ระบบรักษาความปลอดภัยระยะไกล

Computer Datalink Control System

โดย

นายปิลันธน์ อุ่นตรงจิตร 36013161

วัน เดือน ปี 15 ส.ค. 2540
เลขทะเบียน 037255
เลขเรียกหนังสือ T 38348 ป 6217

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

037255

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบรักษาความปลอดภัยระยะไกล

(Computer Data Link Control System)

ผู้จัดทำ

นายปิลันธน์ อุ่นตรงจิตร 36013161

.....อาจารย์ที่ปรึกษา

(อ. สมศักดิ์ มิตะถา)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบรักษาความปลอดภัยระยะไกล

(Computer Data Link Control System)

นายปิลันธน์ อุ่นตรงจิตร 36013161
อาจารย์ สมศักดิ์ มิตะถา
ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการออกแบบและพัฒนาสร้างระบบเก็บข้อมูลและควบคุมระยะไกลตามมาตราฐาน RS-485 การติดต่อสื่อสารข้อมูลภายในระบบเครือข่ายอ้างอิงตามโปรโตคอลแบบ ASN ซึ่งทำการพัฒนาขึ้นโดยเฉพาะ ภายในระบบเครือข่ายประกอบด้วยเครื่องไมโครคอมพิวเตอร์ 1 ตัวทำหน้าที่เป็นตัวรับ ส่งข้อมูลจากโหนดที่มีทั้งหมด 3 โหนด โดยเราจะเรียกว่า สถานีควบคุม ซึ่งจะทำหน้าที่ควบคุมการสื่อสารทั้งหมดของระบบและติดต่อกับผู้ใช้ ส่วนสถานีย่อยทำหน้าที่ตอบสนองคำสั่งจากสถานีควบคุม อ่านค่าสถานะต่างๆ จากโมดูลซึ่งเป็นสวิตช์และส่งไปยังสถานีควบคุมเพื่อทำการแสดงผล โดยโปรแกรมที่ใช้ติดต่อกับผู้ใช้นั้นได้พัฒนาขึ้นจากภาษาซี ซึ่งมีการเข้าถึงระบบได้อย่างรวดเร็วและเป็นที่นิยมใช้กันอย่างกว้างขวางในหมู่นักเขียนโปรแกรมทำให้สะดวกและง่ายในการที่จะนำไปพัฒนาต่อให้เกิดความสมบูรณ์ยิ่งขึ้นในอนาคต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Computer Data Link Control System

Mr. Pilun Ountrongjit 36013161

Advisor Mr. Somsak Mitatha

Academic year 1995

Abstract

Thai paper describes the design, developement and construction of remote data acquisition and control system via RS-485. The Communication of System use ASN potocal for data communicatiion by especially develop. The system to consist of one microcomputer to act as receive and send data from three node. By function of Control Station is all control communication and user interface. The function of Sub Station is respond instruction from Control Station , read status from switch and send to Control Station for display. The program user interface develop by C Language because very quick to approach data and to become popular in programmer. Easy for development in the future.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

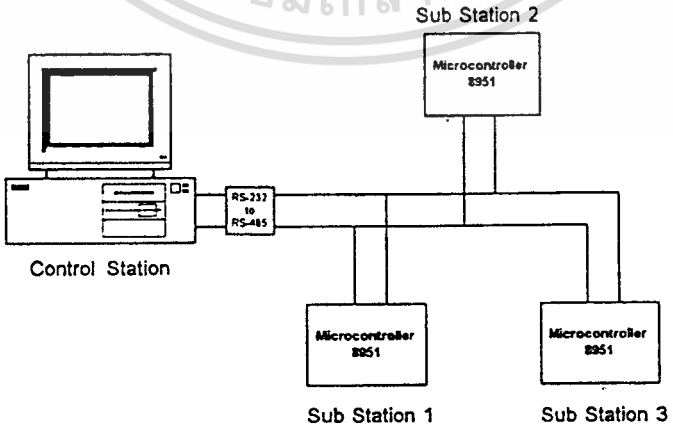
สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในงานอุตสหกรรมบางแห่งหรือระบบควบคุมบางระบบ ศูนย์ควบคุมต้องอยู่ห่างจากอุปกรณ์ที่ต้องการควบคุมเป็นระยะทางไกล ต้องมีการเดินสายสัญญาณ หรือสายไฟฟ้าเป็นจำนวนมาก ทำให้การติดตั้งยุ่งยาก และเสียค่าใช้จ่ายมาก อีกทั้งยังยากต่อการบำรุงรักษา ยกตัวอย่างเช่น ระบบรักษาความปลอดภัยในหมู่บ้านจากสภาพความเป็นไปในปัจจุบันทำให้บุคคลในสังคมดำเนินงานต่างๆ ด้วยความเร่งรีบ และเกิดการแข่งขันกันขึ้นในทุกๆ ด้านทำให้เวลาอยู่กับบ้านน้อยลงเนื่องจากส่วนใหญ่ต้องออกไปทำงานนอกบ้านแต่เช้าตรู่ และเดินทางกลับมาก็มืดค่ำ ทำให้ที่พักอาศัยขาดการดูแลเอาใจใส่ สิ่งมีค่าต่างๆ ที่เราได้สั่งสมมาเป็นเวลาอันยาวนานอาจสูญหายไปได้โดยง่ายจากผู้บุกรุกหรืออุบัติภัยต่างๆ ที่เราคาดไม่ถึงแม้ในปัจจุบันจะมีการจ้างยามรักษาความปลอดภัยมาช่วยดูแลให้เรา แต่เขาก็ไม่สามารถเข้ามาดูแลถึงภายในบ้านได้ ดังนั้นโครงงานนี้จึงนำเสนอแนวทางแก้ปัญหาซึ่งจะทำให้ยามรักษาความปลอดภัยสามารถดูแลได้ถึงภายในที่พักอาศัยได้ โดยผ่านทางโครงข่ายของระบบรักษาความปลอดภัยระยะไกล ซึ่งสามารถจะดูแลระบบได้บนเครื่องไมโครคอมพิวเตอร์ได้อย่างง่ายดาย

จะเห็นได้ว่าจากตัวอย่างที่กล่าวในข้างต้นนั้นศูนย์ควบคุมกับอุปกรณ์ที่ใช้ควบคุมนั้นอยู่ห่างกันจึงต้องเดินสายสัญญาณและสายไฟเป็นจำนวนมากซึ่งไม่สะดวก เราอาจนำเอาโครงงานนี้มาช่วยในการติดต่อสื่อสารกันได้โดย ระบบเก็บข้อมูลและควบคุมระยะไกลตามมาตราฐาน RS-485 จะเน้นการออกแบบให้สามารถติดตั้งหรือปรับปรุงเปลี่ยนแปลงได้สะดวกการซ่อมบำรุงง่ายและเสียค่าใช้จ่ายถูก โดยใช้สาย 2 เส้นมีฉนวลหุ้ม อ้างอิงตามมาตราฐาน EIA RS-485



รูปที่ 1.1 แสดงโครงข่ายทั้งระบบของระบบรักษาความปลอดภัยระยะไกล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปจะประกอบไปด้วยสถานีควบคุม 1 สถานี จะเชื่อมต่อกับสถานีย่อยได้ถึง 32 โมดูล โดยที่ทั้ง 2 เครือข่ายใช้สายสัญญาณเพียง 2 เส้นเท่านั้น โดยที่การเชื่อมต่อระหว่างโมดูลควบคุมกับไมโครคอมพิวเตอร์ผ่านทางพอร์ทอนุกรม โดยการทำงานโดยคร่าวๆ ของระบบรักษาความปลอดภัยระยะไกล จะแบ่งเป็น 2 ส่วน ด้วยกันคือ ภายในระบบเครือข่ายประกอบด้วยเครื่องไมโครคอมพิวเตอร์ 1 ตัวทำหน้าที่เป็นตัวรับ ส่งข้อมูลจากโหนดที่มีทั้งหมด 3 โหนด โดยเราจะเรียกว่า สถานีควบคุม ซึ่งจะทำหน้าที่ควบคุมการสื่อสารทั้งหมดของระบบและติดต่อกับผู้ใช้ ส่วนสถานีย่อยทำหน้าที่ตอบสนองคำสั่งจากสถานีควบคุม อ่านค่าสถานะต่างๆ จากโมดูลซึ่งเป็นสวิตช์และส่งไปยังสถานีควบคุมเพื่อทำการแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและหลักการ

การทำงานในทางทฤษฎีนั้นจะเป็นส่วนของการติดต่อสื่อสารระหว่างศูนย์คอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ซึ่งจะใช้การติดต่อแบบ อนุกรม การติดต่อสื่อสารที่ต้องทำการส่งข้อมูลออกจากไมโครคอนโทรลเลอร์ที่เก็บค่าสถานะต่างๆ ของตัวตรวจจับที่ตรวจจับได้จากส่วนที่หนึ่งและ การติดต่อในส่วนที่สองนี้มีหน้าที่ที่จะต้องรอรับการร้องขอของ HOST ที่ติดตั้งไว้ที่ศูนย์ควบคุมหลัก การติดต่อโดยส่วนใหญ่ของระบบการสื่อสารนี้เราจะใช้หลักการแบบอนุกรมโดยใช้มาตราฐานการติดต่อ RS-485 โดยหลักการนั้นเราจะอาศัย PORT สื่อสารอนุกรมที่มีมาด้วยกับไมโครคอนโทรลเลอร์

2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ที่ใช้ในโครงงานการส่งข้อมและควบคุมระยะไกลนี้เป็นเบอร์ 8031 ซึ่งเป็นไมโครคอนโทรลเลอร์ที่เหมาะสำหรับการควบคุมอุปกรณ์ต่างๆขนาด 8 บิตทำงานได้โดยการเขียนโปรแกรมควบคุมการทำงาน ถูกบรรจุอยู่ในวงจรรวม(SINGLE CHIP) ขนาด 40 ขา ดังรูป 2.1 ซึ่งสามารถที่จะออกแบบให้ระบบมีขนาดเล็ก สามารถตรวจสอบความผิดพลาดได้ง่ายและช่วยลดปัญหาสัญญาณรบกวนในระบบที่จะทำให้ระบบเกิดการทำงานที่ผิดพลาดได้

2.1.1 รายละเอียดของขาต่างๆ

การจัดเรียงขาต่างๆของ 8031 แสดงไว้ดังรูปที่ 2.1 จากรูปขาสัญญาณทั้ง 40 ขามีรายละเอียดดังนี้

VCC ขา 40 เป็นขาที่ต้องการไฟเลี้ยง +5V ให้แก่ 8031

VSS ขา 20 เป็นขาที่ต่อกับ GROUND ของแหล่งจ่ายไฟ

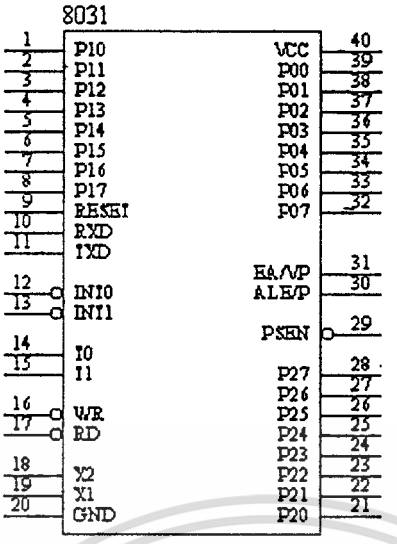
PORT 0 ขา 39-32 เริ่มจาก P0.0-P0.7 ตามลำดับโดย P0.0 เป็น LEAST SIGNIFICANTBIT

PORT 1 ขา 1-8 เริ่มจาก P1.0-P1.7 ตามลำดับ

PORT 2 ขา 21-28 เริ่มจาก P2.0-P2.7 ตามลำดับ

PORT 3 ขา10-17 เริ่มจาก P3.0-P3.7 ตามลำดับ โดยที่ PORT 3 นี้จะมีการทำงานเป็น FUNCTION อื่นด้วยโดยใช้คำสั่งควบคุมการทำงานโดยแต่ละบิทของ PORT 3 มี FUNCTION ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 แสดงการจัดเรียงขาต่างๆ ของ 8031

P3.0 / RXD (SERIAL INPUT PORT) เป็นขาที่ใช้รับข้อมูลอนุกรม

P3.1 / TXD (SERIAL OUTPUT PORT) เป็นขาที่ใช้ส่งข้อมูลอนุกรม

P3.2 / INTO (EXTERNAL INTERRUPT) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.3 / INTI (EXTERNAL INTERRUPT) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.4 / TO (TIMER / COUNTER EXTERNAL INPUT) ทำหน้าที่นับ CYCLE ของสัญญาณที่ป้อนมายัง TO

P3.5 / TI (TIMER / COUNTER 1 EXTERNAL INPUT) ทำหน้าที่นับ CYCLE ของสัญญาณที่ป้อนเข้ามายัง TI

P3.6 / WR (EXTERNAL DATA MEMORY WRITE SYROBE) ขาสัญญาณควบ-คุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอกของ 8031

RST ขา 9 ขา REST ขานี้จะทำการ Reset การทำงาน 8031 ถ้าป้อนสัญญาณ HIGH เข้าไป ALE ขา 30 (ADDRESS LATCH ENABLE) ขานี้จะส่งสัญญาณความถี่ 1/6 เท่าของ CLOCK ตลอดเวลา สัญญาณนี้ใช้บอกกับอุปกรณ์ภายนอก 8031 ว่าขณะที่ ACTIVE (HIGH) จะมีการส่งข้อมูลที่เป็น 8 BIT ล่างของ EXTERNAL MEMORY ที่ 8031 ต้องการติดต่อออกไปทาง PORT 0 จะส่ง ADDRESS ออกมาเพียงชั่วขณะเท่านั้น ซึ่งในเวลาต่อมา PORT 0 จะใช้รับส่งข้อมูลกับ EXTERNAL MEMORY

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PSEN ขา 29 (PROGRAM STORE ENABLE) ปกติ จะเป็น LOGIC 1 แต่จะส่ง LOGIC 0 ออกไป เมื่อต้องการอ่านคำสั่ง (FETCH INSTRUCTION) ที่นำมาจาก EXTERNAL PROGRAM MEMORY

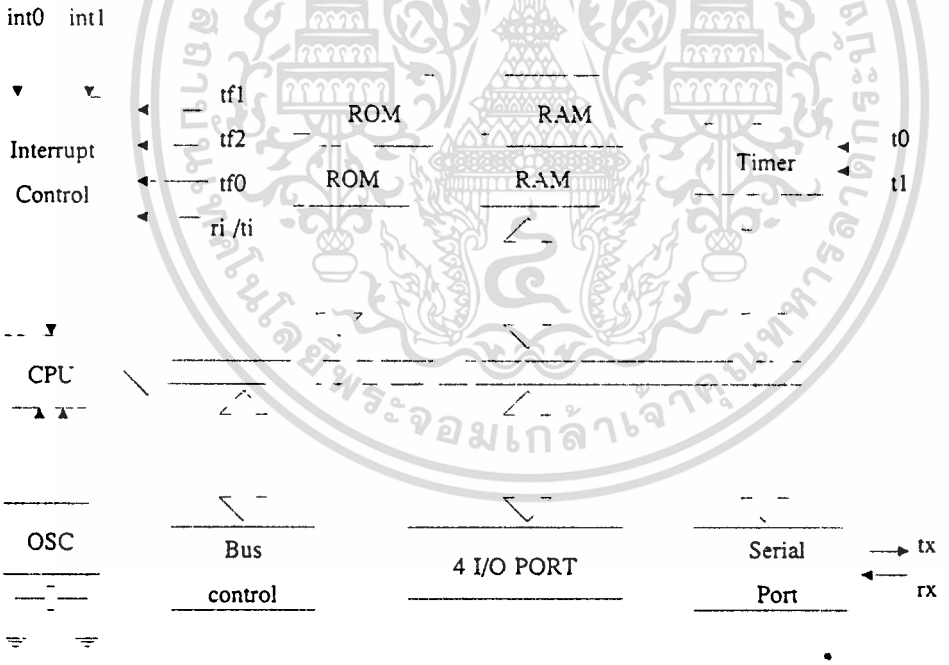
EA ขา 31 (EXTERNAL ACCESS) เป็นขา INPUT ที่ต่อเข้าไปยังวงจร TIMING AND CONTROL เพื่อควบคุมการสร้างสัญญาณ PSEN ถ้าป้อน LOGIC เข้าที่ขานี้จะทำให้ 8031 สร้างสัญญาณ PSEN ออกไปภายนอกเพื่อ FETCH คำสั่งออกไปเรียกใช้ PROGRAM MEMORY จากภายนอก

XTAL1 ขา 19 ขานี้จะต่อกับสัญญาณ OSCILLATOR จากภายนอกถ้าต้องการใช้ CLOCK จากภายนอก

XTAL2 ขา 18 ขานี้จะปล่อยลอยไว้ถ้าใช้ EXTERNAL OSCILLATOR แต่ถ้าใช้ INTERNAL OSILLATOR ต้องต่อใช้งานร่วมกับ XTAL1

2.1.2 โครงสร้างของ 8031

โครงสร้างของ 8031 แสดงเป็นบล๊อกไดอะแกรมได้ดังนี้



รูปที่ 2.2 โครงสร้างของ 8031

โครงสร้างภายใน 8031 ประกอบด้วยส่วนต่างๆ ซึ่งออกแบบมาให้มีหน้าที่การทำงานที่แตกต่างกันดังนี้

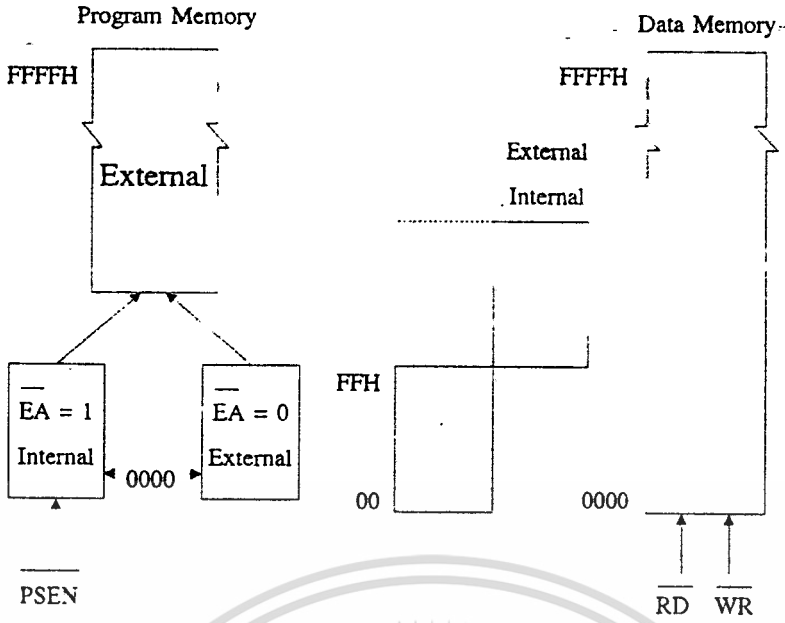
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนที่ 1 คือส่วนของ CPU (CENTRAL PROCESSING UNIT) หรือหน่วยประมวลผลกลางส่วนนี้ทำหน้าที่ในการสร้างสัญญาณ 3 ชนิดคือ CONTROL SINGNAL ADDRESS SIGNAL และ DATA SIGNAL ส่งไปยังบัสภายในของ ซีพียู สัญญาณ CONTROL มีหน้าที่ในการควบคุมการทำงานของระบบโดยอ้างอิงกับสัญญาณนาฬิกาของระบบ เพื่อให้การทำงานเป็นไปอย่างสอดคล้องกันอย่างถูกต้อง สัญญาณ ADDRESS มีหน้าที่ส่ง ADDRESS ของ MEMORY ออกไปอ้างอิงถึงหน่วยความจำที่ต้องการ READ หรือ WRITE สัญญาณ DATA คือส่วนของข้อมูลที่อ่านหรือเขียนลงไปยังหน่วยความจำหน้าที่ของซีพียูอีกอย่างคือ ทำหน้าที่ประมวลผลโดย ALU (ARITHMETIC LOGIC UNIT) จะทำการประมวลผลข้อมูล เช่นการบวก ลบ คูณ หาร แล้วนำผลลัพธ์เก็บลงไปใน REGISTER หรือ MEMORY

ส่วนที่ สองคือส่วนของ MEMORY ทำหน้าที่ในการเก็บข้อมูลแบ่งออกเป็น 2 แบบ ตามลักษณะการใช้งาน แบบแรกคือ PROGRAM MEMORY เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง หน่วยความจำแบบนี้จะเป็นแบบ READ ONLY MEMORY (ROM) คือสามารถอ่านได้อย่างเดียว จำนวนตำแหน่งสูงสุดที่ 8031 จะสามารถเข้าถึงได้คือ 65536 ตำแหน่ง เขียนเป็นเลขฐาน 16 ได้ตั้งแต่ 0000H-FFFFH ซึ่งความจุของ ROM ภายในจะขึ้นอยู่กับไมโครคอนโทรลเลอร์แต่ละเบอร์ หน่วยความจำประเภทที่สองคือ DATA MEMORY จะใช้สำหรับพักหรือเก็บข้อมูลหน่วยความจำประเภทนี้เรียกว่า RANDOM ACCESS MEMORY (RAM) หน่วยความจำนี้ภายใน 8031 จะแบ่งออกเป็น 2 ชุดคือชุดที่อยู่ภายในกับชุดที่อยู่ภายนอก ชุดที่อยู่ภายในจะมีจำนวน 128 ไบต์ที่ตำแหน่ง 00H-7FH หากเป็นเบอร์อื่นอาจมีมากกว่านี้ และชุดที่อยู่ภายนอกจะมีได้สูงสุด 65536 ไบต์ดังรูปที่ 2.3

ส่วนที่ สาม อินพุต/เอาต์พุตพอร์ตไมโครคอนโทรลเลอร์ 8031 จะมีคำสั่งที่สามารถแยกพอร์ตอินพุต/เอาต์พุต ได้โดยไม่เกี่ยวข้องกัน พอร์ต 1,2,3 สามารถที่จะขับ ทีทีแอลหรือ NMOS MCS-51 ทั้งแบบ HMOS CHMOS สามารถทำงานได้ในลักษณะ OPEN-COLLECTOR และเอาต์พุต OPEN-DRAIN โดยไม่ต้องมีพูลอัพภายนอก โดยที่พอร์ต 1 สามารถที่จะขับทีทีแอลได้ 8 ตัว เมื่อทำการรีเซต 8031 จะเขียนลอจิก 1 ไปที่แต่ละขาของพอร์ต 0 ซึ่งเป็นการบังคับให้ส่วนทางด้านเอาท์พุตเป็นอิมพลีแดนซ์สูง (HIGH IMPEDANCE) เนื่องมาจากวงจรภายในเป็นแบบ OPEN DRAIN OUTPUT ส่วนพอร์ต 1,2,3 จะถูกเขียนด้วยลอจิก 1 เป็นการบังคับให้แต่ละขาอยู่ในโหมดของเอาต์พุตแต่จะไม่เป็นอิมพลีแดนซ์สูง เนื่องจากวงจรภายในเป็นแบบ WEAK DEPLETION PULLUP FET ที่เหมาะสำหรับเชื่อมต่อเข้ากับ ทีทีแอล ใน MCS-51 มีพอร์ต 4 พอร์ต แบบสองทิศทาง พอร์ต 0 และพอร์ต 2 ที่จะทำหน้าที่ติดต่อกับหน่วยความจำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 โครงสร้างของหน่วยความจำ 8031

ภายนอกโดยที่พอร์ต 0 เป็นตัวกำหนดไบต์ต่ำของแอดเดรสหน่วยความจำภายนอก โดยค่าแอดเดรสและค่าของข้อมูลจะถูกมัลติเพลกซ์ด้วยช่วงจังหวะการอ่านและการเขียน ส่วนพอร์ตที่ 2 จะทำหน้าที่เป็นตัวกำหนดแอดเดรสทางด้านสูงในการเข้าถึงข้อมูลภายนอก ส่วนพอร์ต 1 และพอร์ต 3 จะทำหน้าที่หลายอย่างในส่วนของพอร์ต 1 ของ 8031 สามารถที่จะทำเป็นพอร์ตอินพุตและเอาต์พุตได้

2.1.3 การอินเทอร์รัปต์

การอินเทอร์รัปต์ของ 8031 จะมีแหล่งเกิดอินเทอร์รัปต์ได้สองแหล่งคือเกิดอินเทอร์รัปต์จากไทม์เมอร์/เคาน์เตอร์ทั้งสองชุดได้หนึ่งแหล่ง และเกิดอินเทอร์รัปต์จากพอร์ตอนุกรม การเกิดการอินเทอร์รัปต์แต่ละแหล่งจะมีเวกเตอร์แอดเดรส ดังแสดงในตารางที่ 2.1 เราสามารถที่จะกำหนดให้เกิดการอินเทอร์รัปต์ หรือไม่ยอมให้เกิดการอินเทอร์รัปต์ได้ และยังสามารถที่จะกำหนดลำดับความสำคัญของการอินเทอร์รัปต์และยังมีการเกิดการอินเทอร์รัปต์จากภายนอกที่ผ่านทางไทเมอร์/เคาน์เตอร์จะทำให้เพิ่มค่าที่ตั้งไว้ผ่านทางขา T0 และ T1

2.1.4 การรับ-ส่งข้อมูลอนุกรม

การรับ-ส่งข้อมูลแบบอนุกรม ที่ใช้ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวเบอร์ 8031 โดยหลักการทั่วไปแล้วในการทำงานของไมโครคอนโทรลเลอร์แบบชิพเดี่ยวจะมีการทำงานที่คล้ายกัน ซึ่งในการรับ-ส่งข้อมูลแบบอนุกรมจะมีพอร์ตอนุกรมที่สามารถกำหนดการทำงานได้หลายโหมด ซึ่งในที่นี้จะกล่าวถึงเฉพาะการรับ-ส่งข้อมูลแบบอนุกรมในโหมด 3 เท่านั้น ในการทำงานของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แหล่งการเกิดอินเทอร์รัปต์	เวกเตอร์แอดเดรส
IE0	0003h
TF0	000Bh
IE1	0013h
TF1	001Bh
RI+TI	0023h

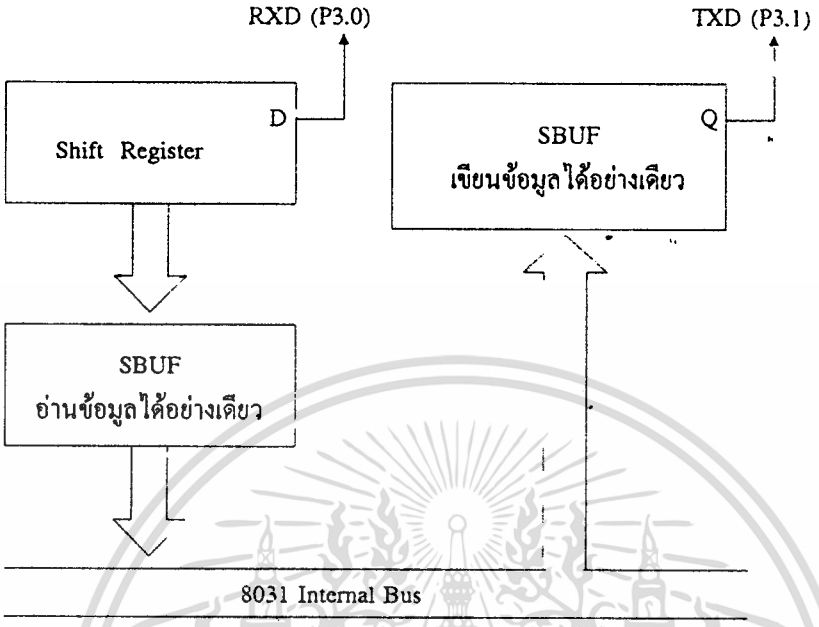
ตารางที่ 2.1 อินเทอร์รัปต์เวกเตอร์ 8031

โหมดนี้จะเป็นแบบฟูลดูเพล็ก (FULL DUPLEX) คือสามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน เนื่องจากมีรีจิสเตอร์ (REGISTER) สำหรับส่งและรับข้อมูลแยกออกจากกัน แต่จะมีชื่อเดียวกันคือ SBUF (SERIAL PORT BUFFER) ที่ตำแหน่งแอดเดรส 99H SBUF จะทำหน้าที่ทางด้านรับข้อมูลแบบอนุกรมที่เข้ามาและ SBUF อีกตัวหนึ่งจะทำการเก็บข้อมูลก่อนที่จะส่งออกจาก 8031 เมื่อมีข้อมูลอนุกรมส่งมา ข้อมูลนั้นจะถูกเก็บอยู่ใน SBUF (READ-ONLY) ในขณะที่รับข้อมูลตัวที่สองหาก 8031 อ่านข้อมูลตัวแรกต่อไปก่อนก็จะรับข้อมูลตัวที่สองเสร็จข้อมูลก็จะไม่สูญหาย โครงสร้างทางกายภาพของพอร์ตอนุกรมได้แสดงดังรูป 2.4

ในการรับ-ส่งข้อมูลอนุกรมของ 8031 จะมีขาสัญญาณที่ใช้สำหรับรับและส่งข้อมูลคือ RXD (P3.0) และ TXD (P3.1) ซึ่งจะเป็นขาที่ใช้ต่อกับวงจรภายนอก ในการทำงานของพอร์ตอนุกรมในโหมด 3 จะมีรีจิสเตอร์ที่ใช้ควบคุมการทำงานและกำหนดรูปแบบการรับส่งข้อมูล คือ Register SCON (SERIAL PORT CONTROL REGISTER) ที่ตำแหน่งแอดเดรส 98H และใช้ไทเมอร์ (timer) เป็นตัวกำหนดอัตราเร็วในการส่งข้อมูลแต่ละบิต (BOUND RATE) ซึ่งจะขึ้นอยู่กับอัตราการเกิดโอเวอร์โฟลว์ (OVERFLOW) ของไทเมอร์ ในการเลือกโหมดการทำงานจะต้องกำหนดในรีจิสเตอร์ SCON การรับ-ส่งข้อมูลในโหมด 3 จะเป็นแบบ 11 บิต ดังรูปที่ 2.4 ในการส่งข้อมูลจะเริ่มด้วยการส่งบิตเริ่มต้นไปก่อน (START BIT) แล้วตามด้วยบิตข้อมูลอีก 9 บิต (โดยบิต 0 จะถูกส่งไปก่อน) จากนั้นจึงเป็นการส่งบิตหยุด (STOP BIT) ทางด้านรับ เมื่อสายสัญญาณเปลี่ยนจากสภาวะจาก “0” ไปเป็น “1” (ที่ขอบบวกของบิตเริ่มต้น) ข้อมูลบิตเริ่มต้นจะถูกข้ามไปไม่สนใจ แล้วจึงสุ่มเอาข้อมูล 9 บิต ที่เหลือเข้ารีจิสเตอร์แบบเลื่อนบิตภายในพอร์ตอนุกรมซึ่งข้อมูลทั้ง 9 บิตนี้ได้กำหนดให้เป็นข้อมูลจริง 8 บิต ส่วนข้อมูลในบิตที่ 9 จะใช้เป็นบิตที่ใช้ตรวจสอบจุดสิ้นสุดของเฟรมข้อมูล (DATA FRAME) บิตที่ 9 ของข้อมูลจะถูกส่งออกไปจะต้องถูกโหลดไปยังบิต TBB ของ SCON ก่อนโดยใช้ซอฟแวร์ (ในกรณีที่เป็นการส่งข้อมูล) ส่วนในการรับข้อมูลบิตที่ 9 จะถูกเก็บในบิต RBB ของ SCON โดยอัตโนมัติ ซึ่งในทางปฏิบัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แฟลก RI จะเป็น "1" เมื่อสิ้นสุดการรับข้อมูลและสามารถตรวจสอบโดยซอฟแวร์ หรือโปรแกรมให้เกิดการอินเทอร์รัพต์ก็ได้



รูปที่ 2.4 บล๊อกไดอะแกรมของพอร์ตอนุกรม

ต้องการรับข้อมูลชุดต่อไปจะต้องทำการเคลียร์แฟลก RI เสียก่อน ส่วนในการส่งข้อมูลจะต้องตรวจสอบว่าบิต TI เป็น "1" หรือไม่ หากเป็น "1" แสดงว่าข้อมูลในบัฟเฟอร์ว่างสามารถส่งข้อมูลชุดใหม่ได้

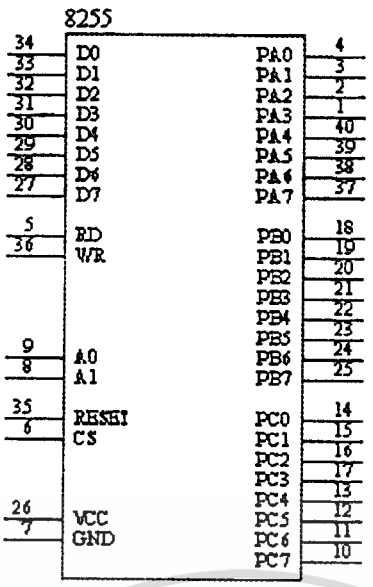
2.2 8255 Programable Peripheral Interface (PPI)

เป็นส่วนที่ทำหน้าที่ติดต่อกับอุปกรณ์รอบข้างได้แก่ ควบคุมการทำงานของตัวตรวจจับอุปกรณ์ไฟฟ้า และ ส่วนของโทรศัพท์ โดยการใช้พอร์ตสำหรับการติดต่อได้แก่พอร์ต A, B และ C โดยมีการรับข้อมูลจากไมโครคอนโทรลเลอร์ทาง DATA BUS

2.2.1 ขาสัญญาณต่างๆ ของ 8255

ส่วนนี้เป็นหน้าที่ของขาแต่ละขาของ 8255 ซึ่งข้อมูลเหล่านี้จะใช้ประโยชน์ในการเชื่อมต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ สำหรับขาต่างๆ แสดงดังรูป 2.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 แสดงการจัดเรียงขาต่างๆ ของ 8255

รายละเอียดของขามีดังนี้

D0-D7 - เป็นขาที่ทั้งข้อมูลอินพุทและเอาท์พุต ต้องผ่านเข้าออก จะต่อเข้ากับบัสของไมโครโปรเซสเซอร์ เพื่อให้สามารถอ่านเขียนข้อมูลจากพอร์ตเหล่านี้ได้

CS - ขานี้เป็นขาสัญญาณอินพุทที่จะรับสัญญาณภายนอกเพื่อเลือกชิป 8255 เพื่อให้ไมโครโปรเซสเซอร์สามารถเขียนหรืออ่านข้อมูลจากพอร์ตได้

RD -เป็นสัญญาณอินพุทที่ต้องส่งมาจากซีพียูเมื่อสัญญาณที่ขานี้เป็น 0 และสัญญาณที่ CS เป็น 0 ด้วย 8255 จะให้ซีพียูอ่านข้อมูลจากบัสในขณะที่เป็นพอร์ตอินพุท

WR - เป็นสัญญาณการเขียน จะแอคทีปเมื่อสัญญาณ WR และสัญญาณ CS เป็น 0 สัญญาณนี้จะมาจาก ซีพียูเมื่อต้องการเขียนข้อมูลลงพอร์ตที่กำหนด

A0-A1 - ลอจิกของสัญญาณทั้งสองจะถอดรหัสออกมาเป็น 4 รหัสเพื่อกำหนด Resister ภายในที่เชื่อมต่อกับพอร์ตอินพุทเอาท์พุทของพอร์ต 8255

RESET - เป็นสัญญาณที่ส่งมาจากภายนอกเข้ามาทำการรีเซ็ท 8255 เพื่อเคลียร์สถานะต่างๆของ 8255

PA0-PA7 - เป็นสายสัญญาณที่เป็นพอร์ตของ 8255 ที่ชื่อพอร์ต A การเลือกพอร์ตเลือกโดยสัญญาณแอดเดรส A0-A1

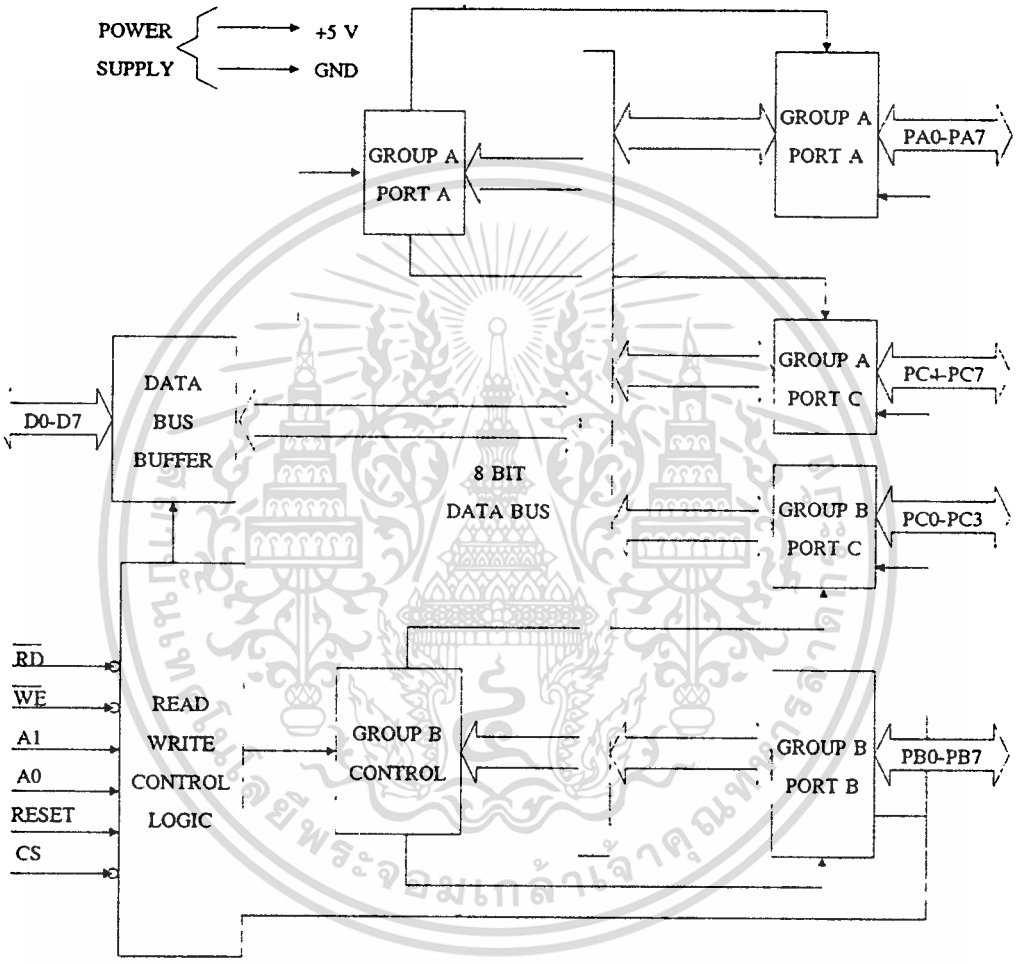
PB0-PB7 - เป็นสายสัญญาณที่เป็นพอร์ตของ 8255 ที่ชื่อพอร์ต B การเลือกพอร์ตเลือกโดยสัญญาณแอดเดรส A0-A1

PC0-PC7 - เป็นสายสัญญาณที่เป็นพอร์ตของ 8255 ที่ชื่อพอร์ต C การเลือกพอร์ตเลือกโดยสัญญาณแอดเดรส A0-A1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 โครงสร้างภายใน 8255

8255 เป็นอุปกรณ์ LSI (LARGE SCALE INTEGRATED CIRCUIT) บรรจุอยู่ในแพกเกทขนาด 40 ขา แบบ DIP(DUAL-IN-LINE PACKAGE) คุณสมบัติเบื้องต้นของการนำ 8255 มาใช้งานคือเป็นอินพุต/เอาต์พุตพอร์ต โดยมีพอร์ตขนาด 8 บิต ที่สามารถสั่งให้เป็นอินพุตหรือเอาต์พุตก็ได้จำนวน 3 พอร์ต รูปที่ 2.6 แสดงโครงสร้างภายในของ 8255 ซึ่งหน้าที่การทำงานแต่ละบล๊อกมีดังนี้



รูปที่ 2.6 โครงสร้างภายในของ 8255

บล๊อกกลุ่มแรกได้แก่บล๊อกส่วนที่อยู่ขวามือของรูปซึ่งเป็นส่วนที่เชื่อมต่อกับอุปกรณ์ภายนอกอื่นๆ โดยมีขา PA0-PA7,PB0-PB7 และ PC0-PC7 เป็นทางผ่านของข้อมูลระหว่างอุปกรณ์ภายนอกกับ 8255 ขาสัญญาณเหล่านี้จะถูกแบ่งออกเป็นอินพุต เอาต์พุต 3 พอร์ตได้แก่ พอร์ต A(PA) พอร์ต B(PB)และพอร์ต C(PC) พอร์ตเหล่านี้แต่ละพอร์ตสามารถเป็นได้ทั้งพอร์ตอินพุตและเอาต์พุตแล้วแต่ผู้ใช้จะกำหนดแต่ละบล๊อกจะมีสัญญาณเชื่อมเข้ากับบัสข้อมูลภายในของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8255 บล็อกกลุ่มถัดมาได้แก่ GROUP A CONTROL และ GROUP B CONTROL ทำหน้าที่เป็นตัวกำหนดลักษณะการทำงานของพอร์ตอินพุตเอาต์พุตทั้ง 3 พอร์ต (8255 มีลักษณะการทำงานที่แตกต่างกันอยู่ 3 โหมด สามารถกำหนดได้จากการโปรแกรมโดยส่งคำสั่งควบคุม) ให้กับ 8255 ในพอร์ตนี้ประกอบด้วยพอร์ตขนาด 4 บิต จำนวน 2 พอร์ตได้แก่ GROUP A CONTROL กับ GROUP B CONTROL โดย GROUP CONTROL จะควบคุมพอร์ต A กับพอร์ต C สี่บิตด้านสูง ส่วน GROUP B CONTROL จะควบคุมพอร์ต B กับ พอร์ต C สี่บิตด้านต่ำบล็อกสุดท้ายได้แก่ บัฟเฟอร์ของบัสข้อมูล(DATA BUS BUFFER) และส่วนควบคุมการอ่าน/เขียน(READ/WRITE CONTROL) ซึ่งบล็อกเหล่านี้จะเป็นส่วนติดต่อกับไมโครโปรเซสเซอร์ ทำหน้าที่เป็นบัฟเฟอร์ให้กับบัสข้อมูลของไมโครโปรเซสเซอร์และส่วนที่ควบคุมให้ข้อมูลเข้าออกจากรีจีสเตอร์ภายในตัวให้ถูกต้องสอดคล้องกับการทำงานของระบบ

2.2.3 การควบคุมการอ่านหรือเขียนข้อมูล

จากระบบบัสที่เป็นแบบสองทิศทางจะเห็นได้ว่าการติดต่อระหว่าง 8255 กับซีพียูจะผ่านบัสข้อมูลทั้งแปดเส้น และถ้ามองจากซีพียูจะเห็นว่ามีพอร์ตอยู่ 4 พอร์ตโดยการอ้างพอร์ตให้ใช้ A0 กับ A1 ประกอบกับสัญญาณ CS ซึ่งทำงานที่ 0

A1	A0	RD	WR	CS	
					Input operation (READ)
0	0	0	1	0	Port A-->Data Bus
0	1	0	1	0	Port B-->Data Bus
1	0	0	1	0	Port C --> Data Bus
					Output operation(WRITE)
0	0	1	0	0	Port A ---> Data Bus
0	1	1	0	0	Port B ---> Data Bus
1	0	1	0	0	Port C ---> Data Bus
1	1	1	0	0	Control ---> Data Bus
					Disable Function
X	X	X	X	1	Data Bus ---> 3-state
1	1	0	1	0	Illegal condition
X	X	1	1	0	Data Bus ---> 3-state
Source:Courtesyof Intel Corporation					

ตารางที่ 2-2 ตารางแสดงการอ่านเขียนของ 8255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตาราง 2-2 เป็นสรุปการอ่านและเขียนของ 8255 เมื่อสัญญาณ RD เป็น 0 จะเป็นการอ่านค่าจากพอร์ตใดพอร์ตหนึ่งจาก 3 พอร์ต ซึ่งถูกเลือกด้วย A0 กับ A1 แต่ถ้า A0 กับ A1 เป็น 1 ทั้งคู่ ก็จะเป็นการระบุว่าต้องการติดต่อกับพอร์ตควบคุม ซึ่งเป็นรีจิสเตอร์พิเศษสำหรับควบคุมการทำงานของ 8255 ซึ่งรีจิสเตอร์นี้ซีพียูสามารถเขียนค่าลงไปได้แต่ไม่สามารถอ่านค่าได้ เมื่อ 8255 ไม่ได้ถูกติดต่อด้วย (CS = 1 หรือ RD และ WR = 1) บัสของ 8255 ที่ติดต่อกับซีพียูจะอยู่ในสถานะ High Impedance ซึ่งจะเป็นการแยกตัวออกจากระบบเพื่อให้ซีพียูติดต่อกับระบบอื่นๆ

2.2.4 โหมดการทำงานของ 8255

- Mode 0 : Basic I/O

เมื่อ 8255 เชื่อมต่อกับซีพียูจะต้องมีการกำหนดโหมดการทำงาน เป็นโหมด 3 โหมดหรืออาจจะเพิ่มโหมดบิทเซ็ท/รีเซ็ทก็ได้ สมมุติว่าถ้าเลือกโหมดการทำงานแบบ Nonhandshaking หรือแบบไม่มีเงื่อนไขในโหมด 0 โหมดนี้มันจะต้องส่งคำสั่งควบคุมหนึ่งเวิร์ดไปยังพอร์ตควบคุม ซึ่งควบคุมโหมดของ 8255

- Mode 1 ; Strobed I/O

ใน Mode นี้เป็นเรื่องเกี่ยวกับการตรวจสอบสัญญาณ handshak และการอินเทอรัปต์ของการติดต่อระหว่าง I/O ในโหมดนี้พอร์ต A และ B จะเป็นพอร์ตข้อมูล ส่วนพอร์ต C ถูกใช้สำหรับการสร้างและตรวจสอบสถานะสัญญาณเรียกว่าพอร์ตควบคุม ในโหมดนี้การทำ DATA TRANSFER นี้ซีพียูไม่สามารถแทรกแซงการทำงานของ 8255 ได้ ลักษณะการทำงานของ 8255 ในโหมด 1 จะมีการเลือกว่าจะให้พอร์ต A B เป็น Input หรือ Output รวม 4 แบบ นอกจากนี้ยังสามารถสั่งให้พอร์ต A ทำงานที่โหมด 1 ขณะที่ พอร์ต B ทำงานที่โหมด 0 ได้อีก

- Polling และ Interrupts

การพอลลิง 8255 ในโหมด 1 โดยการอ่านข้อมูลจากพอร์ต C เป็นการโหลด Status Word ของโหมด 1 มาไว้ที่ Accumulator ซึ่งถ้าพอร์ต A เป็นพอร์ตอินพุท และพอร์ต B เป็นพอร์ตเอาท์พุท ทำให้สามารถตรวจสอบได้ว่าเป็นค่า 0BF หรือ 1BF ได้ PC 6 หรือ 7 (พอร์ต A เป็น Input) หรือ PC 4 หรือ 5 (พอร์ต A เป็น Output) ถ้าไม่ได้ใช้ในการทำ Handshak ก็อาจจะสั่งให้ทำงานเป็น I/O ทั่วไปได้ และอาจจะถูกอ่านเป็นค่าส่วนหนึ่งของสถานะของโหมด 1 ถ้าใช้การทำงานแบบอินเทอรัปต์ สัญญาณ INTR จะถูกเซ็ทเมื่อ ACK กลายเป็น 1 (Buffer ว่างสำหรับข้อมูลแล้ว) หรือเมื่อ STB กลายเป็น 1 (ที่ I/P Buffer มีข้อมูลแล้ว) INTR อาจถูกเซ็ทถ้า INTE ที่สอดคล้องกันได้ถูกเซ็ทโดยซีพียู ซึ่งกลายเป็นโหมดบิทเซ็ท/รีเซ็ทไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Mode 2 : Strobed Bidirectional I/O

ในโหมด 2 พอร์ต A ของ 8255 จะกลายเป็นพอร์ตรับส่งข้อมูลแบบ สองทิศทางมีการทำงานด้วยวิธี Handshake จากสัญญาณจำนวน 5 สัญญาณ สัญญาณ Handshake เหล่านี้จะมีลักษณะการทำงานดังเช่นในโหมด 1 แต่ว่าจะอ้างอิงกับพอร์ต A เท่านั้น การใช้งานในโหมดนี้ได้แก่การส่งข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง เมื่อพอร์ต A ถูกโปรแกรมเพื่อให้ทำงานในโหมด 2 พอร์ต B สามารถทำงานในโหมด 0 หรือ โหมด 1 ถ้าถูกโปรแกรมสำหรับ โหมด 0 PC0-PC2 สามารถถูกโปรแกรมให้เป็น Input หรือ Output ถ้าพอร์ต B เป็นโหมด 1 แล้ว PC0-PC2 จะกลายเป็นสัญญาณ Handshake สำหรับพอร์ตนี้ เมื่อทำการพิจารณาการโปรแกรมโหมดที่เป็นไปได้ทั้งหมดแล้ว 8255 จะมีลักษณะการทำงานอยู่ 4 ลักษณะในโหมด 2 ถ้าเลือกที่จะโปรแกรมพอร์ต A โหมด 2 พอร์ต B เป็น Input โหมด 0 และ PC0-PC2 เป็น Output โหมด 0 โดย Control Word คือ 11XX X010

2.3 หลักการภาคควบคุม

ในส่วนของภาคควบคุมจะใช้ ซีพียู 8031 เป็นตัวหลักในการควบคุม สถาปัตยกรรมภายในของ 8031ซึ่งประกอบไปด้วยส่วนสำคัญหลักๆดังนี้

-เป็น ซีพียู ขนาด 8 บิท ประกอบด้วยรีจิสเตอร์ A (แอคคิวมูเลเตอร์)และรีจิสเตอร์ B

-มีโปรแกรมเคาท์เตอร์ (Program Counter(PC)),ดาต้าพอยท์เตอร์(Data Pointer(DPTR)) ขนาด 16 บิท

-มีโปรแกรมสเตตัสเวิร์ด (Program Status Word(PSW)) ขนาด 8 บิท

-มีสแต็กพอยท์เตอร์ (Stack Pointer(SP)) ขนาด 8 บิท

-มีหน่วยความจำแบบ EPROM ขนาด 4 กิโลไบท์

-มีหน่วยความจำแรมภายในขนาด 128 ไบท์ ประกอบด้วย รีจิสเตอร์แบงค์ 4 แบงค์ แต่ละแบงค์ประกอบด้วย รีจิสเตอร์ขนาด 8 บิทจำนวน 8 รีจิสเตอร์ (R0-R7) มีหน่วยความจำจำนวน 16 ไบท์ ที่สามารถอ้างแอดเดรสเพื่อควบคุมการทำงานในระดับบิททำได้ มีหน่วยความจำสำหรับใช้งานทั่วไป 80 ไบท์

-มีขารับสัญญาณอินพุท/เอาท์พุท 32 ขา แบ่งออกเป็นกลุ่ม ๆ ละ 8 บิท ได้สี่กลุ่มคือ P0,P1,P2,P3

-มีไทม์เมอร์/เคาท์เตอร์ ขนาด 16 บิท สองชุด คือ T0 และ T1

-มีพอร์ตอนุกรมที่ใช้รับส่งสัญญาณแบบฟูลดูเพล็ก (FULL DUPLEX) เรียกว่า SBUF

มีรีจิสเตอร์ควบคุมได้แก่ TCON,TMOD,SCON,PCON,IP และ IE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-สามารถทำการอินเทอรัพท์ทั้งภายในและภายนอก การอินเทอรัพท์ภายในได้มาจากแหล่งกำเนิดการอินเทอรัพท์สามแหล่งการอินเทอรัพท์ภานนอกได้มาจากแหล่งกำเนิดการอินเทอรัพท์จากภายนอกสองแหล่ง

-มีส่วนของออสซิลเลเตอร์ และวงจรสร้างสัญญาณนาฬิกาอยู่ภายใน

การอินเทอรัพท์สามแหล่ง การอินเทอรัพท์ภายนอกได้มาจากแหล่งกำเนิดการอินเทอรัพท์จากภายนอกสองแหล่ง

-มีส่วนของออสซิลเลเตอร์ และวงจรสร้างสัญญาณนาฬิกาอยู่ภายใน

2.3.1 รายละเอียดของรีจิสเตอร์ที่ทำหน้าที่พิเศษ

Accumulator: ACCเป็นแอคคิวมูเลเตอร์หรือรีจิสเตอร์ A

B Registor: รีจิสเตอร์ B ใช้ในคำสั่งคูณและหาร ส่วนในคำสั่งอื่นสามารถใช้เหมือนกับรีจิสเตอร์ทั่วๆ ไปในการพักข้อมูล

Program Status Word: PSW ประกอบด้วยรายละเอียดดังต่อไปนี้

(MSB) (LSB)

CY AC PF0 RS1 RS0 0V - P

SYMBOL	POSITION	NAME AND SIGNIFICANCE
CY	PSW.7	CARRY FLAG
AC	PSW.6	AUXILLARY CARRY FLAG(FOR BCD OPERATION)
F0	PSW.5	FLAG 0(FOR GENERAL PURPOSES)
RS1	PSW.4	REGISTOR BANK SELECT CONTROL BITS 1&2 SET/CLEARED BY SOLFWARE TO DETERMINE
RS0	PSW.3	WORKING REGISTOR BANK(SEE NOTE)0V
	PSW.2	OVERFLOW FLAG
	PSW.1	(RESERVED)
	PSW.0	PARITY FLAG. SET/CLEARED BY HARDWARE EACH INSTRUCTION TO INDICATE AN ODD/EVEN NUMBER OF "ONE BITS INTHE ACCUMULATOR "ic."EVEN PARITY.

ตารางที่ 2.3 รายละเอียดรีจิสเตอร์ทำหน้าที่พิเศษ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Note The consists of (RS1,TS0)enable the working registor banks as follows.

(0.0)-BANK0 (00H-07H)

(0.1)-BANK1 (08H-0FH)

(0.2)-BANK2 (10H-17H)

(0.3)-BANK3 (18H-1FH)

2.3.1.1 Stack Pointer: SP เป็นรีจิสเตอร์ขนาด 8 บิท เมื่อใช้คำสั่ง PUSH และ CALLSP จะเพิ่มขึ้นก่อนที่จะเก็บข้อมูลหรือแอดเดรสซึ่ง STACK จะอยู่ที่ไหนก็ได้ในหน่วยความจำ (RAM) บน 8751 ภายหลังการRESET แสตกจะมาอยู่ที่ 17H นั่นคือแสตก จะเริ่มที่ 80H

2.3.1.2 Data Pointer: DPTR ประกอบด้วย DPH และ DPL ซึ่งจะรวมกันเป็นรีจิสเตอร์ ขนาด 16 บิท การเข้าถึง DPTR สามารถทำได้ทั้งแบบ 16 บิท และ 8 บิท หน้าที่ของ DPTR เมื่อใช้แบบ 16 บิท จะทำหน้าที่เป็นตัวชี้ข้อมูลที่อยู่ในหน่วยความจำภายนอก

2.3.1.3 Port 0-3: พอร์ต 0-3 เป็น SFR ตัวหนึ่งที่สามารถแลทช์ข้อมูลได้ สามารถใช้เป็นอินพุทพอร์ตและเอาท์พุทพอร์ตได้ทั้ง 3 พอร์ต

2.3.1.4 Serial Data Buffer: แบ่งเป็นรีจิสเตอร์บัฟเฟอร์ในทางรับเมื่อมีการส่งข้อมูลภายในให้ SBUF ข้อมูลจะถูกส่งมาที่ บัฟเฟอร์ทางด้านส่งที่ซึ่งจะทำให้การส่งข้อมูลอนุกรม การส่งข้อมูลให้ SBUF ข้อมูลจะถูกอ่านจากบัฟเฟอร์ในทางรับ

2.3.1.5 Timer Regsistor: รีจิสเตอร์คู่ (TH0,TL0) และ (TH1,TL1) เป็นตัวจับเวลาและตัวนับขนาด 16 บิท

2.3.1.6 Control Registors: รีจิสเตอร์หน้าที่พิเศษ IP, IE, TMOD, TCON, T2CON, SCON และ PCON ประกอบด้วยบิทควบคุม และบิทสถานะสำหรับระบบการอินเทอร์รัพท์

2.3.2 รายละเอียดของขาต่าง ๆ

VCC: ต่อกับไฟเลี้ยงของระบบ (5VDC)

VSS: กราวด์

PORT0: พอร์ต 0 เป็นอินพุท/เอาท์พุทแบบ 8 บิท (OPEN DRAIN) สามารถรับกระแส SINK จาก TTL(LS) ได้ 8 ตัว ถ้าเราเขียน 1 ไปที่พอร์ต 0 จะทำให้พอร์ต 0 เป็น H1-Z อินพุท พอร์ต 0 สามารถ MULTIPLEX ระหว่างบัสข้อมูลกับแอดเดรสไบท์ต่ำเมื่อใช้ติดต่อกับหน่วยความจำภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PORT1: พอร์ต 1 เป็น 8 บิทสองทิศทางมีการพูลอัพภายในพอร์ต 1 นี้ขับ TTL(LS) ได้ 4 ตัว เมื่อเขียนค่า 1H และพอร์ต 2 นี้จะให้แอดเดรสไบท์สูงในขณะติดต่อกับหน่วยความจำภายนอก

PORT3: พอร์ต 3 เป็นอินพุท/เอาท์พุทพอร์ตแบบ 2 ทิศทางมีการพูลอัพภายในพอร์ต 3 ขับ TTL (LS) ได้ 4 ตัวและยังสามารถใช้ทำงานในลักษณะพิเศษดังรายละเอียดข้างล่าง

PORT PIN ALTERNATE FUNCTION

P3.0 RXD(SERIAL INPUT PORT)

P3.1 TXD(SERIAL OUTPUT PORT)

P3.2 INT0(EXTERNAL INTERUPT0)

P3.3 INT1(EXTERNAL INTERUPT1)

P3.4 T0(TIMER 0 EXTERNAL INPUT)

P3.5 T1(TIMER 1 EXTERNAL INPUT)

P3.6 WR(EXTERNAL DATA MEMORY WRITE STROBE)

P3.7 RE(EXTERNAL DATA MEMORY READ STROBE)

RST: ขารีเซ็ทสัญญาณ HIGH ที่ขารีเซ็ทนี้นาน 2 แมชีนไซเคิลจะเป็นการรีเซ็ท CPU

ALE/PROG: ADDRESS LATCH ENABLE พัลส์สำหรับแลตซ์แอดเดรสไบท์ต่ำในระหว่างการติดต่อกับหน่วยความจำภายนอก ALE นี้ยังจ่ายความถี่คงที่ 1/6 ของความถี่ออสซิลเลเตอร์ (ถ้าไม่ติดต่อกับหน่วยความจำภายนอก) และ ALE อีกหน้าทีหนึ่งคือเป็นอินพุทรับพัลส์ (LOW) ในระหว่างโปรแกรม EPROM(8751)

PSEN: เป็นขาสัญญาณ REA STROBE ในขณะที่อ่านโปรแกรมจากภายนอก PSEN จะแอดทีฟ 2 ครั้งต่อ 1 แมชีนไซเคิล

EA/UPP: ต่อ LOW จะทำคำสั่งในโปรแกรมภายนอก ถ้าต่อ HIGH จะใช้โปรแกรมภายนอกและใช้เป็นอินพุทรับ 21 V. ในการโปรแกรม 8751H

XTAL1: เป็นขาอินพุทของอินเวอร์เตอร์ของภาคขยายความถี่

XTAL2: เป็นขาเอาท์พุทของอินเวอร์เตอร์ของภาคขยายความถี่

2.3.3 การรีเซ็ท

สัญญาณรีเซ็ทเป็นสัญญาณอินพุททางขา 9 การรีเซ็ทจะสมบูรณ์ต้องรักษาระดับ HIGH อย่างน้อยที่สุด 2 แมชีนไซเคิล(24คาบเวลาของออสซิลเลเตอร์)การรีเซ็ทภายในตัว CPU จะเริ่มในระหว่างไซเคิลที่ 2 นับตั้งแต่ขา RST เป็น HIGH ผลของการรีเซ็ทจะมีผลกับรีจิสเตอร์ดังตารางที่ 2.4 ต่อไปนี้ แต่ RAM ภายในจะไม่ถูกเคลียร์เมื่อ CPU ถูกรีเซ็ท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

REGISTOR	CONTENT
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP	(XX000000)
IE	(0X000000)
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	00H
PCON	00H

ตารางที่ 2.4 ผลกระทบจากการรีเซ็ท

2.4 ADDRESSING MODE

ตระกูล MCS-51 แบ่งการเข้าถึงหน่วยความจำได้ 5 แบบ ดังต่อไปนี้

REGISTOR ADDRESSING

-R0-R7

-ACC,B,CY(BIT),DPTR

DIRECT ADDRESSING

-LOWER 128 BYTES OF INTERNAL RAM

-SPEICAL FUNCTION REGISTER

REGISTER INDIRECT ADDRESSING

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-INTERNAL RAM (@R0,@RI,SP)

-EXTERNAL DATA MEMOTY (@R0,@R1,@DPTR)

IMMEDIATE ADDRESSING

-PROGRAM MEMORY

BASE-REGISTOR PLUS INDEX-REGISTOR INDIRECT ADDRESS

-PROGRAM MEMORY (@DPTR+A,@PC+A)

2.4.1 Register Addressing

เป็นการติดต่อกับรีจิสเตอร์ทั้ง 8 ตัวในแต่ละ BANK โดยใช้ 3 บิทล่างของ OP-CODE เป็นตัวกำหนดรีจิสเตอร์ที่จะทำการติดต่อด้วย ACC,B,DPTR และ CY และการประมวลผลทางบูลีนถือว่าอยู่ในโหมด REGISTOR ADDRESSING ตัวอย่างเช่น MOV A,R0

2.4.2 Direct Addressing

เป็นเพียงวิธีเดียวที่จะเข้าถึง SFR(SPECIAL FUNCTION REGISTOR) ได้การติดต่อกับ RAM 128 ไบท์ล่างก็ใช้โหมดนี้ด้วย ตัวอย่างเช่น MOC A.SBUF

2.4.3 Register Indirect Addressing

ในโหมดนี้ใช้ค่าที่อยู่ในรีจิสเตอร์ R0 หรือ R1 (ใน BANK ที่เลือกไว้) เป็นตัวชี้ไปยังตำแหน่งต่างๆ ภายใน 256 ไบท์ (RAM 128 ไบท์ล่างหรือ 256 ไบท์ล่างของหน่วยความจำ) ภายนอกข้อสังเกตคือ SFR ไปสามารถติดต่อได้ด้วยวิธี REGISTOR-INDIRECT นี้ส่วนการติดต่อกับหน่วยความจำข้อมูลภายนอก 64 K นั้นใช้ตัวชี้ขนาด 16 บิท (DPTR) คำสั่ง PUSH และ POP ก็ทำงานในโหมดนี้ด้วย (ใช้ SP เป็นตัวชี้)

ตัวอย่าง

```
MOV A,@R0
```

2.4.4 Immediate Addressing

โหมดนี้อนุญาตให้ใช้ค่าคงที่เป็นส่วนหนึ่งของ OP-CODE

ตัวอย่าง

```
MOV   A,#01H
MOV   DPTR,#1234
```

2.4.5 Base-Register Plus Index-Register Indirect Address

ในโหมดนี้จะใช้ค่าในแอคคิวมูเลเตอร์ A บวกกับเบสรีจิสเตอร์ เช่น DPTR หรือ PC ประโยชน์ของโหมดนี้คือใช้ในการหาค่าที่ต้องการจากตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

037255

ตัวอย่าง

MOVC A,@A+PC

MOVC A,@A+DPTR

2.5 รีจิสเตอร์ที่ควบคุมการทำงานของพอร์ตสื่อสารอนุกรมใน MCS-51

รีจิสเตอร์ใช้งานเฉพาะที่ควบคุมการทำงานของพอร์ตสื่อสารอนุกรมใน MCS-51 มีอยู่เพียงตัวเดียวเท่านั้นคือ รีจิสเตอร์ใช้งานเฉพาะ SMOD การใช้งานพอร์ตสื่อสารอนุกรม ข้อมูลที่รับหรือส่งจะเก็บไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF

2.5.1 รีจิสเตอร์ใช้งานเฉพาะ SCON (Serial Port Control Register)

เข้าถึงข้อมูลได้ในระดับบิตวิธีการคำนวนและกำหนดค่า Baud rate ในการใช้พอร์ตสื่อสารอนุกรมโหมดต่างๆ

โหมด	SCON	SM2 VARIATION
0	10H	Single Processor
1	50H	Environment
2	90H	(SM2 = 0)
3	D0H	
0	NA	Multiprocessor
1	70H	Environment
2	B0H	(SN2 = 1)
3	F0H	

ตารางที่ 2.5 การใช้พอร์ตสื่อสารอนุกรมโหมดต่างๆ

โหมด 0 ค่า Baud rate ถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้โดยไม่จำเป็นต้องมีการกำหนดหรือใช้รีจิสเตอร์ที่เป็น Timer หรือ Counter แต่อย่างไรดังนั้น Boud rate ของพอร์ตสื่อสารอนุกรมในโหมดนี้สามารถเขียนเป็นสมการได้ง่ายๆ ดังนี้

$$\text{baud rate ในโหมด 0} = \frac{\text{ความถี่ออสซิลเลเตอร์}}{12}$$

โหมด 1 ค่า Baud rate สามารถแปรค่าได้โดยการกำหนดจากไทม์เมอร์ 1 หรือในไทม์เมอร์ 2 (มีใน 8052) ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.2 การใช้ไทม์เมอร์ 1 เป็นตัวกำหนด Baud rate เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด Baud rate จะใช้ไทม์เมอร์ 1 ในโหมด 2 (Auto-Reload) โดยมีสูตรการคำนวนหาค่า baud rate ดังนี้

$$\text{baud rate ในโหมด 1} = \frac{2^{\text{smod}} \times \text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{32 \times 12 \times [256-(TH1)]}$$

การใช้งานทั่วไป เรามักจะทราบว่าต้องการใช้ baud rate เท่าไร ดังนั้นค่าที่ต้องการหาก็คือค่าที่ต้องการโหลดไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 สมการในการคำนวนหาค่าที่ต้องการโหลดไปไว้ในรีจิสเตอร์งานเฉพาะ TH1 เมื่อทราบค่า baud rate คือ

$$TH1 = 256 - \frac{[\text{smod} \times \text{ความถี่ออสซิลเลเตอร์ที่ใช้}]}{384 \times \text{baud rate}}$$

ค่าในรีจิสเตอร์งานเฉพาะ TH1 จำเป็นต้องเป็นเลขจำนวนเต็ม การปัดเศษที่ได้จากการคำนวนทิ้งหรือปัดขึ้นอาจทำให้ไม่ได้ค่า baud rate ตามที่ต้องการ วิธีแก้ปัญหานี้คือ เปลี่ยนค่าความถี่ของคริสตอลที่ใช้ ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCOM ไม่สามารถเข้าถึงได้ในระดับบิต

2.5.3 การใช้ไทม์เมอร์ 2 เป็นตัวกำหนด baud rate ในไทม์เมอร์ 2 มีการทำงานที่ใช้สำหรับเป็นตัวกำหนด baud rate อยู่แล้วเมื่อใช้สัญญาณนาฬิกาเป็นอินพุตให้แก่ไทม์เมอร์ 2 ที่ขา T2(P1.0) ค่า baud rate สามารถคำนวนได้ดังนี้

$$\text{baud rate} = \frac{\text{อัตราการเกิด overflow ของไทม์เมอร์ 2}}{16}$$

เมื่อใช้สัญญาณนาฬิกาจากภายในชิป (กำหนดจากความถี่ออสซิลเลเตอร์) ค่า baud rate สามารถคำนวนได้ดังนี้

$$\text{baud rate} = \frac{\text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{32 \times [65535-(RCAP2H,RCAP2L)]}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หากต้องการหาค่าที่จะโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ RCAP2H,RCAP2L เมื่อทราบค่า baud rate ที่ต้องการ จะหาได้จากสมการดังต่อไปนี้

$$\text{RCAP2H,RCAP2L} = 65535 - \frac{[\text{ความถี่ออสซิลเลเตอร์ที่ใช้}]}{32 \times \text{baud rate}}$$

โหมด 2 ค่า baud rate ที่ให้เลือกได้เพียง 2 ค่า ขึ้นอยู่กับการกำหนดในบิต SMOD ในรีจิสเตอร์งานเฉพาะ PCON ดังนี้

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

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

ในโหมดนี้ไม่จำเป็นต้องใช้รีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์เพื่อกำหนดค่า baud rate แต่อย่างใด ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCON ไม่สามารถเข้าถึงได้ในระดับบิต สูตรคำนวนหา baud rate ในโหมด 2 มีดังนี้

$$\text{baud rate ในโหมด 2} = \frac{[2^{\text{smod}} \times \text{ความถี่ออสซิลเลเตอร์ที่ใช้}]}{64}$$

หากใช้คริสตอลความถี่ 12 เมกะเฮิตรซ์ baud rate สูงสุดในการทำงานโหมดนี้ คือ 375K

โหมด 3 ค่า baud rate ในโหมดนี้คำนวนและกำหนดได้จากวิธีเดียวกันกับโหมดที่ 1

2.5.4 รีจิสเตอร์ใช้งานเฉพาะ SCON เข้าถึงข้อมูลในระดับบิต

SM0	SM1	SM2	REN	TB8	RB8	T1	R1

บิต	ชื่อบิต	
SCON.7	SM0	บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ
SCON.6	SM1	บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

0 0 โหมด 0 : ทำงานเป็น shift register อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์

0 1 โหมด 1 : 8 bit UART อัตราเร็วในการรับส่งข้อมูลกำหนดเอง

1 0 โหมด 2 : 9 bit UART อัตราเร็วในการรับส่งข้อมูลเท่ากับ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์

1 1 โหมด 3 : 9 bit UART อัตราเร็วในการรับส่งข้อมูลกำหนดเอง

SCON.5 SM2 บิตเลือกการใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 2 และ 3 เพื่อใช้ในการติดต่อระหว่างซีพียูด้วยกันเอง

1 : ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมในการติดต่อสื่อสารระหว่างซีพียูด้วยกันเอง เมื่อข้อมูลบิตที่ 9 ที่ได้รับมีค่าเป็น0 (ดาต้าไบต์) บิต RI จะไม่ถูกเซต แต่หากข้อมูลบิตที่ 9 ที่ได้รับมีค่าเป็น 1 (แอดเดรสไบต์) บิต RI จะถูกเซต

0 : ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 และโหมด 3

ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 หากบิต SM2 ถูกเซตบิต RI จะไม่ถูกเซตจนกว่าบิตสิ้นสุดของข้อมูลจะถูกรับเข้ามาในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 บิตนี้ควรถูกเคลียร์ให้ เป็น 0

บิต	ชื่อบิต

SCON.4 REN บิตควบคุมการอนุญาตให้มีการรับส่งข้อมูล ดังนี้

1 : อนุญาตให้มีการรับข้อมูลจากภายนอกได้

0 : ไม่อนุญาตให้มีการรับข้อมูลจากภายนอก

SCON.3 TB8 บิตข้อมูลบิตที่ 9 จะถูกส่งออกไปในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 การเซตหรือเคลียร์กระทำด้วยคำสั่งในโปรแกรมเท่านั้น

SCON.2 RB8 บิตข้อมูลบิตที่ 9 ที่ได้รับเข้ามาจากภายนอก ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 ส่วนในการทำงานโหมด 1 ถ้าบิต SM2 = 0 บิตนี้ จะเป็นบิตสิ้นสุดของข้อมูลที่รับเข้ามา และไม่ถูกกำหนดการใช้งานในโหมด 0

SCON.1 T1 บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการส่งข้อมูล ถูกเซตโดยฮาร์ดแวร์ เมื่อข้อมูลบิตที่ 8 ถูกส่งออกไปแล้วในการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำงานโหมด 0 ส่วนในการทำงานโหมดอื่นๆ จะถูกเซตโดยฮาร์ดแวร์เมื่อเริ่มส่งบิตสิ้นสุดของข้อมูลออกไป และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

SCON.0 R1 บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการรับข้อมูลถูกเซตโดยฮาร์ดแวร์ เมื่อข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 หรือที่จุดครึ่งทางของช่วงรับบิตสิ้นสุดของข้อมูลการทำงานโหมดอื่น (มีข้อยกเว้นในกรณีใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมติดต่อระหว่างซีพียูด้วยกันแอง) และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

2.6 โครงสร้างและการทำงานของพอร์ต

โครงสร้างภายใน MCS-51 ที่เราจะศึกษาอันดับแรกคือ พอร์ตใน MCS-51 ซึ่งมีพอร์ตขนาด 8 บิตรวมทั้งสิ้น 4 พอร์ต พอร์ตแต่ละพอร์ตทำหน้าที่ติดต่อกับวงจรภายนอก โดยสามารถรับและส่งข้อมูลกับวงจรภายนอกกได้ กล่าวคือ ทั้ง 4 พอร์ตจะเป็นพอร์ตแบบ 2 ทิศทาง โครงสร้างของแต่ละพอร์ต ประกอบด้วย

-.วงจรแลตช์ซึ่งสถานะของ O/P ของวงจรแลตช์จะปรากฏที่รีจิสเตอร์ที่ใช้งานเฉพาะ P0-P3

-.วงจรเอาต์พุตไดรเวอร์

-.วงจรอินพุตบัฟเฟอร์

เอาต์พุตไดรเวอร์ของพอร์ต 0 และพอร์ต 2 กับอินพุตบัฟเฟอร์ของพอร์ต 0 จะถูกใช้ในการติดต่อกับหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป โดยมีการทำงานดังนี้

2.6.1 พอร์ต 0 ใช้ส่งค่าแอดเดรสไบต์ต่ำ (Low byte - [A0-A7]) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอกชิป) และยังถูกใช้เป็น Data bus ซึ่งจะใช้ทั้งในการรับและการส่งข้อมูลกับหน่วยความจำภายนอก โดยจะผลัดกันใช้คนละเวลาแบบ Time Multiplex

2.6.2 พอร์ต 2 ใช้ส่งค่าแอดเดรสไบต์สูง (High byte - [A8-A15]) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอกชิป) เมื่อมีการใช้แอดเดรสขนาด 16 บิตและถ้าพอร์ต 2 ไม่ถูกใช้งาน (ติดต่อกับหน่วยความจำภายนอกน้อยกว่า 265 ไบต์) คือถูกใช้ไม่ครบ 16 บิต (ใช้หน่วยความจำภายนอกไม่ครบ 64 กิโลไบต์) พอร์ต 2 บิตที่ไม่ใช้งานจะส่งค่าภายในรีจิสเตอร์ใช้งานเฉพาะ P2 ออกมาที่แต่ละขา (ซึ่งก็คือค่าที่แลตช์อยู่ภายใน)

2,6.3 พอร์ต 3 ทั้งหมดรวมถึงพอร์ต 1 ของ 8052 อีก 2 ขา จะไม่เพียงแต่ใช้เป็นพอร์ตอินพุตและเอาต์พุตเท่านั้น เพราะแต่ละบิตของพอร์ต 3 และพอร์ต 1 บางบิตยังมีหน้าที่พิเศษอีกเช่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พอร์ต	หน้าที่
P1.0(T2)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 2
P1.1(T2EX)	ใช้เป็นสัญญาณควบคุมการทำงานของไทม์เมอร์ 2
P3.0(RXD)	ใช้เป็นอินพุตของพอร์ตสื่อสารอนุกรม
P3.1(TXD)	ใช้เป็นเอาท์พุตของพอร์ตสื่อสารอนุกรม
P3.2(INT0)	ใช้เป็นอินพุตของอินเตอร์รัปต์ภายนอกชนิด 0
P3.3(INT1)	ใช้เป็นอินพุตของอินเตอร์รัปต์ภายนอกชนิด 1
P3.4(T0)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 0
P3.5(T1)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 1
P3.6(WR)	ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอก
P3.7(RD)	ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยข้อมูลภายนอกเข้ามา

ตารางที่ 2.6 หน้าที่ของพอร์ตต่างๆ

หน้าที่พิเศษเหล่านี้จะใช้ได้ก็ต่อเมื่อค่าในบิตแลตช์ของพอร์ตในบิตนั้นๆ มีค่าเป็น 1 มิฉะนั้น ที่ขาของพอร์ตภายนอกชิปจะมีค่าเป็น 0

2.6.4 พอร์ตการสื่อสารข้อมูลแบบอนุกรมใน MCS-51

MCS-51 มีพอร์ตในการสื่อสารแบบอนุกรมที่สามารถรับและส่งข้อมูลแบบอนุกรมได้ โดยผู้ใช้ไม่จำเป็นต้องต่อชิปที่ทำหน้าที่รับส่งข้อมูลโดยเฉพาะเพิ่มเติมอย่างใดเลย การนำ MCS-51 ไปประยุกต์ใช้งานที่ต้องมีการติดต่อสื่อสารข้อมูลแบบอนุกรมกับวงจรภายนอกอื่นๆ จึงทำได้สะดวกและมีความคล่องตัวสูงมาก

พอร์ตสื่อสารข้อมูลแบบอนุกรมที่มีใน MCS-51 สามารถทำงานในแบบ full duplex หมายความว่า MCS-51สามารถรับและส่งข้อมูลได้พร้อมๆ กัน โดยการรับข้อมูลจะมี buffer ข้อมูลให้ด้วย จึงทำให้ MCS-51 สามารถกำหนดการรับข้อมูลไบต์ที่ 2 ที่ถูกส่งตามเข้ามาก่อนที่ไบต์แรกที่ได้รับเข้ามาจะถูกอ่านจากรีจิสเตอร์ใช้งานเฉพาะที่ใช้สำหรับรับข้อมูล (receive register) เพื่อนำไปเก็บในหน่วยความจำต่อไป

พอร์ตสื่อสารแบบอนุกรมใน MCS-51 ประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิตจำนวน 2 ตัวแต่ละตัวมีชื่อเรียกตามหน้าที่ดังนี้

- รีจิสเตอร์สำหรับรับข้อมูลที่ถูกส่งเข้ามาจากภายนอก
- รีจิสเตอร์สำหรับส่งข้อมูล ใช้ส่งข้อมูลจาก MCS-51 ไปยังภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันในรีจิสเตอร์ใช้งานเฉพาะคือ ตรงกับตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะ SBUF (ตำแหน่ง 99H) ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ การเข้าถึงข้อมูลในรีจิสเตอร์แต่ละตัว MCS-51 จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ตัวใด โดยตรวจสอบจากรหัสคำสั่ง ทั้งนี้เพราะในการเขียนข้อมูลไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF หมายถึง การโหลดข้อมูลไปไว้ในรีจิสเตอร์สำหรับส่งข้อมูลเพื่อส่งข้อมูลออกไปภายนอก ส่วนการอ่านข้อมูลจากรีจิสเตอร์ใช้งานเฉพาะ SBUF จะหมายถึงการนำค่าที่ได้รับเข้ามาได้จากภายนอกที่เก็บไว้ในรีจิสเตอร์สำหรับรับข้อมูลมาใช้งาน

การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51 มีความสะดวกและคล่องตัวสูง ทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยสามารถกำหนดได้จากค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON การใช้งานที่แตกต่างถึง 4 ประเภทนี้มีจุดประสงค์เพื่อความคล่องตัวในการรับส่งข้อมูลแบบอนุกรมแต่ละประเภทดังนี้

โหมด 0 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 ขา RDX จะใช้สำหรับรับและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ shift clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิต โดยเริ่มนับจากบิตต่ำสุดก่อน (LSB First) อัตราการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดไว้ที่ 1/12 ของความถี่ออสซิเลเตอร์ที่ใช้ การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นข้อมูล (start bit) และบิตสิ้นสุดของข้อมูล (stop bit) เพราะจังหวะการรับและส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock แล้ว

โหมด 1 การทำงานแบบที่ 2 หรือทำงานโหมด 1 นี้ มีการรับและส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งออกไปภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิต ประกอบด้วยบิตเริ่มต้นข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดของข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าในบิตสิ้นสุดของข้อมูลที่รับได้จะอยู่ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้ ดังนั้นจะได้กล่าวในรายละเอียดต่อไป

โหมด 2 การทำงานแบบที่ 3 หรือการทำงานโหมด 2 จะมีการรับและส่งข้อมูลครั้งละ 11 บิต ข้อมูลจะถูกส่งออกภายนอกผ่านทางขา TXD และรับเข้ามาผ่านทางขา RXD ข้อมูลที่รับและส่งทั้ง 11 บิตประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับหรือส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 (ต่อจากบิตข้อมูลบิตสุดท้าย) ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็นศูนย์หรือ หนึ่งได้ (programmable 9th data bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (มีค่าเป็น 1 เสมอ) ดังนั้นจำนวนบิตที่รับส่งทั้งหมด 11 บิต ในการทำงานส่งข้อมูล บิตที่ 9 จะได้จากค่าในบิต TB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON บิตนี้สามารถกำหนดให้มีค่าเป็น 0 หรือ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ได้ส่วนใหญ่ในการใช้งานจริงมักใช้บิตนี้สำหรับตรวจสอบความถูกต้อง ของข้อมูลที่รับหรือส่ง (parity bit) โดยจะนำบิต P (parity) ในรีจิสเตอร์ PSW ไปไว้ในบิต TB8 ส่วนในขณะรับข้อมูลบิตที่ 9 จะปรากฏอยู่ในบิต RB8 ของรีจิสเตอร์ SCON โดยไม่สนใจบิตสิ้นสุดของข้อมูล ค่าอัตราเร็วในการรับหรือส่งข้อมูลโหมดนี้ถูกกำหนดไว้ที่ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

โหมด 3 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมแบบสุดท้าย คือการทำงานในโหมด 3 ในการทำงานโหมดนี้ข้อมูลจำนวน 11 บิตถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (เป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับหรือส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดค่าได้เหมือนในโหมด 2 (programmable 9th bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (มีค่าเป็น 1 เสมอ) ดังนั้นจะเห็นว่ารูปแบบในการรับส่งข้อมูลในโหมด 3 จะเหมือนกับโหมด 2 ทุกอย่างแต่ในโหมดนี้จะสามารถกำหนดค่าอัตราเร็วในการรับหรือส่งข้อมูลได้ตามความต้องการของผู้ใช้

การทำงานของพอร์ตสื่อสารของข้อมูลแบบอนุกรมทั้ง 4 แบบ โหมดที่กล่าวมานี้ การส่งข้อมูลจะเริ่มส่งทันทีเมื่อมีคำสั่งใดๆ ที่ใช้รีจิสเตอร์ใช้งานเฉพาะ SBUF เป็นรีจิสเตอร์ปลายทาง เช่น

```
MOVE SBUF,A
```

ส่วนในการรับส่งข้อมูลจะเริ่มต้นโดยมีเงื่อนไขดังนี้

- ในโหมด 0 เริ่มเมื่อค่าในบิต RI = 0 และบิต REN = 1
- ในโหมดอื่นๆ การรับข้อมูลเริ่มเมื่อ MCS-51 ได้รับบิตเริ่มต้นของข้อมูลเข้ามา โดยที่บิต REN ในขณะนั้นต้องมีค่าเป็น 1

2.7 การสื่อสารระหว่างไมโครคอนโทรลเลอร์หลายตัว

การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 2 และ 3 MCS-51 จะมีรูปแบบการใช้งานพิเศษนอกเหนือจากการรับส่งข้อมูลธรรมดาที่กล่าวไปแล้ว นั่นคือการใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมในการติดต่อสื่อสารระหว่าง ซีพียูด้วยกันเอง ดังมีรายละเอียดดังต่อไปนี้

ข้อมูลที่รับส่งในการใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 หรือโหมด 3 เพื่อติดต่อระหว่างข้อมูลด้วยกันจะมีทั้งสิ้น 11 บิต การกำหนดให้พอร์ตสื่อสารข้อมูลแบบอนุกรมใช้ติดต่อสื่อสารระหว่างซีพียูด้วยกันเองสามารถกำหนดได้จากบิต SM2 ในรีจิสเตอร์ใช้งานเฉพาะ SCON บิตที่ 9 ที่รับเข้ามาในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมทั้งสองโหมดจะถูกนำไปใช้ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON ตามด้วยบิตสิ้นสุดของข้อมูลเหมือนการรับส่งข้อมูลทั่วไปที่กล่าวมาแล้ว ถ้าบิต SM2 ถูกเซต และบิตสิ้นสุดของข้อมูลถูกรับเข้ามาแล้ว หากบิต RB8 มีค่าเป็น 1 จะส่งผลไปกระตุ้นให้วงจรส่วนควบคุมการอินเตอร์รัปต์ของพอร์ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สื่อสารข้อมูลแบบอนุกรม เริ่มทำงานเพื่ออินเตอร์รัปต์ซีพียูต่อไป หากบิต RB8 มีค่าเป็น 0 วงจรส่วนควบคุมการอินเตอร์รัปต์ของพอร์ตสื่อสารข้อมูลอนุกรมจะไม่ทำงานแต่อย่างไร รายละเอียดของพอร์ตสื่อสารข้อมูลอนุกรมที่ใช้ในการติดต่อสื่อสารระหว่างซีพียูด้วยกันเองมีดังนี้

เมื่อไมโครคอนโทรลเลอร์ตัวแม่หรือตัวหลักต้องการส่งข้อมูลจำนวนหนึ่งไปยังไมโครคอนโทลเลอร์ตัวลูกตัวใดตัวหนึ่ง จากที่มีอยู่หลายตัวในระบบ ขั้นแรกไมโครคอนโทรลเลอร์ตัวแม่จะต้องส่งข้อมูลขนาด 1 ไบต์ ที่มีเรียกเฉพาะว่า address byte ค่าแอดเดรสไบต์นี้จะเป็นค่าที่ต้องระบุหมายเลขประจำ หรือตำแหน่งของไมโครคอนโทรลเลอร์ตัวลูกในระบบที่เป็นเป้าหมายของตัวแม่ที่ติดต่อด้วย ค่าแอดเดรสไบต์ที่ไมโครคอนโทรเลอร์ตัวแม่ส่งไปมีข้อแตกต่างข้อมูลทั่วไปที่รับส่งกันจริงๆ ระหว่างไมโครคอนโทรลเลอร์ที่มีชื่อเรียกเฉพาะว่า data byte ดังนี้

- แอดเดรสไบต์ : บิตที่ 9 จะเป็น 1
- ดาต้าไบต์ : บิตที่ 9 จะเป็น 0

หากในไมโครคอนโทรลเลอร์ตัวลูกมีการเซตบิต SM2 = 1 แล้ว ถ้าข้อมูลที่เข้ามาเป็นดาต้าไบต์จะไม่สามารถอินเตอร์รัปต์ซีพียูได้แต่หากข้อมูลที่ได้รับเป็นแอดเดรสไมโครคอนโทรลเลอร์ตัวลูกทุกตัวจะถูกอินเตอร์รัปต์ การทำเช่นนี้ก็เพื่อที่ไมโครคอนโทรลเลอร์ตัวลูกทุกตัวที่เชื่อมต่อกับตัวแม่จะสามารถตรวจสอบแอดเดรสไบต์ที่ได้รับเข้ามาว่ามีค่าตรงกับหมายเลขตำแหน่งของตัวเองหรือไม่ หากไมโครคอนโทรลเลอร์ตัวลูกตัวใดมีค่าตัวเลขของตัวเองตรงกับแอดเดรสไบต์ที่ได้รับเข้ามาก็จะเคลียร์บิต SM2 และเตรียมรับดาต้าไบต์ซึ่งจะตามเข้ามาภายหลังที่ได้รับแอดเดรสไบต์เรียบร้อยแล้ว นั่นคือหากไมโครคอนโทรลเลอร์ตัวลูกตัวใดตรวจสอบค่าแอดเดรสไบต์ที่ได้รับเข้ามาไม่ตรงกับค่าตำแหน่งของตัวเอง มันจะไม่สนใจดาต้าไบต์ที่ส่งเข้ามาตามหลังแอดเดรสไบต์เลยแต่ถ้าไมโครคอนโทรลเลอร์ตัวลูกจะถูกอินเตอร์รัปต์เพื่อตรวจสอบค่าแอดเดรสไบต์ดูว่ามีค่าตรงกับตำแหน่งตัวเองหรือไม่ทุกครั้ง

บิต SM2 จะไม่มีผลในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 แต่การทำงานในโหมด 1 บิต SM2 สามารถใช้ตรวจสอบบิตสิ้นสุดของข้อมูล (validity of the stop bit) โดยในการรับข้อมูลของโหมด 1 ถ้าบิต SM2 = 1 จะเป็นการตรวจสอบบิตสิ้นสุดของข้อมูลโดยหากบิตสิ้นสุดของข้อมูลมีค่าไม่เป็น 1 การอินเตอร์รัปต์ของพอร์ตสื่อสารข้อมลแบบอนุกรมจะไม่เกิดขึ้น

2.7.1 อัตราเร็วในการรับส่งข้อมูล

baud rate หมายถึงอัตราเร็วในการรับหรือส่งข้อมูล โดย MCS-51 ค่าความเร็วในการรับและส่งข้อมูลจะมีค่าเท่าใด ก็ขึ้นอยู่กับการทำงานในแต่ละโหมดของพอร์ตสื่อสารข้อมูลแบบอนุกรมดังนี้

baud rate โหมด 0 = <u>ความถี่ออสซิลเลเตอร์ที่ใช้</u>

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หากใช้คริสตอลความถี่ 12 MHz ค่า baud rate ของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะมีค่าสูงถึง 1 Mhz ในโหมด 2 ค่า baud rate ขึ้นอยู่กับค่าของบิต SMOD ที่อยู่ในรีจิสเตอร์งานเฉพาะ PCON โดย

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

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

หลังจากการรีเซต MCS-51 ค่าในบิต SMOD จะเป็น 0 เสมอ และเราสามารถเขียนสูตรสำหรับค่า baud rate ได้ดังสมการดังนี้

$$\text{baud rate โหมด 2} = \frac{[2^{smod} \times (\text{ความถี่ออสซิลเลเตอร์ที่ใช้})]}{64}$$

หากใช้คริสตอลความถี่ 12 MHz ค่า baud rate สูงสุดในการทำงานโหมดนี้คือ 375K

- baud rate ในโหมด 1 และ 3 จะถูกกำหนดโดยอัตราการเกิด over flow ของไทม์เมอร์ 1 แต่ถ้าเป็น 8052 ซึ่งมีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ที่สามารถนำมากำหนด baud rate สำหรับการรับข้อมูล ส่วนอีกตัวหนึ่งกำหนด baud rate สำหรับการส่งข้อมูล ทำให้การรับและการส่งข้อมูลมีค่า baud rate ที่ต่างกันได้ รายละเอียดของการใช้ไทม์เมอร์ 1 หรือไทม์เมอร์ 2 เป็นตัวกำหนดค่า baud rate ดังนี้

2.7.2 เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate เมื่อไทม์เมอร์ 1 ถูกใช้เป็นตัวกำหนด baud rate สำหรับการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 1 และโหมด 3 ค่าของ baud rate ที่ได้จะถูกกำหนดด้วยอัตราการเกิด over flow ของไทม์เมอร์ 1 และขึ้นอยู่กับบิต SMOD ในรีจิสเตอร์ PCON ซึ่งเขียนเป็นสมการที่คำนวนหาค่า baud rate ดังนี้

$$\text{baud rate โหมด 1,3} = \frac{[2^{smod} \times (\text{อัตราการเกิด over flow ของไทม์เมอร์ 1})]}{32}$$

การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมที่พบบ่อยที่สุดนั้นไทม์เมอร์ 1 จะถูกกำหนดให้ทำงานเป็นไทม์เมอร์ในโหมด 2 (auto-reload) ในกรณีนี้ baud rate จะถูกกำหนดโดยสมการดังนี้

$$\text{baud rate โหมด 1,3} = \frac{[2^{smod} \times (\text{ความถี่ออสซิลเลเตอร์ที่ใช้})]}{32 \times 12 \times [256-(TH1)]}$$

ดังนั้นค่าที่ต้องโหลดไปยังรีจิสเตอร์ TH1 เพื่อให้ได้ค่า baud rate จะสามารถคำนวนได้ดังนี้

$$TH1 = \frac{[2^{smod} \times (\text{ความถี่ออสซิลเลเตอร์ที่ใช้})]}{384 \times \text{baud rate}}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.3 การใช้ไทม์เมอร์โหมด 2 ในการกำหนด baud rate สำหรับ MCS-51 เบอร์ที่มีไทม์เมอร์ 2 เพิ่มให้แก่ผู้ใช้อีก 1 ตัว เช่นเบอร์ 8052 ผู้สามารถนำไทม์เมอร์ 2 ที่มีเพิ่มขึ้นนี้มาเป็นตัวกำหนด baud rate โดยการเซตบิต TCLK หรือบิต RCLK ในรีจิสเตอร์ใช้งานเฉพาะ T2CON การใช้ไทม์เมอร์ 2 เป็นตัวกำหนด baud rate จะมีการทำงานเหมือนโหมด Auto-Reload ตรงที่เกิดการ over flow ในรีจิสเตอร์ TH2 ทำให้รีจิสเตอร์ที่ประกบขึ้นเป็นไทม์เมอร์ 2 ถูกโหลดด้วยค่า 16 บิตจากรีจิสเตอร์ใช้งานเฉพาะ RCAP2L และ RCAP2H ซึ่งต้องได้รับการตั้งค่าไว้ล่วงหน้าด้วย soft ware เรียบร้อยแล้ว เนื่องจากไทม์เมอร์ 2 สามารถถูกกำหนดการใช้งานเป็นไทม์เมอร์หรือเคาท์เตอร์อย่างใดอย่างหนึ่ง ในการใช้งานทั่วไปที่นำไทม์เมอร์ 2 มาเป็นตัวกำหนด baud rate มักจะกำหนดให้ทำงานเป็นไทม์เมอร์ในโหมดอื่นเล็กน้อย กล่าวคือปกติเมื่อใช้ไทม์เมอร์ 2 เป็นไทม์เมอร์ในโหมดอื่นมันจะถูกเพิ่มค่าทุกๆ แมชชีนไซเกิล แต่เมื่อใช้งานไทม์เมอร์ในโหมดที่ใช้สำหรับกำหนดค่า baud rate สำหรับพอร์ตสื่อสารข้อมูลแบบอนุกรม ไทม์เมอร์ 2 จะถูกเพิ่มค่าขึ้นทุกๆ สเตทในแต่ละแมชชีนไซเกิล เท่ากับความถี่ 1/2 ของความที่ออสซิลเลเตอร์ ดังนั้นสมการสำหรับหาค่า baud rate ของพอร์ตสื่อสารอนุกรมเมื่อใช้ไทม์เมอร์ 2 คือ

$$\text{baud rate โหมด 1,3} = \frac{\text{(ความถี่ออสซิลเลเตอร์ที่ใช้)}}{[32 \times (65536-(\text{RCAP2H,RCAP2L}))]}$$

RCAP2H,RCAP2L ในสมการคือค่าที่ต้องโหลดไว้ในรีจิสเตอร์ใช้งานเฉพาะ RCAP2H และRCAP2L ตามลำดับค่านี้ถูกกำหนดให้เป็นจำนวนเต็มที่ไม่คิดเครื่องหมายขนาด 16 บิต

ค่าที่โหลดไปไว้ในรีจิสเตอร์ RCAP2H และ RCAP2L เพื่อให้ได้ค่า baud rate ตามที่ต้องการสามารถคำนวนได้จากสูตรต่อไปนี้

$$\text{(RCAP2H,RCAP2L)} = 65535-\frac{\text{[ความถี่ออสซิลเลเตอร์ที่ใช้]}}{32 \times \text{baud rate}}$$

สิ่งที่ควรระลึกไว้เสมอในการใช้งานไทม์เมอร์ 2 เป็นตัวกำหนด baud rate นั่นคือไทม์เมอร์ 2 กำลังเป็นไทม์เมอร์ (บิต RCLK + TCLK = 1 และบิต TH2 = 1) เพื่อกำหนดเป็น baud- rate ผู้ใช้ไม่ควรพยายามอ่านหรือเขียนข้อมูลใดๆ ลงไปในรีจิสเตอร์ของไทม์เมอร์ 2 ทั้งนี้เพราะภายใต้สภาพการทำงานเช่นนี้รีจิสเตอร์ของไทม์เมอร์ 2 จะถูกเพิ่มค่าทุกๆ สเตทในแต่ละแมชชีนไซเกิลทำให้ข้อมูลที่ได้จากการอ่านเขียนไม่เที่ยงตรงแต่เราอาจจะอ่านหรือเขียนข้อมูลจากรีจิสเตอร์ RCAP ได้ ถึงแม้การเขียนข้อมูลยังรีจิสเตอร์ทั้งสอง นี้อาจทำให้เกิดการผิดพลาดได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หากข้อมูลที่เขียนลงไปตรงกับช่วงเวลาการโหลดค่าใหม่จากรีจิสเตอร์ไปยังรีจิสเตอร์ของไทม์เมอร์ 2 จากนั้นจึงสามารถกระทำการใดๆ กับรีจิสเตอร์ทั้งสองตัวนี้ได้ตามต้องการ

2.7.4 รีจิสเตอร์ที่ใช้งานเฉพาะ PSW (Program Status Word) เข้าถึงข้อมูลระดับบิตดังรูป

CY	AC	F0	RS1	RS0	0V	-	P

บิต	ชื่อบิต	
PSW.7	CY	carry flag
PSW.6	AC	auxillary carry flag
PSW.5	F0	flag 0 เป็นบิตบอกสถานะที่ผู้ใช้สามารถกำหนดการใช้งานเอง
PSW.4	RS1	ใช้เลือกกลุ่มรีจิสเตอร์ R0-R7
PSW.3	RS0	ใช้เลือกกลุ่มรีจิสเตอร์ R0-R7 0 0 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 0 0 1 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 1 1 0 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 2 1 1 เลือกรีจิสเตอร์ใช้งานทั่วไปกลุ่มที่ 3
PSW.2	0V	บิตแสดงการเกิด overflow
PSW.1	-	บิตที่ผู้ใช้กำหนดการใช้งานเองได้
PSW.0	P	parity flag ถูกเซตหรือเคลียร์โดยวงจรภายใน MCS-51 ในแต่ละไซเกิลของแต่ละคำสั่ง ใช้เป็นตัวบอกให้ทราบว่ารีจิสเตอร์ ACC มีข้อมูลที่เป็น 1 เป็นคู่หรือคี่

2.7.5 รีจิสเตอร์ใช้งานเฉพาะ PCON (Power Control Register) ไม่สามารถเข้าถึงข้อมูลในระดับบิตได้ ดังแผนภาพที่แสดงในรูป

SMOD	-	-	-	GF1	GF0	PD	IDL

บิต	ชื่อบิต	
PCON.7	SMOD	เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate หากบิตนี้มีค่าเป็น 1 ในการใช้งานพอร์ตสื่อสารอนุกรมโหมด 1,2 และ 3 ค่า baud rate จะเพิ่มขึ้นเป็นสองเท่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

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

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

PCON.3 GF1 บิตที่ผู้ใช้กำหนดการใช้งานเองได้

PCON.2 GF0 บิตที่ผู้ใช้กำหนดการใช้งานเองได้

PCON.1 PD บิตควบคุมการใช้พลังงานแบบ Power Down Mode

1 : MCS-51 จะอยู่ในสภาวะ Power Down Mode (หยุดการทำงานจนกว่าจะ reset) ใช้เฉพาะ MCS-51 ที่ผลิตโดยเทคโนโลยี CHMOS เท่านั้น

0 : MCS-51 จะอยู่ในสภาวะทำงานปกติ

PCON.0 IDL บิตควบคุมการใช้พลังงานแบบ idle mode

1 : MCS-51 จะอยู่ในสภาวะ idle mode คือ หยุดการทำงานชั่วคราวจนกว่าจะมีสัญญาณอินเตอร์รัปต์ในระหว่างอินเตอร์รัปต์ MCS-51 จะใช้พลังงานน้อยมาก

0 : MCS-51 จะอยู่ในสภาวะทำงานปกติ (บิต PD จะต้องเป็น 1 ด้วย)

2.7.6 รีจิสเตอร์ใช้งานเฉพาะ EI (Interrupt Enable-Register) เข้าถึงข้อมูลในระดับบิตได้ ดังแผนภาพที่แสดงในรูป

EA	-	ET2	ES	ET1	EX1	ET0	EX0

บิต ชื่อบิต

IE.7 EA ใช้ควบคุมการตอบสนองสัญญาณอินเตอร์รัปต์ทั้งหมด

0 : MCS-51 จะไม่ตอบสนองต่อสัญญาณอินเตอร์รัปต์ใดๆ ทั้งสิ้น

1 : อินเตอร์รัปต์แต่ละชนิดถูกควบคุมการตอบสนองจากบิตในรีจิสเตอร์

IE.6 - ไม่ถูกกำหนดไว้ใช้งาน

IE.5 ET2 ควบคุมการตอบสนองสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 2 เมื่อเกิด overflow หรือเกิดอินเตอร์รัปต์ของไทม์เมอร์ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IE.4 ES ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม

IE.3 ET1 ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 1 เมื่อเกิด overflow

IE.2 EX1 ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1

IE.1 ET0 ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0 เมื่อเกิด overflow

IE.0 EX0 ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 0

2.7.7 รีจิสเตอร์ใช้งานเฉพาะ IP (Interrupt Priority Register) เข้าถึงข้อมูลในระดับบิตได้ ดังแผนภาพที่แสดงในรูป

-	-	PT2	PS	PT1	PX1	PT0	PX0

บิต ชื่อบิต

IP.7 - ไม่ถูกกำหนดไว้ใช้งาน

IP.6 - ไม่ถูกกำหนดไว้ใช้งาน

IP.5 PT2 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 2

IP.4 PS กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม

IP.3 PT1 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 1

IP.2 PX1 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1

IP.1 PT0 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0

IP.0 PX0 กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 การประยุกต์ใช้งาน MCS-51 ร่วมกับ RS-485 และ ASN-Portocol

MCS-51 เป็นไมโครคอนโทรลเลอร์ที่ถูกเลือกมาใช้งานเป็นหน่วยประมวลผลกลางของเทอร์มินอลโมดูลเซริฟเวอร์โมดูล และ เซ็นเตอร์โมดูลสำหรับระบบดูแลความปลอดภัยซึ่งแต่ละโมดูลสามารถรับส่งข้อมูลอนุกรมผ่าน RS-485 และ ASN-Portocol ได้ โดยการเชื่อมต่อ MCS-51 เข้ากับคู่ตัวรับส่ง SN75176B

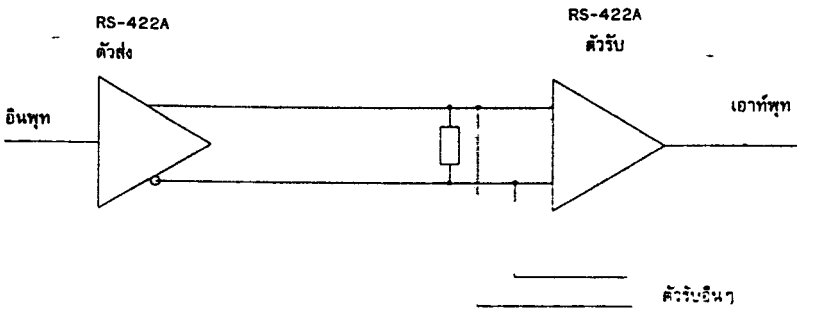
2.8.1 RS-485 และ ASN-Portocol

การสื่อสารระหว่างคอมพิวเตอร์ด้วยกัน หรือ คอมพิวเตอร์กับอุปกรณ์รอบข้าง ด้วยตัวรับและตัวส่งแบบไม่สมดุลย์ เช่น RS-232C และ RS-423A จะประสบปัญหาการเกิดข้อผิดพลาดได้ง่ายเมื่อมีการส่งข้อมูลด้วยอัตราการรับส่งข้อมูลที่สูง การใช้คู่สายสัญญานที่ยาวมากๆ ในการรับส่งข้อมูลระยะไกล รวมทั้งการวางคู่สายในบริเวณที่มีสัญญานรบกวนสูง ดังนั้น The Electronics-Industries Association (EIA) จึงได้พัฒนาการสื่อสารข้อมูลอนุกรมแบบสมดุลย์ (ตัวรับและตัวส่งไม่ต้องใช้สายกราวน์ร่วมกัน) ขึ้นเพื่อแก้ไขปัญหาดังกล่าวโดยได้พัฒนาขึ้นมาสองมาตรฐานคือ RS-422A และ RS-485 ซึ่งมีการประยุกต์ใช้งานที่แตกต่างกัน ขึ้นอยู่กับความเหมาะสมของระบบงานสำหรับการเปรียบเทียบมาตรฐานการรับส่งข้อมูลที่มีตัวรับ-ตัวส่งแบบไม่สมดุลย์ (RS-232C ,RS-423A)กับ ตัวรับ-ตัวส่งแบบสมดุลย์ (RS-485,RS-422A) ดังแสดงในตารางที่ 2.7

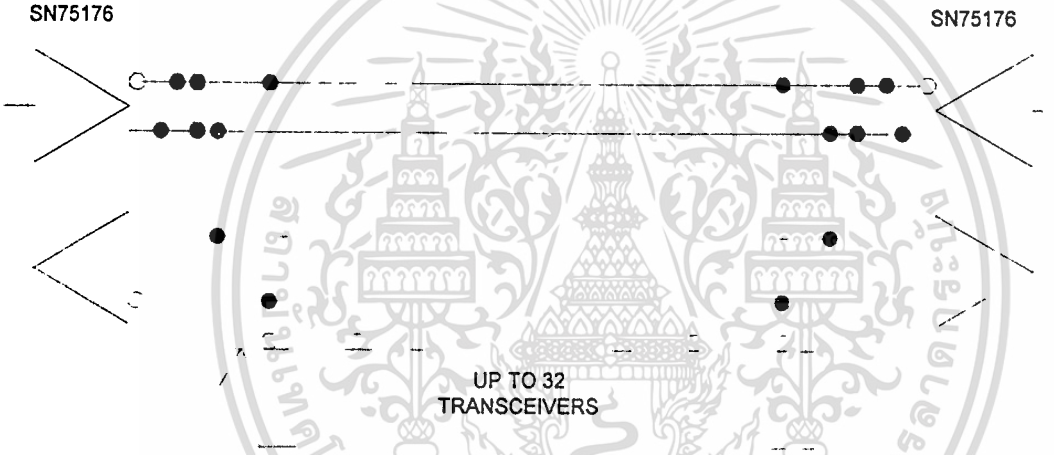
2.8.2 มาตรฐานการสื่อสารข้อมูล RS-485

RS-485 เป็นมาตรฐานการสื่อสารข้อมูลแบบสมดุลย์ที่ได้พัฒนามาจากมาตรฐานRS-422A เพื่อให้มีจำนวนตัวรับตัวส่งมากขึ้น และสามารถใช้คู่สายสัญญานรับส่งร่วมกันได้ (multipoint multiple drivers and receivers) ซึ่งในกรณีของ RS-422A บน 1 คู่สายสัญญานรับส่งหนึ่งคู่จะมีตัวส่งได้หนึ่งชุด และมีตัวรับได้ไม่เกิน 10 ชุด แต่ถ้าใช้ RS-485 สามารถใช้ตัวรับได้ 32 ชุดและมีตัวส่งได้ 32 ชุดเช่นกัน โดยทั่วไป RS-485 จะมีคุณลักษณะทางไฟฟ้าเหมือนกับ RS-422A และไม่จำกัดโปรโตคอลที่จะนำมาประยุกต์ใช้งาน นอกจากนี้ตัวรับส่งที่จะนำมาสร้างเป็นวงจรก็มีราคาไม่แพง จึงทำให้ RS-485 มีการนำมาประยุกต์ใช้งานในระบบการสื่อสารอนุกรมแบบโครงข่ายกันอย่างมาก ตัวอย่างเช่น ระบบบ้านอัตโนมัติ (Home Automation) , ระบบรักษาความปลอดภัย (Security System) ,ระบบควบคุมภายในโรงงานอุตสาหกรรม (Industrial Control System) สำหรับลักษณะการเชื่อมต่อของ RS-422A และ RS-485 ดังแสดงในรูปที่ 2.7 และรูปที่ 2.8 ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 แสดงการเชื่อมต่อของ RS-422A



รูปที่ 2.8 แสดงการเชื่อมต่อของ RS-485

จากรูปที่ 2.7 เป็นการเชื่อมต่อของ RS-422A โดยใช้หนึ่งคู่สายสัญญานรับส่งจะมีตัวส่งเพียงชุดเดียว ส่วนตัวรับมีได้ไม่เกิน 10 ชุด พร้อมกับต้องมีตัวต้านทาน 100 โอห์ม ต่อคร่อมระหว่างคู่สายสัญญาน จากลักษณะดังกล่าวจะเห็นได้ว่าเป็นการสื่อสารข้อมูลแบบทิศทางเดียวทำให้เกิดปัญหาในการสื่อสารข้อมูลจากตัวรับกับมาหาตัวส่ง เพราะจะต้องทำการเพิ่มคู่สายสัญญาณอีกหนึ่งคู่สาย เพื่อทำให้สามารถส่งสัญญานจากตัวรับไปหาตัวส่งได้ แต่ทำให้ค่าใช้จ่ายในการเพิ่มคู่สายสูงขึ้น ดังนั้นถ้าเราใช้ RS-485 จะทำให้สามารถรับ และส่งข้อมูลได้ภายในคู่สายเดียวกันทำให้ประหยัดมากขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.7 การเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA

พารามิเตอร์	RS-232C	RS-423A	RS-422A	RS-485
โหมดการทำงาน	Single-ended	Single-ended	Differential	Differential
จำนวนของตัวรับและตัวส่งที่ยอมรับได้	1 ตัวส่ง 1 ตัวรับ	1 ตัวส่ง 1 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	32 ตัวส่ง 32 ตัวรับ
ความยาวของคู่สายสัญญานรับส่งข้อมูล	50 ฟุต	4000 ฟุต	4000 ฟุต	4000 ฟุต
อัตราการส่งข้อมูลสูงสุด(บิทต่อวินาที)	20 k	100k	10M	10M
แรงดันไฟฟ้าโหมดร่วมสูงสุด	± 2.5 V	± 6 V	± 6 V - 2.5 V	12 V - 7 V
Driver output	± 5 V ต่ำสุด ±15 V สูงสุด	±3.6V ต่ำสุด ± 6 V สูงสุด	± 2 V ต่ำสุด	± 1.5Vต่ำสุด
Driver load Ω	3k ถึง 7k	450	100 ต่ำสุด	60 ต่ำสุด
Driver slew rate	30 V/uS	-	NA	NA
กระแสลิมิตเมื่อเอาท์พุทลัดวงจร	500 mA ลัดวงจรกับ Vcc or GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND 250 mA ลัดวงจรกับ 8 V or 12 V
ค่าความต้านทานเอาท์พุทของตัวส่ง Ω	NA-power on 300-power off	NA-power on 60k-power off	NA-power on 60k-power off	120k NA-power on
ค่าความต้านทานเอาท์พุทของตัวรับ Ω	3 k ถึง 7 k	4 k	4 k	12 k
ความไวของตัวรับ	± 3 V	± 200 mV	± 200 mV	± 200 mV

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8.3 คุณลักษณะเฉพาะของ RS-485 ที่ดีกว่า RS-422A

2.8.3.1 คุณลักษณะเฉพาะของตัวส่ง

- ตัวส่งหนึ่งตัวสามารถขับโหลดได้ถึง 32 ชุด (โหลดหนึ่งตัวประกอบด้วย 1 ตัวรับ และ 1 ตัวส่ง
- เอาท์พุทของตัวส่งในสภาวะออฟ มีกระแสรั่วไหลไม่เกิน 100 uA ในช่วงแรงดันไฟฟ้าโหมดร่วม ระหว่างค่า -7 V ถึง 12 V
- เอาท์พุทของตัวส่งให้แรงดันเอาท์พุท 1.5 V ถึง 5 V ในช่วงแรงดันไฟฟ้าโหมดร่วมระหว่างค่า -7 V ถึง 12 V
- ตัวส่งมีวงจรป้องกันตัวเองที่ส่วนของเอาท์พุท ในกรณีที่ตัวส่งหลายๆ ตัวทำการส่งข้อมูลออกมาพร้อมๆ กัน

2.8.3.2 คุณลักษณะเฉพาะของตัวรับ

- ค่าความต้านทานที่อินพุทมีค่าสูง โดยมีค่าไม่น้อยกว่า 12 กิโลโอห์ม
- ตัวรับมีค่าแรงดันไฟฟ้าโหมดร่วม ระหว่างค่า -7 V ถึง 12 V
- ตัวรับสามารถตอบสนองต่อสัญญานที่แตกต่างจากสัญญาณโหมดร่วมได้ $\pm$200mV(น้อยที่สุด)

2.8.3.3 คุณลักษณะของคู่สายสัญญาน

- คู่สายสัญญานรับส่งควรพันเป็นเกลียวเพื่อหอนสัญญาณรบกวน

2.8.3.4 คุณสมบัติของคู่ตัวรับ-ส่ง (Transceivers) RS-485

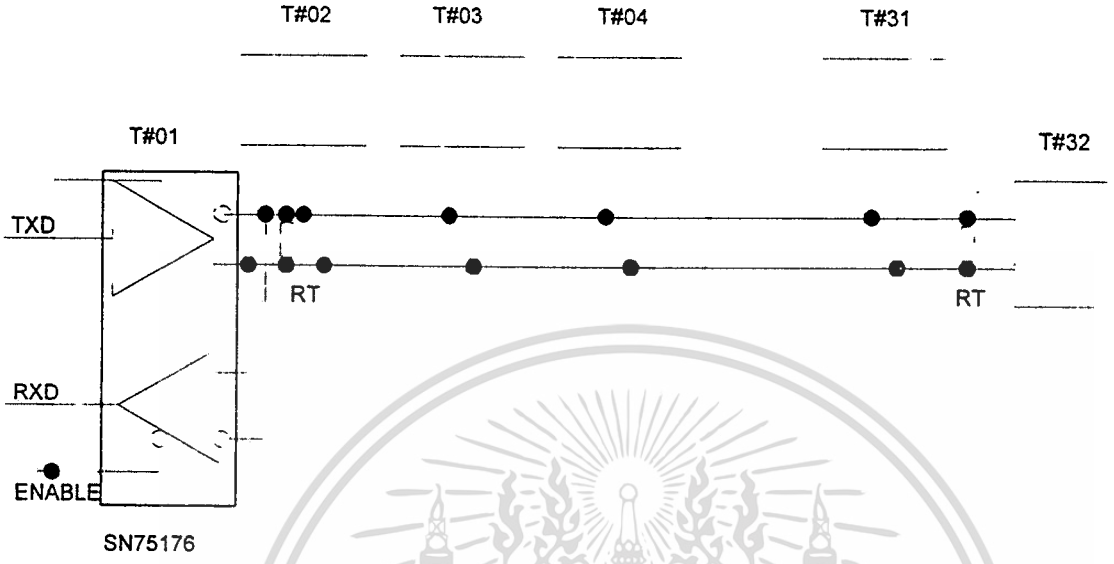
คู่ตัวรับส่ง (Transceivers) เป็นอุปกรณ์ที่มีตัวรับส่งอยู่ภายในตัวชิฟเดียวกันเพื่อความสะดวกในการนำไปใช้งาน และทำให้ไม่เปลืองพื้นที่บนแผ่นควบคุม คู่รับส่งของ RS-485 มีอยู่หลายเบอร์ แต่ที่นำมาใช้คือ SN75176B

- เป็นไปตามมาตรฐาน RS-485,RS-422A,CCITT V.11 และ X.27
- เอาท์พุทของตัวส่งเป็นแบบ 3-state
- เอาท์พุทตัวส่งสามารถขับกระแสได้สูงถึง 60 mA
- Thermal Shutdown Protection
- ค่าความต้านทานที่อินพุทของตัวรับ 20 k (น้อยที่สุด)
- ตัวรับมี input sensitivity $\pm$ 200 mA
- คู่ตัวรับมีค่า input hysteresis 50 mA
- ใช้ไฟเลี้ยงขนาด 5 โวลท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8.4 การประยุกต์ใช้งานแบบพื้นฐานของคู่รับส่ง SN75176

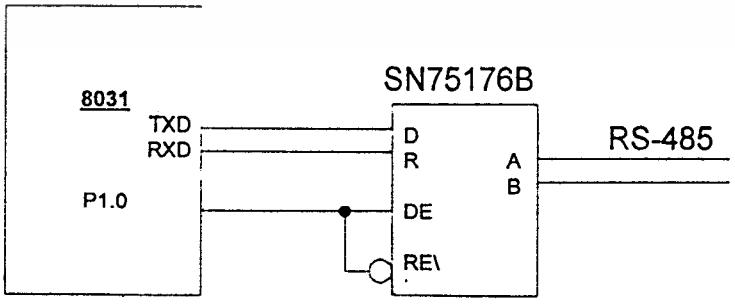
สามารถต่อใช้งาน SN75176 ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 การต่อใช้งาน SN75176

จากรูปที่ 2.9 แต่ละโหนดสามารถเป็นได้ทั้งตัวรับข้อมูลและตัวส่งข้อมูลก็ได้ โดยจะมีโหนดหนึ่งที่เชื่อมต่ออยู่กับเครื่องไมโครคอมพิวเตอร์ เพื่อทำหน้าที่ควบคุมการทำงานของระบบให้เป็นไปตามโปรโตคอลที่กำหนดไว้

MCS-51 เป็นไมโครคอนโทรลเลอร์ที่ถูกเลือกนำมาใช้เป็นหน่วยประมวลผลกลางของโหนดที่ทำหน้าที่ต่างๆของระบบ ซึ่งแต่ละโหนดสามารถรับส่งข้อมูลอนุกรมผ่านทาง RS-485 และ Asn1 Protocol โดยการเชื่อมต่อ MCS-51 เข้ากับคู่ตัวรับส่ง SN75176 ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 แสดงการเชื่อมต่อระหว่าง MCS-51 กับ SN75176B

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MCS-51 มีฟังก์ชั่นสำหรับการสื่อสารอนุกรมกับคอมพิวเตอร์ หรืออุปกรณ์ที่ใช้ในการสื่อสารข้อมูลอนุกรมได้ โดยใช้รีจิสเตอร์ SBUF (Serial Buffer) เป็นที่พักข้อมูลที่ทำดารรับหรือส่ง และใช้รีจิสเตอร์ SCON (Serial Port Control Register) เป็นส่วนควบคุมการสื่อสารอนุกรม นอกจากนี้ยังสามารถใช้งานไทม์เมอร์ 1 ให้ทำงานร่วมกับรีจิสเตอร์ TCON (Timer/Counter Control Register) และ รีจิสเตอร์ TMOD (Timer/Counter Mode Control Register) เพื่อกำหนดอัตราความเร็วในการรับ-ส่งข้อมูลอนุกรม

2.8.5โหมดการสื่อสารข้อมูลอนุกรมของ MCS-51 มีอยู่ด้วยกัน 4 โหมด คือ

โหมด 0 : Shift register mode

โหมด 1 : Standard UART mode

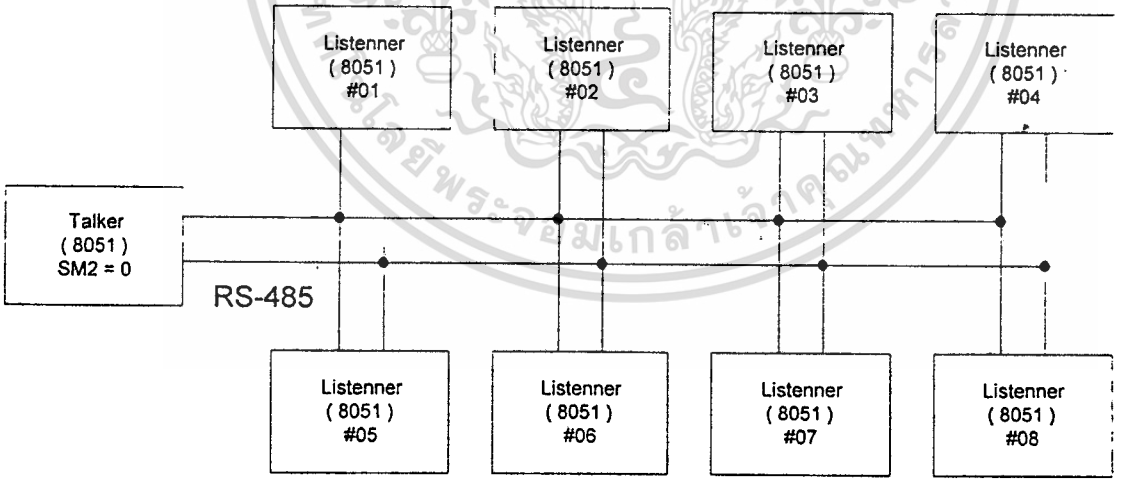
โหมด 2 : Multiprocessor fixed mode

โหมด 3 : Multiprocessor variable mode

การใช้งานด้านการสื่อสารข้อมูลอนุกรมของ MCS-51 สำหรับประยุกต์ใช้ในระบบดูแลความปลอดภัย ได้เลือกการทำงานในโหมด 3 และใช้อัตราความเร็วในการรับส่งข้อมูลอนุกรมเท่ากับ 19,200 บิทต่อวินาที (สามารถทำการเปลี่ยนแปลงได้)

2.8.6 การทำงานของ MCS-51 ในการรับส่งข้อมูลอนุกรมแบบโครงข่าย

MCS-51สามารถรับส่งข้อมูลอนุกรมกับตัวอื่นๆ ที่เชื่อมต่อในโครงข่ายได้ดังนี้



รูปที่ 2.11 แสดงการเชื่อมต่อการรับส่งข้อมูลอนุกรมแบบ multiprocessor ของ MCS-51

จากรูปการทำงานของโครงข่ายจะเริ่มต้นดังนี้

- MCS-51 ตัวแม่จะทำการส่งข้อความให้แก่ตัว MCS-51ลูกโดยระบุตำแหน่งของตัวลูกที่ต้องการจะติดต่อด้วย (address byte)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- MCS-51 ตัวลูกทุกตัวที่ต่ออยู่ในโครงข่ายจะได้รับข้อความจากตัวแม่ พร้อมทั้งทำการอ่านค่าตำแหน่งที่ตัวแม่ส่งมาให้มาทำการตรวจสอบ

- MCS-51 ตัวลูกทุกตัวจะตรวจสอบค่าตำแหน่ง ที่ตัวแม่ส่งมาว่าเป็นค่าตำแหน่งของตัวเองหรือไม่

ถ้าใช่ ให้ส่งข้อความตอบรับไปยังตัวแม่

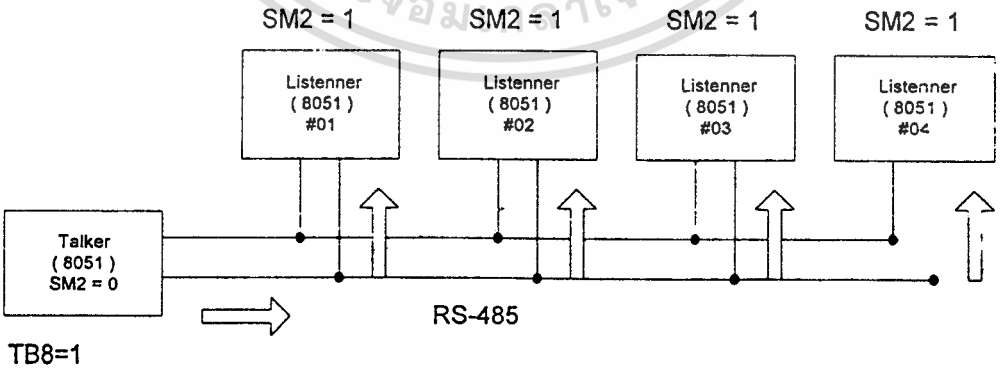
ถ้าไม่ใช่ไม่ต้องส่งข้อความใดๆไปให้ MCS-51 ตัวแม่

เมื่อพิจารณาจากรูปจะเห็นว่าการทำงานของระบบโครงข่ายที่ตัวแม่และตัวลูกใช้ SN75176 เป็นชุดการรับส่งข้อมูลจะเกิดปัญหาขึ้นในกรณีที่ MCS-51 ตัวแม่จะทำการส่งข้อมูลไปยัง MCS-51 ตัวลูกที่จะทำการติดต่อด้วย โดยที่ MCS-51 ตัวลูกได้รับข้อมูลแล้วจะทำการส่งข้อมูลกลับมาเพื่อแจ้งให้ MCS-51 ตัวแม่ได้รับทราบว่าการเชื่อมต่อสมบูรณ์ แต่ผลที่ได้คือตัวลูกุกตัวในระบบโครงข่ายจะได้รับข้อความนี้ด้วยส่งข้อความตอบรับนี้ตัวแม่ควรจะได้รับเพียงตัวเดียวเท่านั้นการแก้ไขในส่วนนี้กระทำได้โดยพิจารณาจากคู่มือการทำงานของรีจิสเตอร์ SCON (Serial Port Register Bit Address)

จากรายละเอียดของ SM2 ได้ระบุว่าถ้าตัวรับทำการตั้งค่า SM2 =1 การอินเทอร์รัพท์การรับข้อมูลอนุกรมโดยข้อมูลอนุกรมจากตัวส่ง (RI) จะเกิดขึ้นเมื่อตัวส่งทำการส่งข้อมูลที่มีการเซ็ตบิท TB8 =1 ซึ่งค่านี้เมื่อตัวรับทำการรับเข้ามาจะเก็บไว้ที่ RB8 ของตัวรับหรือถ้าตัวส่งทำการตั้งค่า SM2 = 0 การอินเทอร์รัพท์การรับข้อมูลอนุกรมโดยข้อมูลอนุกรมจากตัวส่ง (RI) จะเกิดขึ้นเมื่อตัวส่งทำการส่งข้อมูลที่มีการเซ็ตบิท TB8 = 0 ซึ่งค่านี้เมื่อตัวรับทำการรับเข้ามาจะเก็บไว้ที่ RB8 0 ของตัวรับดังนั้นในการใช้งานจะเป็นดังนี้

เริ่มต้นด้วย

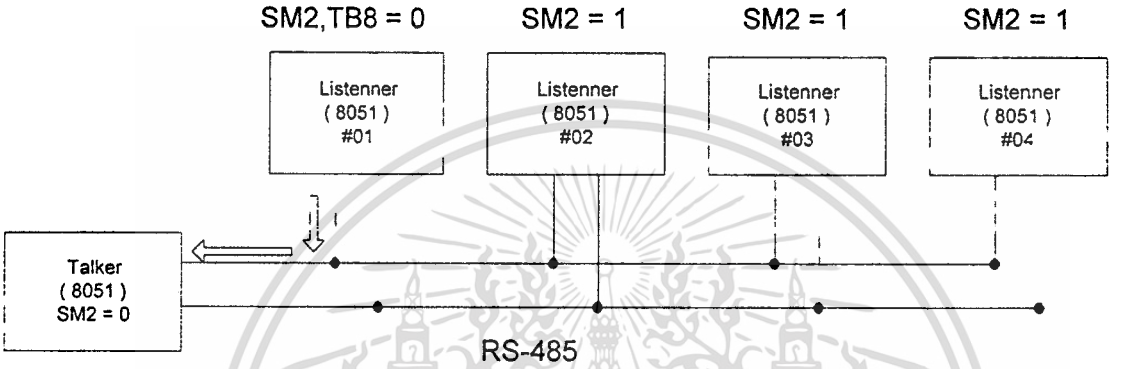
- MCS-51 ตัวแม่ส่งข้อความตำแหน่งไปให้แก่ MCS-51 ตัวลูก (Address byte)



รูปที่ 2.12 แสดงพารามิเตอร์ SCON ของ MCS-51 ตัวแม่ และ MCS-51 ตัวลูกเมื่อทำการส่งค่าตำแหน่งไปให้ตัวลูก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.12 เมื่อ MCS-51 ตัวแม่ทำการส่งข้อมูลตำแหน่งออกไป MCS-51 ตัวลูกทุกตัวในโครงข่ายจะได้รับข้อความจากตัวแม่ พร้อมทั้งอ่านค่าตำแหน่งตัวลูกจากข้อความดังกล่าวมาทำการตรวจสอบว่าเป็นค่าตำแหน่งของตัวเองหรือไม่ ในที่นี้ MCS-51 ตัวลูกที่ต้องการติดต่อด้วยคือ T01

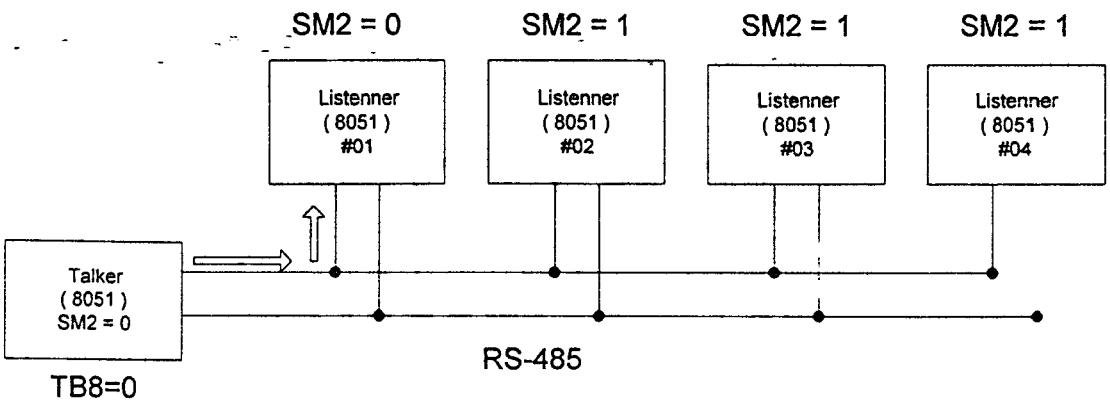


รูปที่ 2.13 แสดงพารามิเตอร์ SCON ของ MCS-51 ตัวแม่ และ MCS-51 ตัวลูก เมื่อ T01 ทำการส่งค่าตอบรับไปให้ตัวแม่

จากรูปที่ 2.13 เมื่อตัวแม่ส่งข้อความตำแหน่งไปให้ตัวลูกในโครงข่ายแล้วตัวแม่จะทำการเปลี่ยนค่า SM2 จาก 1 เป็น 0 เพื่อรอการตอบรับจาก T01 ส่วน T01 เมื่อรับข้อความจากตัวแม่และทำการตรวจสอบค่าตำแหน่งถูกต้องแล้ว จะตั้งค่า SM2 และ TB8 ให้เป็นลอจิก “0” เพื่อให้ข้อความที่ส่งออกไปตัวที่จะรับได้ต้องเซ็ทค่า SM2=0 เท่านั้น (เรียกว่า DATA BYTE) เพียงเท่านี้ก็จะทำให้ตัวลูกอื่นๆในระบบไม่สามารถรับข้อความนี้ได้

- ตัวแม่ทำการส่งข้อความในการใช้ควบคุมการทำงานของ ตัวลูก T01 จากรูปที่ 2.14 ตัวแม่จะทำการส่งค่า ACK ให้กับ T01 ซึ่งข้อความดังกล่าว T01 สามารถรับได้เพียงตัวเดียวเท่านั้นส่วนตัวลูกตัวอื่นๆไม่สามารถรับได้และเมื่อตัวแม่ทำการส่งข้อความ ACK ให้กับ T01 แล้วพารามิเตอร์ส่วน SCON ของทั้งตัวแม่และ T01 จะถูกเซ็ตค่าให้เหมือนตอนเริ่มต้นอีกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.14 แสดงพารามิเตอร์ SCON ของ MCS-51 ตัวแม่ และ MCS-51 ตัวลูก เมื่อ ตัวแม่ ทำการส่งค่า ACK ไปให้ตัวลูก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

ลักษณะของโครงงานระบบรักษาความปลอดภัยระยะไกล

3.1 ทฤษฎีทั่วไปเกี่ยวกับระบบรักษาความปลอดภัยระยะไกล

ระบบรักษาความปลอดภัยระยะไกลเป็นการประยุกต์ใช้เทคโนโลยีทางด้านไมโครโปรเซสเซอร์ทั้งทางด้านฮาร์ดแวร์ (Hard Ware) และซอร์ฟแวร์ (Soft Ware) มาใช้งานในด้านการอุตสาหกรรมต่างๆ ทำให้งานอุตสาหกรรมเป็นไปอย่างมีประสิทธิภาพมากขึ้น อีกทั้งยังสามารถนำระบบไปพัฒนาขีดความสามารถให้สูงขึ้นแล้วนำไปประยุกต์ใช้งานอื่นได้อีกด้วย

3.1.1 หลักการทำงานของระบบรักษาความปลอดภัยระยะไกล

ระบบรักษาความปลอดภัยระยะไกลได้ถูกออกแบบขึ้นมาตามมาตราฐาน RS-485 โดยต่อเป็นเครือข่ายหลายจุด (Multi drop Network) ซึ่งจะได้กล่าวถึงในรายละเอียดต่อไป

3.1.2 มาตราฐานการติดต่อสื่อสารข้อมูลแบบ RS-485

ในการสื่อสารข้อมูลแบบอนุกรม เรามักพบปัญหาที่เกิดขึ้นจากการผิดพลาดของข้อมูลได้ง่ายเนื่องมาจากงานที่ใช้มีการรับส่งข้อมูลด้วยอัตราการรับส่งข้อมูลสูง การรับส่งข้อมูลที่มีขนาดความยาวมากและใช้กับสายสัญญาณที่ไกลมาก รวมทั้งการนำมาใช้งานบางครั้งวางสายสัญญาณผ่านบริเวณที่มีการรบกวนสูง

ดังนั้น EIA จึงได้พัฒนาการสื่อสารข้อมูลอนุกรมแบบสมดุลย์ (ตัวรับและตัวส่งไม่ใช้สายกราวด์ร่วมกัน) เพื่อใช้แก้ปัญหาที่กล่าวมาข้างต้น โดยพัฒนาขึ้นสองมาตราฐาน คือ RS-422A และ RS-485 ซึ่งมีการประยุกต์การใช้งานที่แตกต่างกันขึ้นอยู่กับความเหมาะสมของงาน ในโครงงานนี้เราได้เลือกมาตราฐาน RS-485 มาใช้งานซึ่งถูกพัฒนามาจาก RS-422A เพื่อให้สามารถใช้ตัวรับตัวส่งจำนวนมากได้โดยใช้สายสัญญาณร่วมกันซึ่งมีข้อแตกต่างกันคือ มาตราฐาน RS-422A ในสายสัญญาณหนึ่งคู่ที่ใช้ร่วมกันจะมีตัวส่งได้หนึ่งชุดและมีตัวรับได้ไม่เกิน 10 ชุด แต่มาตราฐานของ RS-485 สามารถมีตัวส่งได้ 32 ชุด และตัวรับได้ 32 ชุดเช่นกัน ลักษณะทางไฟฟ้าของทั้ง 2 มาตราฐานจะสามารถนำมาใช้รับส่งข้อมูลโดยไม่จำกัดรูปแบบของโปรโตคอล ขึ้นอยู่กับรูปแบบในการนำไปประยุกต์ใช้ อีกทั้งตัวรับและตัวส่งของ RS-485 มีราคาไม่สูงนักจึงทำให้ถูกนำมาประยุกต์ใช้งานในระบบการสื่อสารข้อมูลอนุกรมแบบโครงข่ายกันอย่างแพร่หลาย ข้อแตกต่างระหว่างมาตราฐานต่างๆ ที่มีใช้กันอยู่

3.2 ข้อกำหนดในการสื่อสาร (Communication Protocol)

ในระบบที่ได้พัฒนาขึ้นมาเป็นการรับส่งข้อมูลแบบอะซิงโคนัส การออกแบบชุดของข้อมูลที่ใช้ในการติดต่อสื่อสารจะเรียกว่าบล็อกซึ่งถ้าได้รับการออกแบบที่ดีจะทำให้ช่วยลดเวลาใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สื่อสารลงได้มากเพราะเวลาในการตอบสนองการรับส่งข้อมูลของระบบจะขึ้นอยู่กับความเร็วในการส่งผ่านข้อมูล จำนวนของข้อมูลและเวลาในการสแกนสถานะของเทอร์มินัลต่อไปจะเป็นรูปแบบของโปรโตคอลที่ได้พัฒนาขึ้น โดยเฉพาะในโครงงานนี้เรียกว่า ASN Protocol (Asyncronous Serial Data Communication Network Protocal) เพื่อใช้เป็นข้อตกลงในการสื่อสารข้อมูลในการส่งข้อมูลจากสถานีควบคุมผ่านระบบเชื่อมต่อไปยังสถานีย่อย เราจะเรียกว่าบล็อกของคำสั่ง (Command Block) และบล็อกของข้อมูลที่ถูกส่งจากสถานีย่อยเพื่อตอบรับการทำงานจากสถานีควบคุมเราเรียกว่าบล็อกตอบสนอง (Response Block) ลักษณะของบล็อกที่สร้างขึ้นมีรูปแบบดังนี้

3.2.1 รูปแบบของบล็อก (Block Format)

Header	Data	END

Header Part

ในส่วนของ Header นี้จะเป็นส่วนของรหัสที่ใช้ควบคุมการรับส่งข้อมูลโดยมีรายละเอียดดังนี้

50H	Unit No.	IC
8 บิต	24 บิต	

50H : Start of Header (ASCII = 40H) [@=8 bit]

ใช้แสดงการเริ่มต้นของบล็อกซึ่งมีขนาด 8 บิต

Unit No. : หมายเลขของโมดูลที่จะติดต่อด้วยหรือเรียกอีกอย่างว่าหมายเลขแอดเดรสปลายทาง (Destination Address) ซึ่งมีขนาด 24 บิต มี 3 แบบคือ

-. เมื่อตัวรับเป็นสถานีควบคุม

Unit No. = H00 : H=40H (ASCII)

00=00H (ASCII 2 byte)

ใช้สำหรับสถานีควบคุมโดยเฉพาะ

-. เมื่อตัวรับเป็นสถานีย่อย

Unit No. = Sxx : X = 53H (ASCII)

xx = 01H-32H (ASCII 2 byte)

หมายเลขของสถานีย่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-. เมื่อตัวรับเป็นเทอร์มินัล

Unit No. = Txx : T = 54H (ASCII)

xx = 01H-32H (ASCII 2 byte)

หมายเลขของเทอร์มินัล

IC : Instruction Code ใช้แสดงคำสั่งควบคุมเพื่อให้โมดูลปลายทางปฏิบัติมีขนาด 2 ไบท์

Data Part ส่วนที่ใช้เก็บข้อมูลที่จะส่งไป

มีรายละเอียดดังต่อไปนี้

SEP	COD	SDB	DATA	EDB

SEP : Separator (รหัส ASCII = "-" = 2Dh) [8 bit]

ใช้กั้นระหว่างส่วนของ Header กับ DATA

COD : Count of data [16 bit] -> [ASCII 2 byte]

เป็นการแสดงจำนวนของข้อมูลที่บรรจุอยู่ใน Data Field โดยบรรจุได้สูงสุด 256 ไบท์

SDB : Start of Data Block ("[" = 5Bh) [8 bit]

เป็นส่วนที่แสดงการเริ่มต้นของข้อมูล

DATA : ข้อมูลที่จะส่งไปเก็บในรูป ASCII Code มีขนาดไม่เกิน 256 ไบท์

EDB : End of Data Block ("]" = 5Dh) [8 bit]

เป็นส่วนที่แสดงการสิ้นสุดของข้อมูล

End Part

ส่วนปิดท้ายของบล็อกข้อมูล ใช้เก็บค่ารหัสควบคุมความผิดพลาด โดยใช้ค่า FCS (Frame Check Sequence) ในการตรวจสอบรายละเอียดมีดังนี้

*	FCS	FCS	EOB
	[hi]	[lo]	

* : ใช้กำหนดจุดสิ้นสุดที่ใช้ในการคำนวนค่า FCS ("*" = 2Ah) [8 bit]

EOB : End of Block ใช้แสดงจุดสิ้นสุดของข้อมูล ("#" = 23h) [8 bit]

FCS : เป็นข้อมูลขนาด 8 บิทที่ถูกแปลงไปเป็น ASCII และค่าดังกล่าวหาได้จาก การนำข้อมูลที่อยู่ระหว่าง @ ไปจนถึงสิ้นสุดข้อมูลที่ * โดยนำข้อมูลทั้งหมดมา XOR กันเป็นลำดับของแต่ละอักขระในบล็อกเช่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความหมายคือ อ่านค่าสถานะจากเทอร์มินัล 01,02 จากสถานีย่อย 01

3.2.2 ชุดคำสั่งของ ASN Protocol

ในขณะที่ทำการเชื่อมต่อเพื่อทำการสื่อสารข้อมูล Computer ที่สถานีควบคุมจะทำการเฝ้ามองการดำเนินการและสามารถควบคุมการทำงานของสถานีย่อยได้โดยการส่งข้อมูลชุดคำสั่งออกไป ชุดคำสั่งหรือฟังก์ชั่นการทำงานของ ASN Protocol ได้พัฒนาขึ้นเพื่อใช้งานเฉพาะกับระบบนี้ โดยมีคำสั่งต่างๆ ที่เป็นข้อกำหนดดังนี้

-Install บล็อกคำสั่งจะถูกส่งออกไปจากสถานีควบคุมเพื่อทำการติดตั้งสถานีย่อย หรือเทอร์มินัลเข้ากับระบบเครือข่าย ซึ่งมีรูปแบบคำสั่งดังนี้

@	Sxx	IS	-	0	[ ]	*	FCS [hi]	FCS [lo]	#

เช่น Block Command เป็น @S01IS-0[ ]*39#

หมายถึง สถานีควบคุมส่งบล็อกออกไป เพื่อติดตั้งสถานีย่อยหมายเลข 01 เชื่อมต่อเข้ากับระบบ

-รูปแบบการตอบรับ

@	H00	IS	-	0	1	[xx]	*	FCS [hi]	FSC [lo]

xx = Rseponse Code 00 = การติดตั้งสมบูรณ์

08 = มีความผิดพลาดเกิดขึ้น ยกเลิกการ install

3.2.3 การติดตั้งเทอร์มินัล

รูปแบบคำสั่ง

@	Sxx	IT	-	NN[DD,DD,DD]	*	FCS [hi]	FCS [lo]	#

Sxx : สถานีย่อยที่จะติดตั้งเทอร์มินัล

IT : คำสั่งติดตั้งเทอร์มินัล

NN : จำนวนเทอร์มินัลที่ทำการติดตั้ง

DD : หมายเลขเทอร์มินัลที่ทำการติดตั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบการตอบรับ

@	H00	IT	-	NN[DD,DD,DD]	*	FCS [hi]	FCS [lo]	#

S00 : สถานีควบคุม

NN : จำนวนเทอร์มินัลที่ทำการติดตั้ง

DD : หมายเลขเทอร์มินัลที่ทำการติดตั้ง และสถานะการติดตั้ง

3.2.4 Uninstall

เพื่อสถานีควบคุมสามารถที่จะยกเลิกการเชื่อมต่อหน่วยควบคุม กับระบบเครือข่ายได้ มีรูปแบบดังนี้

-.ยกเลิกสถานีย่อย

@	Sxx	US	-	00[ ]	*	FCS [hi]	FCS [lo]	#

US : คำสั่งในการยกเลิกสถานีย่อย

รูปแบบการตอบรับ

@	H00	US	-	01 [xx]	*	FCS [hi]	FCS [lo]	#

xx : Response Code 00 = การยกเลิกติดตั้งสมบูรณ์

08 = มีข้อผิดพลาดเกิดขึ้น ยกเลิกการ Uninstall

3.2.5 Read Status

เพื่อให้สถานีควบคุมอ่านค่าเหตุการณ์ต่างๆ ที่เกิดขึ้นจากเทอร์มินัลได้ โดยมีรูปแบบคำสั่ง ดังนี้

@	Sxx	RS	-	NN[DD,DD,DD]	*	FCS [hi]	FCS [lo]	#

Sxx : สถานีย่อยหมายเลข 01-32

RS : คำสั่งให้เทอร์มินัลอ่านค่าสถานะกลับมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

NN : จำนวนเทอร์มินัลที่ทำการอ่าน

DD : หมายเลขเทอร์มินัลที่ทำการอ่าน

รูปแบบการตอบรับ

@	H00	RS	-	NN[DD,DD,DD]	*	FCS [hi]	FCS [lo]	#

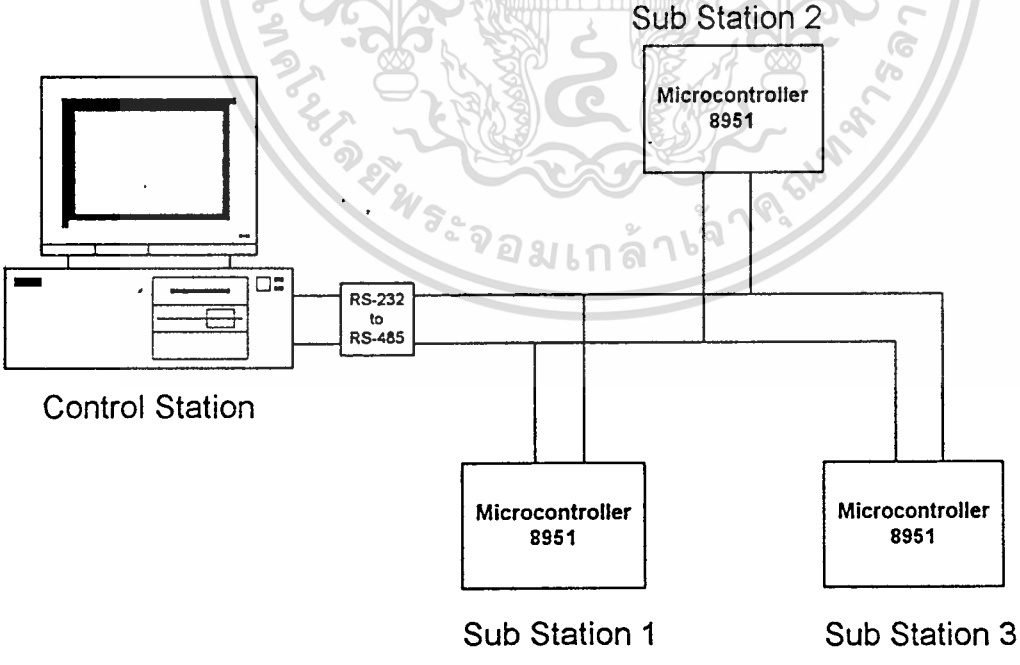
S00 : สถานีควบคุม

NN : จำนวนเทอร์มินัลที่ทำการติดตั้ง

DD : ข้อมูลของเทอร์มินัล N* (ตำแหน่งจะเรียงเหมือนตอนส่งไป)

3.3 โครงสร้างของระบบ

ในรูปที่ 3.1 จะประกอบไปด้วย Center 1 โมดูล จะสามารถเชื่อมต่อกับ Server ได้ถึง 32 โมดูล (โครงข่าย Center) และ Server 1 โมดูล จะสามารถเชื่อมต่อได้อีก 32 Terminal (เรียกว่าโครงข่าย Server) โดยที่ทั้งสองโครงข่ายจะเชื่อมต่อโดยใช้ RS-485 โดยจะใช้สายนำสัญญานเพียง 2 เส้นเท่านั้น นอกจากนี้ในส่วนของ Center จะเชื่อมต่อกับเครื่องไมโครคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม



รูปที่ 3.1 แสดงโครงข่ายของระบบดูแลความปลอดภัย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

หลักการเขียนโปรแกรม

ในโครงการ Computer Data Link Control System เราได้ออกแบบโปรแกรมขึ้นมาเพื่อใช้ทดสอบการทำงานของ RS-485 โดยเราได้แบ่งโปรแกรมออกเป็น 2 ส่วนคือ ส่วนของ Control Station และส่วนของ Sub Station จากการศึกษาการเขียนโปรแกรมด้วยภาษาใดภาษาหนึ่งเราควรเลือกศึกษาภาษาที่เราจะใช้บนเครื่องนั้นแล้ว เรายังต้องศึกษาถึงวิธีการและขั้นตอนในการสร้างโปรแกรมรวมทั้งระบบต่างๆของคอมพิวเตอร์ขอบเขตความสามารถและการใช้งานในส่วนอุปกรณ์ต่างๆ ที่จะประกอบกันขึ้น ยิ่งถ้าเราต้องการเขียนโปรแกรมที่มีความซับซ้อน และติดต่อกับระบบมากเท่าใด การศึกษาถึงตัวระบบของเครื่องคอมพิวเตอร์ ศึกษาถึงระบบปฏิบัติการที่คอมพิวเตอร์นั้นใช้อยู่ก็เป็นสิ่งจำเป็นภาษาซีนับเป็นภาษาหนึ่งที่อาจเรียกได้ว่าเป็นภาษามาตราฐานสำหรับการเขียนโปรแกรมคอมพิวเตอร์เราจะเห็นคอมไพเลอร์สำหรับภาษาซีมีใช้ตั้งแต่บนเครื่องระดับไมโครคอมพิวเตอร์จนถึงเมนเฟรม ในการเขียนโปรแกรมที่เข้าถึงระบบ ภาษาซีก็มีความอ่อนตัวถึงระดับใกล้เคียงกับ แอสแซมบลี อันเป็นภาษาบนระบบคอมพิวเตอร์ จากข้อดีของภาษาซีที่กล่าวมาในข้างต้นทำให้เราเลือกใช้ภาษาซีสำหรับโปรแกรมของเครื่องไมโครคอมพิวเตอร์ที่ทำหน้าที่เป็น Control Station และในส่วนของ Sub Station ซึ่งเราใช้ภาษาแอสแซมบลี ของ MCS-51 เขียนเนื่องจากภาษาแอสแซมบลีไม่ใช่ภาษาของนักคอมพิวเตอร์ทั่วไป เป็นภาษาสำหรับผู้มีความต้องการเป็นพิเศษที่ไม่อาจใช้ภาษาสูงทั่วไปมาทำงานเช่นนั้นได้การเขียนโปรแกรมด้วยภาษาแอสแซมบลีมักถูกมองว่าเป็นเรื่องยาก แต่ในความเป็นจริงการเขียนโปรแกรมภาษาแอสแซมบลีให้ถูกต้องตามไวยากรณ์นั้นง่ายมากเพราะไวยากรณ์ของภาษานี้เป็นแบบง่ายๆ พื้นฐานมากที่สุด นอกจากนี้จำนวนคำสั่งที่ใช้ก็มีไม่มากมายนักเพราะคำสั่งในภาษาแอสแซมบลีมีเฉพาะคำสั่งสำหรับการทำงานพื้นฐาน ภาษาแอสแซมบลีจึงเป็นเครื่องมือที่ใช้ประสิทธิของระบบคอมพิวเตอร์ได้สูงสุดโดยข้อดีของภาษาแอสแซมบลีมีดังนี้คือ ราคาถูก มีความยืดหยุ่นสูง

4.1 ความต้องการของระบบ

-เครื่องไมโครคอมพิวเตอร์รุ่น 386-DX ขึ้นไป

-มีหน่วยความจำอย่างน้อย 4 เมกะไบต์

-มีเนื้อที่ว่างบนฮาร์ดดิสก์ 5 เมกะไบต์

-มีการ์ดแสดงผล 16 สีขึ้นไป

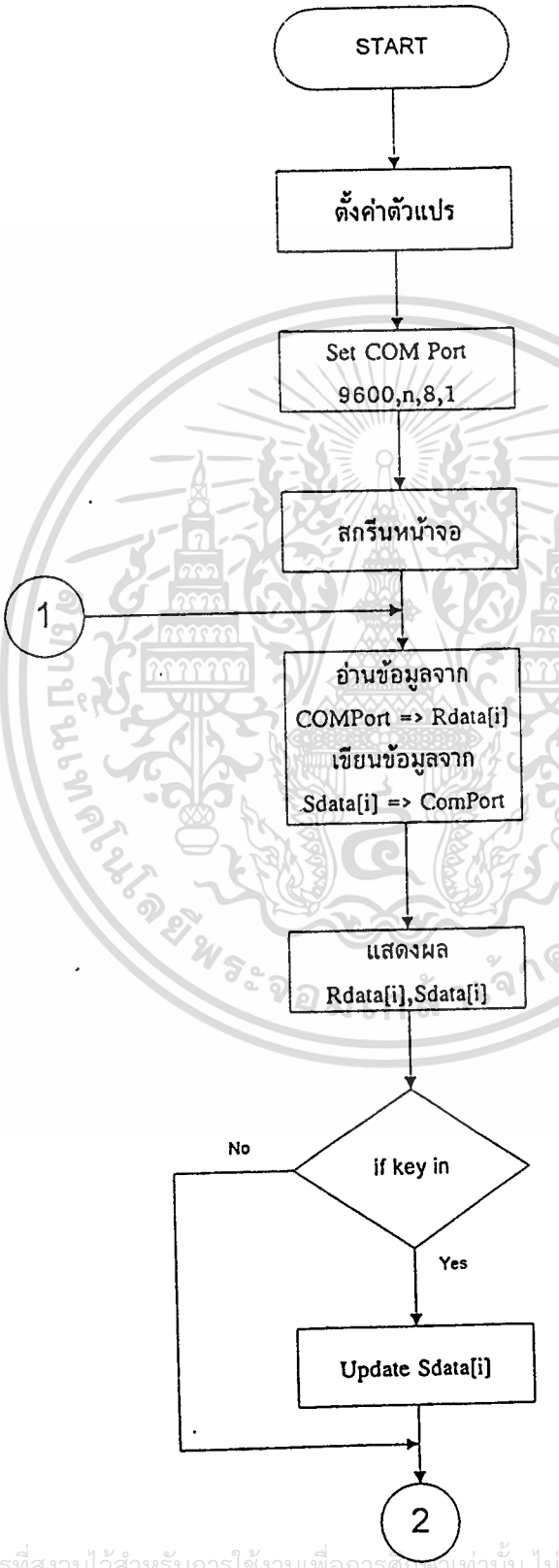
-ตัวแปลงสัญญาณ RS-232 เป็น RS-485

-ไมโครคอลโทรลเลอร์ 8951

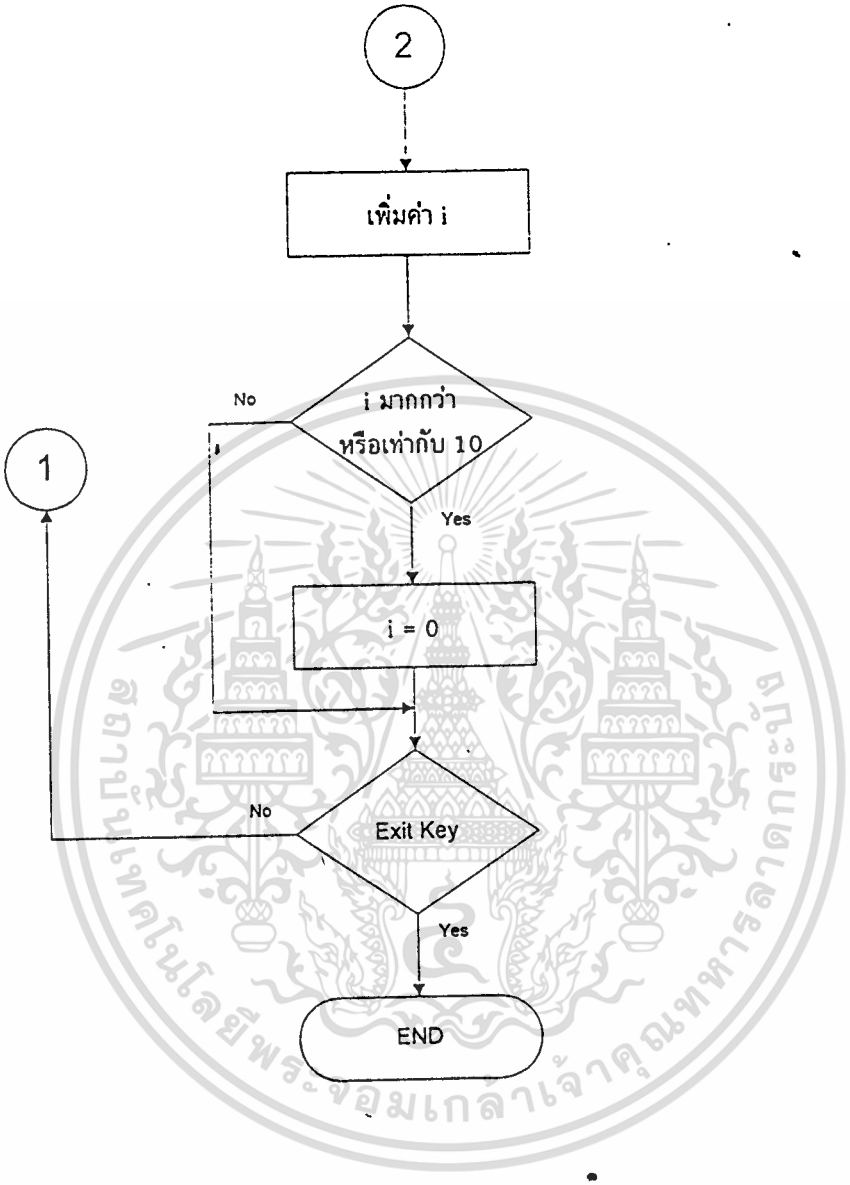
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 การทำงานของโปรแกรม

4.2.1 การทำงานโปรแกรม RMD.CPP ซึ่งเขียนโดยภาษา ซี มีผังการทำงานดังนี้



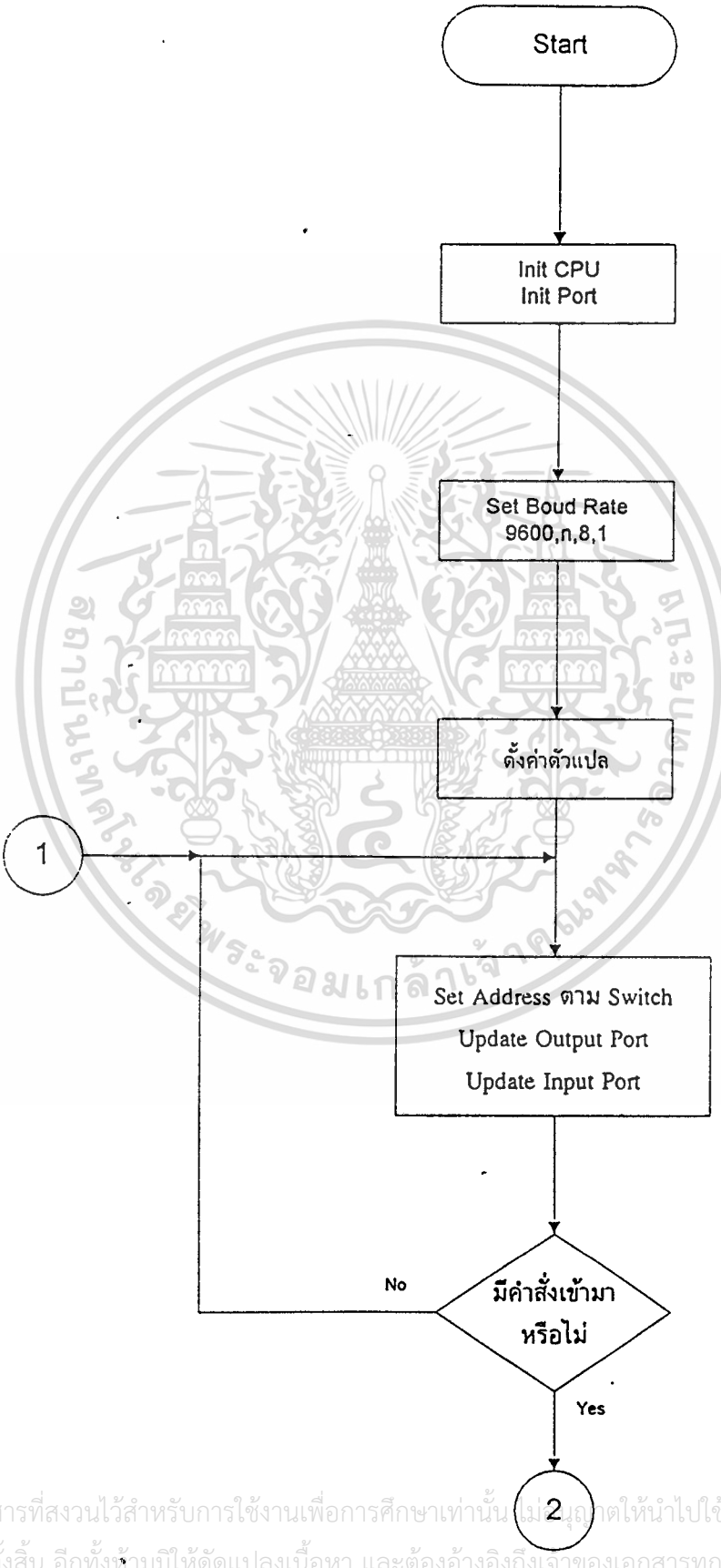
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



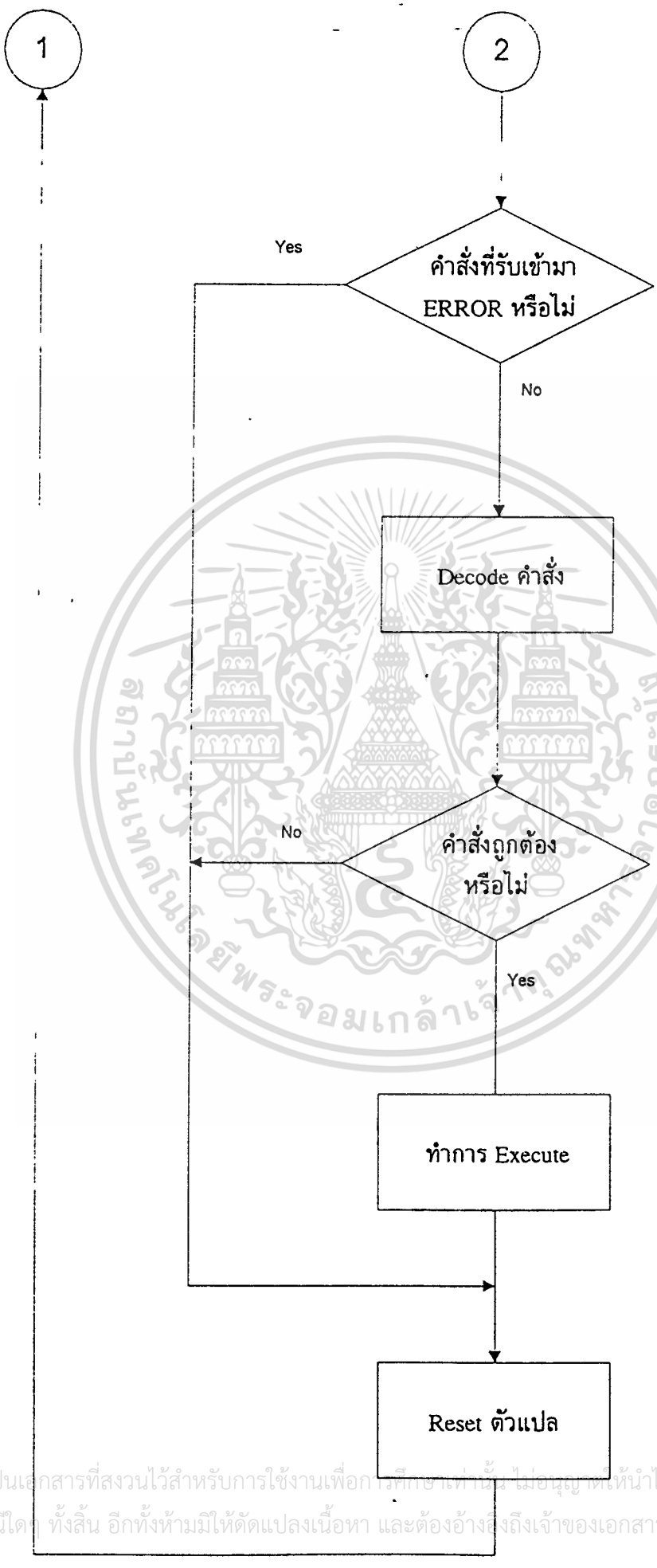
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2 การทำงานของโปรแกรม RMD.ASM ซึ่งเขียนขึ้นโดยแอสแซมบลี MCS-51 มีผังการทำงานของโปรแกรมซึ่งเป็นส่วนของ โปรแกรมหลักและ โปรแกรมย่อย ดังนี้

โปรแกรมหลัก

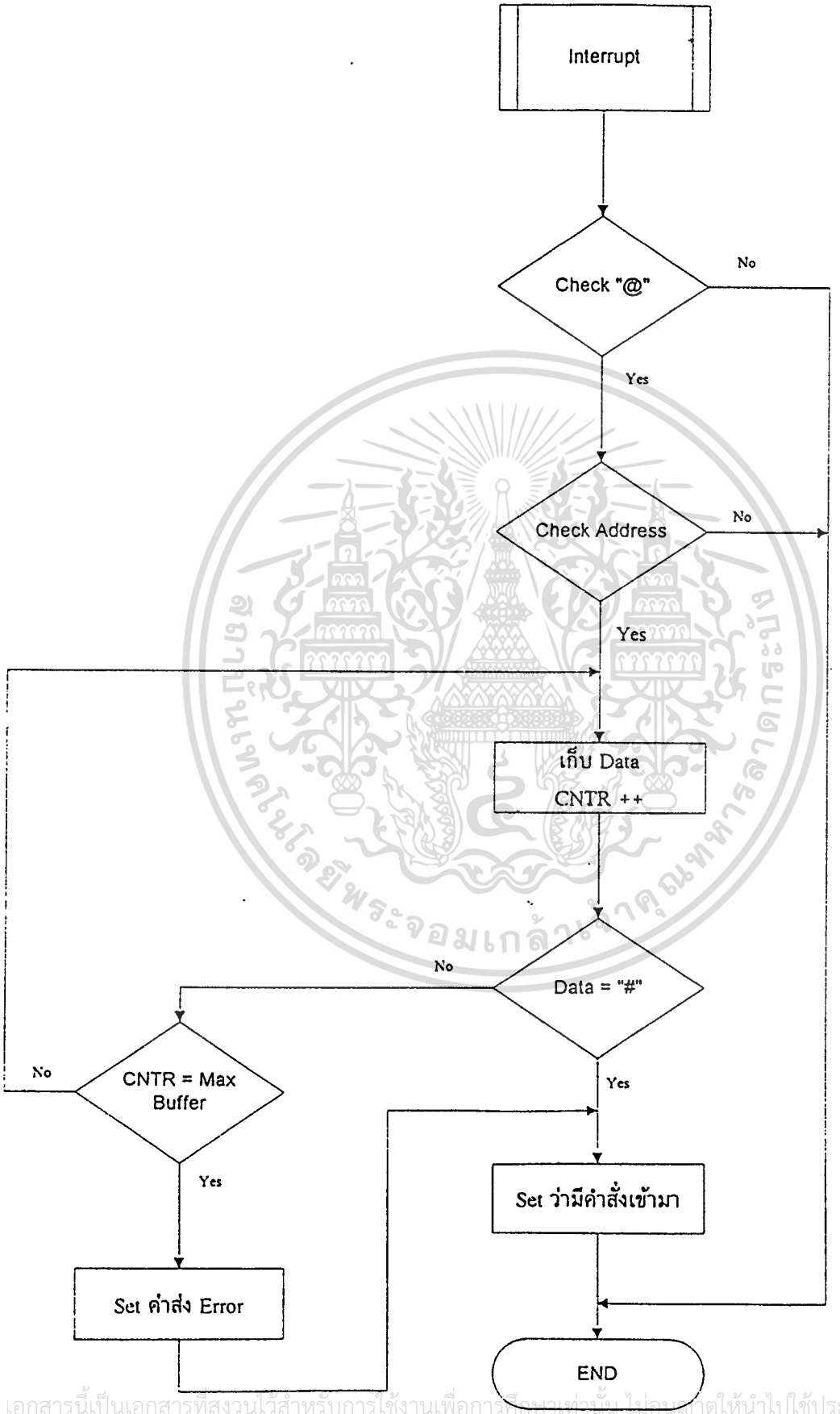


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



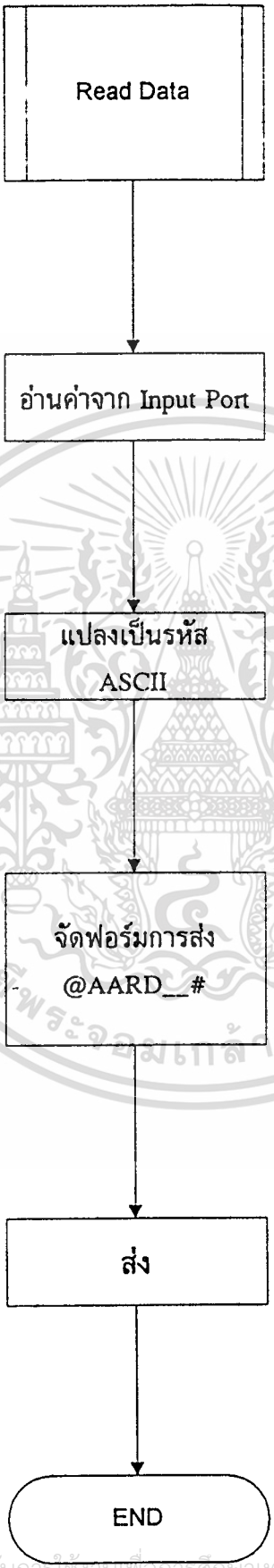
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยการ Interrupt



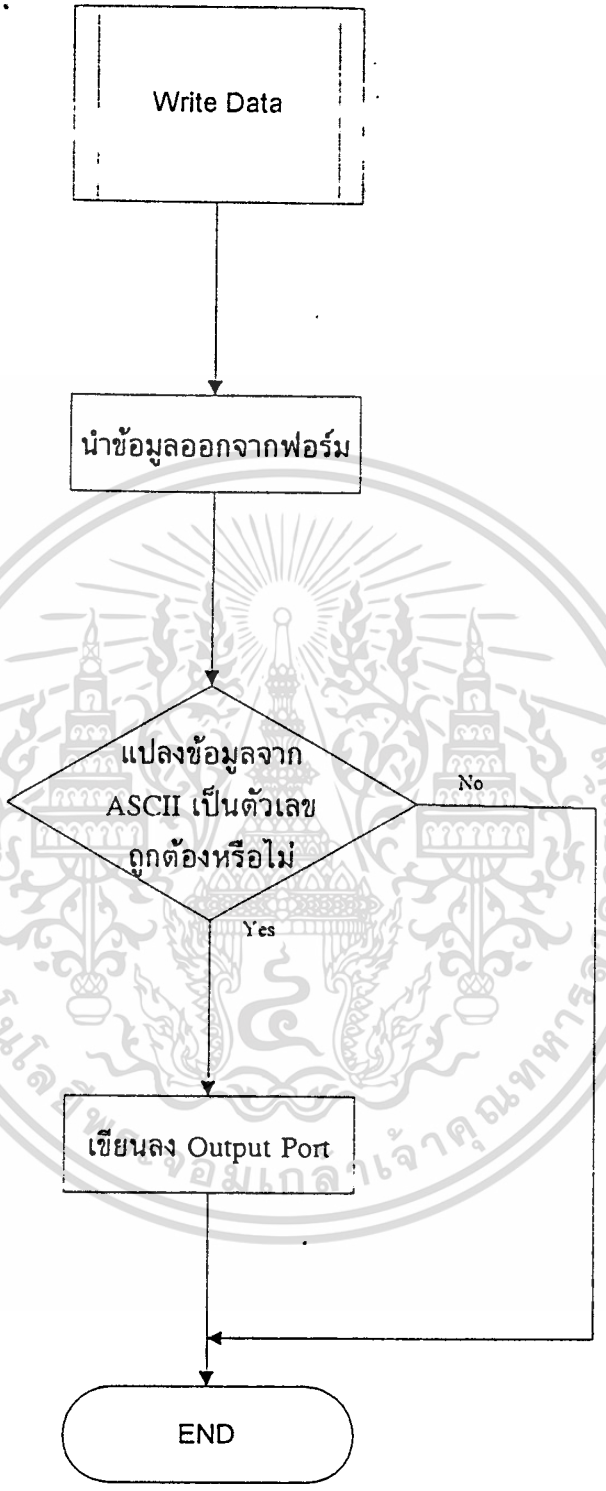
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-- โปรแกรมย่อยการ Read Data



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยการ Write Data



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

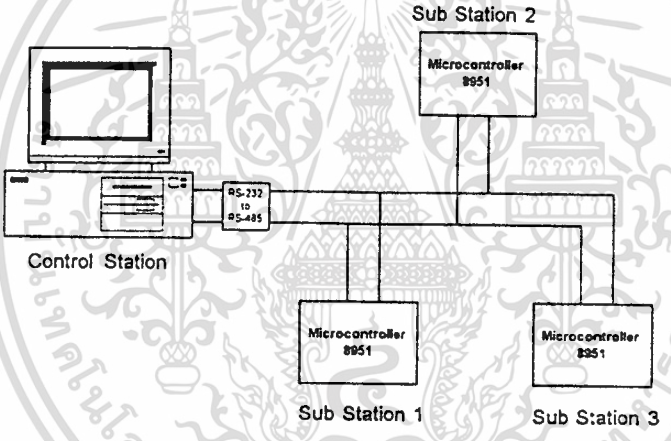
ผลการทดลอง

5.1 ผลการทดลอง

ส่วนประกอบของระบบที่ทำการทดสอบมี 3 ส่วนคือ

- สถานีควบคุมเป็นเครื่องไมโครคอมพิวเตอร์ 32 บิต
- สถานีย่อยใช้ไมโครคอลโทรลเลอร์ 8951 จำนวน 3 สถานี
- สายสัญญาณแบบ 2 เส้นมีฉนวนหุ้ม ยาว 800 เมตร

สำหรับการทดสอบทำการเชื่อมต่อระบบโดยรวมดังรูปที่ 5.1 โดยใช้สายโทรศัพท์ต่อเชื่อมทางกายภาพระหว่างสถานีควบคุม กับสถานีย่อยและเชื่อมต่อระหว่างสถานีย่อยอื่นทั้งหมดจนครบทั้ง 3 สถานี สำหรับความยาวของสายที่ใช้ทำการทดสอบมีความยาว 800 เมตรโดยมีอัตราเร็วในการรับส่งข้อมูลอยู่ที่ 9,600 บิตต่อวินาที



รูปที่ 5.1 แสดงการต่อโครงงานเพื่อทดลอง

จากการที่ได้ทำการทดลองการรับส่งข้อมูลตามมาตราฐาน RS-485 ซึ่งประกอบไปด้วยส่วนสำคัญ 2 ส่วนคือสถานีควบคุมและสถานีย่อย โดยสถานีควบคุมเราใช้เป็นเครื่องไมโครคอมพิวเตอร์ ขนาด 32 บิต ใช้เป็นสถานีหลักของระบบและส่วนการติดต่อสื่อสารจะใช้ตัวแปลงจาก RS-232 เป็น RS-485 โดยเสียบเข้าที่พอร์ต RS-232 ที่หลังเครื่องไมโครคอมพิวเตอร์ ส่วนหน่วยควบคุมระยะไกลได้สร้างและพัฒนาเพื่อเป็นโมดูลต้นแบบโดยใช้ไมโครคอลโทรลเลอร์ 8951 เป็นสถานีสำหรับเก็บข้อมูลและควบคุมการทำงานโดยตอบสนองต่อคำสั่งที่ได้รับจากสถานีควบคุม ซึ่งผลการทดลองแสดงดังตารางที่ 5.1 และตารางที่ 5.2 แสดงความเร็วในสายส่งเป็นสัดส่วนกับความยาวในสายตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mode	differential
Maximum number drivers receivers	 32 32
Maximum cable length	1200m
Maximum data rate (bits/s)	10M
Transmit levels	± 1.5V min
Receive Sensitivity	± 0.2V
Load impedance	60 Ω min
Output current limit	150mA to GND 250mA to -8V or +12V
Driver Load, min (power off)	120K

ตารางที่ 5.1 ผลการทดลอง

	Transmission Distance	Transmission Speed
RS-485	12m 120m 1200m	10M bits/sec 1M bits/sec 100K bits/sec

ตารางที่ 5.2 แสดงความเร็วในสายส่งเป็นสัดส่วนกับความยาวในสาย

5.2 สรุปผลการทดลองและปัญหา

จากผลการทดลองปรากฏว่าได้เกิดปัญหาขึ้นเนื่องจากเราไม่สามารถอ่านข้อมูลจากสถานีย่อยกลับมาที่สถานีหลักได้เมื่อใช้ RS-485 แต่สามารถเขียนข้อมูลจากหลักไปที่สถานีควบคุมได้อย่างถูกต้องตามมาตราฐาน RS-485 เนื่องจากข้อมูลที่อ่านมาจากสวิตช์ของไมโครคอนโทรลเลอร์จะไปกวนทางด้านรับของ RS-485 ทำให้ข้อมูลที่อ่านมาผิดจากข้อมูลที่ส่งไป แต่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ในการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการรับข้อมูล คือสถานีย่อยสามารถรับข้อมูลของสถานีหลักได้อย่างถูกต้องจากผลการทดลองตามตารางที่ 5.1 และตารางที่ 5.2 ได้ศึกษาพัฒนาและออกแบบสร้างระบบเก็บข้อมูลและควบคุมระยะไกลตามมาตราฐาน RS-485 สำหรับรับส่งข้อมูลสามารถสรุปได้ต่อไปนี้

- สถานีภายในระบบประกอยด้วย สถานีควบคุม และสถานีย่อย
- สถานีควบคุมเป็นเครื่องไมโครคอมพิวเตอร์ 32 บิต
- สถานีย่อยเป็นไมโครคอลโทรลเลอร์ 8951
- ระบบการติดต่อสื่อสารใช้ตัวกลาง (Media) เป็นสาย 2 เส้นตามมาตรฐาน RS-485
- อัตราเร็วในการรับส่งข้อมูลของระบบมีอัตราเร็ว 9,600 บิตต่อวินาที
- การเชื่อมต่อลำดับชั้นกายภาพของระบบ ต่อเป็นเครือข่ายหลายจุด (Multipoints) ตามมาตราฐาน RS-485
- จำนวนสถานีย่อยสูงสุด 16 สถานี
- สายที่ใช้ในการทดลองมีความยาว 800 เมตร

จากผลการทดลองปรากฏว่าระบบเก็บข้อมูลและควบคุมระยะไกลตามมาตราฐาน RS-485 ที่สร้างขึ้นมานี้สามารถทำตามฟังก์ชั่นหลักได้เป็นที่น่าพอใจ เพื่อเป็นแนวทางในการนำไปศึกษาปรับปรุงโครงงานและพัฒนานำเอาความสามารถของ RS-485 ไปใช้ได้อย่างเต็มประสิทธิภาพในงานทางด้านอุตสาหกรรมในอนาคต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
; REMOTE DATA Program
; CPU 8751
;-------------------------------------------------------
; define constant value

StartRx     equ   '@'
CR          equ   '#'
RxOL        equ   16
InPutPort   equ   P0
OutPutPort  equ   P1
CommMax     equ   3
Baud9600    equ   0FDh
Baud4800    equ   0FAh
Baud2400    equ   0F4h
Baud1200    equ   0E8h
rR0         equ   00h
rR1         equ   01h
rR2         equ   02h
rR3         equ   03h
rR4         equ   04h
rR5         equ   05h
rR6         equ   06h
rR7         equ   07h

; define address of variable
;base = 20h

Lock        equ   00h
fCommand    equ   01h
fError      equ   02h
base        equ   30h
RxCNTR      equ   base+0h
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
RxBufPTR    equ   base+1h
xSBUF       equ   base+2h
OutPutData  equ   base+3h
InPutData   equ   base+4h
RxBuf       equ   40h
Head        equ   RxBuf+0h
ID1         equ   RxBuf+1h
ID0         equ   RxBuf+2h
COMM1       equ   RxBuf+3h
COMM0       equ   RxBuf+4h
DATAn       equ   RxBuf+5h

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

            ;Reset,          , RST pin
            org 0000h
            SJMP ResetADDR

            ;Interrupt 0, IE0, -INT0 pin
            org 0003
            RETI                ;LJMP INTR0

            ;Timer 0,    TF0, T0 pin
            org 000Bh
             RETI               ;LJMP Timer0

            ;Interrupt 1, IE1, -INT1 pin
            org 0013h
            RETI                ;LJMP INTR1


            ;Timer 1,    TF1, T1 pin
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                org 001Bh
                RETI                    ;LJMP Timer1

                ;Serial,      RI & TI, RxD & TxD pin
                org 0023h
                AJMP Serial

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

                org 0026h
;=========================================================================
                ResetADDR:
                CLR     A
                DJNZ    ACC,S
                MOV     R0,#20h
RAL1:
                MOV     @R0,A
                INC     R0
                CJNE    R0,#80h,RAL1
                MOV     IE,#0           ;Disable all interrupt.
                MOV     SP,#07h
                MOV     A,#0FFh
                MOV     P0,A
                MOV     P1,A
                MOV     P2,A
                MOV     P3,A
                MOV     SCON,#01010000b ;Set serial to mode 1, 8 bits non parity
                MOV     TMOD,#00100000b;Set timer 1 to timer mode 2
                MOV     TH1,#Baud9600   ;
                ANL     PCON,#01111111b; Clear SMOD bit.
                SETB    TR1
                MOV     RxCNTR,#0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        MOV   RxBufPTR,#RxBuf
        CLR   fCommand      ;Enable Rx
        CLR   Lock
        CLR   fError
        ;SETB  P2.4         ; For Rx RS485.
        CLR   P2.4          ; For Rx RS485.
        SETB  EA            ;Enable all INTR.
        SETB  ES            ;Enable serial INTR.

;-----------------------
MainLoop:
        CPL   P2.7
        MOV   Head,#StartRx
        MOV   P2,#0FFh
        MOV   A,P2
        CPL   A
        ANL   A,#0Fh        ;
        ACALL BinHex        ;
        MOV   ID1,R7        ;
        MOV   ID0,R6        ;Get ID.
        MOV   A,OutPutData
        CPL   A
        MOV   OutPutPort,A
        MOV   InPutPort,#0FFh
        MOV   A,InPutPort
        CPL   A
        MOV   InPutData,A
        SETB  ES
        NOP
        NOP
        CLR   ES
        CPL   P2.7
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        JNB     fCommand,MainLoop

    ;----------------------
        JB      fError,EndLoop
        ACALL   DeCodeCom
        JC      EndLoop
        ACALL   Execute
EndLoop:
        MOV     RxCNTR,#0
        MOV     RxBufPTR,#RxBuf
        CLR     fCommand        ;Enable Rx
        CLR     Lock
        CLR     fError
        SJMP    MainLoop

;================================================================

;----------------------------------------------------------
; Interrupt service routine for serial.
;----------------------------------------------------------
Serial:
        CLR     TI
        JNB     RI,RxExit1
        CLR     RI
        JB      fCommand,RxExit1
        PUSH    ACC
        MOV     A,SBUF          ;Get data
        MOV     xSBUF,A
        JNB     Lock,HeadIn
        MOV     A,RxCNTR
        DEC     A
        JZ      CHK_ID1         ;=1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        DEC   A
        JZ    CHK_ID0      ;=2
                           ;=3
        PUSH  01           ;save R1
        MOV   R1,RxBufPTR  ;Get Pointer (PTR)
        MOV   A,xSBUF
        MOV   @R1,A        ;Keep the data
        POP   01           ;resave R1
        INC   RxCNTR
        INC   RxBufPTR
        CJNE  A,#CR,CHK_Tail ;Check CR
        SETB  fCommand     ;Command input
        CLR   Lock         ;UnLock
        SJMP  RxExit2      ;Rx Complete

CHK_Tail:
        MOV   A,RxCNTR
        CJNE  A,#RxOL,RxExit2
        SETB  fError
        CLR   Lock
        SJMP  RxExit2

HeadIn:
        MOV   A,xSBUF
        CJNE  A,Head,RxUnLock
        INC   RxCNT        ;=1
        INC   RxBufPTR
        SETB  Lock         ;To Lock
        SJMP  RxExit2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CHK_ID1:
        MOV   A,xSBUF
        CJNE  A,ID1,RxUnLock
        INC   RxCNTR            ;=2
        INC   RxBufPTR
        SJMP  RxExit2

CHK_ID0:
        MOV   A,xSBUF
        CJNE  A,ID0,RxUnLock
        INC   RxCNTR            ;=3
        INC   RxBufPTR
        SJMP  RxExit2

RxUnLock:
        CLR   Lock              ;To UnLock
        MOV   RxCNTR,#0         ;Reset Counter
        MOV   A,#RxBuf          ;
        MOV   RxBufPTR,A        ;Set start PTR
        ;SJMP  RxExit2

RxExit2:  POP   ACC
RxExit1:  CLR   RI

        RETI

;-------------------------------------------------------
;-------------------------------------------------------
; Output :    A (code of Comm..)
;             C (bit, if set is error)
; LEVEL 1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DeCodeCom:
        MOV   R5,#0           ;Set pointer = 0.
        MOV   DPTR.#COMMTable
        MOV   A,R5            ;\
        MOVC  A,@A+DPTR       ;\
        MOV   R4,A            ;\Get amount of command.
        INC   DPTR            ; Point to 1's Comm..

DCCL1:
        MOV   A,R5
        MOVC  A,@A+DPTR
        INC   R5              ;INC pointer.
        CJNE  A,COMM1,DCCL2
        MOV   A,R5
        MOVC  A,@A+DPTR
        INC   R5              ;INC pointer.
        CJNE  A,COMM0.DCCL2
        MOV   A,#CommMax      ;\
        SUBB  A,R4            ;\ Get code
        CLR   C
        SJMP  DCCL3

DCCL2:
        MOV   A,R5            ;\
        ANL   A,#11111110b    ;\
        ADD   A,#2            ;\
        MOV   R5,A            ;\ To next Comm..
        DJNZ  R4,DCCL1
        SETB  C               ;Set error
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DCCL3:
                RET

;---------------------------------------------------------
;  Input :      A ( Code of Comm.. )
;  LEVEL 1

Execute:
                JZ      COMM_00     ;=00
                DEC     A
                JZ      COMM_01     :=01
                DEC     A
                JZ      COMM_02     ;=02
                RET

;-----------------------------------
COMM_00:        ;-----TEST COMMAND.

                MOV     COMM1,#'T'  :\
                MOV     COMM0,#'S'  ;\ Put "TS".
                MOV     R1,#DATAn
                CJNE    @R1,#CR,C0L0
                ACALLSendCom        ; And send it.
C0L0:
                RET

;---------------------------------------
COMM_01:        ;-----READ COMMAND.
                MOV     COMM1,#'R'  ;\
                MOV     COMM0,#'D'  ;\ Put "RD".
                MOV     R1,#DATAn
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        CJNE  @R1,#CR,C1L0
        MOV   A,InPutData
        ACALLBinHex
        MOV   A,R7          ;
        MOV   @R1,A         ;nib 1
        INC   R1            ;
        MOV   A,R6          ;
        MOV   @R1,A         ;nib 0
        INC   R1            ;
        MOV   @R1,#CR
        CLR   A
        DEC   A
        JNZ   $-1
        DEC   A
        JNZ   $-1
        ACALLSendCom        ;Send data for read(RD) Comm..
C1L0:
        ;SETB  P2.7         ;RESET
        ;JMP   $
        RET

;-------------------------------------
COMM_02:  ;-----WRITE INSTRUTION.

        MOV   R1,#DATAn
        CJNE  @R1,#CR,C2L0
        JMP   C2Lx
C2L0:
        INC   R1
        INC   R1
        CJNE  @R1,#CR,C2Lx
        MOV   R1,#DATAn
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        MOV   R4,#2
C2L1:
        MOV   A,@R1
        INC   R1
        ACALL HexBin
        JC    C2Lx
        DJNZ  RR4,C2L
        MOV   R1,#DATAn     ;
        MOV   A,@R1         ;Lo
        ACALL HexBin        ;
        SWAP  A             ;
        MOV   R5,A          ;nib 1
        INC   R1            ;
        MOV   A,@R1         ;
        ACALL HexBin        ;
        ORL   A,R5          ;nib 0
        MOV   OutPutData,A
        CPL   A
        MOV   OutPutPort,A
C2Lx:
        RET

;-------------------------------------------------------
SendCom:
        PUSH  00
        PUSH  01
        PUSH  ACC
        MOV   R1,#RxBuf
        CPL   P2.4          ; For Tx RS485.
        CLR   ES            ;Disable serial INTR.
        SETB  TI
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SCL1:
        MOV   A,@R1
        JNB   TI,$
        CLR   TI
        MOV   SBUF,A
        ;ACALL Send
        INC   R1
        MOV   R0,#0
        DJNZ  R0,$
        DJNZ  R0,$
        DJNZ  R0,$
        CJNE  A,#CR,SCL1
        JNB   TI,$
        CLR   TI
        CPL   P2.4          : For Rx RS485.
        SETB  ES
        POP   ACC
        POP   01
        POP   00

        RET

;-------------------------------------------------------
; Input   :   A ( ASCII code of hex. )
; Output  :   A ( Binary code. )
;         :   C ( if set is error. )

HexBin:     CJNE  A,#30h,$+3    ; \
            JC    HBL2          ; \ jmp, if < 0 ,ERROR
            CJNE  A,#39h,$+5    ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                SJMP    HBL0            ;=
                JNC     HBL1            ; \ jmp, if > 9
HBL0:           ANL     A,#0Fh          ;
                CLR     C               ;
                SJMP    HBL3            ; 0 - 9
HBL1:           CJNE    A,#41h,$+3      ;
                JC      HBL2            ;< A ,ERROR
                CJNE    A,#46h,$+5      ;
                SJMP    HBL11           ;=
                JNC     HBL2            ;> F ,ERROR
HBL11:          ANL     A,#0Fh          ;
                ADD     A,#9            ; A - F
                SJMP    HBL3            ;
HBL2:           SETB    C               ;Set error
HBL3:           RET

;-------------------------------------------------------
; Input :       A
; Output :      R7_R6

BinHex:         MOV     R7,A
                ANL     A,#0Fh          ;Lo
                ORL     A,#30h
                CJNE    A,#3Ah,BHL1
BHL1:           JC      BHL2
                ADD     A,#7
BHL2:           MOV     R6,A
                MOV     A,R7            ;Hi
                SWAP    A
                ANL     A,#0Fh
                ORL     A,#30h
                CJNE    A,#3Ah,BHL11
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BHL11:    JC    BHL21
          ADD   A,#7
BHL21:    MOV   R7,A
          RET

;-------------------------------------------------------
; Input :   A ( Data for send. )
;
Send:     JNB   TI,$
          CLR   TI
          MOV   SBUF,A
          RET

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

          END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//Program CSP.CPP
//test RS-485
#include <bios.h>
#include <dos.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <conio.h>

#define ESCAPE      0x011b
#define UP_ARROW    0x4800
#define RIGHT_ARROW 0x4d00
#define LEFT_ARROW  0x4b00
#define DOWN_ARROW  0x5000
#define PGDN        0x5100
#define PGUP        0x4900
#define K_0         0x0b30
#define K_1         0x0231
#define K_2         0x0332
#define K_3         0x0433
#define K_4         0x0534
#define K_5         0x0635
#define K_6         0x0736
#define K_7         0x0837
#define K_8         0x0938
#define K_9         0x0a39

#define SIZE        16

int PORT = 1;
unsigned char Sbuff[ SIZE ];
unsigned char Rbuff[ SIZE ];
unsigned char Sdata;
unsigned char Rdata;
unsigned char Rxdata;
unsigned char Rxstatus;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
struct  time t, t1;
int en = 0;
int dataCNTR = 0;
int headin = 0;
int commin = 0;
int commerr = 0;

/*-------------------------------------------------------*/

unsigned int ReadKey( void );
void port_init( void );
int status( void );
int sport( unsigned char ch );
unsigned char rport( void );
int RComm( void );
void SComm( void );
char HexBin( unsigned char data );
char BinHex( unsigned char data );
void Write( unsigned char addr, unsigned char data );
int Read( unsigned char addr );
void pntbuf( void );

/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
unsigned int ReadKey( void )
{
  /*
   * Return ->
   * High byte : Scan code
   * Low byte  : ASCII code
   */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    int keycheck = bioskey( 1 );
    if( keycheck )
            return bioskey( 0 );
    else
            return 0;
}

/*-------------------------------------------------------------*/
void port_init( void )
{
    union REGS r;
    r.x.dx = PORT;
    r.h.ah = 0;
    r.h.al = 0xe3;
    int86( 0x14, &r, &r );
}

/*-------------------------------------------------------------*/
int status( void )
{
    union REGS r;
    r.x.dx = PORT;
    r.h.ah = 3;
    int86( 0x14, &r, &r );
    return r.x.ax;
}

/*-------------------------------------------------------------*/
int sport( unsigned char ch )
{
    union REGS r;
    r.x.dx = PORT;
    r.h.al = ch;
    r.h.ah = 1;
    int86( 0x14, &r, &r );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if( r.h.ah & 128 )
    //printf( "Send error detected in serial port" );
    return 0;
  else
    return 1;
}

/*-------------------------------------------------------*/
unsigned char rport( void )
{
  union REGS r;
  r.x.dx = PORT;
  r.h.ah = 2;
  int86( 0x14, &r, &r );
  Rxstatus = r.h.ah;
  /*
  if( r.h.ah & 128 )
    printf( "error" );
  */
  return r.h.al;
}

/*-------------------------------------------------------*/
int RComm( void )
{
  int head = 0;
  int tf = 0;
  int tf1 = 0;
  unsigned char data = 0;
  char i = 0;

  do
  {
    data = 0;
    gettime( &t );
    if( t.ti_sec == 59 ) t.ti_sec = 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    do
    {
      if( status() & 256 )
        tf1 = 1;
      else
        tf1 = 0;
      gettime( &t1 );
      /*
      printf("The current time1 is: %2d:%02d:%02d.%02d\n",
          t1.ti_hour, t1.ti_min, t1.ti_sec, t1.ti_hund);
      */
    } while( !tf1 && (t.ti_sec > t1.ti_sec-1) );

    if( tf1 )
    {
      data = rport();
      //printf(" %c_%x ",temp,temp);
      //printf(" i=%d ",i);
      if( !(Rxstatus & 128) )
      {
        if( !head )
        {
          if( data == '@' )
          head = 1;
        }
        else
        {
          if( data == '#' )
          tf = 1;
        }

        Rbuff[ i++ ] = data;
      }
    }
  } while( head && !tf && i<SIZE );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    //printf( " h=%d_tf1=%d_tf=%d ", head, tf1, tf );
    //pntbuf();

    return tf;
}

/*-------------------------------------------------------*/
void SComm( void )
{
    char i = 0;
    do
    {
        sport( Sbuff[ i ] );
        //printf( "%c",Sbuff[i] );
    }
    while( Sbuff[ i++ ] != '#' );
    //printf("\n");
}

/*-------------------------------------------------------*/
char HexBin( unsigned char data )
{
    if( data > 0x39 ) data -= 7;
    if( data < 0x30 || data > 0x3F )
        data = -1;
    else
        data &= 15;
    return data;
}

/*-------------------------------------------------------*/
char BinHex( unsigned char data )
{
    data |= 0x30;
    if( data>0x39 ) data += 7;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    return data;
}

/*-------------------------------------------------------*/
void Write( unsigned char addr, unsigned char data )
{
    Sbuff[ 0 ] = '@';
    Sbuff[ 1 ] = '0';
    Sbuff[ 2 ] = BinHex( addr );
    Sbuff[ 3 ] = 'W';
    Sbuff[ 4 ] = 'R';
    Sbuff[ 5 ] = BinHex( data>>4 ):
    Sbuff[ 6 ] = BinHex( data&15 ):
    Sbuff[ 7 ] = '#';
    SComm();
}
/*-------------------------------------------------------*/
int Read( unsigned char addr )
{
    int tf = 1;
    char i = 0;

    Sbuff[ 0 ] = '@';
    Sbuff[ 1 ] = '0';
    Sbuff[ 2 ] = BinHex( addr );
    Sbuff[ 3 ] = 'R';
    Sbuff[ 4 ] = 'D';
    Sbuff[ 5 ] = '#';
    SComm();
    /*
    en = 1;
    do{}while( !commin && commerr );
    en = 0;
    */
    if( RComm() )
    {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    for( ; i<5; i++ )
    if( Sbuff[ i ] != Rbuff[ i ] )
        tf = 0;
    if( Rbuff[ 7 ] != '#' ) tf = 0;
    if( tf )
    {
      tf = 0;
      if( HexBin( Rbuff[ 5 ] ) != -1 && HexBin( Rbuff[ 6 ] ) != -1 )
      {
        Rdata = (HexBin( Rbuff[ 5 ] ) << 4) | (HexBin( Rbuff[ 6 ] ) & 15);
        tf = 1;
      }
    }
  }
  else
    tf = 0;
  //port_init();
  //printf(" tf=%d ",tf);

  return tf;
}

/*-------------------------------------------------------*/
void pntbuf( void )
{
  char ch, i=0;

  do
  {
    ch = Rbuff[ i++ ];
    printf( "%c", ch );
  }
  while( (ch != '#') && (i < SIZE) );
  printf( "\n" );
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------------------------*/
void printbin( unsigned char data )
{
   char i;
   unsigned int d = (unsigned int)data;
   for( i=0; i<8; i++ )
      if( (d <<= 1) & 256 ) cprintf( "1" ); else cprintf( "0" );
   cprintf( " (%X%X)", data>>4, data&15 );
}
/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
void main( void )
{
   unsigned char ipdata[ 16 ];
   unsigned char opdata[ 16 ];
   unsigned char addr = 0;

   char i = 0;
   unsigned int key = 0;
   //int c = 1;
   int Exit = 0;


   for( i=0; i<SIZE; i++ )
      Rbuff[ i ] = Sbuff[ i ] = 0;
   for( i=0; i<16; i++ )
      ipdata[ i ] = opdata[ i ] = 0;

   port_init();

   textbackground( LIGHTGRAY );
   clrscr();
   textcolor( LIGHTRED );
   cprintf( "\r\n" );
   cprintf("                    REMOTE DATA\r\n");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  textcolor( BROWN );
  cprintf("
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
\r\n");
  textcolor( WHITE );
  cprintf("     ADDRESS          INPUT               OUTPUT                    \r\n");
  textcolor( BROWN );
  cprintf("
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
\r\n");
  gotoxy( 1, 22 );
  cprintf("
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
\r\n");

  i = 0;
  textcolor( BLACK );
  gotoxy( 70, 6+i ); cprintf( "=");
  gotoxy(  6, 6+i ); cprintf( "=");

  do
  {

    Write( addr, opdata[ addr ] );
    //Read( addr );
    //ipdata[ addr ] = Rdata;

    /*
    if( Read( addr ) )
    {
      ipdata[ addr ] = Rdata;
    }
    else
    {
      printf( " Error! \n" );
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
*/

textcolor( BLUE );
gotoxy(  8, 6+addr ); cprintf("%X", addr );
textcolor( YELLOW );
gotoxy( 20, 6+addr ); printbin( ipdata[ addr ] );
gotoxy( 42, 6+addr ); printbin( opdata[ addr ] );

textcolor( BLACK );
key = ReadKey();
if( key )
{
  switch( key )
  {
    case ESCAPE    : Exit = 1; break;
    case UP_ARROW  : gotoxy( 70, 6+i ); cprintf( " " );
                     gotoxy(  6, 6+i ); cprintf( " " );
                     if( --i <= 0  ) i =  0;
                     gotoxy(  6, 6+i ); cprintf( "" );
                     gotoxy( 70, 6+i ); cprintf( "" );
                     break;
    case DOWN_ARROW : gotoxy( 70, 6+i ); cprintf( " " );
                     gotoxy(  6, 6+i ); cprintf( " " );
                     if( ++i >= 16 ) i = 15;
                     gotoxy(  6, 6+i ); cprintf( "" );
                     gotoxy( 70, 6+i ); cprintf( "" );
                     break;
    case PGUP      : gotoxy( 70, 6+i ); cprintf( " " );
                     gotoxy(  6, 6+i ); cprintf( " " );
                     i =  0;
                     gotoxy(  6, 6+i ); cprintf( "" );
                     gotoxy( 70, 6+i ); cprintf( "" );
                     break;
    case PGDN      : gotoxy( 70, 6+i ); cprintf( " " );
                     gotoxy(  6, 6+i ); cprintf( " " );
                     i = 15;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                gotoxy(  6, 6+i ); cprintf( "□");
                gotoxy( 70, 6+i ); cprintf( "□");
                break;
        case K_8  : if( opdata[ i ] & 0x01 ) opdata[ i ] &= ~0x01;
                else opdata[ i ] |= 0x01;
                break;
        case K_7  : if( opdata[ i ] & 0x02 ) opdata[ i ] &= ~0x02;
                else opdata[ i ] |= 0x02;
                break;
        case K_6  : if( opdata[ i ] & 0x04 ) opdata[ i ] &= ~0x04;
                else opdata[ i ] |= 0x04;
                break;
        case K_5  : if( opdata[ i ] & 0x08 ) opdata[ i ] &= ~0x08;
                else opdata[ i ] |= 0x08;
                break;
        case K_4  : if( opdata[ i ] & 0x10 ) opdata[ i ] &= ~0x10;
                else opdata[ i ] |= 0x10;
                break;
        case K_3  : if( opdata[ i ] & 0x20 ) opdata[ i ] &= ~0x20;
                else opdata[ i ] |= 0x20;
                break;
        case K_2  : if( opdata[ i ] & 0x40 ) opdata[ i ] &= ~0x40;
                else opdata[ i ] |= 0x40;
                break;
        case K_1  : if( opdata[ i ] & 0x80 ) opdata[ i ] &= ~0x80;
                else opdata[ i ] |= 0x80;
                break;
    }
    key = 0;
}
else
    key = 0;

if( ++addr >= 16 )
{
    addr = 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        gotoxy( 1, 6);
      }
    }
    while( !Exit );
    clrscr();

}

/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงงานนี้สร้างสำเร็จลุล่วงลงได้ด้วยดี ก็เนื่องจากอาจารย์ในภาควิชาทุกท่านให้การสนับสนุนเป็นอย่างดี และบุคลที่สำคัญที่สุดคือ อาจารย์สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษาที่ให้การสนับสนุนและคอยให้คำชี้แนะในการทำงานเป็นอย่างดียิ่ง นอกจากนี้ยังขอขอบคุณเพื่อนๆ และน้องๆ ทุกคนที่ช่วยกันทำให้โครงงานนี้สำเร็จลงได้

และเพื่อเป็นการทดแทนความมีอุปการะคุณของอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ ผู้จัดทำจึงขอมอบโครงงานชิ้นนี้ให้เป็นสมบัติของภาควิชาเพื่อใช้ในการเรียนการสอนหรือพัฒนาโครงงานชิ้นใหม่ๆ แก่นักศึกษาของภาควิชาในรุ่นต่อไป

นายปิลันธน์ อุ่นตรงจิตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

ปรเมษฐ์ ประนยานันทน์, ปิยพงศ์ เผ่าวณิช. คู่มือและการประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) ,2536.

Herbert Schildt. การประยุกต์ใช้งานภาษาซี. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน), 2536.

ศิลารีเสิร์ช จำกัด. Data Book 2. กรุงเทพฯ : ประชุมทองการพิมพ์, 2539

จิรศักดิ์ เหลืองอุไร. คัมภีร์การใช้งานการสื่อสารอนุกรมบน PC. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) ,2538

Douglas V.Hall. Microprocessor And Interfacing Programming and Hardware. McGRAW-Hill Book Company: Singapore ,1986

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้