สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เครื่องควบคุมซาวด์แลบและส่วนรับฟัง
SOUNDLAB MASTER CONTROL AND HEARING UNIT

โดย

นายวราวุธ ดวงแก้ว 42015704

นายศุภฤกษ์ จวนชัยนาท 42015706

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เลขหมู่.....
เลขทะเบียน 45822
วัน, เดือน, ปี 18 ก.พ. 2546

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต
ภาควิชาเทคนิคอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องควบคุมซาวด์แลบและส่วนรับฟัง
SOUNDLAB MASTER CONTROL AND HEARING UNIT

โดย

นายวราวุธ ดวงแก้ว 42015704

นายศุภฤกษ์ จวนชัยนาท 42015706

อาจารย์ที่ปรึกษา

ผศ.อุทัย ศรีธีระวิโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	เครื่องควบคุมซาวด์แลปและส่วนรับฟัง
TITLE	Soundlab Master Control and Hearing Unit
โดย	นายวราวุธ ดวงแก้ว รหัสประจำตัว 42015704
	นายศุภฤกษ์ จวนชัยนาท รหัสประจำตัว 42015706
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ.อุทัย ศรีธีระวิโรจน์
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

( )

อาจารย์ผู้ควบคุมปริญญานิพนธ์

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	เครื่องควบคุมซาวด์แลบและส่วนรับฟัง
นักศึกษา	นายวราวุธ ดวงแก้ว รหัสประจำตัว 42015704
	นายศุภฤกษ์ จวนชัยนาท รหัสประจำตัว 42015706
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ.อุทัย ศรีธีระวิโรจน์
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตรบัณฑิต
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้มีรายละเอียดเกี่ยวกับการควบคุมห้องซาวด์แลบด้วยคอมพิวเตอร์ ซึ่งระบบนี้ทำงานโดย ส่วนควบคุมที่ทำหน้าที่รับสัญญาณเสียงต่างๆจาก เครื่องเล่นเทป เครื่องเล่น CD ไมโครโฟนแล้วเลือกสัญญาณเหล่านั้นเพื่อส่งไปยัง ส่วนรับฟังต่างๆซึ่งเป็นชุดไมโครโปรเซสเซอร์ MCS-51และแหล่งจ่ายไฟ โดยใช้คอมพิวเตอร์ส่วนบุคคลทำหน้าที่เป็นศูนย์กลางการติดต่อระหว่าง อุปกรณ์ส่วนรับฟังต่างๆ โดยผ่านไมโครโพเซสเซอร์(MCS-51) และระบบนี้โปรแกรมประยุกต์ที่ใช้เป็นเครื่องมือในการพัฒนาระบบ คือ โปรแกรมวิชวลเบสิค (Visual Basic)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PROJECT TITLE	Sound LabMaster Control and Hearing Unit
Student	Warawut Duangkaew
	Supparorg Juanchainat
ADVISOR	Asst.Prof.Uthai Sritheeravirojana
COURSE	Bachelor of Industrial Technology
DEPARTMENT	Telecommunication
YEAR	2001

ABSTRACT

This thesis describes the development of computer control sound lab system , The audio source such as tape play as CD player and microphone are connect to a sound controller , the selection signal is sent to hearing unit which are Microprocessor MCS-51 and electric supply , A PC is used as a host computer to control the communication among hearing unit devices by MCS-51 , Window Interface package Visual Basic is used as a development tool .

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำนำ

ในสมัยก่อนการสร้างห้องซาวด์แลบนั้น อุปกรณ์ต่างๆมีขนาดใหญ่และมากมาย ในต่อมาได้มีการพัฒนามาเรื่อยๆ จากที่ควบคุมด้วยกลไกมาเป็นระบบที่ใช้ดิจิตอลตอลควบคุม และในปัจจุบันคอมพิวเตอร์ ได้เข้ามามีบทบาทในทุกสาขาวิชาชีพ เนื่องจากมีความสะดวกและง่ายต่อการใช้งาน เราจึงได้นำระบบที่เป็นคอมพิวเตอร์ควบคุมซึ่งเป็นระบบหนึ่งในอีกหลายๆระบบที่ใช้คอมพิวเตอร์ควบคุม โดยสั่งงานผ่านพอร์ตอนุกรมของคอมพิวเตอร์ ไปยังส่วนรับฟังต่างๆ ซึ่งจะต้องมีคอนโทรลเลอร์บอร์ดเป็นตัวส่งผ่านข้อมูลระหว่างส่วนรับฟังกับคอมพิวเตอร์ ทำให้สามารถสั่งงานได้ทางคอมพิวเตอร์โดยตรง

ในปริญญานิพนธ์นี้ เริ่มต้นด้วยแนวคิดที่จะสร้างเครื่องควบคุมซาวด์แลป ที่เป็นอีกรูปแบบหนึ่งที่เคยมีมาในอดีต เพื่อที่จะให้ใช้งานได้ง่าย ต้นทุนถูก และประหยัดสายสัญญาณที่ต่อเชื่อมโยงระหว่าง เครื่องควบคุมของผู้สอนกับส่วนรับฟังกับผู้เรียน

สำหรับผู้ที่สนใจสามารถศึกษาได้จากปริญญานิพนธ์นี้ แล้วนำไปพัฒนาให้ดียิ่งขึ้นได้ เนื่องจากรูปแบบของโครงงานชิ้นนี้สามารถเพิ่มเติม หรือแต่งเติมได้อีกหลายแบบ

ผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

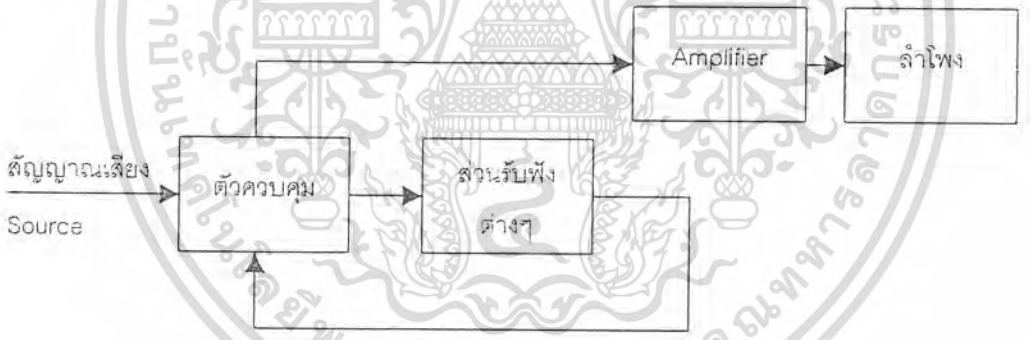
หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในสมัยโบราณมนุษย์เรามีการติดต่อสื่อสารกันเฉพาะกลุ่ม แต่ละกลุ่มแต่ละเชื้อชาติจะมีภาษาแตกต่างกันไป จนเมื่อมีการติดต่อแลกเปลี่ยนซึ่งกันและกัน ทำให้ต้องมีการติดต่อด้วยภาษาต่างๆมากมาย ทำให้ต้องเรียนรู้และศึกษาอื่นๆเพื่อที่จะทำให้เราติดต่อสื่อสารกัน รู้เรื่องและเข้าใจตรงกัน

จนมาถึงในปัจจุบัน ได้มีเทคโนโลยีต่างๆเข้ามามากมายจากต่างชาติ ซึ่งย่อโลกให้เล็กลง จึงทำให้มีการติดต่อสื่อสารกันมากขึ้น ซึ่งทำให้การที่ต้องเรียนรู้ภาษาต่างประเทศมีความสำคัญมาก และการเรียนภาษาต่างประเทศนั้นก็มีเทคโนโลยีในการสอนที่เพิ่มขึ้น มีการเรียนการสอนในรูปแบบที่ต้องใช้ห้องชาวด์แลบ ในการเรียนการสอนนักเรียน นักศึกษาให้ได้ฝึกพูด ฝึกฟังตามที่อาจารย์เสนอสอน ซึ่งในห้องชาวด์แลบนั้น เป็นที่รวบรวมเป็นที่รวบรวมเทคโนโลยี ในการสอนภาษาที่ปัจจุบันนิยมใช้คือ มีเทปคลาสเซ็ส ,CD-ROM ซึ่งมีเนื้อหาจากหลายๆผู้ผลิตมากมาย ในสื่อการสอนอันทันสมัยนี้ และส่วนประกอบหลักของเครื่องควบคุมชาวด์แลบมีดังนี้



ในการสร้างห้องชาวด์แลบนั้นสำคัญที่สุดคือ ต้องมีแหล่งกำเนิดเสียงซึ่งมาจากหลายแห่ง คือ เครื่องเล่นเทป , เครื่องเล่นCD , ไมโครโฟน เป็นต้น ซึ่งสัญญาณเสียงต่างๆจะถูกเลือกด้วยตัวควบคุม ซึ่งจะถูกส่งไปยังส่วนรับฟังทุกส่วนตามการควบคุมของตัวควบคุม

ในสมัยก่อนกลไกของเครื่องมือต่างๆเริ่มตั้งแต่การใช้เทคนิคพื้นฐานจากสวิตช์ ซึ่งเป็นคันโยกมาเป็นปุ่มกด และพัฒนามาเป็นสวิตช์อิเล็กทรอนิกส์ ซึ่งมีการพัฒนาให้มีขนาดเล็กที่สุดเท่าที่จะสามารถทำงานได้ปกติ มีรูปร่างลักษณะน่าใช้งาน และง่ายต่อการใช้งานที่สุด จากเดิมที่การควบคุม ทำงานด้วยระบบกลไก(MACHANICS) ก็พัฒนาเป็นการควบคุมด้วยระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดิจิตอล(DIGITAL)ซึ่งใช้การกดปุ่มเพื่อควบคุมการทำงานทั้งหมดของระบบ จึงมีความสะดวกและใช้งานง่ายขึ้น

ในปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทในทุกสาขาวิชาชีพ เนื่องจากมีความสะดวกและง่ายต่อการใช้งาน ตั้งแต่การแสดงผลซึ่งมีความละเอียด สวยงาม หลากหลาย ต่อการออกแบบสั่งงานที่สามารถรองรับคำสั่งได้ทั้งขนาด และจำนวนที่มหาศาล และยังมีความแม่นยำสูง ทั้งยังปรับเปลี่ยนการควบคุม อุปกรณ์ภายนอกได้อย่างหลายแบบจึงเป็นที่มาของโครงงานนี้

โครงงานเครื่องควบคุมซาวด์แลบและส่วนรับฟัง (SOUNDLAB MASTER CONTROL AND HEARING UNIT) เป็นการควบคุมระบบการติดต่อของส่วนรับฟังต่างๆ โดยมีตัวควบคุมเป็นตัวกำหนดรูปแบบต่างในการติดต่อสื่อสาร และเป็นตัวกำหนดเลือกแหล่งจ่ายข้อมูล โดยมีการควบคุมหลักจากเครื่องคอมพิวเตอร์ ซึ่งการสั่งงานของคอมพิวเตอร์ติดต่อผ่านพอร์ตอนุกรม (series port) จำนวนสามเส้นคือสายส่ง(TX) สายรับ(RX) และกราวด์(GROUND) ติดต่อกับคอนโทรลเลอร์บอร์ด ที่มีไอซี ไมโครคอนโทรลเลอร์ MCS-51 เป็นหัวใจในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2
ทฤษฎีไมโครคอนโทรลเลอร์ MCS-51 ชิปเดี่ยว

2.1 แนะนำไมโครคอนโทรลเลอร์ชิปเดี่ยว

ก่อนที่จะทำความรู้จักกับไมโครคอนโทรลเลอร์ เราควรจะทำความรู้จักกับที่มาที่ไปของชิปตัวนี้ เพื่อความเข้าใจที่ดีในการกล่าวถึงเกี่ยวกับไมโครคอนโทรลเลอร์กันซะก่อน เดิมทีเดียวนั้นในงานควบคุมระบบต่างๆ ได้นำไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์มาใช้ในงานควบคุมระบบต่างๆ โดยนำไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์เข้ามาใช้ในงานควบคุม โดยในการใช้งานนั้นจะต้องมีการต่อร่วมกันกับอุปกรณ์ภายนอก จำพวกพอร์ต I/O หน่วยความจำข้อมูล (RAM) หน่วยความจำโปรแกรม (ROM) ทั้งยังมีชิปจำพวก UART (Universal Aschronous Receiver Transmitter) เพื่อใช้ในการรับส่งข้อมูลแบบอนุกรม โดยมีหน้าที่เปลี่ยนรูปแบบข้อมูลแบบขนาน (Parallel) ไปเป็นข้อมูลในรูปแบบอนุกรม (Series) สำหรับทำการรับส่งชุดข้อมูลและแปลงข้อมูลจากรูปแบบอนุกรมกลับเป็นรูปแบบขนานในขั้นตอนการรับข้อมูล ซึ่งจะเห็นว่ากว่าจะนำไมโครโปรเซสเซอร์ หรือไมโครคอมพิวเตอร์ดังกล่าวมาใช้ในระบบควบคุมโดยเฉพาะนั้น จะค่อนข้างยุ่งยากลิ้นเปลืองอุปกรณ์ร่วมต่างๆ เพื่อที่จะให้คลอบคลุมในการใช้งานควบคุมที่มีประสิทธิภาพ ฉะนั้นจากปัญหาต่างๆที่ผ่านมา จึงทำให้บริษัทผู้ผลิตชิปไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์ดังกล่าวได้ทำการพัฒนา และผลิตชิปที่มีการรวมคุณสมบัติๆ เอาไว้ครบถ้วนเพื่องานควบคุมโดยเฉพาะ

โดยในชิปจะประกอบไปด้วย ไมโครโปรเซสเซอร์ , พอร์ต I/O , หน่วยความจำข้อมูลขนาดเล็ก ตลอดจนอาจจะมีพอร์ตใช้งานในการรับส่งข้อมูล (UART) และหน่วยความจำโปรแกรม (ROM) เข้าไปด้วยภายในชิปเพื่อใช้ในระบบควบคุมอย่างมีประสิทธิภาพสามารถใช้งานง่ายและประหยัดค่าใช้จ่ายเกี่ยวกับอุปกรณ์เสริมต่างๆ ชิปที่รวมคุณสมบัติต่างๆดังที่กล่าวมาข้างต้นจึงถูกเรียกว่า ไมโครคอนโทรลเลอร์ชิปเดี่ยว (Microtroller Unit :MCU) นั่นเอง โดยบริษัทผู้ผลิตก็ได้ผลิต MCU ออกมาหลายรูปแบบซึ่งจะมีคุณสมบัติและความสามารถที่แตกต่างกันเพื่อให้เหมาะกับงานในรูปแบบต่างๆ จึงทำให้มีการแยกประเภทของ MCU ออกเป็นตระกูลๆ หรือเบอร์ต่างๆโดยขึ้นอยู่กับคุณสมบัติของ MCU นั้นๆ และบริษัทผู้ผลิตเอง

โดยตระกูลที่ได้รับความนิยมก็คือ ไมโครคอนโทรลเลอร์ MCS-51 เนื่องจากมีข้อมูลหรือหนังสือที่เกี่ยวกับ MCU ตระกูลนี้อยู่พอสมควร

2.1.1 คุณสมบัติของ MCS-51

1. ใช้เทคโนโลยีชั้นสูงในการสร้างโดยมีทั้งประเภท HMOS , CMOS และ CHMOS ทำงานด้วยแหล่งจ่ายไฟ +5 Vdc เพียงแหล่งเดียว
2. มีหน่วยประมวลผลขนาด 8 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. สามารถติดต่อกับหน่วยความจำภายนอกทั้งหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้สูงสุด 64 Kbyte
4. มีพอร์ต I/O แบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิต รวมทั้งหมดเป็น 32 บิต แต่จะเหลือเพียง 16 บิต สำหรับเบอร์ 8031/8032 เนื่องจากพอร์ต 0 และพอร์ต 2 รวม 16 บิตจะใช้ในการเข้าถึงแอดเดรสและข้อมูลสำหรับติดต่อกับหน่วยความจำภายนอก
5. พอร์ตใช้งานทุกพอร์ตจะมีลักษณะเป็นพอร์ตแล็ช (Latch) คงสภาวะ
6. มีขาพอร์ตที่ใช้สำหรับทำการรับส่งข้อมูลแบบอนุกรม
7. หนึ่ง Machine cycle จะใช้เวลา 1 ไมโครวินาที โดยใช้ X-TAL 12 MHz
8. สามารถกำหนดการใช้งานพอร์ต I/O ได้ในระดับไบต์หรือบิตได้โดยตรง
9. ตัวเลขทางคณิตศาสตร์ใช้ได้ทั้งระบบฐานสองและฐานสิบหก

ตระกูลของ MCS-51 จะมีทั้งแบบมี EPROM ในตัวหรือไม่มี EPROM ภายใน (ในการกล่าวถึงหน่วยความจำโปรแกรมต่อไปนั้นผู้เขียนจะกล่าวถึง EPROM เป็นหลัก เนื่องจากเป็นชนิดประเภทของหน่วยความจำโปรแกรมที่สามารถนำมาประยุกต์ใช้งาน และเป็นที่รู้จักกันดี โดยขอให้ผู้อ่านเข้าใจว่าหน่วยความจำโปรแกรม EEPROM และ FLASH Memory ก็เป็นหน่วยความจำโปรแกรมที่สามารถนำมาใช้แทน EPROM ได้เช่นกัน ตารางที่2.1 แสดงถึงรายละเอียดของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 89XX ของบริษัท AMTEL และ เบอร์ 80C31,80C32 ของบริษัท INTEL

ตารางที่2.1 แสดงรายละเอียดของไมโครคอนโทรลเลอร์ตระกูล MCS-51

Features	89C1051	89C2051	80C31	80C32	89C51	89C52	89LV51	89LV52	89C55	89S8252
Flash Memory	1K	2K	-	-	4K	8K	4K	8K	20K	8K
Ram	64	128	128	256	128	256	128	256	256	256
EEPROM	-	-	-	-	-	-	-	-	-	2K
Programming	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Timers/Counters	1	2	2	3	2	3	2	3	3	4
Serial UART	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Power supply	2.7-6.0	2.7-6.0	4.0-6.0	4.0-6.0	4.0-6.0	4.0-6.0	2.7-6.0	2.7-6.0	2.7-6.0	2.7-6.0
Frequency	0-24	0-24	0-24	0-24	0-24	0-24	0-24	0-24	0-33	0-33
I/O Pins	15	15	32	32	32	32	32	32	32	32
External address / Data bus	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pin Count	20	20	40/44	40/44	40/44	40/44	40/44	40/44	40/44	40/44

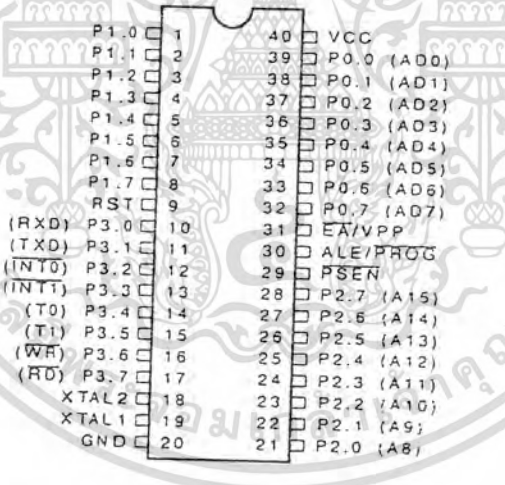
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.2 ลักษณะการจัดขาภายนอกของ MCS-51

รูปที่ 2.1 แสดงการจัดขาตามลักษณะภายนอกของชิป MCS-51 ซึ่งจะมีการแบ่งกลุ่มการจัดขาของ MCS-51 มีอยู่ 4 กลุ่ม คือ

1. กลุ่มขาแหล่งจ่ายไฟเลี้ยง และสัญญาณนาฬิกา
2. กลุ่มขาสำหรับอ้างแอดเดรสและรับส่งข้อมูล
3. กลุ่มขาที่ใช้ในการควบคุม
4. กลุ่มพอร์ตที่ใช้งานแบบขนานและอนุกรม

พอร์ตใช้งานบางพอร์ตจะทำหน้าที่ได้สองหน้าที่ขึ้นอยู่กับการสั่งงานด้วยซอฟต์แวร์ หรือการติดตั้งทางฮาร์ดแวร์เช่น พอร์ต 0 จะมีหน้าที่ใช้ในการอ้างแอดเดรส และอ่านข้อมูลจาก EPROM ภายนอก หรือจะทำหน้าที่เป็นกลุ่มขาพอร์ตแบบขนาน I/O ปกติในกรณี MCU ตัวนั้นมี EPROM ภายในเป็นตัวต้น รายละเอียดการทำงานตลอดจนโครงสร้างภายใน Microcontroller MCS-51 สามารถศึกษาได้จากคู่มือหรือ Data Sheet การใช้งาน Microcontroller MCS-51 โดยในที่นี้จะแนะนำรายละเอียดและคุณสมบัติในเบื้องต้นสำหรับนำมาประยุกต์ใช้งานต่อไป



รูปที่2.1 แสดงการจัดขาของไมโครคอนโทรลเลอร์ MCS-51 DIP40

2.1.3 ขาที่สำคัญของไมโครคอนโทรลเลอร์ MCS-51

1. ขา Vcc เป็นขารับแรงดันไฟกระแสตรง +5 Vdc
2. ขา GND เป็นขากราวด์
3. พอร์ต 0 มี 8 บิต ได้แก่บิต P0.0 – P0.7 เป็นพอร์ตอินพุทเอาต์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทต้องทำการเซตค่า 1 ไปยังพอร์ตเมื่อต้องการใช้งานพอร์ตนั้น ทั้งพอร์ตเป็นอินพุทถ้าต้องการใช้งานแต่ละบิตของพอร์ตเป็นอินพุตใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระดับบิต ก็สามารถกระทำได้โดยการเซตค่า 1 ไปยังแต่ละบิตที่ต้องการใช้งานเป็นพอร์ตอินพุทในระดับบิต เพื่อกำหนดให้ขาพอร์ตหรือแต่ละบิตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่นำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุทเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอก (EPROM , RAM) ได้อีกด้วย โดยทำหน้าที่ในการกำหนดแอดเดรสไบต์ต่ำ (A0-A7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กสำหรับการรับส่งข้อมูลขนาด 8 บิต (D0-D7)

4. พอร์ต 1 มี 8 บิต ได้แก่ บิต P1.0 – P1.7 เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าจะใช้งานเป็นอินพุตพอร์ตหรืออินพุตในระดับบิตสามารถกระทำได้โดยวิธีเช่นเดียวกันกับพอร์ต 0 ข้างต้น
5. พอร์ต 2 มี 8 บิต ได้แก่ บิต P2.0 – P2.7 เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าจะใช้งานเป็นอินพุตพอร์ตหรืออินพุตในระดับบิตสามารถกระทำได้โดยวิธีเช่นเดียวกันกับพอร์ต 0 ข้างต้น เช่นเดียวกันกับพอร์ต 0 นอกจากการใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้ว มันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอก (EPROM , RAM) ได้อีกด้วยโดยทำหน้าที่ในการอ้างตำแหน่งแอดเดรสไบต์สูง (A8 – A15)
6. พอร์ต 3 มี 8 บิต ได้แก่บิต P3.0 –P3.7 เป็นพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตหรืออินพุตในระดับบิตสามารถกระทำได้เช่นเดียวกันกับพอร์ต 0 ข้างต้น นอกจากจะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังสามารถใช้งานในหน้าที่พิเศษต่างๆ ดังตารางที่ 2.2

ตารางที่ 2.2 แสดงหน้าที่พิเศษของแต่ละขาของพอร์ต 3 ของไมโครคอนโทรลเลอร์

Pin Port	Discription
P3.0	RXD (Serial Input Port)
P3.1	TXD (Serial Output Port)
P3.2	INT0 (External Interrupt 0)
P3.3	INT1 (External Interrupt 1)
P3.4	T0 (Timer 0 External Input)
P3.5	T0 (Timer 1 External Input)
P3.6	RW (External Data Mamory Write Storbe)
P3.7	RW (External Data Mamory Read Storbe)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. ขารีเซต (RST) ใช้สำหรับการรีเซต การทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสภาวะ high อย่างน้อยนาน 2 Machine cycle ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่

8. ขา ALE / PROG เป็นขาสัญญาณเพื่อทำหน้าที่แลช (Latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุตรับพัลส์ในการโปรแกรม (Program Pulse Input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่มีหน่วยความจำโปรแกรมภายในเป็น EPROM

9. ขา PSEN (Program store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอกเมื่อไมโครคอนโทรลเลอร์ประมวลผลจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้ง ในแต่ละ Machine Cycle แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณ Store แต่อย่างใด

10. ขา EA/Vcc (External Access Enable/Vcc) เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมภายในหรือจากภายนอกโดยมีสถานะเป็น 0 และ 1 จะหมายถึงให้ไมโครคอนโทรลเลอร์ รับคำสั่งหน่วยความจำภายนอก และภายในตามลำดับ อย่างไรก็ตามถ้าบิตป้องกัน (Security Bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลยนอกจากนี้ ขานี้ยังมีหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (Vcc) ขนาด 12 โวลต์ เพื่อใช้ในระหว่างการโปรแกรมหน่วยความจำ โปรแกรม (EPROM) ภายในตัว MCU

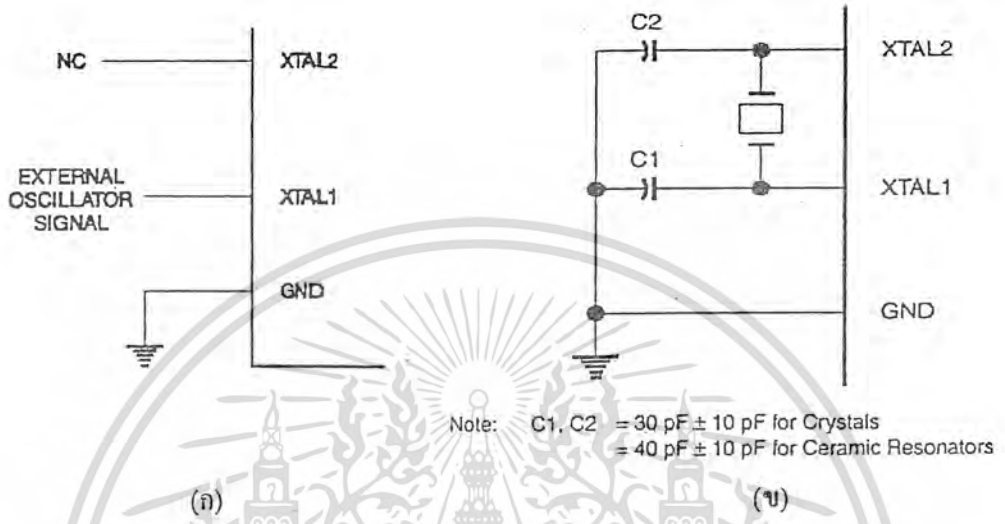
11. ขา XTAL1 และขา XTAL2 เป็นขาใช้งานของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (Inverting Oscillator Amplifier) สำหรับใช้ต่อร่วมกับคริสตัลภายนอก

2.1.4 Machine Cycle

ในการทำงานของ MCU นั้นจะถูกกำหนดความเร็วในการทำงานหรือการประมวลผลต่างๆ ด้วยสัญญาณนาฬิกา ที่ป้อนให้กับ MCU ไม่ว่าจะเป็นการต่อสัญญาณนาฬิกาโดยตรง ซึ่งวิธีการนี้ต้องคำนึงถึงชนิดหรือประเภทของ MCU นั้นๆ ด้วยว่าเป็น CMOS , CHMOS หรือ HMOS โดยแต่ละชนิดประเภทจะมีการต่อสัญญาณนาฬิกาโดยตรงที่แตกต่างกันไป จากรูปที่ 2.2(ก) จะเป็นการต่อสัญญาณนาฬิกาโดยตรงจากภายนอกสำหรับ MCU ชนิดประเภท CMOS ,CHMOS โดยวิธีการต่อสัญญาณนาฬิกาจากภายนอกวิธีนี้ออกจะยุ่งยาก และต้องระมัดระวังในการต่อใช้งาน ผู้เขียนจึงขอแนะนำให้ใช้วิธีที่ 2 ดังแสดงในรูปที่ 2.2(ข) เป็นการใช้ X-TAL ในการกำเนิดสัญญาณนาฬิกาให้กับ MCU ซึ่งจะมีความแน่นนอนและมีวงจรการใช้งานที่มีความซับซ้อนน้อยกว่าวิธีแรก แต่ขอให้ผู้อ่านคำนึงถึงการเลือกใช้ความถี่ของ X-TAL ให้เหมาะสมกับประเภทและชนิดของงานด้วยเพราะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X-TAL นั้นยิ่งมีความถี่มากก็จะเป็นตัวกำเนิดสัญญาณรบกวนมากเช่นกัน (การแก้ไขสัญญาณรบกวนในเบื้องต้นนั้นทำให้การ ชีลด์ (Shield) หรือ ต่อตัวถัง X-TAL ลงกราวด์ก็จะแก้ปัญญาสัญญาณรบกวนได้บ้าง)



รูปที่2.2 (ก) แสดงการต่อสัญญาณนาฬิกาโดยตรงจากภายนอก
(ข) แสดงการใช้ X-TAL ในการกำเนิดสัญญาณนาฬิกาให้กับ MCU

สัญญาณนาฬิกาจะเป็นตัวควบคุมการทำงาน การประมวลผลคำสั่งต่างๆของ MCU โดยการประมวลผลในคำสั่งต่างๆของ MCS-51 จะทำงานเป็นรอบวัฏจักรของภาษาเครื่อง หรือที่เรียกว่า Macmine Cycle โดย Machine Cycle จะใช้ช่วงเวลาในการทำงานเท่ากับคาบเวลาของสัญญาณนาฬิกาจำนวน 12 ลูก (12 Oscillator period) จากคุณสมบัติข้างต้นเราสามารถ คำนวณเวลาการทำงานใน 1 Machine Cycle ของ MCS-51 จากความถี่ของ X-TAL ที่เลือกนำมาใช้งานได้จากสมการ

$$1 \text{ Machine Cycle} = 12 / \text{X-TAL oscillator} \quad \text{(Second)}$$

โดยเมื่อเรารู้ระยะเวลาในการทำงานใน 1 Machine Clcle ของ CPU แล้วจะช่วยให้สามารถประมาณช่วงเวลาในการทำงานของคำสั่งแต่ละคำสั่งในการเขียนโปรแกรมได้ ซึ่งคำสั่งแต่ละคำสั่งจะถูกกำหนดการทำงานเป็น Machine Cycle เช่นกัน โดยรายละเอียดการทำงานของคำสั่งในรูป Machine Cycle หรือ Oscillator Period สามารถดูได้จากชุดคำสั่งไมโครคอนโทรลเลอร์ MCS-51 ในหัวข้อ ที่ 2.1.6

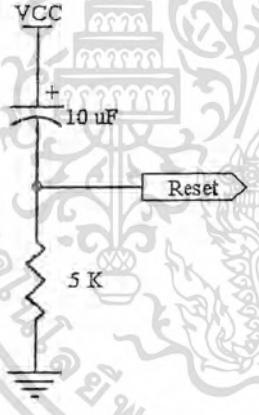
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากที่ผ่านมาคงจะพอเป็นที่รู้จัก MCU กันพอสมควรแล้ว แต่สิ่งที่ต้องคำนึงถึงอีกประการหนึ่งในการนำ MCU ไปใช้งานนั่นคือ ขั้นตอนในการรีเซตตัว MCU เพื่อให้ MCU ทำงานได้ถูกต้อง ไม่เกิดความผิดพลาดหรือ เมื่อเกิดสิ่งผิดปกติก็สามารถทำการรีเซต MCU ให้เริ่มต้นการทำงานต่อไปได้

2.1.5 Reset Operation

ในการรีเซต ไมโครคอนโทรลเลอร์ MCS-51 นั้นจะต้องทำให้สภาวะที่ขา RST ของตัว MCU อยู่ในสภาวะ Hight ที่ค่าเวลาที่เหมาะสมโดยครอบคลุมช่วงเวลา 2 Machine Cycle ของการทำงานเป็นอย่างน้อย (ทำนองว่าเกินได้แต่ต่ำกว่าไม่ได้) แล้วกลับสู่สภาวะ Low โดย Timing Diagram ต่างๆถ้าผู้อ่านสนใจก็ศึกษาได้จาก Data Sheet ของ MCU นั้นๆซึ่งการรีเซต MCU ที่นิยมใช้มีด้วยกัน 2 วิธี วิธีแรกก็คือ การใช้อุปกรณ์ R , C ในการรีเซต MCU แต่วิธีนี้ดูจะล้าสมัยไปหน่อย เพราะโดยส่วนมากจะนิยมใช้วิธีที่ 2 คือ การใช้อุปกรณ์ Semiconductor หรือ IC ในการรีเซต MCU ซึ่งจะมีความแน่นอน และใช้งานง่ายกว่าวิธีแรกมาก

วิธีที่1 R , C Reset

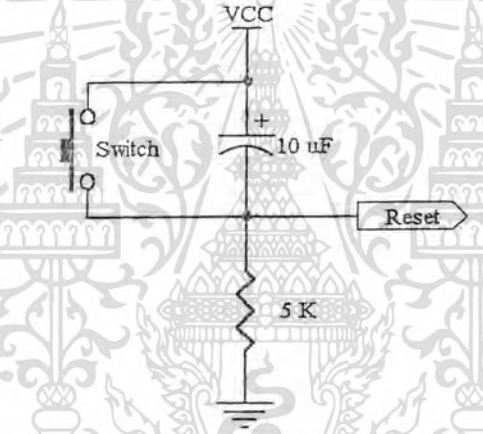


รูปที่ 2.3 วงจรรีเซตอัตโนมัติ

จากรูปที่ 2.3 นั้นจะเป็นการต่อวงจร R , C ใช้ในการรีเซตอัตโนมัติให้กับ MCS-51 โดยการทำงานของวงจรนั้น เมื่อมีการจ่ายไฟเลี้ยงให้กับระบบ วงจรจะกำเนิดสัญญาณ Active High เพื่อทำการรีเซต MCS-51 โดยอัตโนมัติ ท่านผู้อ่านคงจะสงสัยว่า จากวงจรนั้นขา RST ของ MCU ต่อผ่าน R-5K ลงกราวด์ ทั้งยังมี C ต่อกั้นระหว่างขา RST และ Vcc อีก จะทำให้ขา RST ของ MCU มีสภาวะ High ได้อย่างไร ด้วยคุณสมบัติของ C ก็จะทำหน้าที่ Block ไฟกระแสตรงไม่ให้ผ่านตัวมันอยู่แล้ว ฉะนั้นเพื่อแก้ข้อสงสัยจากวงจรในรูปที่ 1-3 ขา RST ของตัว MCU ก็น่าจะมีสภาวะ Low ตลอดเวลาฉะนั้นผู้เขียนจะขอจะขออธิบายการทำงานเบื้องต้นของวงจรดังนี้

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.3 ที่สภาวะปกติยังไม่จ่ายไฟเลี้ยงให้กับระบบ การต่อวงจรดังรูปจะทำให้ C รักษาสภาวะเป็นกลาง คือ C จะถูก Discharge ประจุออกจนหมด จากนั้นเมื่อเริ่มจ่ายไฟให้กับระบบ จะทำให้มีกระแสไหลผ่าน C ได้ในช่วงเวลาหนึ่ง (Initial State) โดยขึ้นอยู่กับค่าความจุของ C นั่นคือ ยิ่ง C มีค่ามาก ช่วงเวลาในการไหลของกระแสผ่าน C ก็จะยิ่งมีค่ามาก (เป็นคุณสมบัติของ C) ฉะนั้น ขณะที่มีกระแสไหลผ่าน C กระแสก็จะไหลผ่าน R ซึ่งก็จะทำให้เกิดแรงดันตกคร่อม R ขึ้น เป็นผลให้สภาวะที่ขารีเซตของ MCU เปลี่ยนสภาวะจาก High เป็น Low ในที่สุดเสมือนเป็นการส่งสัญญาณ Active high ให้กับขา RST ของ MCU ทำให้ MCU กลับมาเริ่มต้นทำงานต่อไป โดยจากวิธีนี้จะเห็นว่าสภาวะ High ของขา RS นั้นขึ้นอยู่กับคุณสมบัติของ C และ R มีค่าที่ไม่สัมพันธ์กัน ก็รีเซต โดยเลือกใช้ค่า R และ C ค่าต่างๆ พบว่าค่า R และ C ที่เหมาะสมในการรีเซต MCU จะใช้ค่า C ประมาณ 10 uF ชนิดอิเล็คโตรไลต์ และ R ค่าประมาณ 5-100 KΩ ¼ W 5 %



รูปที่ 2.4 วงจรรีเซตอัตโนมัติพร้อม Switch รีเซต

จากวงจรในรูปที่ 2.4 นั้นยังจัดอยู่ในรูปแบบการรีเซตแบบเดียวกับวิธีแรก แต่จะต่อสวิตช์ตามวงจรเพื่อทำการรีเซต MCU แบบ Manual Reset กรณี MCU เกิดการทำงานผิดพลาดในระบบแล้วไม่ต้องการเปิด / ปิด การจ่ายไฟให้แก่ระบบใหม่

จากวงจรในรูปที่ 2.4 เมื่อกำหนดการกดสวิตช์จะทำให้มีแรงดันตกคร่อม R จึงเกิดสภาวะ High ขึ้นที่ขารีเซต โดยระยะเวลาก็จะขึ้นกับช่วงเวลาการกดสวิตช์นั่นเอง ถ้าใช้วิธีนี้คงจะให้ความแน่นอนน้อยกว่าวิธีแรกแต่ก็สามารถรีเซต MCU ได้เช่นกันจากที่กล่าวมาแล้วสภาวะ High ที่ขารีเซต นั้นอาจจะนานเกินได้แต่ห้ามน้อยกว่าช่วงเวลาการรีเซต นั่นก็คือการกดสวิตช์ด้วยมือนั้นช่วงเวลาจะเกินช่วงเวลาที่ทำให้ MCU รู้ว่าถูกรีเซตแน่นอน จึงไม่มีปัญหาอะไรสำหรับวงจรนี้ แต่จากการทดลองใช้การรีเซตแบบ Manual Reset ผลปรากฏว่าไม่สามารถรีเซต MCU ได้เป็นบางครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

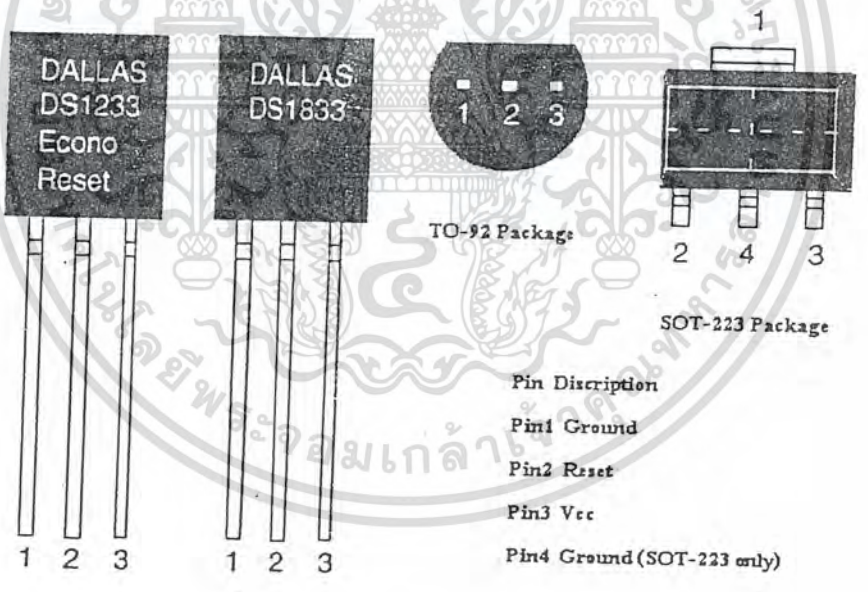
ทั้งนี้น่าจะขึ้นอยู่กับชนิดและคุณสมบัติของสวิตช์ที่นำมาใช้ การกระเด้งของหน้าสัมผัสความแรงของหน้าสัมผัส (Bounce) เป็นต้น

วิธีที่2 IC Reset

มาถึงวิธีที่ 2 ของการรีเซตนี้เป็นวิธีที่สะดวก ใช้งานง่าย และมีประสิทธิภาพมากกว่าวิธีที่ 1 เนื่องจากมีการนำ IC สำหรับงานรีเซต MCU มาใช้งานโดยเฉพาะ คือ IC ของบริษัท Dallus Semiconductor เบอร์ DS 1233D และเบอร์ DS 1833 ซึ่งทั้งสองเบอร์นั้นเป็น IC สำหรับรีเซต MCU ที่ใช้กับแหล่งจ่ายไฟ +5 Vdc โดย DS 1233D จะให้สัญญาณรีเซต Low และ DS 1833 จะให้สัญญาณรีเซตเป็น High

คุณสมบัติและโครงสร้าง

- รีเซตอัตโนมัติให้กับ MCU และ MCU เมื่อมีการจ่ายไฟให้กับระบบ (Power On)
- สัญญาณรีเซตมีรูปแบบเป็น Active Low ในเบอร์ DS 1233D โดยมีช่วงเวลา 350 ms
- ใช้ระบบที่มีแหล่งจ่ายไฟ +5 Vdc
- มีราคาถูก อยู่ในรูปตัวถังทั้งแบบ TO – 92 และ SOT – 223

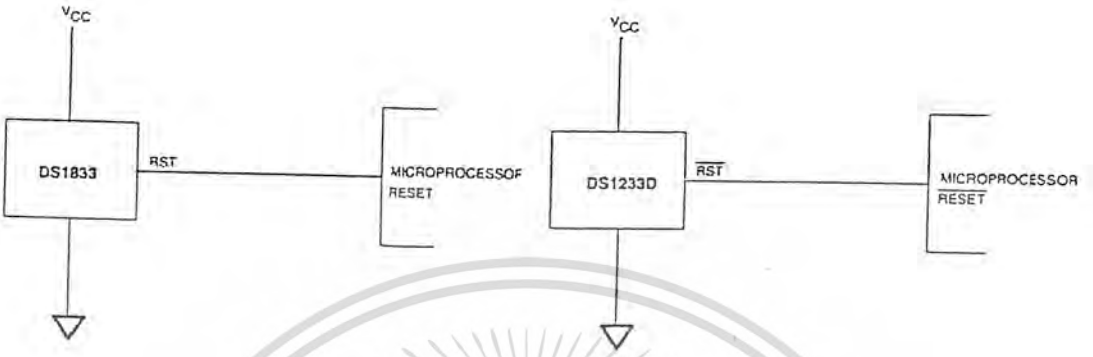


รูปที่ 2.5 แสดงคุณลักษณะของ IC รีเซต

จากรูปที่ 2.5 นั้นจะแสดงลักษณะภายนอกของ IC รีเซตทั้งสองเบอร์ โดยส่วนมากในการใช้งานนั้นมักจะนิยมใช้ในรูปของตัวถัง TO – 92 มากกว่าเพราะจะเป็นในลักษณะของ Bottom Layer PCB ทั้งยังหาซื้อมาใช้งานได้ง่าย ส่วนในตัวถัง SOT – 223 นั้นจะเป็นในลักษณะของอุปกรณ์ติดบนผิวหน้า PCB (Surface Mount Device, SMD) หรือเป็นในลักษณะของ Top Layer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PCB ซึ่งจะผลิตเพื่อใช้ในทางอุตสาหกรรมเป็นหลักจึงจัดหามาใช้งานได้ยาก และการใช้งานก็ยาก กว่าในกรณีตัวถัง TO – 92



รูปที่ 2.6 การต่อใช้งาน IC ในการรีเซต MPU , MCU

เมื่อต่อวงจรใช้งาน IC รีเซต ตามรูปที่ 2.6 การทำงานของวงจรจะเริ่มทำงานโดยเมื่อเริ่มต้นมีการจ่ายไฟเลี้ยงให้กับระบบ (ซึ่งระบบที่ให้สัญญาณรีเซตต่างๆ มีความแน่นอน จะเป็นระบบที่ใช้แหล่งจ่ายไฟ +5 Vdc) จะทำให้มีสัญญาณรีเซตออกมาที่ขารีเซตในลักษณะ Active Low หรือ Active High เป็นพัลส์ระยะเวลา 350 ms โดยชนิดของสัญญาณรีเซตที่กล่าวมาก็จะขึ้นอยู่กับเบอร์ของ IC ที่ใช้ ซึ่ง IC เบอร์ DS 1233D จะมีสัญญาณรีเซตเป็น Active Low และ เบอร์ DS 1833 จะมีสัญญาณรีเซตเป็น Active High (สำหรับใช้รีเซตให้กับไมโครคอนโทรลเลอร์ MCS-51) ดังนั้นสัญญาณที่ขารีเซตก็จะกลับสู่สภาวะปกติ โดยผลของการรีเซตไม่ว่าจะด้วยวิธีการใดก็จะทำให้หน่วยประมวลผลภายใน MCU เริ่มต้นการทำงานที่ตำแหน่ง 0000H ของหน่วยความจำนั่นเอง

2.1.6 ชุดคำสั่งในการควบคุมไมโครคอนโทรลเลอร์ MCS-51

ARITHMETIC OPERATIONS

Mnemonic	Description
ADD A,Rn	บวกค่าของรีจีสเตอร์ใดๆ กับแอกคิวมูเลเตอร์
ADD A,direct	บวกค่าภายในหน่วยความจำกับแอกคิวมูเลเตอร์
ADD A,@Ri	บวกค่าที่ชี้โดยรีจีสเตอร์กับแอกคิวมูเลเตอร์
ADD A,#data	บวกค่าคงที่กับแอคคิวมูเลเตอร์
ADDC A,Rn	บวกค่าของรีจีสเตอร์กับแอคคิวมูเลเตอร์รวมแฟลก Carry
ADDC A,direct	บวกค่าในหน่วยความจำกับแอคคิวมูเลเตอร์รวมแฟลก Carry

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ADDC A,@Ri	บวกค่าที่ชี้โดยรีจีสเตอร์กับแอคคิวมูเลเตอร์รวมแฟลก Carry
ADDC A,#data	บวกค่าคงที่กับแอคคิวมูเลเตอร์รวมแฟลก Carry
SUBB A,Rn	ลบค่าของรีจีสเตอร์กับแอคคิวมูเลเตอร์และแฟลก Barow
SUBB A,direct	ลบค่าในหน่วยความจำกับแอคคิวมูเลเตอร์และแฟลก Barow
SUBB A,@Ri	ลบค่าที่ชี้โดยรีจีสเตอร์กับแอคคิวมูเลเตอร์และแฟลก Barow
SUBB A,#data	ลบค่าคงที่กับแอคคิวมูเลเตอร์และแฟลก Barow
INC A	เพิ่มค่าภายในแอกคิวมูเลเตอร์เพิ่มขึ้นอีกหนึ่ง
INC Rn	เพิ่มค่าภายในรีจีสเตอร์เพิ่มขึ้นอีกหนึ่ง
INC direct	เพิ่มค่าให้กับค่าในตำแหน่งหน่วยความจำขึ้นอีกหนึ่ง
INC @Ri	เพิ่มค่าให้กับค่าในตำแหน่งหน่วยความจำที่ชี้โดยรีจีสตอร์ขึ้นอีกหนึ่ง
INC DPTR	เพิ่มค่าให้กับรีจีสเตอร์ DPTR ขึ้นอีกหนึ่ง
DEC A	ลดค่าภายในแอคคิวมูเลเตอร์ลงอีกหนึ่ง
Mnemonic	**Description**
DEC Rn	ลดค่าภายในรีจีสเตอร์ลงอีกหนึ่ง
DEC dierect	ลดค่าให้กับค่าในตำแหน่งหน่วยความจำลงอีกหนึ่ง
DEC @Ri	ลดค่าให้กับค่าในตำแหน่งหน่วยความจำที่ชี้โดยรีจีสเตอร์ลงอีกหนึ่ง
MUL AB	คูณค่าในแอกคิวมูเลเตอร์กับรีจีสเตอร์ B
DIV AB	หารค่าในแอกคิวมูเลเตอร์กับรีจีสเตอร์ B
DA A	แปลงค่าในแอกคิวมูเลเตอร์ให้เป็นเลขฐานสิบ
ANL A,Rn	AND ค่าในรีจีสเตอร์กับแอกคิวมูเลเตอร์
ANL A,direct	AND ค่าในหน่วยความจำกับแอกคิวมูเลเตอร์
ANL A,@Ri	AND ค่าที่ชี้โดยรีจีสเตอร์กับแอกคิวมูเลเตอร์
ANL A,#data	AND ค่าคงที่กับแอกคิวมูเลเตอร์
ANL direct,A	AND ค่าภายในแอกคิวมูเลเตอร์กับค่าในหน่วยความจำ
ANL direct,#data	AND ค่าคงที่กับค่าในหน่วยความจำ
ORL A,Rn	OR ค่าในรีจีสเตอร์กับแอกคิวมูเลเตอร์
ORL A,direct	OR ค่าในหน่วยความจำกับแอกคิวมูเลเตอร์
ORL A,@Ri	OR ค่าที่ชี้โดยรีจีสเตอร์กับแอกคิวมูเลเตอร์
ORL A,#data	OR ค่าคงที่กับแอกคิวมูเลเตอร์
ORL direct,A	OR ค่าภายในแอกคิวมูเลเตอร์กับค่าในหน่วยความจำ
ORL direct,data	OR ค่าคงที่กับค่าในหน่วยความจำ
XRL A,Rn	EX-OR ค่าในรีจีสเตอร์กับแอกคิวมูเลเตอร์

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

XRL A,direct	EX-OR ค่าในหน่วยความจำกับแอกคิวมูเลเตอร์
XRL A,Ri	EX-OR ค่าที่ชี้โดยรีจีสเตอร์กับแอกคิวมูเลเตอร์
XRL A,#data	EX-OR ค่าคงที่กับแอกคิวมูเลเตอร์
XRL direct,A	EX-OR ค่าภายในแอกคิวมูเลเตอร์กับค่าในหน่วยความจำ
XRL direct,data	EX-OR ค่าคงที่กับค่าในหน่วยความจำ
CLR A	เคลียร์ค่าในแอกคิวมูเลเตอร์
CPL A	คอมพลีเมนต์ค่าในแอกคิวมูเลเตอร์
RL A	เลื่อนบิตในแอกคิวมูเลเตอร์ไปทางซ้ายหนึ่งบิต
RLC A	เลื่อนบิตในแอกคิวมูเลเตอร์ไปทางซ้ายหนึ่งบิตรวมแฟลก Carry
RR A	เลื่อนบิตในแอกคิวมูเลเตอร์ไปทางขวาหนึ่งบิต
RRC A	เลื่อนบิตในแอกคิวมูเลเตอร์ไปทางขวาหนึ่งบิตรวมกับแฟลก Carry
SWAP A	สลับค่า 4 บิตบนกับ 4 บิตล่าภายในแอกคิวมูเลเตอร์
DATA TRANSTERT	
Mnemonic	Description
MOV A,Rn	นำค่าภายในรีจีสเตอร์เก็บไว้ในแอกคิวมูเลเตอร์
MOV A,direct	นำค่าภายในหน่วยความจำเก็บไว้ในแอกคิวมูเลเตอร์
MOV A,@Ri	นำค่าที่ชี้โดยรีจีสเตอร์เก็บไว้ในแอกคิวมูเลเตอร์
MOV A,#data	นำค่าคงที่เก็บไว้ในแอกคิวมูเลเตอร์
MOV Rn,A	นำค่าภายในแอกคิวมูเลเตอร์ไปเก็บไว้ในรีจีสเตอร์
MOV Rn,direct	นำค่าภายในหน่วยความจำไปเก็บไว้ในรีจีสเตอร์
MOV Rn,#data	นำค่าคงที่เก็บไว้ในรีจีสเตอร์
MOV direct,A	นำค่าภายในแอกคิวมูเลเตอร์ไปเก็บไว้ในหน่วยความจำ
MOV direct,Rn	นำค่าภายในรีจีสเตอร์ไปเก็บไว้ในหน่วยความจำ
MOV direct,direct	นำค่าภายในหน่วยความจำไปเก็บไว้ในหน่วยความจำ
MOV direct,@Ri	นำที่ชี้โดยรีจีสเตอร์ไปเก็บไว้ในหน่วยความจำ
MOV direct,#data	นำค่าคงที่ไปเก็บไว้ในหน่วยความจำ
MOV @Ri,A	นำค่าภายในแอกคิวมูเลเตอร์เก็บไว้ในตำแหน่งหน่วยความจำที่ชี้โดยรีจีสเตอร์
MOV @Ri,direct	นำค่าภายในหน่วยความจำเก็บไว้ในตำแหน่งหน่วยความจำที่ชี้โดยรีจีสเตอร์
MOV @Ri,#data	นำค่าคงที่ไปเก็บไว้ในตำแหน่งหน่วยความจำที่ชี้โดยรีจีสเตอร์
MOV DPTR,#data16	นำค่าคงที่ขนาด 16 เก็บไว้ในรีจีสเตอร์ DPTR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MOVC A,@A+DPTR	นำค่าที่เป็นตำแหน่งผลรวมที่ชี้โดยแอกคิวมูเลเตอร์กับรีจีสเตอร์ DPTR ไปเก็บไว้ในแอกคิวมูเลเตอร์
MOV A,@A+PC	นำค่าที่เป็นตำแหน่งผลรวมที่ชี้โดยแอกคิวมูเลเตอร์กับรีจีเตอร์ PC ไปเก็บไว้ในแอกคิวมูเลเตอร์
MOVX A,@Ri	นำค่าในหน่วยความจำภายนอกที่ชี้โดยรีจีสเตอร์ (8บิต) มาเก็บไว้ในแอกคิวมูเลเตอร์
MOVX A,@DPTR	นำค่าในหน่วยความจำภายนอกที่ชี้โดย DPTR (16 บิต) มาเก็บไว้ในแอกคิวมูเลเตอร์
MOVX @Ri,A	นำค่าในแอกคิวมูเลเตอร์ไปเก็บไว้ในหน่วยความจำภายนอกที่ชี้โดยรีจีสเตอร์
MOVX @DPTR,A	นำค่าภายในแอกคิวมูเลเตอร์ไปเก็บไว้ในหน่วยความจำภายนอกที่ชี้โดยรีจีสเตอร์ DPTR
Mnemonic	**Description**
PUSH direct	PUSH ค่าภายในหน่วยความจำลงสแต็ก
POP direct	POP ค่าภายในหน่วยความจำจากในสแต็ก
XCH A,Rn	สลับข้อมูลแอกคิวมูเลเตอร์กับรีจีสเตอร์
XCH A,direct	สลับข้อมูลแอกคิวมอเลเตอร์กับค่าในหน่วยความจำ
XCH A,@Ri	สลับข้อมูลแอกคิวมูเลเตอร์กับค่าที่ชี้โดยรีจีสเตอร์
XCHD A,@Ri	สลับข้อมูล 4 บิตล่างของแอกคิวมูเลเตอร์กับค่าที่ชี้โดยรีจีสเตอร์
BOOLEAN VARIABLE MANIPULATION	
CLR C	เคลียร์แฟลก Carry
CLR bit	เคลียร์ค่าบิตในหน่วยความจำภายในที่อ้างถึงในระดับบิต
SETB C	เซตแฟลก Carry
SETB bit	เซตค่าบิตในหน่วยความจำภายในที่อ้างถึงได้ระดับบิต
CPL C	คอมพลีเมนต์แฟลก Carry
CPL bit	คอมพลีเมนต์ค่าบิตในหน่วยความจำภายในที่อ้างถึงในระดับบิต
ANL C,bit	AND ค่าบิตในหน่วยความจำภายในที่อ้างถึงในระดับบิตกับแฟลก Carry
ANL C,/bit	AND ค่าคอมพลีเมนต์ของบิตในหน่วยความจำภายในที่อ้างถึงระดับบิตแฟลก Carry
ORL C,bit	OR ค่าบิตในหน่วยความจำภายในที่อ้างถึงในระดับบิตกับแฟลก Carry
ORL C,/bit	OR ค่าคอมพลีเมนต์ของบิตในหน่วยความจำภายในที่อ้างถึงระดับบิตแฟลก Carry

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MOV C,bit	เคลื่อนย้ายบิตในหน่วยความจำภายในที่อ้างถึงได้ระดับบิตไปยังแฟลก Carry
MOV bit,C	เคลื่อนย้ายแฟลก Carry ไปยังบิตในหน่วยความจำภายในที่อ้างถึงได้ระดับบิต
JC rel	กระโดดเมื่อแฟลก Carry ถูกเซตบิต
JNC rel	กระโดดเมื่อแฟลก Carry ไม่ถูกเซตบิต
JB bit,rel	กระโดดเมื่อบิตในหน่วยความจำที่อ้างถึงได้ระดับบิตถูกเซต
JNB bit,rel	กระโดดเมื่อบิตในหน่วยความจำที่อ้างถึงได้ระดับบิตไม่ถูกเซต
JBC bit,rel	กระโดดเมื่อบิตในหน่วยความจำที่อ้างถึงได้ระดับบิตถูกเซตและเคลียร์บิตนั้น

PROGRAM BRANCHING

Mnemonic	Description
ACLL addr11	เรียกใช้ซับรูทีนโดยอ้างแอดเดรส 11 บิต
ACLL addr16	เรียกใช้ซับรูทีนโดยอ้างแอดเดรส 16 บิต
RET	สิ้นสุดการทำงานของซับรูทีน
RETI	สิ้นสุดการทำงานของอินเตอร์รัปต์
ALMP addr11	กระโดดไปตำแหน่งในหน่วยความจำโดยอ้างแอดเดรส 11 บิต
ALMP addr16	กระโดดไปตำแหน่งในหน่วยความจำโดยอ้างแอดเดรส 16 บิต
SLMP rel	กระโดดไปยังตำแหน่งในหน่วยความจำโดยใช้ค่าสัมพัทธ์
JMP @A+DPTR	กระโดดไปตำแหน่งที่ชี้โดยผลรวมของแอกคิวมูเลเตอร์กับรีจีสเตอร์ DPTR
JZ rel	กระโดดเมื่อแอกคิวมูเลเตอร์มีค่าเป็นศูนย์
JNZ rel	กระโดดเมื่อแอกคิวมูเลเตอร์มีค่าไม่เป็นศูนย์
CJNE A,direct,rel	กระโดดไปยังตำแหน่งค่าสัมพัทธ์ เมื่อค่าในแอกคิวมูเลเตอร์ไม่เท่ากับค่าในหน่วยความจำ
CJNE A,data,rel	กระโดดไปยังตำแหน่งค่าสัมพัทธ์ เมื่อค่าในแอกคิวมูเลเตอร์ไม่เท่ากับค่าคงที่
CJNE @Ri,#data,rel	กระโดดไปยังตำแหน่งค่าสัมพัทธ์ เมื่อค่าในหน่วยความจำที่ชี้โดยรีจีสเตอร์ไม่เท่ากับค่าคงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

DJNZ direct,rel	ลดค่าภายในหน่วยความจำลงอีกหนึ่ง ถ้าไม่เท่ากับศูนย์จะทำการกระโดดไปยังตำแหน่งค่าสัมพัทธ์
NOP	คำสั่ง No operation

หมายเหตุ

Rn	รีจิสเตอร์ R0-R7 ของแต่ละแบงค์ที่ถูกเลือกใช้งาน
Direct	ตำแหน่งในหน่วยความจำมีขนาด 8 บิต
@Ri	ค่าตำแหน่งที่ในหน่วยความจำมีขนาด 8 บิตที่อ้างถึงโดยรีจิสเตอร์ R0 และ R1 (Indirect Addressing)
#data	ค่าคงที่ขนาด 8 บิต
#data 16	ค่าคงที่ขนาด 16 บิต
addr16	ตำแหน่งแอดเดรสที่อ้างถึงได้ 16 บิต ในคำสั่ง LCALL และ LJMP
addr11	ตำแหน่งแอดเดรสที่อ้างถึงได้ 11 บิต ในคำสั่ง ACALL และ AJMP

Mnemonic	Description
rel	ค่าตำแหน่งในการกระโดดด้วยคำสั่ง SJMP ซึ่งมีค่าอยู่ระหว่างช่วงตำแหน่ง -128 ถึง +127 ไบต์
Bit	บิตของหน่วยความจำภายในที่อ้างถึงได้ระดับบิต (Direct Addressing) หรือรีจิสเตอร์ SFR

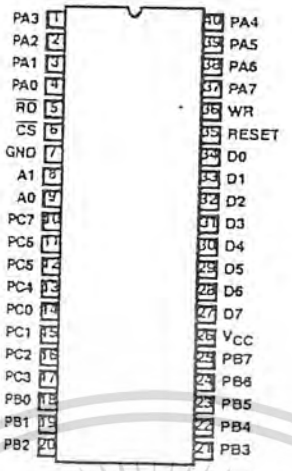
2.2 8255 Programmable Peripheral Interface

IC อีกตัวหนึ่งเป็นที่รู้จักกันดีและนิยมนำมาใช้งานร่วมกับ ไมโครคอนโทรลเลอร์ MCS-51 ในลักษณะงานควบคุมระบบที่พอร์ต I/O ของ MCU ไม่เพียงพอต่อการใช้งานนั้น IC 8255 PPI ตัวนี้เป็น IC ที่ผลิตมาโดยมีคุณสมบัติในการควบคุมการทำงานที่ง่ายและมีจำนวนพอร์ต I/O ขนาด 8 บิตจำนวน 3 พอร์ตภายในตัวจึงเป็น IC ที่น่าสนใจที่จะนำมาใช้งานในการเพิ่มขยายพอร์ต I/O ให้กับ MCU โดนรูปแบบโครงสร้างคุณสมบัติต่างๆ มีดังนี้

- เป็นชิปขนาด 40 ขา
- มีพอร์ตใช้งาน 3 พอร์ต แบ่งออกเป็นพอร์ต A,B และ C
- พอร์ตใช้งานมีลักษณะเป็นพอร์ตแลีช (Latch) คงสภาวะ
- ควบคุมการทำงานพอร์ตด้วยขา A0 และ A1
- สามารถโปรแกรมเลือกลักษณะการใช้งานพอร์ตได้ ทั้งในลักษณะ Input , Output หรือ Bi Directional พอร์ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

45822



รูปที่ 2.7 แสดงลักษณะการจัดขา 8255 PPI

คุณสมบัติขาสัญญาณต่างๆ ของ 8255

Vcc	เป็นขารับไฟเลี้ยงของระบบ +5 Vdc
GND	เป็นขาต่อกราวด์ของระบบ
D0-D7	เป็นขา Bi – Directional Bus โดยใช้เป็น Data Bus ของระบบ
RESET	เมื่อขานี้มีสภาวะ High จะทำให้ 8255 ถูกรีเซตมีผลทำให้ทุกพอร์ตเปลี่ยนเป็นอินพุตพอร์ตทันที หรือในลักษณะอินพุตโหมด
CS	(Chip Select) เป็นขาที่ใช้เป็นตัวควบคุมเลือกจังหวะการทำงานของ 8255 โดย 8255 จะทำงานเมื่อขานี้อยู่ในสภาวะ low
RD	(Read) เมื่อขานี้มีสภาวะ Low MCU จะทำการอ่านข้อมูลผ่านทาง Data Bus ของ 8255 ได้โดยพร้อมกับขา CS ต้องมีสภาวะ low
WR	(Write) เมื่อขานี้มีสภาวะ Low 8255 จะทำการรับข้อมูลหรือ Control Word ที่ส่งมาจาก MCU ได้โดยพร้อมกับขา CS ต้องมีสภาวะ Loc
A0-A1	ใช้ในการรับสัญญาณควบคุมหรือเลือกการใช้งาน พอร์ต A,B และ C ของ 8255
PA0-PA7	เป็นพอร์ต I/O ขนาน 8 บิต
PB0-PB7	เป็นพอร์ต ณฝฯ ขนาน 8 บิต
PC0-PC7	เป็นพอร์ต I/O ขนาน 8 บิตและมีคุณสมบัติแตกต่างจากพอร์ต A และ คือสามารถแบ่งการใช้งานเป็น 4 บิตบน (C4-C7) และ 4 บิตล่าง (C0-C3) ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.3 ตารางความจริงของ 8255

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
1	1	0	1	0	Control Word → Data Bus
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					DISABLE FUNCTION
X	X	X	X	1	Data Bus → Three-State
X	X	1	1	0	Data Bus → Three-State

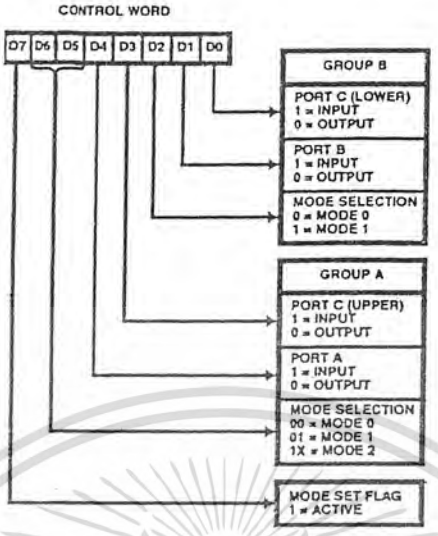
8255 นั้นสามารถที่จะเลือกโหมดการใช้งานได้จากคำสั่งทาง Software จาก MCU ผ่านทาง Data Bus D0-D7 ของ 8255 โดยโหมดการใช้งาน 8255 จะมีด้วยกัน 3 โหมด คือ โหมด 0 (Basic I/O) จะเป็นลักษณะการใช้งานพอร์ต A,B และ C ของ 8255 เป็น I/O ปกติ

โหมด 1 (Strobed I/O) จะมีลักษณะการตรวจสอบสัญญาณ Strobe ก่อนที่จะทำงาน

โหมด 2 (Bi-directional Bus) จะใช้งานพอร์ตในลักษณะ I/O แบบ 2 ทิศทาง (รับส่งข้อมูล)

โดยข้อมูลที่ถูกส่งไปควบคุมการทำงานของ 8255 ผ่านทาง Data Bus D0-D7 นั้นจะถูกเรียกว่า Control Word หรือ Control Code โดยมีรูปแบบหรือคำสั่งอยู่ในแต่ละบิตของชุด Control Word ที่ส่งไปควบคุม 8255 ดังรูป 2.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 แสดงลักษณะการจัด Control Word

ต่อไปนี้ผู้เขียนจะกล่าวถึงการใช้งาน 8255 PPI ในระดับต้นหรือการใช้งาน 8255 ในโหมด 0 เพียงเท่านั้น เนื่องจากการประยุกต์ใช้งาน MCU ร่วมกับ 8255 PPI ทั่วๆไปนั้น จะเป็นการประยุกต์ใช้งาน 8255 ในลักษณะ Basic I/O ให้กับ MCU ซึ่งเป็นส่วนของการใช้งาน 8255 ในโหมด 0 นั่นเอง และงานโดยมากในเบื้องต้นการใช้งาน 8255 จะนำมาใช้งานลักษณะ Basic I/O หรือโหมด 0 เป็นส่วนมาก โดนสามารถสรุป Control Word ในการใช้งาน 8255 PPI ในโหมด 0 ดังตารางที่ 1.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4 สรุปการทำงาน 8255 PPI โหมด 0

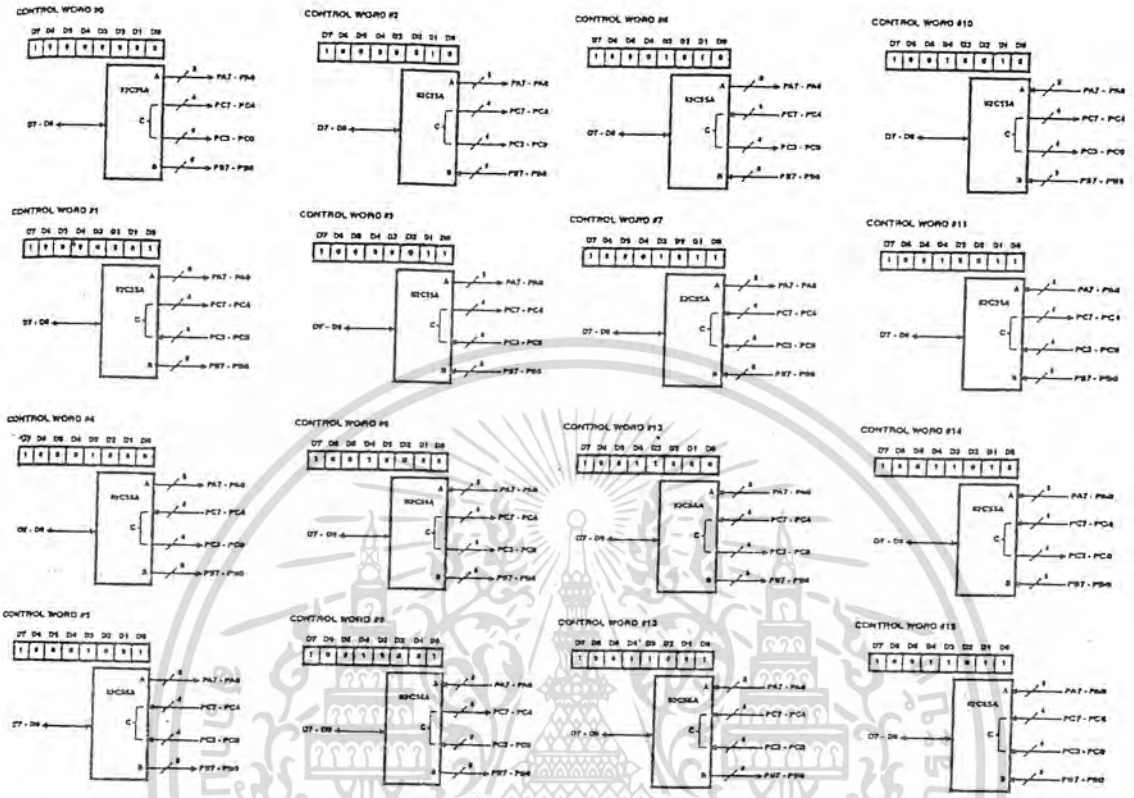
PORTA (PA0-7)	PORTB (PB0-PB7)	PORTC (PC4-PC7)	PORTC (PC0-PC3)	CONTROL WORD(HEX)
OUTPUT	OUTPUT	OUTPUT	OUTPUT	80H
OUTPUT	OUTPUT	OUTPUT	INPUT	81H
OUTPUT	INPUT	OUTPUT	OUTPUT	82H
OUTPUT	INPUT	OUTPUT	INPUT	83H
OUTPUT	OUTPUT	INPUT	OUTPUT	88H
OUTPUT	OUTPUT	INPUT	INPUT	89H
OUTPUT	INPUT	INPUT	OUTPUT	8AH
OUTPUT	INPUT	INPUT	INPUT	8BH
INPUT	OUTPUT	OUTPUT	OUTPUT	90H
INPUT	OUTPUT	OUTPUT	INPUT	91H
INPUT	INPUT	OUTPUT	OUTPUT	92H
INPUT	INPUT	OUTPUT	INPUT	93H
INPUT	OUTPUT	INPUT	OUTPUT	98H
INPUT	OUTPUT	INPUT	INPUT	99H
INPUT	INPUT	INPUT	OUTPUT	9AH
INPUT	INPUT	INPUT	INPUT	9BH

ในการใช้งาน 8255 PPI นั้น ผู้เขียนจะขออธิบายการใช้งานในโหมด 0 หรือการใช้งาน 8255 เป็น Basic I/O Port ในระดับต้นดังต่อไปนี้

เมื่อเริ่มต้นก่อนที่จะใช้งาน 8255 นั้นจำเป็นจะต้อง กำหนดโหมดการใช้งานรวมถึงลักษณะการใช้งาน พอร์ตต่างๆ ของ 8255 ในลักษณะใดด้วย ซึ่งคำสั่งในการกำหนดโหมดของการรวมถึงลักษณะการใช้งานพอร์ตต่างๆของ 8255 ทั้งหมดที่กล่าวมาถูกเรียกว่า Control Word ดังในรูปที่ 2.8 จะแสดงลักษณะการจัด Control Word ในการควบคุม 8255 โดยจะเห็นว่าการใช้งาน 8255 ในโหมด 0 นั้น ชุดคำสั่งของ Control Word ในบิตที่ 2 , 5 และ 6 จะต้องเป็น '0' ส่วนบิตที่เหลือจะเป็นบิตกำหนดการใช้งานพอร์ตของ 8255 ในลักษณะ I/O Port สรุปแล้วก็คือ ชุดคำสั่งหรือ Control Word ที่กล่าวมาได้ถูกสรุปไว้ในตารางที่ 2.4 หรือถ้าจะดูรายละเอียดค่าแต่ละบิตของ Control Word สามารถดูได้จากหัวข้อ 2.2.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 ลักษณะการจัด Control Word Mode 0



ตรานี้ย้อนกลับไปดูที่ตาราง 2.3 เป็นตารางความจริงของ 8255 โดยจะพบว่า A0 และ A1 เป็นตัวกำหนดการใช้งานพอร์ตของ 8255 ว่าจะใช้งานพอร์ตใด ที่นี้ในการที่จะส่ง Control Word ไปยัง 8255 เพื่อให้ 8255 รับรู้ว่าเป็นชุดข้อมูลซึ่งเป็น Control Word ไม่ใช่ชุด Data ธรรมดา จะต้องเซตค่าให้ A0 และ A1 เป็น '1' ทั้งคู่ หลังจากนั้นขา A0 และ A1 ก็จะถูกใช้งานเป็นขาเลือก (Select) ความต้องการใช้พอร์ตของ 8255 ว่าต้องการรับส่งข้อมูลกับพอร์ตใดของ 8255 ต่อไป

สรุปการใช้งาน 8255 ในโหมด 0 อย่างง่ายๆ ออกเป็น 2 ขั้นตอนคือ

1. ทำการเซต A0 และ A1 ของ 8255 ให้เป็น '1' ทั้งคู่แล้วทำการส่ง Control Word 8 บิต ไปยัง 8255 เพื่อเริ่มการใช้งาน
2. เลือกการใช้งานพอร์ตของ 8255 ได้อย่างอิสระ โดยการควบคุมขา A0 และ A1 ของ 8255 ดังตารางที่ 2.4

ที่ผ่านมานั้นเป็นเพียงการกล่าวถึงลักษณะและคุณสมบัติการใช้งาน โดยเบื้องต้นของ 8255 PPI เพียงเท่านั้นซึ่งการนำมาประยุกต์ใช้งาน ก็จะกล่าวเน้นในการใช้งาน 8255 ในโหมด 0 ซึ่งก็คือการใช้งาน 8255 เป็น Basic I/O พอร์ต เพื่อจะนำไปใช้งานร่วมกับ MCU

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในลักษณะช่องเพิ่มเพอร์ตใช้งานให้กับ MCU เพื่อขยายความสามารถของ MCU ให้ใช้งานได้มากยิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
หน่วยความจำของ MCS – 51

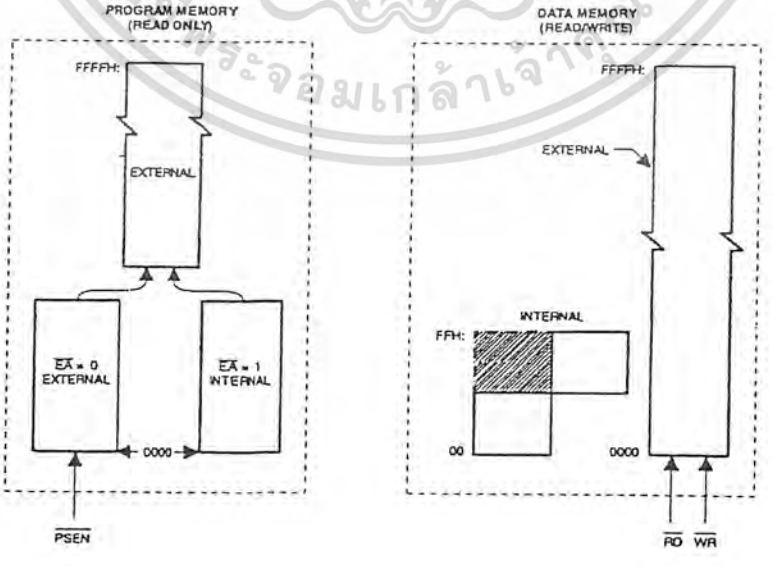
ส่วนประกอบสำคัญภายในตัวไมโครคอนโทรลเลอร์ ที่เรานำมาใช้งานนั้นจะประกอบไปด้วยองค์ประกอบหลักๆ อยู่ 2 ส่วน เพื่อให้ไมโครคอนโทรลเลอร์ทำงานได้ตามความต้องการของเรา ส่วนแรกก็คือส่วนควบคุมการประมวลผลหรือที่เรียกว่า CPU (Central Processing Unit) และส่วนที่สอง คือ ส่วนของหน่วยความจำ โดยหน่วยความจำนั้นก็จะมีการแยกชนิดประเภทตามคุณสมบัติและการใช้งานได้เป็น 2 ชนิดประเภท คือ

1.หน่วยความจำโปรแกรม (Read Only Memorry ,ROM)

2. หน่วยความจำข้อมูล (Random Acceaa Memory ,Ram)

3.1 หน่วยความจำโปรแกรม

การใช้งานไมโครคอนโทรลเลอร์นั้นจำเป็นจะต้องมีส่วนของหน่วยความจำโปรแกรมเข้ามามีส่วนร่วมในการใช้งานเสมอเพื่อให้ตัวไมโครคอนโทรลเลอร์ทำงานตามที่เราต้องการและทำให้งานนั้นสมบูรณ์ยิ่งขึ้นโดยหน่วยความจำประเภทแรกที่จะกล่าวถึงต่อไปนี้ จะเป็นหน่วยความจำประเภท Nonvolatile (สามารถเก็บรักษาข้อมูลเอาไว้ได้ แม้จะหยุดจ่ายไฟเลี้ยงให้ก็ตาม) หน่วยความจำที่กล่าวถึง นี้คือหน่วยความจำโปรแกรมหรือ ROM (Read Only Memory) นั่นเอง โดยเราจะใช้ในการเก็บข้อมูลหลักหรือโปรแกรมควบคุมที่สำคัญเอาไว้ หรือจะเก็บรูปแบบของตัวอักษร สัญลักษณ์ต่างๆ เป็นต้น แต่ที่นิยมใช้โดยมาก จะใช้การเก็บโปรแกรมในการควบคุมและสั่งงานให้กับ CPU ให้ทำงานตามที่โปรแกรมอาไว้



รูปที่ 3.1 แสดงลักษณะการจัดตำแหน่งการใช้งานหน่วยความจำ MCS – 51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.1 ชนิดประเภทของ ROM

- ROM (Read Only Memory) ประเภทนี้เป็นประเภทที่ถูกโปรแกรมมาแล้วจากโรงงาน ไม่สามารถลบแล้วโปรแกรมใหม่ได้โดยจะเป็นประเภทที่สั่งให้โรงงานผลิต ขึ้นโดยเฉพาะ
- PROM (Programmable ROM) โดย ROM ประเภทนี้จะทำการโปรแกรมได้ด้วยเครื่องโปรแกรม ROM แต่จะสามารถกระทำได้เพียงครั้งเดียว และไม่สามารถลบแล้วโปรแกรมใหม่ได้
- EPROM (Erasable PROM) โดย ROM ประเภทนี้จะเป็นที่รู้จักและนิยมกันมากในอดีต ซึ่ง ROM ประเภทนี้เป็น ROM ที่สามารถทำการโปรแกรมลงไปแล้วลบออกเพื่อทำการโปรแกรมใหม่ได้ วิธีการลบนั้นกระทำได้โดยการฉายแสงอัลตร้าไวโอเลต (UV) ผ่านแผ่นกระจกของ IC โดยมีช่วงระยะเวลาตามมาตรฐาน เมื่อลบเสร็จแล้วสามารถนำมาโปรแกรมใช้ใหม่ได้อีก แต่จะมีข้อเสียตรงที่ ถ้ากระทำการฉายแสงไม่ได้มาตฐาน จะลบโปรแกรมเก่าออกไม่หมดและจะเกิดการด้านทำให้ ROM ตัวนี้ใช้งานไม่ได้อีก
- EEPROM (Electrical EPROM) ROMประเภทนี้ผลิตออกมาเพื่อแก้ไขข้อเสียของ EPROM ก็คือเสียเวลามากในการลบโปรแกรมโดยกรรมวิธีฉายแสง และเสี่ยงต่อการด้านของ IC ฉะนั้น EEPROM จะเป็น ROM ที่สามารถลบและเขียนใหม่ได้ด้วยวิธีทางการไฟฟ้า ซึ่งให้ความสะดวกและรวดเร็วมากในการใช้งานทั้งยังสามารถลบเขียนใหม่ได้หลายร้อยหลายพันครั้ง มีความแน่นอน และไม่ทำให้ IC เกิดการด้านได้ง่ายๆ
- FLASH MEMORY ล่าสุดได้มีหน่วยความจำที่ใช้เทคโนโลยีใหม่โดยเป็นหน่วยความจำที่สามารถลบได้ทางไฟฟ้าเช่นเดียวกันกับ EEPROM แต่มีความสามารถเข้าถึงข้อมูลได้รวดเร็วกว่า โดยก่อนที่จะทำการโปรแกรมลงภายในชิปทุกครั้งจะต้องทำการลบข้อมูลเดิมในหน่วยความจำทั้งหมดก่อน ซึ่งหน่วยความจำชนิดนี้จะมีอยู่ภายในตัว MCU เป็นส่วนมาก

จากรูปที่ 3.1 นั้นจะเป็นการแสดงลักษณะการจัดตำแหน่งการใช้งานหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 โดยรายละเอียดเบื้องต้นเกี่ยวกับหน่วยความจำของไมโครคอนโทรลเลอร์แต่ละเบอร์นั้นได้กล่าวหรือแสดงโดยสรุปไว้แล้วในตารางที่ 2.1 บทที่2 และจากตารางที่ 3.1 ข้างต้นนั้น ในกรณีเบอร์ 27XX จะเป็น EPROM ถ้าเป็นเบอร์ 28XX คือ EEPROM จากที่กล่าวมาผู้อ่านคงพอเข้าใจตลอดจนรู้จัก ROM ประเภทต่างๆกันบ้างแล้วพอสมควร ทีนี้เมื่อรู้จักหน่วยความจำโปรแกรม (ROM) กันแล้วก็จะมาทำความรู้จักถึงหน่วยความจำข้อมูล (RAM) กันต่อไป

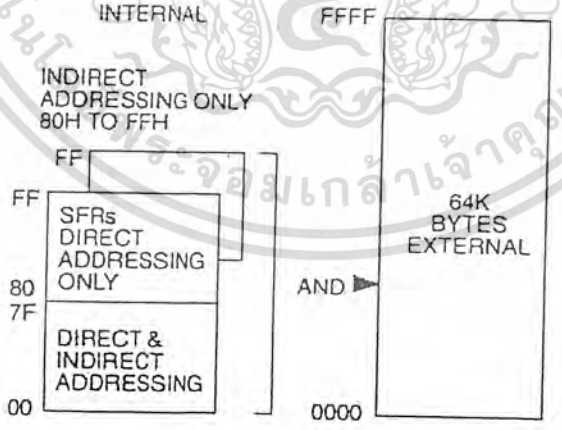
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.1 แสดงเบอร์ IC EPROM และ EEPROM

เบอร์	หน่วยความจำ	จำนวนขา
2716/2816	2K	24
2732/2832	4K	24
2764/2864	8K	24
27128/28128	16K	24

3.2 หน่วยความจำข้อมูล

คงพอเป็นที่ทราบกันดีแล้วว่าภายในไมโครคอนโทรลเลอร์นั้น จะมีหน่วยความจำข้อมูลภายในตัวชิปอยู่แล้ว ซึ่งจะเป็นประเภท Volatile (เมื่อหยุดจ่ายไฟเลี้ยงข้อมูลจะสูญหายไป) โดยขนาดของหน่วยความจำข้อมูล (RAM) นั้นแต่ละเบอร์ของ MCU จะมีขนาดไม่เท่ากันสามารถดูได้จากตารางที่ 2.1 ในบทที่ 2 ซึ่งประโยชน์ของหน่วยความจำข้อมูลนั้น จะถูกใช้งานในลักษณะจัดเก็บหรือพักข้อมูลเพียงชั่วคราวในขณะที่ CPU ทำการประมวลผล โดยขึ้นอยู่กับการเขียนซอฟต์แวร์การใช้งานนั่นเอง



รูป 3.2 แสดงการจัดตำแหน่งหน่วยความจำข้อมูลภายใน MCS-51

หน่วยความจำข้อมูล (RAM) ภายใน MCS- 51 ได้ถูกแบ่งย่อยลักษณะการทำงานหรือการใช้งานออกเป็น 3 ส่วนดังนี้

เอกสารนี้สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนที่ 1 หน่วยความจำข้อมูลทั่วไป 128 ไบต์ล่าง (00H-7FH) มีทุกเบอร์ใน MCS-51

ส่วนที่ 2 หน่วยความจำข้อมูลทั่วไป 128 ไบต์บน (80H-FFH) มีเฉพาะบางเบอร์

ส่วนที่ 3 หน่วยความจำข้อมูลที่ใช้งานเป็นรีจีสเตอร์ใช้งานเฉพาะ (80H-FFH) มีทุกเบอร์

ในหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์นั้นจะประกอบไปด้วยหน่วยความจำข้อมูลทั่วไป 128 ไบต์ล่าง และหน่วยความจำข้อมูลที่ใช้เป็นรีจีสเตอร์เฉพาะเป็นอย่างน้อย และมีหน่วยความจำข้อมูลที่ใช้งานทั่วไป 128 ไบต์บน เพิ่มขึ้นมาในบางเบอร์ของ MCU นั่นคือกรณีทางผู้ผลิตระบุไว้ว่า MCU เบอร์นั้นๆมี RAM ภายใน 256 ไบต์จะหมายถึง มี RAM สำหรับใช้งานทั่วไป 128 ไบต์ล่าง และ 128 ไบต์บน รวมทั้ง RAM ที่ใช้เป็น รีจีสเตอร์เฉพาะร่วมอยู่ด้วย

ทีนี้ปัญหาก็คือ ในกรณีที่ MCU มี RAM 256 ไบต์จะมีตำแหน่งของ RAM ใช้งานทั่วไป 128 ไบต์บนซ้อนทับอยู่กับ RAM ที่ใช้งานเป็นรีจีสเตอร์เฉพาะ กรณีนี้ CPU สามารถที่จะแยกแยะการงาน RAM ได้ด้วยคำสั่งในการเข้าถึงหรือใช้งาน โดยลายละเอียดเกี่ยวกับการเข้าถึงข้อมูลหรือหน่วยความจำที่จะกล่าวในหัวข้อต่อไป

3.2.1 การเข้าถึงหน่วยความจำข้อมูลของ MCS-51

ในการที่จะใช้งานหรือเข้าถึงหน่วยความจำของ ไมโครคอนโทรลเลอร์ MCS-51 นั้นในแต่ละส่วนของหน่วยความจำ จำเป็นที่จะต้องมีวิธีการใช้งานหรือเข้าถึงที่แตกต่างกัน เพื่อให้ CPU สามารถรับรู้หรือแยกแยะการใช้งานได้ โดยวิธีในการเข้าถึงหน่วยความจำนั้นมีหลายวิธีแต่วิธีสำคัญและจำเป็นจะต้องรู้จักเพื่อใช้งานหน่วยความจำมี 2 วิธีดังนี้

1. วิธีการเข้าถึงหน่วยความจำข้อมูลโดยตรง (Direct Addressing) วิธีนี้จะเป็นการกำหนดตำแหน่งหน่วยความจำที่ต้องการเข้าถึงหรืออ้างถึงโดยตรง ซึ่งหน่วยความจำที่สามารถเข้าถึงหรืออ้างถึงโดยวิธีนี้ จะเป็นหน่วยหน่วยความจำข้อมูลใช้งานทั่วไปภายในชิป 128 ไบต์ล่าง , หน่วยความจำข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ และหน่วยความจำภายนอกชิป (RAM ภายนอก) เช่นกรณีการใช้งานหน่วยความจำข้อมูลที่ใช้งานเป็นรีจีสเตอร์เฉพาะนั้น สามารถเข้าถึงหรือใช้งานได้โดยแสดงดังตัวอย่างเช่น

MOV A,0FH ; นำค่าภายในตำแหน่ง 0FH ไปเก็บไว้ใน A

MOV DPTR,#0FH ; นำค่า 0FH เก็บไว้ใน DPTR หรือกรณีใช้งานหน่วยความจำข้อมูล 128 ไบต์ล่างนั้นก็สามารถกระทำได้ดังตัวอย่าง

MOV 50H,#0FH ; นำค่า 0FH เก็บไว้ในหน่วยความจำข้อมูลตำแหน่ง 50H

2. วิธีการเข้าถึงหน่วยความจำทางอ้อม (Indirect Addressing) วิธีนี้จะเป็นการเข้าถึงหน่วยความจำข้อมูล โดยจะเก็บค่าตำแหน่งของหน่วยความจำที่ต้องการเข้าถึงในรีจีสเตอร์ใช้งานทั่วไปหรือรีจีสเตอร์เฉพาะ นั่นคือในวิธีนี้จะใช้รีจีสเตอร์เป็นตัวชี้ตำแหน่งของหน่วยความจำนั่นเอง ซึ่งหน่วยความจำที่สามารถเข้าถึงหรืออ้างถึงได้วิธีนี้คือ หน่วยความจำ ข้อมูลใช้งานทั่วไปภายในชิป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

128 ไบต์ล่างและ128ไบต์บน (กรณี MCU เบอร์นั้นมีหน่วยความจำ 128 ไบต์บน) และหน่วยความจำข้อมูลภายนอกชิป โดยรีจีสเตอร์ใช้งานทั่วไปและรีจีสเตอร์ใช้งานเฉพาะที่สามารถใช้เป็นตัวเก็บตำแหน่งของหน่วยความจำที่อ้างถึงได้มีดังนี้

- รีจีสเตอร์ใช้งานทั่วไป R0 และ R1 ของแต่ละแบงค์
- รีจีสเตอร์ใช้งานเฉพาะ SP (Stack Point)
- รีจีสเตอร์ใช้งานเฉพาะ DPTR (Data Pointer)

ตัวอย่างเช่น ถ้าต้องการใช้งานหน่วยความจำข้อมูลใช้งานทั่วไป 128 ไบต์บน จะสามารถเข้าถึงได้โดยรีจีสเตอร์ดังที่กล่าวมาข้างต้นในการเก็บค่าตำแหน่งของหน่วยความจำข้อมูลที่ต้องการใช้งานทั่วไป R0 ในการเก็บค่าตำแหน่งไบต์ ของหน่วยความจำข้อมูลที่ต้องการใช้งาน โดยรีจีสเตอร์ที่อ้างถึงนั้นจะต้องมีเครื่องหมาย " @ " อยู่ด้านหน้าโดยมีตัวอย่างการใช้งานดังนี้

MOV R0,#80H ; เก็บค่าตำแหน่ง 80 H ไว้ในรีจีสเตอร์ใช้งานทั่วไป R0

MOV @ R0,0FH ; นำค่า 0FH ไปเก็บไว้ในหน่วยความจำตำแหน่งที่ R0 ชี้อยู่จากตัวอย่างโปรแกรมข้างต้น จะเป็นการส่งค่า 0FH ไปเก็บไว้ยังหน่วยความจำข้อมูลใช้งานทั่วไป 128 ไบต์บน ณ ตำแหน่ง 80 H และจากที่ผ่านมาจะเห็นว่าถ้าต้องการใช้งานหน่วยความจำข้อมูลใช้งานทั่วไป 128 ไบต์ล่างก็จะสามารถกระทำได้ทั้งวิธีการเข้าถึงข้อมูลทั้งโดยทางตรงและทางอ้อมนั่นเอง

จากที่ทราบกันดีแล้ว ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์ ไม่ว่าจะมีหน่วยความจำโปรแกรม (EPROM) ภายในตัวชิปหรือไม่มีก็จะมีหน่วยความจำข้อมูล (RAM) อยู่ภายในตัวชิปทุกเบอร์ โดยขนาดน้อยที่สุดจะมีจำนวน 128 ไบต์ ซึ่งจำนวนขนาดของหน่วยความจำทั้งสองชนิดดังที่กล่าวมานั้นจะขึ้นอยู่กับเบอร์ของ MCU อย่างไรก็ตามถ้าต้องการใช้ขนาดหน่วยความจำข้อมูลภายในตัวชิป สามารถต่อขยายกับ IC หน่วยความจำข้อมูลภายนอกได้เช่นกัน โดยสามารถขยายได้สูงสุดถึง 64 กิโลไบต์

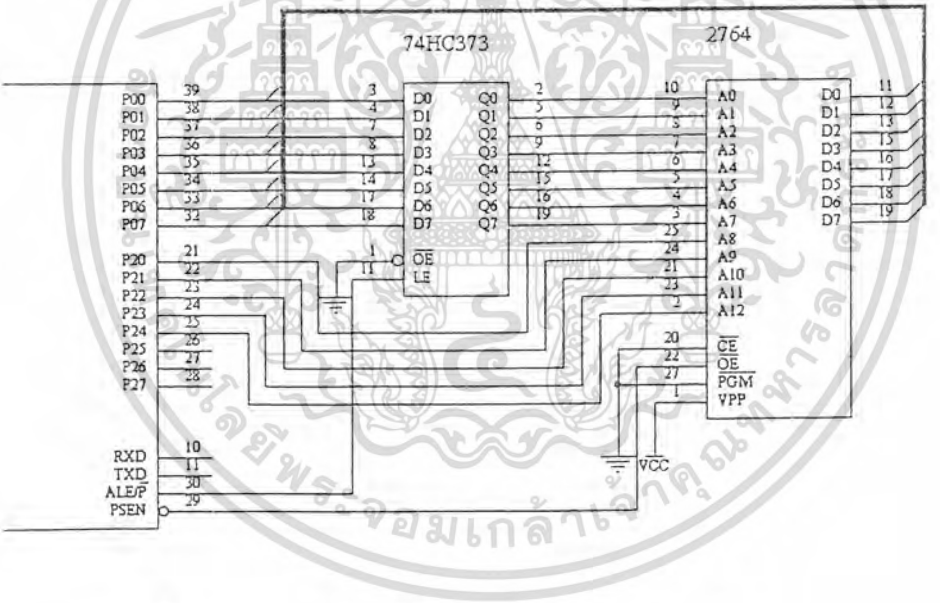
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การประยุกต์ใช้งานหน่วยความจำ

4.1 การอินเตอร์เฟส MCS-51 กับ EPROM ภายนอก / ภายใน

จากการที่ได้รู้คุณสมบัติของพอร์ตต่างๆของไมโครคอนโทรลเลอร์ 8051 กันมาพอสมควรแล้วจะเห็นว่า พอร์ต 0 และพอร์ต 2 ของไมโครคอนโทรลเลอร์ 8051 นั้นจะมีคุณสมบัติที่นอกจากจะเป็นพอร์ตใช้งาน I/O ธรรมดาแล้ว ยังสามารถใช้เป็นพอร์ตในการเข้าถึงหน่วยความจำภายนอกได้อีกด้วยโดยพอร์ต 0 และพอร์ต 2 ของ MCU จะใช้เป็นแอดเดรสไบต์ต่ำและแอดเดรสไบต์สูงตามลำดับ

4.1.1 การอินเตอร์เฟส MCS-51 กับ EPROM ภายนอก



รูปที่ 4.1 แสดงการอินเตอร์เฟส MCU กับหน่วยความจำภายนอก

ดังนั้นในการใช้งาน MCU ร่วมกับ EPROM ภายนอกก็จะใช้พอร์ต 0 และพอร์ต 2 ในรูปที่ 4.1 โดยจากวงจรจะเห็นว่ามี IC 74HC373 มาต่อร่วมระหว่างพอร์ต 0 กับขาแอดเดรสไบต์ต่ำของ EPROM ทั้งนี้ก็เพื่อทำหน้าที่เป็นตัว Latch แอดเดรสไบต์ต่ำ ให้กับ EPROM โดยมีขา ALE ของ MCU เป็นตัวควบคุมจังหวะการทำงานให้กับ IC 74HC373 ทุกครั้งที่มีการอ้างแอดเดรสติดต่อกับ EPROM โดยขอให้ผู้อ่านเข้าใจง่ายๆว่า พอร์ต 0 เมื่อต่อใช้งานดังรูปแล้วจะทำหน้าที่ทั้งการอ้างแอดเดรสไบต์ต่ำให้กับ EPROM และอ่านข้อมูลจากขา DATA ของ EPROM ซึ่งผู้อ่านคงจะพอทราบกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดีแล้วว่าการทำงานของพอร์ต I/O ของ MCU นั้นมีการทำงานอย่างไร ฉะนั้นในจังหวะที่อ้างแอดเดรสไบต์ต่ำให้กับ EPROM พอร์ต 0 จะไม่สามารถอ่านข้อมูลได้จาก EPROM ได้พร้อมๆกัน ดังนั้นจะต้องมี IC 74HC373 มาต่อร่วมเพื่อทำการ Latch แอดเดรสไบต์ต่ำเอาไว้แล้วพอร์ต 0 จึงจะสามารถอ่านข้อมูลจากขา DATA ของ EPROM ส่วนพอร์ต 2 นั้นไม่มีปัญหาอะไร เพราะหน้าที่การใช้งานมีหน้าที่เดียว คือการอ้างแอดเดรสไบต์สูงให้กับ EPROM เท่านั้น

จะเห็นว่ารูปที่ 4.1 เป็นวงจรตัวอย่างสำหรับการต่อใช้งาน MCU กับ EPROM ภายนอก ซึ่งเป็นเบอร์ 2764 โดยจะมีหน่วยความจำขนาด 8 Kbyte และจำนวนขาแอดเดรสที่ใช้อ้างตำแหน่งภายในตัว EPROM จะมีขา A0 – A12 ซึ่งเป็นขาที่ใช้ในการอ้างแอดเดรสของหน่วยความจำ

ในการคิดค่าขนาดของหน่วยความจำนั้นสามารถคำนวณได้จากขาแอดเดรสของ EPROM ได้จากรูปที่ 4.2 (ก) จะเห็นว่า EPROM เบอร์ 2764 จะมีขาแอดเดรสคือ A0 – A12 หรือ 13 บิต ขนาดหน่วยความจำโปรแกรมก็เท่ากับ 2^{13} = 8 Kbyte นั่นเอง ส่วนในเบอร์ 27128 ในรูปที่ 4.2(ข) ขาแอดเดรสคือ A0 – A13 จึงทำให้มีขนาดหน่วยความจำเป็น 2^{14} = 16 Kbyte ส่วนเบอร์อื่นๆ นอกเหนือจากที่กล่าวมาก็ใช้หลักการเดียวกันนี้ในการหาค่าขนาดหน่วยความจำได้เช่นกัน หรือจะใช้วิธีนำ 8 ไปหารค่า 3 หลักท้ายของเบอร์ IC ก็ได้เช่นกันโดยจะมีหน่วยเป็น Kbyte ในลักษณะการต่อใช้งาน MCU กับ EPROM ภายนอก รูปแบบหลักที่ง่ายและเป็นที่นิยมใช้ก็จะเป็นดังรูปที่ 4.1

รูปที่ 4.2

(ก) แสดงลักษณะภายนอกของ EPROM เบอร์ 27C64

(ข) แสดงลักษณะภายนอกของ EPROM เบอร์ 27C128

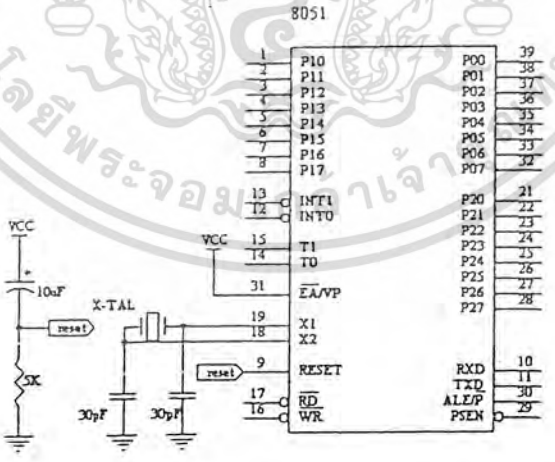
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทั้งหมดที่ได้อธิบายมาก็เพียงเพื่อให้ผู้อ่านรู้และทำความเข้าใจเอาไว้เพื่อที่จะได้นำความรู้นี้ไปเชื่อมต่อความสัมพันธ์ในเรื่อง ขาแอดเดรสกับขนาดหน่วยความจำของ EPROM ได้

เมื่อกลับมายังวงจรดังรูปที่ 4.1 จะเห็นว่ามีการต่อใช้ขาแอดเดรสของพอร์ต 2 เหลืออยู่คือ P2.5 , P2.6 , และ P2.7 ซึ่งไม่ต้องกังวลใดๆทั้งสิ้น เนื่องจาก EPROM เบอร์นี้ ต้องการพอร์ตสำหรับอ้างแอดเดรสของหน่วยความจำโปรแกรมเพียง 13 บิต เท่านั้น ฉะนั้นขา P2.5 , P2.6 และ P2.7 ก็จะเว้นว่างเอาไว้หรือจะนำไปใช้งานก็ได้ แต่มีรูปแบบที่สามารถใช้งานได้ซึ่งจะเป็นรูปแบบเฉพาะไม่สามารถนำไปใช้งานในระดับอิสระได้อีก แต่จะสามารถนำไปใช้งานในลักษณะจัด Memory Map ใช้งานได้ ซึ่งจะได้กล่าวต่อไปในบทที่ 7 เรื่องการอินเตอร์เฟส MCS – 51 ร่วมกับ 8255 โดยการใช้ EPROM ภายนอก ต่อไปจะกล่าวถึงการใช้งาน MCU โดยการใช้งาน EPROM ภายในดูบ้าง (EPROM บางเบอร์เท่านั้นสามารถดูได้จากตารางแสดงตระกูลไมโครคอนโทรลเลอร์ MCS – 51 ในบทที่ 2)

4.1.2 การใช้งานหน่วยความจำ (EPROM) ภายใน MCS –51

จากบทที่ 2 ผู้อ่านคงพอจะทราบและพอเข้าใจคุณสมบัติขาต่างๆของไมโครคอนโทรลเลอร์ MCS-51 กันมาแล้ว และจากวงจรในรูปที่ 4.3 จะเห็นว่าขา EA ของ MCU จะถูกต่อกับ Vcc (+5Volt) ซึ่งเท่ากับว่าเป็นการเซตเพื่อใช้งาน EPROM ภายในตัวของ MCU นั้น จะขึ้นอยู่กับเบอร์ของ MCU จะใช้วิธีหาขนาดของหน่วยความจำแบบ EPROM ภายนอกดังที่กล่าวมาข้างต้นไม่ได้ โดย MCU เบอร์อะไรหรือมีขนาดหน่วยความจำภายในเท่าไร สามารถดูได้จากตารางแสดงตระกูลไมโครคอนโทรลเลอร์ MCS-51 ในบทที่ 2



รูปที่ 4.3 แสดงการใช้งานหน่วยความจำภายในตัว MCU

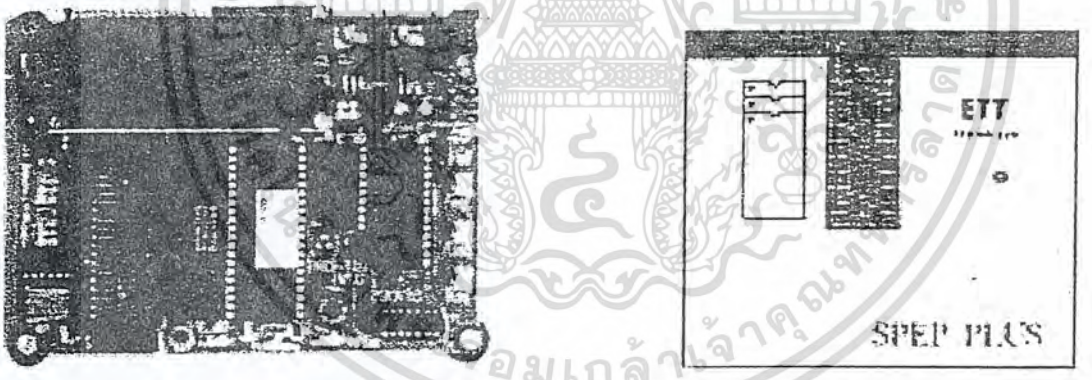
กลับมาพิจารณาวงจรในรูปที่ 4.3 เพียงต่อวงจรตามรูปนั้น ก็สามารถที่จะใช้งาน EPROM ภายในของตัว MCU ได้แล้วโดยจะมีพอร์ตเหลือให้ใช้งานได้ครบทุกพอร์ตของ MCU ข้อเสียของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้งาน EPROM ภายในตัว MCU คือ ในขั้นต้นการพัฒนาโปรแกรมนั้นไม่สามารถที่จะใช้ EPROM Emulator เข้ามาช่วยในการพัฒนาได้ ซึ่งการที่จะใช้งานในลักษณะนี้ จะต้องมีการเขียนโปรแกรมที่สมบูรณ์และแน่นอน แล้วจึงทำการโหลดโปรแกรมลงภายใน EPROM ภายในตัว MCU ด้วยเครื่องโปรแกรม EPROM ภายในตัว MCU วิธีเดียวเท่านั้น

ฉะนั้นวงจรดังรูป 4.3 หรือการใช้งาน EPROM ภายในตัว MCU มักจะใช้งานในขั้นตอนสุดท้ายในการนำไปใช้งานจริง โดยหลังจากที่ได้ทำการพัฒนาโปรแกรมจากวงจรภายในรูปที่ 4.1 แล้ว ซึ่งจะใช้ ERROM Emulator ช่วยในการพัฒนา ซึ่งจะเห็นได้ว่าการต่อวงจรใช้งานในรูที่ 4.3 จะมีความซับซ้อนน้อยกว่าและมีความประหยัดค่าใช้จ่ายมากกว่าในวงจรรูปที่ 4.1 ดังนั้นจึงเหมาะสมที่จะนำไปใช้งานจริง แต่ข้อเสียคือยากต่อการพัฒนาโปรแกรม ซึ่งทั้งสองวงจรก็จะมีข้อได้เปรียบและเสียเปรียบดังที่กล่าวมา

แต่สรุปว่าในการพัฒนาในด้านไมโครคอนโทรลเลอร์นั้น ควรจะมีการทดลองหรือพัฒนาโปรแกรมโดยใช้ EPROM Emulator โดยใช้วงจรดังรูปที่ 4.1 ก่อนเมื่อโปรแกรมถูกต้องและสมบูรณ์แล้วจึงนำไป Load ลง EPROM ของ MCU ที่มี EPROM ภายในเพื่อนำไปสู่การใช้งานจริงๆ นั่นเอง



(ก) (ข)

รูปที่ 4.4

(ก) แสดงเครื่องโปรแกรม EPROM ภายใน / ภายนอกตัว MCU

(ข) แสดงเครื่องโปรแกรมเฉพาะ EPROM ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 การประยุกต์ใช้งาน MCS-51 กับ RAM ภายใน / ภายนอก

จากที่ผ่านมาคงจะพอรู้จักหน่วยความจำข้อมูลภายในชิป MCS-51 กันมาบ้างแล้วทีนี้มาถึงขั้นตอนในการใช้งานหน่วยความจำข้อมูลกันดูบ้าง โดยจะเริ่มต้นอธิบายการประยุกต์ใช้งานหน่วยความจำข้อมูลกันดูบ้าง โดยจะเริ่มต้นอธิบายการประยุกต์ใช้งานหน่วยความจำข้อมูลภายในชิป MCS-51 กันก่อน

4.2.1 การประยุกต์ใช้งานหน่วยความจำข้อมูลภายในชิป MCS – 51

โดยการใช้งานหน่วยความจำข้อมูลภายในนั้น จะขออธิบายการใช้ข้อมูลรีจีสเตอร์สำหรับการใช้งานทั่วไปภายใน RAM และรีจิสเตอร์ใช้งานเฉพาะที่เกี่ยวข้องในการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 ในระดับต้น เพราะรีจีสเตอร์ใช้งานเฉพาะนั้นก็จะมีรูปแบบการใช้งานเฉพาะตัว ซึ่งก็จะมีรายละเอียดในการใช้งานที่ค่อนข้างซับซ้อน ในขั้นต้นแล้วส่วนมากจะนิยมใช้งาน RAM ในการจัดเก็บหรือพักข้อมูลระหว่างการประมวลผลของ CPU เพียงเท่านั้น

ฉะนั้นรีจีสเตอร์ที่จะถูกนำมาใช้งานในลักษณะดังกล่าวจะเป็นรีจีสเตอร์ใช้งานทั่วไป 128 ไบต์ล่าง (รีจีสเตอร์ที่หน่วยความจำข้อมูล ณ ตำแหน่ง 00H – 1FH รวม 32 ไบต์จะมีอยู่ในทุกเบอร์ของ MCS-51) ซึ่งจะถูกจัดแบ่งออกเป็น 4 กลุ่มในการใช้งาน โดยในแต่ละกลุ่มจะประกอบไปด้วยรีจีสเตอร์ใช้งานทั่วไปจำนวน 8 ไบต์ (รีจีสเตอร์ R0 - R7) ซึ่งจะเรียกแต่ละกลุ่มดังกล่าวว่าแบงค์ (Bank) นั่นก็คือภายใน MCS-51 จะมีการแบ่งรีจีสเตอร์ใช้งานทั่วไป ณ ตำแหน่ง 00H – 1FH ออกเป็นแบงค์ 0 ถึงแบงค์ 3 ซึ่งในแต่ละแบงค์นั้นก็จะมีชื่อรีจีสเตอร์ในการใช้งานเหมือนกันคือ R0 – R7 ซึ่ง CPU จะรับรู้ความแตกต่างในการใช้งานรีจีสเตอร์แต่ละแบงค์ได้โดย คำสั่งในการควบคุมแบงค์ในการใช้งานรีจีสเตอร์ R0 – R7 ตัวอย่างเช่น รีจีสเตอร์ R0 ในแบงค์ 0 จะเป็นคนละรีจีสเตอร์กับ R0 ในแบงค์ 1 เป็นต้น ฉะนั้นเราจะมีรีจีสเตอร์ใช้งานทั่วไปซึ่งมีจำนวนถึง 32 ตัวนั่นเอง

ทีนี้มาถึงการควบคุมแบงค์และการใช้งานบ้างว่าจะทำอย่างไร ก่อนอื่นต้องขอบอกไว้ก่อนว่า เมื่อทำการเริ่มต้นใช้งานหรือจ่ายไฟเลี้ยงให้กับ MCU แบงค์การใช้งานหน่วยความจำข้อมูลจะเริ่มที่แบงค์ 0 โดยอัตโนมัติทุกครั้งและถ้าจะเปลี่ยนไปใช้งานรีจีสเตอร์ในแบงค์ใดจะต้องมีการเปลี่ยนหรือกำหนดใหม่เท่านั้น

จากที่กล่าวไปแล้วว่าจะมีรีจีสเตอร์ใช้งานเฉพาะ โดยในการควบคุมแบงค์ใช้งานนั้นก็จะใช้รีจีสตอร์ใช้งานเฉพาะในการควบคุม โดยในการควบคุมการเลือกใช้งานรีจีสเตอร์ใช้งานทั่วไป R0 – R7 นั่นก็คือบิต RS1 และ RS0 ซึ่งอยู่ภายในรีจีสเตอร์ใช้งานเฉพาะ PSW ซึ่งมีรายละเอียดดังต่อไปนี้

รีจีสเตอร์ใช้งานเฉพาะ PSW % : Program Status Word (bit Addressress

CY	AC	FO	RS1	RS0	0V	-	P

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รีจีสเตอร์ใช้งานเฉพาะ PSW ภายในนั้นจะมีการจัดเรียงของบิตใช้งานเฉพาะต่างๆจำนวน 8 บิตดังตารางข้างต้น โดยเราจะกล่าวถึงนั้นก็คือ บิต RS1 และ RS0 ซึ่งจะมีหน้าที่ในการใช้งานควบคุม ในการเลือกใช้งานแบงค์ของรีจีสเตอร์ใช้งานทั่วไป R0- R7 โดยตารางการใช้งานแบงค์ที่ควบคุมโดยบิต RS1 และ RS0 แสดงดังตาราง 4.1

ตารางที่ 4.1 แสดงบิตควบคุมแบงค์ RS1 , RS0

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

จากตารางที่ 4.1 ผู้เขียนจะยกโปรแกรมตัวอย่างการใช้งานหน่วยความจำข้อมูลทั่วไป โดยโปรแกรมจะนำค่า 0FH เก็บไว้ในรีจีสเตอร์ R0 ในแบงค์ที่ 1 ของหน่วยความจำข้อมูลทั่วไป ซึ่งโปรแกรมการใช้งานสามารถเขียนได้ 2 วิธี โดยมีการทำงานเช่นเดียวกันดังต่อไปนี้

```
EX1  วิธีการเลือกแบงค์การใช้งานแบบที่ 1 (Bit Addressable)
     CLR  RS1
     SETB RS0
     MOV  R0,#0FH
EX2  วิธีการเลือกแบงค์การใช้งานแบบที่ 2 (Not Bit Addressable)
     MOV PSW,#08H ; RS1 = 0 ,RS0 = 1
     MOV R0,#0FH
```

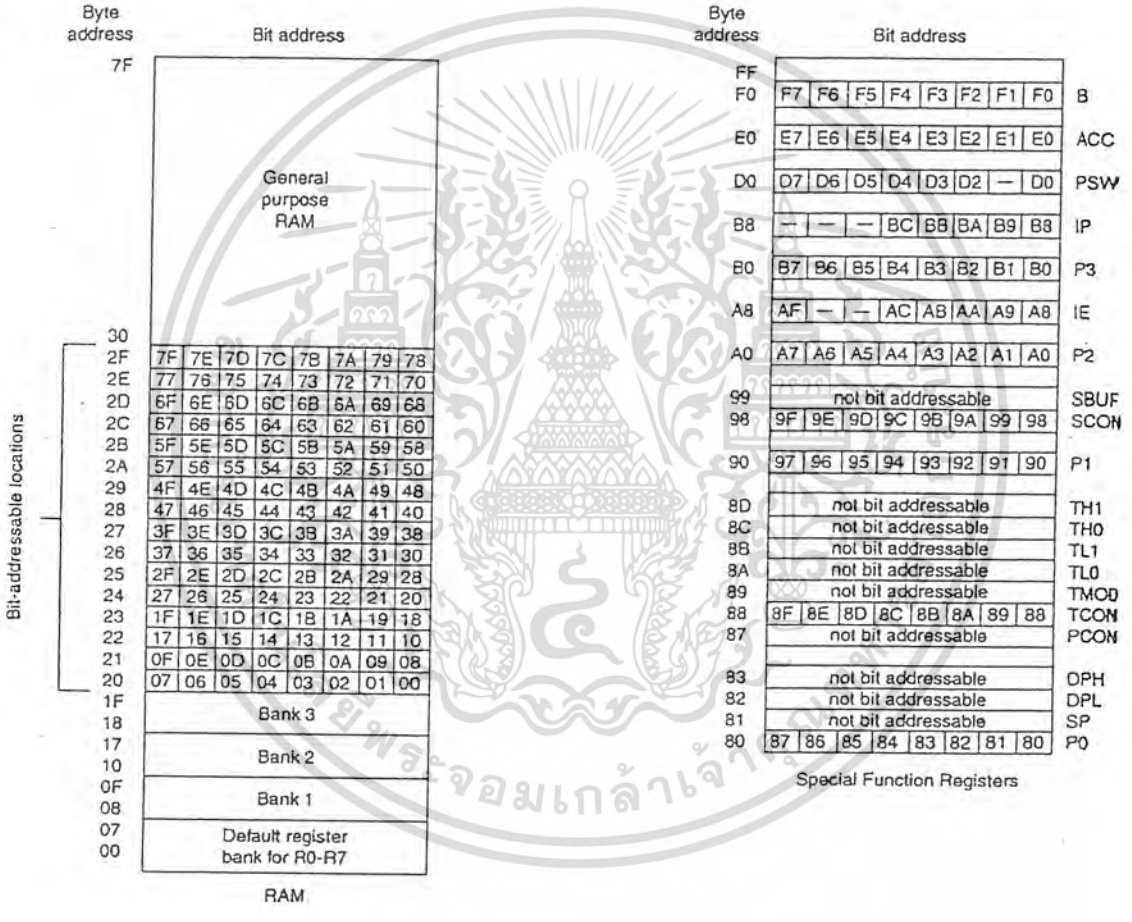
หลังจากเขียนโปรแกรมดังตัวอย่างข้างต้นแล้ว เมื่อมีการใช้งานรีจีสเตอร์ต่อไปก็จะยังคงใช้งานรีจีสเตอร์ในแบงค์ที่ 1อยู่ อย่างที่กล่าวไว้แล้วในตอนแรก ถ้าต้องการใช้งานรีจสเตอร์ในแบงค์ใดก็จะต้องทำการควบคุมบิต RS0 และ RS1 ในการเลือกแบงค์การใช้งานก่อนทุกครั้ง เช่น หลังจากเขียนโปรแกรมดังตัวอย่างข้างต้นแล้วเมื่อต้องการกลับไปยังแบงค์ 0 จะต้องทำการ Clear Bit RS0 และ RS1 ซึ่งก็คือให้บิต RS0 และ RS1 เป็น 0 จึงจะกลับไปใช้งานรีจีสเตอร์ทั่วไปในแบงค์ 0 ได้

จากนั้นหน่วยความจำข้อมูลตั้งแต่ตำแหน่ง 20H – 2FH รวมทั้งหมด 16 ไบต์(128บิต) จะเป็นหน่วยความจำข้อมูลชนิดพิเศษ (Bit Addressable Area) ซึ่งหน่วยความจำข้อมูลในช่วงตำแหน่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังกล่าวสามารถใช้งานในการเข้าถึงข้อมูลในระดับบิตได้ โดยจากรูปที่ 4.5 จะเห็นว่าที่ตำแหน่งไบต์ 20H จะมีบิตย่อยอยู่ภายใน 8 บิต คือ 00H – 07H โดนบิตภายในนี้สามารถที่เข้าถึงข้อมูลโดยตรงในระดับบิตได้ ดังเช่นใช้คำสั่ง SETB 00H ก็จะเป็นการเซ็ตค่าบิต 00H ให้มีค่าเป็น "1" ที่ตำแหน่งไบต์ 20H นั่นเอง

ถัดมาในส่วนหน่วยความจำข้อมูลตำแหน่ง 30H – 7FH จะเป็นอีกส่วนของหน่วยความจำข้อมูลที่สามารถใช้งานได้ทั่วไปสำหรับใน MCU ทุกเบอร์ โดยจากที่กล่าวมาแล้วนั้นพื้นที่ในช่วง 00H ถึง 7FH นี้สามารถใช้งานในการเข้าถึงข้อมูลทั้งทางตรงและทางอ้อม



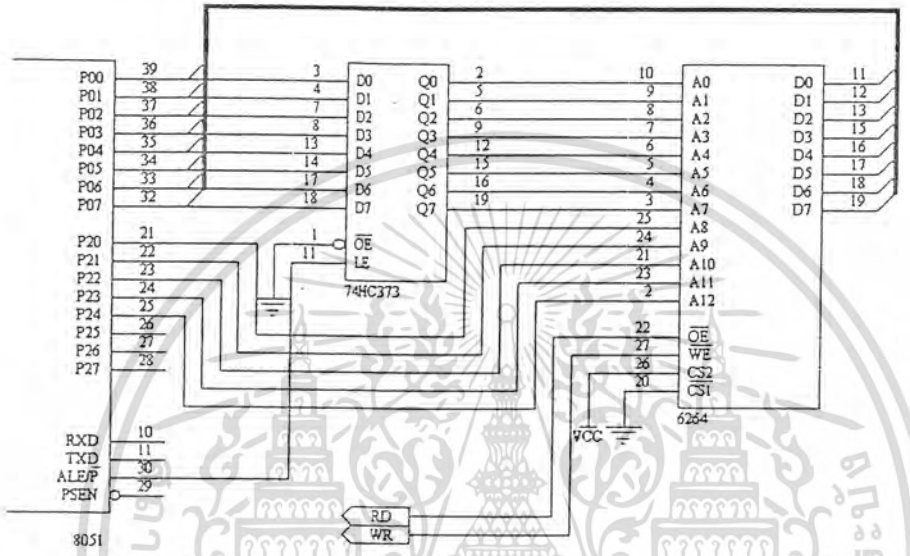
รูปที่ 4.5 แสดงหน่วยความจำข้อมูล 128 ไบต์ล่างและรีจิสเตอร์ใช้งานเฉพาะ

ทั้งหมดที่กล่าวมาก็เป็นเพียงการแนะนำการประยุกต์ใช้งานหน่วยความจำข้อมูลภายในตัว MCU ในเบื้องต้น ซึ่งต่อไปจะเป็นการกล่าวถึงการประยุกต์ใช้งานหน่วยความจำข้อมูลภายนอกนั่นเอง

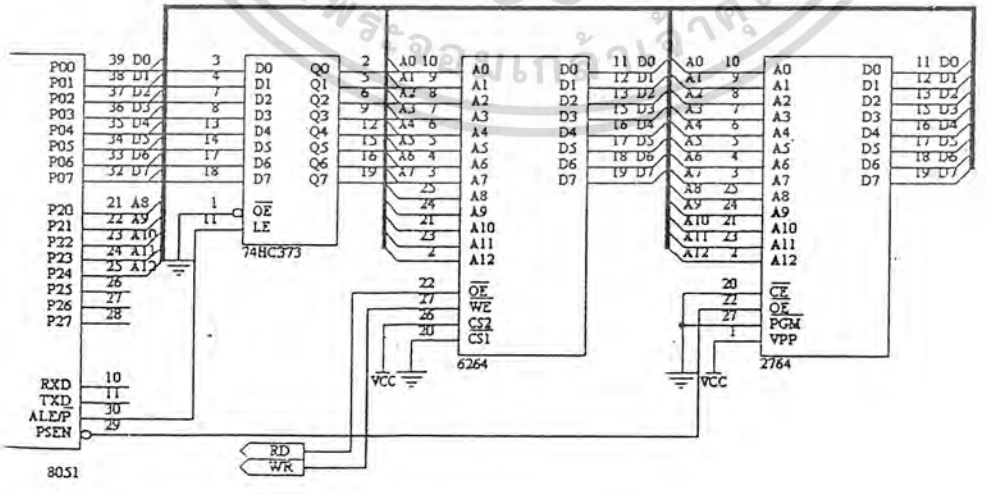
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4..2.2 การประยุกต์ใช้งานหน่วยความจำข้อมูลภายนอกชิป MCS-51

จากที่ได้รู้มาแล้วว่าภายในชิป MCS – 51 นั้น จะมีหน่วยความจำข้อมูลภายในตัวชิปแต่ในบางครั้งจำเป็นจะต้องใช้ขนาดของหน่วยความจำข้อมูลเกินกว่าขนาดของหน่วยความจำข้อมูลที่มีอยู่ภายชิป ซึ่งสามารถที่จะนำ IC หน่วยความจำข้อมูลเข้ามาต่อใช้งานร่วมกับ MCU เพื่อเพิ่มขนาดของหน่วยความจำข้อมูลใช้งานได้โดยมีการต่อวงจรใช้งานดังรูปที่ 4.6



รูปที่ 4.6 แสดงวงจรใช้งาน RAM ภายนอกชิป MCS-51

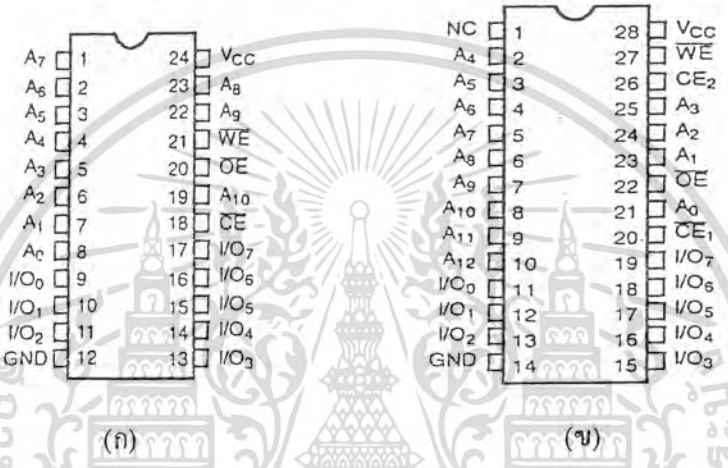


รูปที่ 4.7 แสดงวงจรใช้งาน RAM , EPROM ภายนอกชิป MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากวงจรในรูปที่ 4.6 จะพบว่าเป็นการใช้งานหน่วยความจำข้อมูลภายนอกร่วมกับ MCU ที่มีหน่วยความจำโปรแกรม (EPROM) ภายใน หรือถ้าต้องการใช้งานร่วมกับ MCU ที่มีการต่อใช้งานหน่วยความจำโปรแกรมภายนอก ก็สามารถที่จะต่อวงจรได้ดังรูปที่ 4.7

โดย IC ที่นิยมนำมาต่อเป็นหน่วยความจำข้อมูลภายนอกร่วมกับ MCU คือ IC เบอร์ 6116 , 6264 โดยมีขนาดหน่วยความจำข้อมูล 2 Kbyte และ 8Kbyte ตามลำดับ เท่านี้ก็เพียงพอกับการใช้งานหรืออาจจะใช้ IC หน่วยความจำข้อมูลเบอร์อื่น โดยสามารถต่อขยายได้ถึง 64 Kbyte



รูปที่ 4.8

(ก) แสดง IC หน่วยความจำข้อมูล (RAM) เบอร์ 6116

(ข) แสดง IC หน่วยความจำข้อมูล (RAM) เบอร์ 6264

จากรูปที่ 4.8 จะเห็นว่าขนาดหน่วยความจำข้อมูลที่เพิ่มขึ้นนั้นขาแอดเดรสของ IC จะเป็นตัวกำหนดขนาดของหน่วยความจำข้อมูลที่เพิ่มขึ้น แต่ขาใช้งานอื่นๆยังคงมีลักษณะการใช้งานเหมือนเดิม

มาถึงขั้นตอนในการใช้งานหน่วยความจำข้อมูลภายนอกชิป โดยจากวงจรในรูปที่ 4.6 จะใช้พอร์ต 0 ของ MCU ในการอ้างแอดเดรสไบต์ต่ำ และพอร์ต 2 ในการอ้างแอดเดรสไบต์สูงให้กับ RAM ภายนอก โดยจะเห็นว่าพอร์ต 0 ก็จะต่อเข้ากับพอร์ต Data Bus ของ RAM ภายนอกซึ่งพอร์ต 0 นั้นไม่สามารถที่จะใช้งานในการอ้างแอดเดรสไบต์ต่ำและอ่านเขียนข้อมูลไปยัง RAM ภายนอกได้พร้อมๆกันด้วย ด้วยเหตุนี้จึงมีการต่อ IC 74HC373 ร่วมเข้าไปเพื่อช่วยในการ Latch แอดเดรสไบต์ต่ำให้กับ RAM ภายนอก โดยขา ALE ของ MCU จะเป็นตัวควบคุมจังหวะการ Latch ให้กับ IC 74HC373 และขณะที่ IC 74HC373 Latch แอดเดรสไบต์ต่ำอยู่นั้นพอร์ต 0 จะทำการอ่านเขียนข้อมูลกับRAMภายนอกได้นั่นเองซึ่งจากวงจรจะเห็นว่าRAMภายนอกนั้น จะมีสัญญาณควบคุมในการ

เอกสารนี้ได้รับการสงวนลิขสิทธิ์ สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อ่านเขียนข้อมูล โดยใช้ขาสัญญาณ RD และ WR ของ MCU เป็นตัวควบคุม ในพอร์ต 2 นั้น ไม่มีปัญหาแต่อย่างใดเพราะใช้ในการอ้างตำแหน่งแอดเดรสไบต์สูงเพียงหน้าที่เดียวเท่านั้น

โดยในการใช้งานหน่วยความจำข้อมูลภายนอกนั้น จะมีการจัดตำแหน่งโดยเริ่มตั้งแต่ 0000H - ค่าของขนาดหน่วยความจำข้อมูลภายนอกนี้ จะใช้รีจีสเตอร์เป็นตัวชี้ตำแหน่งหรือเก็บตำแหน่งของหน่วยความจำข้อมูลที่อ้างถึงรีจีสเตอร์ที่สามารถอ้างถึงตำแหน่งของหน่วยความจำข้อมูลภายนอกได้ทั้งหมดตลอด 64 Kbyte (0000H – FFFFH) นั้นจะใช้รีจีสเตอร์ใช้งานเฉพาะ DPTR ซึ่งมีขนาด 16 บิต ในการเก็บค่าตำแหน่งของหน่วยความจำข้อมูลภายนอกที่อ้างถึง และยังสามารถใช้รีจีสเตอร์ R0 และ R1 ของแต่ละแบงค์ ในการเก็บค่าตำแหน่ง 256 ไบต์แรก (8 บิต) ของหน่วยความจำข้อมูลภายนอกที่ต้องการอ้างถึงได้เช่นกัน ในการติดต่อกับหน่วยความจำข้อมูลภายนอกนั้นจะต้องใช้คำสั่ง MOVX ในการติดต่อเพื่ออ่านหรือเขียนข้อมูลภายนอกเท่านั้น

โดยตัวอย่างโปรแกรมในการติดต่อกับหน่วยความจำข้อมูลภายนอก ซึ่งจะใช้รีจีสเตอร์ DPTR ในการเก็บค่าตำแหน่งของหน่วยความจำข้อมูลภายนอก โดยรีจีสเตอร์ที่เก็บค่าตำแหน่งของหน่วยความจำข้อมูลที่อ้างถึงนั้นจะต้องมีเครื่องหมาย "@" ไว้ข้างหน้าหรือเรียกว่าการเข้าถึงข้อมูลโดยทางอ้อม (Indirect Addressable) ดังนี้

```
MOV DPTR,#1000H ; ให้รีจีสเตอร์ DPTR เก็บค่าตำแหน่ง 1000H
MOV A,#0FH      ; กำหนดค่าใน A มีค่า 0FH
MOVX @DPTR,A    ; ส่งค่าภายใน A ไปเก็บไว้ในตำแหน่งที่ DPTR ชี้อยู่
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การใช้งานอินเตอร์รัพท์ภายนอก INT0 และ INT1

จากการที่ได้แนะนำการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 กับหน่วยความจำมากันเป็นที่รู้จักหรือเข้าใจพอสมควรแล้ว ต่อไปนี้จะกล่าวถึงจุดที่ไม่ควรมองข้ามในการนำไมโครคอนโทรลเลอร์มาใช้งาน นั่นก็คือการใช้งานอินเตอร์รัพท์ภายนอกนั่นเอง ซึ่งถ้าจะพูดกันตรงๆ สำหรับผู้ที่มีความสนใจทางด้านไมโครคอนโทรลเลอร์นั้น ในการนำไปใช้งานก็ควรที่จะรู้จักการใช้งานขาอินเตอร์รัพท์ภายนอก INT0 และ INT1 ของไมโครคอนโทรลเลอร์ MCS-51 บ้างหรือใช้งานให้ชำนาญก็จะสามารถนำไมโครคอนโทรลเลอร์ไปประยุกต์ใช้งานได้ง่ายและมีประสิทธิภาพยิ่งขึ้นโดยในการใช้งานอินเตอร์รัพท์ภายนอกนั้นก็จะมีรีจีสเตอร์ใช้งานเฉพาะต่างๆ ในการควบคุมการทำงานการใช้งานอินเตอร์รัพท์ภายนอก ซึ่งจะขอแนะนำรีจีสเตอร์ใช้งานเฉพาะต่างๆที่เกี่ยวข้องดังนี้

รีจีสเตอร์ IE : Interrupt Enable Register (Bit Addressable)

EA	-	ET2	ES	ET1	EX1	ET0	EX0

ตารางที่ 5.1 แสดงบิตในการควบคุมการใช้งานอินเตอร์รัพท์ภายนอกที่เกี่ยวข้องภายในรีจีสเตอร์ IE

สัญลักษณ์	หน้าที่การใช้งาน
EA	EA = 0 จะไม่มีการตอบรับการ Interrupt ภาย นอกใดๆทั้งหมด EA = 1 จะทำการตอบการ Interrupt จากภายนอ
EX0	EX0 = 0 จะไม่ทำการตอบรับ Interrupt ภายนอกที่ขา INT0 EX0 = 1 จะทำการตอบรับ Interrupt จากภายนอกเมื่อเกิดการเปลี่ยนแปลงลักษณะสัญญาณที่ขา INT0 ตามที่กำหนดไว้
EX1	EX1 = 0 จะไม่ทำการตอบรับ Interrupt ภายนอกที่ขา INT1 EX1 = 1 จะตอบรับการ Interrupt ภายนอกเมื่อเกิดการเปลี่ยนแปลงลักษณะสัญญาณที่ขา INT1 ตามที่กำหนดไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเริ่มการใช้งานขา INT0 และ INT1 หรืออินเตอร์รัพท์ภายนอก นั้นจำเป็นจะต้องรู้จักโครงสร้างหรือรีจีสเตอร์ภายในต่างๆที่เกี่ยวข้องกับการควบคุม การใช้งานอินเตอร์รัพท์ภายนอกกันก่อน โดยรีจีสเตอร์ที่ใช้ในการควบคุมการใช้งานอินเตอร์รัพท์ภายนอกนั้นจะประกอบไปด้วยรีจีสเตอร์ที่สำคัญจำนวน 3 ตัว คือ รีจีสเตอร์ IE (Interrupt Enable Register) รีจีสเตอร์ TCON (Timer / Counter Control Register) และรีจีสเตอร์ IP (Interrupt Priority Register) ซึ่งรีจีสเตอร์ทั้งสามตัวนั้นเป็นรีจีสเตอร์ใช้งานเฉพาะ โดยสามารถเข้าถึงข้อมูลได้ในระดับบิต (Bit Addressabie)

ในตาราง ที่ 5.1 จะเป็นรายละเอียดของบิตต่างๆที่เกี่ยวข้องในการใช้งานอินเตอร์รัพท์ภายนอก INTO , INT1 ภายในรีจีสเตอร์ IE โดยจากรายระเอียดของบิตควบคุมการอินเตอร์รัพท์ข้างต้นผู้อ่านคงจะสามารถศึกษาการทำงานของบิตต่างๆ ในการควบคุมการใช้งานอินเตอร์รัพท์ภายนอกภายในรีจีสเตอร์ IE ได้ด้วยตัวเองเนื่องจากไม่มีความซับซ้อนแต่อย่างใด พอรู้จักบิตควบคุมการใช้งานอินเตอร์รัพท์ภายนอกต่างๆ ที่เกี่ยวข้องและการทำงานต่างๆภายในรีจีสเตอร์ IE จากตารางที่5.1 แล้ว ต่อไปในส่วนของรีจีสเตอร์ตัวที่สอง ที่มีส่วนสำคัญในการใช้งานอินเตอร์รัพท์ภายนอกนั่นก็คือ รีจีสเตอร์ใช้งานเฉพาะ TCON

รีจีสเตอร์ TCON : Timer / Counter Control Register (Bit AddressAble)

EA	-	ET2	ES	ET1	EX1	ET0	EX0

ตารางที่ 5.2 แสดงบิตในการควบคุมการใช้งานอินเตอร์รัพท์ภายนอกที่เกี่ยวข้องภายในรีจีสเตอร์ TCON

สัญลักษณ์	หน้าที่การทำงาน
IT0	IT0 = 0 จะเป็นการกำหนดลักษณะการ Interrupt ภาย นอกของ ขา INT0 โดยจะเกิดการ Interrupt เมื่อเกิดสภาวะ Low ที่ขา INT0 (ระยะเวลาพอสมควร) IT = 1 จะเป็นการกำหนดรูปแบบลักษณะการเปลี่ยนแปลงสัญญาณในการ Interrupt ภายนอกของขา INT0 โดยให้มีรูปแบบเป็น Active Low (ขอบขาลง) โดยสภาวะขา INT0 เริ่มต้นต้องอยู่ในสภาวะ Hight
IT1	IT1 = 0จะเป็นการกำหนดลักษณะการ Interrupt ภายนอกของขา INT 1 โดยจะเกิดการ Interrupt เมื่อเกิดสภาวะ Low ที่ขา INT1 (ระยะเวลาพอสมควร) IT1 = 1 จะเป็นการกำหนดรูปแบบลักษณะการเปลี่ยนแปลงสัญญาณในการ Interrupt ภายนอกของขา INT1 โดยให้มีรูปแบบเป็น Active Low (ขอบขาลง) โดยสภาวะขา INT1 เริ่มต้นต้องอยู่ในสภาวะ High

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางที่ 5.2 จะแสดงรายละเอียดของบิต IT0 และ IT1 ซึ่งเกี่ยวข้อง ในการใช้งานอินเตอร์รัพท์ภายนอก INT0 , INT1 ภายในรีจีสเตอร์ TCON โดยจากรายละเอียดของบิตควบคุมการอินเตอร์รัพท์ในตารางที่ 5.2 ผู้อ่านคงจะสามารถศึกษาการทำงานของบิต IT0 และ IT1 ในการควบคุมการใช้งานอินเตอร์รัพท์ภายนอกภายในรีจีสเตอร์ TCON ได้ด้วยตัวเองเนื่องจากไม่มีความซับซ้อนแต่อย่างใด เช่นกันพอรู้จักบิตควบคุมการใช้งานอินเตอร์รัพท์ภายนอกต่างๆ ที่เกี่ยวข้องและการทำงานต่างๆภายในรีจีสเตอร์ TCON จากตารางที่ 5.2 แล้วต่อไปในส่วนของรีจีสเตอร์ตัวที่สามที่มีส่วนสำคัญในการใช้งานอินเตอร์รัพท์ภายนอกนั่นก็คือ รีจีสเตอร์ใช้งานเฉพาะ IP

รีจีสเตอร์ IP จะทำหน้าที่ในการจัดลำดับความสำคัญของการอินเตอร์รัพท์ ซึ่งรายระเอียดต่างๆของบิตที่เกี่ยวข้องในการใช้งานอินเตอร์รัพท์ภายนอก INT0 , INT1 ภายในรีจีสเตอร์ IP แสดงดังตารางที่ 5.3

รีจีสเตอร์ IP : Interrupt Priority Register (Bit Address)

-	-	PT2	PS	PT1	PX1	PT0	PX0

ตารางที่ 5.3 แสดงบิตในการจัดลำดับอินเตอร์รัพท์ภายนอกที่เกี่ยวข้องภายในรีจีสเตอร์ IP

สัญลักษณ์	หน้าที่การทำงาน
PX 1	PX 1 = '1' จะเป็นการกำหนดลำดับความสำคัญในการอินเตอร์รัพท์ภายนอกที่ขา INT 1 ในระดับสูง PX 1 = '0' จะเป็นการกำหนดลำดับความสำคัญในการอินเตอร์รัพท์ภายนอกที่ขา INT 1 ในระดับต่ำ
PX 0	PX 0 = '1' จะเป็นการกำหนดลำดับความสำคัญในการอินเตอร์รัพท์ภายนอกที่ขา INT 0 ในระดับสูง PX 0 = '0' จะเป็นการกำหนดลำดับความสำคัญในการอินเตอร์รัพท์ภายนอกที่ขา INT 0 ในระดับต่ำ

จากตารางที่ 5.3 จะเห็นว่าบิต PX 1 และ PX 0 ภายในรีจีสเตอร์ IP นั้นจะเป็นตัวจัดลำดับหรือกำหนดความสำคัญในการอินเตอร์รัพท์ภายนอกที่ขา INT 0 และ INT 1 โดยในกรณีที่มีสัญญาณอินเตอร์รัพท์ภายนอกเข้ามาพร้อมกันทั้งขา INT 0 และ INT 1 ขาที่ถูกทำการจัดลำดับความสำคัญ ระดับสูงกว่าจะทำงานก่อนหลังจากนั้น อินเตอร์รัพท์ภายนอกที่ถูก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดระดับความสำคัญที่ต่ำกว่าจึงจะทำงานต่อไป ก่อนที่จะเข้าสู่โปรแกรมหลัก(Main Program) หรือทำงานต่อไป ตามปกติ

ส่วนกรณีที่มีการจัดลำดับความสำคัญเท่ากันและเกิดสัญญาณอินเตอร์รัพท์ภายนอกพร้อมกัน ลำดับในการทำงานของอินเตอร์รัพท์ จะถูกควบคุมหรือกำหนดโดยการจัดลำดับภายในตัว MCU โดยมีการแสดงลำดับในการอินเตอร์รัพท์ดังตารางที่ 5.4

ตารางที่ 5.4 แสดงการจัดลำดับการอินเตอร์รัพท์ ภายในตัว MCU

อินเตอร์รัพท์	ลำดับความสำคัญ
INT 0	ระดับสูงทำงานก่อน
INT 1	ระดับต่ำทำงานก่อน

จากที่ผ่านมาผู้อ่านคงจะรู้จักการทำงานหรือรายละเอียดต่างๆ ของบิตใช้งานและรีจีสเตอร์ที่เกี่ยวข้องกับการประยุกต์ใช้งานอินเตอร์รัพท์ภายนอกกันมาพอสมควรแล้ว ต่อไปจะเป็นการกล่าวถึงขั้นตอนการใช้งานอินเตอร์รัพท์ภายนอก โดยมีการสรุปลำดับขั้นตอนในการใช้งานอินเตอร์รัพท์ภายนอกเป็นขั้นตอน (Step) ดังนี้

STEP 1

เริ่มต้นด้วยการเซตบิต EA โดยจะต้องเป็น 1 เพื่อบอกให้ CPU รู้ว่าจะใช้งานอินเตอร์รัพท์ภายนอก

STEP 2

เลือกการใช้งานอินเตอร์รัพท์ภายนอกที่ขาของ INT0 หรือ INT1 หรือทั้ง 2 ขา โดยเริ่จากเซตบิต EX0 หรือ EX1 ตามความต้องการที่จะใช้งาน

STEP 3

จากที่ผ่านมา 2 ขั้นตอนแรก เป็นการบอกให้ CPU รู้ว่าจะใช้งานอินเตอร์รัพท์ภายนอกและใช้งานขา INT0 และ/หรือ INT1 จากนั้นเราจะต้องทำการเซตบิต IT0 สำหรับการอินเตอร์รัพท์ภายนอกด้วยสัญญาณในรูป Active Low โดยถ้าไม่ทำการเซตบิต IT0 , IT1 ดังกล่าวแล้วจะทำให้ไมโครคอนโทรลเลอร์รอรับสัญญาณอินเตอร์รัพท์ภายนอกในรูปแบบ Low Signal คือสภาวะ Logic '0' เป็นระยะเวลาหนึ่ง ฉะนั้นถ้าเกิดสัญญาณ Logic '0' เป็นระยะเวลานานก็เท่ากับว่าเป็นการเรียกการอินเตอร์รัพท์นั้นซ้ำๆหลายๆครั้งนั่นเอง จึงแนะนำให้ทำการเซตบิต IT0 และ IT1 ทุกครั้งที่มีการใช้งานอินเตอร์รัพท์ภายนอก

อย่าลืมว่าการเซตบิต IT0 และ IT1 แล้วเมื่อจะใช้งานอินเตอร์รัพท์ภายนอก ขา INT0 และ INT1 จะต้องมรสภาวะ High แล้วจึงเปลี่ยนเป็นสภาวะ Low จึงจะเกิดการอินเตอร์รัพท์ ในกรณีเริ่มต้นขา INT0 และ INT1 มีสภาวะ Low แล้วมีสัญญาณเปลี่ยนจากสภาวะ Low ไปเป็น

เอกสารนี้ใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สภาวะ High แล้วกลับสู่สภาวะ Low ไมโครคอนโทรลเลอร์ MCS-51 จะไม่ถือว่าการเปลี่ยนแปลงสัญญาณข้างต้นนั้นเป็นการอินเตอร์รัพท์ หรือไม่รับรู้การอินเตอร์รัพท์ภายนอก จากการเปลี่ยนแปลงสัญญาณในลักษณะนี้ เพราะฉะนั้นการใช้งานอินเตอร์รัพท์ภายนอกควรจะคำนึงพิจารณาถึงจุดนี้ด้วย

STEP 4

ทำการจัดลำดับความสำคัญการอินเตอร์รัพท์ภายนอกโดยควบคุมบิต PX 0 และ PX 1 (ตารางที่ 5.3)

จาก 4 ขั้นตอนการใช้งานอินเตอร์รัพท์ภายนอก INT0 , INT1 ที่ผ่านมา จะเป็นเพียงการเซตค่าต่างๆ เพื่อเริ่มต้นการใช้งานอินเตอร์รัพท์ภายนอกโดยต่อไปจะเป็นการกล่าวถึงหลังจากเกิดการอินเตอร์รัพท์ขึ้นแล้ว จะมีขั้นตอนการทำงานภายในโปรแกรมอย่างไรต่อไป

ตารางที่5.5 แสดงตำแหน่งแอดเดรสของหน่วยความจำหลังจากถูกอินเตอร์รัพท์ภายนอก

อินเตอร์รัพท์	ตำแหน่งแอดเดรสของหน่วยความจำหลังจากถูกอินเตอร์รัพท์
INT0 (IE0)	กระโดดไปยังตำแหน่ง 0003H
INT1 (IE1)	กระโดดไปยังตำแหน่ง 0013H

จากตารางที่ 5.5 จะเห็นว่าถ้าเกิดการอินเตอร์รัพท์ภายนอกขึ้นที่ขา INT0 ตำแหน่งหน่วยความจำภายในไมโครคอนโทรลเลอร์จะกระโดดไปอ่านโปรแกรมหรือข้อมูล ณ ตำแหน่ง 0003H ของหน่วยความจำโปรแกรมทันที (EPROM) จากนั้นจะกะทำตามโปรแกรมที่ได้เขียนไว้ ณ ตำแหน่ง 0003H ซึ่งโดยส่วนมากจะเขียนคำสั่งในการกระโดดไปยังตำแหน่งที่ต้องการต่อไป เช่น คำสั่ง JMP เป็นต้น

กรณีที่เกิดการอินเตอร์รัพท์ภายนอกที่ขา INT1 เช่นกัน ตำแหน่งหน่วยความจำภายในไมโครคอนโทรลเลอร์จะกระโดดไปอ่านโปรแกรมหรือข้อมูล ณ ตำแหน่ง 0013H ของหน่วยความจำโปรแกรมทันที (EPROM) จากนั้นจะกระทำตามโปรแกรมที่ได้เขียนไว้ ณ ตำแหน่ง 0013H ต่อไป ซึ่งโดยส่วนมากจะเขียนคำสั่งในการให้กระโดดไปยังตำแหน่งที่ต้องการหลังจากเกิดการอินเตอร์รัพท์ขึ้นต่อไป เช่นคำสั่ง JMP เป็นต้น

สิ่งที่ไม่ควรลืม นั่นคือจากตำแหน่งแอดเดรสต่างๆ ที่เกิดขึ้นหลังจากการอินเตอร์รัพท์สังเกตุว่าจะอยู่ในช่วงต้นๆของแอดเดรสของหน่วยความจำ ฉะนั้นเวลาใช้งานอินเตอร์รัพท์ภายนอกนั้น ในการเขียนโปรแกรมเริ่มต้นที่ตำแหน่งเริ่มต้นของหน่วยความจำโปรแกรม (ORG 0000H) ควรจะใช้คำสั่งในการกระโดดไปยังส่วนของ Main Program หรือควรจะเขียน Main Program ที่ตำแหน่งเริ่มต้นหลังจากตำแหน่งของการอินเตอร์รัพท์เช่น เขียน Main Program ไว้ตำแหน่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประมาณ 0050H (ORG 0050H) เพื่อเว้นพื้นที่ของหน่วยความจำหรือแอดเดรส เพื่อที่ Main Program จะไม่ไปซ้อนทับกับตำแหน่งแอดเดรสในส่วนของการอินเตอร์รัพท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
พอร์ตสื่อสารอนุกรม UART

จากบทที่ 2 ที่ผ่านมา ได้อธิบายถึงการจัดโครงสร้างตลอดจนคุณสมบัติของชิปไมโครคอนโทรลเลอร์ MCS-51 กันมาแล้ว ผู้อ่านจะพบว่าไมโครคอนโทรลเลอร์ตระกูล 8051 นี้จะมีพอร์ตสื่อสารข้อมูลอนุกรม UART (Universal Asynchronous Receiver Transmitter) อยู่ภายในชิปของ MCU ไปประยุกต์ใช้งานในการติดต่อสื่อสารข้อมูลหรือสามารถใช้ในงานควบคุมการทำงานของอุปกรณ์ภายนอกได้สะดวกและมีประสิทธิภาพมากยิ่งขึ้น

6.1 พอร์ตสื่อสารข้อมูลอนุกรม UART ภายในชิป MCS-51

โดยในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 นั้น MCS-51 สามารถที่จะให้มีการเซ็ตหรือกำหนดอัตราความเร็วในการติดต่อหรือรับส่งข้อมูล (Baud Rate) ได้ เพื่อยืดหยุ่นในการใช้งานในการติดต่อสื่อสารข้อมูลได้ครอบคลุมหลายรูปแบบ ทั้งการติดต่อสื่อสารระหว่าง MCU ด้วยกันเองหรือจะติดต่อสื่อสารผ่านพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ PC กับ MCU ก็สามารถกระทำได้ โดยรูปแบบในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมของ MCS-51 นั้น สามารถที่จะจัดรูปแบบการใช้งานได้ถึง 4 รูปแบบ โดยแต่ละรูปแบบดังกล่าวจะสามารถควบคุมเลือกการใช้งานได้โดยรีจิสเตอร์ใช้งานเฉพาะ SCON ภายในชิปซึ่งรูปแบบในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรม UART ภายในชิปจะมีลักษณะเป็นโหมดๆดังนี้

โหมด 0

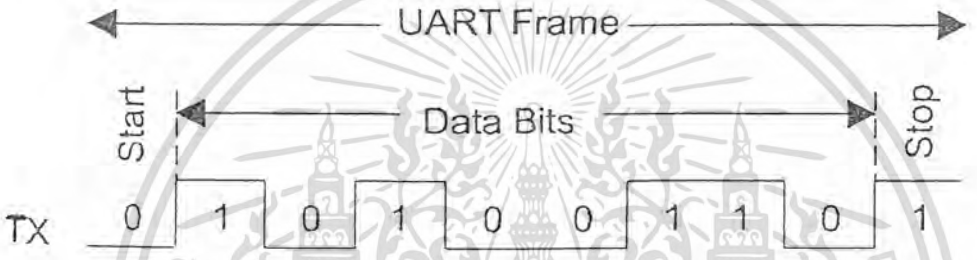
การทำงานของพอร์ตอนุกรม UART ในโหมด 0 นี้จะเป็นการใช้งานขา P3.0 (RXD) ของตัว MCU สำหรับการรับส่งข้อมูลและใช้ขา P3.1 (TXD) ในการผลิตสัญญาณนาฬิกา (Clock) เพื่อให้อุปกรณ์ภายนอกรับรู้จังหวะการรับส่งข้อมูลที่ขา RXD นั่นคือสัญญาณนาฬิกาที่ขา TXD จะเป็นตัวกำหนดการเลื่อน (Shift) ของข้อมูลที่ขา RXD นั่นเอง (ในการรับส่งข้อมูลที่ขา RXD จะใช้สัญญาณนาฬิกาที่ขา TXD เป็นตัวกำหนดจังหวะรับส่งข้อมูล) โดยข้อมูลที่ขา RXD นั้นจะเป็นลักษณะของข้อมูล 8 บิต ซึ่งจะมีการส่งหรือรับข้อมูลที่บิตต่ำสุด (Least Significant Bit ; LSB) ของชุดข้อมูลก่อน แล้วจึงเรียงลำดับข้อมูลจนถึงบิตสูงสุด (Most Significant Bit ; MSB) จึงจะเป็นการสิ้นสุดของชุดข้อมูล ในโหมด 0 นี้จะไม่สามารถกำหนดอัตราความเร็วในการรับส่งข้อมูล (Baud Rate) ได้ ซึ่งอัตราในการส่งความเร็วในการรับส่งข้อมูลที่โหมด 0 นี้ จะกำหนดไว้ที่ 1/12 ของความถี่ X-TAL Oscillator ที่ใช้ โดยสามารถคำนวณได้ดังสมการ

$$\text{Baud Rate โหมด 0} = \text{X-TAL} / 12$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหมด 1

การทำงานของพอร์ตอนุกรม UART ในโหมดที่ 1 นี้มีลักษณะของชุดข้อมูลจำนวน 10 บิต ดังแสดงในรูปที่ 6.1 โดยจะมีจุดเริ่มต้นของชุดข้อมูล (Start Bit มีค่าเป็น 0 เสมอ) ตามมาด้วยข้อมูลจำนวน 8 บิต และบิตสุดท้ายของชุดข้อมูล (Stop Bit มีค่าเป็น 1 เสมอ) ซึ่งชุดข้อมูลจะถูกส่งออกที่ขา TXD ของตัว MCU และรับชุดข้อมูลข้ามาภายใน MCU โดยขา RXD อัตราความเร็วในการรับส่งข้อมูล (Baud Rate) ของการทำงานในโหมด 1 นี้ สามารถกำหนดหรือเปลี่ยนแปลงได้โดยใช้ Timer ภายในตัว MCU เป็นตัวควบคุมซึ่งรายละเอียดในการใช้งาน Timer ในการควบคุมอัตราความเร็วในการรับส่งข้อมูลจะกล่าวในหัวข้อการประยุกต์ใช้งานพอร์ตสื่อสารอนุกรมต่อไป



รูปที่ 6.1 แสดงชุดข้อมูล UART ในโหมดที่ 1

โหมด 2

ในโหมด 2 นี้จะมีชุดข้อมูลที่ใช้ในการรับส่งข้อมูลแตกต่างจากชุดข้อมูลในโหมดที่ 1 คือจะมีชุดข้อมูลจำนวน 11 บิต ดังแสดงในรูปที่ 6.2 ซึ่งจะมีวิธีการในการรับและส่งข้อมูลผ่านทางขาของ MCU เช่นเดียวกันกับในโหมดที่ 1 โดยชุดของข้อมูลในโหมดที่ 2 นั้นจะประกอบไปด้วยบิตแรก (Start Bit มีค่าเป็น 0 เสมอ) และตามมาด้วยข้อมูลจำนวน 8 บิต จากนั้นจะมีบิตที่ 9 เพิ่มขึ้นมาโดยจะเป็นบิตที่สามารถ จะเซตค่าให้เป็น 0 หรือ 1 ได้ ซึ่งใช้ในการตรวจสอบความผิดพลาดของข้อมูลชุดในการรับและส่งนั่นเอง (Parity Bit) จากนั้นจะเป็นบิตสุดท้ายของชุดข้อมูล (Stop Bit มีค่าเป็น 1 เสมอ) โดยโหมด 2 นี้จะไม่สามารถกำหนดอัตราความเร็วในการรับส่งข้อมูล (Baud Rate) ได้เช่นเดียวกันกับในโหมด 0 ซึ่งอัตราความเร็วในการรับส่งข้อมูลในโหมด 2 นี้ จะถูกกำหนดไว้ที่ 1/32 หรือ 1/64 ของความถี่ X-TAL Oscillator ที่ใช้ โดยจะมีบิต SMOD ภายในรีจีสเตอร์ใช้งานเฉพาะ PCON เป็นตัวกำหนดดังนี้

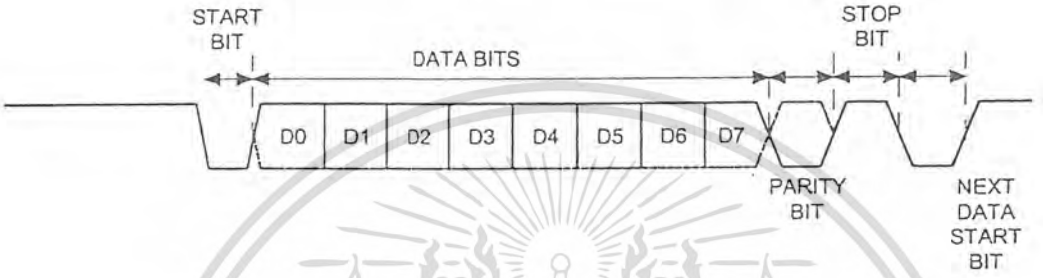
- ถ้าค่าบิต SMOD เป็น 0 ค่า Baud Rate จะถูกกำหนดเป็น 1/64 ของความถี่ X-TAL Oscillator ที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้าค่าบิต SMOD เป็น 1 ค่า Baud Rate จะถูกกำหนดเป็น 1/32 ของความถี่ X-TAL Oscillator ที่ใช้

ซึ่งมีสมการในการคำนวนดังนี้

Baud Rate 2 = X-TAL / 32OR64 , SMOD = 1 OR 0



รูปที่ 6.2 แสดงชุดข้อมูล UART ในโหมดที่ 2

โหมด 3

จะเป็นโหมดใช้งานในโหมดสุดท้ายของการใช้งานพอร์ตสื่อสารข้อมูล อนุกรม UART ภายในชิป MCS-51 ในโหมด 3 นี้ จะมีชุดข้อมูลในการรับและส่งข้อมูลจำนวน 11 บิต ซึ่งจะมีการรับและส่งผ่านทางขาของ MCU เช่นเดียวกันกับในโหมดที่ 1 โดยองค์ประกอบของชุดข้อมูลจะเหมือนกับชุดข้อมูลในโหมดที่ 2 ทุกประการ แต่ในโหมดที่ 3 นี้ จะสามารถกำหนดอัตราความเร็วในการรับส่งข้อมูลได้เช่นเดียวกันกับในโหมดที่ 1 โดยใช้ Timer ภายใน MCU เป็นตัวควบคุม

จากที่ผ่านมาผู้อ่านคงจะได้รู้จักโหมดการทำงานของพอร์ตสื่อสารอนุกรม UART ภายในชิป MCS-51 กันพอสมควร โดยในการใช้งานในโหมดการทำงานข้างต้นดังที่กล่าวมาจะนิยมใช้พอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 ในการใช้งานเนื่องจากในโหมดที่ 1 นั้นจะมีลักษณะของชุดข้อมูลที่เป็นมาตรฐานทั่วไป และสามารถเซ็ตหรือกำหนดอัตราความเร็วในการสื่อสารข้อมูลได้จากคุณสมบัติหรือเหตุผลดังกล่าว เนื้อหาในการประยุกต์ใช้งานพอร์ตสื่อสารข้อมูลที่จะกล่าวถึงต่อไป จะอธิบายถึงการใช้งานพอร์ตสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 ในเบื้องต้น

ก่อนที่จะเข้าถึงการประยุกต์ใช้งานพอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 ต่อไปนั้น ผู้เขียนจะขอแนะนำรีจิสเตอร์ใช้งานเฉพาะต่างๆ ที่ใช้ในการควบคุมหรือกำหนดค่าต่างๆ เพื่อการใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

งานพอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 และ ไทม์เมอร์ (Timer) เพื่อการกำหนดอัตราเร็วในการรับส่งข้อมูลอนุกรม ดังนี้

รีจีสเตอร์ใช้งานเฉพาะ SBUF : Serial Data Buffer (Not Bit Address)

เป็นรีจีสเตอร์ที่ใช้สำหรับการรับส่งสื่อสารชุดข้อมูลอนุกรม โดยจะทำหน้าที่เป็นบัฟเฟอร์ (Buffer) ขนาด 8 บิตไม่สามารถเข้าถึงในระดับบิต ซึ่งรีจีสเตอร์ SBUF นี้ จะมีหน้าที่ทั้งในการรับและส่งชุดข้อมูลอนุกรมโดยตัว CPU จะทำการควบคุมการใช้งานรีจีสเตอร์ SUBP ทั้งในการรับและส่งชุดข้อมูลโดยอัตโนมัติ

รีจีสเตอร์ใช้งานเฉพาะ SCON : Serial Port Control Register (Bit Address)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ภายในรีจีสเตอร์ต่างๆ เกี่ยวกับไมโครคอนโทรลเลอร์ MCS-51 นั้น ผู้อ่านคงพอจะรู้แล้วว่าภายใน 1 รีจีสเตอร์ที่สามารถเข้าถึงข้อมูลได้ในระดับบิตนั้น นั่นก็คือ 1 ไบต์ซึ่งจะประกอบไปด้วยบิตต่างๆ จำนวน 8 บิต รีจีสเตอร์ใช้งานเฉพาะ SCON นั่นก็เช่นกัน คือจะมีบิตต่างๆ จำนวน 8 บิต ที่สามารถเข้าถึงได้ระดับบิตที่ใช้ในการควบคุมหรือกำหนดค่าต่างๆในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรม

โดยบิตที่เราจำเป็นจะต้องรู้จักเพื่อการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 ภายในรีจีสเตอร์ SCON จะแสดงไว้ในตาราง 6.1

ตารางที่ 6.1 แสดงบิตภายในรีจีสเตอร์ SCON เพื่อการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมโหมดที่ 1

ชื่อบิต	รายละเอียด
SM0	บิตเลือกโหมดการใช้งานพอร์ตสื่อสารข้อมูลอนุกรม 'ต่ำ'
SM1	บิตเลือกโหมดการใช้งานพอร์ตสื่อสารข้อมูลอนุกรม 'สูง'
SM2	บิตเลือกโหมดการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมโดย SM2 = 0 จะเป็นการใช้งานโหมด 1 และ 3 ปกติ และบิต RI จะเป็น 0 จนกว่าจะรับบิตสุดท้ายของชุดข้อมูลเป็นที่เรียบร้อย
REN	REN = 1 เป็นการเซ็ตให้ตัว CPU รับข้อมูลจากภายนอกได้ REN = 0 เป็นการเซ็ตให้ตัว CPU ไม่รับข้อมูลจากภายนอก
TI	บิตบอกสถานะในการส่งข้อมูลอนุกรมทำงานโดยอัตโนมัติ TI = 0 จะแสดงว่าเป็นช่วงจังหวะในขั้นตอนการส่งชุดข้อมูลออกทางพอร์ตสื่อสารข้อมูลอนุกรม TI = 1 จะแสดงว่าชุดข้อมูลถูกส่งออกไปเป็นที่เรียบร้อยแล้วพร้อมที่จะส่งข้อมูลชุด ต่อไป โดยก่อนที่จะส่งข้อมูลชุดต่อไปนั้นต้องทำการเคลียร์บิต TI ใหม่ก่อนทุกครั้ง
RI	บิตบอกสถานะในการรับข้อมูลอนุกรมทำงานโดยอัตโนมัติ RI = 0 เป็นการรอรับชุดข้อมูลและถ้าข้อมูลถูกรับเรียบร้อยแล้ว บิต RB8 ของรีจิสเตอร์ SCON จะมีสภาวะ 1 พร้อมๆ กับชุดข้อมูล 8 บิตจะถูกเก็บไว้ที่รีจิสเตอร์ SBUF และบิต RI จะกลับเป็น 1 โดยอัตโนมัติ จากนั้นจะต้องเคลียร์บิต RI เพื่อที่จะรับข้อมูลชุดใหม่ต่อไป RI = 1 เป็นการแสดงว่าให้รับข้อมูลที่ส่งมาไปเก็บไว้ในรีจิสเตอร์ SBUF โดยก่อนหน้านี้ต้องเคลียร์บิต RI ก่อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยจากตารางที่ 6.1 จะพบว่าบิต TI และ RI จะสามารถทำการเซตหรือเคลียร์ได้ทั้งทางซอฟต์แวร์หรือตัวของ CPU ภายใน MCS-51 เอง จากนั้นจะเห็นว่าบิต SM0 และ SM1 นั้นจะเป็นบิตควบคุมในการเลือกโหมดการใช้งานพอร์ตสื่อสารข้อมูลอนุกรม โดยรายละเอียดการใช้งานหรือตารางความจริงของบิต SM0 และ SM1 แสดงดังตารางที่ 6.2

ตารางที่ 6.2 แสดงบิตในการควบคุมโหมดการใช้งานด้วยบิต SM0 และ SM1

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR FOSC./32
1	1	3	9-Bit UART	Variable

6.2 ไทม์เมอร์ในการกำหนดอัตราเร็วในการรับส่งข้อมูล

ไทม์เมอร์ภายในตัวไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะประกอบไปด้วยไทม์เมอร์ 0 และไทม์เมอร์ 1 และอาจจะมีไทม์เมอร์ 2 ในบางเบอร์ของ MCS-51 โดยการกำหนดอัตราเร็วในการรับส่งข้อมูล(Baud Rate) เพื่อการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 นั้นจะใช้ไทม์เมอร์ 1 เป็นตัวกำหนด Baud Rate ซึ่งการใช้งานไทม์เมอร์นั้นจะถูกควบคุมด้วยรีจิสเตอร์ใช้งาน TMOD โดยมีรายละเอียดดังต่อไปนี้

รีจิสเตอร์ใช้งานเฉพาะ TMOD : Timer Counter Mode ControlRegister (Notbit Addressable)

Timer 1				Timer 0			
SM0	SM1	SM2	REN	TB8	RB8	T1	R1

ภายในรีจิสเตอร์ TMOD จะประกอบไปด้วยบิตควบคุมการใช้งานต่างๆจำนวน 8 บิตจากข้างต้นจะพบว่ามีบิตในการใช้งานไทม์เมอร์ 1 และไทม์เมอร์ 0 โดย 4 บิตบนจะควบคุมการใช้งานไทม์เมอร์ 1 และ 4 บิตล่างจะเป็นชุดบิตที่ควบคุมไทม์เมอร์ 0 ซึ่งทั้ง 4 บิตบนและล่างจะมีลักษณะการใช้งานเช่นเดียวกันรายละเอียดของชุดบิตดังที่กล่าวมา เพื่อการใช้งานไทม์เมอร์ในการกำหนดอัตราเร็วในการรับส่งข้อมูลผ่านพอร์ตอนุกรมแสดงดังตาราง ที่ 6.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.3 แสดงบิตในการควบคุมการใช้งานไทม์เมอร์ 1

ชื่อบิต	รายละเอียด
GATE	เท่ากับ 0 จะสามารถเลือกใช้งานไทม์เมอร์หรือเคาน์เตอร์ได้
C/T	บิตเลือกใช้งานไทม์เมอร์หรือเคาน์เตอร์โดย เท่ากับ 0 หมายถึง ใช้งานเป็นเค้าน์เตอร์นับพัลส์ภายนอกที่ขา TX (T0,T1) เท่ากับ 1 หมายถึงใช้เป็นไทม์เมอร์นับการทำงานของแมชชีนไวเกิล
M0	บิตเลือกโหมดการใช้งานไทม์เมอร์ 0 และ ไทม์เมอร์ 1
M1	บิตเลือกโหมดการใช้งานไทม์เมอร์ 0 และ ไทม์เมอร์ 1

การควบคุมบิต TRX (TR0 , TR1) ทางซอฟต์แวร์นั้นบิต TRX จะอยู่ภายในรีจีสเตอร์ TCON ซึ่งสามารถเข้าถึงได้ในระดับบิตโดยถ้า TR1 =1 จะเป็นการใช้งานไทม์เมอร์ 1 และถ้า TR0 = 1 จะเป็นการใช้งานไทม์เมอร์ 0 ซึ่งนั่นก็คือบิต TR0 และ TR1 ภายในรีจีสเตอร์ใช้งานเฉพาะ TCON จะเป็นตัวควบคุมการใช้งานไทม์เมอร์ 0 และไทม์เมอร์ 1 ตามลำดับ

โดยจากตารางที่ 6.3 จะเห็นว่าบิต M0 และ M1 นั้นจะใช้เป็นบิตในการควบคุมหรือเลือกโหมดการทำงานไทม์เมอร์ ซึ่งสามารถสรุปตารางความจริงของบิต M0 และ M1 ได้ แสดงดังในตารางที่ 6.2

การใช้งานไทม์เมอร์ 1 เพื่อการกำหนดอัตราเร็วในการสื่อสารข้อมูลอนุกรมนั้นจะใช้งานไทม์เมอร์ 1 ในโหมดที่ 2 โดยจากตารางที่ 6.2 นั่นก็คือ บิต M1 ถูกเซตให้เป็น 1 และบิต M0 ให้เป็น 0 ในส่วนของรายละเอียดของการทำงานภายในของไทม์เมอร์ 1 ในโหมดที่ 2 หรือในโหมดอื่นๆ นั้น จะไม่ขอแสดงรายละเอียด เนื่องจากเป็นขั้นตอนการทำงานภายในโดยตัว CPU เป็นตัวควบคุมการทำงานทั้งหมด

จากที่กล่าวมาการใช้งานไทม์เมอร์ 1 ในโหมดที่ 2 เพื่อการกำหนดอัตราเร็วในการสื่อสารข้อมูลอนุกรมนั้น จะมีบิต SMOD ภายในรีจีสเตอร์ใช้งานเฉพาะ PCON เป็นตัวกำหนดอัตราการเกิดโอเวอร์โฟลว์ (Over Flow) ให้กับไทม์เมอร์ 1 โดยลักษณะชุดข้อมูลของรีจีสเตอร์ PCON มีดังนี้

รีจีสเตอร์ PCON : Power Control Register (Not Bit Address)

SMOD	-	-	-	GF1	GF0	PD	IDL

ข้างต้นเป็นลักษณะรูปแบบชุดข้อมูลของ รีจีสเตอร์ PCON จะเห็นว่าบิต SMOD จะอยู่ในตำแหน่งบิตสูงสุด (Most Significant Bit ; MSB) ของชุดข้อมูล 8 บิต และรีจีสเตอร์ PCON นี้ จะไม่สามารถเข้าถึงข้อมูลในระดับบิตได้ฉะนั้นการเซตค่า SMOD ในการควบคุมการเกิดโอเวอร์โฟวล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ให้ไทม์เมอร์ 1 นั้นจะสามารถระทำได้โดยการส่งชุดข้อมูลครั้งละ 8 บิต เข้าไปในรีจีสเตอร์ PCON ดังตัวอย่าง

MOV PCON,#80H ; บิต SMOD เท่ากับ 1

MOV PCON,#00H ; บิต SMOD เท่ากับ 0

รายละเอียดการใช้งานบิต SMOD นั้นจะเป็นบิตในการควบคุมอัตราความเร็วในการรับส่งข้อมูล (Baud Rate) โดยถ้าหากถูกเซตหรือบิต SMOD มีค่าเป็น 1 จะทำให้มีการเพิ่มอัตราความเร็วในการรับส่งข้อมูลเป็น 2 เท่านั่นเอง

จากที่กล่าวมาแล้วว่ารูปแบบในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมของ MCS-51 นั้นสามารถที่จะจัดรูปแบบการใช้งานได้ถึง 4 โหมด โดยจะมีเพียงโหมดที่ 1 และ 3 เท่านั้น ที่สามารถกำหนดอัตราความเร็วในการรับส่งข้อมูลได้ ซึ่งสามารถสรุปเป็นสมการการคำนวณอัตรความเร็วในการรับส่งข้อมูลได้ดังสมการที่ 1

$$\text{Baud Rate} = (K \times \text{oscillator Frequency}) / (32 \times 12 \times [256-(TH1)]) \qquad \text{(สมการที่ 1)}$$

จากสมการที่ 1 ค่า K จะมีค่าเท่ากับ 2^{SMOD} โดยค่าอัตราความเร็วในการรับส่งข้อมูลนั้นเราจะทำการกำหนดขึ้นมาได้ตามต้องการแต่โดยมาตรฐานทั่วไปจะใช้อัตราความเร็ว 4,800 หรือ 9,600 บิตต่อวินาที (bps) ฉะนั้นตัวแปรที่เราต้องการหาจากสมการที่ 1 ก็คือค่าของ TH1 ซึ่งสามารถเปลี่ยนสมการที่ 1 ได้ใหม่เพื่อการคำนวณหาค่าของ TH1 ให้ได้ตามค่า Baud Rate ตามที่เราต้องการได้ใหม่ดังสมการที่ 2

$$TH = 256 - [(K \times \text{Oscillator Frequency}) / (3\,84 \times \text{Baud Rate}) \qquad \text{(สมการที่ 2)}$$

จากสมการที่ 2 สามารถสรุปและแสดงตัวอย่างค่าต่างๆ ของ TH1 ที่สามารถกำหนดอัตราความเร็วในการรับส่งข้อมูลมาตรฐาน(Baud Rare) ดังแสดงในตารางที่ 6.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.4 แสดงค่าพารามิเตอร์ต่างๆ ในการใช้งานพอร์ตสื่อสารอนุกรมโหมดที่ 1

Baud Rate	Crystal Frequency	SMOD	TH1 Reload Value	Actual Baud Rate	Error
9600	12.000 MHz	1	-7(F9H)	8923	7%
2400	12.000 MHz	0	-13(F3H)	2404	0.16%
1200	12.000 MHz	0	-26(E6H)	1202	0.16%
9300	11.059 MHz	1	-3(FDH)	19200	0
9600	11.059 MHz	0	-3(FDH)	9600	0
2400	11.059 MHz	0	-12(F4H)	2400	0
1200	11.059 MHz	0	-24(E8H)	1200	0

จากตารางที่ 6.4 จะเห็นว่าในการใช้งาน X-TAL Oscillator ที่ความถี่ 11.059 MHz นั้นจะสามารถกำหนดอัตราความเร็วในการรับส่งข้อมูล (Baud Rate) ได้ค่า Actual Baud Rate ที่ลงตัวในการใช้งานพอร์ตสื่อสารข้อมูลอนุกรมในโหมดที่ 1 โดยใช้งานไทม์เมอร์ 1 ในโหมดที่ 2 เป็นตัวควบคุมหรือกำหนดอัตราความเร็ว (Baud Rate) ที่ไม่มีความผิดพลาด (Error) เกิดขึ้นเลยหรือเป็นไปตามทฤษฎีนั่นเอง โดยจะเห็นว่าเมื่อต้องการกำหนดอัตราความเร็วในการรับส่งข้อมูล ที่อัตราความเร็ว 9,600 บิตต่อวินาที (bps) นั้น จะต้องทำการเคลียร์บิต SMOD (ถูกเซตเป็น 0) และทำการเซตค่าของรีจีสเตอร์ใช้งานเฉพาะ TH1 ให้มีค่าเป็น FDH(253 = FDH จากการคำนวนในสมการที่ 2) จากนั้นก็ทำการเซตบิต TR1 (เซตเป็น1) เพื่อเริ่มต้นการใช้งานไทม์เมอร์ 1 ก็จะได้อัตราเร็วในการรับส่งข้อมูล 9,600 บิตต่อวินาที (bps) ตามต้องการ

ซึ่งอัตราเร็วในการรับส่งข้อมูลที่กล่าวถึงมาโดยตลอดนั้น จะใช้กับชุดข้อมูลที่มีขนาด 10 บิต (ประกอบด้วย Srart Bit , ข้อมูล 8 บิต และ Stop Bit) ซึ่งเป็นชุดข้อมูลมาตรฐานในการคำนวณ Baud Rate ดังที่กล่าวมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

Visual Basic

ในปัจจุบัน ระบบปฏิบัติการ (Operating System) ในลักษณะของ Window ได้เข้ามาแทนที่ระบบปฏิบัติการในลักษณะเดิม ซึ่งส่วนใหญ่ที่นิยมใช้กันอยู่ก็คือ MS-DOS เนื่องจากรูปแบบของจอภาพที่ใช้ติดต่อระหว่างคอมพิวเตอร์ และผู้ใช้ อยู่ในรูปแบบของ Graphic User Interface (GUI) ที่ใช้รูปภาพแทนคำสั่งต่างๆ แทน ซึ่งต่างจาก MS-DOS ที่รูปแบบของคำสั่งจะอยู่ในรูปแบบของตัวอักษร และเป็นแบบป้อนทีละบรรทัด หรือที่เรียกว่า " Command Line" ซึ่งผู้ใช้จะต้องเรียนรู้ และจดจำรูปแบบของแต่ละคำสั่งให้ถูกต้องและแม่นยำ จึงจะใช้งานโปรแกรมนั้นๆ ได้เป็นอย่างดี และด้วยเหตุนี้ ได้ส่งผลต่อการพัฒนาโปรแกรมเช่นเดียวกันเนื่องจากโปรแกรมเมอร์ ซึ่งแต่เดิมพัฒนาโปรแกรมอยู่บน MS-DOS ต้องเปลี่ยนแปลงรูปแบบและแนวความคิด และหันมาพัฒนาโปรแกรมบน Windows แทน

การพัฒนาโปรแกรมบน Windows ในปัจจุบัน กระทำได้ง่ายและสะดวกขึ้น เนื่องจากมีการใช้เทคโนโลยีทางด้าน Visualize เข้ามาประกอบในการออกแบบจอภาพ ซึ่งต่างจากในยุคแรกที่การพัฒนาโปรแกรมบน Windows นั้นค่อนข้างจะทำได้ยากเนื่องจากการพัฒนาโปรแกรมหนึ่งๆ ให้แล้วเสร็จ โปรแกรมเมอร์จะต้องเขียน Routine ต่างๆขึ้นเป็นจำนวนมาก ซึ่ง Visual Basic ก็จัดเป็นภาษาหนึ่งที่ได้รับความนิยม และถูกนำมาใช้ในการพัฒนาโปรแกรมเพื่อใช้งานบน Windows

7.1 ความเป็นมาของภาษา BASIC

ภาษา BASIC ถูกสร้างขึ้นในปี 1963 โดย Hohn Keneny และ Thomas Kurtz ที่วิทยาลัย Dartmouth ในเบื้องต้นพวกเขามีจุดมุ่งหมายในการพัฒนา BASIC ขึ้น เพื่อใช้ในการสอนแนวการเขียนโปรแกรม (Programming Concept) โดยเน้นให้รูปแบบของภาษานั้นง่ายต่อการเข้าใจและการใช้งาน รวมทั้งการทำงานในลักษณะของ Interpreter ซึ่งจะแตกต่างจากภาษาคอมพิวเตอร์อื่นๆ ในยุคนั้นที่จะอาศัย Job Control Language (JCL) และขั้นตอนในการ Ccompile และ Link ผลก็คือภาษา BASIC ได้กลายเป็นภาษาคอมพิวเตอร์ ที่นิยมใช้กันอย่างกว้างขวาง โดยเฉพาะกลุ่มคอมพิวเตอร์ส่วนบุคคลจึงอาจกล่าวได้ว่าภาษา BASIC ได้รับการพัฒนาควบคู่ไปกับการพัฒนาคอมพิวเตอร์ส่วนบุคคล ในปี 1970 Microsoft ได้เริ่มผลิตตัวแปรภาษา BASIC ใน ROM ซึ่งเรียกว่า ROM- Based BASIC ขึ้นเช่น ชิป Radio Sheek TRS-80 เป็นต้น ต่อมาได้พัฒนาเป็น GW-BASIC ซึ่งเป็น Interpreter ภาษาที่ใช้กับ MS-DOS และในปี 1982 Microsolf QuickBasic ได้รับการพัฒนาขึ้น โดนการเพิ่มความสามารถในการ Compile ให้เป็น Executed Program รวมทั้งทำให้ BASIC มีความเป็น "Structured Programming" ,มากขึ้น โดยการตัด Line Number ทิ้งไป เพื่อลบข้อกล่าวหาว่าเป็นภาษาคอมพิวเตอร์ที่มีโครงสร้างภาษาในลักษณะ Spaghetti Code (Logical flow ของภาษาขาดโครงสร้าง) มาใช้รูปแบบของ Subprogram และ User Defined รวมทั้งการใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Structured Data Type และการใช้งานด้านกราฟฟิกให้มีการใช้งานในลักษณะที่สูงขึ้น รวมทั้งมีการใช้เสียงประกอบได้เหมือนกับภาษาคอมพิวเตอร์อื่นๆ เช่น C หรือ Pascal

7.2 สาเหตุที่ต้องใช้ Visual Basic

Visual Basic เป็นภาษาคอมพิวเตอร์ที่ได้รับความนิยมนำมาใช้ในการพัฒนาโปรแกรมบน Windows เนื่องจากเป็นภาษาคอมพิวเตอร์ ที่ใช้เทคโนโลยีใยลักษณะ Visualize ซึ่งเพียงแต่เลือก Control ที่เหมาะสมแล้ววาดลงบน Form ก็สามารถสร้างจอภาพที่ใช้สำหรับติดต่อกับผู้ใช้ รวมทั้งเทคนิคการเขียนโปรแกรมแบบ Event-driven ซึ่งเป็นการเขียนโปรแกรมเพื่อกำหนดขั้นตอนการทำงาน ให้กับ Control ต่างๆ ที่สร้างขึ้นตามเหตุการณ์ (Event) ต่างๆที่เกิดขึ้น เช่น การเลื่อนเม้าส์ หรือการรับข้อมูลจากคีย์บอร์ด เป็นต้น ประกอบกับภาษาที่ใช้เขียนโปรแกรมเป็นภาษา BASIC ซึ่งเป็นภาษาคอมพิวเตอร์ที่ผู้ใช้คอมพิวเตอร์ส่วนบุคคลส่วนใหญ่คุ้นเคยจึงส่งผลให้ การพัฒนาโปรแกรมบน Windows ด้วย Visual Basic มีขั้นตอนน้อยกระทำได้ง่าย และสะดวกต่อการใช้งาน จึงทำให้ผู้ใช้สามารถเรียนรู้ได้ภายในเวลา 2-3 ชั่วโมง ก็สามารถพัฒนาโปรแกรมบน Windows ขึ้นเป็นโปรแกรมแรกได้

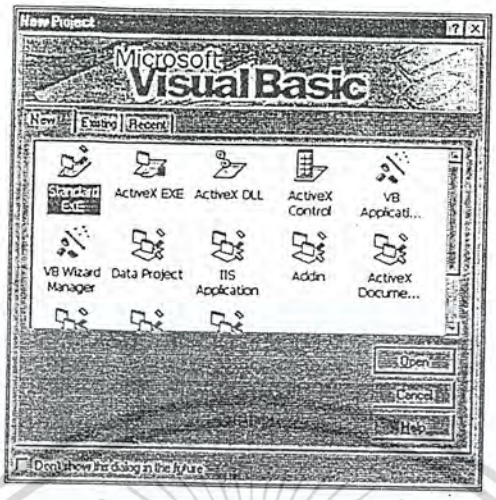
Visual Basic นี้เป็นเครื่องมือที่ใช้ในการพัฒนาโปรแกรมขึ้นใช้งาน ที่ใช้ได้ตั้งแต่ผู้ใช้ระดับต้น เพื่อใช้สร้างโปรแกรมง่ายๆบน Windows หรือโปรแกรมเมอร์ระดับกลาง ที่จะเรียกใช้ฟังก์ชั่นการทำงานต่างๆของ Visual Basic ได้อย่างมีประสิทธิภาพ ตลอดจนโปรแกรมเมอร์ในระดับมืออาชีพ ที่จะพัฒนาโปรแกรมในระดับสูง โดยการใช้ Object Linking and Embedding (OLE) และ Application Interface (API) ของ Windows มาประกอบในการเขียนโปรแกรม

7.3 เริ่มต้นกับ Visual Basic 6.0

สิ่งแรกที่จะพบเมื่อเข้าสู่โปรแกรม Microsoft Visual Basic Version 6.0 รุ่น Enterprise ได้แก่ จอภาพที่ใช้สำหรับเปิด Project (Project จะเป็นชื่อที่ใช้เรียกแทนระบบงานที่พัฒนาขึ้นด้วย Visual Basic ซึ่งรายละเอียดจะกล่าวถึงในลำดับต่อไป) ประกอบไปด้วย 3 Tab ดังนี้

1. Tab "New" เป็นจอภาพที่ประกอบไปด้วย Icon ต่างๆ ที่ใช้สำหรับเรียกใช้ Project ใหม่ขึ้นมาใช้งานดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

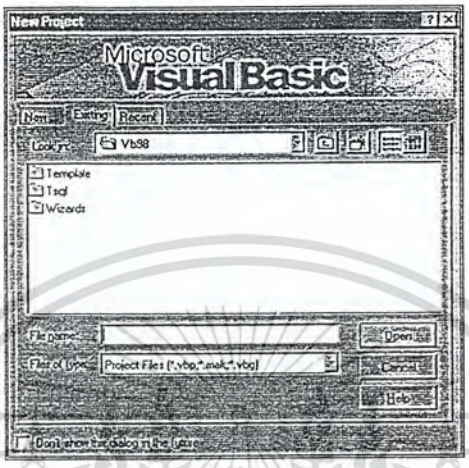


ซึ่งประกอบไปด้วย Icon ต่างๆ ดังนี้

ชื่อเรียก	รูปแบบการใช้งาน
Standard EXE	ใช้สร้างโปรแกรมในแบบ GUI โดยทั่วไป
ActiveX EXE	ใช้สร้างโปรแกรมที่ใช้ติดต่อกับโปรแกรมอื่นในรูปแบบของ OLE แบบหนึ่ง ซึ่งเรียกว่า OLE Automation Server
ActiveX DLL	ใช้สร้างโปรแกรมเช่นเดียวกับ ActiveX EXE แต่จะเก็บอยู่ใน File นามสกุล DLL แทน ซึ่งไม่สามารถ Run ได้ด้วยตัวเอง จะต้องถูกเรียกใช้โดยโปรแกรมอื่น
ActiveX Control	ใช้สร้าง ActiveX Control ขึ้นใช้งาน
VB Application Wizard	เป็นเครื่องมือที่ช่วยสร้างโปรแกรมขึ้นใช้งาน
Addin	ใช้เพิ่มเติม Utility อื่นเข้าไว้ใน Visual Basic
ActiveX Document DLL	ใช้สร้างโปรแกรม ActiveX ที่อยู่ในรูปของ File นามสกุล DLL
ActiveX Document EXE	ใช้สร้างโปรแกรม ที่อยู่ในรูปของ File นามสกุล EXE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

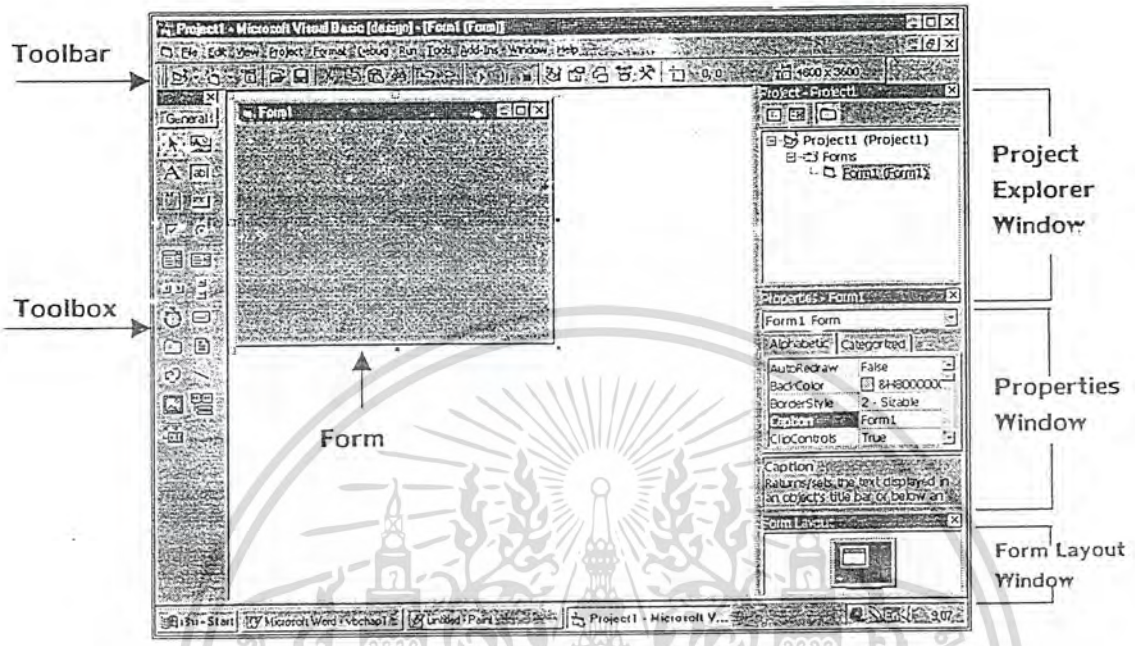
2. Tab "existing" เป็นจอภาพที่ใช้สำหรับเรียกใช้ Project เดิมที่พัฒนาขึ้นแล้วและเก็บไว้ใน Directory ต่างๆที่เคยถูกเรียกมาพัฒนาดังรูป



3. tab "Recent" เป็นจอภาพที่แสดงประวัติของ Project ต่างๆที่เคยถูกเรียกขึ้นมาพัฒนาดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในเบื้องต้นนี้ให้เลือก Icon "Standard EXE" ใน Tab "New" เพื่อเข้าสู่จอภาพของ Visual Basic ที่ใช้ในการพัฒนาโปรแกรม ดังรูป



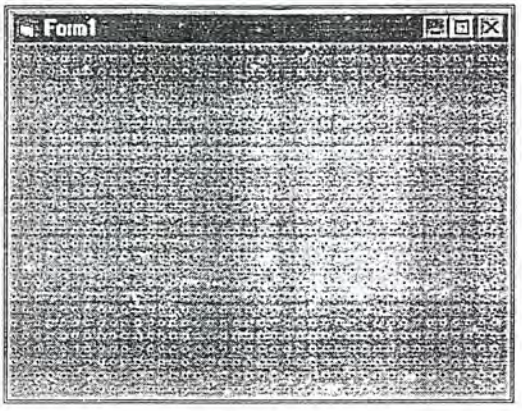
ส่วนประกอบของจอภาพ Visual Basic 6.0 มีดังนี้

ส่วนประกอบ	รายละเอียด
Form	เป็นส่วนที่ใช้สำหรับสร้างจอภาพของโปรแกรมขึ้นใช้งาน
Toolbox	เป็นส่วนที่ประกอบด้วย Icon ต่าง ๆ ซึ่งใน Visual Basic เรียกว่า "Control" ที่จะนำไปใช้งานโดยการนำลงไปวางบน Form สำหรับหน้าที่ของแต่ละ Control จะกล่าวถึงในลำดับต่อไป
Toolbar	เป็นส่วนที่ประกอบด้วย Icon ต่าง ๆ ที่ใช้ในการพัฒนาโปรแกรม
Project Explorer Window	เป็นส่วนสำหรับเรียก Form ต่าง ๆ ขึ้นมาแก้ไข ในกรณีที่ Project ประกอบด้วย Form มากกว่า 1 Form
Properties Window	เป็นจอภาพที่ใช้สำหรับกำหนดคุณสมบัติ (Property) ให้กับ Form และ Object ต่าง ๆ ที่ปรากฏอยู่บน Form
Form Layout Window	ใช้สำหรับดูตำแหน่งของ Form บนจอภาพ ทำให้จัดตำแหน่งของ Form ได้สะดวกขึ้น

7.4 Form

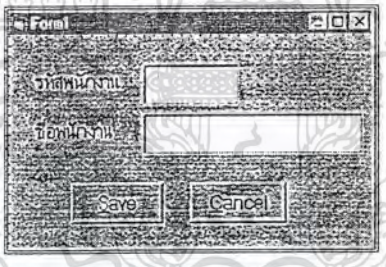
เป็นส่วนที่ใช้สำหรับสร้างจอภาพของโปรแกรม โดยจะทำหน้าที่เป็นพื้น (Background) ของจอภาพทุกครั้งของการเปิด Project ใหม่ จะได้ Form เปล่าดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



7.5 Toolbox

Toolbox เป็นแถบเครื่องมือที่ประกอบไปด้วย Icon ต่างๆ ซึ่งเรียกว่า "Control" เพื่อสร้างจอภาพของ Project แต่ละ Control จะใช้เป็นเครื่องมือที่ใช้สำหรับสร้างส่วนที่ใช้ติดต่อกับผู้ใช้ หรือที่เรียกว่า "User Interface" เช่น ข้อความต่างๆ ช่องว่างสำหรับข้อมูลจากคีย์บอร์ด ปุ่มต่างๆ เป็นต้น และถูกนำไปใช้งานด้วย โดยการนำ Control ที่ต้องการไปวางลงบน From ยกตัวอย่างเช่น จอภาพ ดังรูป

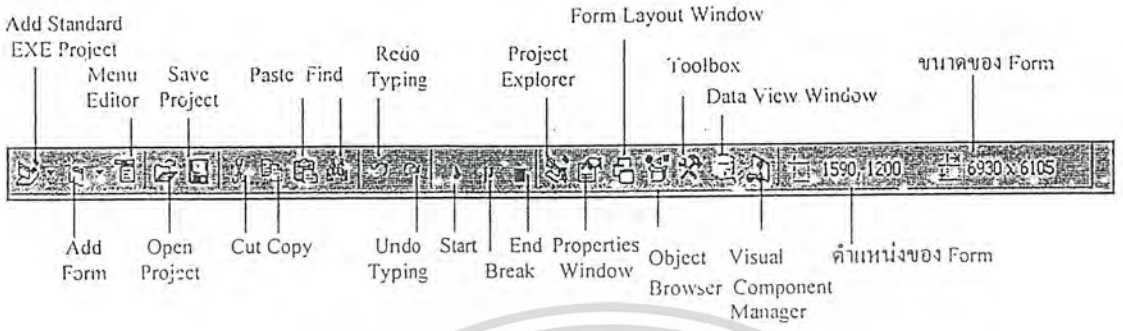


Control แต่ละตัวจะมีชื่อและหน้าที่แตกต่างกันไป เมื่อต้องการดูชื่อของ Control ใด ก็เพียงแต่เลื่อนเม้าส์ชี้ไปยัง Control นั้น ชื่อของ Control จะปรากฏขึ้นดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.6 Toolbar

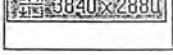
เป็นแถบเครื่องมือที่ประกอบด้วย Icon ต่างๆ ดังรูป



Toolbar ทำหน้าที่เป็นผู้ช่วยในการพัฒนาโปรแกรม ซึ่งเมื่อเลื่อนเมาส์ชี้ไปยัง Icon ใด ก็จะปรากฏชื่ออยู่ใต้ Icon นั้น แต่ละ Icon จะมีหน้าที่ต่างกันไปดังนี้

รูป Icon	ชื่อเรียก	รายละเอียดการใช้งาน
	Add Standard EXE-Project	ใช้สำหรับเปิด Project ใหม่ ในกรณีที่มีหลาย Project อยู่บนจอภาพ สามารถสลับไปมาระหว่าง Project ได้โดยเข้าไปเลือกในเมนู Windows
	Add Form	ใช้ในการเพิ่ม Form ให้กับ Project ซึ่งอาจเรียกจากเมนู Project และ Add Form ตามลำดับ
	Menu Editor	ใช้เรียก Menu Editor ซึ่งเป็น Tool สำหรับสร้างเมนูให้กับ Form ซึ่งอาจใช้การกดปุ่ม Ctrl+E หรือเลือกจากเมนู Tools และ Editor ตามลำดับ
	Open Project	ใช้สำหรับเปิด Project ซึ่งอาจใช้ Hot Keys Ctrl+O หรือเรียกจากเมนู File และ Open Project ตามลำดับ
	Save Project	ใช้สำหรับบันทึก Project และ Form ซึ่งอาจเลือกจากเมนู File และ Save Project หรือ Save Project As ตามลำดับ ถ้าต้องการระบุชื่อของ Project ใหม่
	Cut	ใช้สำหรับตัด Object ต่าง ๆ บน Form ซึ่งอาจใช้ Hot Keys Ctrl+X หรือเลือกจากเมนู Edit และ Cut ตามลำดับ
	Copy	ใช้สำหรับ Copy Object บน Form ซึ่งอาจใช้ Hot Keys Ctrl+C หรือเลือกจากเมนู Edit และ Copy ตามลำดับ สำหรับ Object ที่ถูก Copy จะเรียกว่า Control Array ซึ่งจะกล่าวถึงในส่วนต่อไป
	Paste	ใช้สำหรับ Paste Object ที่ Cut หรือ Copy ไว้ ซึ่งอาจใช้ Hot Keys Ctrl+V หรือเลือกจากเมนู Edit และ Paste ตามลำดับ
	Find	ใช้สำหรับค้นหาคำใน Editor ซึ่งเป็น Tool ที่ใช้ในการเขียนโปรแกรม โดยอาจใช้ Hot Keys Ctrl+F หรือเลือกจากเมนู Edit และ Find ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูป Icon	ชื่อเรียก	รายละเอียดการใช้งาน
	Undo Typing	ใช้สำหรับยกเลิกคำที่พิมพ์ใน Editor ซึ่งอาจใช้ Hot Keys Ctrl+Z หรือเลือกจากเมนู Edit และ Undo Typing ตามลำดับ ในกรณีที่ไม่ได้อยู่ในจอภาพ Editor จะปรากฏข้อความ Can't Undo แทนทั้งในส่วนของชื่อใต้ Icon และในเมนู Edit
	Redo Typing	ใช้สำหรับทำซ้ำคำที่พิมพ์ใน Editor ซึ่งอาจเลือกจากเมนู Edit และ Redo Typing ตามลำดับ และก็เช่นเดียวกับ Undo Typing ในกรณีที่ไม่ได้อยู่ในจอภาพ Editor ปรากฏข้อความ Can't Redo แทน
	Start	ใช้สำหรับ Run Project ที่จัดทำขึ้น ซึ่งอาจใช้ Hot Keys F5 หรือเลือกจากเมนู Run และ Start ตามลำดับ
	Break	ใช้สำหรับหยุดการทำงานของ Project ชั่วคราว ซึ่งอาจใช้ Hot Keys Ctrl+Break หรือเลือกจากเมนู Run และ Break ตามลำดับ
	End	ใช้สำหรับหยุดการทำงานของ Project ซึ่งอาจเลือกจากเมนู Run และ End ตามลำดับ
	Project Explorer	ใช้แสดงว่า Project นั้นประกอบไปด้วย Form และ Module ใดบ้าง ซึ่งอาจใช้ Hot Keys Ctrl-R หรือเลือกจากเมนู View และ Project Explorer ตามลำดับ
	Properties Window	ใช้สำหรับกำหนดคุณสมบัติ (Property) ของ Object และ Form ซึ่งอาจจะใช้ Hot Keys F4 หรือเลือกจากเมนู View และ Properties Window ตามลำดับ
	Form layout Window	ใช้สำหรับเรียกจอภาพ Form Layout ซึ่งใช้แสดงตำแหน่งของ Form บนจอภาพ โดยอาจเลือกจากเมนู View และ Form Layout Window ตามลำดับ
	Object Browser	ใช้สำหรับเรียกจอภาพ Object Browser ซึ่งใช้แสดงถึง Class และสมาชิกของแต่ละ Class อาจใช้ Hot Keys F2 หรือเลือกจากเมนู View และ Object Browser ตามลำดับ
	Toolbox	ใช้สำหรับเรียก Toolbox ขึ้นมาบนจอภาพ ซึ่งอาจเลือกจากเมนู View และ Toolbar ตามลำดับ
	ตำแหน่งของ Form	ใช้บอกตำแหน่งในแกน X และ Y ของ Form
	ขนาดของ Form	ใช้บอกขนาดของ Form ตามแนวแกน X และ Y

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.7 Project

โดยทั่วไประบบงานหนึ่งๆ มักจะประกอบไปด้วยหลายๆ จอภาพ เช่น จอภาพสำหรับป้อนข้อมูล (Data Entry) จอภาพสำหรับค้นหาข้อมูล (Data Inquiry) เป็นต้น ดังนั้นในการพัฒนาโปรแกรม จึงนิยมที่จะแยกแต่ละจอภาพออกเป็นโปรแกรม เพื่อความสะดวกต่อการแก้ไขตามหลักการเขียนโปรแกรมแบบ Modularity เช่น ภาษาตระกูล XBase ที่แยกออกเป็นไฟล์นามสกุล PRG หรือในภาษา BASIC ที่แยกออกเป็นไฟล์นามสกุล BAS เป็นต้น แล้วจึงนำแต่ละโปรแกรมย่อยมาประกอบกันขึ้นเป็นระบบ โดยการ Ccompile ไฟล์เหล่านั้นร่วมกันเป็น Eexecuted Program (ไฟล์นามสกุล EXE) เพื่อนำไปใช้งาน

ใน Visual Basic ก็เช่นเดียวกัน แต่ละจอภาพที่พัฒนาขึ้นจะได้แก่ Form ต่างๆ และเมื่อนำมารวมกันก็จะกลายเป็นระบบงานระบบหนึ่ง หรือที่เรียกว่า Project ดังนั้น Project และ Form จึงต้องทำงานร่วมกัน จะขาดไฟล์ใดไฟล์หนึ่งไม่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
การทำงานของเครื่องควบคุมซาวด์แลปและส่วนรับฟัง

8.1 การทำงานของระบบ

รูปที่ 8.1 การทำงานของระบบ

ในการควบคุมอุปกรณ์ Hard ware เมื่อเรา Click mouse บนหน้าจอคอมพิวเตอร์ คอมพิวเตอร์จะส่งข้อมูลที่เป็นรหัสที่ตั้งไว้ออกทางพอร์ตอนุกรม ไปยังคอลโทรลเลอร์ MCS-51 เพื่อให้คอนโทรลเลอร์ MCS-51 ทำการตรวจสอบรหัสที่ตั้งไว้บน MCS-51 ว่าตรงกับรหัสของตัวเองหรือไม่ ถ้าตรงก็ให้ไปทำงานตามคำสั่งที่ตั้งไว้ เช่น ไปสั่งให้ได้ยินเสียงของนักเรียนคนใดคนหนึ่ง ถ้ารหัสไม่ตรงกับรหัสที่ตั้งไว้ก็กลับไปเก็บค่าจาก Display ใหม่

การทำงานของระบบ มีส่วนประกอบหลักๆคือ เครื่องคอมพิวเตอร์ซึ่งเชื่อมต่อกับส่วนควบคุม ซึ่งเป็นไมโครคอลโทรลเลอร์ทำหน้าที่เสมือนกับเป็นวงจร ซีเลคเตอร์สวิตซ์ (Selector Switch) ซึ่งมีหน้าที่รับสัญญาณเสียงเข้ามา 4 ช่องสัญญาณเสียง โดยช่องสัญญาณที่ 1 จะรับสัญญาณเสียงอาจารย์ผู้สอน และสัญญาณเสียงของผู้เรียน ที่ถูกอาจารย์เลือกให้พูด โดยพูดให้ทุกคนทั้งหมดได้ยิน ส่วนสัญญาณที่ 2และสัญญาณที่ 3 นั้น คือสัญญาณของเทป 1 และเทป 2 และสัญญาณสุดท้ายนั้นเป็นสัญญาณเสียงของ CD-ROM จากเครื่องคอมพิวเตอร์

อีกส่วนหนึ่งของส่วนควบคุม คือวงจรควบคุมเสียง ซึ่งทำหน้าที่รับสัญญาณเสียงที่มาจากซีเลคเตอร์สวิทซ์ แล้วไปขยายด้วยแอมป์พลิฟลายเออร์ และออกลำโพงเพื่อให้ได้ยินเสียงดังทั้งหมดภายในห้อง

ส่วนต่อมาที่จะกล่าวถึง คือ ส่วนรับฟังต่างๆ ซึ่งในส่วนรับฟังนี้จะรับสัญญาณทั้ง 4 ช่องจาก ซีเลคเตอร์สวิตซ์ และจะมีรีเลย์สวิตซ์ เลือกฟังสัญญาณต่างๆตามต้องการ โดยมีไมโคร

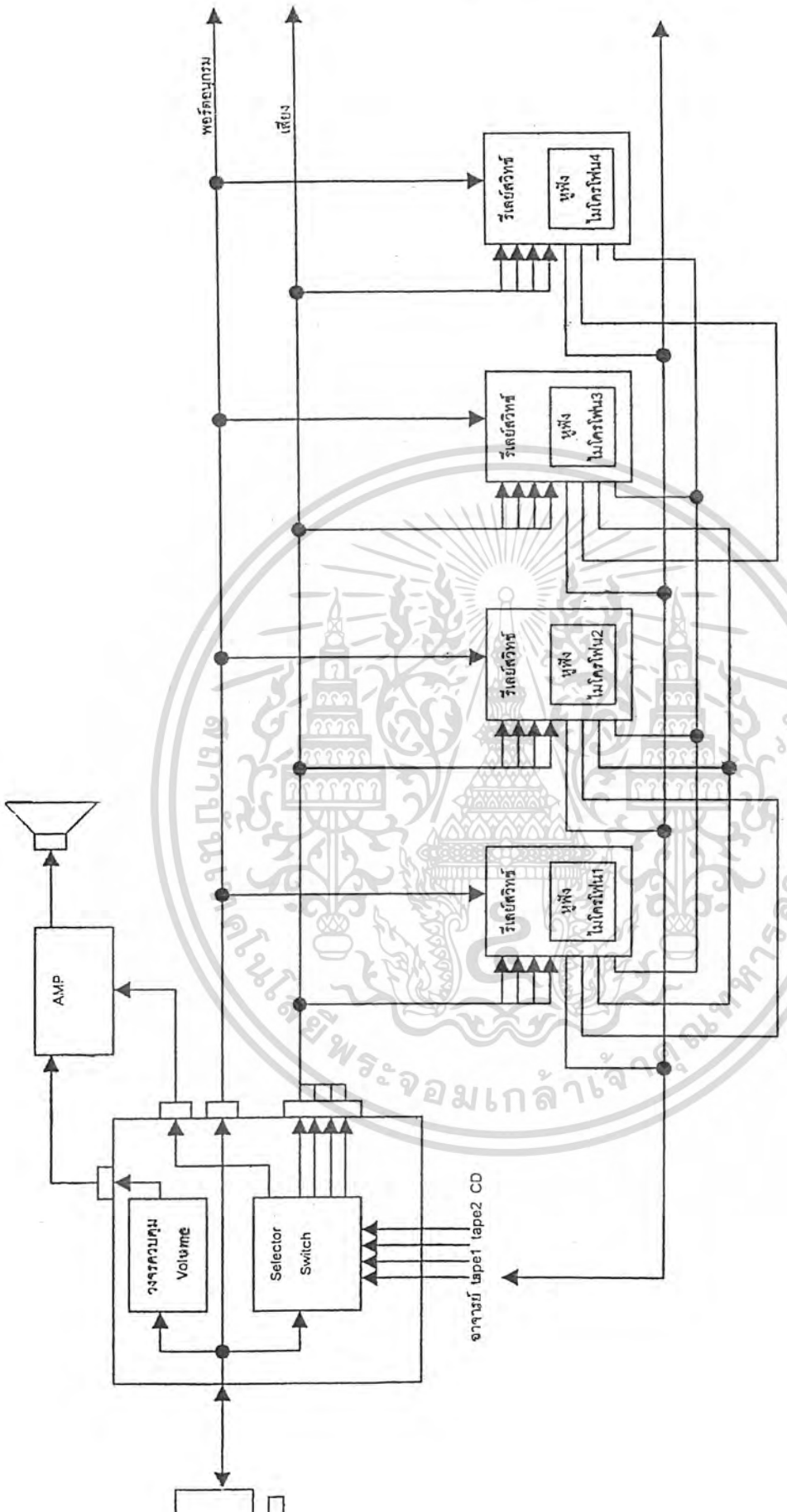
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คอนโทรลเลอร์ MCS-51 เป็นตัวควบคุม สวิตซ์ดังกล่าว โดยไมโครคอนโทรลเลอร์จะรับคำสั่งจากคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรม แล้วสั่งงานรีเลย์สวิตซ์

ในการต่อรีเลย์สวิทซ์ได้มีฟังก์ชันการทำงานเพิ่มเติมคือมีการจับกลุ่มการสนทนาระหว่างผู้เรียนด้วยกัน โดยมีการสนทนาแบบจับคู่ แบบกลุ่ม 3 คนและ แบบกลุ่ม 4 คน โดยมีหลักการทำงานคือ มีการแบ่งกลุ่มของผู้เรียนไว้ และเชื่อมต่อสัญญาณถึงกันเป็นกลุ่มๆ ทั้งแบบจับคู่ แบบกลุ่ม 3 คน และ แบบกลุ่ม 4 คน ซึ่งเมื่อผู้สอนต้องการให้มีรูปแบบการสนทนาแบบไหน ก็เลือกฟังก์ชั่นการทำงานแบบนั้น

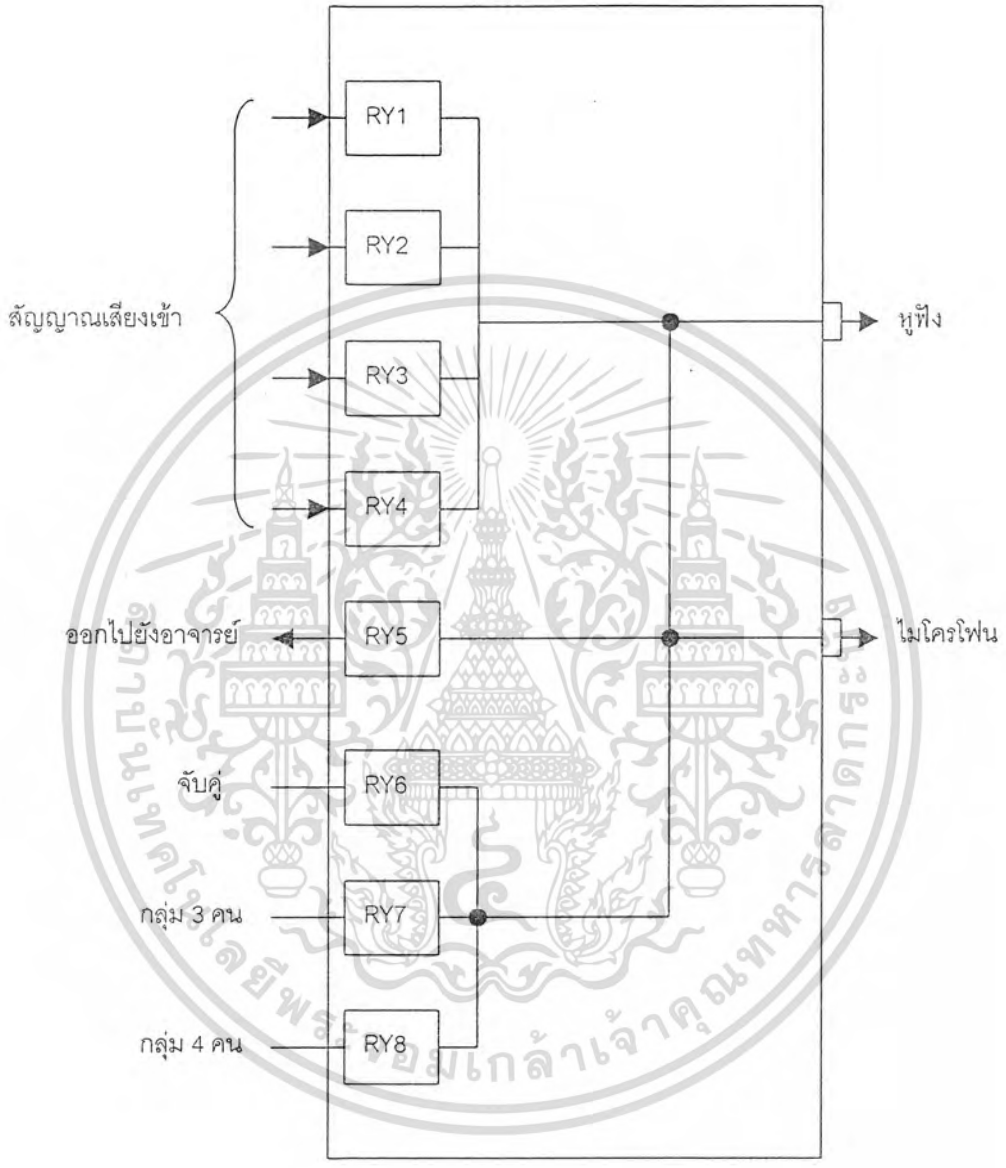
ส่วนสุดท้าย คือส่วนของ EAR PHONE ซึ่งประกอบไปด้วย HEADSET (หูฟัง) และไมโครโฟน(MICROPHONE) ซึ่งใช้สำหรับให้ผู้เรียนพูดและฟัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.2 แสดงถึงส่วนประกอบของ SOUNDLAB MASTER CONTROL AND HEARING UNIT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.3 รีเลย์สวิตซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

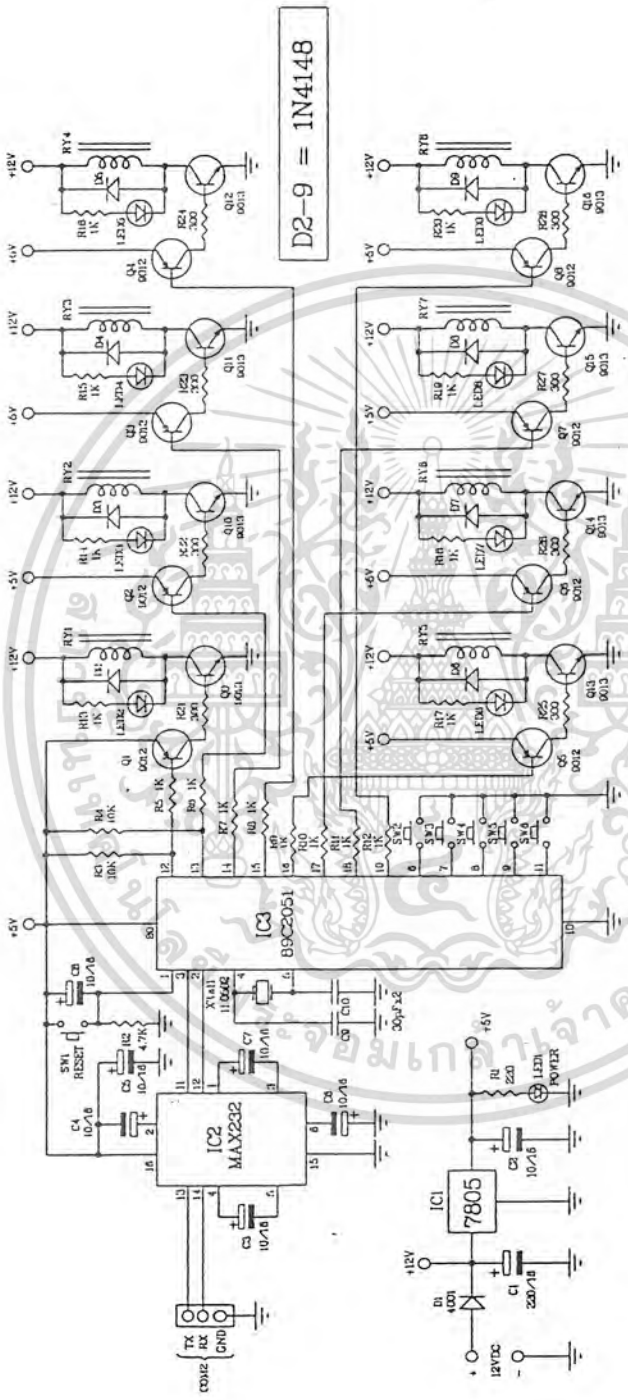
8.2 การทำงานของ HARD WARE

ตัวการใหญ่ที่ทำให้เกิดคอลโทรลเลอร์ คือ ตัวไมโครคอนโทรลเลอร์เอง ในที่นี้จะใช้ IC ตระกูล MCS-51 โดยใช้เบอร์ 89C2051 เป็นไมโครคอนโทรลเลอร์ ซึ่ง IC เบอร์ 89C2051 นั้นมีขนาดเล็กเพียง 20 ขา ประสิทธิภาพสูง กินไฟต่ำ คือใช้ได้ตั้งแต่ 2.7-6.0 โวลต์ มีหน่วยความจำแบบแฟลชบรรจุภายในตัว 2 กิโลไบต์ ขนาดความจำแรม 128 ไบต์ สามารถขับ LED ได้โดยตรง และยังมีวงจรเปรียบเทียบทางอนาล็อก และฟังก์ชั่นอื่นๆอีก ส่วนการติดต่อกับอุปกรณ์ต่างๆนั้น จะมี IC MAX 232 เป็นตัวควบคุมการไหลของข้อมูล ซึ่งแยกเป็น 2 ประเภทขา คือขาที่ทำหน้าที่เป็นเอาต์พุต และขาที่ทำหน้าที่เป็นอินพุต

ในการที่ไมโครคอนโทรลเลอร์จะติดต่อกับคอมพิวเตอร์ได้ ต้องใช้อุปกรณ์ช่วยคือ IC MAX 232 ดังที่กล่าวไปแล้วนั้น จะใช้บัฟเฟอร์รับส่งข้อมูลผ่านจากคอมพิวเตอร์สู่ MCS-51 และจาก MCS-51 สู่คอมพิวเตอร์ ซึ่งเมื่อมีการติดต่อระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ก็จะไม่เกิดปัญหาอะไร แต่ถ้าเป็นการติดต่อระหว่าง MCS-51 หลายๆตัว กับคอมพิวเตอร์ตัวเดียว ในเวลาเดียวกัน ขาส่งจาก IC MAX 232 จะเกิดปัญหา จะต้องมีวงจรตัดการทำงานให้เข้าจังหวะซึ่งกันและกัน ของ MCS-51 ทั้งหลาย ถ้าไม่ทำเช่นนี้ข้อมูลจะผิดพลาดได้

เมื่อเริ่มการจ่ายไฟเข้าเครื่อง MCS-51 วงจรรีเซตอัตโนมัติจะทำงานโดยเคลียร์ค่าต่างๆเพื่อเริ่มต้น วงจรจะทำงานเมื่อวงจรหยุดการรีเซต MCS-51 ก็เริ่มทำงาน MCS-51 จะทำการเรียกข้อมูลจากภายในตัว ตั้งแต่แอดเดรส 0000H มาทำการวิเคราะห์ และทำงานตามโปรแกรมที่มีอยู่ในตัวเองต่อไป วงจรคอนโทรลเลอร์แสดงดังรูปที่ 8.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.4 วงจรคอลโทรลเกอร์

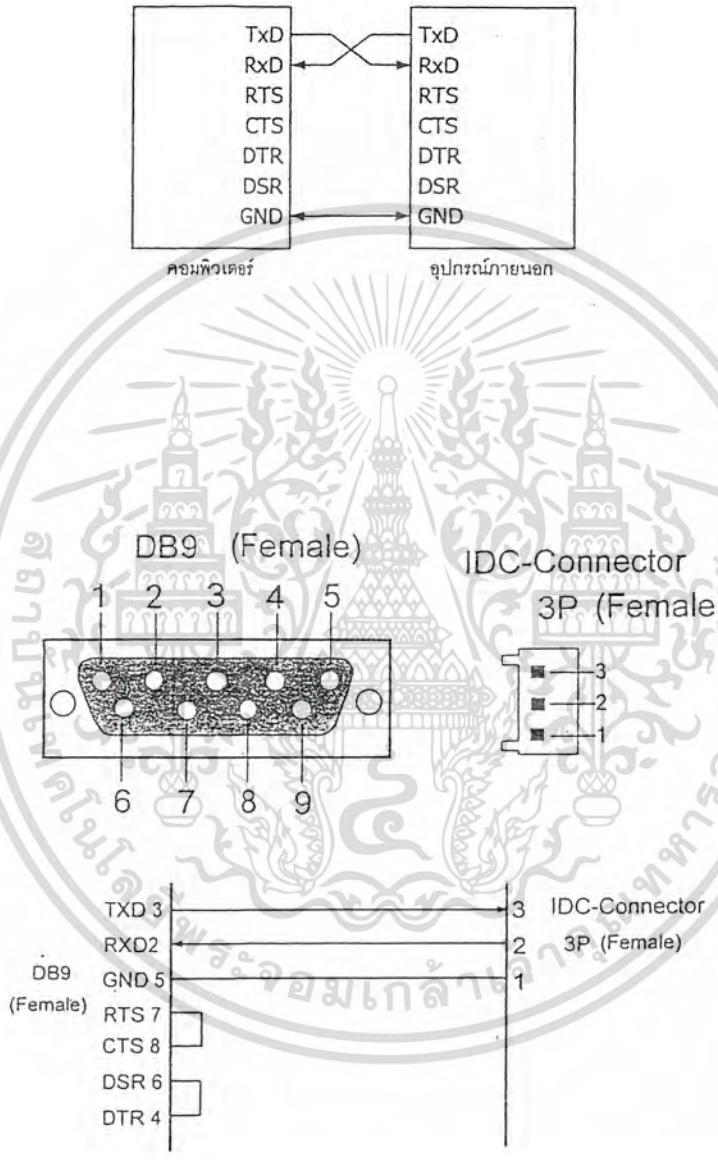
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 8.4 จะแสดงวงจรในการควบคุมอุปกรณ์ต่างๆ โดยเริ่มมาจากพอร์ตCom ของเครื่องคอมพิวเตอร์มาต่อเข้ากับวงจรควบคุมชาวด์แลปทางด้าน พอร์ตCom เมื่อโปรแกรมควบคุมได้รับคำสั่งจากผู้ใช้โปรแกรม จะส่งข้อมูลออกมาจากคอมพิวเตอร์ไปเข้าวงจรในการควบคุม โดยผ่าน IC MAX 232 ก่อน เพื่อทำการเปลี่ยนระดับแรงดันที่ได้มาจากคอมพิวเตอร์ ให้เป็นระดับแรงดันแบบทีทีแอล กล่าวคือระดับแรงดันที่ได้มาจากคอมพิวเตอร์ จะมีระดับแรงดันอยู่ที่ ±3 โวลต์ ถึง ±12 โวลต์ ซึ่งไม่เหมาะที่จะนำมาต่อใช้งานโดยตรงกับ MCS-51 ดังนั้นจึงต้องใช้ MAX 232 เพื่อเปลี่ยนระดับแรงดันเสียก่อน เมื่อผ่าน MAX 232 มาแล้วจึงมาเข้า MCS-51 ซึ่งเป็นไอซีไมโครคอนโทรลเลอร์จะทำการประมวลผลตามโปรแกรมที่ได้โปรแกรมไว้ในตัวไอซี

ไอซีคอนโทรลลอร์ MCS-51 เมื่อทำการประมวลผลตรงกับข้อมูลที่เครื่องคอมพิวเตอร์ส่งมา ก็จะทำการเปลี่ยนระดับของลอจิกที่เดิมเป็น HIGH อยู่ให้กลายเป็น LOW เพื่อให้ Q1-Q8 ตัวใดตัวหนึ่งทำงานส่งผลให้ Q9-Q16 ตัวใดตัวหนึ่ง ทำงานตามการได้รับไบอัสรีเลย์ จึงทำงานด้วย

ภาคจ่ายไฟจะได้มาจากอะเดปเตอร์ 12 โวลต์จากภายนอกมาจ่ายเข้าวงจรโดยผ่าน D1 ซึ่งทำหน้าที่ป้องกัน การต่อแรงดันผิดขั้วแล้วไปเข้า รีเลย์1 ถึง รีเลย์ 8 และอีกส่วนหนึ่งจะไปเข้า ไอซี 7805 ซึ่งทำหน้าที่ลดระดับแรงดันจาก 12 โวลต์ให้เหลือ 5 โวลต์ เพื่อจ่ายให้วงจรต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.5 แสดงการต่อสายของพอร์ตระหว่างคอมพิวเตอร์และวงจรควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขา 2 RXD (Receive Data) ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจีสเตอร์บัฟเฟอร์

ขา 3 TXD (Transmitted Data) ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

ขา 4 DTR (Data Terminal Ready) เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อขากับ DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางต้องเชื่อมต่อกับ DSR ของคอมพิวเตอร์ และเนื่องจากวงจรนี้เชื่อมต่อเป็นแบบ Null Modem คือใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อ DTR และ DSR ของตัวมันเองเข้าด้วยกัน

ขา 5 GND (Signal Ground) ขานี้เป็นขากราวด์ของระบบ

ขา 6 DSR (Data Set Ready) ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

ขา 7 RTS (Request To Send) เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ซึ่งในกรณีนี้ใช้การเชื่อมต่อแบบ Null modem 3สาย จึงจะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

ขา 8 CTS (Clear To Send) ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TXD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3 การทำงานของ SOLF WARE

ในส่วนส่วนของโปรแกรมการควบคุมนั้น จะมีการต่อสายสัญญาณส่งผ่านข้อมูลระหว่างคอลโทรลเลอร์บอร์ดกับคอมพิวเตอร์ส่วนบุคคลทางพอร์ตอนุกรม RS-232 มีการใช้งาน 3 เส้น คือ RXD , TXD และ GND ซึ่งในส่วนของคอลโทรลเลอร์บอร์ด จะมีการเขียนโปรแกรมภาษาแอสแซมบลี้ โดยโปรแกรมภาษาแอสแซมบลี้ที่เขียนไว้นั้น จะเก็บไว้ในไอซี MCS-51 ซึ่งจะตอบสนองต่อสัญญาณที่เข้ามาทางพอร์ตอนุกรม โดยคอมพิวเตอร์ได้ส่งสัญญาณมา เพื่อให้บอร์ดได้รับรู้และทำงานได้ตามต้องการซึ่งจะทำให้ควบคุมส่วนรับฟังต่างๆได้

ในด้านของส่วนรับฟังก็สามารถที่จะติดต่อกลับไปยังคอมพิวเตอร์ได้ โดยการกดCallเรียกเพื่อให้มีสัญญาณปรากฏที่หน้าจอคอมพิวเตอร์ตามตำแหน่งของส่วนรับฟังนั้น เพื่อให้ผู้สอนได้รับรู้ว่ามีส่วนรับฟังเครื่องที่เท่าไรต้องการที่จะติดต่อกับผู้สอน โดยในส่วนรับฟังแต่ละเครื่องจะมีรหัสแอดเดรสต่างกัน ซึ่งจะถูกกำหนดโดยการเขียนโปรแกรมภาษาแอสแซมบลี้ลงในไอซี MCS-51 แต่ละตัว โดยการที่ส่วนรับฟังแต่ละเครื่องมีรหัสแอดเดรสต่างกันทำให้สามารถกำหนดได้ว่า จะให้ส่วนรับฟังเครื่องไหนทำอะไรและอีกเครื่องทำอะไร โดยคอลโทรลเลอร์แต่ละตัวจะเชคข้อมูลที่ส่งมาว่าตรงกับรหัสแอดเดรสหรือไม่ ถ้าตรงกับรหัสแอดเดรสก็จะทำตามคำสั่ง ถ้าไม่ตรงกับรหัสแอดเดรสก็จะไม่ทำตามคำสั่ง

ในส่วนของคอมพิวเตอร์จะใช้โปรแกรมวิชวลเบสิคเขียนโปรแกรม โดยการสร้างหน้าต่างเพื่อใช้สำหรับสั่งการให้ส่งข้อมูลออกไปทางพอร์ตอนุกรมด้วยการคลิกปุ่มที่อยู่ในหน้าต่างเพื่อเลือกว่าจะให้ทำอะไร

โดยจากหน้าจอจะประกอบด้วยส่วนต่างๆดังนี้

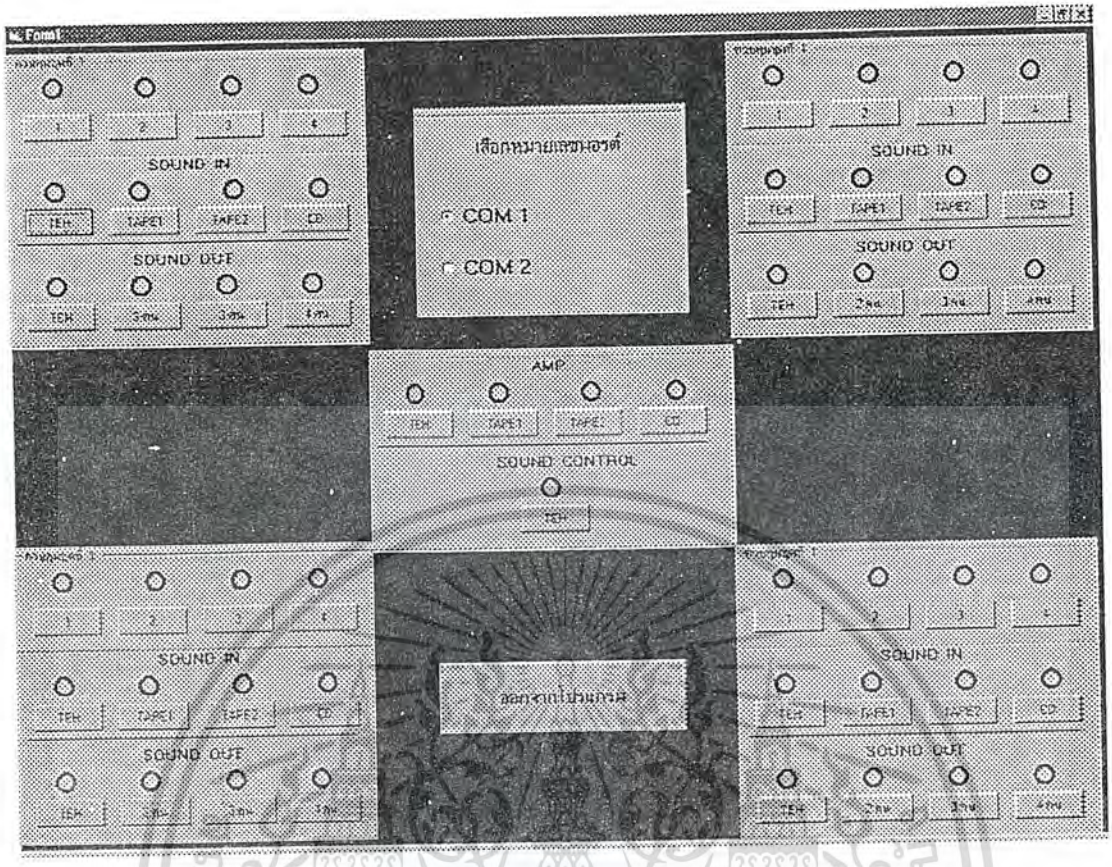
ชุดควบคุมกลุ่ม ภายในหน้าต่างนี้จะประกอบไปด้วยตำแหน่งของส่วนรับฟังแต่ละส่วน ซึ่งจะแทนแต่ละตำแหน่งด้วยตัวเลข โดยในแต่ละตำแหน่งจะแสดงสัญญาณไฟกระพริบตามตำแหน่งที่ถูกกดสัญญาณเรียกเข้ามา ผู้สอนก็จะสามารถรู้ได้ว่าตำแหน่งไหนกดปุ่มCall ซึ่งในโปรแกรมนี้กำหนดให้มี 4 คนต่อกลุ่ม ซึ่งเมื่อใช้งานจริงสามารถเขียนเพิ่มให้มากขึ้นได้เช่น 16 คนต่อกลุ่ม ในส่วนควบคุมนี้ผู้สอนสามารถเลือกเสียงที่ต้องการให้กลุ่มใดๆได้ยินได้ตามคำสั่งของผู้สอน ไม่ว่าจะเป็นเสียงของผู้สอนเอง , TAPE1 , TAPE2 หรือ CD และยังมีคำสั่งพิเศษให้กลุ่มนั้นๆสนทนากัน 2 คน , 3 คน , 4 คน หรือให้เสียงของคนใดคนหนึ่งกลับมายังผู้สอนเพื่อให้ทุกคนได้ยินเสียงของคนนั้นได้

AMP ภายในหน้าต่างนี้จะมีปุ่มกำหนดเลือกเสียงที่ต้องการ เพื่อออกลำโพง

พอร์ต หน้าต่างนี้ใช้เลือกว่าต้องการพอร์ต COM1 หรือ COM2

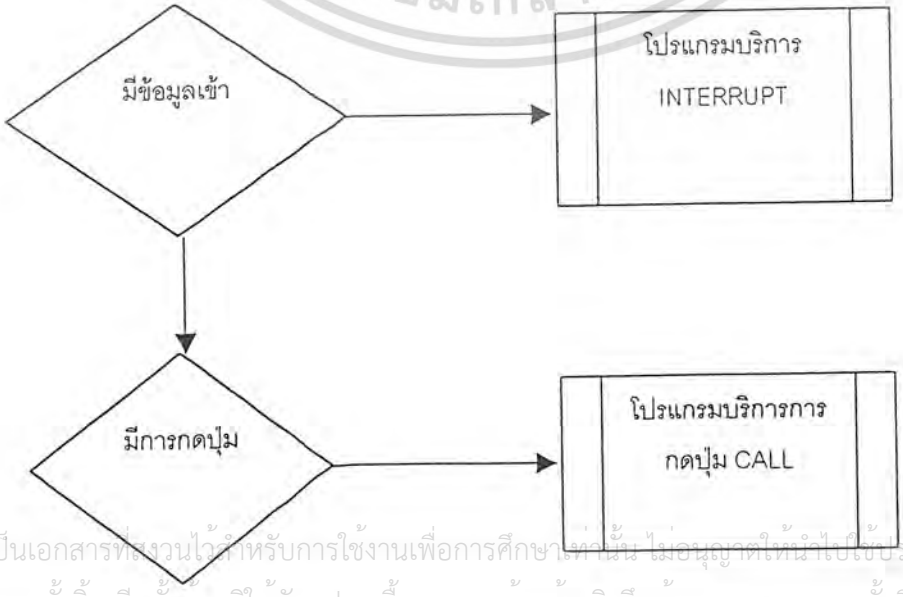
ออกจากโปรแกรม เมื่อกดปุ่มนี้ก็จะเป็นการปิดโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

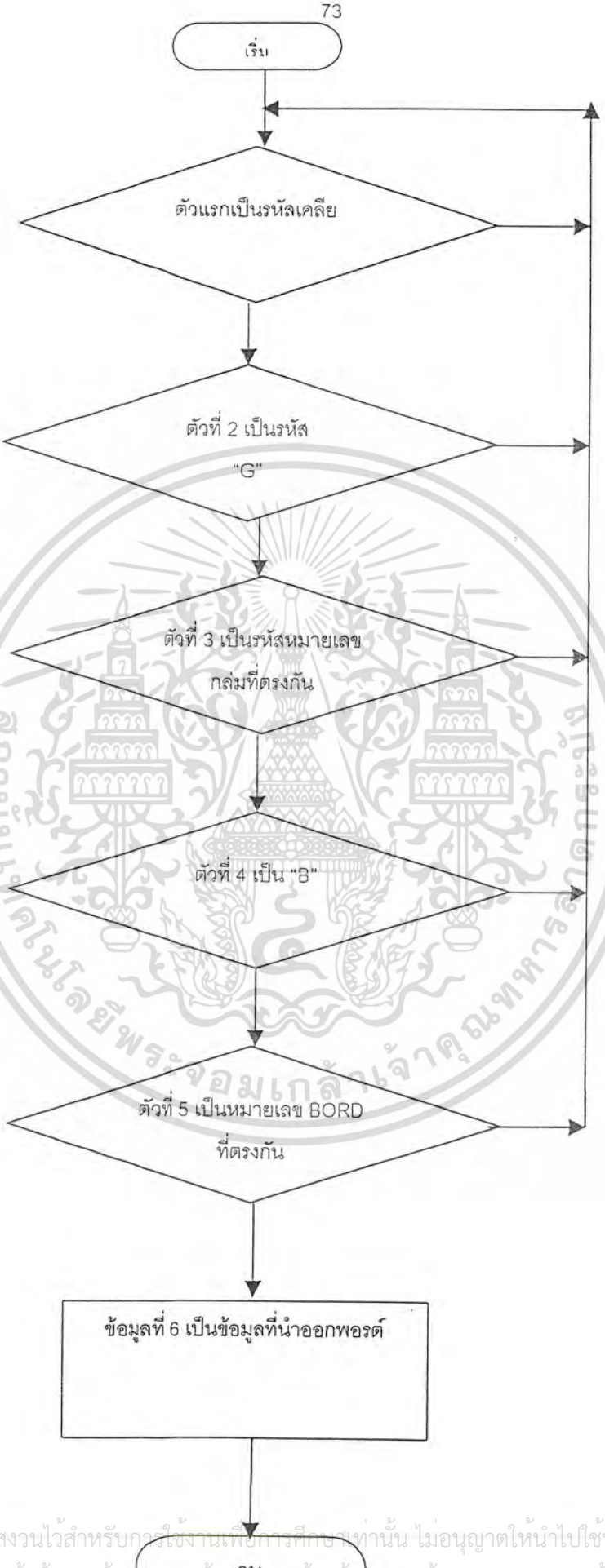


รูปที่ 8.6 แสดงหน้าต่างของโปรแกรมควบคุมซาวด์แลบ

8.4 แสดงโฟรชาร์ทการทำงาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9
บทสรุป

ในโครงงานเครื่องควบคุมซาวด์แลบและส่วนรับฟังนี้ (Soundlab Master Control And Hearing Unit) เป็นการศึกษาวิธีการควบคุมสัญญาณเสียงที่ต้องการให้ได้ยินด้วยคอมพิวเตอร์วิธีหนึ่ง โดยการควบคุมนี้เป็นการควบคุมโดยใช้โปรแกรมวิชวลเบสิค (Visual Basic) ซึ่งเขียนไว้ในเครื่องคอมพิวเตอร์ โดยการสร้างอุปกรณ์ที่มีคอมโปแนนต์ที่มีอยู่ในวิชวลเบสิคมาใช้ และเขียนโปรแกรมเพื่อให้โปรแกรมวิชวลเบสิค ติดต่อรับส่งข้อมูลต่างๆ เพื่อควบคุมอุปกรณ์ต่างๆ และแสดงผลที่หน้าจอคอมพิวเตอร์ ข้อมูลต่างๆจะผ่านพอร์ตอนุกรมของคอมพิวเตอร์แล้วติดต่อกับไมโครคอลโทลเลอร์ MCS-51 ซึ่งถูกโปรแกรมภาษาแอสแซมบลี้ เพื่อให้รับ-ส่งข้อมูลกับคอมพิวเตอร์ และติดต่อกับไอซี MAX232 ซึ่งเป็นอินพุตพอร์ตและเอาต์พุตพอร์ต เพื่อรับข้อมูลมาแล้วส่งออกที่พอร์ตไปควบคุมการทำงานของอุปกรณ์ต่างๆ และส่งค่าหรือข้อมูลแล้วแสดงผลที่คอมพิวเตอร์ โดยการควบคุมของโครงงานนี้เป็นการควบคุม เสียงผู้สอน,เสียงนักเรียน,เสียงเทปและเสียงCD ให้ออกมาตามที่ผู้สอนควบคุมหรือมีการรับสัญญาณจากนักเรียน เมื่อนักเรียนกดปุ่มCallเรียกจะเป็นการส่งสัญญาณจากอุปกรณ์ไปยังคอมพิวเตอร์ ในโครงงานยังนี้ยังสามารถเพิ่มอุปกรณ์หรือเพิ่มฟังก์ชั่นต่างๆให้มากขึ้นอีก โดยการเขียนโปรแกรมเพิ่มเติมจากโปรแกรมที่เขียนไปแล้ว และยังสามารถจัดรูปแบบให้เหมะสมได้ตามต้องการ ในกระบวนการรับส่งข้อมูล สัญญาณที่ใช้จะประกอบด้วยส่วนต่างๆ คือ สัญญาณข้อมูลแจ้งการรับหรือการส่ง , สัญญาณข้อมูลเพื่อแจ้งการเลือกอุปกรณ์ที่คอมพิวเตอร์ต้องการติดต่อด้วย , สัญญาณข้อมูลที่ต้องการสื่อสาร โดยทั้งสามสัญญาณจะถูกควบคุมด้วยคอมพิวเตอร์

วิธีการควบคุมในโครงงานนี้เป็นเพียงวิธีการหนึ่งเท่านั้น การควบคุมนี้สามารถประยุกต์เพื่อใช้ในการควบคุมอุปกรณ์ต่างๆ ชนิดอื่นได้ โดยทำให้สัญญาณที่ต้องการตดต่อสื่อสารเป็นสัญญาณดิจิตอลก็สามารถควบคุมการทำงานและแสดงผลที่คอมพิวเตอร์ได้

การแสดงผลในโครงงานนี้ยังนับว่าแสดงผลได้ช้า เช่นในกรณีที่มีการส่งข้อมูลกลับไปยังคอมพิวเตอร์เมื่อมีการกดปุ่มCall จะเห็นไฟกระพริบช้า เนื่องจากเป็นพอร์ตอนุกรม หากผู้ใช้ระบบต้องการให้มีความรวดเร็วขึ้น อาจต้องใช้การติดต่อระหว่างคอมพิวเตอร์กับไมโครคอลโทรเลอร์ทางพอร์ตขนาน และควรเขียนโปรแกรมวิชวลเบสิคเป็นโปรแกรมเดียวกันสำหรับการควบคุมทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปัญหาและอุปสรรค์

- ใช้สายสัญญาณค่อนข้างเยอะ ดูรุงรังไม่ค่อยสะดวกต่อการใช้งาน
- สัญญาณเสียงบางครั้งจะดังไม่พอ หรือไม่ชัดเจน
- ปุ่มCall บางครั้งตอบสนองช้า เนื่องจากไมโครคอลโทรลเลอร์ยังค้างสถานะจากคำสั่งก่อน
- รูปแบบและการใช้งานของระบบเครื่องควบคุมซาวด์แลบและส่วนรับฟังนี้ อาจจะดูแล้วไม่ค่อยมาตรฐานเนื่องจากข้อมูลเกี่ยวกับเรื่องของซาวด์แลบในท้องตลาดหาได้ยาก ส่วนใหญ่จะเป็นบริษัทที่ไม่ค่อยมีคนรู้จักและมีจำนวนน้อยราย
- รูปแบบของเครื่องควบคุมซาวด์แลปและส่วนรับฟังนี้ อาจจะดูแล้วไม่ค่อยทันสมัยเนื่องจากไม่ค่อยได้นำรูปแบบของเครื่องควบคุมซาวด์แลบและส่วนรับฟัง ที่เคยมีการใช้มาในอดีตมาเป็นต้นแบบ เนื่องจากมีข้อมูลไม่มากจึงเป็นการคิดคอนเซปท์เอง ทำให้ได้งานเบื้องต้นที่ต้องพัฒนาต่อไป

แนวทางแก้ปัญหา

- ใช้สายสัญญาณแบบมีหลายเส้นเป็นชุดรวมในเส้นเดียว ทำให้งานสวยงามไม่รุงรัง
- ใช้วงจรขยายเสียงที่มีคุณภาพ และเพิ่มวงจรขจัดเสียงฮัมของไมค์
- ผู้สนใจนำโครงงานนี้ไปศึกษาและพัฒนาต่อให้มีความทันสมัยขึ้น ใช้งานได้ง่ายขึ้นและแก้ไขข้อบกพร่องต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.

โปรแกรมภาษาแอสแซมบลี้ บน MCS-51

```
                ORG         20H
BUF_COUNT:      DS          1
KEY_CALL:       DS          1
STACK:          DS          35
BR              EQU         256-50

                ORG         0000H
                SJMP        START
                ORG         0023H
                LJMP        RESIVE_DATA

START:          MOV         SP,#STACK
                MOV         BUF_COUNT,#0H
                MOV         P1,#0FFH
                MOV         PCON,#80H
                MOV         TMOD,#20H
                MOV         TH1,#BR
                MOV         TL1,#BR
                SETB        TCON.6
                MOV         SCON,#50H
                CLR         SCON.1
                CLR         SCON.0
                SETB        SCON.4
                SETB        EA
                SETB        ES
                MOV         P0,#0FFH
                MOV         P2,#0FFH

;#################################################################
;###################  start main program ########################
;#################################################################
MAIN:           LCALL       READ_KEY_CALL
                SETB        P3.7
                SJMP        MAIN

;*****************************************************************
;**********  function read key CALL tobe send num bord to rs232**********
;*****************************************************************
READ_KEY_CALL:  SETB        P3.2
                NOP
                JB          P3.2,END_READ_KEY
                SETB        P3.2
                NOP
                JNB         P3.2,$
                MOV         R2,#2
READ_KEY_CALL1: MOV         A,#'C'
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            LCALL     SEND_ONE_BYTE
            MOV       A,#'A'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'L'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'L'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'B'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'Y'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'G'
            LCALL     SEND_ONE_BYTE
            LCALL     READ_DIP_SW
            ANL       A,#00000011B
            ORL       A,#00110000B
            LCALL     SEND_ONE_BYTE
            MOV       A,#'B'
            LCALL     SEND_ONE_BYTE
            LCALL     READ_DIP_SW
            ANL       A,#00001100B
            RR        A
            RR        A
            ORL       A,#00110000B
            LCALL     SEND_ONE_BYTE
            MOV       A,#'E'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'N'
            LCALL     SEND_ONE_BYTE
            MOV       A,#'D'
            LCALL     SEND_ONE_BYTE
            DJNZ      R2,READ_KEY_CALL1
END_READ_KEY:     RET
;*************************************************************
;****************** function send data by ACC to rs232  *********
;*************************************************************

SEND_ONE_BYTE:    MOV       SBUF,A
            JNB       SCON.1,$
            CLR       SCON.1
            RET

;*************************************************************
;****************** function read dip sw ******************
;*************************************************************
READ_DIP_SW:      SETB      P3.6
            SETB      P3.5
            SETB      P3.4
            SETB      P3.3
            NOP
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            NOP
            MOV     A,P3
            CPL     A
            ANL     A,#10111000B
            JNB     ACC.7,READ_DIP_SW1
            RL      A
            SETB    ACC.7
            SJMP    READ_DIP_SW2
READ_DIP_SW1:   RL      A
            CLR     ACC.7
READ_DIP_SW2:   ANL     A,#11110000B
            SWAP    A
            RET
;*************************************************
;*********function resive data by rs232**************
;*************************************************
RESIVE_DATA:    PUSH    ACC
            PUSH    PSW
            JB      SCON.1,END_RESIVE
            MOV     A,SBUF
            MOV     R7,A
            CJNE    A,#'G',RESIVE_DATA1
            MOV     BUF_COUNT,#01H
            SJMP    END_RESIVE
RESIVE_DATA1:   MOV     A,BUF_COUNT
            CJNE    A,#01H,RESIVE_DATA2
            LCALL   READ_DIP_SW
            ANL     A,#00000011B
            MOV     R0,A
            MOV     A,R7
            ANL     A,#00001111B
            MOV     PSW,#0H
            SUBB    A,R0
            JNZ     END_RESIVE  ;check Group
            MOV     BUF_COUNT,#02H
            SJMP    END_RESIVE
RESIVE_DATA2:   MOV     A,BUF_COUNT
            CJNE    A,#02H,RESIVE_DATA3
            MOV     A,R7
            CJNE    A,#'B',END_RESIVE
            MOV     BUF_COUNT,#03H
            SJMP    END_RESIVE
RESIVE_DATA3:   MOV     A,BUF_COUNT
            CJNE    A,#03H,RESIVE_DATA4
            LCALL   READ_DIP_SW
            ANL     A,#00001100B
            RR      A
            RR      A
            MOV     R0,A
            MOV     A,R7
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            ANL     A,#00001111B
            MOV     PSW,#0H
            SUBB    A,R0
            JNZ     END_RESIVE  ;Check bord in group
            MOV     BUF_COUNT,#04H
            SJMP    END_RESIVE
RESIVE_DATA4:   MOV     A,BUF_COUNT
            CJNE    A,#04H,END_RESIVE
            MOV     A,R7
            CPL     A
            MOV     P1,A
            MOV     BUF_COUNT,#0H
END_RESIVE:     CLR     SCON.0
            POP     PSW
            POP     ACC
            RETI


            END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

โปรแกรมวิชวล เบสิค

```
Type Status
 STATUS_LED_B1 As Boolean
 STATUS_LED_B2 As Boolean
 STATUS_LED_B3 As Boolean
 STATUS_LED_B4 As Boolean
 STATUS_LED_THE_SI_B1 As Boolean
 STATUS_LED_TAPE1_SI_B1 As Boolean
 STATUS_LED_TAPE2_SI_B1 As Boolean
 STATUS_LED_CD_SI_B1 As Boolean
 STATUS_LED_THE_SO_B1 As Boolean
 STATUS_LED_2P_SO_B1 As Boolean
 STATUS_LED_3P_SO_B1 As Boolean
 STATUS_LED_4P_SO_B1 As Boolean
End Type

Function Create_data(DataMe As Status) As Integer
Dim Temp As Integer
Temp = 0
'แสดงวิธีการ AND
'1110011
'      AND
'1001001
'       =
'1000001

'แสดงวิธีการ OR
'1110011
'       OR
'1001001
'=
'1111011
If DataMe.STATUS_LED_4P_SO_B1 = False Then  'ถ้าปุ่ม "4คน"  ถูกกดให้เซ็ตบิทที่ 0 ให้
เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And (&H1&))
Else                                        'ถ้าปุ่ม "4คน"  ไม่ถูกกดให้เซ็ตบิทที่ 0 ให้เป็น 0 แล้ว
นำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HFE&))
End If
If DataMe.STATUS_LED_3P_SO_B1 = False Then  'ถ้าปุ่ม "3คน"  ถูกกดให้เซ็ตบิทที่ 1 ให้
เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H2&)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Else                                    'ถ้าปุ่ม "3คน" ไม่ถูกกดให้เซ็ตบิทที่ 1 ให้เป็น 0 แล้วนำ
ไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HFB&))
End If
If DataMe.STATUS_LED_2P_SO_B1 = False Then  'ถ้าปุ่ม "2คน" ถูกกดให้เซ็ตบิทที่ 2 ให้
เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H4&)
Else                                    'ถ้าปุ่ม "2คน" ไม่ถูกกดให้เซ็ตบิทที่ 2 ให้เป็น 0 แล้ว
นำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HFD&))
End If
If DataMe.STATUS_LED_THE_SO_B1 = False Then  'ถ้าปุ่ม "TEH" ของชุด SOUND
OUT ถูกกดให้เซ็ตบิทที่ 3 ให้เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H8&)
Else                                    'ถ้าปุ่ม "TEH" ของชุด SOUND OUT ไม่ถูก
กดให้เซ็ตบิทที่ 3 ให้เป็น 0แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HF7&))
End If
If DataMe.STATUS_LED_CD_SI_B1 = False Then  'ถ้าปุ่ม "CD" ของชุด SOUND IN
ถูกกดให้เซ็ตบิทที่ 4 ให้เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H10&)
Else                                    'ถ้าปุ่ม "CD" ของชุด SOUND IN ไม่ถูกกดให้
เซ็ตบิทที่ 4 ให้เป็น 0แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HEF&))
End If
If DataMe.STATUS_LED_TAPE2_SI_B1 = False Then  'ถ้าปุ่ม "TAPE2" ของชุด
SOUND IN ถูกกดให้เซ็ตบิทที่ 5 ให้เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H20&)
Else                                    'ถ้าปุ่ม "TAPE2" ของชุด SOUND IN ไม่ถูก
กดให้เซ็ตบิทที่ 5 ให้เป็น 0แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HDF&))
End If
If DataMe.STATUS_LED_TAPE1_SI_B1 = False Then  'ถ้าปุ่ม "TAPE1" ของชุด
SOUND IN ถูกกดให้เซ็ตบิทที่ 6 ให้เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H40&)
Else                                    'ถ้าปุ่ม "TAPE1" ของชุด SOUND IN ไม่ถูกกด
ให้เซ็ตบิทที่ 6 ให้เป็น 0แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&HBF&))
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If DataMe.STATUS_LED_THE_SI_B1 = False Then   'ถ้าปุ่ม "TEH" ของชุด SOUND
IN ถูกกดให้เซ็ตบิทที่ 7 ให้เป็น 1แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (255 And &H80&)
Else                                          'ถ้าปุ่ม "TEH" ของชุด SOUND IN ไม่ถูกกด
ให้เซ็ตบิทที่ 7 ให้เป็น 0แล้วนำไปรวมกับค่าเก่า
   Temp = Temp Or (Temp And (&H7F&))
End If
Create_data = Temp
End Function
Function Send_data(Data_Status As Status, MyGroup As Integer)
Dim BufTemp As Integer
BufTemp = 0
If Data_Status.STATUS_LED_B1 = False Then   'ถ้าข้อมูลที่จะส่งไปยัง MCS51 เป็นเป็นของ
บอรท์ ที่ 1
If Form1.OptionCOM1.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM1
   Form1.MSComm1.Output = "G"
   DELAY
   Form1.MSComm1.Output = Chr(MyGroup)
   DELAY
   Form1.MSComm1.Output = "B"
   DELAY
   Form1.MSComm1.Output = "0"
   DELAY
   BufTemp = Create_data(Data_Status)
   Form1.MSComm1.Output = Chr(Int(BufTemp))
End If                                  'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM2
If Form1.OptionCOM2.Value = True Then
   Form1.MSComm2.Output = "G"
   DELAY
   Form1.MSComm2.Output = Chr(Int(MyGroup))
   DELAY
   Form1.MSComm2.Output = "B"
   DELAY
   Form1.MSComm2.Output = "0"
   DELAY
   BufTemp = Create_data(Data_Status)
   Form1.MSComm2.Output = Chr(Int(BufTemp))
End If

End If

If Data_Status.STATUS_LED_B2 = False Then   'ถ้าข้อมูลที่จะส่งไปยัง MCS51 เป็นเป็นของ
บอรท์ ที่ 2
If Form1.OptionCOM1.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM1
```

เอกสารนี้มีลิขสิทธิ์ ห้ามนำไปใช้ในเชิงพาณิชย์ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Form1.MSComm1.Output = "G"
    DELAY
    Form1.MSComm1.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm1.Output = "B"
    DELAY
    Form1.MSComm1.Output = "1"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm1.Output = Chr(Int(BufTemp))
  End If
  If Form1.OptionCOM2.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM2
    Form1.MSComm2.Output = "G"
    DELAY
    Form1.MSComm2.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm2.Output = "B"
    DELAY
    Form1.MSComm2.Output = "1"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm2.Output = Chr(Int(BufTemp))
  End If
  End If

  If Data_Status.STATUS_LED_B3 = False Then   'ถ้าข้อมูลที่จะส่งไปยัง MCS51 เป็นเป็นของ
  บอรท์ ที่ 3
  If Form1.OptionCOM1.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM1
    Form1.MSComm1.Output = "G"
    DELAY
    Form1.MSComm1.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm1.Output = "B"
    DELAY
    Form1.MSComm1.Output = "2"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm1.Output = Chr(Int(BufTemp))
  End If
  If Form1.OptionCOM2.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM2
    Form1.MSComm2.Output = "G"
    DELAY
    Form1.MSComm2.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm2.Output = "B"
    DELAY
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Form1.MSComm2.Output = "2"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm2.Output = Chr(Int(BufTemp))
  End If
  End If

  If Data_Status.STATUS_LED_B4 = False Then    'ถ้าข้อมูลที่จะส่งไปยัง MCS51 เป็นเป็นของ
  บอรท์ ที่ 4
  If Form1.OptionCOM1.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM1
    Form1.MSComm1.Output = "G"
    DELAY
    Form1.MSComm1.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm1.Output = "B"
    DELAY
    Form1.MSComm1.Output = "3"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm1.Output = Chr(Int(BufTemp))
  End If
  If Form1.OptionCOM2.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM2
    Form1.MSComm2.Output = "G"
    DELAY
    Form1.MSComm2.Output = Chr(MyGroup)
    DELAY
    Form1.MSComm2.Output = "B"
    DELAY
    Form1.MSComm2.Output = "3"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm2.Output = Chr(Int(BufTemp))
  End If
  End If
  If MyGroup = 4 Then    'ถ้าข้อมูลที่จะส่งไปยัง MCS51 เป็นเป็นของบอรท์ ที่ 5
  If Form1.OptionCOM1.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM1
    Form1.MSComm1.Output = "G"
    DELAY
    Form1.MSComm1.Output = "4"
    DELAY
    Form1.MSComm1.Output = "B"
    DELAY
    Form1.MSComm1.Output = "4"
    DELAY
    BufTemp = Create_data(Data_Status)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    Form1.MSComm1.Output = Chr(Int(BufTemp))
End If
If Form1.OptionCOM2.Value = True Then 'ถ้าได้มีการเลือกข้อมูลที่จะส่งออกไปยัง COM2
    Form1.MSComm2.Output = "G"
    DELAY
    Form1.MSComm2.Output = "4"
    DELAY
    Form1.MSComm2.Output = "B"
    DELAY
    Form1.MSComm2.Output = "4"
    DELAY
    BufTemp = Create_data(Data_Status)
    Form1.MSComm2.Output = Chr(Int(BufTemp))
End If

End If
End Function
'เป็น Function ที่ทำไว้เพื่อหน่วงเวลาการส่งข้อมูลเพื่อให้ข้อมูลที่ส่งไปให้ MCS 51 พร้อมที่จะรับข้อมูลตัว
ต่อไป
Function DELAY()
Dim I As Integer
Dim j As Integer
Dim k As Integer
k = 0
For I = 1 To 200
 If I > 200 Then
 Exit For
 End If
  For j = 1 To 200
  If j > 200 Then
  Exit For
  End If
   k = k
   Next j
Next I
I = I
End Function
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Dim BufStatus(4) As Status  'ทำการลอกแบบตัวแปรเป็นอีกชื่อหนึ่ง
Dim Temp As Integer

Private Sub CM_2P_SO_Click(Index As Integer)

If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "2คน" ใน SOUND OUT ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_2P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_2P_SO.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_2P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(0).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(0).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(0).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "2คน" ใน SOUND OUT ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_2P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_2P_SO.Item(1).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(1).STATUS_LED_2P_SO_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(1).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(1).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(1).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_THE_SO_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_3P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_4P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "2คน" ใน SOUND OUT ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_2P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_2P_SO.Item(2).BackColor = &HFF&       'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_2P_SO_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SO_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_3P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(2).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "2คน" ใน SOUND OUT ของกลุ่มีที่ 4
If BufStatus(3).STATUS_LED_2P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_2P_SO.Item(3).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_2P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(3).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(3).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

End Sub

Private Sub CM_3P_SO_Click(Index As Integer)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "3คน" ใน SOUND OUT ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_3P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็นสีแดง
Form1.LED_3P_SO.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_3P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(0).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(0).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(0).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อมูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "3คน" ใน SOUND OUT ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_3P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็นสีแดง
Form1.LED_3P_SO.Item(1).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_3P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(1).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(1).STATUS_LED_THE_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_2P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_4P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "3คน" ใน SOUND OUT ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_3P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_3P_SO.Item(2).BackColor = &HFF&         'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_3P_SO_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(2).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(2).BackColor = &H8000000F    'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(2).BackColor = &H8000000F    'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_2P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_4P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "3คน" ใน SOUND OUT ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_3P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_3P_SO.Item(3).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_3P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

End Sub

Private Sub CM_4P_SO_Click(Index As Integer)

If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "4คน" ใน SOUND OUT ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_4P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_4P_SO.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(0).STATUS_LED_4P_SO_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_THE_SO_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_2P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_3P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "4คน" ใน SOUND OUT ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_4P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_4P_SO.Item(1).BackColor = &HFF&       'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_4P_SO_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_THE_SO_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_2P_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(1).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "4คน" ใน SOUND OUT ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_4P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_4P_SO.Item(2).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_4P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม "4คน" ใน SOUND OUT ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_4P_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Form1.LED_4P_SO.Item(3).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_4P_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_2P_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_THE_SO_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

End Sub

Private Sub CM_CD_SI_Click(Index As Integer)

If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม CD ใน SOUND IN ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_CD_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_CD_SI.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_CD_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(0).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(0).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_TAPE1_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_TAPE2_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม CD ใน SOUND IN ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_CD_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_CD_SI.Item(1).BackColor = &HFF&       'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_CD_SI_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_TAPE1_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_TAPE2_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม CD ใน SOUND IN ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_CD_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_CD_SI.Item(2).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_CD_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_TAPE1_SI_B1 = True       'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_TAPE2_SI_B1 = True       'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม CD ใน SOUND IN ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_CD_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_CD_SI.Item(3).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_CD_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้มีลิขสิทธิ์ ห้ามนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(3).STATUS_LED_THE_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_TAPE1_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_TAPE2_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 4 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม CD ใน SOUND IN ของกลุ่มที่ 5
If BufStatus(4).STATUS_LED_CD_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็น
สีแดง
Form1.LED_CD_SI.Item(4).BackColor = &HFF&         'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(4).STATUS_LED_CD_SI_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(4).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(4).STATUS_LED_THE_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_TAPE1_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_TAPE2_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(4), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End Sub

Private Sub CM_TAPE1_SI_Click(Index As Integer)
If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE1  ใน SOUND IN ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_TAPE1_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE1_SI.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_TAPE1_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_TAPE2_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_CD_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE1  ใน SOUND IN ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_TAPE1_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE1_SI.Item(1).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_TAPE1_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Form1.LED_TEH_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_TAPE2_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_CD_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE1  ใน SOUND IN ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_TAPE1_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE1_SI.Item(2).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_TAPE1_SI_B1 = False     'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_TAPE2_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_CD_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Temp = Send_data(BufStatus(2), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE1  ใน SOUND IN ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_TAPE1_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE1_SI.Item(3).BackColor = &HFF&         'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_TAPE1_SI_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_THE_SI_B1 = True             'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_TAPE2_SI_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_CD_SI_B1 = True              'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 4 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE1  ใน SOUND IN ของกลุ่มที่ 5
If BufStatus(4).STATUS_LED_TAPE1_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE1_SI.Item(4).BackColor = &HFF&         'ให้กำหนดให้สีของ LED เป็นสีแดง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(4).STATUS_LED_TAPE1_SI_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(4).STATUS_LED_THE_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_TAPE2_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_CD_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
.SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(4), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

End Sub

Private Sub CM_TAPE2_SI_Click(Index As Integer)

If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE2  ใน SOUND IN ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_TAPE2_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE2_SI.Item(0).BackColor = &HFF&          'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_TAPE2_SI_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(0).BackColor = &H8000000F    'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(0).BackColor = &H8000000F      'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(0).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_TAPE1_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_CD_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE2  ใน SOUND IN ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_TAPE2_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE2_SI.Item(1).BackColor = &HFF&    'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_TAPE2_SI_B1 = False     'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(1).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_THE_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_TAPE1_SI_B1 = True      'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_CD_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)           'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE2 ใน SOUND IN ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_TAPE2_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็นสีแดง
Form1.LED_TAPE2_SI.Item(2).BackColor = &HFF&  'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_TAPE2_SI_B1 = False  'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(2).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_THE_SI_B1 = True  'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_TAPE1_SI_B1 = True  'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_CD_SI_B1 = True  'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)  'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อมูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE2 ใน SOUND IN ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_TAPE2_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้เป็นสีแดง
Form1.LED_TAPE2_SI.Item(3).BackColor = &HFF&  'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_TAPE2_SI_B1 = False  'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(3).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(3).STATUS_LED_THE_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_TAPE1_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_CD_SI_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 4 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TAPE2  ใน SOUND IN ของกลุ่มที่ 5
If BufStatus(4).STATUS_LED_TAPE2_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TAPE2_SI.Item(4).BackColor = &HFF&      'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(4).STATUS_LED_TAPE2_SI_B1 = False       'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TEH_SI.Item(4).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE1_SI.Item(4).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(4).BackColor = &H8000000F    'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(4).STATUS_LED_THE_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_TAPE1_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_CD_SI_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(4), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End Sub

Private Sub CM_TEH_SI_Click(Index As Integer)
If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND IN ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_THE_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SI.Item(0).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_THE_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TAPE1_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(0).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(0).BackColor = &H8000000F    'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_TAPE1_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_TAPE2_SI_B1 = True        'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_CD_SI_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND IN ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_THE_SI_B1 = True Then  'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SI.Item(1).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_THE_SI_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Form1.LED_TAPE1_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(1).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_TAPE1_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_TAPE2_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_CD_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index) 'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND IN ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_THE_SI_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SI.Item(2).BackColor = &HFF& 'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_THE_SI_B1 = False 'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TAPE1_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(2).BackColor = &H8000000F 'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_TAPE1_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_TAPE2_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_CD_SI_B1 = True 'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Temp = Send_data(BufStatus(2), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND IN ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_THE_SI_B1 = True Then   'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SI.Item(3).BackColor = &HFF&          'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_THE_SI_B1 = False           'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TAPE1_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(3).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(3).BackColor = &H8000000F      'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_TAPE1_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_TAPE2_SI_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_CD_SI_B1 = True             'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)                'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 4 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND IN ของกลุ่มที่ 5
If BufStatus(4).STATUS_LED_THE_SI_B1 = True Then   'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SI.Item(4).BackColor = &HFF&          'ให้กำหนดให้สีของ LED เป็นสีแดง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(4).STATUS_LED_THE_SI_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_TAPE1_SI.Item(4).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_TAPE2_SI.Item(4).BackColor = &H8000000F  'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_CD_SI.Item(4).BackColor = &H8000000F     'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(4).STATUS_LED_TAPE1_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_TAPE2_SI_B1 = True         'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(4).STATUS_LED_CD_SI_B1 = True            'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(4), Index)              'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

End Sub

Private Sub CM_TEH_SO_Click(Index As Integer)

If Index = 0 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND OUT ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_THE_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SO.Item(0).BackColor = &HFF&         'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(0).STATUS_LED_THE_SO_B1 = False          'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_2P_SO.Item(0).BackColor = &H8000000F     'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(0).BackColor = &H8000000F     'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(0).BackColor = &H8000000F     'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(0).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(0).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(0), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 1 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND OUT ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_THE_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SO.Item(1).BackColor = &HFF&       'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(1).STATUS_LED_THE_SO_B1 = False        'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_2P_SO.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(1).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(1).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(1), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 2 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND OUT ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_THE_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SO.Item(2).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(2).STATUS_LED_THE_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_2P_SO.Item(2).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(2).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(2).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_2P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_3P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(2).STATUS_LED_4P_SO_B1 = True           'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(2), Index)             'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 3 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND OUT ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_THE_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SO.Item(3).BackColor = &HFF&        'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(3).STATUS_LED_THE_SO_B1 = False         'กำหนดให้ตัวแปรที่เก็บค่ามีค่าเป็น
FALSE เพื่อนำไปกำหนดข้อมูลที่จะส่ง
Form1.LED_2P_SO.Item(3).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_3P_SO.Item(3).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
Form1.LED_4P_SO.Item(3).BackColor = &H8000000F   'เคลียสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(3).STATUS_LED_2P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_3P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
BufStatus(3).STATUS_LED_4P_SO_B1 = True          'กำหนดให้ตัวแปรที่เก็บค่าอื่นๆใน
SOUND IN มีค่าเป็น TRUE เพื่อจะได้ไม้นำไปกำหนดค่าในการส่ง
Temp = Send_data(BufStatus(3), Index)            'เรียกฟังชั่นที่จะสร้างข้อมูลที่จะส่งและส่งข้อ
มูล
End If
End If

If Index = 4 Then 'ถ้าหากปุ่มที่กดเป็นปุ่ม TEH ใน SOUND OUT ของกลุ่มที่ 5
If BufStatus(4).STATUS_LED_THE_SO_B1 = True Then 'ถ้า Status ของ LED ของปุ่มนี้ไม่ได้
เป็นสีแดง
Form1.LED_TEH_SO.Item(4).BackColor = &HFF&       'ให้กำหนดให้สีของ LED เป็นสีแดง
BufStatus(4).STATUS_LED_THE_SO_B1 = False        'กำหนดให้อีก 4 บิทเป็น FALSE เพื่อให้
RELAY 4 ตัวทำงาน
BufStatus(4).STATUS_LED_2P_SO_B1 = False         'กำหนดให้อีก 4 บิทเป็น FALSE เพื่อให้
RELAY 4 ตัวทำงาน
BufStatus(4).STATUS_LED_3P_SO_B1 = False         'กำหนดให้อีก 4 บิทเป็น FALSE เพื่อให้
RELAY 4 ตัวทำงาน
BufStatus(4).STATUS_LED_4P_SO_B1 = False         'กำหนดให้อีก 4 บิทเป็น FALSE เพื่อให้
RELAY 4 ตัวทำงาน
Temp = Send_data(BufStatus(4), Index)            'เรียกฟังชั่นเพื่อกำหนดค่าที่จะส่งและส่งออก
พอรท์
Else 'ถ้าก่อนหน้านี้มีการกดปุ่มนี้แล้ว
Form1.LED_TEH_SO.Item(4).BackColor = &H8000000F  'ให้กำหนดให้สีของ LED ดับ
BufStatus(4).STATUS_LED_THE_SO_B1 = True         'กำหนดให้อีก 4 บิทเป็น TRUE เพื่อให้
RELAY 4 ตัวไม่ทำงาน
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(4).STATUS_LED_2P_SO_B1 = True        'กำหนดให้อีก 4 บิทเป็น TRUE เพื่อให้
RELAY 4 ตัวไม่ทำงาน
BufStatus(4).STATUS_LED_3P_SO_B1 = True        'กำหนดให้อีก 4 บิทเป็น TRUE เพื่อให้
RELAY 4 ตัวไม่ทำงาน
BufStatus(4).STATUS_LED_4P_SO_B1 = True        'กำหนดให้อีก 4 บิทเป็น TRUE เพื่อให้
RELAY 4 ตัวไม่ทำงาน
Temp = Send_data(BufStatus(4), Index)          'เรียกฟังชั่นเพื่อกำหนดค่าที่จะส่งและส่งออก
พอรท์
End If
End If

End Sub
'เป็นการตอบสนองเหตุการการกด ปุ่ม เลือกบอด์ที่ 1 ของแต่ละกลุ่ม
Private Sub COMMAND_B1_Click(Index As Integer)
If Index = 0 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 1 ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_B1 = True Then      'ถ้า LED ของ BORD ที่ 1 ของกลุ่มที่ 1
ไม่เป็นสีแดง
Form1.LED_BORD1.Item(0).BackColor = &HFF&      'กำหนดให้ LED ของ BORD ที่ 1กลุ่ม
ที่ 1 ให้เป็นสีแดง
BufStatus(0).STATUS_LED_B1 = False             'กำหนดให้ STATUS ของ LED เป็น
False เพื่อป้องกันการกดซ้ำและกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B2 = True              'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B3 = True              'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B4 = True              'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Temp = Send_data(BufStatus(0), Index)
End If
End If

If Index = 1 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 1 ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_B1 = True Then 'ถ้า LED ของ BORD ที่ 1 ของกลุ่มที่ 2 ไม่เป็นสี
แดง
Form1.LED_BORD1.Item(1).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 1กลุ่มที่ 2 ให้
เป็นสีแดง
BufStatus(1).STATUS_LED_B1 = False
Form1.LED_BORD2.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B2 = True               'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B3 = True               'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B4 = True               'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(1), Index)
End If
End If

If Index = 2 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 1 ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_B1 = True Then 'ถ้า LED ของ BORD ที่ 1 ของกลุ่มที่ 3 ไม่เป็นสี
แดง
Form1.LED_BORD1.Item(2).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 1กลุ่มที่ 3 ให้
เป็นสีแดง
BufStatus(2).STATUS_LED_B1 = False
Form1.LED_BORD2.Item(2).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(2).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(2).BackColor = &H8000000F   'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(2).BackColor = &H8000000F   'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(2), Index)
End If
End If

If Index = 3 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 1 ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_B1 = True Then 'ถ้า LED ของ BORD ที่ 1 ของกลุ่มที่ 4 ไม่เป็นสี
แดง
Form1.LED_BORD1.Item(3).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 1กลุ่มที่ 4 ให้
เป็นสีแดง
BufStatus(3).STATUS_LED_B1 = False
Form1.LED_BORD2.Item(3).BackColor = &H8000000F   'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(3).BackColor = &H8000000F   'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(3).BackColor = &H8000000F   'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(3), Index)
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End Sub

Private Sub COMMAND_B2_Click(Index As Integer)
If Index = 0 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 2 ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_B2 = True Then 'ถ้า LED ของ BORD ที่ 2 ของกลุ่มที่ 1 ไม่เป็นสี
แดง
Form1.LED_BORD2.Item(0).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 2 กลุ่มที่ 1
ให้เป็นสีแดง
BufStatus(0).STATUS_LED_B2 = False
Form1.LED_BORD1.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B1 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B3 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B4 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(0), Index)
End If
End If

If Index = 1 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 2 ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_B2 = True Then 'ถ้า LED ของ BORD ที่ 2 ของกลุ่มที่ 2 ไม่เป็นสี
แดง
Form1.LED_BORD2.Item(1).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 2 กลุ่มที่ 2
ให้เป็นสีแดง
BufStatus(1).STATUS_LED_B2 = False
Form1.LED_BORD1.Item(1).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(1).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(1), Index)
End If
End If

If Index = 2 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 2 ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_B2 = True Then 'ถ้า LED ของ BORD ที่ 2 ของกลุ่มที่ 3 ไม่เป็นสี
แดง
Form1.LED_BORD2.Item(2).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 2 กลุ่มที่ 3
ให้เป็นสีแดง
BufStatus(2).STATUS_LED_B2 = False
Form1.LED_BORD1.Item(2).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(2).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(2).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(2), Index)
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 3 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 2 ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_B2 = True Then 'ถ้า LED ของ BORD ที่ 2 ของกลุ่มที่ 4 ไม่เป็นสี
แดง
Form1.LED_BORD2.Item(3).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 2 กลุ่มที่ 4
ให้เป็นสีแดง
BufStatus(3).STATUS_LED_B2 = False
Form1.LED_BORD1.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B1 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B3 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B4 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(3), Index)
End If
End If

End Sub

Private Sub COMMAND_B3_Click(Index As Integer)
If Index = 0 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 3 ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_B3 = True Then 'ถ้า LED ของ BORD ที่ 3 ของกลุ่มที่ 1 ไม่เป็นสี
แดง
Form1.LED_BORD3.Item(0).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 3 กลุ่มที่ 1
ให้เป็นสีแดง
BufStatus(0).STATUS_LED_B3 = False
Form1.LED_BORD1.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(0).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(0).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(0).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(0), Index)
End If
End If

If Index = 1 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 3 ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_B3 = True Then 'ถ้า LED ของ BORD ที่ 3 ของกลุ่มที่ 2 ไม่เป็นสี
แดง
Form1.LED_BORD3.Item(1).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 3 กลุ่มที่ 2
ให้เป็นสีแดง
BufStatus(1).STATUS_LED_B3 = False
Form1.LED_BORD1.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(1).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(1), Index)
End If
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If Index = 2 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 3 ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_B3 = True Then 'ถ้า LED ของ BORD ที่ 3 ของกลุ่มที่ 3 ไม่เป็นสี
แดง
Form1.LED_BORD3.Item(2).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 3 กลุ่มที่ 3
ให้เป็นสีแดง
BufStatus(2).STATUS_LED_B3 = False
Form1.LED_BORD1.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B1 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B2 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B4 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(2), Index)
End If
End If

If Index = 3 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 3 ของกลุ่มที่ 4
If BufStatus(3).STATUS_LED_B3 = True Then 'ถ้า LED ของ BORD ที่ 3 ของกลุ่มที่ 4 ไม่เป็นสี
แดง
Form1.LED_BORD3.Item(3).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 3 กลุ่มที่ 4
ให้เป็นสีแดง
BufStatus(3).STATUS_LED_B3 = False
Form1.LED_BORD1.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B1 = True 'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
```

เอกสารนี้ใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(3).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD4.Item(3).BackColor = &H8000000F  'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B4 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(3), Index)
End If
End If

End Sub

Private Sub COMMAND_B4_Click(Index As Integer)
If Index = 0 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 4 ของกลุ่มที่ 1
If BufStatus(0).STATUS_LED_B4 = True Then 'ถ้า LED ของ BORD ที่ 4 ของกลุ่มที่ 1 ไม่เป็นสี
แดง
Form1.LED_BORD4.Item(0).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 4 กลุ่มที่ 1
ให้เป็นสีแดง
BufStatus(0).STATUS_LED_B4 = False                   'กำหนดให้ STATUS  ของ LED เป็น
False เพื่อป้องกันการกดซ้ำและกำหนดหมายเลข BORD
Form1.LED_BORD1.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(0).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(0).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED  ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(0), Index)
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End If

If Index = 1 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 4 ของกลุ่มที่ 2
If BufStatus(1).STATUS_LED_B4 = True Then 'ถ้า LED ของ BORD ที่ 4 ของกลุ่มที่ 2 ไม่เป็นสี
แดง
Form1.LED_BORD4.Item(1).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 4 กลุ่มที่ 2
ให้เป็นสีแดง
BufStatus(1).STATUS_LED_B4 = False                    'กำหนดให้ STATUS ของ LED เป็น
False เพื่อป้องกันการกดซ้ำและกำหนดหมายเลข BORD
Form1.LED_BORD1.Item(1).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B1 = True                     'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(1).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B3 = True                     'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(1).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(1).STATUS_LED_B2 = True                     'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(1), Index)
End If
End If

If Index = 2 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 4 ของกลุ่มที่ 3
If BufStatus(2).STATUS_LED_B4 = True Then 'ถ้า LED ของ BORD ที่ 4 ของกลุ่มที่ 3 ไม่เป็นสี
แดง
Form1.LED_BORD4.Item(2).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 4 กลุ่มที่ 3
ให้เป็นสีแดง
BufStatus(2).STATUS_LED_B4 = False                    'กำหนดให้ STATUS ของ LED เป็น
False เพื่อป้องกันการกดซ้ำและกำหนดหมายเลข BORD
Form1.LED_BORD1.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BufStatus(2).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(2).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(2).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(2), Index)
End If
End If

If Index = 3 Then 'ถ้าเป็นการกดปุ่ม เลือก BORD ที่ 4 ของกลุ่มที่ 3
If BufStatus(3).STATUS_LED_B4 = True Then 'ถ้า LED ของ BORD ที่ 4 ของกลุ่มที่ 3 ไม่เป็นสี
แดง
Form1.LED_BORD4.Item(3).BackColor = &HFF& 'กำหนดให้ LED ของ BORD ที่ 4 กลุ่มที่ 3
ให้เป็นสีแดง
BufStatus(3).STATUS_LED_B4 = False                   'กำหนดให้ STATUS ของ LED เป็น
False เพื่อป้องกันการกดซ้ำและกำหนดหมายเลข BORD
Form1.LED_BORD1.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B1 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD3.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B3 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Form1.LED_BORD2.Item(3).BackColor = &H8000000F 'เปลี่ยนสีของ LED อื่นๆให้ดับ
BufStatus(3).STATUS_LED_B2 = True                    'กำหนดให้ STATUS ของ LED ตัวอื่น
เป็น True เพื่อจะได้ไม่นำค่าตัวนี้ไปกำหนดหมายเลข BORD
Temp = Send_data(BufStatus(3), Index)
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End If
End Sub


Private Sub Command8_Click()
End
End Sub

Private Sub Form_Load()
Dim I As Integer
'โหลดค่าเริ่มต้นให้กับตัวแปรที่ใช้แสดงสถานะของ LED
For I = 0 To 4
BufStatus(I).STATUS_LED_2P_SO_B1 = True
BufStatus(I).STATUS_LED_3P_SO_B1 = True
BufStatus(I).STATUS_LED_4P_SO_B1 = True
BufStatus(I).STATUS_LED_B1 = True
BufStatus(I).STATUS_LED_B2 = True
BufStatus(I).STATUS_LED_B3 = True
BufStatus(I).STATUS_LED_B4 = True
BufStatus(I).STATUS_LED_CD_SI_B1 = True
BufStatus(I).STATUS_LED_TAPE1_SI_B1 = True
BufStatus(I).STATUS_LED_TAPE2_SI_B1 = True
BufStatus(I).STATUS_LED_THE_SI_B1 = True
BufStatus(I).STATUS_LED_THE_SO_B1 = True
Next I
MSComm1.PortOpen = True
MSComm2.PortOpen = True
End Sub

Private Sub Frame1_DragDrop(Index As Integer, Source As Control, X As Single, Y As Single)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
End Sub

Private Sub MSComm1_OnComm()
Dim BufText As String       'เป็นตัวแปรไว้เก็บข้อมูลที่รับเข้ามาจาก RS232
Dim Group As String         'เป็นตัวแปรที่ใช้เก็บหมายเลขของกลุ่ม
Dim Bord As String          'เป็นตัวแปรที่ใช้เก็บหมายเลขของ BORD
If MSComm1.InBufferCount >= 15 Then 'ถ้าข้อมูลที่รับเข้ามามีจำนวนมากกว่า 14
BufText = ""                        'เคลียข้อมูลเก่าที่เก็บในตัวแปร
BufText = MSComm1.Input             'นำข้อมูลที่อยู่ในที่พักข้อมูลมาเก็นในตัวแปร
If (Int(InStr(BufText, "CALL")) > Int(InStr(BufText, "END"))) And Int(InStr(BufText,
"CALL")) <> 0 Then 'ถ้าหากคำอื่นอยู่ก่อนหน้าคำว่า "CALL"
  BufText = Right(BufText, Int(Len(BufText)) - Int(InStr(BufText, "END") + 2)) 'ทำการตัดคำที่
อยู่ก่อนหน้า "CALL" ออก
 End If
'ก่อนที่จะตอบสนองการเรียก CALL จากบอรท์ต้องตรวจก่อนว่ามีคำว่า CALL และ END อยู่ใน
ประโยคข้อมูลที่รับเข้ามา
 If InStr(BufText, "CALLBY") <> 0 And InStr(BufText, "END") <> 0 Then 'ถ้ามีคำว่า CALL
และ END
BufText = Mid(BufText, Int(InStr(BufText, "CALLBY") + 6), Int(InStr(BufText, "END") - 1) -
Int(InStr(BufText, "CALLBY") + 5)) 'ตัดคำว่า "CALLBY" และ "END"
Group = Mid(BufText, Int(InStr(BufText, "G") + 1), Int(InStr(BufText, "B") - 1) -
Int(InStr(BufText, "G"))) 'ทำการแยกเอาเฉพาะหมายเลขของกลุ่มที่เป็นตัวอักษร
Bord = Mid(Right(BufText, 2), 2, 1) 'ทำการแยกเอาเฉพาะหมายเลขของ BORD ที่เป็นตัวอักษร

If Bord = "0" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 0
Form1.LED_BORD1.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "1" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 1
Form1.LED_BORD2.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "2" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 2
Form1.LED_BORD3.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "3" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 3
Form1.LED_BORD4.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If


End If
End If 'สิ้นสุดการตอบสนองเมื่อมีการกดปุ่ม CALL
End Sub

Private Sub MSComm2_OnComm()
Dim BufText As String      'เป็นตัวแปรไว้เก็บข้อมูลที่รับเข้ามาจาก RS232
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Dim Group As String        'เป็นตัวแปรที่ใช้เก็บหมายเลขของกลุ่ม
Dim Bord As String         'เป็นตัวแปรที่ใช้เก็บหมายเลขของ BORD
'CALLBY G0B1  END
If MSComm2.InBufferCount >= 15 Then 'ถ้าข้อมูลที่รับเข้ามามีจำนวนมากกว่า 14
BufText = ""                       'เคลียข้อมูลเก่าที่เก็บในตัวแปร
BufText = MSComm2.Input            'นำข้อมูลที่อยู่ในที่พักข้อมูลมาเก็นในตัวแปร
 If (Int(InStr(BufText, "CALL")) > Int(InStr(BufText, "END"))) And Int(InStr(BufText,
"CALL")) <> 0 Then  'ถ้าหากคำอื่นอยู่ก่อนหน้าคำว่า "CALL"
   BufText = Right(BufText, Int(Len(BufText)) - Int(InStr(BufText, "END") + 2)) 'ทำการตัดคำที่
อยู่ก่อนหน้า "CALL" ออก
 End If
 'ก่อนที่จะตอบสนองการเรียก CALL จากบอรท์ต้องตรวจก่อนว่ามีคำว่า CALL และ END อยู่ใน
ประโยคข้อมูลที่รับเข้ามา
 If InStr(BufText, "CALLBY") <> 0 And InStr(BufText, "END") <> 0 Then  'ถ้ามีคำว่า CALL
และ END
BufText = Mid(BufText, Int(InStr(BufText, "CALLBY") + 6), Int(InStr(BufText, "END") - 1) -
Int(InStr(BufText, "CALLBY") + 5)) 'ตัดคำว่า "CALLBY" และ "END"
Group = Mid(BufText, Int(InStr(BufText, "G") + 1), Int(InStr(BufText, "B") - 1) -
Int(InStr(BufText, "G"))) 'ทำการแยกเอาเฉพาะหมายเลขของกลุ่มที่เป็นตัวอักษร
Bord = Mid(Right(BufText, 2), 2, 1) 'ทำการแยกเอาเฉพาะหมายเลขของ BORD ที่เป็นตัวอักษร

If Bord = "0" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 0
Form1.LED_BORD1.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000   'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True    'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "1" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 1
Form1.LED_BORD2.Item(Val(Group)).BackColor = Val("&HFFFF&")
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "2" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 2
Form1.LED_BORD3.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If
If Bord = "3" Then 'ถ้าข้อมูลที่เรียกเข้ามาเป็นของกลุ่มที่ 3
Form1.LED_BORD4.Item(Val(Group)).BackColor = Val("&HFFFF&")
Form1.Timer1.Interval = 1000  'กำหนดเวลาให้ TIMRE เพื่อไปทำงานในฟังชั่นเมื่อครบ 1 วินาที
Form1.Timer1.Enabled = True   'กำหนดให้ TIMER เริ่มนับ
 MSComm1.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
 MSComm2.InBufferCount = 0  'ทำการเคลียตัวชี้ข้อมูลให้เริ่มต้นนับใหม่
End If


End If
End If 'สิ้นสุดการตอบสนองเมื่อมีการกดปุ่ม CALL


End Sub


Private Sub Timer1_Timer()
Dim I As Integer
For I = 0 To 3 'กำหนดให้มีการวน Loop ที่จะแสดงข้อมูลเก่าที่แสดงถึงการติดต่อกับ บอด ใดๆของ
กลุ่มทั้งหมด
If BufStatus(I).STATUS_LED_B1 = True Then 'ถ้าเดิมสีของ LEDของบอร์ทที่ 1 เดิมเป็น ดับ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Form1.LED_BORD1.Item(I).BackColor = &H80000004 'ถ้า Status ที่ใช้แสดง สีของ LED เดิม เป็นดับก็ให้แสดงว่าดับเหมื่อนเดิม
Else                                    'ถ้าเดิมสีของ LEDของบอรท์ที่ 1 เดิมเป็น แดง
Form1.LED_BORD1.Item(I).BackColor = &HFF& 'ถ้า Status ที่ใช้แสดง สีของ LED เดิมเป็นติดก็ให้แสดงว่าติดเหมื่อนเดิม
End If
If BufStatus(I).STATUS_LED_B2 = True Then 'ถ้าเดิมสีของ LEDของบอรท์ที่ 2 เดิมเป็น ดับ
Form1.LED_BORD2.Item(I).BackColor = &H80000004 'ถ้า Status ที่ใช้แสดง สีของ LED เดิม เป็นดับก็ให้แสดงว่าดับเหมื่อนเดิม
Else                                    'ถ้าเดิมสีของ LEDของบอรท์ที่ 1 เดิมเป็น แดง
Form1.LED_BORD2.Item(I).BackColor = &HFF& 'ถ้า Status ที่ใช้แสดง สีของ LED เดิมเป็นติดก็ให้แสดงว่าติดเหมื่อนเดิม
End If
If BufStatus(I).STATUS_LED_B3 = True Then 'ถ้าเดิมสีของ LEDของบอรท์ที่ 3 เดิมเป็น ดับ
Form1.LED_BORD3.Item(I).BackColor = &H80000004 'ถ้า Status ที่ใช้แสดง สีของ LED เดิม เป็นดับก็ให้แสดงว่าดับเหมื่อนเดิม
Else                                    'ถ้าเดิมสีของ LEDของบอรท์ที่ 1 เดิมเป็น แดง
Form1.LED_BORD3.Item(I).BackColor = &HFF& 'ถ้า Status ที่ใช้แสดง สีของ LED เดิมเป็นติดก็ให้แสดงว่าติดเหมื่อนเดิม
End If
If BufStatus(I).STATUS_LED_B4 = True Then 'ถ้าเดิมสีของ LEDของบอรท์ที่ 4 เดิมเป็น ดับ
Form1.LED_BORD4.Item(I).BackColor = &H80000004 'ถ้า Status ที่ใช้แสดง สีของ LED เดิม เป็นดับก็ให้แสดงว่าดับเหมื่อนเดิม
Else                                    'ถ้าเดิมสีของ LEDของบอรท์ที่ 1 เดิมเป็น แดง
Form1.LED_BORD4.Item(I).BackColor = &HFF& 'ถ้า Status ที่ใช้แสดง สีของ LED เดิมเป็นติดก็ให้แสดงว่าติดเหมื่อนเดิม
End If
Next I
Timer1.Enabled = False                  'กำหนดให้ TIMER หยุดทำงาน
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

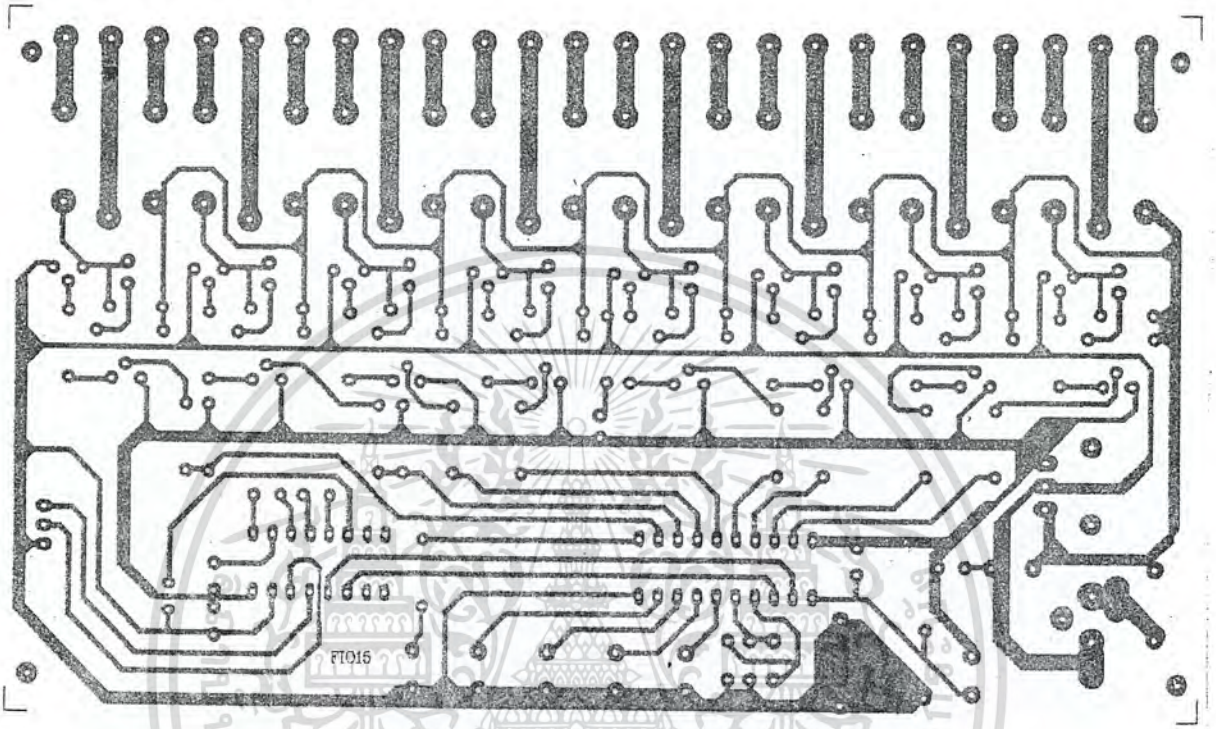
ภาคผนวก ค.

ตารางแสดงตำแหน่งการเลือกปรับสวิตซ์ใช้งานส่วนรับฟัง

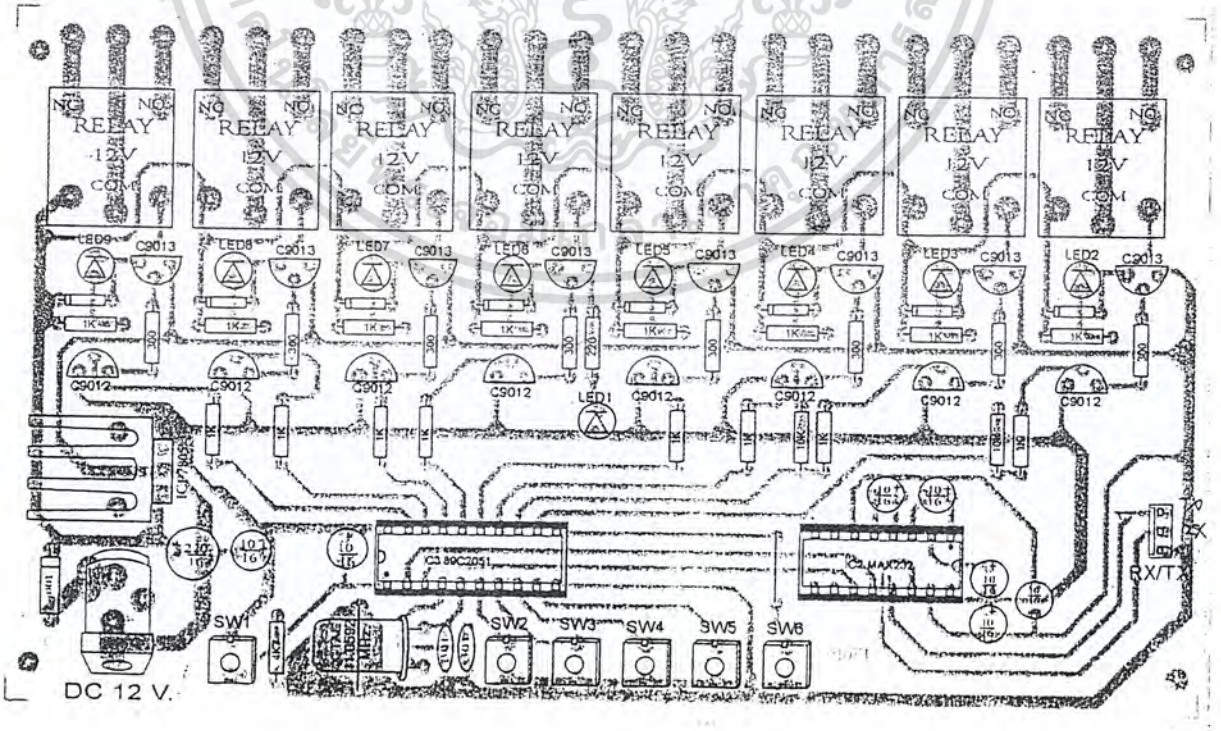
บิตที่1	บิตที่2	บิตที่3	บิตที่4	ตำแหน่ง
0	0	0	0	กลุ่มที่ 1 คนที่1
0	1	0	0	กลุ่มที่ 2 คนที่1
1	0	0	0	กลุ่มที่ 3 คนที่1
1	1	0	0	กลุ่มที่ 4 คนที่1
0	0	0	1	กลุ่มที่ 1 คนที่2
0	1	0	1	กลุ่มที่ 2 คนที่2
1	0	0	1	กลุ่มที่ 3 คนที่2
1	1	0	1	กลุ่มที่ 4 คนที่2
0	0	1	0	กลุ่มที่ 1 คนที่3
0	1	1	0	กลุ่มที่ 2 คนที่3
1	0	1	0	กลุ่มที่ 3 คนที่3
1	1	1	0	กลุ่มที่ 4 คนที่3
0	0	1	1	กลุ่มที่ 1 คนที่4
0	1	1	1	กลุ่มที่ 2 คนที่4
1	0	1	1	กลุ่มที่ 3 คนที่4
1	1	1	1	กลุ่มที่ 4 คนที่4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลายวงจรรูปที่8.4

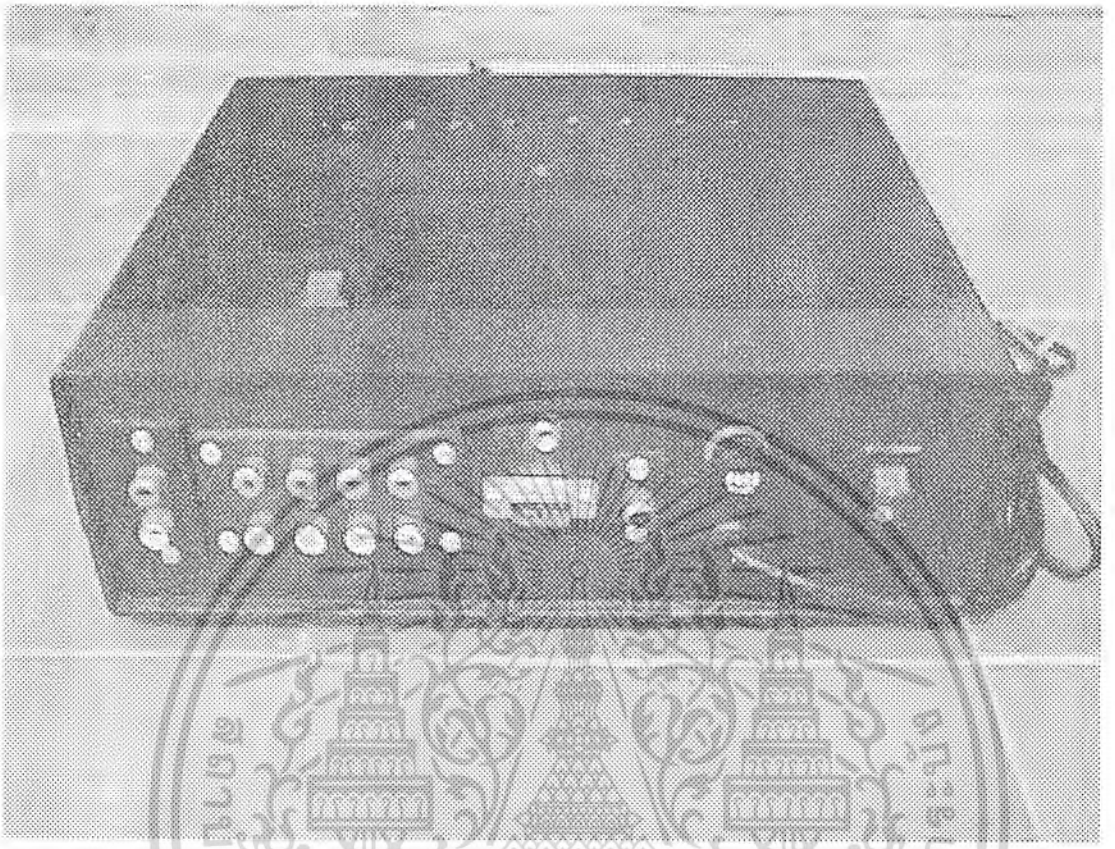


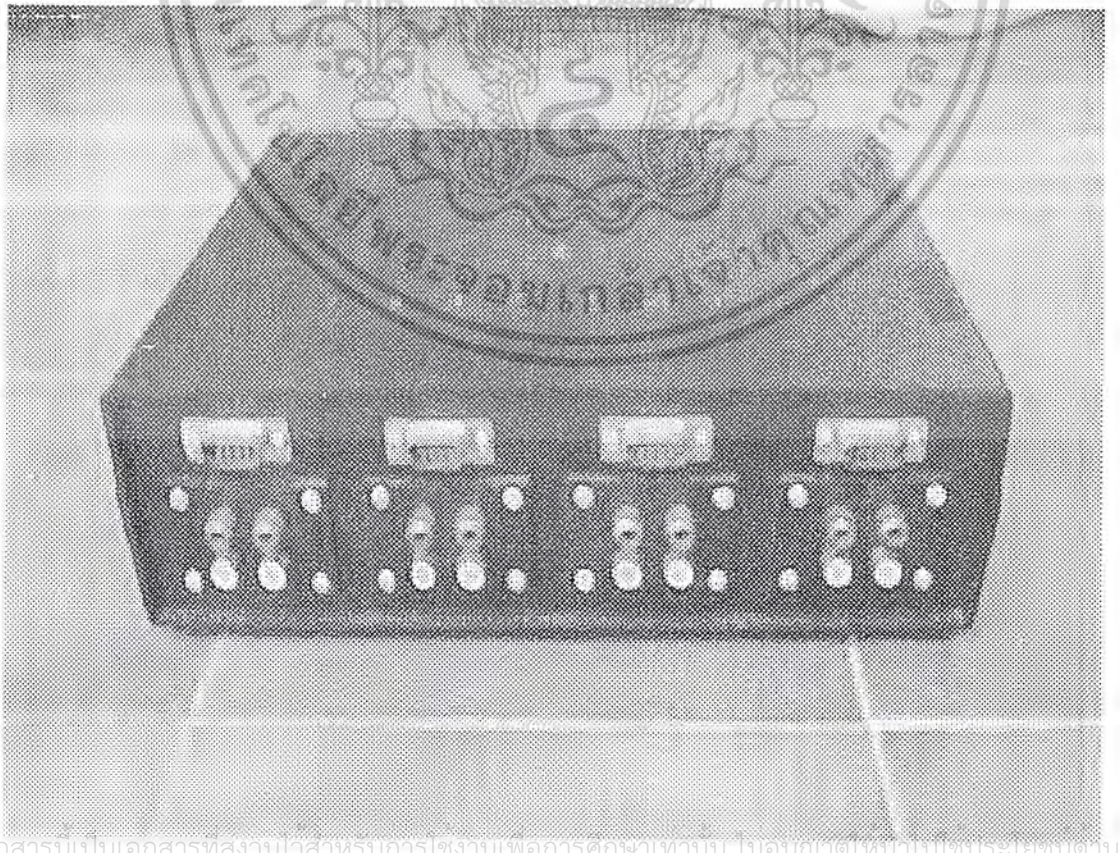
รูปที่ 3 แสดงลายวงจรพิมพ์ขนาดเท่าของจริง



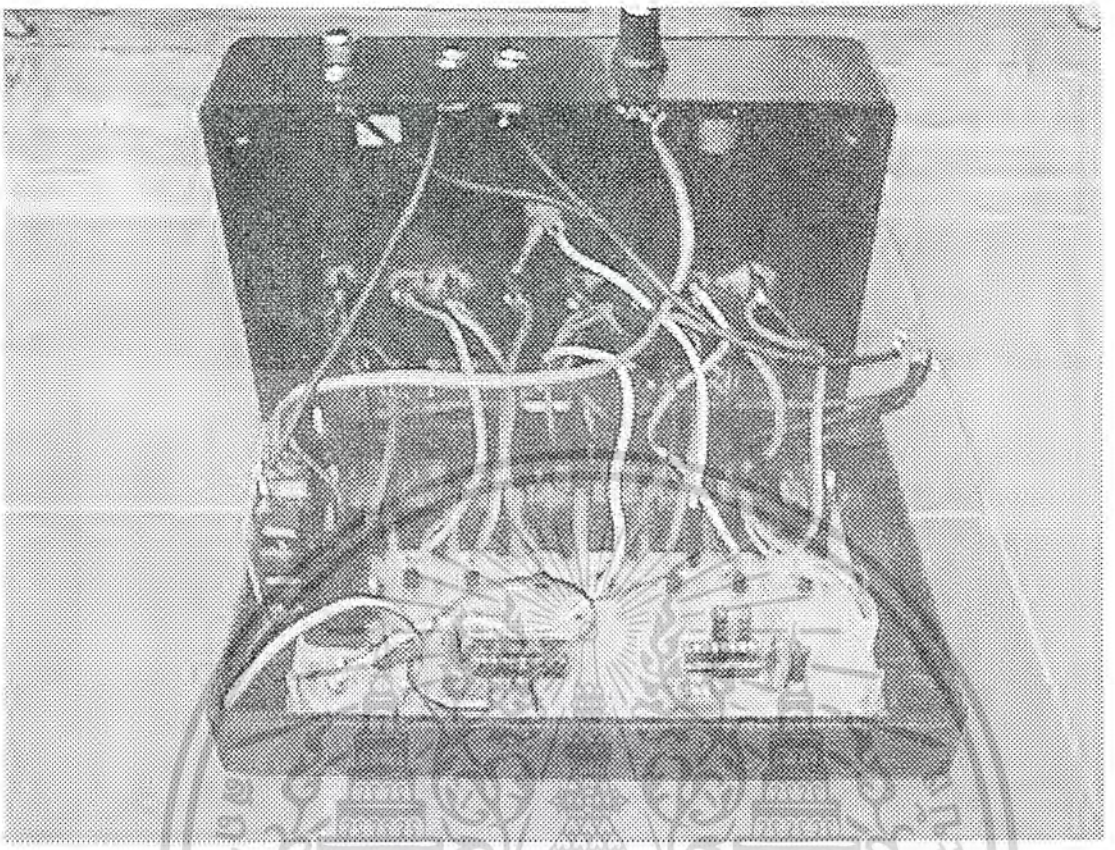
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

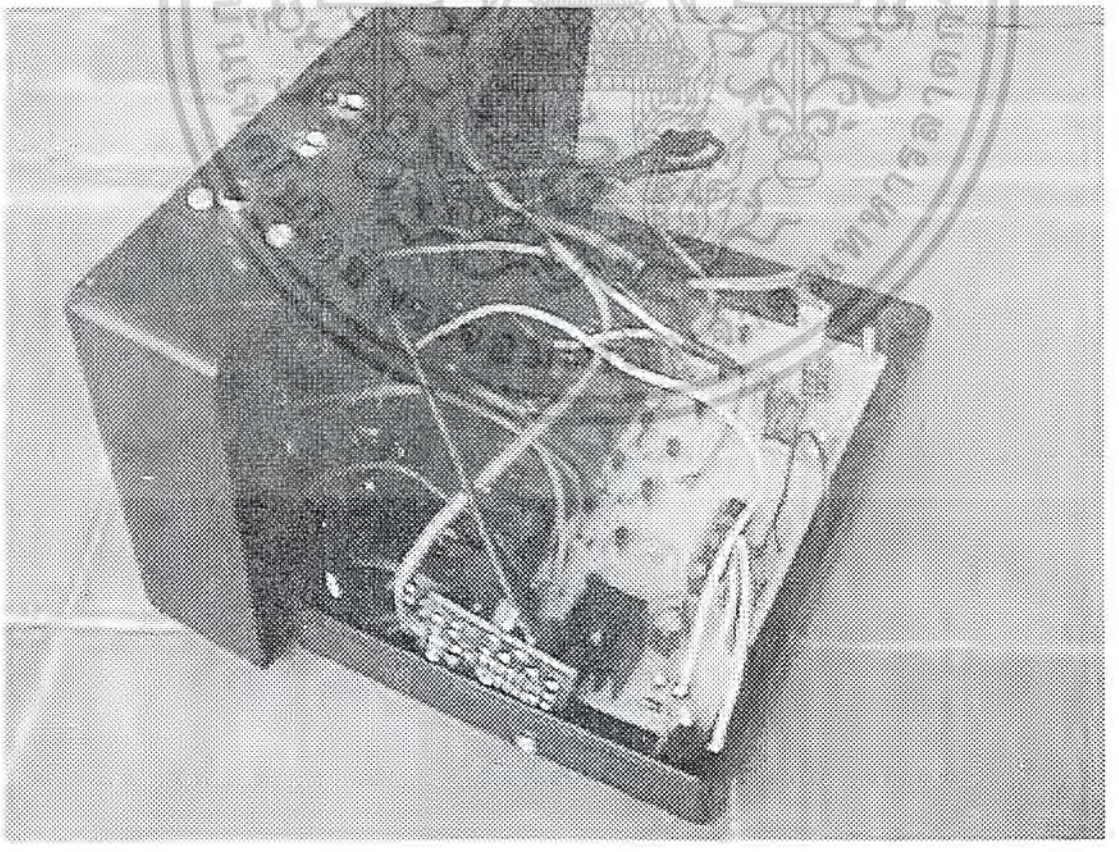
ภาพถ่ายเครื่องควบคุมซาวด์และส่วนรับฟัง



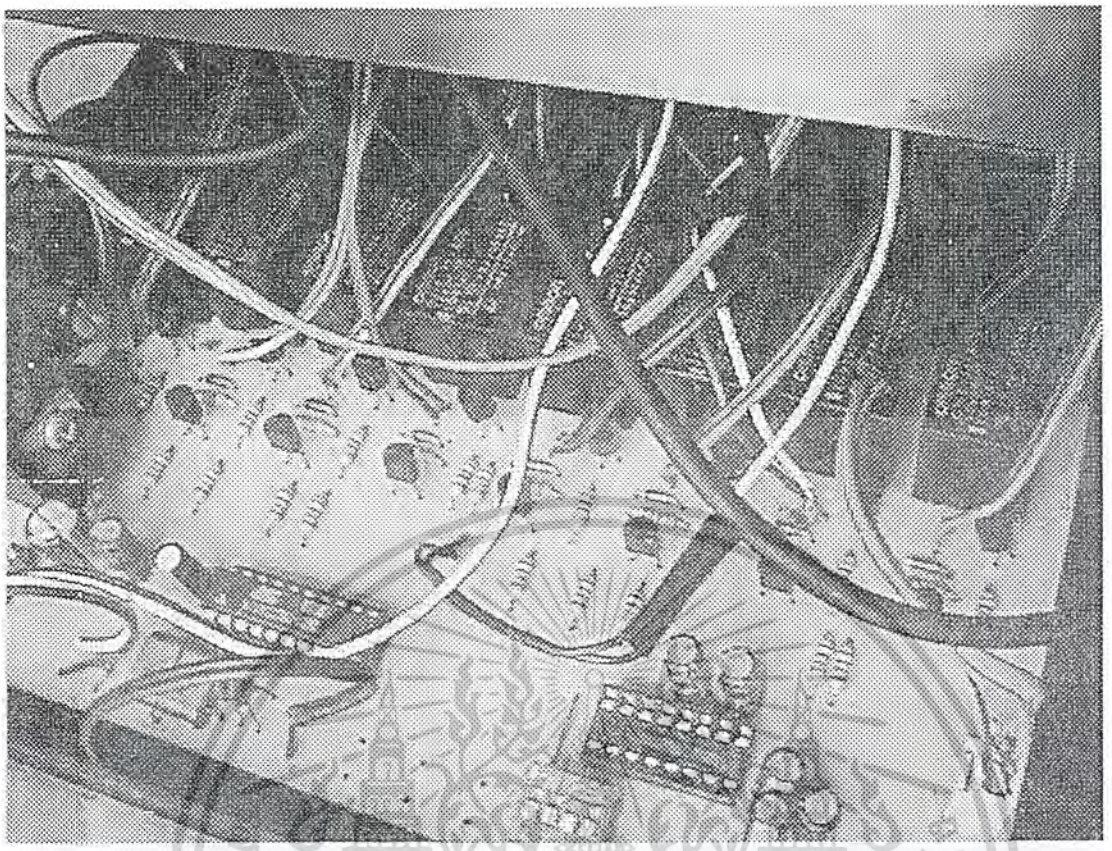


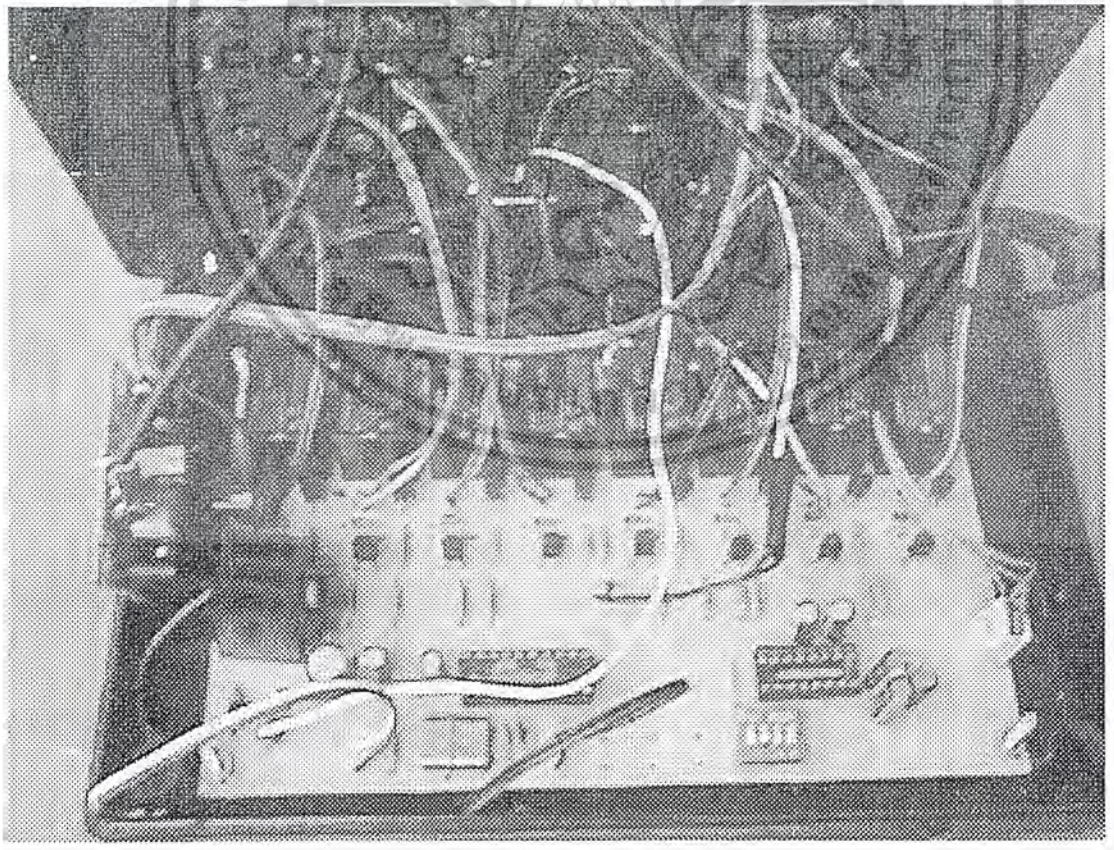
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Features

- Compatible with MCS-51™ Products
- 2 Kbytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7 V to 6 V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes

8-Bit Microcontroller with 2 Kbytes Flash

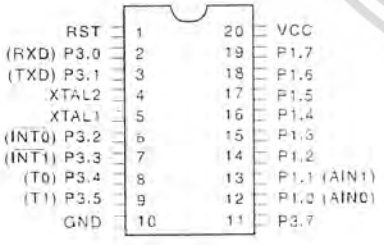
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2 Kbytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2 Kbytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

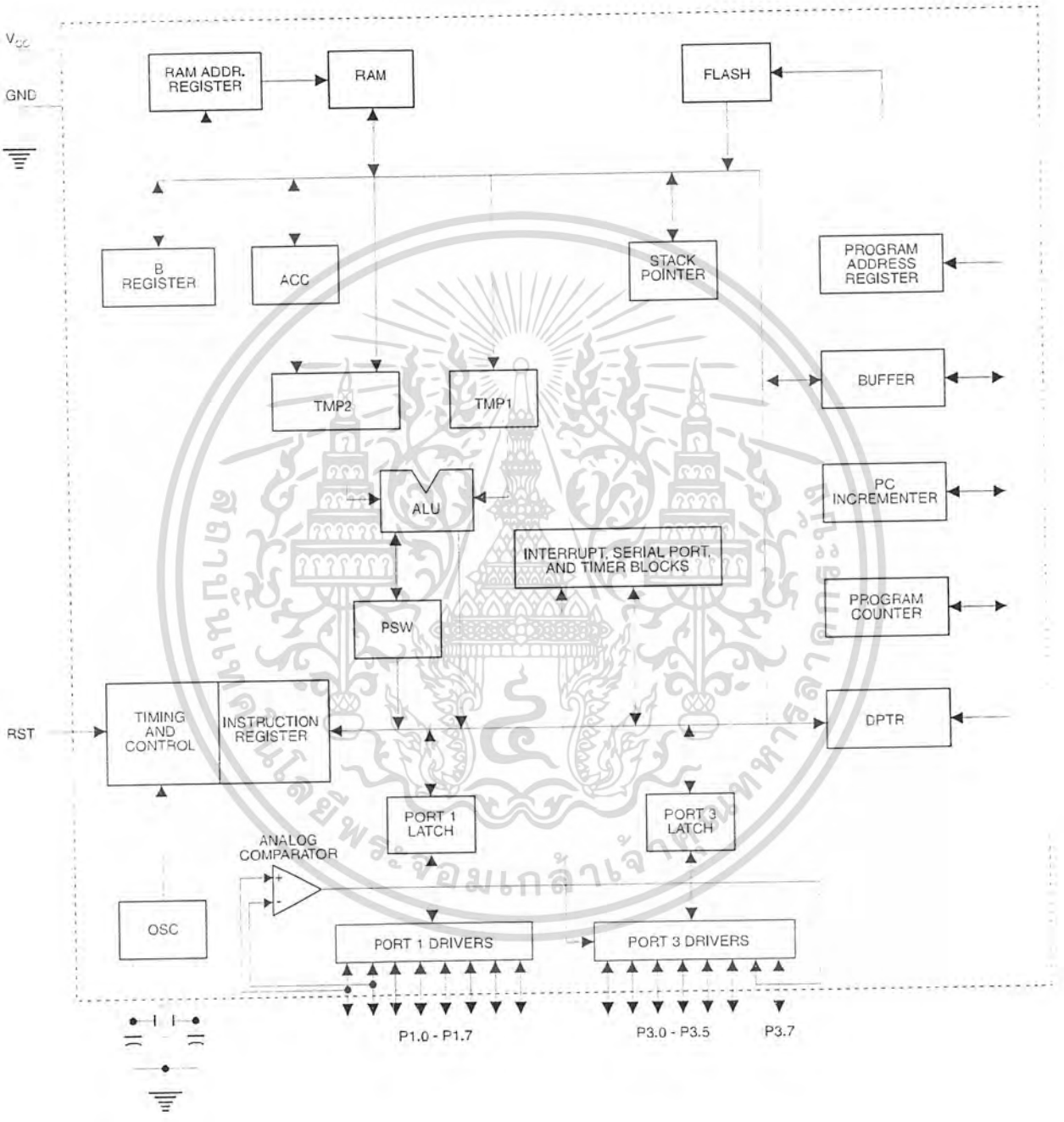
PDIP/SOIC



0368C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Block Diagram



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Pin Description

V_{CC}

Supply voltage.

GND

Ground.

Port 1

Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and program verification.

Port 3

Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

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

RST

Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

XTAL1

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

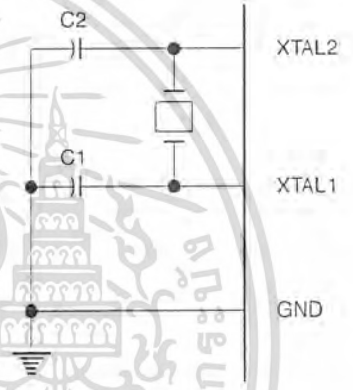
XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

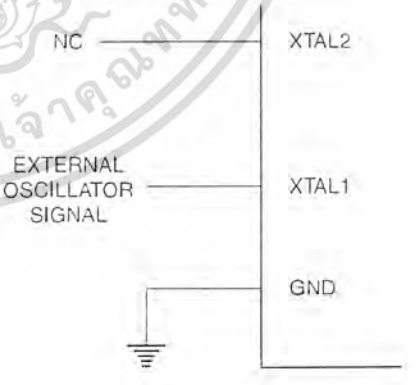
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Notes: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Special Function Registers

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

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

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

Table 1. AT89C2051 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H								0CFH
0C0H								0C7H
0B8H	IP XXX00000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0XX00000							0AFH
0A0H								0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		8FH
80H		SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000 87H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Restrictions on Certain Instructions

The AT89C2051 and is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 2 Kbytes of flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LJMP 7E0H would be a valid instruction for the AT89C2051 (with 2K of memory), whereas LJMP 900H would not.

1. Branching instructions:

LCALL, LJMP, ACALL , AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:

The AT89C2051 contains 128 bytes of internal data memory. Thus, in the AT89C2051 the stack depth is limited to 128 bytes, the amount of available RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Program Memory Lock Bits

On the chip are two lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

Lock Bit Protection Modes[1]

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2 Kbytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
   With all other pins floating, wait for greater than 10 milliseconds
2. Set pin RST to 'H'
   Set pin P3.2 to 'H'
3. Apply the appropriate combination of 'H' or 'L' logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic 'H' level and set pins P3.3 to P3.7 to the appropiate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2 Kbytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to 'L'
    set RST to 'L'
    Float all other I/O pins
    Turn Vcc power off

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$\overline{\text{Data}}$ Polling: The AT89C2051 features $\overline{\text{Data}}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{\text{Busy}}$: The Progress of byte programming can also be monitored by the RDY/$\overline{\text{BSY}}$ output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from 'L' to 'H'.
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire PEROM array (2 Kbytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST	P3.2/$\overline{\text{PROG}}$	P3.3	P3.4	P3.5	P3.7
Write Code Data(1,3)		12V	⌐‾_	L	H	H	H
Read Code Data(1)		H	H	L	L	H	H
Write Lock	Bit - 1	12V	‾_	H	H	H	H
	Bit - 2	12V	‾_	H	H	L	L
Chip Erase		12V	‾_(2)	H	L	L	L
Read Signature Byte		H	H	L	L	L	L

Notes: 1. The internal PEROM address counter is reset to 000H on the rising edge of RST and is advanced by a positive pulse at XTAL1 pin.
2. Chip Erase requires a 10 ms $\overline{\text{PROG}}$ pulse.
3. P3.1 is pulled Low during programming to indicate RDY/$\overline{\text{BSY}}$\.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Figure 3. Programming the Flash Memory

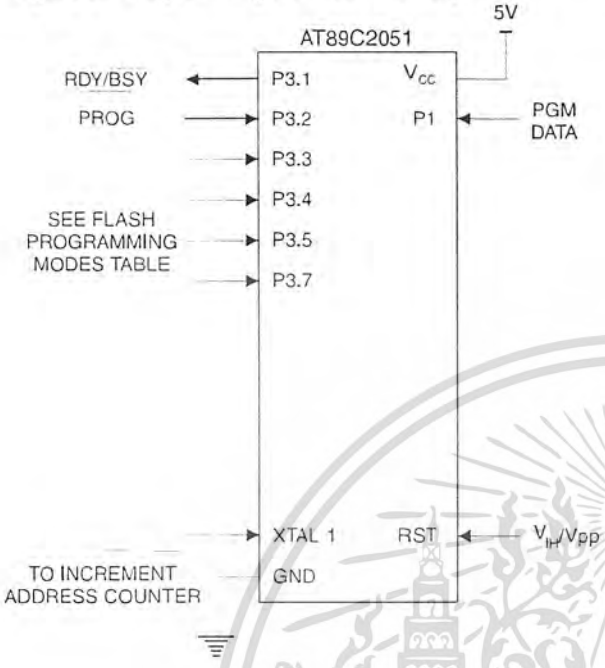
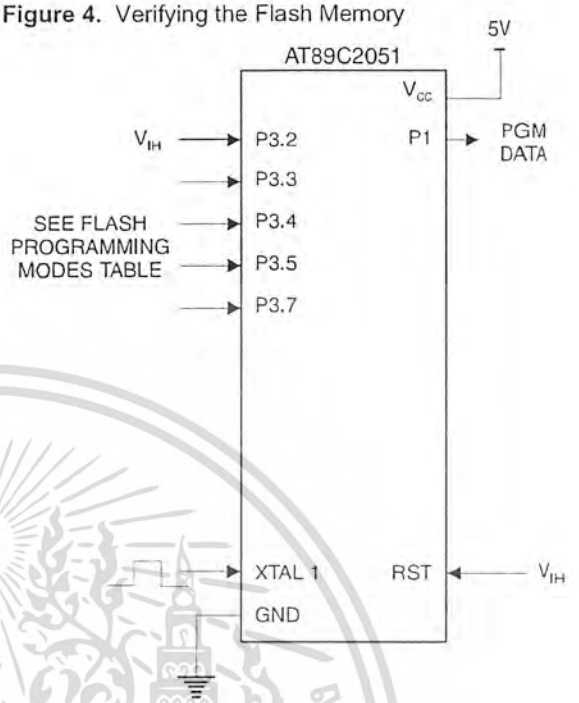


Figure 4. Verifying the Flash Memory



Flash Programming and Verification Characteristics

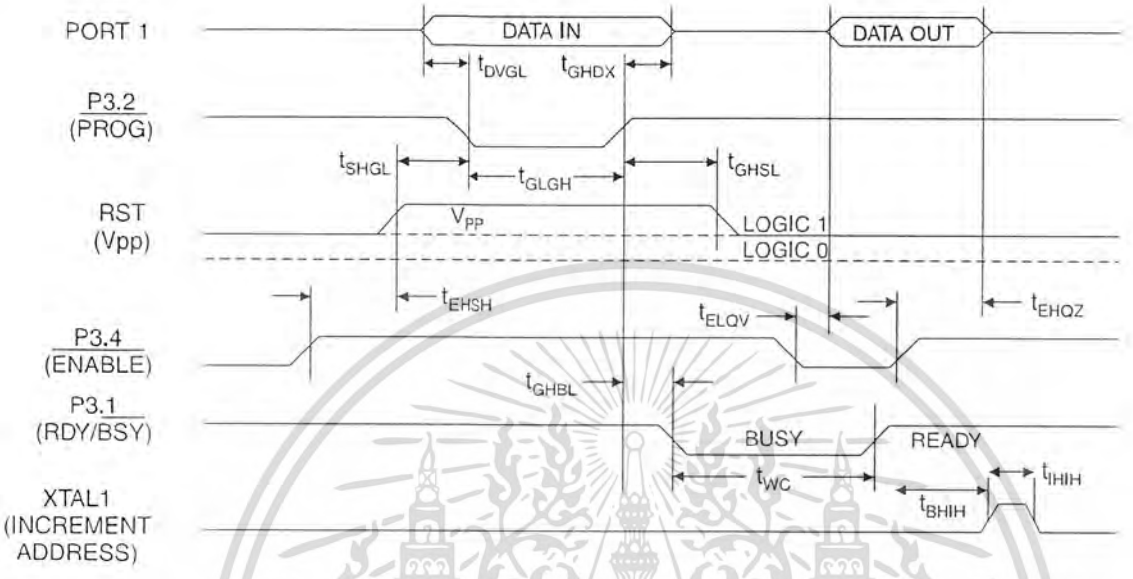
T_A = 21°C to 27°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		250	µA
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	1.0		µs
t_{GHDX}	Data Hold After $\overline{PROG}$	1.0		µs
t_{EHSH}	P3.4 ($\overline{ENABLE}$) High to V_{PP}	1.0		µs
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		1.0	µs
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	1.0	µs
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		50	ns
t_{WC}	Byte Write Cycle Time		2.0	ms
t_{BHIH}	RDY/$\overline{BSY}$\ to Increment Clock Delay	1.0		µs
t_{IHIL}	Increment Clock High	200		ns

Note: 1. Only used in 12-volt programming mode.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0 V to +7.0 V
Maximum Operating Voltage	6.6 V
DC Output Current	25.0 mA

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

D.C. Characteristics

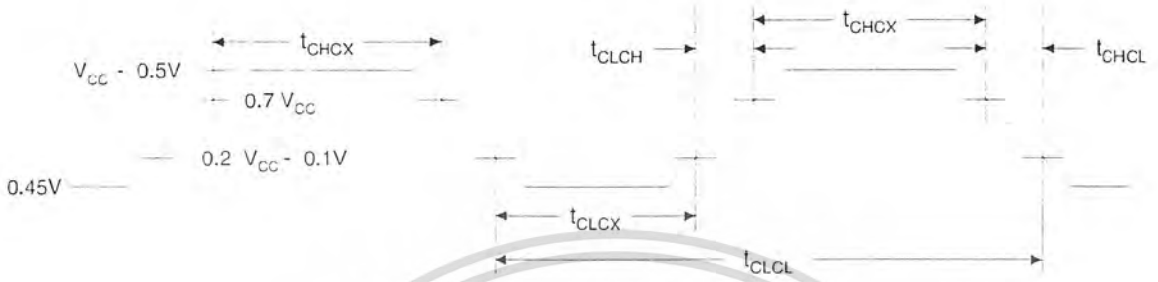
T_A = -40°C to 85°C, V_{CC} = 2.7 V to 6.0 V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage		-0.5	0.2 V_{CC}-0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5 V I_{OL} = 10 mA, V_{CC} = 2.7 V		0.5	V
V_{OH}	Output High Voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)	V_{IN} = 0.45 V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3)	V_{IN} = 2 V		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5 V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6 V/3 V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6 V/3 V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power Down Mode[2]	V_{CC} = 6 V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3 V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin:20 mA
Maximum total IOL for all output pins:80 mA
If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power Down is 2 V.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

External Clock Drive Waveforms



External Clock Drive

Symbol	Parameter	V_{CC} = 2.7 V to 6.0 V		V_{CC} = 4.0 V to 6.0 V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

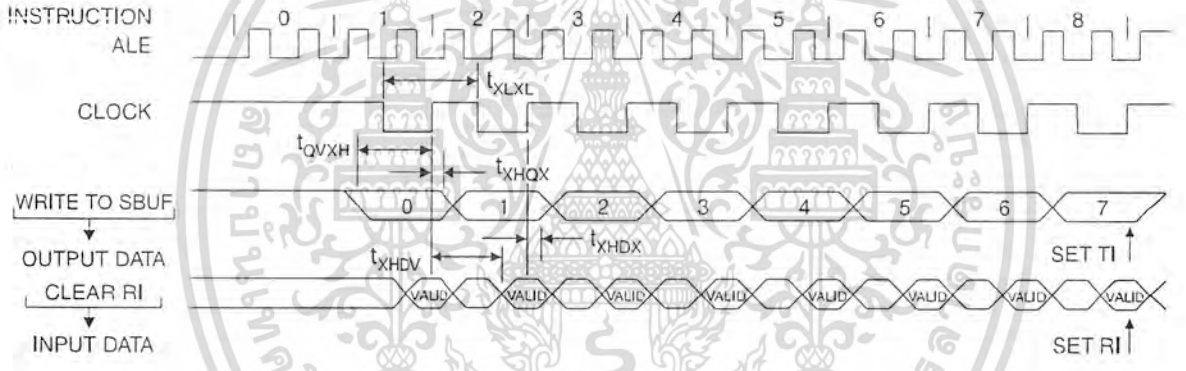
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Serial Port Timing: Shift Register Mode Test Conditions

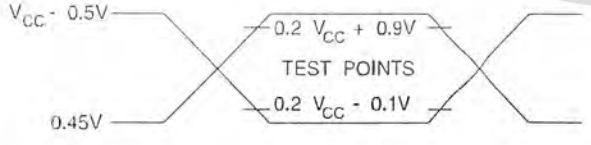
(V_{CC} = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-33		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		10tCLCL-133	ns

Shift Register Mode Timing Waveforms

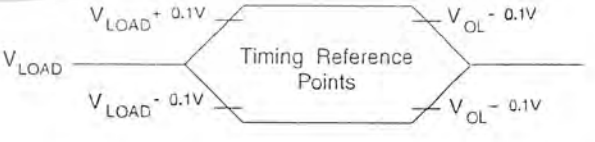


AC Testing Input/Output Waveforms[1]



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

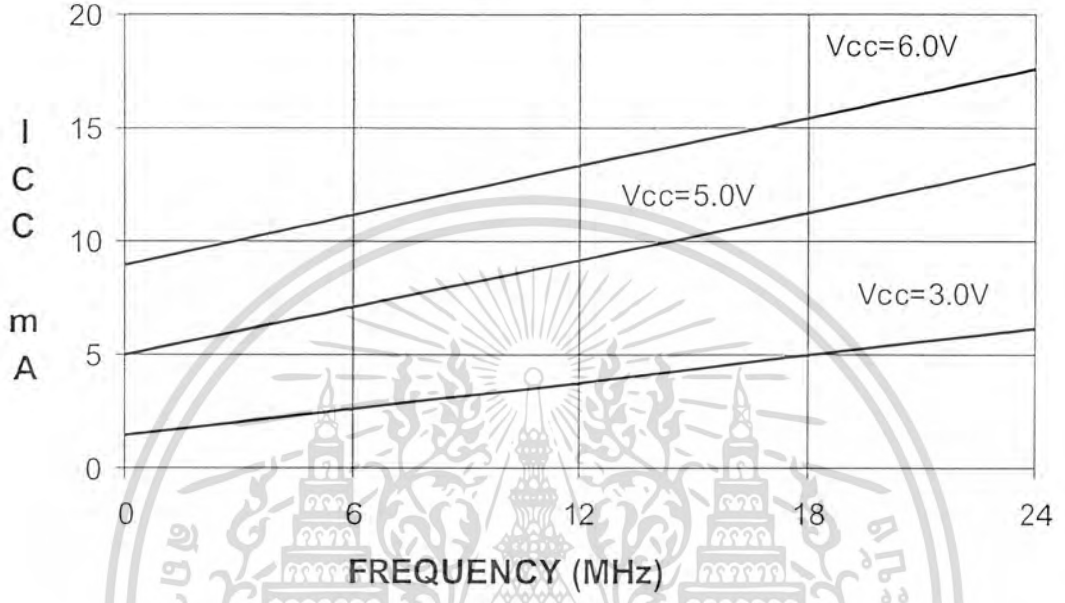
Float Waveforms[1]



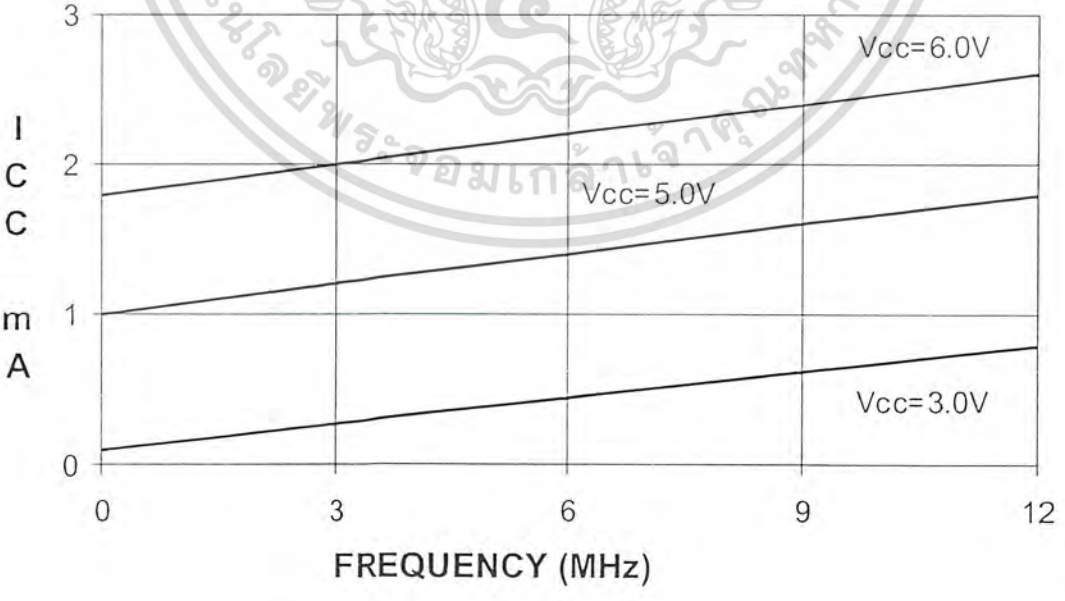
Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AT89C2051
TYPICAL ICC - ACTIVE (85°C)

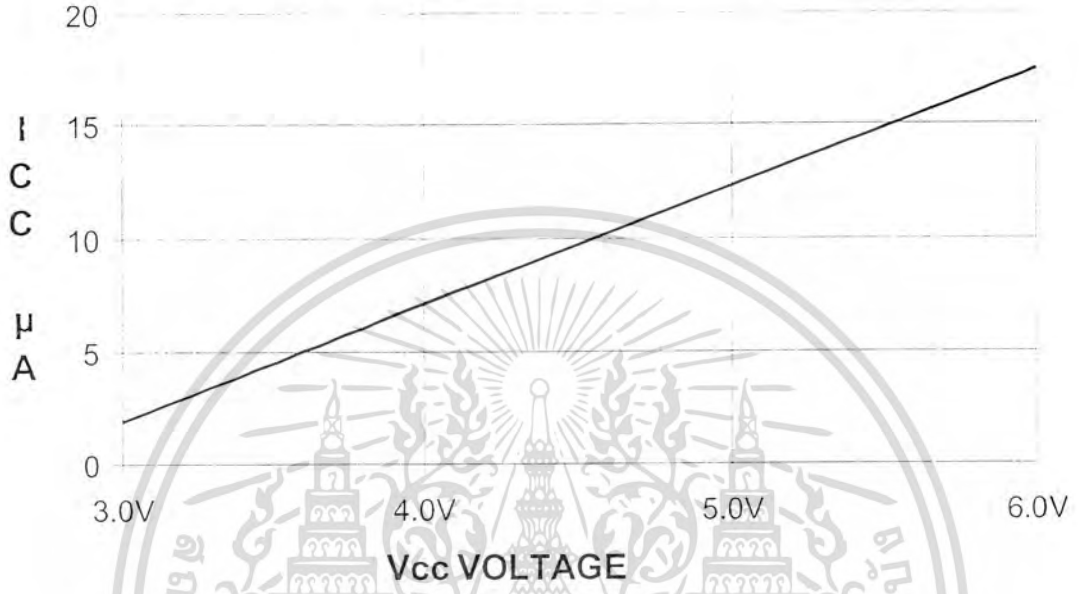


AT89C2051
TYPICAL ICC - IDLE (85°C)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TYPICAL ICC vs. VOLTAGE- POWER DOWN (85°C)



Note: 1. XTAL1 tied to GND for Icc (power down).
2. P.1.0 and P1.1 = Vcc or GND.
3. Lock bits programmed.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7 V to 6.0 V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
24	4.0 V to 6.0 V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300" Wide, Plastic Dual In-line Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

HARRIS SEMICONDUCTOR

ICL232

+5V Powered Dual RS-232 Transmitter/Receiver

December 1993

Features

- Meets All RS-232C Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/µs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges

Applications

- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
  - Dataloggers

Description

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

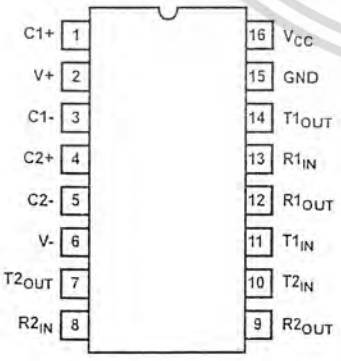
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

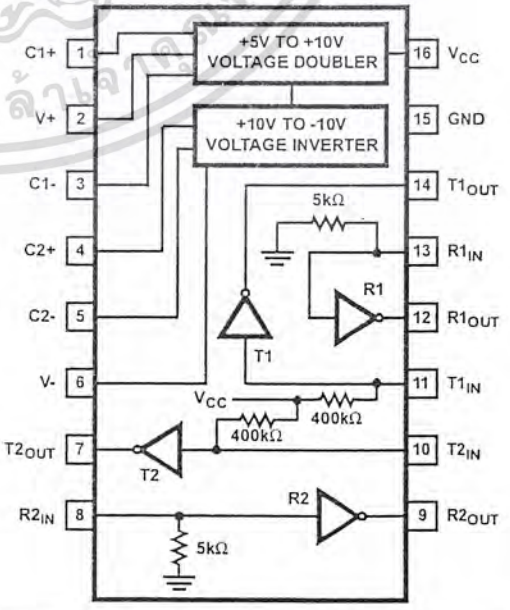
PART NUMBER	TEMPERATURE RANGE	PACKAGE
ICL232CPE	0°C to +70°C	16 Lead Plastic DIP
ICL232CJE	0°C to +70°C	16 Lead Ceramic DIP
ICL232CBE	0°C to +70°C	16 Lead SOIC (W)
ICL232IPE	-40°C to +85°C	16 Lead Plastic DIP
ICL232IJE	-40°C to +85°C	16 Lead Ceramic DIP
ICL232IBE	-40°C to +85°C	16 Lead SOIC (W)
ICL232MJE	-55°C to + 125°C	16 Lead Ceramic DIP

Pinouts

ICL232
(PDIP, CDIP, SOIC)
TOP VIEW



Functional Diagram



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper I.C. Handling Procedures.

Copyright © Harris Corporation 1993

File Number 3020.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Absolute Maximum Ratings

V_{CC} to Ground . . . . . (GND -0.3V) <V_{CC} < 6V
V+ to Ground . . . . . (V_{CC} -0.3V) <V+ < 12V
V- to Ground . . . . . -12V < V- < (GND +0.3V)
Input Voltages
  $T1_{IN}$, $T2_{IN}$ . . . . . (V- -0.3V) < V_{IN} < (V+ +0.3V)
  $R1_{IN}$, $R2_{IN}$ . . . . . ±30V
Output Voltages
  $T1_{OUT}$, $T2_{OUT}$ . . . . . (V- -0.3V) < V_{TXOUT} < (V+ +0.3V)
  $R1_{OUT}$, $R2_{OUT}$ . . . . . (GND -0.3V) < V_{RXOUT} < (V_{CC} +0.3V)
Short Circuit Duration
  $T1_{OUT}$, $T2_{OUT}$ . . . . . Continuous
  $R1_{OUT}$, $R2_{OUT}$ . . . . . Continuous
Storage Temperature Range . . . . . -65°C to +150°C
Lead Temperature (Soldering 10s) . . . . . +300°C

Thermal Information

Thermal Resistance	θ_{JA}	θ_{JC}
Ceramic DIP Package	80°C/W	24°C/W
Plastic DIP Package	100°C/W	-
SOIC Package	100°C/W	-

Maximum Power Dissipation 250mW
Operating Temperature Range
  ICL232C . . . . . 0°C to +70°C
  ICL232I . . . . . -40°C to +85°C
  ICL232M . . . . . -55°C to +125°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Specifications

Test Conditions: V_{CC} = +5V ±10%, T_A = Operating Temperature Range. Test Circuit as in Figure 8 Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	LIMITS			UNITS
		MIN	TYP	MAX	
Transmitter Output Voltage Swing, T_{OUT}	$T1_{OUT}$ and $T2_{OUT}$ loaded with 3kΩ to Ground	±5	±9	±10	V
Power Supply Current, I_{CC}	Outputs Unloaded, T_A = +25°C	-	5	10	mA
T_{IN}, Input Logic Low, V_{IL}		-	-	0.8	V
T_{IN}, Input Logic High, V_{IH}		2.0	-	-	V
Logic Pullup Current, I_P	$T1_{IN}$, $T2_{IN}$ = 0V	-	15	200	µA
RS-232 Input Voltage Range, V_{IN}		-30	-	+30	V
Receiver Input Impedance, R_{IN}	V_{IN} = ±3V	3.0	5.0	7.0	kΩ
Receiver Input Low Threshold, V_{IN} (H-L)	V_{CC} = 5.0V, T_A = +25°C	0.8	1.2	-	V
Receiver Input High Threshold, V_{IN} (L-H)	V_{CC} = 5.0V, T_A = +25°C	-	1.7	2.4	V
Receiver Input Hysteresis, V_{HYST}		0.2	0.5	1.0	V
TTL/CMOS Receiver Output Voltage Low, V_{OL}	I_{OUT} = 3.2mA	-	0.1	0.4	V
TTL/CMOS Receiver Output Voltage High, V_{OH}	I_{OUT} = -1.0mA	3.5	4.6	-	V
Propagation Delay, t_{PD}	RS-232 to TTL	-	0.5	-	µs
Instantaneous Slew Rate, SR	C_L = 10pF, R_L = 3kΩ, T_A = +25°C (Notes 1, 2)	-	-	30	V/µs
Transition Region Slew Rate, SR_T	R_L = 3kΩ, C_L = 2500pF Measured from +3V to -3V or -3V to +3V	-	3	-	V/µs
Output Resistance, R_{OUT}	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V	300	-	-	Ω
RS-232 Output Short Circuit Current, I_{SC}	$T1_{OUT}$ or $T2_{OUT}$ shorted to GND	-	±10	-	mA

NOTES:

1. Guaranteed by design.
2. See Figure 4 for definition.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Typical Performance Curves

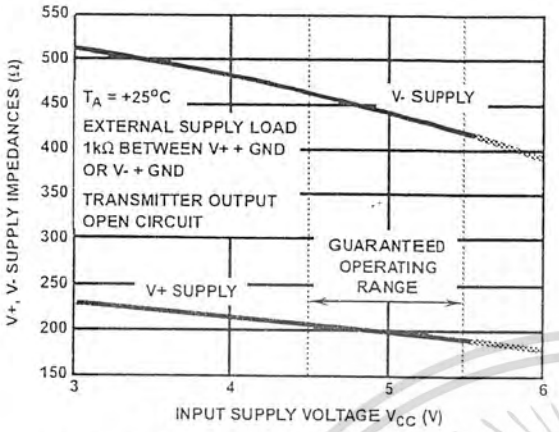


FIGURE 1. V+, V- OUTPUT IMPEDANCES vs V_{CC}

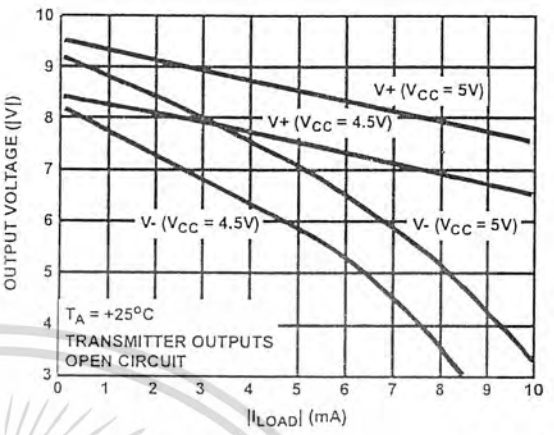


FIGURE 2. V+, V- OUTPUT VOLTAGES vs LOAD CURRENT

Pin Descriptions

PLASTIC DIP, CERAMIC DIP	SOIC	PIN NAME	DESCRIPTION
1	1	C1+	External capacitor "+" for internal voltage doubler.
2	2	V+	Internally generated +10V (typical) supply.
3	3	C1-	External capacitor "-" for internal voltage doubler.
4	4	C2+	External capacitor "+" internal voltage inverter.
5	5	C2-	External capacitor "-" internal voltage inverter.
6	6	V-	Internally generated -10V (typical) supply.
7	7	$T2_{OUT}$	RS-232 Transmitter 2 output ±10V (typical).
8	8	$R2_{IN}$	RS-232 Receiver 2 input, with internal 5K pulldown resistor to GND.
9	9	R2out	Receiver 2 TTL/CMOS output.
10	10	$T2_{IN}$	Transmitter 2 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
11	11	$T1_{IN}$	Transmitter 1 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
12	12	$R1_{OUT}$	Receiver 1 TTL/CMOS output.
13	13	$R1_{IN}$	RS-232 Receiver 1 input, with internal 5K pulldown resistor to GND.
14	14	$T1_{OUT}$	RS-232 Transmitter 1 output ±10V (typical).
15	15	GND	Supply Ground.
16	16	VCC	Positive Power Supply +5V ±10%

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Detailed Description

The ICL232 is a dual RS-232 transmitter/receiver powered by a single +5V power supply which meets all EIA RS232C specifications and features low power consumption. The functional diagram illustrates the major elements of the ICL232. The circuit is divided into three sections: a voltage doubler/inverter, dual transmitters, and dual receivers.

Voltage Converter

An equivalent circuit of the dual charge pump is illustrated in Figure 3.

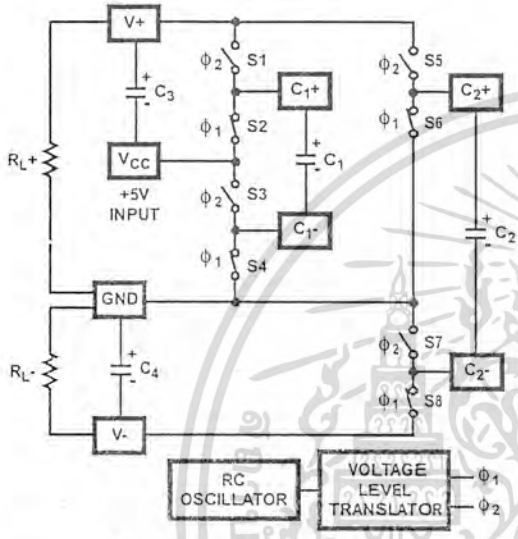
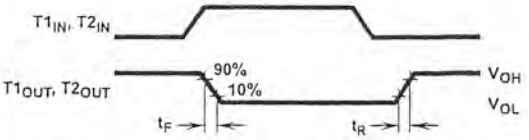


FIGURE 3. DUAL CHARGE PUMP

The voltage quadrupler contains two charge pumps which use two phases of an internally generated clock to generate +10V and -10V. The nominal clock frequency is 16kHz. During phase one of the clock, capacitor C1 is charged to V_{CC}. During phase two, the voltage on C1 is added to V_{CC}, producing a signal across C2 equal to twice V_{CC}. At the same time, C3 is also charged to $2V_{CC}$, and then during phase one, it is inverted with respect to ground to produce a signal across C4 equal to $-2V_{CC}$. The voltage converter accepts input voltages up to 5.5V. The output impedance of the doubler (V+) is approximately 200Ω, and the output impedance of the inverter (V-) is approximately 450Ω. Typical graphs are presented which show the voltage converters output vs input voltage and output voltages vs load characteristics. The test circuit (Figure 8) uses 1µF capacitors for C1-C4, however, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, and increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.



Instantaneous Slew Rate (SR) = $\frac{(0.8)(V_{OH} - V_{OL})}{t_R}$ or $\frac{(0.8)(V_{OL} - V_{OH})}{t_F}$

FIGURE 4. SLEW RATE DEFINITION

Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of V_{CC}, or 1.3V for V_{CC} = 5V. A logic 1 at the input results in a voltage of between -5V and V- at the output, and a logic 0 results in a voltage between +5V and (V+ -0.6V). Each transmitter input has an internal 400kΩ pullup resistor so any unused input can be left unconnected and its output remains in its low state. The output voltage swing meets the RS-232C specification of ±5V minimum with the worst case conditions of: both transmitters driving 3kΩ minimum load impedance, V_{CC} = 4.5V, and maximum allowable operating temperature. The transmitters have an internally limited output slew rate which is less than 30V/µs. The outputs are short circuit protected and can be shorted to ground indefinitely. The powered down output impedance is a minimum of 300Ω with ±2V applied to the outputs and V_{CC} = 0V.

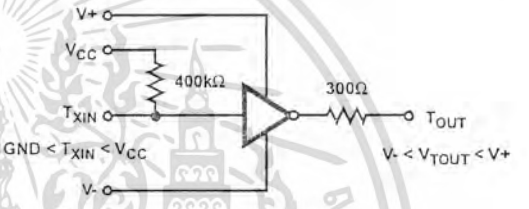


FIGURE 5. TRANSMITTER

Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even it the power is off (V_{CC} = 0V). The receivers have a typical input threshold of 1.3V which is within the ±3V limits, known as the transition region, of the RS-232 specification. The receiver output is 0V to V_{CC}. The output will be low whenever the input is greater than 2.4V and high whenever the input is floating or driven between +0.8V and -30V. The receivers feature 0.5V hysteresis to improve noise rejection.

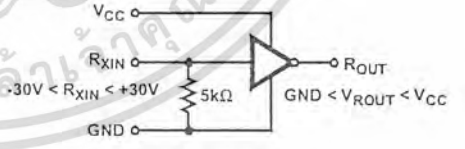
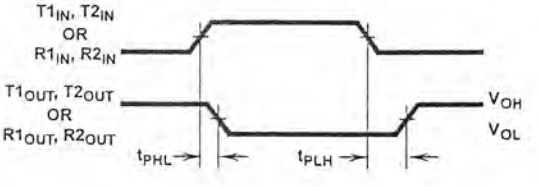


FIGURE 6. RECEIVER



Average Propagation Delay = $\frac{t_{PHL} + t_{PLH}}{2}$

FIGURE 7. PROPAGATION DELAY DEFINITION

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Test Circuits

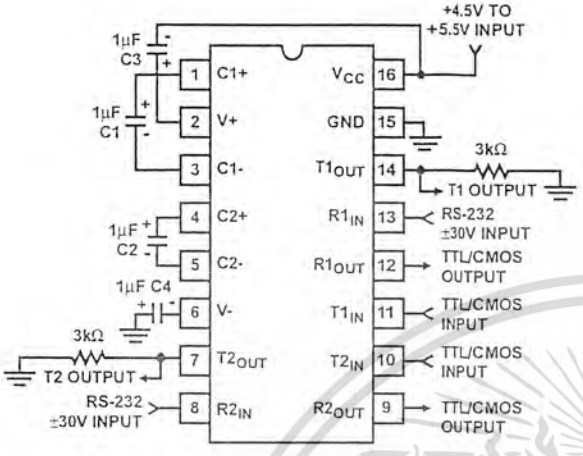


FIGURE 8. GENERAL TEST CIRCUIT

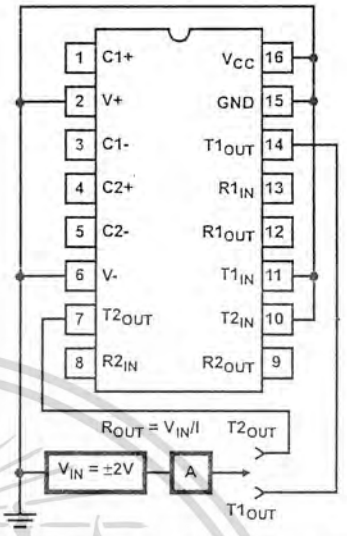


FIGURE 9. POWER-OFF SOURCE RESISTANCE CONFIGURATION

Applications

The ICL232 may be used for all RS-232 data terminal and communication links. It is particularly useful in applications where ±12V power supplies are not available for conventional RS-232 interface circuits. The applications presented represent typical interface configurations.

A simple duplex RS-232 port with CTS/RTS handshaking is illustrated in Figure 10. Fixed output signals such as DTR (data terminal ready) and DSRS (data signaling rate select) is generated by driving them through a 5kΩ resistor connected to V+.

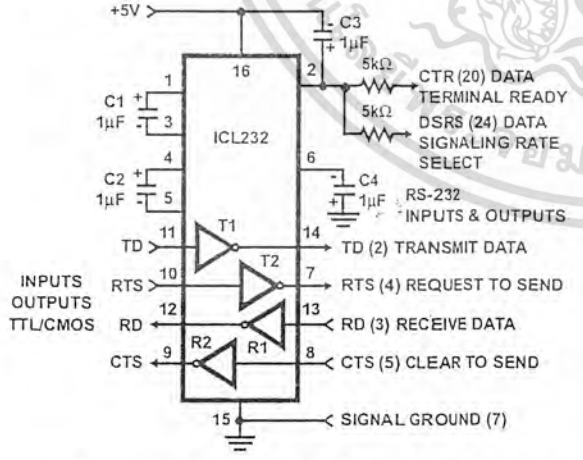


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

In applications requiring four RS-232 inputs and outputs (Figure 11), note that each circuit requires two charge pump capacitors (C1 and C2) but can share common reservoir capacitors (C3 and C4). The benefit of sharing common reservoir capacitors is the elimination of two capacitors and the reduction of the charge pump source impedance which effectively increases the output swing of the transmitters.

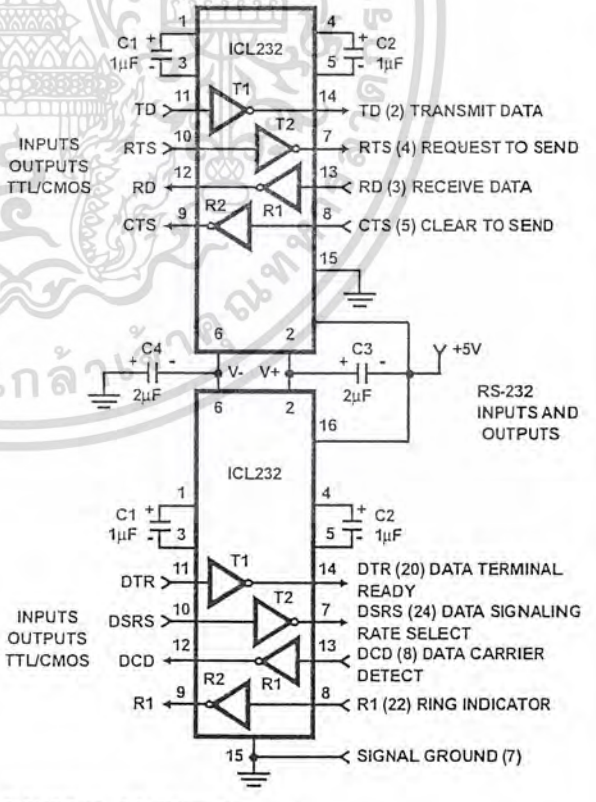


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

1. สุเจตน์ จันทรังษ์ , "MCS-51" มหาวิทยาลัยมหานคร ไมโครคอนโทรลเลอร์ชิพเดี่ยว 8051

2. ฉันทวุฒิ พืชผล , พิชิต สันติกุลนนท์ , คู่มือเรียน Visual Basic6 , บริษัทPROVISION จำกัด

3.กฤษดา ใจเย็น,อรรถพล บุญญโกคา, ชัยวัตร ลิ้มพรกิจวิลัย,เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม

4. วันสุระ ศรีไสดี , ไมโครคอลโทรลเลอร์ภาคปฏิบัติ , สำนักพิมพ์ดวงกมล

5. ฟิวเจอร์อิเลคทรอนิคส์ ฉบับที่ 8 , ประจำเดือน มกราคม – มีนาคม 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้