IC จะประกอบด้วยวงจร Input Amplifier ซึ่งทำการขยายสัญญาณจากหัวเทปโดยมี R2 เป็นตัวกำหนด Gain การขยาย หลังจากนั้นสัญญาณที่ได้จะถูกป้อนผ่าน C2 , R3 ทำการ Coupling สัญญาณส่งไปยัง Summing Amp. ซึ่งทำการเปรียบเทียบระดับแรงดันให้ได้

ระดับ “0” หรือ “1” ส่งผ่านไปยัง Schmitt Trigger เพื่อทำการเปลี่ยนระดับแรงดัน Pulse ให้มีค่าเป็นระดับแรงดัน TTL (0-5V) เพื่อจะส่งไปเข้าส่วนของ CPU เพื่อทำการ Decode สัญญาณต่อไป



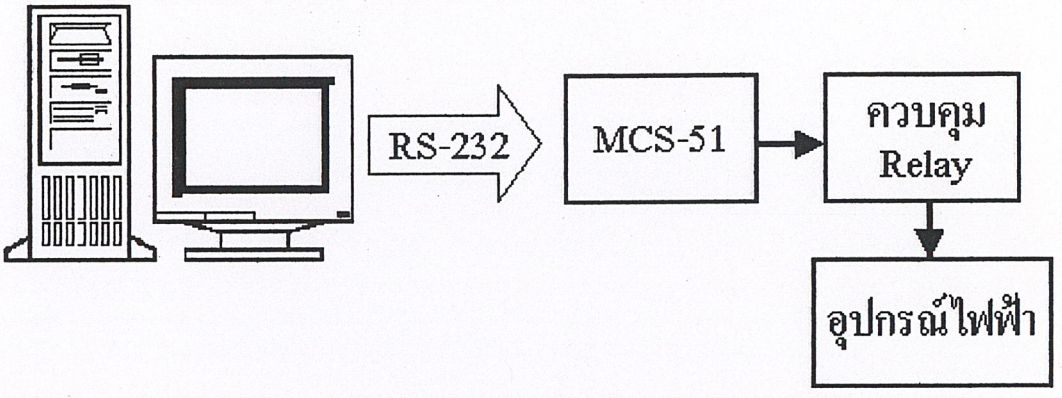
รูปที่ 3.2 วงจรหัวอ่านบัตรแถบแม่เหล็ก



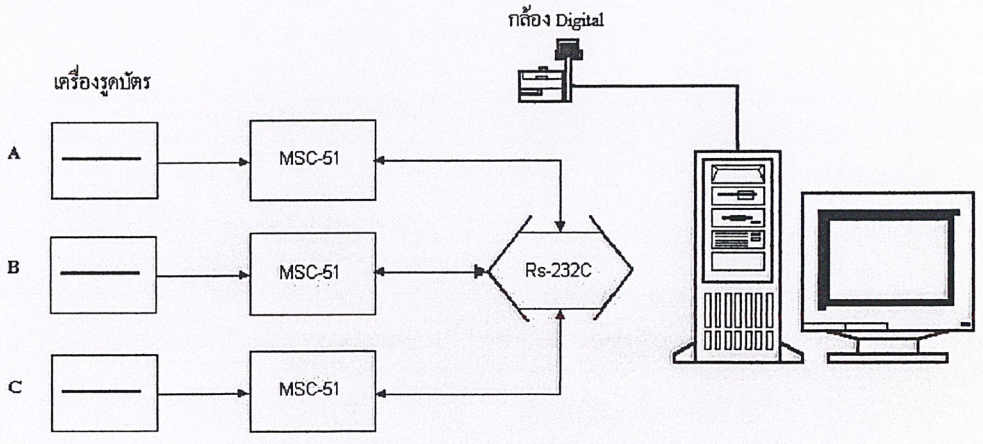
รูปที่ 3.3 วงจรเชื่อมต่อส่วนควบคุมการทำงานของระบบ และการต่อสายสัญญาณจากเครื่องอ่านบัตรแม่เหล็ก

3.3 การออกแบบส่วนควบคุมไฟฟ้า

เป็นการออกแบบให้ Computer สามารถทำหน้าที่เป็นตัวควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า โดยผ่านการสื่อสารทาง RS-232C Computer จะส่งข้อมูลมายัง MCS-51 เมื่อรับค่าเข้ามาแล้ว MCS-51 จะทำการประมวลผลแล้วส่งสัญญาณไปเปิดปิดอุปกรณ์



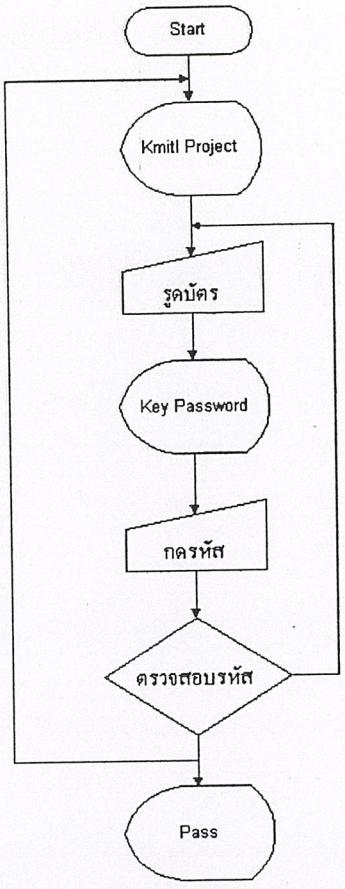
รูปที่ 3.4 แสดงการทำงานของส่วนระบบควบคุมอุปกรณ์ไฟฟ้า



รูปที่ 3.5 แผนผังแสดงการทำงานของระบบ

3.4 การออกแบบ Soft Ware บน Board MCS-51

ใช้ภาษา Assembly ในการเขียนโปรแกรม โดยสามารถสั่งการไปยัง MCS-51 ได้ รูปแบบการใช้งานทั่วไปจะประกอบด้วย การอ้างอิงถึง Address ของ Memory ใน MCS-51



รูปที่ 3.6 แผนผังโปรแกรมควบคุมการทำงานบนไมโครคอนโทรลเลอร์

โปรแกรมเริ่มทำงาน เมื่อมีการตั้งค่าต่าง ๆ ในการสื่อสารแบบอนุกรม เพื่อรอรับรหัสสัญญาณจากการรูดบัตรแถบแม่เหล็ก จนเมื่อมีการรูดบัตรแถบแม่เหล็ก โปรแกรมจะทำการอ่านรหัสข้อมูลที่เข้ามาแล้วทำการแปลงเป็นรหัสแอสกี และส่งข้อมูลไปยังไมโครคอมพิวเตอร์ เพื่อทำการประมวลผลและรอรับสัญญาณตอบรับข้อมูลจากไมโครคอมพิวเตอร์เพื่อทำการควบคุมวงจรกำเนิดเสียงและวงจรควบคุมโซลินอยด์ จากนั้นไมโครโทรลเลอร์ก็จะไปรอรับสัญญาณที่เกิดจากการรูดบัตรใหม่อีกครั้ง

3.5 โครงสร้างและการทำงานของระบบ

ในหลักการทำงานของเครื่องรูดบัตรสามารถนำไปประยุกต์ใช้งานได้หลายอย่าง เช่น นำเครื่องรูดบัตรติดตั้งที่ประตู เพื่อทำการตรวจเช็คจำนวนสมาชิกที่เข้ามา ติดตั้งระบบนิรภัยโดยยอมให้ผ่านเฉพาะคนที่กำหนด ซึ่งในขณะที่ทำงานอาจจะให้คนมาคอยทำการตรวจเช็คเท่านั้น หลักใน

การทำงาน คือ การใช้บัตรที่มีแถบแม่เหล็กมาผ่านการบันทึกข้อมูลและติดตั้ง Password เพื่อนำไปใช้งาน โดยมีโซลินอยด์ เป็นตัว Operate ในรูปที่ 3.8 จะแสดงลำดับการทำงานของระบบ

3.6 ขั้นตอนการทำงาน

การเพิ่มข้อมูล (Add Data) นำบัตรที่มีแถบแม่เหล็กมารูดที่เครื่องรูดบัตรซึ่งเชื่อมโยงอยู่กับ MCS-51 โดย MCS-51 จะนำค่าที่ได้จากบัตรซึ่งอยู่ในรูปของ Assembly ส่งไปยังคอมพิวเตอร์

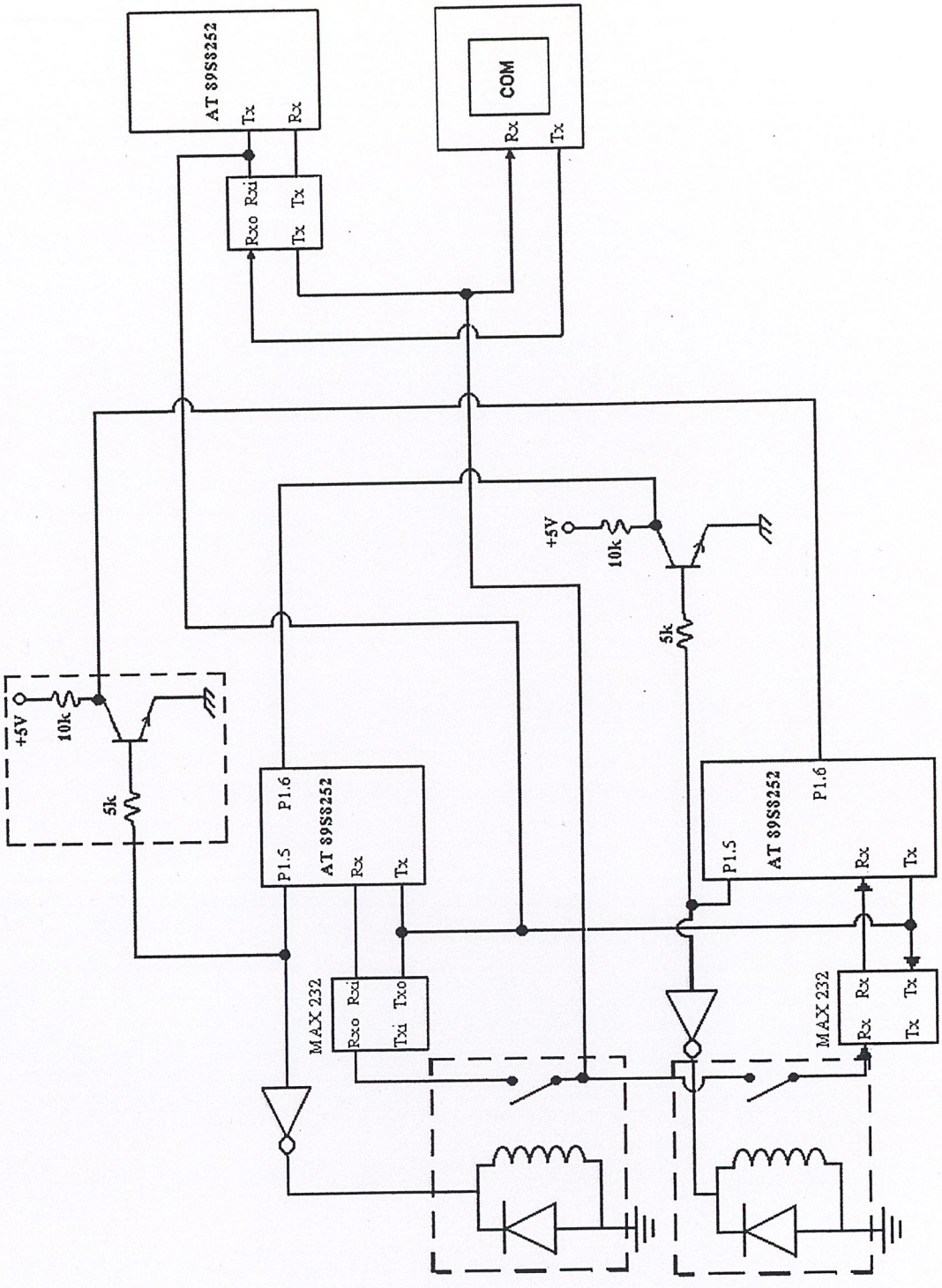
คอมพิวเตอร์จะมีโปรแกรมที่ใช้เฉพาะในการบันทึก ข้อมูลที่ได้รับ โดยที่เราจะต้องกำหนด Password เพื่อทำการเก็บข้อมูลที่ได้ลงในฐานข้อมูล และทำการ บันทึกข้อมูล

การควบคุมการเข้าออก

1. นำบัตรมารูดที่เครื่องอ่านบัตร MCS-51 จะทำหน้าที่อ่านข้อมูลของบัตรและทำการส่งข้อมูลบัตรไปยังคอมพิวเตอร์ หลังจากนั้นคอมพิวเตอร์จะทำการตรวจสอบข้อมูลของบัตร และเรียกข้อมูลจากฐานข้อมูลนั้นออกมา (ในกรณีที่มีข้อมูลของบัตรอยู่ในฐานข้อมูล)

2. ทำการกด Password จาก Keyboard เพื่อทำการส่งรหัสไปยังคอมพิวเตอร์ ตัวประมวลผลของคอมพิวเตอร์จะทำการตรวจเช็ครหัสว่าตรวกันกับฐานข้อมูลหรือไม่หลังจากนั้น คอมพิวเตอร์จะส่งข้อมูลมายัง MCS-51 เพื่อบอกว่ารหัสตรงกันหรือไม่ เพื่อส่งสัญญาณไปควบคุมโซลีนอยด์

3. คอมพิวเตอร์จะทำการบันทึก วัน เวลา ลงในฐานข้อมูลด้วย



รูปที่ 3.7 แสดงการเชื่อมต่อของตัวไมโครคอนโทรลเลอร์แต่ละตัว

3.7 การออกแบบทางด้าน Soft Ware

การเขียนโปรแกรมควบคุมแบ่งเป็น 2 ส่วน คือ การเขียนโปรแกรมควบคุมบน Computer โดยใช้ Visual Basic version 5 และเขียนโปรแกรมบน Board Micro-controller ควบคุมการเข้า-ออก ซึ่งสามารถออกแบบได้ดังต่อไปนี้

3.7.1 การออกแบบทางด้าน Software ส่วนที่ทำงานบน PC

ในส่วนนี้จะรับข้อมูลที่ส่งมาจาก Board Micro-controller ซึ่งเป็นข้อมูลจากบัตร โดยรูปแบบของข้อมูลที่ได้รับจะประกอบด้วยส่วนของ Header, Data และ Check Sum โดยลักษณะของ Frame ข้อมูลจะเป็นดังนี้

Destination	Source	Command	Length	Data

โดยปกติวิธีการสร้างฐานข้อมูลที่นำมาใช้งานในวิชวลเบสิกทำได้ 3 วิธี คือ

1. สร้างโดยอาศัยฐานข้อมูลของระบบจัดการฐานข้อมูลนั้น

โดยหากต้องการใช้ฐานข้อมูลของ dBase ก็ใช้เครื่องมือของ dBase เป็นตัวสร้างหรือหากต้องการฐานข้อมูลของ Paradox ก็ใช้โปรแกรม Paradox เป็นตัวสร้างฐานข้อมูล

2. ใช้โปรแกรม Visual Data manager

โปรแกรมนี้มีอยู่ในส่วนของ Visual Basic อยู่แล้วซึ่งใช้สร้างฐานข้อมูลในรูปแบบของ Jet 1.1 หรือ Access โดยตัวโปรแกรมสามารถทำได้เพียงลบหรือเพิ่มข้อมูลได้เท่านั้น ไม่สามารถทำการแก้ไขข้อมูลหรือเปลี่ยนแปลงรายละเอียดที่มีอยู่แล้วได้

3. ใช้ Data Control

Data Control เป็น Control ประเภทหนึ่งที่สามารถจัดใน Form ของ Visual Basic ได้ และใช้ในการจัดการข้อมูลต่าง ๆ ในฐานข้อมูลโดยไม่ต้องเขียนโปรแกรม

การสร้างฐานข้อมูล

ขั้นตอนการสร้างฐานข้อมูลขึ้นมาใหม่สามารถทำได้ดังนี้

1. ต้องกำหนดตัวแปรสำหรับฐานข้อมูลนั้นขึ้นมาก่อน โดยให้มีประเภทเป็นออปเจ็คฐานข้อมูล หรือ Database เช่น Dim Data As Database

2. ใช้คำสั่ง Set ร่วมกับฟังก์ชัน Create Database เพื่อกำหนดค่าให้กับตัวแปรชนิดฐานข้อมูลดังตัวอย่าง เช่น Set Data = Create Database (C:\

รายละเอียดของแต่ละส่วนที่ใช้ในการสร้างฐานข้อมูลของ Create Database มีดังนี้

3.8 กำหนดชื่อแฟ้มของฐานข้อมูล (Databasename)

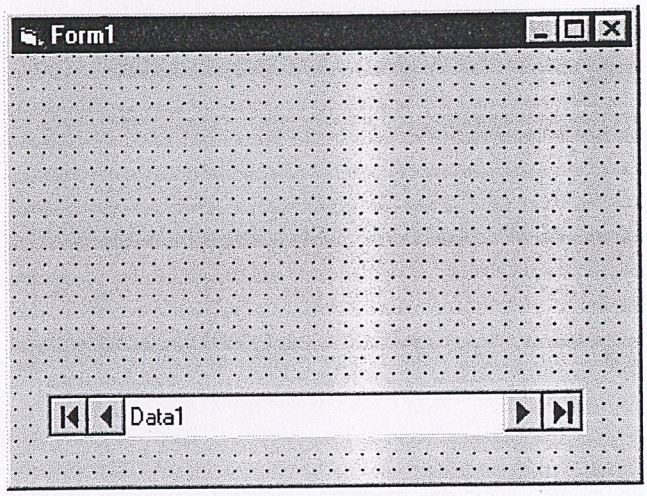
เป็นการระบุชื่อแฟ้มฐานข้อมูล รวมทั้งตำแหน่งหรือไดเร็กตอรี่ที่ต้องการ ในลักษณะเช่นเดียวกับการระบุชื่อแฟ้มข้อมูลทุกประการ

การใช้ Data Control นั้นมีข้อดีอยู่ว่าใช้งานค่อนข้างง่าย เพราะมีลักษณะการใช้เหมือน Control ประเภทอื่น ๆ เช่น คอนโทรลแสดงและรับตัวอักษร (Text Box) Data Control มีลักษณะแสดงเป็นฟอร์มเพื่อเป็นการตอบโต้กับผู้ใช้โดยตรง ซึ่งเกิดเป็นเหตุการณ์ขึ้นกับ Data Control นั้น ๆ

3.9 การใช้งาน Data Control

วิธีการใช้ Data Control ในการจัดการกับฐานข้อมูลมีดังนี้

1. กำหนด Data Control ขึ้นในฟอร์มที่เราต้องการจะใช้ จะได้ดังรูปที่ 3.7



รูปที่ 3.7 แบบฟอร์ม Data Control

2. จากนั้นกำหนดคาคุณสมบัติ (property) ที่สำคัญบางตัวที่เกี่ยวข้องกับการใช้ฐานข้อมูล

ตารางที่ 3.1 กำหนดค่าคุณสมบัติ

คุณสมบัติ	ความหมาย
Databasename	เพื่อบอกว่าเราต้องการจะใช้ดาต้าคอนโทรลนี้กับข้อมูลตัวใด โดยระบุชื่อฐานข้อมูลนั้นเลย
Recordsource	สำหรับกำหนดค่าที่จะกำหนดให้คุณสมบัติตัวนี้ อาจจะกำหนดเป็นชื่อของตาราง (table) ที่มีอยู่ในฐานข้อมูลที่เรากำหนดใน databasename

คุณสมบัติ	ความหมาย
ReadOnly	กำหนดให้เป็น True หากต้องการจะเปิดใช้ฐานข้อมูลให้อ่านเพียงอย่างเดียว โดยค่าเริ่มต้นที่กำหนดจะเป็น false
Connect	เป็นการระบุว่าฐานข้อมูลที่เราจะใช้นั้น เป็นฐานข้อมูลแบบใดเป็น MS-Access หรือว่าเป็น Dbase หรือว่าเป็นฐานข้อมูลภายนอกที่ต้องผ่าน ODBC ซึ่งค่าที่กำหนดในที่นี้ จะเหมือนกับค่าพารามิเตอร์ Connect ที่เราต้องกำหนดในตอนสั่ง OpenDatabace
Exclusive	กำหนดว่าเป็น True หรือ False เท่านั้น ซึ่งจะหมายถึงว่าฐานข้อมูลที่กำหนดให้กับดาต้าคอนโทรลตัวนี้เราต้องเปิดแบบให้ใช้คนเดียว (กำหนดให้เป็น True) หรือต้องการที่จะใช้เป็นฐานข้อมูลร่วมกับผู้อื่นหรือดาต้าคอนโทรลตัวอื่น (กำหนดให้เป็น False)

3. กำหนดคอนโทรลอื่น ๆ ขึ้นบนฟอร์ม เพื่อใช้เป็นที่แสดงข้อมูลในฐานข้อมูล หรือรับข้อมูลผู้ใช้กลับเข้าไปในฐานข้อมูล ซึ่งในวิชวลเบสิกนี้จะมีคอนโทรลต่าง ๆ ดังนี้

คอนโทรลแสดงและรับตัวอักษร (Text Box)

คอนโทรลตัวนี้จะผูกกับฟิลด์ข้อมูลหลาย ๆ ประเภท เช่น ฟิลด์ชนิดตัวอักษร (Text) และตัวเลขฝนลักษณะต่าง ๆ

คอนโทรลแสดงตัวอักษร (Label)

ใช้ได้กับฟิลด์หลาย ๆ ประเภทเหมือนคอนโทรลแสดงและรับตัวอักษร แต่ผู้ใช้ไม่สามารถแก้ไขข้อมูลที่แสดงนั้นได้

คอนโทรลสำหรับเลือก (Check Box)

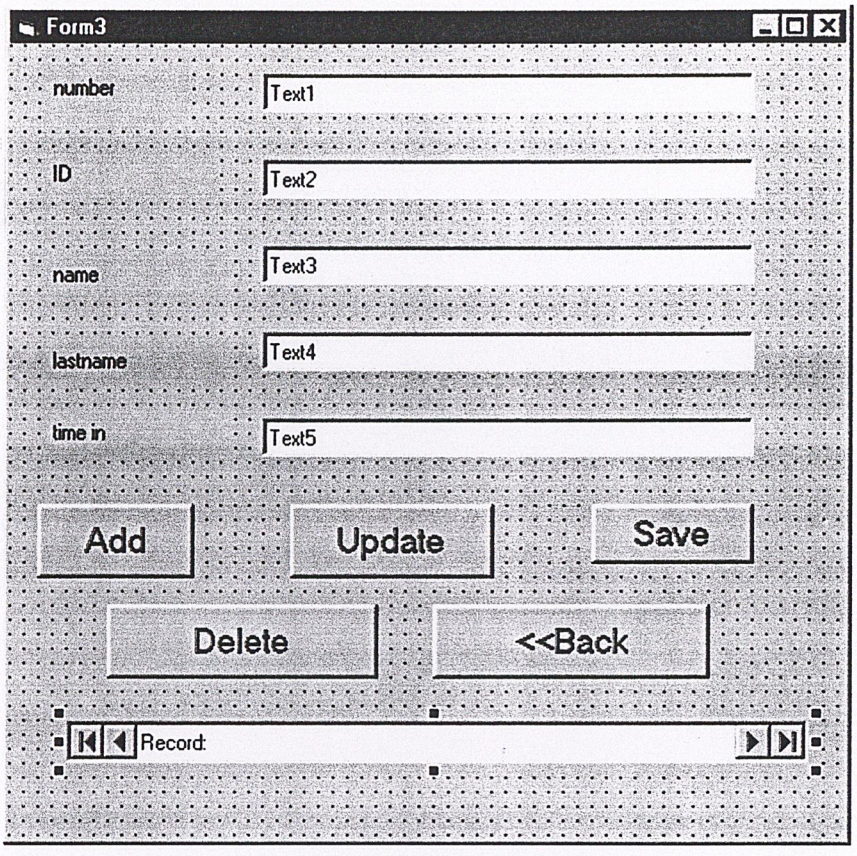
จะใช้ได้กับข้อมูลประเภทตรรกะ (Boolean) เท่านั้น ซึ่งหากในฟิลด์มีค่าเป็นจริง (True) คอนโทรลจะแสดงในรูปของการถูกเลือก

คอนโทรลแสดงรูป (Picture Box)

จะใช้ผูกเข้ากับฟิลด์ข้อมูลประเภทรูปภาพเท่านั้น ซึ่งอาจจะเป็นฟิลด์ประเภทบันทึก (memo) หรือ Long Binary

คอนโทรลแสดงและรับตัวอักษรแบบควบคุม (Masked Edit)

ใช้กับฟิลด์ประเภทต่าง ๆ โดยเราสามารถจัดการรูปแบบการแสดงผลและรับตัวอักษรกลับตามที่ต้องการ



รูปที่ 3.8 การออกแบบคอนโทรล

3.10 รูปแบบการใช้ฟังก์ชั่น Seek

Seek เป็นการหาข้อมูล หรือเรคคอร์ดที่ต้องการ โดยค้นหาจากดรรชนี แต่มีข้อจำกัดคือ จะใช้ได้กับออปเจ็คประเภทตารางเท่านั้น

ตัวอย่างรูปแบบการใช้

table_object.Index = index_name

table_object.Seek comparison_operate.comparison_value

index_name	คือตัวแปรค่าคงที่ที่เป็นชนิดตัวอักษร มีค่าเป็นชื่อของดรรชนี
comparison_operater	เป็นตัวแปรหรือค่าคงที่ชนิดตัวอักษร ระบุวิธีการเปรียบเทียบจะกำหนดได้เพียง =, <>, >, <, >=, =< เท่านั้น
comparison_value	เป็นตัวแปรชนิดของฟิลด์ที่ต้องการค้นหา

การอ่านค่าหรือกำหนดค่าของฟิลด์ในเรคคอร์ดในตำแหน่งนั้น

เมื่อระบุตำแหน่งที่ต้องการได้แล้ว เราจะอ่านค่าของฟิลด์ต่าง ๆ ในเรคคอร์ดตำแหน่งนั้นได้ หรือทำการเปลี่ยนแปลงแก้ไขค่าของฟิลด์ในตำแหน่งนั้นให้เป็นตามที่ต้องการ

วิธีง่ายต่อการระบุค่าที่อยู่ในฟิลด์ ทำได้โดยอ้างถึงชื่อฟิลด์ที่ต้องการ ด้วยรูปแบบดังนี้

recordset_object.field("field_name").value

ดังตัวอย่างนี้ เป็นการอ่านค่าหมายเลขพนักงาน (EmplD) ของเรคคอร์ดที่ตำแหน่งชี้ปัจจุบันของตาราง EmployeeTB

Current_EMP_Id = EmployeeTB.Fields("EmpID").value

วิธีการกำหนดค่าให้กับฟิลด์

หากมองว่าในออปเจ็คเร็คคอร์ดเซ็ท ก็เป็นเหมือนตัวแปรใด ๆ ตัวหนึ่ง วิธีการกำหนดค่าก็ทำได้ลักษณะเดียวกับตัวแปรชนิดอื่น ๆ ต่างกันเพียงวิธีการอ้างฟิลด์เท่านั้น

ดังรูปแบบ

recordset_object.Fields("fields_name").value = value

ตัวอย่างเช่น

EmployeeTB.Fields("Firstname").value = "สมชาย"

การเพิ่มข้อมูลใหม่

วิธีการเพิ่มข้อมูลใหม่เข้าไปในเรคคอร์ดเซ็ทนั้น ต้องเรียกใช้ฟังก์ชันการทำงานกับออปเจ็คเร็คคอร์ดเซ็ทนั้น ๆ 2 ฟังก์ชันด้วยกัน คือ AddNew และ Update โดยมีรูปแบบการใช้ฟังก์ชันดังนี้

recordset_object.AddNew เป็นการสั่งให้เพิ่มเร็คคอร์ดว่าง ๆ หนึ่งเรคคอร์ดแล้วตามด้วยคำสั่งสำหรับกำหนดค่าให้กับฟิลด์ต่าง ๆ ตามต้องการ

ดังตัวอย่างที่แสดง เป็นการเพิ่มข้อมูลตำแหน่งงานใหม่สำหรับ Programmer

```
JobTB.Addnew                          'สร้างเรคคอร์ดใหม่ในตาราง
    JobTB.Fields("JobCode")    =   "PRGM" 'กำหนดค่าฟิลด์
    JobTB.Fields("JobName")    =   "Programmer" ชื่อเรคคอร์ดใหม่
JobTB.Update                          'แก้ไขข้อมูลเป็นตามที่กำหนด
```

การแก้ไขข้อมูลเดิม

ทำได้โดยต้องเรียกการทำงานฟังก์ชัน 2 ฟังก์ชัน แต่ต้องเลื่อนตำแหน่งไปยังข้อมูลที่ต้องการแก้ไขเสียก่อน โดยอาจใช้ Move, Find, Bookmark

```
recordset_object.Edit                    'ประกาศว่าเริ่มแก้ไขข้อมูลที่ตำแหน่งตัวชี้
'กำหนดค่าให้กับฟิลด์ที่ต้องการเปลี่ยนค่า
recordset_object.Fields("Field_name") = new_value
.....
Recordset_object.update                  'สั่งให้เปลี่ยนข้อมูลในฐานข้อมูลได้
```

ตัวอย่างการใช้งาน จะแสดงส่วนคำสั่งที่ต้องการแก้ไขจากหน่วยความจำจาก DVLP ให้เป็น R&D ในทุก ๆ เรคคอร์ดของตาราง Job_History

```
JobhistoryTB.Findfirst = "DeptCode" = "DVLP"
Do While No JobhistoryTB.Nomatch
        JobHistoryTB.Edit
                JobhistoryTB.Fields("DeptCode") = "R&D"
JobhistoryTB.Update
JobhistoryTB.Findnext "DeptCode = 'DVLP"
Loop
```

การลบข้อมูล

ให้เรียกฟังก์ชันการทำงาน Delete กับเรคคอร์ดเซ็ทนั้น โดยเลื่อนตำแหน่งไปยังเรคคอร์ดเซ็ทที่ต้องการเสียก่อน โดยมีรูปแบบการใช้งานดังนี้

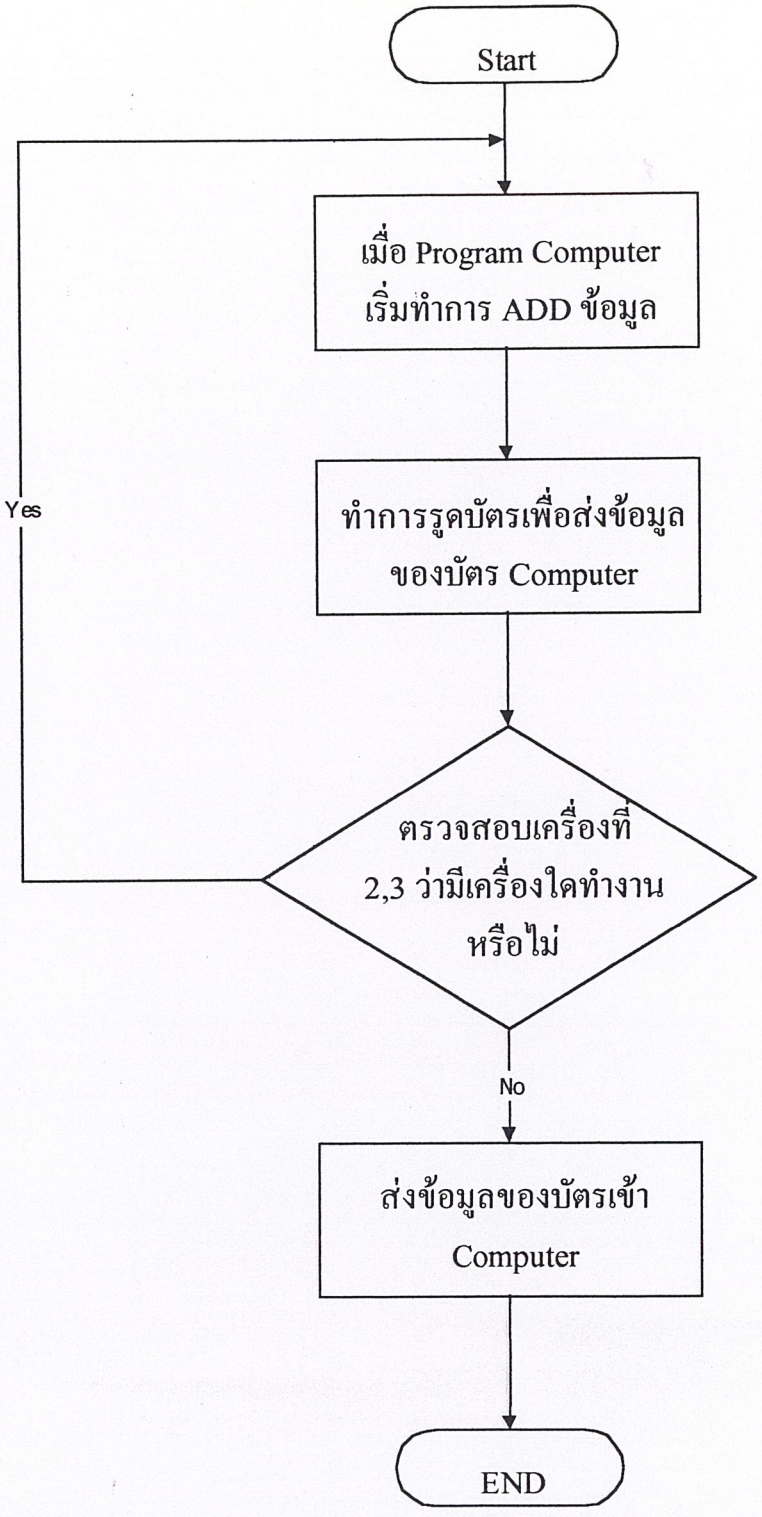
```
recordset_object.Delete
```

ตัวอย่างนี้เป็นการลบข้อมูลทั้งหมดใน Job_history ของพนักงานที่มี ID เท่ากับ 1

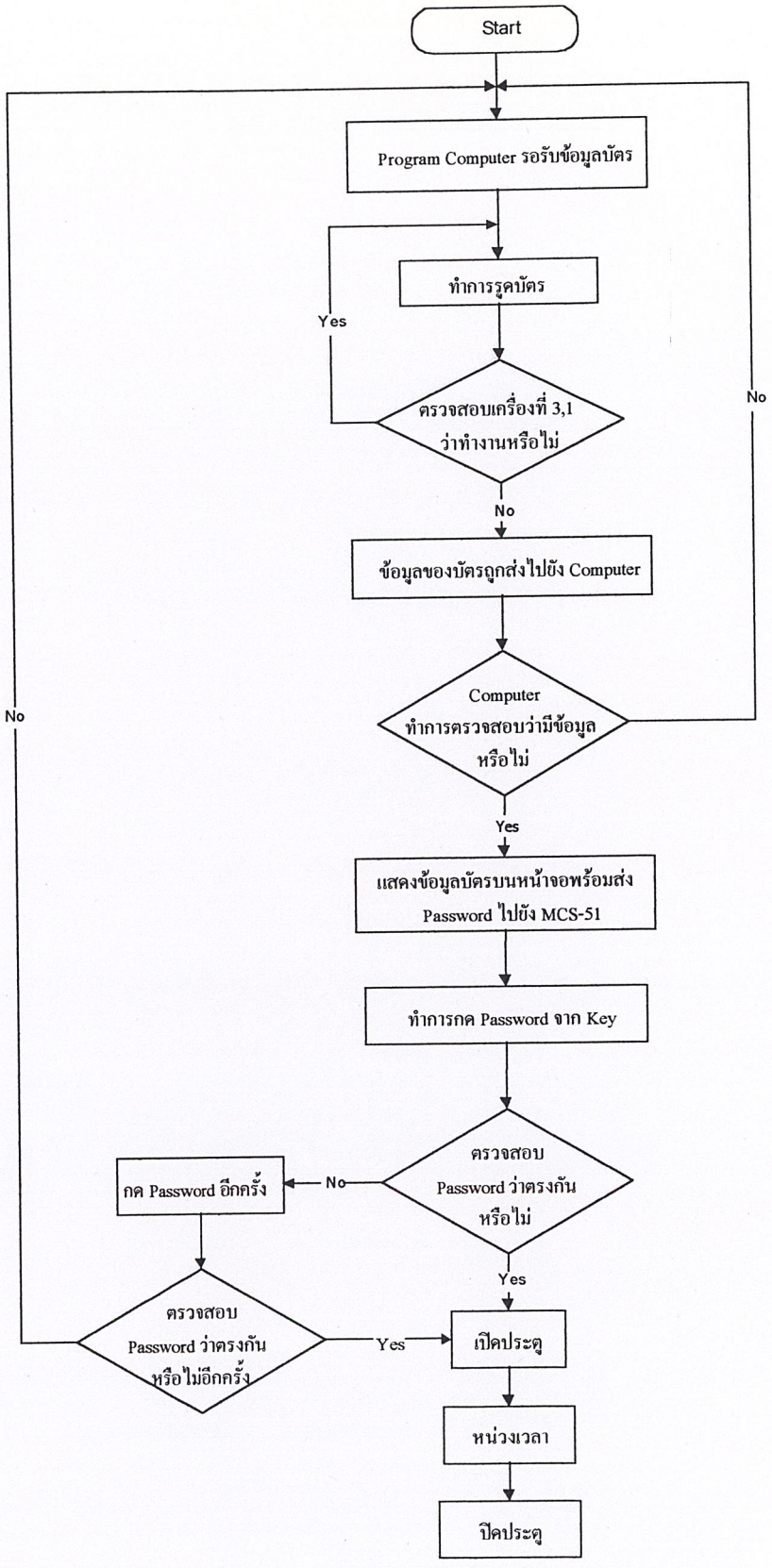
```
JobHistoryTB.FindFirst = "EmpID = 1"
Do While Not JobHistoryTB.EOF
JobHistoryTB.Delete
JobHistoryTB.FindNext = "EmpID = 1"
Loop
```

โปรแกรมเริ่มทำงานเมื่อมีข้อมูลส่งมาจากไมโครคอนโทรลเลอร์โดยทำการตรวจสอบข้อมูลในฐานข้อมูลว่าถูกต้องหรือไม่ เมื่อทำการตรวจสอบเสร็จสิ้นแล้ว หากข้อมูลถูกต้องโปรแกรมจะทำการบันทึกเวลาแล้วแสดงผล จากนั้นจะส่งสัญญาณตอบรับกลับไปยังไมโครคอนโทรลเลอร์ เมื่อสิ้นสุดการทำงานโปรแกรมควบคุมจะกลับสู่สภาวะการทำงานเริ่มต้นใหม่อีกครั้ง

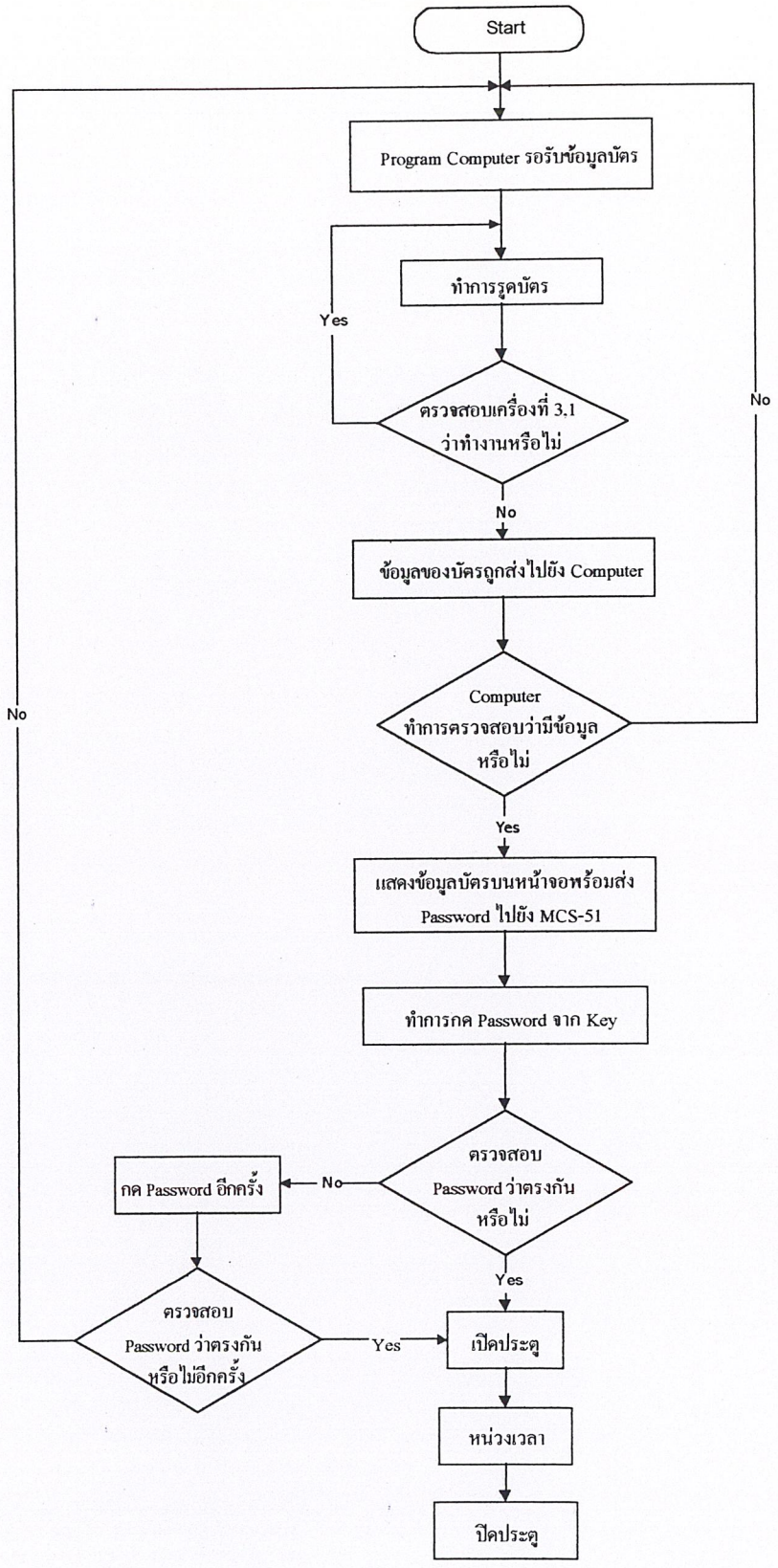
โปรแกรมเริ่มทำงาน เมื่อมีการตั้งค่าต่าง ๆ ในการสื่อสารแบบอนุกรม เพื่อรอรับรหัสสัญญาณจากการรูดบัตรแถบแม่เหล็ก จนเมื่อมีการรูดบัตรแถบแม่เหล็ก โปรแกรมจะทำการอ่านรหัสข้อมูลที่เข้ามาแล้วทำการแปลงเป็นรหัสแอสกี และส่งข้อมูลไปยังไมโครคอมพิวเตอร์ เพื่อทำการประมวลผลและรอรับสัญญาณตอบรับข้อมูลจากไมโครคอมพิวเตอร์เพื่อทำการควบคุมวงจรกำเนิดเสียงและวงจรควบคุมโซลินอยด์ จากนั้นไมโครคอนโทรลเลอร์ก็จะไปรอรับสัญญาณที่เกิดจากการรูดบัตรใหม่อีกครั้ง



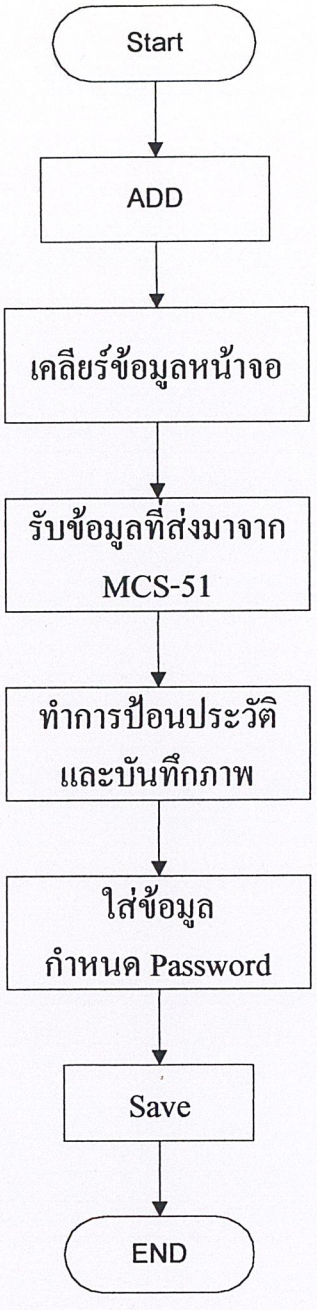
รูปที่ 3.9 Flow Chart แสดงขั้นตอนการทำงานของเครื่องอ่านบัตรเครื่องที่ 1



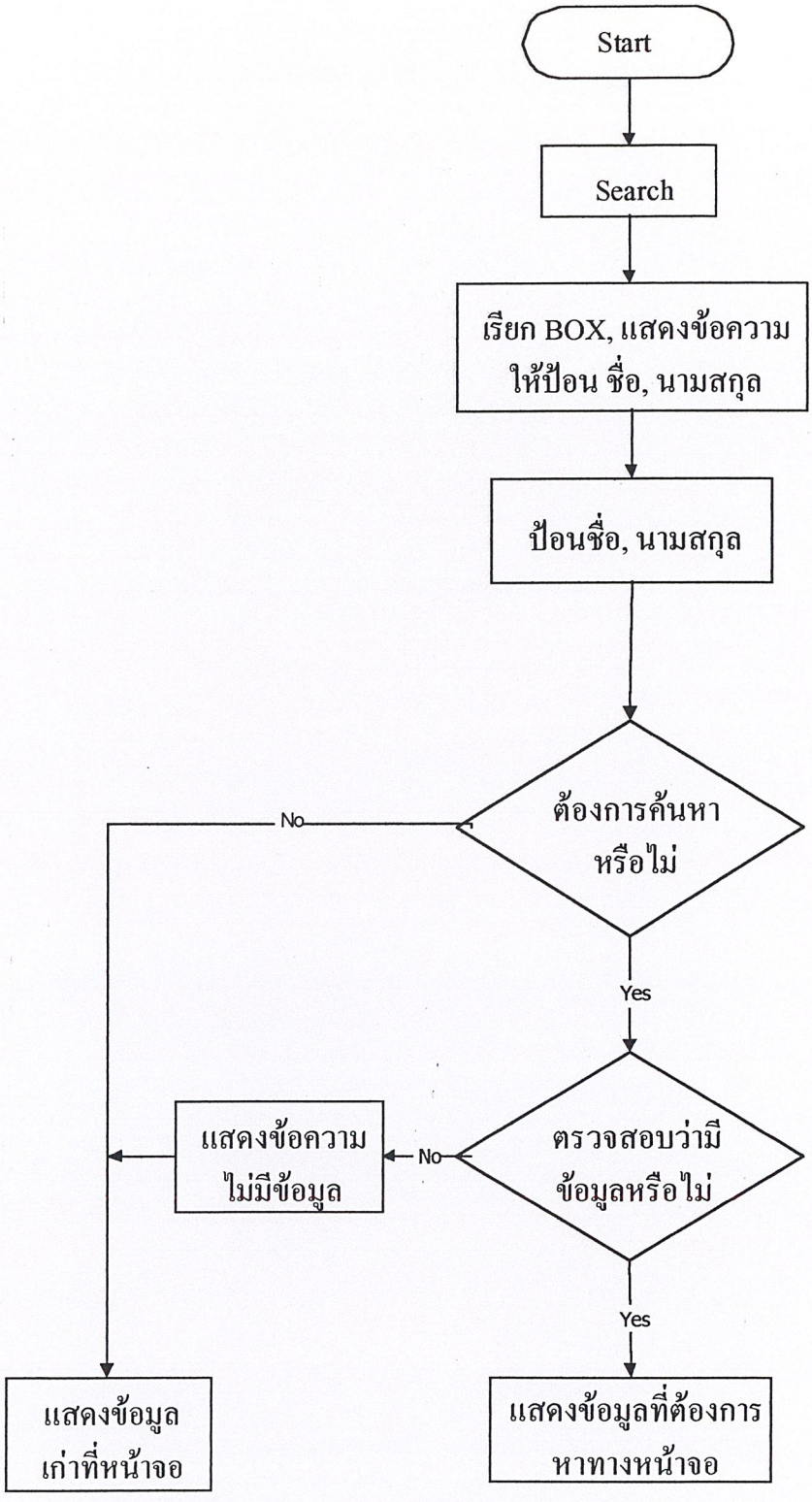
รูปที่ 3.10 Flow Chart แสดงขั้นตอนการทำงานของเครื่องอ่านบัตรเครื่องที่ 2



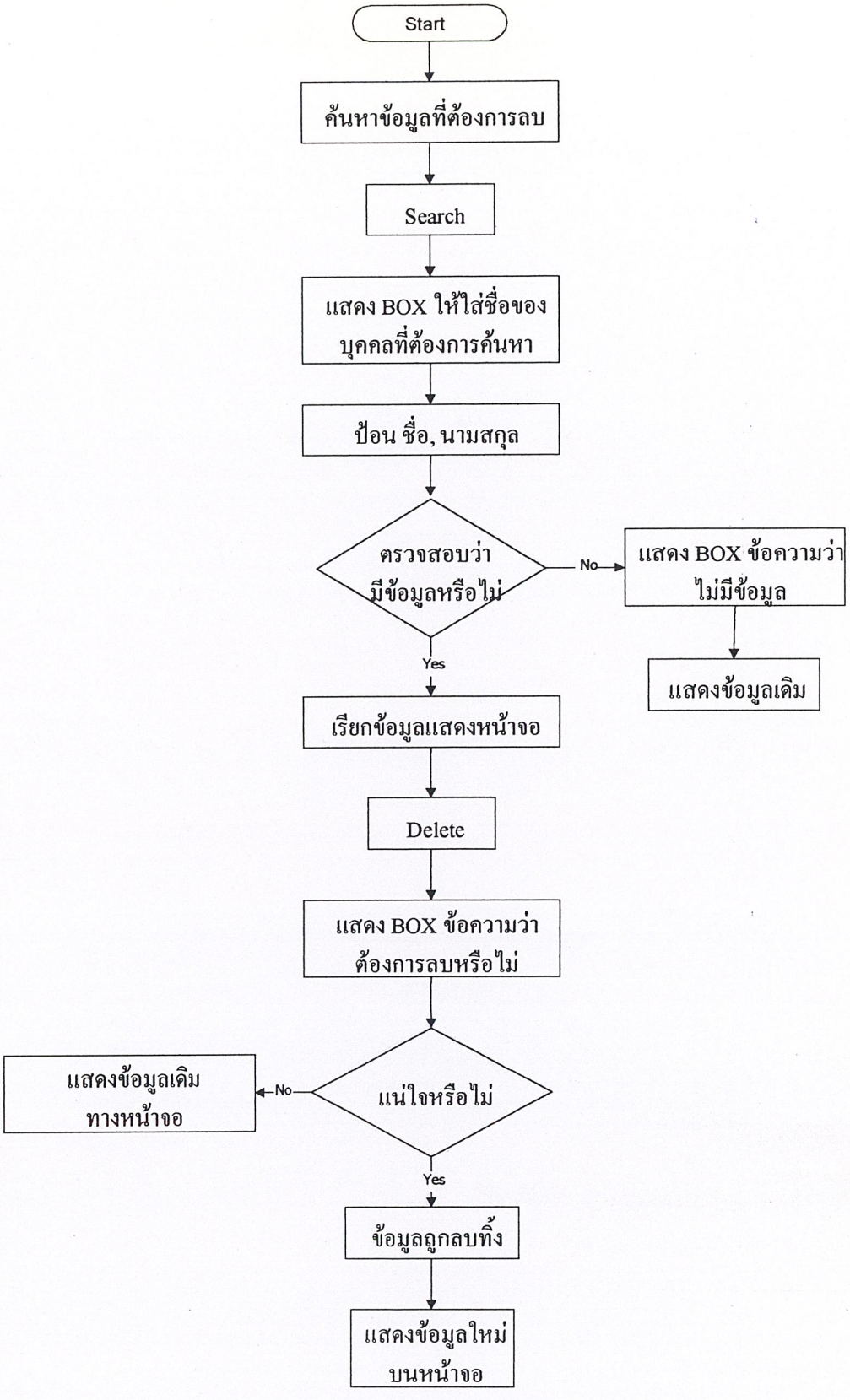
รูปที่ 3.11 Flow Chart แสดงขั้นตอนการทำงานของเครื่องอ่านบัตรเครื่องที่ 3



รูปที่ 3.12 Flow Chart แสดงขั้นตอนการ ADD ข้อมูล



รูปที่ 3.13 Flow Chart แสดงขั้นตอนการ Search ข้อมูล



รูปที่ 3.14 Flow Chart แสดงขั้นตอนการ Delete

บทที่ 4

ผลการทดลอง

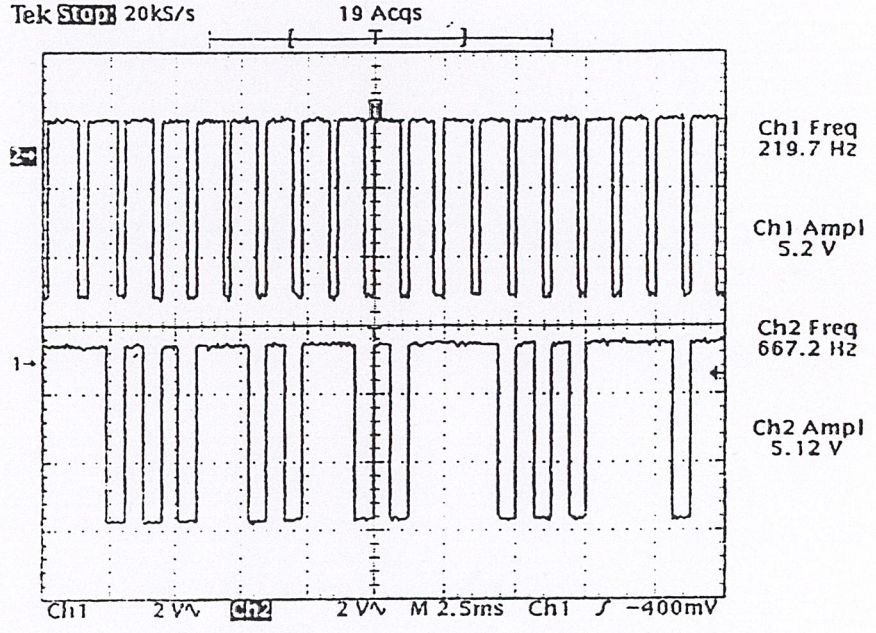
ในบทนี้จะกล่าวถึงวิธีการทดสอบและการประเมินผลจากการทดสอบ โดยจะแบ่งการทดสอบและผลจากการทดสอบ ดังนี้คือ

4.1 สัญญาณในส่วนของการอ่านบัตรแม่เหล็ก

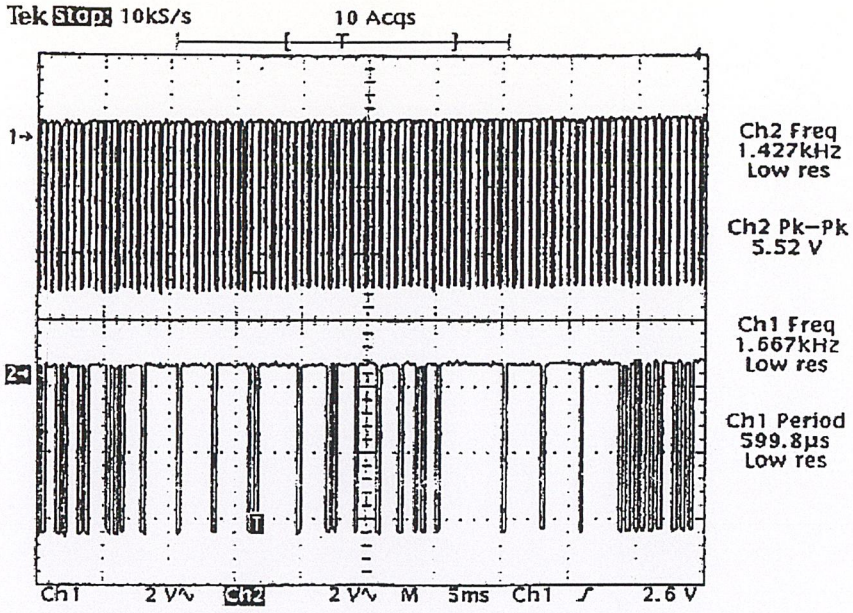
ลักษณะสัญญาณที่ได้จากการรูดบัตรแม่เหล็กผ่านหัวอ่านจะได้ลักษณะดังรูปที่ 4.1 และ 4.2 ซึ่งลักษณะที่ได้จะประกอบด้วยส่วน 3 ส่วนคือ

1. สัญญาณเริ่มต้นของการรูดบัตร (Start)
2. สัญญาณนาฬิกา (Clock)
3. สัญญาณข้อมูล (Data)

จากรูปจะเห็นได้ว่า ลักษณะของสัญญาณมีคาบเวลาไม่คงที่หรือก็คือมีความถี่ไม่คงที่ เนื่องจากการรูดบัตรด้วยมือนั้นมีความเร็วไม่คงที่ คือถ้าเรารูดบัตรเร็วสัญญาณที่ได้ก็จะมีค่าความถี่สูง ในทางตรงกันข้าม ถ้าเรารูดบัตรช้าสัญญาณที่ได้จะมีค่าความถี่ต่ำ แต่โดยปกติแล้ว ความเร็วในการรูดบัตรทั่วไปประมาณ 500-700 ไซเคิล/วินาที



รูปที่ 4.1 แสดงลักษณะสัญญาณของการรูดบัตรแม่เหล็กด้วยความเร็วปกติ



รูปที่ 4.2แสดงลักษณะของการรูดบัตรแม่เหล็กด้วยความเร็วสูง

4.2 ขั้นตอนการทำงาน

ในหัวข้อนี้จะอธิบายถึงขั้นตอนการทำงานของระบบรักษาความปลอดภัย โดยมีขั้นตอนดังนี้

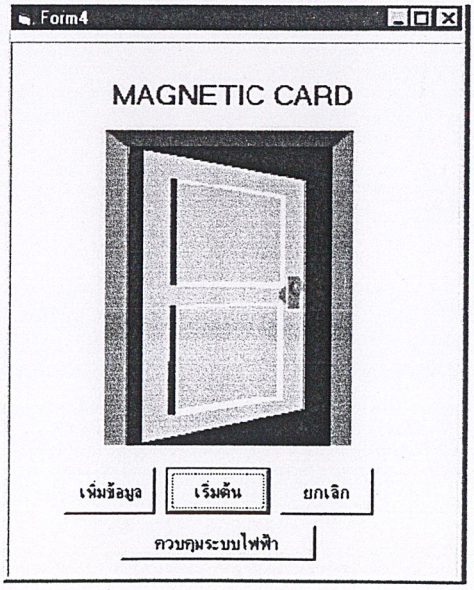
1. ทำการ Run Program เพื่อเข้าไปสู่ Menu ซึ่งมี Menu ดังนี้

   1.1 ปุ่มเพิ่มข้อมูลใช้สำหรับเพิ่มข้อมูลของบัตรลง Computer

   1.2 ปุ่มเริ่มต้นใช้สำหรับควบคุมการเข้าภายในอาคาร

   1.3 ปุ่มควบคุมอุปกรณ์ไฟฟ้าผ่านทาง Computer

   1.4 ปุ่มออกจาก Program



รูปที่ 4.3 แสดงเมนูหลักของโปรแกรม

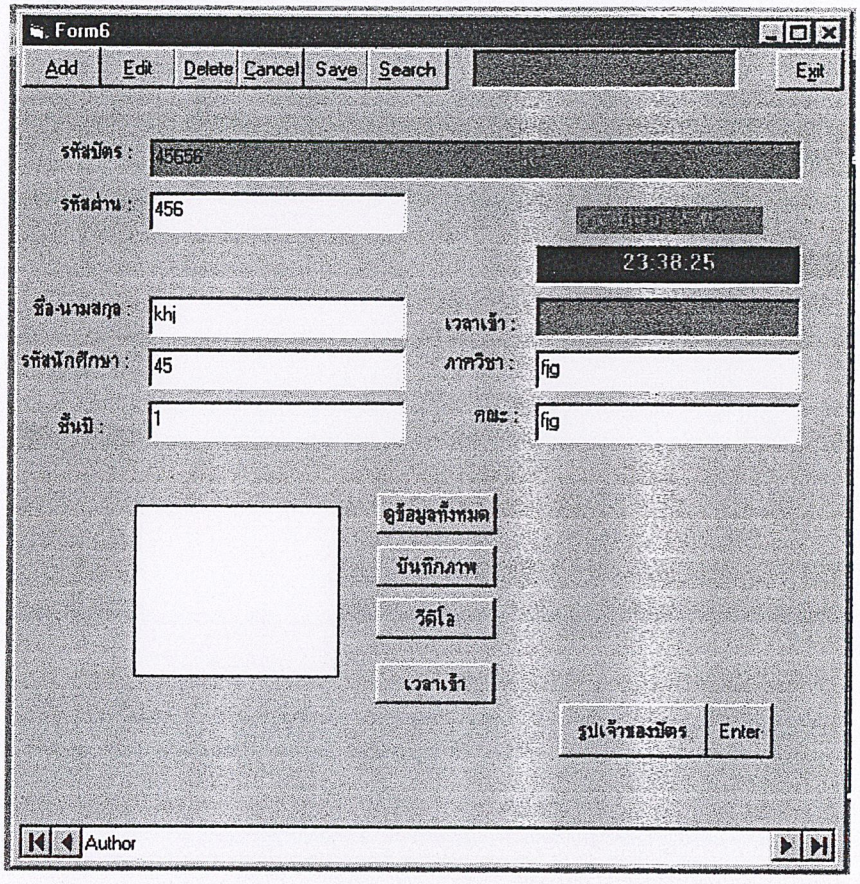
2. ทำการเพิ่มข้อมูลในกรณีที่บัตรแถบแม่เหล็ก ยังไม่มีการลงข้อมูลใน Computer ซึ่งมีขั้นตอนการเพิ่มข้อมูล ดังนี้

   2.1 กดปุ่ม ADD เพื่อทำการเคลียร์ข้อมูลหน้าจอและทำการถ่ายภาพผ่านทางกล้องดิจิตอล

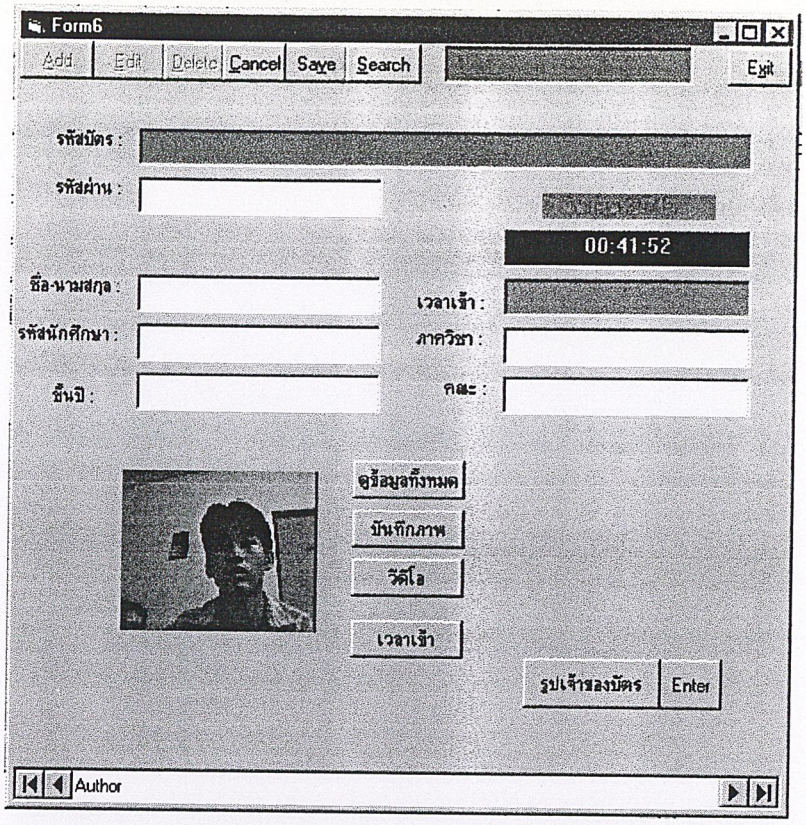
   2.2 ทำการรูดบัตรและเขียนข้อมูลของเจ้าของบัตรลงใน Computer

   2.3 ทำการกดบันทึกของเจ้าของบัตร

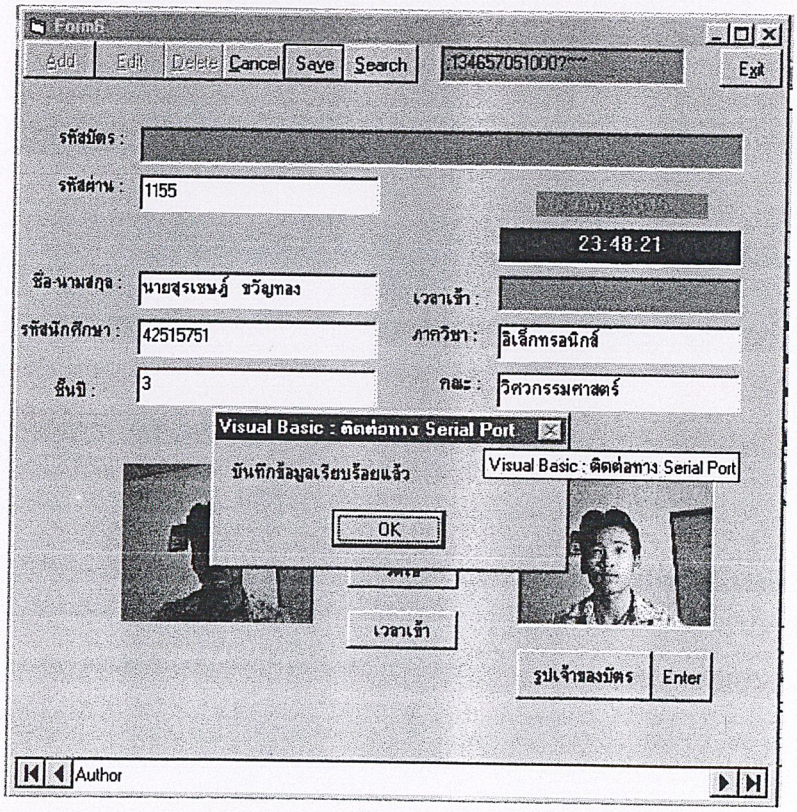
   2.4 ทำการ Save ข้อมูล



รูปที่ 4.4 แสดงหน้าจอแรกหลังจากเลือกเมนูเพิ่มข้อมูล



รูปที่ 4.5 แสดงการทำงานหลังจากกดปุ่ม ADD ข้อมูล

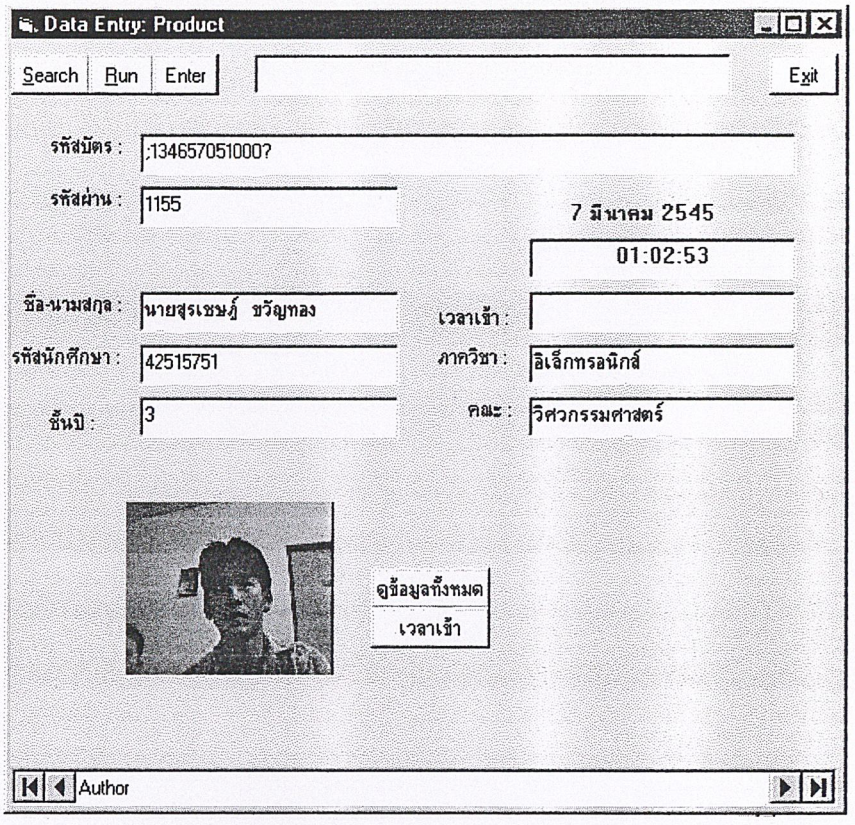


รูปที่ 4.6 แสดงการทำงานหลังจาก Save ข้อมูลเรียบร้อยแล้ว

3. แสดงขั้นตอนการทำงานเมื่อทำการเลือกเมนูเริ่มต้น

เมื่อเลือกเมนูเริ่มต้นเป็นเมนูสำหรับควบคุมการเข้าภายในตัวอาคาร โดยมีหลักการทำงานดังนี้

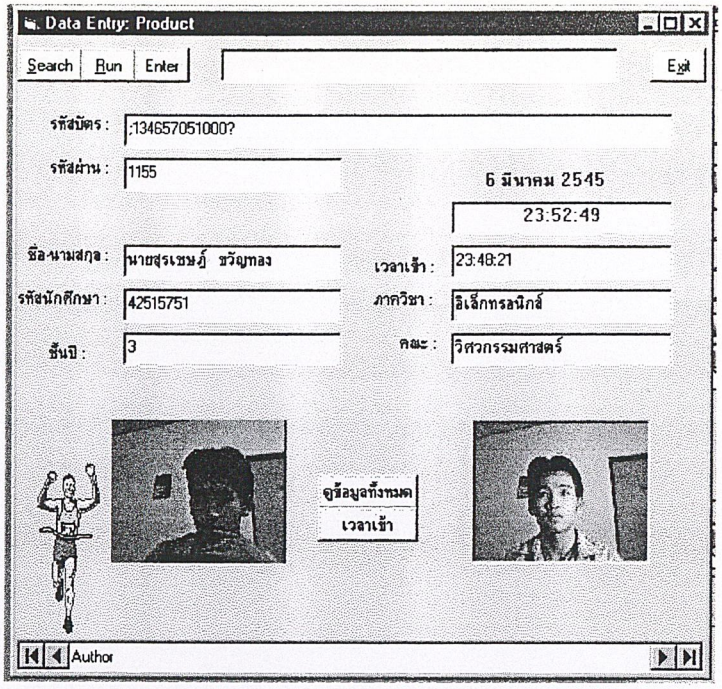
3.1 หลังจากทำการเลือกเมนูเพิ่มข้อมูลแล้วจะแสดงหน้าจอดังนี้



รูปที่ 4.7 แสดงการทำงานหลังจากเลือกเมนูเริ่มต้น

3.2 เมื่อทำการรูดบัตรไมโครคอนโทรลเลอร์จะส่งข้อมูลของบัตรโดยผ่านทาง RS-232C เข้ามายัง Computer หลังจากนั้นโปรแกรมจะทำการตรวจสอบข้อมูลของบัตรว่ามีข้อมูลของบัตรอยู่ในฐานข้อมูลหรือไม่ ถ้าหากมีข้อมูลโปรแกรมจะทำการแสดงข้อมูลที่มีในฐานข้อมูล

3.3 ทำการกด Password จาก คีย์บอร์ด Password จะถูกส่งมายัง Computer หลังจากนั้น Computerจะทำการตรวจสอบ Password ว่าตรงกับฐานข้อมูลหรือไม่ เมื่อตรวจสอบเสร็จแล้ว Computer จะส่งสัญญาณไปยัง ไมโครคอนโทรลเลอร์

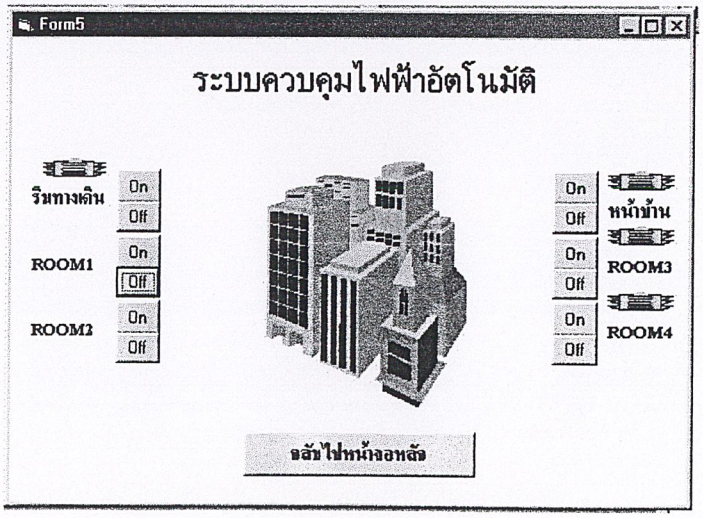


รูปที่ 4.8 แสดงหน้าจอหลังจากการรูดบัตร

3.4 เมื่อ Computer ทำการตรวจเช็ค Password ว่าถูกต้องหรือไม่ ถ้าหากถูกต้อง Computerจะส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ เพื่อทำการเปิดประตู หาก Password ไม่ถูกต้องไมโครคอนโทรลเลอร์ก็จะไม่ทำการเปิดประตู

4. เมื่อทำการเลือกเมนูควบคุมอุปกรณ์ไฟฟ้า

Computer จะแสดงหน้าจอการทำงานโดยให้เลือกเปิดสวิตซ์ตัวอุปกรณ์ เมื่อทำการกดสวิตซ์ โปรแกรมจะส่งสัญญาณไปยัง ไมโครคอนโทรลเลอร์เพื่อทำการควบคุมอุปกรณ์ไฟฟ้า



รูปที่ 4.9 แสดงระบบควบคุมอุปกรณ์ไฟฟ้า

บทที่ 5
บทสรุปปัญหาที่พบและแนวทางแก้ไข

5.1 บทสรุป

การทำงานของระบบรักษาความปลอดภัยโดยใช้บัตรแม่เหล็ก สามารถนำไปใช้งานได้ตรงตามวัตถุประสงค์ที่ตั้งไว้คือ สามารถประยุกต์ตัวไมโครคอนโทรลเลอร์ MCS-51 สื่อสารร่วมกับแบบอนุกรม RS-232C โดยมีหลักการทำงานของระบบ คือนำบัตรแถบแม่เหล็กมาทำการลงทะเบียน (บันทึกประวัติของเจ้าของบัตร) เมื่อผู้ถือบัตรต้องการเข้ามาภายในอาคาร ผู้ถือบัตรจะต้องนำบัตรแถบแม่เหล็กมาทำการรูดเข้ากับเครื่องอ่านบัตร เครื่องอ่านจะทำการอ่านข้อมูลบัตรแถบแม่เหล็กไปรูดกับเครื่องอ่านบัตรแม่เหล็ก เครื่องอ่านบัตรจะทำการอ่านข้อมูลของบัตร และส่งผ่านไมโครโทรลเลอร์ไปยัง Computer ซึ่งข้อมูลของบัตรถูกอ่านออกมาให้อยู่ในรูปของรหัสแอสกี้ Computer จะทำการตรวจสอบข้อมูลของบัตรและทำการแสดงประวัติของเจ้าของบัตรที่หน้าจอที่หน้าจอ Computer หลังจากนั้น Computer จะรอรับรหัสที่กดจาก Keyboard เมื่อกดเสร็จหลังนั้น Computer จะทำการถ่ายภาพของผู้ที่กดรหัสบัตรเอาไว้ พร้อมกับตรวจสอบรหัสว่าถูกต้องหรือไม่ เพื่อทำการส่งสัญญาณไปควบคุมโซลินอยด์

5.2 วิจารณ์ผลการทดลอง

จากการทำงานของระบบค่อข้างมีประสิทธิภาพ แต่การทำงานค่อนข้างช้าเหมาะกับการนำเอาไปใช้งานในการควบคุมการตรวจสอบบุคคลเข้ามาภายในโรงงาน ว่ามีใครเข้ามา เมื่อไร เวลาไหน

5.3 ปัญหาที่พบ

5.3.1 ปัญหาที่เกิดจากการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์ กับ ไมโครคอมพิวเตอร์ โดยผ่านการติดต่อสื่อสารมาตรฐาน RS-232C คือ

1. ข้อมูลที่ถูกส่งไปยัง Computer หากมีความเร็วในการส่งสูงข้อมูลที่ถูกส่งไปจะไม่ครบทุกตัว
2. หากมีการถ่ายภาพในขณะที่มีการส่งข้อมูลเข้าไปยัง Computer ข้อมูลบางตัวที่ส่งไปอาจสูญหายได้

5.3.2 ปัญหาที่เกิดขึ้นในกรณีที่เพิ่มจำนวนเครื่องอ่านบัตรให้มากขึ้น

1. ปัญหาในการส่งข้อมูลของ RS-232C เมื่อมีการรูดบัตรแต่ละเครื่องพร้อมกัน ข้อมูลที่ถูกส่งมาของแต่ละเครื่องจะแย่งกันเข้ามายัง Computer (คือ ตัวไหนมาก่อนจะถูกส่งเข้าไปก่อนทำให้การทำงานของระบบเกิดความผิดพลาด

2. ปัญหาในการรับข้อมูล RS-232C ไม่สามารถ Jump สาย Rx ของแต่ละตัวด้วยกันได้ เพราะว่า ข้อมูลที่ถูกส่งเข้ามาในครั้งสุดท้ายจะถูกเก็บเอาไว้ รอจนกว่าจะมีการนำข้อมูลครั้งสุดท้ายไปใช้งาน

5.4 แนวทางการแก้ปัญหา

5.4.1 แนวทางแก้ปัญหาของที่เกิดจากการติดต่อสื่อสารมาตรฐาน RS-232C ในกรณีโปรแกรมจะต้องส่งข้อมูลให้มีความเร็วในการส่งมากที่สุด โดยที่เมื่อส่งออกไปแล้ว ข้อมูลจะต้องไปสูญหาย โดยการกำหนดเวลาในการส่งข้อมูลแต่ละครั้ง

5.4.2 แนวทางแก้ปัญหาในกรณีที่เพิ่มจำนวนเครื่องอ่านบัตร

1. จะต้องเขียนโปรแกรมควบคุมการทำงานโดยกำหนดให้ ในขณะที่มีเครื่องใดเครื่องหนึ่งทำงานอยู่เครื่องอื่นก็จะไม่สามารถทำงานได้
2. จำเป็นต้องต่อ Relay ระหว่างสายที่เชื่อมต่อกับขา Rx ของ RS-232C

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี ก็ด้วยความเมตตาของ ศ.ดร. วัลลภ สุระกำพลธร และ ผศ. สุชาติ คูณทวีเทพ ที่ท่านช่วยชี้แนะแนวทางให้คำปรึกษา เอื้ออำนวยเรื่องอุปกรณ์การทดลองอีกทั้งยังช่วยเหลือในเรื่องอื่น ๆ อีกด้วย ขอขอบพระคุณอย่างสูง

นาย อนิรุทธ์ บัญชา
นาย ฐนวัฒน์ คงสงค์
นายสุรเชษฎ์ ขวัญทอง

ภาคผนวก ก

โปรแกรม Form1

```
Dim datacard2 As String
Private Sub Command1_Click()
Data2.Recordset.FindFirst "Password = '" & datacard2 & "'"
        If Data2.Recordset.NoMatch Then
        MsgBox "ไม่มีรหัสบัตรนี้ในฐานข้อมูลโปรแกรมจะกลับไปหน้าจอ แรก", vbOKOnly, "คำเตือน!!!"
        Exit Sub
        End If
        End Sub
        Private Sub Form_Load()
        Data2.DatabaseName = App.Path & "\Login.Mdb"
        Data2.RecordSource = "DataS"
        'datacard2 = InputBox("ป้อนรหัส(Password)ของผู้ที่ต้องการจะดูเวลา")
        End Sub
         Private Sub PrintCmnd_Click()
         Dim copy As Integer
        CommonDialog1.Flags = cdlPDHidePrintToFile Or cdlPDNoSelection Or cdlPDNoPageNums
Or cdlPDCollate
        CommonDialog1.CancelError = True
        CommonDialog1.PrinterDefault = True
        CommonDialog1.Copies = 1
        End Sub
```

โปรแกรม Form2

```
Dim datacard2 As String
Private Sub Command1_Click()
Data2.Recordset.FindFirst "Password = '" & datacard2 & "'"
        If Data2.Recordset.NoMatch Then
        'MsgBox "ไม่มีรหัสบัตรนี้ในฐานข้อมูลโปรแกรมจะกลับไปหน้าจอ แรก", vbOKOnly, "คำเตือน!!!"
        Exit Sub
        End If
        End Sub
        Private Sub Form_Load()
        Data1.DatabaseName = App.Path & "\Login.Mdb"
        Data1.RecordSource = "Login"
        'datacard2 = InputBox("ป้อนรหัส(Password)ของผู้ที่ต้องการจะดูเวลา")
        End Sub
```

โปรแกรม Form3

```
Private Sub Command1_Click()
Image1.Picture = LoadPicture(App.Path & "\" & Text5.Text)
Image2.Picture = LoadPicture(App.Path & "\" & Text7.Text)
End Sub

Private Sub Command2_Click()
    Dim Data As New Form4
          Hide
    'ปิดฟอร์ม
   Unload Me
   Data.Show
End Sub

Private Sub Form_Load()
 If Text5.Text = "" Then
      Image1.Visible = False
      Command1.Visible = False
      Else
      Image1.Visible = True
      Command1.Visible = True
      End If
End Sub

Private Sub Text5_Change()
If Text7.Text = "" Then
      Image1.Visible = False
      Command1.Visible = False
       On Error GoTo HandleError
       ProductDyn.Delete
HandleError:
```

```
    Else
        Image1.Visible = True
    Command1.Visible = True
    End If
End Sub
Private Sub Text7_Change()
If Text5.Text = "" Then
    Image1.Visible = False
    Command1.Visible = False
     On Error GoTo HandleError
     'ลบข้อมูลเรคคอร์ดปัจจุบัน
     ProductDyn.Delete
     'ไปที่เรคคอร์ดแรกและแสดงข้อมูลจากในตารางบนหน้าจอ
HandleError:
  'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
     Else
        Image1.Visible = True
    Command1.Visible = True
    End If
End Sub
```

โปรแกรม Form4

```
'Main Menu
  Option Explicit
        Private Sub Command1_Click()
         Dim start As New FormM
          Hide
          start.Show
       End Sub
       Private Sub Command2_Click()
          Dim Data As New Form6
           Hide
           Data.Show
       End Sub
          Private Sub Command3_Click()
         End
       End Sub
Private Sub Command4_Click()
Dim Data1 As New Form5
          Hide
          Data1.Show
End Sub
Private Sub Command5_Click()
Dim Data1 As New Form3
          Hide
          Data1.Show
End Sub
```

โปรแกรม Form 5

```
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim ChkStopLoopUnLess As Boolean
Private Sub Command1_Click(KeyAscii As Integer)
Dim Ve As Single
Image2.Visible = True
  Ve = A
  Text12.Text = Format(Ve, "A")
  MSComm1.Output = Text12.Text
   End Sub
Private Sub Command13_Click()
End
End Sub
Private Sub Command2_Click()
Dim Ve As Single
Image2.Visible = False
Ve = T
  Text12.Text = Format(Ve, "T")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command20_Click(KeyAscii As Integer)
Dim Ve As Single
Image2.Visible = False
Ve = Z
  Text12.Text = Format(Ve, "Z")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command3_Click(KeyAscii As Integer)
Dim Ve As Single
```

```
Image3.Visible = True
Ve = P
  Text12.Text = Format(Ve, "P")
  MSComm1.Output = Text12.Text
End Sub

Private Sub Command4_Click(KeyAscii As Integer)
Dim Ve As Single
Image3.Visible = False
Ve = V
  Text12.Text = Format(Ve, "V")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command50_Click(KeyAscii As Integer)
Dim Ve As Single
Image4.Visible = True
Ve = U
  Text12.Text = Format(Ve, "U")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command6_Click(KeyAscii As Integer)
Dim Ve As Single
Image4.Visible = False
Ve = i
  Text12.Text = Format(Ve, "I")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command7_Click(KeyAscii As Integer)
Dim Ve As Single
Image5.Visible = True
Ve = X
  Text12.Text = Format(Ve, "X")
```

```
  MSComm1.Output = Text12.Text
End Sub

Private Sub Command8_Click(KeyAscii As Integer)
Dim Ve As Single
Image5.Visible = False
Ve = J
  Text12.Text = Format(Ve, "J")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Command9_Click(Index As Integer)
Dim Ve As Single
Image6.Visible = True
Ve = E
  Text12.Text = Format(Ve, "E")
  MSComm1.Output = Text12.Text
End Sub

Private Sub Command10_Click(Index As Integer)
Dim Ve As Single
Image6.Visible = False
Ve = K
  Text12.Text = Format(Ve, "K")
  MSComm1.Output = Text12.Text
End Sub

Private Sub Command11_Click(Index As Integer)
Dim Ve As Single
Image7.Visible = True
Ve = O
  Text12.Text = Format(Ve, "O")
  MSComm1.Output = Text12.Text
```

```
End Sub

Private Sub Command12_Click(Index As Integer)
Dim Ve As Single
Image7.Visible = False
Ve = L
  Text12.Text = Format(Ve, "L")
  MSComm1.Output = Text12.Text
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
  '
  If MSComm1.PortOpen Then

    MSComm1.Output = Chr$(KeyAscii)

    If Not Echo Then

      Text1.SelStart = Len(Text12)
       KeyAscii = 0
    End If
  End If

End Sub

Private Sub Form_Load()
Image6.Visible = False
Image7.Visible = False
Image5.Visible = False
Image4.Visible = False
Image3.Visible = False
Image2.Visible = False
'ส่วนรูดบัตร
```

```
Dim CommPort As String, Handshaking As String, Settings As String
'On Error Resume Next

   App.Title = "Visual Basic : ติดต่อทาง Serial Port" 'ชื่อไตเติ้ลโปรแกรม

   'กำหนดค่าปกติของโปรแกรม
   ComOnOff = False          ' ปิดพอร์ต
    'ตั้งค่าในรีจิตเตอร์
   Settings = GetSetting(App.Title, "Properties", "Settings", "")
   If Settings <> "" Then
      MSComm1.Settings = Settings
   End If
   CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
   If CommPort <> "" Then MSComm1.CommPort = CommPort
      Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
   If Handshaking <> "" Then
      MSComm1.Handshaking = Handshaking
   End If
   Echo = GetSetting(App.Title, "Properties", "Echo", "")
   ChkStopLoopUnLess = False      'ตัวแปรสำหรับออกจากระบบ
         MSComm1.PortOpen = True
         ChkStopLoopUnLess = False
         ComOnOff = True          'on port
    Call optModeRXTX_Click(ModeRXTX)
'**************
   'เลือกใช้ Workspace(0)
   Set OrderWs = DBEngine.Workspaces(0)
   'เปิดฐานข้อมูล Order.Mdb
   Set OrderDb = OrderWs.OpenDatabase(App.Path & "\Login.Mdb", False, False)
   'เปิดตาราง Product โดยใช้เรคคอร์ดเซท ProductDyn
   Set ProductDyn = OrderDb.OpenRecordset("SELECT * FROM Login ORDER BY CardID", _
                              dbOpenDynaset)
```

```
    'ไปที่เรคคอร์ดแรกของเรคคอร์ดเซท ProductDyn
  ProductDyn.MoveFirst
  'แสดงข้อมูลเรคคอร์ดแรกบนหน้าจอ
   Exit Sub
End Sub
Private Sub MSComm1_OnComm()
        Dim Buffer As Variant
        Buffer = MSComm1.Input
       ShowData Text1, (StrConv((Buffer), vbUnicode))
        Exit Sub
     End Sub

Private Sub optModeRXTX_Click(Index As Integer)
   Select Case Index
   Case 0
      SecRXTX = 0
      If MSComm1.PortOpen = True Then MSComm1.RThreshold = 1
    End Select
End Sub
Public Static Sub ShowData(Term As Control, Data As String)
   Const MAXTERMSIZE = 16000
   Dim TermSize As Long, i
      TermSize = Len(Term.Text)
   If TermSize > MAXTERMSIZE Then
     Term.Text = Mid$(Term.Text, 4097)
     TermSize = Len(Term.Text)
   End If
   Term.SelStart = TermSize
   Do
     i = InStr(Data, Chr$(10))
     If i Then
       Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
```

```
    End If
  Loop While i
  i = 1
  Do
    i = InStr(i, Data, Chr$(13))
        If i Then
      Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
      i = i + 1
  '*** ส่วนที่จะทำอะไรต่อเมื่อรูดบัตรแล้ว
      '*********************************
    End If
  Loop While i
  Term.SelText = Data
Exit Sub
End Sub
Private Sub Text12_KeyPress(KeyAscii As Integer)
  If MSComm1.PortOpen Then
        MSComm1.Output = Chr$(KeyAscii)
        If Not Echo Then
          Text12.SelStart = Len(Text12)
       KeyAscii = 0
     End If
  End If
  End Sub
```

โปรแกรม Form6

```
'เพื่อใช้งานในทุก ๆ โปรแกรมย่อย
Public OrderWs          As Workspace
Public OrderDb          As Database
Public ProductDyn       As Recordset
Public CategorySnap     As Recordset
'*** ประกาศตัวแปรของ  ImageHandle สำหรับการ capture ภาพ
Public Hand As Long

'    ติดต่อกับ Serial Port
'-------------------------------------------------------
Dim Vp As Single
Dim Vt As Single
Dim Ve As Single
Dim endrun As Integer
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim npw As Integer
Dim ChkStopLoopUnLess As Boolean



Private Sub btnAdd_Click()
   On Error GoTo HandleError:
   Author.Recordset.AddNew
    'Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch ในขณะที่ทำการเพิ่มข้อมูล
   Call EnableOperationButton(False)
   'Enable ปุ่ม btnSave และ btnCancel
```

```
    Call EnableSaveCancelButton(True)
    'Clear หน้าจอเพื่อรอรับข้อมูลใหม่
    Call ClearData
    'เลื่อนเคอร์เซอร์ ไปที่ฟิล์ด บนหน้าจอเพื่อรอรับข้อมูล
    Text8.SetFocus

    'ใช้เมดทอด AddNew เพื่อเพิ่มข้อมูลเรคคอร์ดใหม่
    ProductDyn.AddNew

    '******


HandleError:
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
    'MsgBox Error(Err.Number)
End Sub

Private Sub Command3_Click()
        Dim start As New Form1
         Hide
         start.Show
    End Sub

Private Sub Command4_Click()
        Dim start As New Form2
         Hide
         start.Show
   End Sub

Private Sub EnterButton_Click()
Vp = Val(Text11.Text)
Vt = Val(Text8.Text)
```

```
Error:    If Vp = Vt Then
  Ve = 1
  Text12.Text = Format(Ve, "")
  MSComm1.Output = Text12.Text
   Text11.Text = ""
   Text11.SetFocus
 Else
 Ve = 2
Text12.Text = Format(Ve, "")

 If MSComm1.PortOpen Then

      MSComm1.Output = Text12.Text
     Text11.Text = ""
      Text11.SetFocus
      If Not Echo Then

        Text12.SelStart = Len(Text12)
         KeyAscii = 0

        End If
       End If
    End If
End Sub


Private Sub Text4_Change()
Timer2.Interval = 15000
Dim strProductId        As String
   Dim strRunCriteria As String
    'รับรหัสสินค้าที่ต้องการค้นหา
      strProductId = Text11.Text
```

```
'กำหนดกฎเกณฑ์ที่ต้องการค้นหา
  strRunCriteria = "CardID = '" & strProductId & "'"
'ค้นหาข้อมูลในเรคคอร์ดเซทโดยใช้เมดทอด FindFirst
  ProductDyn.FindFirst (strRunCriteria)
  If ProductDyn.NoMatch Then
  Timer2.Interval = 5000
  Text11.Text = ""
  Text11.SetFocus
        Else
    'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
    Call DisplayFields
    Timer2.Interval = 2000
    Text11.Text = ""
    Text11.SetFocus
   End If
 Exit Sub
End Sub

Private Sub Timer1_Timer()
  lblTime.Caption = Format(Time(), "HH:MM:SS")
    Label9.Caption = FormatDateTime(Date, vbLongDate)
 If EventMsg$ = " Receive " Then
    lblStatus.Caption = "Status Com : "
 End If

End Sub

Private Sub ClearData()
  Text1.Text = ""
 Text8.Text = ""
 Text2.Text = ""
 Text3.Text = ""
```

```
    Text10.Text = ""
    Text5.Text = ""
    Text6.Text = ""
    Text9.Text = ""
    Text11.Text = ""
End Sub

Private Sub btnCancel_Click()
    'ยกเลิกการเพิ่มเติมหรือแก้ไขข้อมูลเมื่ออยู่ในสถานะ AddNew และ Edit
    If ProductDyn.EditMode = dbEditAdd Or ProductDyn.EditMode = dbEditInProgress Then
      ProductDyn.CancelUpdate
    End If
    'Clear ฟิล์ดต่าง ๆ บนหน้าจอ
    Call ClearData
    'แสดงข้อมูลจากตารางบนหน้าจอ
    Call DisplayFields
    'Disable ปุ่ม btnSave และ btnCancel หลังจากยกเลิกการบันทึกข้อมูล
    Call EnableSaveCancelButton(False)
    'Enable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch หลังจากยกเลิกการบันทึกข้อมูล
    Call EnableOperationButton(True)
End Sub

Private Sub btnDelete_Click()
    On Error GoTo HandleError
    'ยืนยันการลบข้อมูล
    If MsgBox("ลบข้อมูลเรคคอร์ดนี้, แน่ใจหรือไม่?", [vbYesNo]) = vbYes Then
      'ลบข้อมูลเรคคอร์ดปัจจุบัน
      ProductDyn.Delete
      'ไปที่เรคคอร์ดแรกและแสดงข้อมูลจากในตารางบนหน้าจอ
      Call btnMoveFirst_Click
    End If
    Exit Sub
```

```
HandleError:
   'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
   MsgBox Error(Err.Number)
End Sub

Private Sub btnEdit_Click()
   On Error GoTo HandleError

   'Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch ในขณะที่ทำการแก้ไขข้อมูล
   Call EnableOperationButton(False)
   'Enable ปุ่ม btnSave และ btnCancel
   Call EnableSaveCancelButton(True)

   'เลื่อนเคอร์เซอร์ ไปที่ฟิลด์ Product Id บนหน้าจอเพื่อรอรับข้อมูล
   Text1.SetFocus

   'ใช้เมดทอด AddNew เพื่อเพิ่มข้อมูลเรคคอร์ดใหม่
   ProductDyn.Edit
   Exit Sub

HandleError:
   'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
   MsgBox Error(Err.Number)
End Sub

Private Sub btnExit_Click()
    Dim Data As New Form4
        Hide
       'ปิดเรคคอร์ดเซท
   ProductDyn.Close
   'ปิดฐานข้อมูล
```

```
    OrderDb.Close
    'ปิดฟอร์ม
    Unload Me
    Data.Show
End Sub


Private Sub btnMoveFirst_Click()
    'ไปเรคคอร์ดแรก
    ProductDyn.MoveFirst
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields
        'Enable btnMoveLast และ btnMoveNext
    btnMoveLast.Enabled = True
    btnMoveNext.Enabled = True
        'Disable btnMoveFirst และ btnMovePrevious
    btnMoveFirst.Enabled = False
    btnMovePrevious.Enabled = False
End Sub

Private Sub btnMoveLast_Click()
    'ไปเรคคอร์ดสุดท้าย
    ProductDyn.MoveLast
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields
    'Enable ปุ่ม btnMoveFirst และ btnMovePrevious
    btnMoveFirst.Enabled = True
    btnMovePrevious.Enabled = True
     'Disable ปุ่ม btnMoveLast และ btnMoveNext
    btnMoveLast.Enabled = False
    btnMoveNext.Enabled = False
End Sub
```

```
Private Sub btnMoveNext_Click()
  'ไปเรคคอร์ดถัดไป
  ProductDyn.MoveNext
  'ถ้าไม่ใช่เรคคอร์ดสุดท้าย
  If ProductDyn.EOF = False Then
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields
      btnMoveFirst.Enabled = True
    btnMovePrevious.Enabled = True
  Else
    'Disable ปุ่ม btnMoveLast และ btnMoveNext
    btnMoveLast.Enabled = False
    btnMoveNext.Enabled = False
  End If
End Sub

Private Sub btnMovePrevious_Click()
  'ไปเรคคอร์ดก่อนหน้านี้
  ProductDyn.MovePrevious
  'ถ้าไม่ใช่เรคคอร์ดแรก
  If ProductDyn.BOF = False Then
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields
     btnMoveLast.Enabled = True
    btnMoveNext.Enabled = True
  Else
     btnMoveFirst.Enabled = False
     btnMovePrevious.Enabled = False
  End If
End Sub
```

```
Private Sub btnSave_Click()
  On Error GoTo HandleError
'    'กำหนดค่าฟิลด์ VatFlag ตามค่า CheckBox chkVat บนหน้าจอ
'**********
      ProductDyn("CardID") = Text11.Text
      ProductDyn("Password") = Text8.Text
      ProductDyn("Date") = Label9.Caption
      ProductDyn("Name") = Text2.Text
      ProductDyn("StudentID") = Text3.Text
      ProductDyn("Time") = lblTime.Caption
      ProductDyn("Year") = Text5.Text
      ProductDyn("Dep") = Text6.Text
      ProductDyn("Fac") = Text9.Text
      ProductDyn("Picture") = Text7.Text
      'ใช้เมดทอด Update เพื่อบันทึกข้อมูลลงในเรคคอร์ดเซท
      ProductDyn.Update
      'จัดเรียงข้อมูลในเรคคอร์ดเซทใหม่
      ProductDyn.Requery
      MsgBox "บันทึกข้อมูลเรียบร้อยแล้ว"
      'Disable ปุ่ม btnSave และ btnCancel หลังจากบันทึกข้อมูลเรียบร้อยแล้ว
      Call EnableSaveCancelButton(False)
      'Enable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch หลังจากบันทึกข้อมูลเรียบร้อยแล้ว
      Call EnableOperationButton(True)
   ' End If
    Exit Sub

HandleError:
   'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
   MsgBox Error(Err.Number)
End Sub
```

```
Private Sub btnSearch_Click()
   Dim strProductId        As String
   Dim strSearchCriteria As String
     'รับรหัสสินค้าที่ต้องการค้นหา
   strProductId = InputBox("กรุณาใส่รหัสสินค้าที่ต้องการค้นหา")
       'กำหนดกฎเกณฑ์ที่ต้องการค้นหา
      strSearchCriteria = "Name= '" & strProductId & "'"
      'ค้นหาข้อมูลในเรคคอร์ดเซทโดยใช้เมดทอด FindFirst
      ProductDyn.FindFirst (strSearchCriteria)
      'ถ้าค้นหาไม่พบให้แสดงข้อความบอก Error
      If ProductDyn.NoMatch Then
        MsgBox "ไม่พบรหัสสินค้าที่ต้องการ, กรุณาใส่รหัสสินค้าที่ถูกต้องอีกครั้ง"
      Else
        'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
        Call DisplayFields

    End If
    Exit Sub
    End Sub
Private Sub btnRun_Click()
    Dim strProductId        As String
    Dim strRunCriteria As String
       strProductId = Text11.Text
       'กำหนดกฎเกณฑ์ที่ต้องการค้นหา
      strRunCriteria = "CardID = '" & strProductId & "'"
      'ค้นหาข้อมูลในเรคคอร์ดเซทโดยใช้เมดทอด FindFirst
      ProductDyn.FindFirst (strRunCriteria)
       'ถ้าค้นหาไม่พบให้แสดงข้อความบอก Error
      If ProductDyn.NoMatch Then
     Text11.Text = ""
     Text11.SetFocus
          Else
```

```
        'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
        Call DisplayFields
        Text11.Text = ""
        Text11.SetFocus

    End If
    Exit Sub
    End Sub
Private Sub btnRun1_Click()
    Dim strProductId        As String
    Dim strRunCriteria As String
       strProductId = Text1.Text
        strRunCriteria = "CardID = '" & strProductId & "'"
       ProductDyn.FindFirst (strRunCriteria)
                If ProductDyn.NoMatch Then
                  Else
                  Call DisplayFields
              End If
     Exit Sub
     End Sub

 Private Sub Command1_Click()
 Dim picName As String
 'VideoOCX.ReleaseImageHandle ImageHandle
 Image1.Visible = True
 'ให้รหัสนักศึกษาเป็นชื่อไฟล์รูปภาพ
 picName = Text3.Text
 'กำหนดให้ นำรูปที่ capture แล้วมาเป็นภาพนิ่งที่ Image1.picture

 '**** ส่วนการบันทึกภาพ เป็นนามสกุล *. jpg
 Video.start
  Han = Video.GetColorImageHandle
```

```
Video.Capture (Han)
If (Video.SaveJPEG(Han, 50, Text3.Text & ".jpg")) Then
 Image1.Picture = Video.Picture
 Text7.Text = Text3.Text & ".jpg"
 Video.Visible = True
 End If
End Sub

Private Sub Command2_Click()
Video.Visible = True
End Sub

Private Sub Form_Load()
 Video.Visible = True
    Video.Driver = 0
  If (Video.Init = 0) Then
     End If
  If (Video.SetPreview(True) = False) Then
     MsgBox (Video.GetLastErrorString())
  End If
'ส่วนรูดบัตร
Dim CommPort As String, Handshaking As String, Settings As String
     App.Title = "Visual Basic : ติดต่อทาง Serial Port" 'ชื่อไตเติ้ลโปรแกรม
        'ตั้งค่าในรีจิตเตอร์
  Settings = GetSetting(App.Title, "Properties", "Settings", "")
  If Settings <> "" Then
     MSComm1.Settings = Settings
  End If
  CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
  If CommPort <> "" Then MSComm1.CommPort = CommPort
     Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
  If Handshaking <> "" Then
```

```
    MSComm1.Handshaking = Handshaking
  End If
  Echo = GetSetting(App.Title, "Properties", "Echo", "")
  ChkStopLoopUnLess = False      'ตัวแปรสำหรับออกจากระบบ
                MSComm1.PortOpen = True
       ChkStopLoopUnLess = False
       ComOnOff = True          'on port
    Call optModeRXTX_Click(ModeRXTX)
'On Error GoTo 0  'ถ้าเกิดผิดผลาด
'**************
  '   'เลือกใช้ Workspace(0)
  Set OrderWs = DBEngine.Workspaces(0)
  'เปิดฐานข้อมูล Order.Mdb
  Set OrderDb = OrderWs.OpenDatabase(App.Path & "\Login.Mdb", False, False)
  'เปิดตาราง Product โดยใช้เรคคอร์ดเซท ProductDyn
  Set ProductDyn = OrderDb.OpenRecordset("SELECT * FROM DataS ORDER BY CardID", _
                        dbOpenDynaset)
   'ไปที่เรคคอร์ดแรกของเรคคอร์ดเซท ProductDyn
  ProductDyn.MoveFirst
    Call EnableSaveCancelButton(False)
    Exit Sub
End Sub

Private Sub DisplayFields()
     On Error GoTo HandleError
   '****
  Image1.Visible = True
  Text1.Text = ProductDyn.Fields("CardID")
  Text8.Text = ProductDyn.Fields("Password")
  Text2.Text = ProductDyn.Fields("Name")
  Text3.Text = ProductDyn.Fields("StudentID")
  Text10.Text = ProductDyn.Fields("Time")
```

```
    Text5.Text = ProductDyn.Fields("Year")
    Label9.Caption = ProductDyn.Fields("Date")
    Text6.Text = ProductDyn.Fields("Dep")
    Text9.Text = ProductDyn.Fields("Fac")
    Text7.Text = ProductDyn.Fields("Picture")
    Image1.Picture = LoadPicture(App.Path & "\" & Text7.Text)
        Exit Sub
HandleError:
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
    MsgBox Error(Err.Number)
End Sub

Private Sub EnableOperationButton(EnableFlag As Boolean)
    'Enable หรือ Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch
    btnAdd.Enabled = EnableFlag
    btnEdit.Enabled = EnableFlag
    btnDelete.Enabled = EnableFlag
    'btnSearch.Enabled = EnableFlag
End Sub

Private Sub EnableSaveCancelButton(EnableFlag As Boolean)
    'Enable หรือ Disable ปุ่ม btnSave และ btnCancel
    btnSave.Enabled = EnableFlag
    btnCancel.Enabled = EnableFlag
End Sub
' ส่วนประกอบ รูดบัตร ที่เหลือ
Private Sub UnLessLoop()
Dim Buffer
Dim Check As Boolean

MEVENT_DRIVEN:
    Check = True
```

```
Do
    If ChkStopLoopUnLess Then Exit Sub
    DoEvents
    If SecRXTX = 1 Then Check = False
Loop Until Check = False

If SecRXTX = 0 Then GoTo MEVENT_DRIVEN     'mode  Event-Driven

End Sub
Private Sub Text12_KeyPress(KeyAscii As Integer)
'
If MSComm1.PortOpen Then

    MSComm1.Output = Chr$(KeyAscii)

    If Not Echo Then

      Text12.SelStart = Len(Text12)
       KeyAscii = 0
    End If
End If

End Sub

Private Sub cmdOnOffPort_Click()
        MSComm1.PortOpen = True
        ChkStopLoopUnLess = False
        ComOnOff = True          'on port
    Call optModeRXTX_Click(ModeRXTX)
     Exit Sub
End Sub
```

```
Private Sub MSComm1_OnComm()

    Dim Buffer As Variant
    Buffer = MSComm1.Input
    ShowData Text11, (StrConv((Buffer), vbUnicode))

  End Sub
Private Sub optModeRXTX_Click(Index As Integer)
  Select Case Index
  Case 0
    SecRXTX = 0
    If MSComm1.PortOpen = True Then MSComm1.RThreshold = 1
  End Select
End Sub

Public Static Sub ShowData(Term As Control, Data As String)
'  On Error GoTo Handler
  Const MAXTERMSIZE = 16000
  Dim TermSize As Long, i

  TermSize = Len(Term.Text)
  If TermSize > MAXTERMSIZE Then
    Term.Text = Mid$(Term.Text, 4097)
    TermSize = Len(Term.Text)
  End If

  Term.SelStart = TermSize
   Do
    i = InStr(Data, Chr$(10))
    If i Then
      Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
    End If
```

```
  Loop While i

  i = 1
  Do
    i = InStr(i, Data, Chr$(13))

    If i Then
      Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
      i = i + 1
   '*** ส่วนที่จะทำอะไรต่อเมื่อรูดบัตรแล้ว
   '*********************************
    End If
  Loop While i

  Term.SelText = Data
  Exit Sub
End Sub


Private Sub Timer2_Timer()
Timer2.Interval = 2000
Text4.Text = Text11.Text
End Sub
```

โปรแกรม FormM

```
'ประกาศตัวแปรแบบ Workspace, Database และ Recordset
'เพื่อใช้งานในทุก ๆ โปรแกรมย่อย
Public OrderWs          As Workspace
Public OrderDb          As Database
Public ProductDyn       As Recordset
Public CategorySnap   As Recordset
'*** ประกาศตัวแปรของ  ImageHandle สำหรับการ capture ภาพ
Public Hand As Long
'    ติดต่อกับ Serial Port
'-------------------------------------------------------
Dim Vp As Single
Dim Vt As Single
Dim Ve As Single
Dim endrun As Integer
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim npw As Integer
Dim cir As Integer
 Dim strProductId  As String
Dim ChkStopLoopUnLess As Boolean
Private Sub btnAdd_Click()
  'On Error GoTo HandleError:
   Author.Recordset.AddNew
    'Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch ในขณะที่ทำการเพิ่มข้อมูล
   Call EnableOperationButton(False)
   'Enable ปุ่ม btnSave และ btnCancel
   Call EnableSaveCancelButton(True)
   'Clear หน้าจอเพื่อรอรับข้อมูลใหม่
```

```
Call ClearData
'เลื่อนเคอร์เซอร์ ไปที่ฟิล์ด บนหน้าจอเพื่อรอรับข้อมูล
Text8.SetFocus
'ใช้เมดทอด AddNew เพื่อเพิ่มข้อมูลเรคคอร์ดใหม่
ProductDyn.AddNew
'******
Video.Visible = True
  Video.Driver = 0
If (Video.Init = 0) Then
   ' An Error occured
   MsgBox (Video.GetLastErrorString())
End If
' Set Preview mode
If (Video.SetPreview(True) = False) Then
   ' An Error occured
   MsgBox (Video.GetLastErrorString())
End If
'  m_Video.ScaledDisplay = True
  Exit Sub

HandleError:
  'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
  MsgBox Error(Err.Number)
End Sub

Private Sub Command3_Click()
        Dim start As New Form1
         Hide
         start.Show
    End Sub

Private Sub Command4_Click()
```

```
        Dim start As New Form2
         Hide
         start.Show
  End Sub

Private Sub cr_Click()
Data1.Recordset.AddNew
End Sub

Private Sub EnterButton_Click()

cir = cir + 1
Vp = Val(Text11.Text)
Vt = Val(Text8.Text)
Error:
' ถ้า ใส่พาสเวิร์สผิด มากกว่า 2 ครั้ง สถานะของ endrun จะ รีเซ็ท
If cir >= 2 Then
    endrun = 0
    End If

If Vp = Vt Then
    endrun = 0
  Ve = 1
  Text12.Text = Format(Ve, "")
  MSComm1.Output = Text12.Text
   Text11.Text = ""
   Text11.SetFocus
  Else

 Ve = 2
Text12.Text = Format(Ve, "")
```

```
If MSComm1.PortOpen Then

    MSComm1.Output = Text12.Text
   Text11.Text = ""
    Text11.SetFocus
    If Not Echo Then

      Text12.SelStart = Len(Text12)
       KeyAscii = 0

      End If
     End If
  End If
End Sub


Private Sub Text4_Change()
Timer2.Interval = 15000
Dim strProductId        As String
  Dim strRunCriteria As String
  'รับรหัสสินค้าที่ต้องการค้นหา
  strProductId = Text11.Text
  '("กรุณาใส่รหัสสินค้าที่ต้องการค้นหา")
    'กำหนดกฎเกณฑ์ที่ต้องการค้นหา
    strRunCriteria = "CardID = '" & strProductId & "'"
    'ค้นหาข้อมูลในเรคคอร์ดเซทโดยใช้เมดทอด FindFirst
    ProductDyn.FindFirst (strRunCriteria)
          'ถ้าค้นหาไม่พบให้แสดงข้อความบอก Error
    If ProductDyn.NoMatch Then
    Timer2.Interval = 5000
   Text11.Text = ""
   Text11.SetFocus
```

```
        'MsgBox "ไม่พบรหัสสินค้าที่ต้องการ, กรุณาใส่รหัสสินค้าที่ถูกต้องอีกครั้ง"
      Else
        'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
        Call DisplayFields
        Timer2.Interval = 2000
        Text11.Text = ""
        Text11.SetFocus
    End If
    Exit Sub
End Sub
Private Sub stin_Click()
 On Error GoTo HandleError
      ProductDyn.Requery
      Call EnableSaveCancelButton(False)
      'Enable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch หลังจากบันทึกข้อมูลเรียบร้อยแล้ว
      Call EnableOperationButton(True)
    Exit Sub
HandleError:
  End Sub

Private Sub Text11_Change()
Dim Buffer As Variant
  If (Buffer = "?") Then
       Call btnRun_Click
         endrun = 1
        End If
End Sub



Private Sub Timer1_Timer()
    lblTime.Caption = Format(Time(), "HH:MM:SS")
```

```
    Label14.Caption = Format(Time(), "HH:MM:SS")
        Label9.Caption = FormatDateTime(Date, vbLongDate)
       Label10.Caption = FormatDateTime(Date, vbLongDate)
   If EventMsg$ = " Receive " Then
      lblStatus.Caption = "Status Com : "
   End If

End Sub

Private Sub ClearData()
     Text1.Text = ""
    Text8.Text = ""
    Text2.Text = ""
    Text3.Text = ""
    Text10.Text = ""
    Text5.Text = ""
    Text6.Text = ""
    Text9.Text = ""
    Text11.Text = ""
End Sub

Private Sub btnCancel_Click()
    'ยกเลิกการเพิ่มเติมหรือแก้ไขข้อมูลเมื่ออยู่ในสถานะ AddNew และ Edit
    If ProductDyn.EditMode = dbEditAdd Or ProductDyn.EditMode = dbEditInProgress Then
      ProductDyn.CancelUpdate
    End If
    'Clear ฟิล์ดต่าง ๆ บนหน้าจอ
    Call ClearData
    'แสดงข้อมูลจากตารางบนหน้าจอ
    Call DisplayFields
    'Disable ปุ่ม btnSave และ btnCancel หลังจากยกเลิกการบันทึกข้อมูล
    Call EnableSaveCancelButton(False)
```

```
    'Enable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch หลังจากยกเลิกการบันทึกข้อมูล
    Call EnableOperationButton(True)
End Sub

Private Sub btnDelete_Click()
    On Error GoTo HandleError
    'ยืนยันการลบข้อมูล
    If MsgBox("ลบข้อมูลเรคคอร์ดนี้, แน่ใจหรือไม่?", [vbYesNo]) = vbYes Then
        'ลบข้อมูลเรคคอร์ดปัจจุบัน
        ProductDyn.Delete
        'ไปที่เรคคอร์ดแรกและแสดงข้อมูลจากในตารางบนหน้าจอ
        Call btnMoveFirst_Click
    End If
    Exit Sub

HandleError:
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
    MsgBox Error(Err.Number)
End Sub

Private Sub btnEdit_Click()
    On Error GoTo HandleError
        'Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch ในขณะที่ทำการแก้ไขข้อมูล
    Call EnableOperationButton(False)
    'Enable ปุ่ม btnSave และ btnCancel
    Call EnableSaveCancelButton(True)
        'เลื่อนเคอร์เซอร์ ไปที่ฟิลด์ Product Id บนหน้าจอเพื่อรอรับข้อมูล
    Text1.SetFocus
        'ใช้เมดทอด AddNew เพื่อเพิ่มข้อมูลเรคคอร์ดใหม่
    ProductDyn.Edit
    Exit Sub
HandleError:
```

```
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
    MsgBox Error(Err.Number)
End Sub

Private Sub btnExit_Click()
    Dim Data As New Form4
        Hide
        'ปิดเรคคอร์ดเซท
'   CategorySnap.Close
    ProductDyn.Close
    'ปิดฐานข้อมูล
    OrderDb.Close

    'ปิดฟอร์ม
    Unload Me
    Data.Show
End Sub


Private Sub btnMoveFirst_Click()
    'ไปเรคคอร์ดแรก
    ProductDyn.MoveFirst
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields

    'Enable btnMoveLast และ btnMoveNext
    btnMoveLast.Enabled = True
    btnMoveNext.Enabled = True

    'Disable btnMoveFirst และ btnMovePrevious
    btnMoveFirst.Enabled = False
    btnMovePrevious.Enabled = False
```

```
End Sub

Private Sub btnMoveLast_Click()
   'ไปเรคคอร์ดสุดท้าย
   ProductDyn.MoveLast
   'แสดงข้อมูลบนหน้าจอ
   Call DisplayFields

   'Enable ปุ่ม btnMoveFirst และ btnMovePrevious
   btnMoveFirst.Enabled = True
   btnMovePrevious.Enabled = True

   'Disable ปุ่ม btnMoveLast และ btnMoveNext
   btnMoveLast.Enabled = False
   btnMoveNext.Enabled = False
End Sub

Private Sub btnMoveNext_Click()
   'ไปเรคคอร์ดถัดไป
   ProductDyn.MoveNext
      'ถ้าไม่ใช่เรคคอร์ดสุดท้าย
   If ProductDyn.EOF = False Then
      'แสดงข้อมูลบนหน้าจอ
      Call DisplayFields
            'Enable ปุ่ม btnMoveFirst และ btnMovePrevious
      btnMoveFirst.Enabled = True
      btnMovePrevious.Enabled = True
   Else
      'Disable ปุ่ม btnMoveLast และ btnMoveNext
      btnMoveLast.Enabled = False
      btnMoveNext.Enabled = False
   End If
```

```
End Sub

Private Sub btnMovePrevious_Click()
  'ไปเรคคอร์ดก่อนหน้านี้
  ProductDyn.MovePrevious
    'ถ้าไม่ใช่เรคคอร์ดแรก
  If ProductDyn.BOF = False Then
    'แสดงข้อมูลบนหน้าจอ
    Call DisplayFields
        'Enable ปุ่ม btnMoveLast และ btnMoveNext
    btnMoveLast.Enabled = True
    btnMoveNext.Enabled = True
  Else
    'Disable ปุ่ม btnMoveFirst และ btnMovePrevious
    btnMoveFirst.Enabled = False
    btnMovePrevious.Enabled = False
  End If
End Sub
Private Sub btnSave_Click()
  On Error GoTo HandleError
'    'กำหนดค่าฟิลด์ VatFlag ตามค่า CheckBox chkVat บนหน้าจอ
'**********
      ProductDyn("CardID") = Text11.Text
      ProductDyn("Password") = Text8.Text
      ProductDyn("Date") = Label9.Caption
      ProductDyn("Name") = Text2.Text
      ProductDyn("StudentID") = Text3.Text
      ProductDyn("Time") = lblTime.Caption
      ProductDyn("Year") = Text5.Text
      ProductDyn("Dep") = Text6.Text
      ProductDyn("Fac") = Text9.Text
      ProductDyn("Picture") = Text7.Text
```

```
    'ใช้เมดทอด Update เพื่อบันทึกข้อมูลลงในเรคคอร์ดเซท
    ProductDyn.Update
    'จัดเรียงข้อมูลในเรคคอร์ดเซทใหม่
    ProductDyn.Requery
    MsgBox "บันทึกข้อมูลเรียบร้อยแล้ว"
    'Disable ปุ่ม btnSave และ btnCancel หลังจากบันทึกข้อมูลเรียบร้อยแล้ว
    Call EnableSaveCancelButton(False)
    'Enable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch หลังจากบันทึกข้อมูลเรียบร้อยแล้ว
    Call EnableOperationButton(True)
  ' End If
  Exit Sub

HandleError:
  'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
  MsgBox Error(Err.Number)
End Sub

Private Sub btnSearch_Click()
  Dim strProductId       As String
  Dim strSearchCriteria As String

  'รับชื่อที่ต้องการค้นหา
     strProductId = InputBox("กรุณาใส่ชื่อที่ต้องการค้นหา")

  'กำหนดกฎเกณฑ์ที่ต้องการค้นหา
     strSearchCriteria = "Name= '" & strProductId & "'"

  'ค้นหาข้อมูลในเรคคอร์ดเซทโดยใช้เมดทอด FindFirst
     ProductDyn.FindFirst (strSearchCriteria)
  'ถ้าค้นหาไม่พบให้แสดงข้อความบอก Error
     If ProductDyn.NoMatch Then
      MsgBox "ไม่พบชื่อที่ต้องการ, กรุณาใส่ชื่อที่ถูกต้องอีกครั้ง"
```

```
        Else
  'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
        Call DisplayFields


    End If
    Exit Sub

'HandleError:
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
  '  MsgBox Error(Err.Number)
End Sub
Private Sub btnRun_Click()
     Dim strRunCriteria As String
        strProductId = Text11.Text
            strRunCriteria = "CardID = '" & strProductId & "'"
             ProductDyn.FindFirst (strRunCriteria)
             If ProductDyn.NoMatch Then
           Text11.Text = ""
      Text11.SetFocus
      Else
         'ถ้าค้นหาพบให้แสดงข้อมูลเรคคอร์ดนั้นบนหน้าจอ
         Call DisplayFields
         Text11.Text = ""
         Text11.SetFocus


    End If


    Exit Sub


End Sub
Private Sub btnRun1_Click()
    Dim strProductId        As String
```

```
Dim strRunCriteria As String
   strProductId = Text1.Text
    strRunCriteria = "CardID = '" & strProductId & "'"
   ProductDyn.FindFirst (strRunCriteria)
            If ProductDyn.NoMatch Then
            Else
             Call DisplayFields

 End If
 Exit Sub
 End Sub

Private Sub Command1_Click()
Dim picName As String
Image4.Visible = True
'ให้รหัสนักศึกษาเป็นชื่อไฟล์รูปภาพ
picName = Text8.Text
'กำหนดให้ นำรูปที่ capture แล้วมาเป็นภาพนิ่งที่ Image1.picture
'**** ส่วนการบันทึกภาพ เป็นนามสกุล *. jpg
Video.start
 Han = Video.GetColorImageHandle
 Video.Capture (Han)
 If (Video.SaveJPEG(Han, 200, Text8.Text & ".jpg")) Then
  Image4.Picture = Video.Picture
  Label15.Caption = Text8.Text & ".jpg"
  Video.Visible = True
  End If
End Sub

 Private Sub Command2_Click()
 Video.Visible = True
 End Sub
```

```
Private Sub Form_Load()
Call ClearData
 Image3.Visible = False
Image2.Visible = False
 Video.Visible = True
    Video.Driver = 0
  If (Video.Init = 0) Then
   '      MsgBox (Video.GetLastErrorString())
  End If

  ' Set Preview mode
  If (Video.SetPreview(True) = False) Then
     ' An Error occured
     MsgBox (Video.GetLastErrorString())
  End If
'  m_Video.ScaledDisplay = True

'End Sub
'ส่วนรูดบัตร
Dim CommPort As String, Handshaking As String, Settings As String
'On Error Resume Next

   App.Title = "Visual Basic : ติดต่อทาง Serial Port" 'ชื่อไตเติ้ลโปรแกรม

   'กำหนดค่าปกติของโปรแกรม
   ComOnOff = False          'ปิดพอร์ต
   optModeRXTX.Item(0).Value = True   'การทำอินเตอร์รัพ
   SecRXTX = 0
     'ตั้งค่าในรีจิตเตอร์
   Settings = GetSetting(App.Title, "Properties", "Settings", "")
   If Settings <> "" Then
```

```
    MSComm1.Settings = Settings
  End If
  CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
  If CommPort <> "" Then MSComm1.CommPort = CommPort
    Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
  If Handshaking <> "" Then
    MSComm1.Handshaking = Handshaking
  End If
  Echo = GetSetting(App.Title, "Properties", "Echo", "")
  ChkStopLoopUnLess = False      'ตัวแปรสำหรับออกจากระบบ
                MSComm1.PortOpen = True
        ChkStopLoopUnLess = False
        ComOnOff = True          'on port
   Call optModeRXTX_Click(ModeRXTX)
  'เลือกใช้ Workspace(0)
  Set OrderWs = DBEngine.Workspaces(0)
  'เปิดฐานข้อมูล Order.Mdb
  Set OrderDb = OrderWs.OpenDatabase(App.Path & "\Login.Mdb", False, False)
  'เปิดตาราง Product โดยใช้เรคคอร์ดเซท ProductDyn
  Set ProductDyn = OrderDb.OpenRecordset("SELECT * FROM DataS ORDER BY CardID", _
                        dbOpenDynaset)
  'ไปที่เรคคอร์ดแรกของเรคคอร์ดเซท ProductDyn
  ProductDyn.MoveFirst
  'แสดงข้อมูลเรคคอร์ดแรกบนหน้าจอ
  'Disable ปุ่ม btnSave และ btnCancel
  Call EnableSaveCancelButton(False)
    Exit Sub
End Sub

Private Sub DisplayFields()
  On Error GoTo HandleError
  '****
```

```
    Image1.Visible = True
    Text1.Text = ProductDyn.Fields("CardID")
    Text8.Text = ProductDyn.Fields("Password")
    Text2.Text = ProductDyn.Fields("Name")
    Text3.Text = ProductDyn.Fields("StudentID")
    Text10.Text = ProductDyn.Fields("Time")
    Text5.Text = ProductDyn.Fields("Year")
    Label9.Caption = ProductDyn.Fields("Date")
    Text6.Text = ProductDyn.Fields("Dep")
    Text9.Text = ProductDyn.Fields("Fac")
    Text7.Text = ProductDyn.Fields("Picture")
    Image1.Picture = LoadPicture(App.Path & "\" & Text7.Text)
    Exit Sub
HandleError:
    'แสดง Error Message เมื่อเกิดข้อผิดพลาดขึ้น
    MsgBox Error(Err.Number)
End Sub
Private Sub EnableOperationButton(EnableFlag As Boolean)
    'Enable หรือ Disable ปุ่ม btnAdd, btnEdit, btnDelete และ btnSearch
    btnAdd.Enabled = EnableFlag
    btnEdit.Enabled = EnableFlag
    btnDelete.Enabled = EnableFlag
    btnSearch.Enabled = EnableFlag
End Sub

Private Sub EnableSaveCancelButton(EnableFlag As Boolean)
    'Enable หรือ Disable ปุ่ม btnSave และ btnCancel
    btnSave.Enabled = EnableFlag
    ' btnCancel.Enabled = EnableFlag
End Sub
' ส่วนประกอบ รูดบัตร ที่เหลือ
Private Sub UnLessLoop()
```

```
Dim Buffer
Dim Check As Boolean
MEVENT_DRIVEN:
  Check = True
  Do
    If ChkStopLoopUnLess Then Exit Sub
    DoEvents
    If SecRXTX = 1 Then Check = False
  Loop Until Check = False

  If SecRXTX = 0 Then GoTo MEVENT_DRIVEN     'mode  Event-Driven

End Sub
Private Sub Text12_KeyPress(KeyAscii As Integer)
  '
  If MSComm1.PortOpen Then

    MSComm1.Output = Chr$(KeyAscii)

    If Not Echo Then

      Text12.SelStart = Len(Text12)
       KeyAscii = 0
    End If
  End If

End Sub

Private Sub cmdOnOffPort_Click()
        MSComm1.PortOpen = True
        ChkStopLoopUnLess = False
        ComOnOff = True          'on port
```

```
    Call optModeRXTX_Click(ModeRXTX)
      Exit Sub
End Sub
Private Sub MSComm1_OnComm()
      Dim Buffer As Variant
       Buffer = MSComm1.Input
       ShowData Text11, (StrConv((Buffer), vbUnicode))
    'เช็ค พาสเวิร์ด
        If endrun = 1 Then
    If (Buffer >= "0") And (Buffer <= "9") Then
          npw = npw + 1
          If npw >= 2 Then
                Call cr_Click
                   End If
            If npw >= 3 Then
                Text40.Text = Text7.Text
                 Call Command1_Click
                 Command1.Visible = False
                    End If
          If npw >= 4 Then
             npw = 0
             Text22.Text = Text2.Text
             Text21.Text = Text1.Text
             Call EnterButton_Click

                      End If
       End If
    End If
     'เช็คสถานะ
      Label11.Caption = endrun
      Label12.Caption = npw
```

```
'เช็คห้อง
    If Buffer = ">" Then
    Image3.Visible = True
    Image2.Visible = False
     Text22.Text = Text2.Text
    Text20.Text = "ที่จอดรถ"
     Call btnRun_Click
     Call stin_Click
     Call ClearData
     endrun = 0
     End If

     If Buffer = "<" Then
     Image2.Visible = True
     Image3.Visible = False
      Text20.Text = "ประตูหน้า"
     endrun = 0
     Call btnRun_Click
      Call stin_Click
     Call ClearData
     End If
' เช็ค จบรหัสบัตร และ เออเรอร์
     If (Buffer = "?") Then
        cir = 0
        'รีเซ็ทรอบพาสเวิร์ส
         Timer2.Enabled = True
                        endrun = 1
      End If
      ' รีเซ็ท
    If (Buffer = "!") Then
      Call btnRun_Click
      endrun = 0
```

```
    End If

    End Sub
Private Sub optModeRXTX_Click(Index As Integer)
  Select Case Index
  Case 0
    SecRXTX = 0
    If MSComm1.PortOpen = True Then MSComm1.RThreshold = 1
  End Select
End Sub

Public Static Sub ShowData(Term As Control, Data As String)
'  On Error GoTo Handler
  Const MAXTERMSIZE = 16000
  Dim TermSize As Long, i

  TermSize = Len(Term.Text)
  If TermSize > MAXTERMSIZE Then
    Term.Text = Mid$(Term.Text, 4097)
    TermSize = Len(Term.Text)
  End If
  Term.SelStart = TermSize
  Do
    i = InStr(Data, Chr$(10))
    If i Then
      Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
    End If
  Loop While i
  i = 1
  Do
    i = InStr(i, Data, Chr$(13))
```

```
    If i Then
      Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
      i = i + 1
      End If
  Loop While i

  Term.SelText = Data
Exit Sub
End Sub

Private Sub delay()
        Dim X As Variant
        For X = 0 To 150000
        Next X

  End Sub
Private Sub Timer2_Timer()
Timer2.Enabled = False
Call btnRun_Click
End Sub
```

โปรแกรมในส่วน Microcontroler

เครื่องอ่านบัตรเครื่องที่1

```
;*****************************
;* Program Test Magnetic Card *
;* Read Track2 Modulo5 Format *
;* HardWare   : ET CP2051 V2 *
;* Display Data Code to RS232 *
;* CPU Control : MCS51       *
;* Complier   : SXA51        *
;*****************************
;
DATA_MAG   EQU   P1.0          ; Data From Magnetic Card
PRESENT    EQU   P1.1          ; Enable Magnetic Card
CLK_MAG    EQU   P1.2          ; Clock Sync Magnetic Card

       ORG   20H
FLAG_BUF:  DS    1             ; Flag Buffer
CHECKSUM:  DS    1
BUFFER:    DS    40            ; 40 Byte Buffer
;
START      EQU   FLAG_BUF.0     ; Start Flag Status
STOP       EQU   FLAG_BUF.1     ; Stop Flag Status
ERROR      EQU   FLAG_BUF.2     ; Error Flag Status
PARITY     EQU   FLAG_BUF.3     ; Parity Error Flag Status

       ORG   0000H
 AJMP  INITIAL
INITIAL:        MOV   P0,#00000000B
            MOV   P1,#00000100B
            MOV   P2,#00000000B
            MOV   P3,#00111111B
```

```
PROGRAM:  MOV    SP,#128-32        ; Stack 32 Byte
          MOV    FLAG_BUF,#0       ; Clear All Flag
          MOV    P1,#00H
          SETB   DATA_MAG          ; Standy Signale
          SETB   PRESENT
          SETB   CLK_MAG
          ;
INIT_SR:  MOV    A,#0FDH           ; Set baud rate 9600
          MOV    TH1,A
          MOV    TL1,A
          MOV    TMOD,#00100000B   ; TIMER1 MODE 1
          CLR    ES                ; Disable serial interupt
          CLR    ET1               ; Disable timer1 interupt
          SETB   TR1               ; Set timer1 control
          MOV    SCON,#01010000B   ; Serial mode 1
          ;
TEST_MAG: LCALL  PRINT_SER
          DB     'TEST MAGNETIC CARD',0DH,0AH,00
          ;

DSP_NUM0:
     LCALL  GET_DATA          ; Get Number of Card
          MOV    R0,#BUFFER
          JB     ERROR,SUM_ERR
          JB     PARITY,PAR_ERR
          SJMP   DSP_NUM1
          ;
SUM_ERR:  LCALL  PRINT_SER        ; Checksum Error
          DB     'Checksum Error!',0DH,0AH,00
          CLR    ERROR
          SJMP   DSP_NUM0
          ;
```

```
PAR_ERR:  LCALL  PRINT_SER        ; Parity Error
          DB     'Parity Error!',0DH,0AH,00
          CLR    PARITY
          SJMP   DSP_NUM0
          ;

DSP_NUM1: MOV    A,@R0
          CJNE   A,#0FH,DSP_NUM2
          ADD    A,#30H
          LCALL  TX_BYTE          ; Display End
          SJMP   DSP_NUM3
DSP_NUM2: ADD    A,#30H
          LCALL  TX_BYTE
          INC    R0
          SJMP   DSP_NUM1
DSP_NUM3: MOV    A,#0DH           ; Carrier Return
          LCALL  TX_BYTE
          MOV    A,#0AH           ; Line Feed
          LCALL  TX_BYTE
          SJMP   DSP_NUM0

;**********************
;* Clear Data Buffer *
;* Register : R0,ACC *
;**********************
;
CLR_BUFF: PUSH   B
          MOV    B,#40
          MOV    R0,#BUFFER       ; Clear Data Buffer
CLR_BUF1: CLR    A
          MOV    @R0,A
          INC    R0
```

```
        DJNZ   B,CLR_BUF1
        POP    B
        RET

;*******************************
;* Read Data From Magnetic Card *
;* Get Number of Card (Modulo5) *
;* Output : BUFFER (BCD 40Byte) *
;* Reg.  : R1,R2,ACC,CHECKSUM  *
;*******************************
;
GET_DATA: LCALL  CLR_BUFF        ; Clear Buffer
        MOV    CHECKSUM,#0
        MOV    R1,#BUFFER-1      ; Pointer to Save Data
        CLR    START             ; Clear Any Flag
        CLR    STOP
        CLR    ERROR
        ;

  GET_DAT0:   JNB RI,A14
        . LCALL  RX_BYTE
         CJNE A,#'O',AA
        SETB  P2.7
AA:       CJNE A,#'P',A0
        SETB  P2.0
A0:         CJNE A,#'A',A1
        SETB  P2.1
A1:       CJNE A,#'B',A2
         SETB  P2.2
A2:         CJNE A,#'U',A3
        SETB  P2.3
```

```
A3:         CJNE A,#'X',A4
            SETB  P2.4
A4:         CJNE A,#'Y',A5
            SETB  P2.5
A5:         CJNE A,#'E',A6
            SETB  P2.6
A6:         CJNE A,#'Z',A7
            CLR  P2.1
A7:         CJNE A,#'G',A8
            CLR  P2.2
A8:         CJNE A,#'I',A9
            CLR  P2.3
A9:         CJNE A,#'J',A10
            CLR  P2.4
A10:        CJNE A,#'Y',A11
            CLR  P2.5
A11:        CJNE A,#'K',A12
            CLR  P2.6
A12:        CJNE A,#'L',A13
            CLR  P2.7
A13:        CJNE A,#'V',GET_DAT1
            CLR  P2.0
A14:    LCALL  GET_CLK          ; Get Start Sentinel

   MOV    C,DATA_MAG
     CPL    C
       JB     DATA_MAG,GET_DAT0  ; Loop Until Start Bit
       SETB   START              ; Strat Bit OK
       ;


GET_DAT1:     MOV    R2,#5          ; Modulo 5 Format Counter
```

```
        CLR   A
        CLR   PARITY
        INC   R1              ; Point to Next Byte Save

GET_DAT2: JB    START,GET_DAT3    ; Get 1-Byte Data
        LCALL GET_CLK
        MOV   C,DATA_MAG
        CPL   C
GET_DAT3: RRC   A
        CLR   START
        DJNZ  R2,GET_DAT2
        RR    A
        RR    A
        RR    A
        JB    P,GET_DAT4        ; Parity Odd OK
        SETB  PARITY            ; Parity Error
        SJMP  GET_DAT8
        ;
GET_DAT4: ANL   A,#0FH          ; Ignore Parity
        JB    STOP,GET_DAT5
        PUSH  ACC
        XRL   A,CHECKSUM        ; Checksum Data
        MOV   CHECKSUM,A
        POP   ACC
GET_DAT5: MOV   @R1,A           ; Save Data
        JB    STOP,GET_DAT6     ; Stop Operation
        CJNE  A,#0FH,GET_DAT1
        SETB  STOP
        SJMP  GET_DAT1
GET_DAT6: CJNE  A,CHECKSUM,GET_DAT7
        CLR   ERROR             ; Checksum OK
        SJMP  GET_DAT8
```

```
GET_DAT7:  SETB   ERROR          ; Checksum Error

GET_DAT8:  RET

;**********************
;* Get Magnetic Clock *
;**********************
;
GET_CLK:   JB     CLK_MAG,$      ; Wait Falling Clock
           NOP
           NOP
           JNB    CLK_MAG,$      ; Wait Rising Clock
           NOP
           NOP
           RET

;************************
;* Send 1-Byte to RS-232 *
;* Input  : ACC       *
;* Output : Serial port *
;************************
;
TX_BYTE:   CLR    TI
           MOV    SBUF,A
           JNB    TI,$
           CALL DELAY
            CLR    TI
           RET
RX_BYTE:       JNB  RI,$          ;WAIT DATA FROM COMPUTER
               CLR  RI
               MOV  A,SBUF
               RET
```

```
DELAY:      MOV  R7,#00H
    L2:      MOV  R6,#0FFFH
    L1:      DJNZ R6,L1
             DJNZ R7,L2
             RET
;************************************
;* Print Data to Serial Port RS232 *
;* Usage   : LCALL PRINT_SER     *
;*         : DB  'xxxx',00       *
;* Register : ACC                *
;* Note    : last byte must be 00 *
;************************************
;
PRINT_SER: POP   DPH
       POP   DPL
PRINT1:   CLR   A
       MOVC  A,@A+DPTR
       CJNE  A,#00H,PRINT2
       SJMP  PRINT3
PRINT2:   LCALL  TX_BYTE
       INC   DPTR
       SJMP  PRINT1
PRINT3:   PUSH  DPL
       PUSH  DPH
       RET

       END
```

โปรแกรมในส่วน Microcontroler

เครื่องอ่านบัตรเครื่องที่2,3

```
;*****************************
;* Program Test Magnetic Card *
;* Read Track2 Modulo5 Format *
;* HardWare   : ET CP2051 V2 *
;* Display Data Code to RS232 *
;* CPU Control : MCS51       *
;* Complier   : SXA51        *
;*****************************
;
DATA_MAG   EQU   P1.0          ; Data From Magnetic Card
PRESENT    EQU   P1.1          ; Enable Magnetic Card
CLK_MAG    EQU   P1.2          ; Clock Sync Magnetic Card
;_____________________________________
;Define Port&Pin Name
;_____________________________________
LCD_EN       BIT   P3.6
LCD_RS       BIT   P3.7
KPAD_ROW0    BIT   P2.0
KPAD_ROW1    BIT   P2.1
KPAD_ROW2    BIT   P2.2
KPAD_ROW3    BIT   P2.3
KPAD_COL3    BIT   P2.4
KPAD_COL2    BIT   P2.5
KPAD_COL1    BIT   P2.6
KPAD_COL0    BIT   P2.7
;_____________________________________
;Define User Register
;_____________________________________
LCD_ADDR     EQU   030H
```

```
LCD_DATA      EQU    031H
KPAD_DATA     EQU    032H
PAS1_DATA     EQU    033H
PAS2_DATA     EQU    034H
PAS3_DATA     EQU    035H
PAS4_DATA     EQU    036H
WMCON         EQU    096H

        ORG    20H
FLAG_BUF:  DS     1              ; Flag Buffer
CHECKSUM:  DS     1
BUFFER:    DS     40             ; 40 Byte Buffer
;
START      EQU    FLAG_BUF.0     ; Start Flag Status
STOP       EQU    FLAG_BUF.1     ; Stop Flag Status
ERROR      EQU    FLAG_BUF.2     ; Error Flag Status
PARITY     EQU    FLAG_BUF.3     ; Parity Error Flag Status

        ORG    0000H
           AJMP   INITIAL
INITIAL:          MOV    P0,#00000000B
                  MOV    P1,#00000100B
                  MOV    P2,#11111111B
                  MOV    P3,#00111111B
PROGRAM:   MOV    SP,#128-32     ; Stack 32 Byte
        MOV    FLAG_BUF,#0    ; Clear All Flag
        MOV    P1,#00H
        SETB   DATA_MAG       ; Standy Signale
        SETB   PRESENT
        SETB   CLK_MAG
         ;
INIT_SR:   MOV    A,#0FDH        ; Set baud rate 9600
```

```
        MOV    TH1,A
        MOV    TL1,A
        MOV    TMOD,#00100000B    ; TIMER1 MODE 1
        CLR    ES                 ; Disable serial interupt
        CLR    ET1                ; Disable timer1 interupt
        SETB   TR1                ; Set timer1 control
        MOV    SCON,#01010000B    ; Serial mode 1
        ;
TEST_MAG:  LCALL   PRINT_SER
        DB     'TEST MAGNETIC CARD',0DH,0AH,00
        ;

DSP_NUM0: MOV    A,#0cH
            ADD    A,#30H
            LCALL  TX_BYTE
           CLR P1.5
           LCALL   LOOP
          LCALL   GET_DATA          ; Get Number of Card
        MOV    R0,#BUFFER
        JB     ERROR,SUM_ERR
        JB     PARITY,PAR_ERR
        SJMP   DSP_NUM1
        ;
SUM_ERR:   LCALL   PRINT_SER         ; Checksum Error
        DB     'Checksum Error!',0DH,0AH,00
        CLR    ERROR
        SJMP   DSP_NUM0
        ;
PAR_ERR:   LCALL   PRINT_SER         ; Parity Error
        DB     'Parity Error!',0DH,0AH,00
        CLR    PARITY
        SJMP   DSP_NUM0
```

```
        ;
DSP_NUM1:  MOV    A,@R0
        CJNE   A,#0FH,DSP_NUM2
        ADD    A,#30H
        LCALL  TX_BYTE          ; Display End
        SJMP   DSP_NUM3
DSP_NUM2:  ADD    A,#30H
        LCALL  TX_BYTE
        INC    R0
        SJMP   DSP_NUM1
DSP_NUM3:  MOV    A,#0DH          ; Carrier Return
        LCALL  TX_BYTE
        MOV    A,#0AH          ; Line Feed
        LCALL  TX_BYTE
         SETB P1.5

         LCALL  PASSWORD
        LCALL RX_BYTE
        CJNE   A,#'1',Q1
        SETB   P1.4
         CALL DELAY500mS
       CLR  P1.4
   CALL DELAY500mS
  SETB  P1.4
         CALL DELAY500mS
       CLR  P1.4
   CALL DELAY500mS
  SETB  P1.4
         CALL DELAY500mS
       CLR  P1.4
   CALL DELAY500mS
  SETB  P1.4
```

```
        CALL DELAY500mS
       CLR  P1.4
 CALL DELAY500mS
SETB  P1.4
        CALL DELAY500mS
       CLR  P1.4
 CALL DELAY500mS
         SETB  P1.4
        CALL   DELAY_10s
         CLR   P1.4
          SJMP   Q2
Q1:     LCALL ERROR1
SETB  P1.7
         CALL DELAY500mS
        CLR  P1.7
 CALL DELAY500mS
SETB  P1.7
         CALL DELAY500mS
        CLR  P1.7
 CALL DELAY500mS
SETB  P1.7
         CALL DELAY500mS
        CLR  P1.7
 CALL DELAY500mS
SETB  P1.7
         CALL DELAY500mS
        CLR  P1.7
 CALL DELAY500mS
SETB  P1.7
         CALL DELAY500mS
        CLR  P1.7
 DSP_NUM4:  MOV    A,#0DH          ; Carrier Return
```

```
        LCALL  TX_BYTE
        MOV    A,#0AH          ; Line Feed
        LCALL  TX_BYTE
        LCALL  PASSWORD
        LCALL RX_BYTE
         CJNE   A,#'1',Q2
        SETB  P1.4
         CALL  DELAY500mS
       CLR  P1.4
 CALL  DELAY500mS
SETB  P1.4
         CALL  DELAY500mS
       CLR  P1.4
 CALL  DELAY500mS
SETB  P1.4
         CALL  DELAY500mS
       CLR  P1.4
 CALL  DELAY500mS
SETB  P1.4
         CALL  DELAY500mS
       CLR  P1.4
 CALL  DELAY500mS
SETB  P1.4
         CALL  DELAY500mS
       CLR  P1.4
 CALL  DELAY500mS
          SETB  P1.4
        CALL   DELAY_10s
         CLR    P1.4
Q2:     JMP   DSP_NUM0

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

```
;PRESS PASSWORD
;**********************
PASSWORD:      CALL   INIT_LCD

KP26:            MOV    LCD_DATA,#'*'
                LCALL  WRCHAR_LCD
                CALL   LCD_BLINK
                SJMP   LOOP7


LOOP7:          MOV    LCD_ADDR,#00H
                CALL   SET_ADDR_LCD
                MOV    DPTR,#TITLE_3
                CALL   WRLINE_LCD
                MOV    LCD_ADDR,#040H
                CALL   SET_ADDR_LCD
               CALL   LCD_BLINK
                CALL   DELAY500mS
                CALL   KP40
                MOV    A,KPAD_DATA
                ADD    A,#030H
                CALL TX_BYTE
                  MOV    LCD_DATA,A
                CALL   WRCHAR_LCD
                CALL   LCD_BLINK
                MOV    DPTR,#0000H
                MOV    A,KPAD_DATA
                MOV    PAS1_DATA,A
                CALL   DELAY500mS
                CALL   KP40
                MOV    A,KPAD_DATA
               ADD    A,#030H
```

```
  CALL TX_BYTE
 MOV    LCD_DATA,A
 CALL   WRCHAR_LCD
 CALL   LCD_BLINK
 MOV    DPTR,#0001H
 MOV    A,KPAD_DATA
 MOV    PAS2_DATA,A
 CALL   DELAY500mS

 CALL   KP40
 MOV    A,KPAD_DATA
   ADD    A,#030H
 CALL  TX_BYTE
    MOV    LCD_DATA,A
 CALL   WRCHAR_LCD
 CALL   LCD_BLINK
 MOV    DPTR,#0002H
 MOV    A,KPAD_DATA
 MOV    PAS3_DATA,A
  CALL   DELAY500mS

  CALL   KP40
  MOV    A,KPAD_DATA
ADD    A,#030H
 CALL  TX_BYTE
  MOV    LCD_DATA,A
  CALL   WRCHAR_LCD
  CALL   LCD_BLINK
  MOV    DPTR,#0003H
  MOV    A,KPAD_DATA
  MOV    PAS4_DATA,A
   CALL   DELAY500mS
```

```
            RET
KP40:        MOV  R2,#64H
KP401:      MOV  R1,#64H
KP402:     LCALL  GET_KPAD
            MOV    A,KPAD_DATA
            CJNE   A,#00,KP41
            CALL DELAY2_50mS
           DJNZ  R1, KP402
           DJNZ  R2, KP401

           JMP  DSP_NUM0

KP41:           RET
;*********************
;SHOW 'KMITL'
;*******************
LOOP:        CALL   INIT_LCD
          MOV    LCD_ADDR,#00H
            CALL   SET_ADDR_LCD
            MOV    DPTR,#TITLE_1
            CALL   WRLINE_LCD

            MOV    LCD_ADDR,#040H
            CALL   SET_ADDR_LCD
            MOV    DPTR,#TITLE_2
            CALL   WRLINE_LCD
            CALL   DELAY_1s
           RET
;*********************
;SHOW 'ERROR'
;*********************
ERROR1:     CALL   INIT_LCD
```

```
        MOV    LCD_ADDR,#00H
          CALL   SET_ADDR_LCD
          MOV    DPTR,#TITLE_6
          CALL   WRLINE_LCD

          RET

;*********************
;* Clear Data Buffer *
;* Register : R0,ACC *
;*********************
;
CLR_BUFF:  PUSH   B
       MOV    B,#40
       MOV    R0,#BUFFER       ; Clear Data Buffer
CLR_BUF1:  CLR    A
       MOV    @R0,A
       INC    R0
       DJNZ   B,CLR_BUF1
       POP    B
       RET

;*********************************
;* Read Data From Magnetic Card *
;* Get Number of Card (Modulo5) *
;* Output : BUFFER (BCD 40Byte) *
;* Reg.  : R1,R2,ACC,CHECKSUM  *
;*********************************
;
GET_DATA:  LCALL  CLR_BUFF         ; Clear Buffer
       MOV    CHECKSUM,#0
       MOV    R1,#BUFFER-1      ; Pointer to Save Data
```

```
        CLR   START          ; Clear Any Flag
        CLR   STOP
        CLR   ERROR
        ;
GET_DAT0: LCALL  GET_CLK     ; Get Start Sentinel
        MOV   C,DATA_MAG
        CPL   C
        JB    DATA_MAG,GET_DAT0  ; Loop Until Start Bit
         SETB P1.6
        JNB  P1.6,GET_DATA
         SETB  START         ; Strat Bit OK
        ;
GET_DAT1: MOV   R2,#5        ; Modulo 5 Format Counter
        CLR   A
        CLR   PARITY
        INC   R1             ; Point to Next Byte Save
GET_DAT2: JB    START,GET_DAT3    ; Get 1-Byte Data
        LCALL  GET_CLK
        MOV   C,DATA_MAG
        CPL   C
GET_DAT3: RRC   A
        CLR   START
        DJNZ  R2,GET_DAT2
        RR    A
        RR    A
        RR    A
        JB    P,GET_DAT4     ; Parity Odd OK
        SETB  PARITY         ; Parity Error
        SJMP  GET_DAT8
        ;
GET_DAT4: ANL   A,#0FH       ; Ignore Parity
        JB    STOP,GET_DAT5
```

```
        PUSH   ACC
        XRL    A,CHECKSUM        ; Checksum Data
        MOV    CHECKSUM,A
        POP    ACC
GET_DAT5: MOV   @R1,A            ; Save Data
        JB     STOP,GET_DAT6     ; Stop Operation
        CJNE   A,#0FH,GET_DAT1
        SETB   STOP
        SJMP   GET_DAT1
GET_DAT6: CJNE  A,CHECKSUM,GET_DAT7
        CLR    ERROR             ; Checksum OK
        SJMP   GET_DAT8
GET_DAT7: SETB  ERROR            ; Checksum Error
GET_DAT8: RET

;**********************
;* Get Magnetic Clock *
;**********************
;
GET_CLK:  JB    CLK_MAG,$        ; Wait Falling Clock
        NOP
        NOP
        JNB    CLK_MAG,$        ; Wait Rising Clock
        NOP
        NOP
        RET

;*************************
;* Send&Receiver 1-Byte to RS-232 *
;* Input  : ACC       *
;* Output  : Serial port *
;*************************
```

```
;
TX_BYTE:  CLR   TI
          MOV   SBUF,A
          JNB   TI,$
          CALL DELAY
           CLR   TI
          RET
RX_BYTE:    JNB  RI,$          ;WAIT DATA FROM COMPUTER
            CLR  RI
            MOV  A,SBUF
            RET
DELAY:      MOV  R7,#00H
    L2:      MOV  R6,#0FFFH
    L1:      DJNZ R6,L1
            DJNZ R7,L2
            RET
;************************************
;* Print Data to Serial Port RS232 *
;* Usage   : LCALL PRINT_SER     *
;*         : DB  'xxxx',00       *
;* Register : ACC                *
;* Note    : last byte must be 00 *
;************************************
;
PRINT_SER: POP   DPH
          POP   DPL
PRINT1:   CLR   A
          MOVC  A,@A+DPTR
          CJNE  A,#00H,PRINT2
          SJMP  PRINT3
PRINT2:   LCALL  TX_BYTE
          INC   DPTR
```

```
        SJMP   PRINT1
PRINT3:  PUSH   DPL
        PUSH   DPH
        RET
;*********************************
;Keypad Scan key Subroutine
;*********************************
GET_KPAD:          MOV    P2,#0FFH
                MOV    KPAD_DATA,#0

CHK_COL0:          CLR    KPAD_COL0
                MOV    A,P2
                ANL    A,#00FH
                CJNE   A,#00FH,COL0_DETECT
                AJMP   CHK_COL1

COL0_DETECT:       MOV    KPAD_DATA,#1
                AJMP   GET_ROW

CHK_COL1:          SETB   KPAD_COL0
                CLR    KPAD_COL1
                MOV    A,P2
                ANL    A,#00FH
                CJNE   A,#00FH,COL1_DETECT
                AJMP   CHK_COL2

COL1_DETECT:       MOV    KPAD_DATA,#2
                AJMP   GET_ROW

CHK_COL2:          SETB   KPAD_COL1
                CLR    KPAD_COL2
                MOV    A,P2
```

```
        ANL   A,#00FH
        CJNE  A,#00FH,COL2_DETECT
        AJMP  CHK_COL3

COL2_DETECT:    MOV   KPAD_DATA,#3
        AJMP  GET_ROW

CHK_COL3:       SETB  KPAD_COL2
        CLR   KPAD_COL3
        MOV   A,P2
        ANL   A,#00FH
        CJNE  A,#00FH,COL3_DETECT
        RET

COL3_DETECT:    MOV   KPAD_DATA,#13

GET_ROW:        CLR   KPAD_COL0
        CLR   KPAD_COL1
        CLR   KPAD_COL2
        CLR   KPAD_COL3
        JB    KPAD_ROW0,CHK_ROW1
        RET

CHK_ROW1:       JB    KPAD_ROW1,CHK_ROW2
        MOV   A,KPAD_DATA
        ADD   A,#3
        MOV   KPAD_DATA,A
        RET

CHK_ROW2:       JB    KPAD_ROW2,CHK_ROW3
        MOV   A,KPAD_DATA
        ADD   A,#6
```

```
                MOV    KPAD_DATA,A
                RET

CHK_ROW3:       MOV    A,KPAD_DATA
                ADD    A,#9
                MOV    KPAD_DATA,A
                RET

;_____________________________________
;LCD Initialre
;_____________________________________
INIT_LCD:       ACALL  DELAY_100ms
                CLR    LCD_RS
                MOV    P0,#00111000B
                ACALL  LCD_CLK
                ACALL  DELAY_10ms
                ACALL  LCD_OFF
                ACALL  LCD_CLR
                MOV    P0,#00000110B
                ACALL  LCD_CLK
                ACALL  LCD_HOME

;_____________________________________
;LCD Clear Dsplay
;_____________________________________
LCD_CLR:        CLR    LCD_RS
                MOV    P0,#00000001B
                ACALL  LCD_CLK
                RET

;_____________________________________
;LCD Return Home
```

```
;_____________________________________
LCD_HOME:       CLR   LCD_RS
                MOV   P0,#00000001B
                ACALL LCD_CLK
                RET

;_____________________________________
;LCD Display Off
;_____________________________________
LCD_OFF:        CLR   LCD_RS
                MOV   P0,#00001000B
                ACALL LCD_CLK
                RET

;_____________________________________
;LCD Clk
;_____________________________________
LCD_CLK:        SETB  LCD_EN
                ACALL LCD_DELAY
                CLR   LCD_EN
                ACALL LCD_DELAY
                RET

;_____________________________________
;LCD Display On
;_____________________________________
LCD_ON:         CLR   LCD_RS
                MOV   P0,#00001100B
                ACALL LCD_CLK
                RET

;_____________________________________
```

```
;LCD Cursor On
;_____________________________________
LCD_BLINK:          CLR    LCD_RS
                    MOV    P0,#00001111B
                    ACALL  LCD_CLK
                    RET

;_____________________________________
;LCD Left Shift Display
;_____________________________________
LCD_LSHF:           CLR    LCD_RS
                    MOV    P0,#00011000B
                    ACALL  LCD_CLK
                    RET

;_____________________________________
;LCD Right Shift Display
;_____________________________________
LCD_RSHF:           CLR    LCD_RS
                    MOV    P0,#00011100B
                    ACALL  LCD_CLK
                    RET

;_____________________________________
;Set LCD Address
;I/P:               LCD_ADDR
;_____________________________________
SET_ADDR_LCD:       CLR    LCD_RS
                    MOV    A,LCD_ADDR
                    SETB   ACC.7
                    MOV    P0,A
                    ACALL  LCD_CLK
```

```
                RET

;_____________________________________________
;Write Character to show LCD
;I/P:            LCD_DATA
;_____________________________________________
WRCHAR_LCD:         SETB   LCD_RS
                    MOV    P0,LCD_DATA
                    ACALL  LCD_CLK
                    ACALL  LCD_ON
                    RET

;_____________________________________________
;Write Line of 20 Character from ROM
;I/P:            DPTR : Locate ROM Address
;_____________________________________________
WRLINE_LCD:         MOV    R0,#0
WRLINE_LCD_1:       SETB   LCD_RS
                    CLR    A
                    MOVC   A,@A+DPTR
                    MOV    P0,A
                    ACALL  LCD_CLK
                    INC    DPTR
                    INC    R0
                    CJNE   R0,#16,WRLINE_LCD_1
                    ACALL  LCD_ON
                    RET

;_____________________________________________
;Dummy Delay time LCD_DELAY, 10m, 100m, 1s
;_____________________________________________
LCD_DELAY:          MOV    7,#002
```

```
LCD_DELAY_1:      MOV   6,#0E6H
LCD_DELAY_2:      NOP
                  NOP
                  DJNZ  R6,LCD_DELAY_2
                  DJNZ  R7,LCD_DELAY_1
                  RET

DELAY_10ms:       MOV   7,#010
DELAY_10ms_1:     MOV   6,#0E6H
DELAY_10ms_2:     NOP
                  NOP
                  DJNZ  R6,DELAY_10ms_2
                  DJNZ  R7,DELAY_10ms_1
                  RET

DELAY_100ms:      MOV   7,#100
DELAY_100ms_1:    MOV   6,#0E6H
DELAY_100ms_2:    NOP
                  NOP
                  DJNZ  R6,DELAY_100ms_2
                  DJNZ  R7,DELAY_100ms_1
                  RET

DELAY_500ms:      MOV   7,#50
DELAY_500ms_1:    MOV   6,#0E6H
DELAY_500ms_2:    NOP
                  NOP
                  DJNZ  R6,DELAY_500ms_2
                  DJNZ  R7,DELAY_500ms_1
                  RET

DELAY_1s:         MOV   5,#100
```

```
DELAY_1s_1:     ACALL  DELAY_10ms
                DJNZ   R5,DELAY_1s_1
                RET

DELAY_10s:      MOV    4,#10
DELAY_10s_1:    ACALL  DELAY_1s
                DJNZ   R4,DELAY_10s_1
                RET

;**********************
;กระพริบ
;**********************
DELAY500mS: MOV R5,#0C8H
        LO1:    CALL DELAY2_50mS
                DJNZ R5,LO1
                RET
DELAY2_50mS:    MOV  R7,#0AH
     LMO2:      MOV  R6,#73H
     LMO1:      DJNZ R6,LMO1
                DJNZ R7,LMO2
                RET
              ;_____________________________________
;Define Constant < Store in Flash EEPROM Program Memory >
;_____________________________________
;               01234567890123456
TITLE_1:        DB   '   KMITL    '
TITLE_2:        DB   '  PROJECT   '
TITLE_3:        DB   '  PASSWORD  '
TITLE_4:        DB   '    OK.     '
TITLE_5:        DB   '            '
TITLE_6:        DB   '   ERROR    '
TITLE_7:        DB   ' NEW PASSWORD '
```

```
TITLE_8:        DB    ' OLD PASSWORD '

        END
```

ภาคผนวก ข

Features

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4.0V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low Power Idle and Power Down Modes
- Interrupt Recovery From Power Down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power Off Flag

Description

The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard 80C51 instruction set and pinout. The on-chip Downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89S8252 provides the following standard features: 8K bytes of Downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two Data Pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S8252 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The Downloadable Flash can be changed a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated.

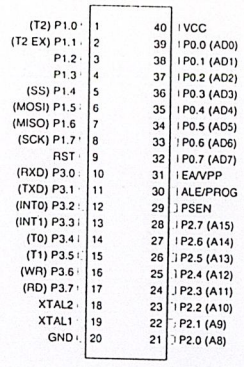
8-Bit Microcontroller with 8K Bytes Flash

AT89S8252

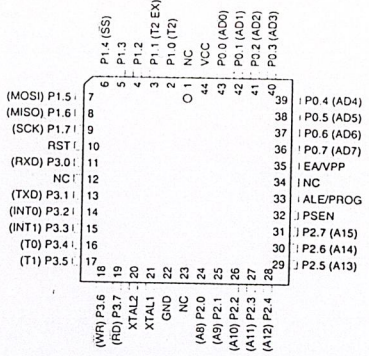
0401D-A-12/97

Pin Configurations

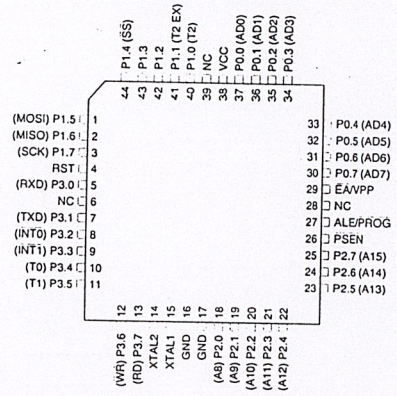
PDIP



PLCC



PQFP/TQFP



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

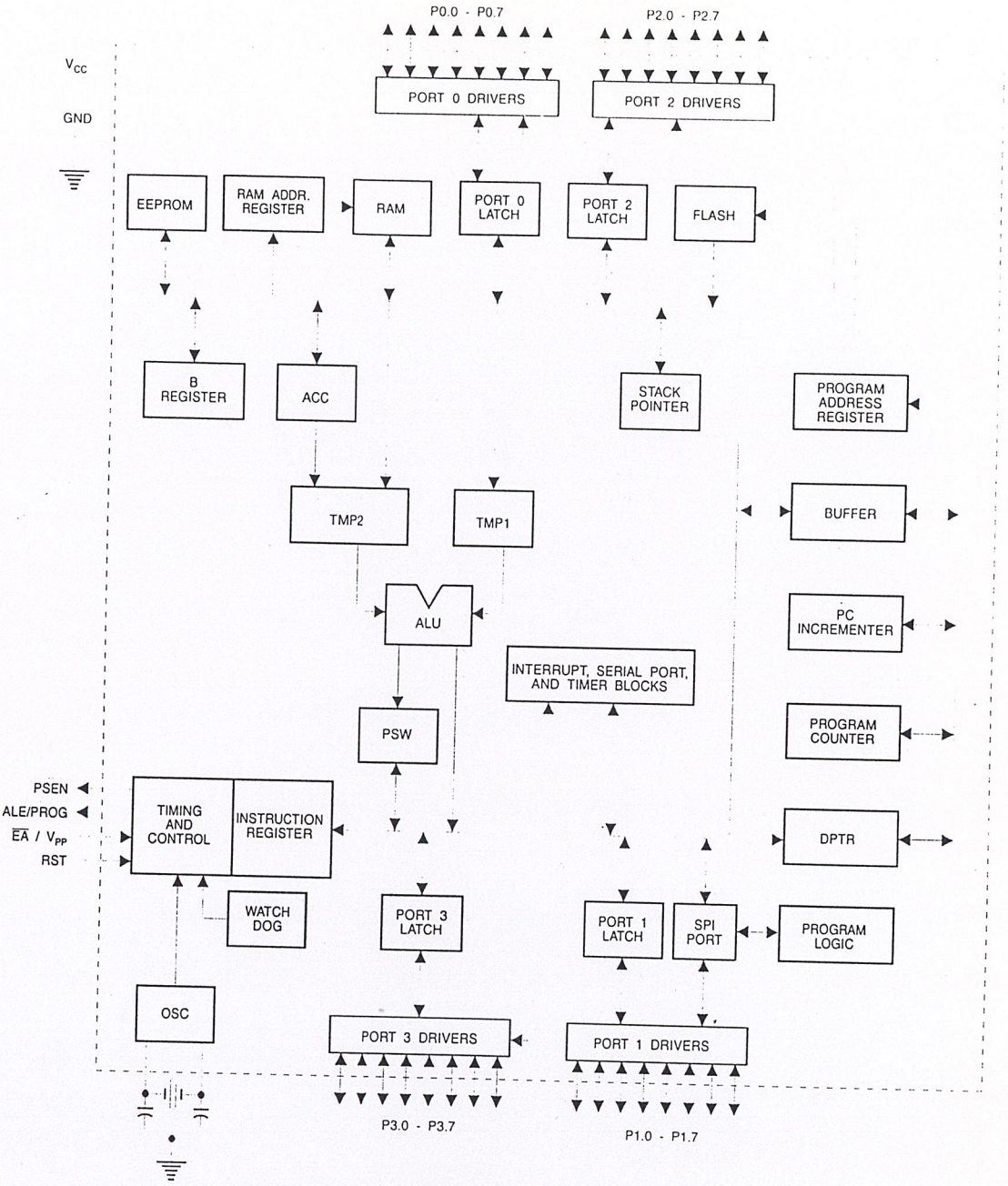
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Some Port 1 pins provide additional functions. P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Block Diagram



Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8 bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89S8252, as shown in the following table.

Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89S8252 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/V_{PP}

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions. This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming when 12-volt programming is selected.

XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

Table 1. AT89S8252 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001XX			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 9) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16 bit capture mode or 16-bit auto-reload mode.

Watchdog and Memory Control Register The WMCON register contains control bits for the Watchdog Timer (shown in Table 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The DPS bit selects one of two DPTR registers available.

SPI Registers Control and status bits for the Serial Peripheral Interface are contained in registers SPCR (shown in Table 4) and SPSR (shown in Table 5). The SPI data bits are contained in the SPDR register. Writing the SPI data register during serial data transfer sets the Write Collision bit, WCOL, in the SPSR register. The SPDR is double buffered for writing and the values in SPDR are not changed by Reset.

Interrupt Registers The global interrupt enable bit and the individual interrupt enable bits are in the IE register. In addition, the individual interrupt enable bit for the SPI is in the SPCR register. Two priorities can be set for each of the six interrupt sources in the IP register.

Table 2. T2CON—Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflows to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Dual Data Pointer Registers To facilitate accessing both internal EEPROM and external data memory, two banks of 16 bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR WMCON selects DP0 and DPS = 1 selects DP1. The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag The Power Off Flag (POF) is located at bit_4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and reset under software control and is not affected by RESET.

Table 3. WMCON—Watchdog and Memory Control Register

WMCON Address = 96H — Reset Value = 0000 0010B

	PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN
Bit	7	6	5	4	3	2	1	0

Symbol	Function
PS2 PS1 PS0	Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms.
EEMWE	EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed.
EEMEN	Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory.
DPS	Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1
WDTRST RDY/$\overline{BSY}$	Watchdog Timer Reset and EEPROM Ready/$\overline{Busy}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{BSY}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{BSY}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{BSY}$ bit equals "0" and is automatically reset to "1" when programming is completed.
WDTEN	Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer.

Table 4. SPCR—SPI Control Register

SPCR Address = D5H — Reset Value = 0000 01XXB

Bit	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
	7	6	5	4	3	2	1	0

Symbol	Function
SPIE	SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts.
SPE	SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel.
DORD	Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission.
MSTR	Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode.
CPOL	Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control.
CPHA	Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control.
SPR0 SPR1	SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 1 1 128

Table 5. SPSR—SPI Status Register

SPSR Address = AAH — Reset Value = 00XX XXXXB

Bit	SPIF	WCOL	—	—	—	—	—	—
	7	6	5	4	3	2	1	0

Symbol	Function
SPIF	SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then accessing the SPI data register.
WCOL	Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register.

Figure 14. Programming the Flash/EEPROM Memory

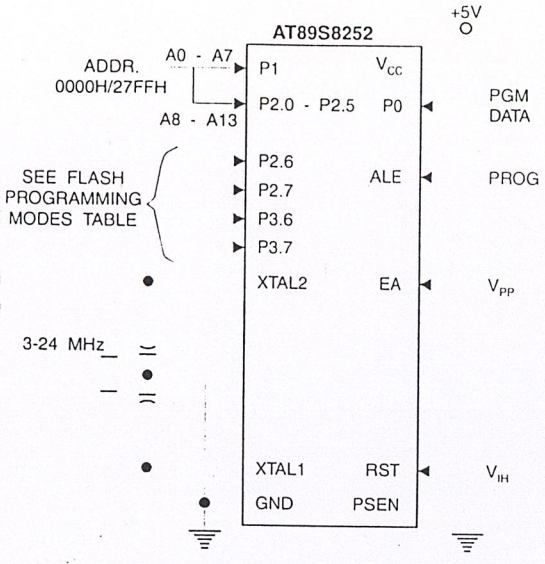


Figure 15. Flash/EEPROM Serial Downloading

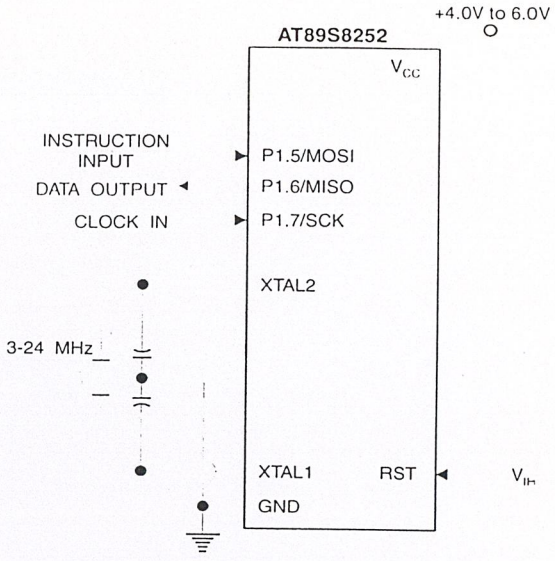
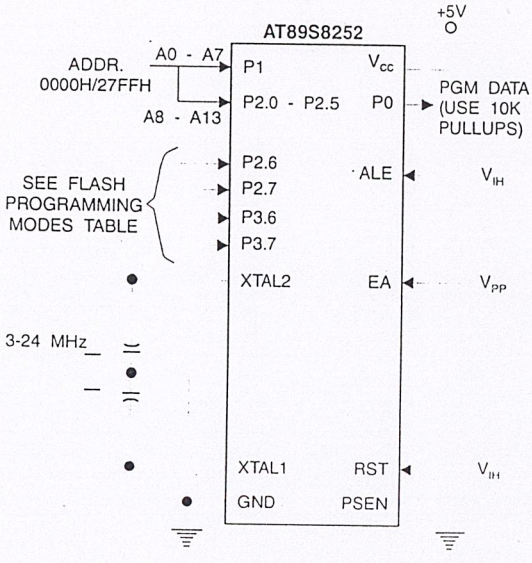


Figure 16. Verifying the Flash/EEPROM Memory

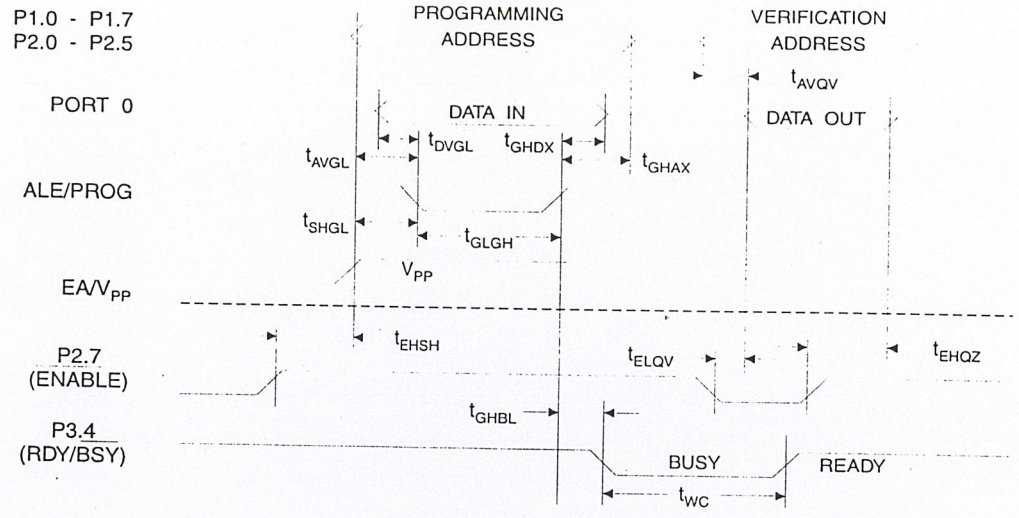


Flash Programming and Verification Characteristics-Parallel Mode

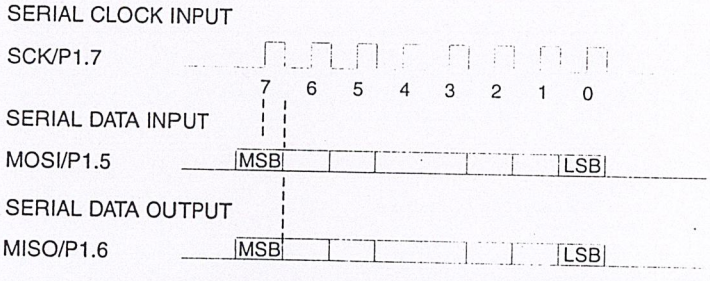
T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

Flash/EEPROM Programming and Verification Waveforms - Parallel Mode



Serial Downloading Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

The values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ± 20%, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage (1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.5	V
V_{OL1}	Output Low Voltage (1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.5	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power Down Mode (2)	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10 mA
Maximum I_{OL} per 8-bit port:
Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total I_{OL} for all output pins: 71 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V

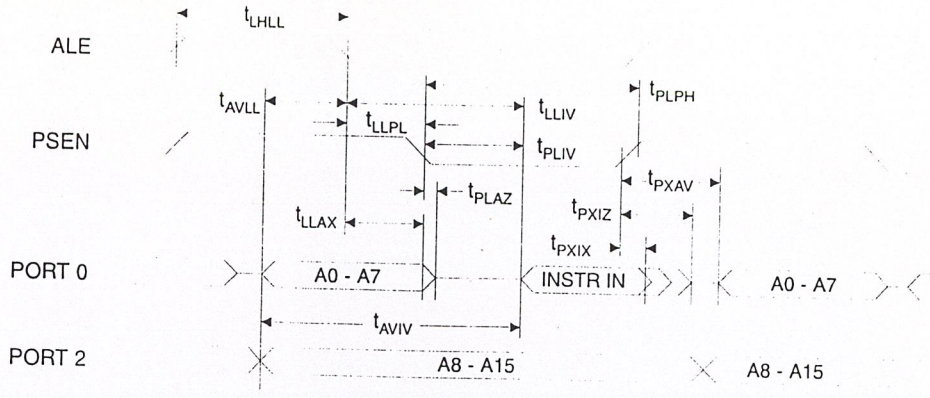
AC Characteristics

Under operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other outputs = 80 pF.

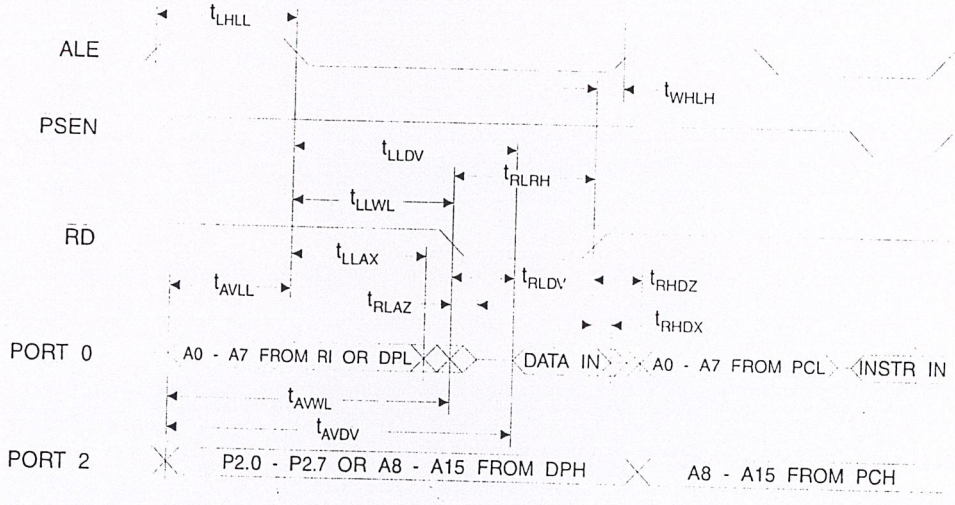
External Program and Data Memory Characteristics

Symbol	Parameter	Variable Oscillator		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{LHLL}	ALE Pulse Width	$2t_{CLCL}$ - 40		ns
t_{AVLL}	Address Valid to ALE Low	t_{CLCL} - 13		ns
t_{LLAX}	Address Hold After ALE Low	t_{CLCL} - 20		ns
t_{LLIV}	ALE Low to Valid Instruction In		$4t_{CLCL}$ - 65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	t_{CLCL} - 13		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	$3t_{CLCL}$ - 20		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		$3t_{CLCL}$ - 45	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		t_{CLCL} - 10	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	t_{CLCL} - 8		ns
t_{AVIV}	Address to Valid Instruction In		$5t_{CLCL}$ - 55	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	$6t_{CLCL}$ - 100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	$6t_{CLCL}$ - 100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		$5t_{CLCL}$ - 90	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		$2t_{CLCL}$ - 28	ns
t_{LLDV}	ALE Low to Valid Data In		$8t_{CLCL}$ - 150	ns
t_{AVDV}	Address to Valid Data In		$9t_{CLCL}$ - 165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$3t_{CLCL}$ - 50	$3t_{CLCL}$ + 50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	$4t_{CLCL}$ - 75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	t_{CLCL} - 20		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	$7t_{CLCL}$ - 120		ns
t_{WHQX}	Data Hold After $\overline{WR}$	t_{CLCL} - 20		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{CLCL} - 20	t_{CLCL} + 25	ns

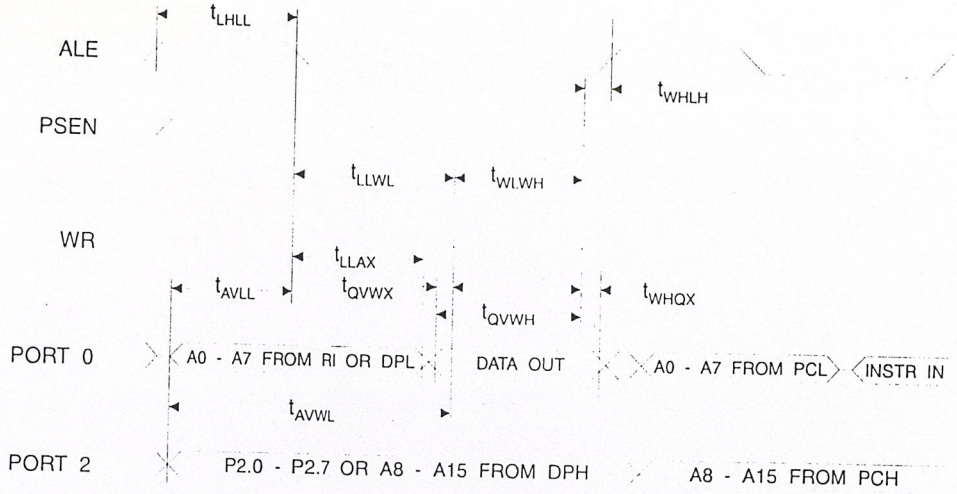
External Program Memory Read Cycle



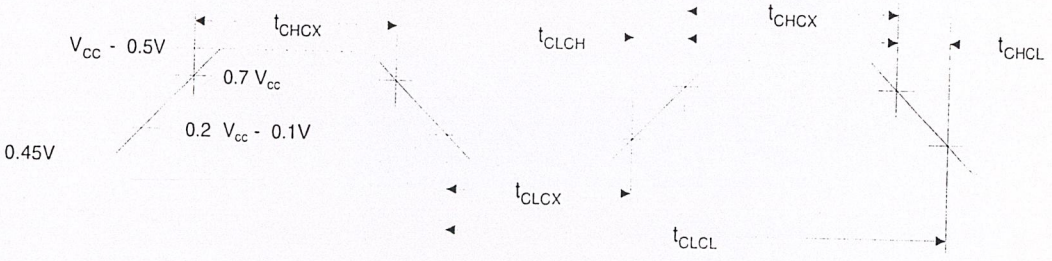
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

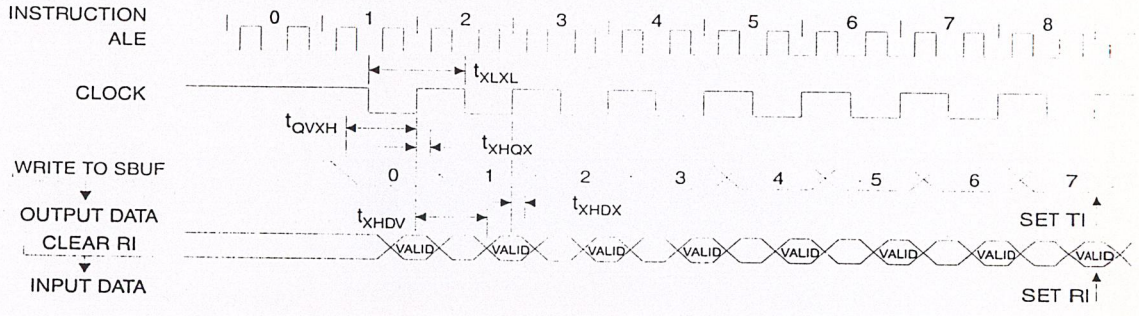
Symbol	Parameter	V_{CC} = 4.0V to 6.0V		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

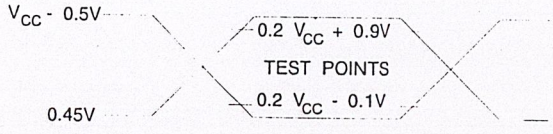
The values in this table are valid for V_{CC} = 4.0V to 6V and Load Capacitance = 80 pF.

Symbol	Parameter	Variable Oscillator		Units
		Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	$10t_{CLCL}$ - 133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	$2t_{CLCL}$ - 117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		$10t_{CLCL}$ - 133	ns

Shift Register Mode Timing Waveforms

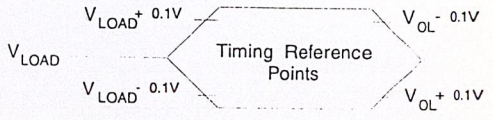


AC Testing Input/Output Waveforms(1)



Notes: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms(1)



Notes: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.