

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

แผงสวิตช์เพื่อทดสอบในโหมด พารามตริก ของแฟลชเมมโมรี
Parametric test mode Switch-Box for Flash Memory.



เลขหมู่.....
เลขทะเบียน..... 73164
วัน,เดือน,ปี..... 6 ก.ค. 2550

b. A1 486304
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แผงสวิตช์เพื่อทดสอบในโหมด พารามตริก ของแฟลชเมมโมรี

Parametric test mode switch-box for flash memory.

โดย

นายโอภาส เต็มมัน 45010996



ปริญญาานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญาโทปีการศึกษา 2548

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แผงสวิตช์เพื่อทดสอบในโหมด พารามตริก ในห้องปฏิบัติการ ดีไวซ์ แอนนาไลซิส

ผู้จัดทำ

นายโอภาส เสือมัน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แผนผังสวิตช์เพื่อทดสอบในโหมดของแพลตฟอร์มโมรี

นาย โสภาส เสือมัน รหัส 45010996

อาจารย์ชินภัทร นันทจิวงกรชัย

ปีการศึกษา 2548

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอ แผนผังสวิตช์ที่ควบคุมโดยไมโครคอนโทรลเลอร์ เพื่อเพิ่มความสามารถของ เบนช์เทส (Bench Test) ในการทดสอบชิ้นงาน ที่ใช้อยู่ในห้องปฏิบัติการ Device Analysis ของ บริษัท สแปนชั่น (ประเทศไทย) จำกัด แผนผังสวิตช์นี้ประกอบด้วยสวิตช์ 3 ทาง จำนวนตั้งแต่ 48 สวิตช์ขึ้นไปปัจจุบัน แผนผังสวิตช์เหล่านี้ใช้เลือกสัญญาณที่จะ โบนแอสให้แก่วางต่างๆของชิ้นงานที่เราต้องการทดสอบ ด้วยระบบแมนนวล ในทางทฤษฎีจะต้องสับสวิตช์เหล่านี้พร้อมกันแต่ในทางปฏิบัติเป็นการยากที่จะสับสวิตช์ได้อย่างถูกต้องและไม่ สามารถสับสวิตช์ทั้งหมดในเวลาเดียวกันได้ เพื่อที่จะแก้ปัญหานี้และเพิ่มความสามารถ รวมถึงป้องกันการ สับสวิตช์ที่ผิดพลาดจึง ได้มีการศึกษาและพัฒนา แผนผังสวิตช์ที่ควบคุมโดยคอมพิวเตอร์ โดยใช้โปรแกรม Visual Basic เพื่อพัฒนาโปรแกรมที่ใช้รับคำสั่งต่างๆ โดยส่งข้อมูลผ่านทางพอร์ตอนุกรม มาตรฐาน อาร์เอส 232 ตัว ไมโครคอนโทรลเลอร์จะนำข้อมูลที่ ได้มาวิเคราะห์ เพื่อควบคุมสวิตช์ตามคำสั่งที่ได้รับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาติให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Parametric test mode switch-box for flash memory.

Mr.Opart Sauemun ID.45010996

Mr.Chinnaphat Nuntajivakhonchai Advisor.

Educational Year 2005

Abstract

This project proposes switch-box controlled by Microcontroller to increase capability of the Bench Test for testing device in Device Analysis (DA) Laboratory of SPANSION (Thailand) Limited. The switch-box comprises of three-way switch more and equal to 48 switches. Nowadays, these switches are used for selecting signal(s) to bias the device by manual adjustment. Ideally, they must be adjusted at the same time. However, in practical adjusting the switches to apply the signal(s) to the device by manual adjustment is quite difficult and cannot do at the same time. For solving this problem, improving capability, and preventing error, this switch-box has been studied and developed. It has been controlled by Personal Computer (PC) using Visual Basic program as developing program for User Interface (UI) and data transmission through the RS-232 serial port. Microcontroller of the switch-box will process all transmitted data and control all switches to apply the signal(s) to the device according to the processed data.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงการชิ้นนี้เกิดขึ้นได้เนื่องจากบริษัท สแปนชัน(ประเทศไทย) จำกัด ได้ให้โอกาสให้นักศึกษาได้มีส่วนร่วมในการพัฒนาสิ่งประดิษฐ์ใหม่ๆภายใต้การดูแลของอาจารย์ชินภัทร นันทจิวงกรชัย และ คร.ปรอง กองทรัพย์โต และมีพี่ๆและเพื่อนๆ ที่ช่วยให้คำปรึกษาแนะนำต่างๆทำให้โครงการชิ้นนี้สำเร็จลงได้ ข้าพเจ้าขอขอบคุณในบุคคลเหล่านี้อย่างสุดซึ่งที่ช่วยเอื้อเพื่อให้คำแนะนำที่เป็นประโยชน์ต่อการทำโครงการชิ้นนี้ ช่วยแก้ปัญหาที่เกิดขึ้นระหว่างการทำโครงการนี้ ข้าพเจ้าขอสดุดีในความเอื้อเฟื้อและความมีน้ำใจของทุกๆท่าน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ไมโครคอนโทรลเลอร์	4
2.1 ความหมายของไมโครคอนโทรลเลอร์	4
2.2 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช	5
2.3 การจัดการหน่วยความจำของไมโครคอนโทรลเลอร์ MCS - 51แบบแฟลช	17
2.4 ไทมเมอร์/ เกล็นต์เตอร์	30
2.5 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์	50
2.6 การอินเทอร์รัพต์	62
บทที่ 3 การสื่อสารข้อมูลแบบอนุกรม	72
3.1 หลักการและโครงสร้างของการสื่อสารแบบอนุกรม	72
3.2 ข้อดีของการสื่อสารแบบอนุกรม	72
3.3 รูปแบบของการรับส่งข้อมูลแบบอนุกรม	73
3.4 มาตรฐานของการสื่อสารข้อมูลแบบอนุกรม	76
3.5 หัวต่อที่ใช้มาตรฐาน RS- 232C	80
บทที่ 4 การใช้งาน Visual Basic	90
4.1 การใช้งานคอนโทรลในการสร้างอินเตอร์เฟซ	90
4.2 รูปแบบการปรากฏของคอนโทรลแต่ละชนิดบนฟอร์ม	90
4.3 แนวทางในการออกแบบอินเตอร์เฟซแรก	91
4.4 การนำคอนโทรลมาใช้งาน	95
4.5 พื้นฐานการเขียนโค้ด	96
4.6 การใช้งาน Editor	96
4.7 โปรซีเจอร์ประจำคอนโทรล	97
4.8 โค้ดชุดแรก	100
4.9 หมายเหตุในการเขียนโค้ด (Comment)	102
4.10 คุณสมบัติและเมธอดของคอนโทรล	102
4.11 ความสามารถพิเศษของ editor	103

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เรื่อง	หน้า
4.12 คุณสมบัติประจำตัวของคอนโทรล	105
4.13 การจัดการ โปรเจกต์	107
4.14 การทดสอบ (Run) โปรเจกต์	112
4.15 การกำหนดฟอร์มเริ่มต้น	112
4.16 วิธีการกำหนดให้ Editor ของ VB สามารถแสดงผลภาษาไทยได้	114
บทที่ 5 การออกแบบและการทำงานของวงจร	116
5.1 ส่วนของสวิทช์	116
5.2 ส่วนที่ใช้ในการควบคุมสวิทช์	117
5.2.1 หลักการทำงานของส่วนควบคุมสวิทช์	117
5.2.2 วงจรและการทำงานของวงจร	118
5.3 ส่วนที่เป็นการติดต่อระหว่างผู้ใช้งานกับระบบ(User Interface)	130
5.3.1 หลักการทำงานของโปรแกรม	130
5.3.2 ลำดับการทำงานของโปรแกรมที่ใช้ใน Visual Basic	133
บทที่ 6 การทดลองและผลการทดลอง	137
6.1 โปรแกรมที่เขียนจากโปรแกรม Visual Basic	137
6.1.1 ทดสอบการรับค่าจำนวนขาของตัวชิ้นงาน	137
6.1.2 ทดสอบการเช็คเมื่อผู้ทดสอบไม่ได้ใส่ค่าใดๆ	141
6.1.3 ทดสอบการเช็คเมื่อใส่ข้อความที่ไม่ใช่ตัวเลข	142
6.2 ไมโครคอนโทรลเลอร์	143
6.2.1 ทดสอบการทำงานในโปรแกรม Proteus	143
6.2.2 การทดลองทางเครื่องต้นแบบ	145
บทที่ 7 บทสรุป	150

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

หมายเลขรูป	หน้า
รูปที่ 1.1 เครื่อง Tester (ซ้าย)และ Handler (ขวา)	2
รูปที่ 1.2 ลักษณะภายนอกของ Curve Tracer model 370Aและแผงสวิทช์ที่ใช้ในการเลือกสัญญาณให้แก่ชิ้นงาน	3
รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครโปรเซสเซอร์	4
รูปที่ 2.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์	5
รูปที่ 2.3 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์MCS-51 แบบแฟลชในอนุกรม AT89Cxx	5
รูปที่ 2.4 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์MCS-51 แบบแฟลชในอนุกรม AT89sxx	6
รูปที่ 2.5 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x	6
รูปที่ 2.6 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51	7
รูปที่ 2.7 วงจรภายในของพอร์ต 0	10
รูปที่ 2.8 วงจรภายในของพอร์ต 1	11
รูปที่ 2.9 วงจร พูลอัปภายในพอร์ตของไมโครคอนโทรลเลอร์	12
รูปที่ 2.10 วงจรภายในของพอร์ต 2	12
รูปที่ 2.11 วงจรภายในของพอร์ต 3	13
รูปที่ 2.12 ไชเกิตการทำงานของไมโครคอนโทรลเลอร์	15
รูปที่ 2.13 ไคอะแกรมเวลาแสดงการติดต่อและเข้าถึงหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์	17
รูปที่ 2.14 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ AT89C51	18
รูปที่ 2.15 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ AT89C52	18
รูปที่ 2.16 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์	20
รูปที่ 2.17 การเชื่อมต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์MCS-51 แบบแฟลช	21
รูปที่ 2.18 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์เบอร์ AT89C51	21

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

หมายเลขรูป	หน้า
รูปที่ 4.15 แสดงโปรซีเจอร์เหตุการณ์ LostFocus ของคอนโทรล CommandButton ที่ถูกเพิ่มเข้ามา	99
รูปที่ 4.16 แสดงโปรซีเจอร์เหตุการณ์คลิกของคอนโทรล CommandButton 2 ตัว	100
รูปที่ 4.17 แสดงอินเตอร์เฟซเพื่อสั่งให้คอนโทรล TextBox แสดงข้อความ hello world	101
รูปที่ 4.18 แสดง ToolTip ของ editor ขณะเขียนโค้ด	103
รูปที่ 4.19 แสดงไวยากรณ์ประจำฟังก์ชันนั้นๆ	104
รูปที่ 4.20 แสดงข้อความช่วยเหลือ เมื่อคุณพิมพ์ผิดไวยากรณ์	104
รูปที่ 4.21 แสดงไดอะล็อกบ็อกซ์ Open Project	107
รูปที่ 4.22 แสดงไดอะล็อกบ็อกซ์ Add Project	108
รูปที่ 4.23 แสดงการเลือกโปรเจกต์ที่คุณต้องการถอดออกจากสภาพแวดล้อม	109
รูปที่ 4.24 แสดงการเพิ่ม Form เข้ามาในโปรเจกต์	109
รูปที่ 4.25 แสดง Form ที่ถูกเพิ่มเข้ามาในโปรเจกต์	111
รูปที่ 2.26 แสดงไดอะล็อกบ็อกซ์เซฟโปรเจกต์	111
รูปที่ 4.27 แสดง Pop-up เมนู เมื่อคุณคลิกขวา	112
รูปที่ 4.28 แสดงไดอะล็อกบ็อกซ์ Project Properties	113
รูปที่ 4.29 แสดงการกำหนดให้ฟอร์ม2 เป็นฟอร์มเริ่มต้น	113
รูปที่ 4.30 เลือกคำสั่ง Options...	114
รูปที่ 4.31 ไดอะล็อกบ็อกซ์ Options	114
รูปที่ 4.32 กำหนดให้ editor ใช้ฟอนต์ MS Sans Serif ขนาด 10	115
รูปที่ 4.33 แสดงรูปแบบของฟอนต์ MS Sans Serif ขนาด 10 ใน editor	115
รูปที่ 5.1 แสดงการต่อรีเลย์เพื่อทำเป็นสวิตช์สามทาง	117
รูปที่ 5.2 แสดงวงจรที่ใช้ กับชิ้นงาน 8 พิน	119
รูปที่ 5.3 แสดงการต่อวงจรในส่วนของไมโครคอนโทรลเลอร์	120
รูปที่ 5.4 แสดงวงจรในส่วนการเอ็นเอเบิลชิปรีจิสเตอร์	121
รูปที่ 5.5 แสดงส่วนของวงจรที่ทำหน้าที่เป็นวงจรสำหรับวงจรจัดเรียงข้อมูลและขับสวิตช์	122
รูปที่ 5.6 วงจรในส่วนกับสวิตช์	122

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

หมายเลขรูป	หน้า
รูปที่ 5.7 แสดงวงจรในส่วนของการแสดงผล	123
รูปที่ 5.8 โฟลวชาร์ตการทำงานของ ไมโครคอนโทรลเลอร์	124
รูปที่ 5.9 โฟลวชาร์ตโปรแกรมย่อยส่งค่าเอาต์พุต	125
รูปที่ 5.10 โฟลวชาร์ตโปรแกรมย่อยส่งค่ากลับไปให้คอมพิวเตอร์	126
รูปที่ 5.11 โฟลวชาร์ตโปรแกรมย่อยบริการอินเตอร์รัปต์รับ-ส่งข้อมูล	127
รูปที่ 5.12 โฟลวชาร์ตโปรแกรมย่อยนำเฟิร์มเอนเอเบิล ชิพ รีจิสเตอร์	128
รูปที่ 5.13 โฟลวชาร์ตโปรแกรมย่อยเขียนเอเบิลชิปรีจิสเตอร์	129
รูปที่ 5.14 โฟลวชาร์ตโปรแกรมย่อยเคลียร์ข้อมูล	130
รูปที่ 5.15 แสดงหน้าจอของโปรแกรมเพื่อรับจำนวนขาของชิ้นงาน	131
รูปที่ 5.16 แสดงหน้าจอของโปรแกรม Visual Basic เมื่อเข้าสู่สถานะรอรับค่าจากผู้ใช้งาน	132
รูปที่ 5.17 แสดงตัวอย่างผลการเปรียบเทียบค่าที่ส่งไปให้ไมโครคอนโทรลเลอร์กับค่าที่ได้รับกลับมา	132
รูปที่ 5.18 โฟลวชาร์ตการทำงานของโปรแกรม Visual Basic	133
รูปที่ 5.19 โฟลวชาร์ตโปรแกรมย่อยรับค่าจำนวนพิน	134
รูปที่ 5.20 โปรแกรมย่อยบริการอินเตอร์รัปต์	135
รูปที่ 5.21 โฟลวชาร์ตโปรแกรมย่อยส่งข้อมูล	136
รูปที่ 6.1 แสดงหน้าจอรับจำนวนขาของอุปกรณ์	137
รูปที่ 6.2 แสดงการใส่จำนวนขาของชิ้นงาน	137
รูปที่ 6.3 แสดงผลของโปรแกรมให้ยืนยันจำนวนขา	137
รูปที่ 6.4 แสดงหน้าต่างของโปรแกรมเพื่อเลือกพอร์ตสำหรับ รับ ส่งข้อมูล	138
รูปที่ 6.5 แสดงการเกิดการผิดพลาดเนื่องจากไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ไม่สามารถติดต่อกันได้	138
รูปที่ 6.6 แสดงหน้าต่างรอคำสั่ง	139
รูปที่ 6.7 เมื่อทำการเลือกสัญญาณ ปุ่มที่เลือกจะ Active	139
รูปที่ 6.8 แสดงหน้าต่างเมื่อข้อมูลที่รับมาตรงกัน	140
รูปที่ 6.9 แสดงหน้าต่างเมื่อข้อมูลที่รับมาไม่ตรงกัน	140

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

หมายเลขรูป	หน้า
รูปที่ 6.12 หน้าต่างเมื่อป้อนข้อมูล Tawan	142
รูปที่ 6.13 หน้าต่างของโปรแกรมเมื่อป้อนข้อมูล 123T	142
รูปที่ 6.14 แสดงหน้าต่างเมื่อใส่จำนวนเขาไม่ถูกต้อง	142
รูปที่ 6.15 แสดงการทดลองส่งค่า “0” ไปยังไมโครคอนโทรลเลอร์ เพื่อทดสอบสาย สัญญาณว่ามีปัญหาหรือไม่	143
รูปที่ 6.16 เมื่อส่งข้อมูลไปยัง MCS-51	143
รูปที่ 6.17 แสดงผลเมื่อทำการส่ง “U” ไปยัง MCS-51	144
รูปที่ 6.18 แสดงผลเมื่อทำการส่งข้อมูลไปยัง MCS-51 อีกครั้ง	144
รูปที่ 6.19 เมื่อคลิกปุ่ม F ทั้ง 4 ฟันแล้วคลิก OK	145
รูปที่ 6.20 ผลที่ออกที่เครื่องต้นแบบเมื่อป้อน Float ทั้ง 4 ฟัน	145
รูปที่ 6.21 เมื่อคลิก H ทั้ง 4 ฟันแล้วคลิก OK	146
รูปที่ 6.22 ผลที่ออกที่เครื่องต้นแบบเมื่อป้อน High ทั้ง 4 ฟัน	146
รูปที่ 6.23 เมื่อคลิก L ทั้ง 4 ฟันแล้วคลิก OK	147
รูปที่ 6.24 ผลที่ออกที่เครื่องต้นแบบเมื่อป้อน Low ทั้ง 4 ฟัน	147
รูปที่ 6.25 เมื่อ 2 ฟันแรกคลิก H และ 1 ใน 2 ฟันหลัง	148
รูปที่ 6.26 ผลเมื่อ ป้อน High 2 ฟันแรก และ Low 2 ฟันหลัง	148
รูปที่ 6.27 เมื่อคลิก L,F,I และ H ตามลำดับ	149
รูปที่ 6.27 ผล เมื่อป้อน แต่ละฟันเป็น Low,Float,Low และ High ตามลำดับ	149

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หมายเลขตาราง	หน้า
ตารางที่ 2.1 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51	
แบบแฟลชของ Atmel	10
ตารางที่ 2.2 การเลือกแบงก์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงก์ R0-R7	27
ตารางที่ 2.3 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 0 ทำงานเป็นไทมเมอร์	38
ตารางที่ 2.4 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 0 ทำงานเป็นเคาน์เตอร์	38
ตารางที่ 2.5 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 1 ทำงานเป็นไทมเมอร์	39
ตารางที่ 2.6 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 1 ทำงานเป็นเคาน์เตอร์	39
ตารางที่ 2.7 การเลือกโหมดทำงานของไทมเมอร์/เคาน์เตอร์ 2	43
ตารางที่ 2.8 แสดงข้อมูลของรีจิสเตอร์ T2CON เพื่อกำหนดให้ไทมเมอร์/เคาน์เตอร์ 2 ทำงานเป็นไทมเมอร์	46
ตารางที่ 2.9 แสดงข้อมูลของรีจิสเตอร์ T2CON เพื่อกำหนดให้ไทมเมอร์/เคาน์เตอร์ 2 ทำงานเป็นเคาน์เตอร์	46
ตารางที่ 2.10 การเลือกเวลาให้แก่วอตซ์ได้ออก	48
ตารางที่ 2.11 การเลือกอัตราบอดของวงจรถ่ายโอนข้อมูลภายใน ไมโครคอนโทรลเลอร์ MCS-51	62
ตารางที่ 2.12 บิตและหน้าที่ต่างๆของรีจิสเตอร์ เอนิเมิลการอินเตอร์รัปต์	65
ตารางที่ 2.13 ตารางแสดงการจำลำดับความสำคัญลัดโนมิคของไมโครคอนโทรลเลอร์	66
ตารางที่ 2.14 บิตและหน้าที่ต่างๆของรีจิสเตอร์รีจิสเตอร์จัดลำดับความสำคัญการ ตอบสนองการอินเตอร์รัปต์	66
ตารางที่ 2.15 แฟลคที่จะทำงานเมื่อถูกอินเตอร์รัปต์	67
ตารางที่ 2.16 อินเตอร์รัปต์เวกเตอร์ของอินเตอร์รัปต์ต่างๆ	69
ตารางที่ 3.1 การเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA	79
ตารางที่ 3.2 ความสัมพันธ์ระหว่างสถานะคู่ต่างของสัญญาณ	82
ตารางที่ 3.3 แสดงขาต่างๆของหัวต่อตัวผู้ชนิด 25 ขา	84
ตารางที่ 3.3 แสดงขาต่างๆของหัวต่อตัวผู้ชนิด 9 ขา	85
ตารางที่ 4.1 แสดงคุณสมบัติประจำตัวแต่ละคอนโทรล	106
ตารางที่ 4.2 ความหมายของไอคอนในการสร้างฟอร์มชนิดต่างๆ	110

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความเป็นมาของโครงการ

โครงการนี้จัดทำขึ้นเนื่องจาก บ.สเปนชัน ประเทศไทย จำกัด ได้เล็งเห็นถึงความสำคัญของการพัฒนาระบบการทดสอบชิ้นงานที่ทางบริษัทผลิตขึ้น เพื่อให้สามารถทดสอบชิ้นงานได้รวดเร็วขึ้น รวมทั้งมีความเที่ยงตรงในการทดสอบและป้องกันความผิดพลาดที่อาจเกิดขึ้นเนื่องจากผู้ปฏิบัติงานในการป้อนสัญญาณไฟฟ้าให้แก่ตัวชิ้นงาน อีกทั้งเป็นการสนับสนุนนักศึกษาให้ฝึกการออกแบบและพัฒนาสิ่งประดิษฐ์ ให้สามารถนำไปใช้ได้จริงในงานอุตสาหกรรม ซึ่งเป็นสิ่งที่กำลังขาดแคลนในระบบอุตสาหกรรมของไทย เพราะสิ่งประดิษฐ์ที่นักศึกษาของไทยประดิษฐ์คิดค้นขึ้นส่วนใหญ่มักจะไม่ได้รับการพัฒนาเพื่อให้สามารถนำไปใช้ในอุตสาหกรรมตามที่ได้ออกแบบมา

1.2 ขอบเขตของโครงการ

โครงการนี้จัดทำขึ้นเพื่อพัฒนาแผงสวิทช์ที่ควบคุมการทำงานโดยไมโครคอนโทรลเลอร์เพื่อใช้ร่วมกับเครื่องเบนซ์เทส ในการทดสอบชิ้นงานแบบพาราเมตริก ที่ใช้ในห้องปฏิบัติการ Device Analysis ของบริษัท สเปนชัน ประเทศไทย จำกัด

1.3 เป้าหมายของโครงการ

สร้างต้นแบบแผงสวิทช์ที่ควบคุมโดยไมโครคอนโทรลเลอร์ สั่งงานผ่านคอมพิวเตอร์

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาและทำความเข้าใจลำดับการทำงานทางไฟฟ้าและหลักการทำงานของเครื่องเบนซ์เทส

1.4.2 ศึกษาและทำความเข้าใจหลักการทำงานของไมโครคอนโทรลเลอร์ พร้อมทั้งออกแบบแผง สวิทช์ที่ควบคุมโดยไมโครคอนโทรลเลอร์

1.4.3 ศึกษาและทำความเข้าใจหลักการใช้โปรแกรม Visual Basic เพื่อพัฒนาโปรแกรมในการติดต่อระหว่างผู้ปฏิบัติงานกับไมโครคอนโทรลเลอร์

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงการ

เมื่อโครงการนี้เสร็จสมบูรณ์คาดว่าจะช่วยลดเวลาในการเซตระบบและลดความผิดพลาดโดยผู้ปฏิบัติงาน

1.6 การทดสอบคุณสมบัติทางไฟฟ้า(Electrical Test)

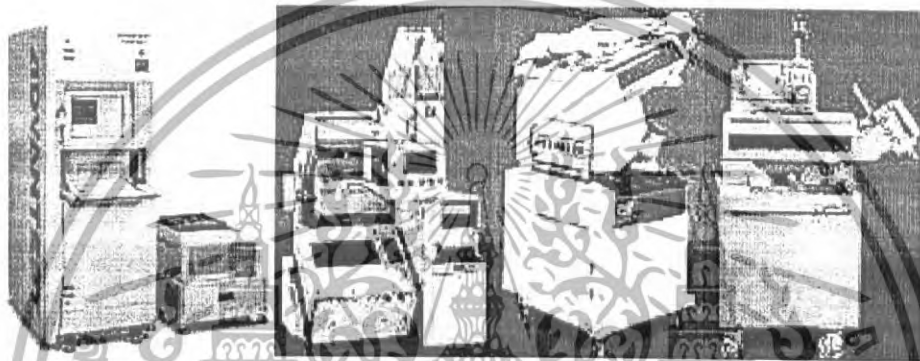
การทดสอบคุณสมบัติทางไฟฟ้าคือการตรวจสอบและคัดแยกอุปกรณ์ที่เสียคุณสมบัติทางไฟฟ้า

(Electrical failures) ออกจากอุปกรณ์อื่นๆ อุปกรณ์ที่เสียคือการที่อุปกรณ์ตัวนั้นแสดงคุณสมบัติไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นไปตามที่กำหนด หรือกล่าวได้ว่า การทดสอบคุณสมบัติทางไฟฟ้าเป็นการกระตุ้นอุปกรณ์ด้วย Device Under Test (DUT) แล้ววัดการตอบสนองออกมาโดยอยู่ในช่วงที่กำหนด

ในระบบการผลิต ระบบทดสอบจะประกอบด้วย เครื่องทดสอบ(Tester) กับ เครื่องลำเลียง (Handler) โดยเครื่องลำเลียงจะทำหน้าที่ส่งยูนิตมาที่บริเวณทดสอบ(Test Side) และจะตำแหน่งของยูนิตให้เหมาะสมกับการทดสอบ เมื่อเครื่องทดสอบทำการทดสอบยูนิตเรียบร้อยแล้วจะทำการบอกให้เครื่องลำเลียงทำการแยกยูนิตที่ติดกับที่เสียออกจากกัน



รูปที่ 1.1 เครื่อง Tester (ซ้าย) และ Handler (ขวา)

ในกระบวนการทดสอบจะมีโปรแกรมสำหรับทดสอบ โดยเฉพาะซึ่งเขียนจากภาษาชั้นสูง เช่น C++ หรือ Pascal ซึ่งประกอบด้วย Test blocks หลากหลายชุด ในระบบการทดสอบจะมี 2 ส่วนคือ Production และ Quality assurance(QA) ในส่วนของ Production จะมีการทดสอบที่มีข้อจำกัดมากกว่าการทดสอบแบบ QA โดยแบบ QA จะทดสอบให้อุปกรณ์มีคุณสมบัติตามค่าดัชนี(Data Sheet) ช่วงของค่าเอาต์พุตที่ยอมรับได้เรียกว่า guardbands โดยจะต้องมีค่าที่ยอมรับผลที่เกิดจากการกระเพื่อมของสัญญาณหรือสัญญาณรบกวนได้บ้าง แต่ก็ต้องไม่กว้างจนเกินไป ถ้าความกว้างของ Guardbands ที่เหมาะสมด้วยยูนิตที่ผ่านการทดสอบแบบ Production ก็มีความเป็นไปได้สูงที่จะผ่านการทดสอบแบบ QA ด้วย โดยไม่ต้องกังวลเมื่อนำไปใช้งานปกติ

การทดสอบทางไฟฟ้าสามารถแบ่งตามลักษณะการทดสอบได้ 2 แบบคือ

- การทดสอบตามฟังก์ชัน(Function Test) จะเป็นการทดสอบว่าอุปกรณ์นั้นสามารถทำงานตามฟังก์ชันการทำงานของเครื่องนั้นหรือไม่

- การทดสอบแบบพารามตริก(Parametric Test) เป็นการทดสอบอุปกรณ์นั้นสามารถแสดงคุณสมบัติทางไฟฟ้าต่างๆเช่น แรงดัน , กระแส หรือ กำลังงานไฟฟ้า ได้ถูกต้องหรือไม่ การทดสอบ

แบบพารามตริกจะทำการ โดยการป้อนแรงดันคงที่ ที่ไหนใดไหนหนึ่งแล้ววัดผลตอบสนองทางเอกสารนี้เป็นเอกสารที่ส่งมอบไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

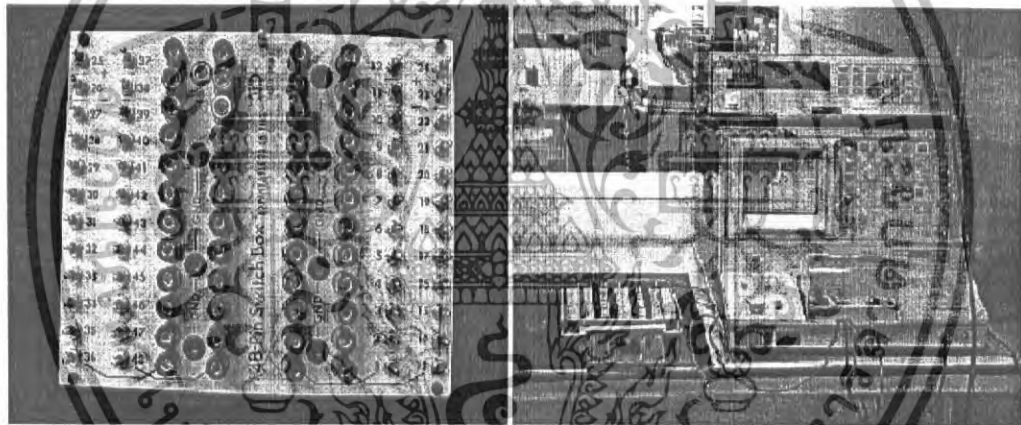
กระแส (Force-Voltage-Measure-Current : FVMC) หรือป้อนกระแสคงที่ที่โหนดใดโหนดหนึ่ง แล้ววัดผลการตอบสนองของแรงดัน (Forct-Current-Measure-Voltage)

การทดสอบทางไฟฟ้ามักจะทดสอบที่อุณหภูมิห้องที่ 25 องศาเซลเซียส และจะทดสอบที่อุณหภูมิต่างๆ ค่าหนึ่งและที่อุณหภูมิสูงๆ อีกค่าหนึ่งซึ่งต่างกันตามการใช้งานดังนี้

- Commercial Grade 0 และ 75 องศาเซลเซียส
- Extended Grade - 40 และ 85 องศาเซลเซียส
- Military Grade - 40 และ 100,125 องศาเซลเซียส

1.7 เครื่องเคฟ เทรเซอร์ (curve tracer 370A)

1.7.1 ลักษณะทางกายภาพของเครื่องเคฟ เทรเซอร์ (Curve Tracer)



รูปที่ 1.2 ลักษณะภายนอกของ Curve Tracer model 370A และแผงสวิตช์ที่ใช้ในการเลือกสัญญาณให้แก่ชิ้นงาน

1.6.2 หลักการทำงาน

เครื่องเคฟ เทรเซอร์ เป็นเครื่องที่ใช้ในการทดสอบชิปแบบพาราเมตริก โดยการป้อนสัญญาณไฟฟ้าที่ป้อนให้กับชิปจะแบ่งเป็นแรงดันไฟบวก, ไฟลบและสถานะปล่อยลอยซึ่งในสถานะปล่อยลอยนี้สามารถจ่ายแรงดันค่าต่างให้กับชิป แล้วทำการวัดเอาต์พุตที่เกิดขึ้น ในการเลือกที่จะจ่ายสัญญาณชนิดใดนั้นจะทำการเลือกผ่านทางแผงสวิตช์สามทางดังในรูปที่ 1.1 จากรูปจะเห็นว่าเป็นแผงสวิตช์ที่ทำการเลือกโดยผู้ใช้งานทำการเลือกที่ละสวิตช์ จนครบทุกสวิตช์ ซึ่งในความเป็นจริงในการทดสอบบางโหมดจำเป็นต้องเลือกสวิตช์มากกว่าหนึ่งสวิตช์พร้อมๆกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

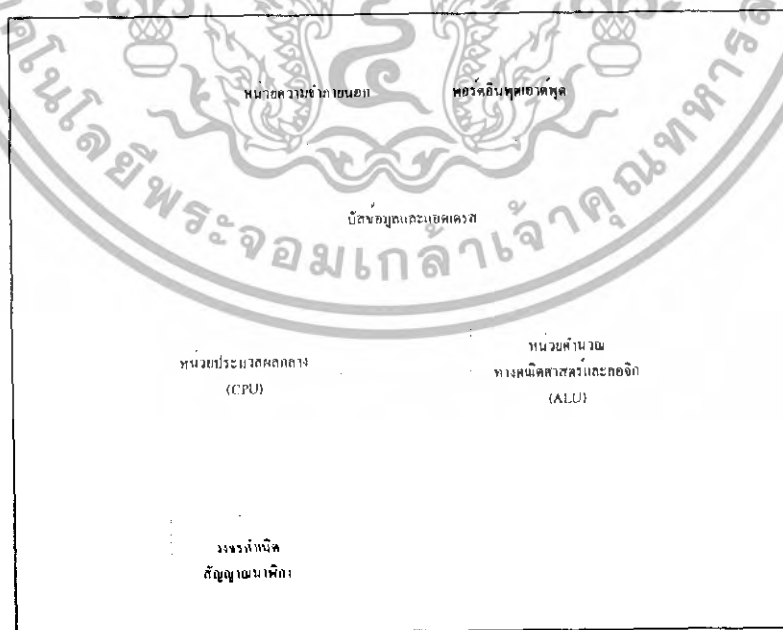
บทที่ 2

ไมโครคอนโทรลเลอร์

2.1 ความหมายของไมโครคอนโทรลเลอร์

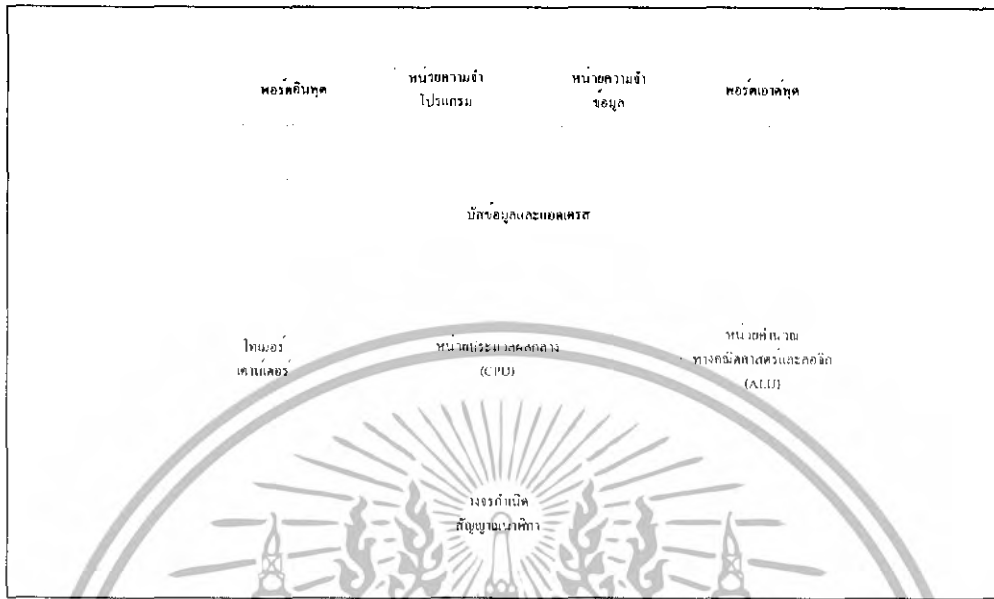
ไมโครคอนโทรลเลอร์(microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งซึ่งรวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรจับสัญญาณเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรรีเลย์อิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำมารวมกันคือ “ไมโคร”(Micro) ซึ่งหมายถึง ไมโคร โพรเซสเซอร์(Microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วย หน่วยประมวลผลกลางหรือซีพียู(CPU : Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก(ALU : Arithmetic Logic Unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า “คอนโทรลเลอร์” (Controller) หมายถึงอุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียน โปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ



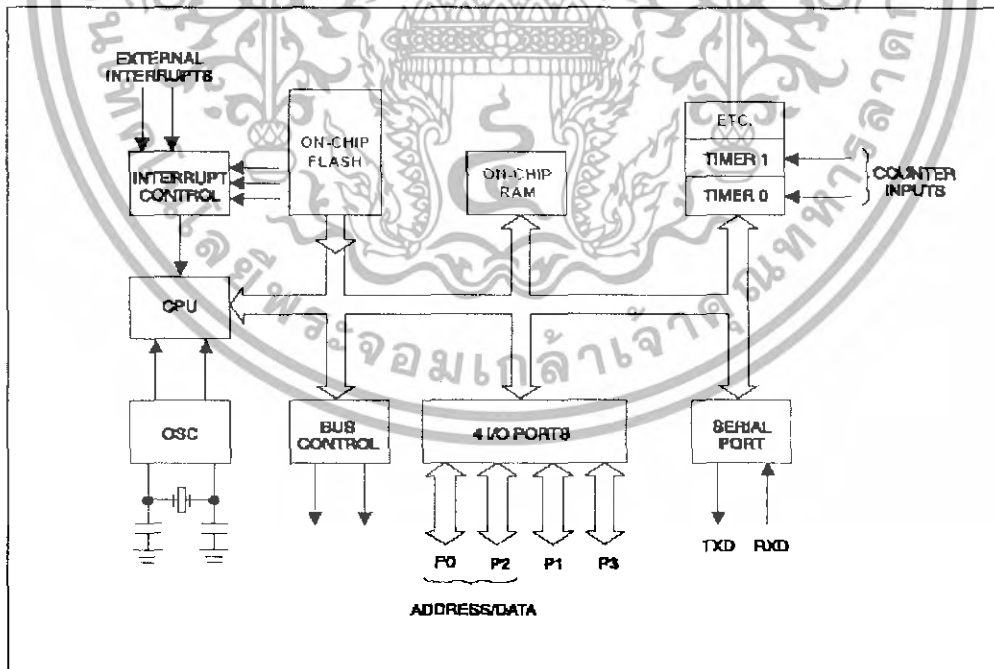
รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครโพรเซสเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



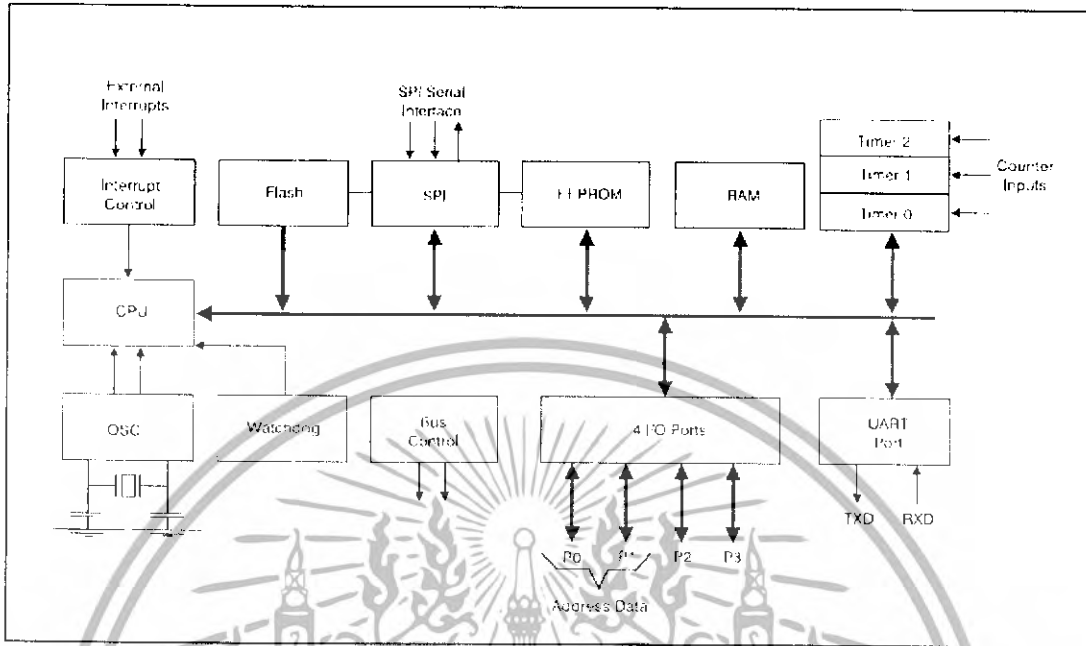
รูปที่ 2.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

2.2 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช



รูปที่ 2.3 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx

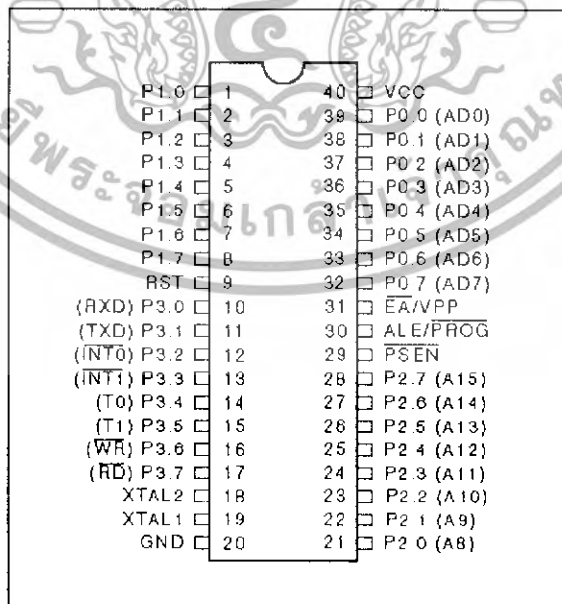
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.4 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89sxx

2.2.1 การจัดการของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูป 2.5 และ 2.6 โดยมีรายละเอียดขั้นต้น ดังนี้



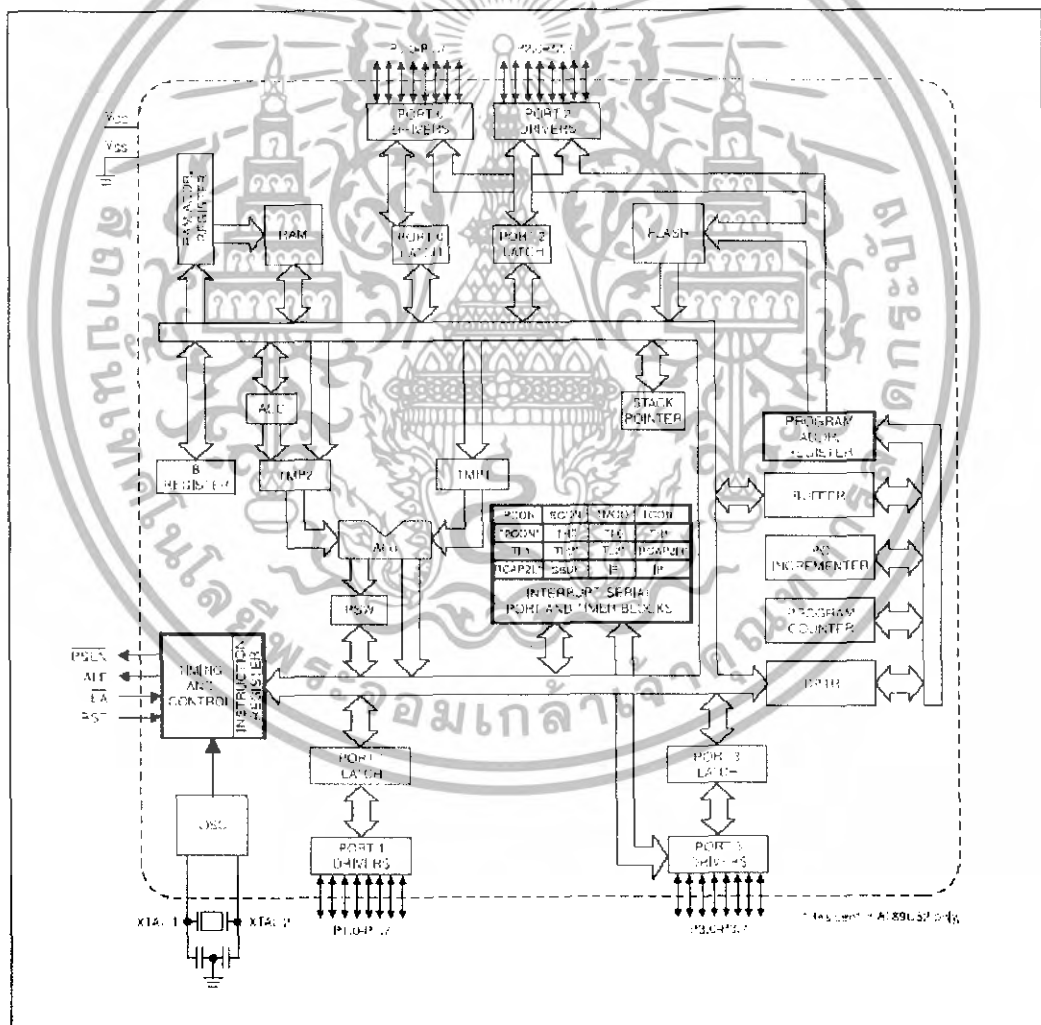
รูปที่ 2.5 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขาราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 (P0.0 – P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนี้ขาพอร์ตนี้ยังถูกใช้งานในการติดต่อดับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0 – A7) และขาข้อมูล (D0 – D7) ด้วยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อดับแอดเดรสและข้อมูล



รูปที่ 2.6 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขาพอร์ต 1 (P1.0 – P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนี้ในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต 2 (P2.0 – P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนี้ขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8 – A15)

ขาพอร์ต 3 (P3.0 – P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนี้ขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือ ขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือ ขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0

P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือ T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเทอร์รัปต์จากภายนอกช่อง 1 หรือ T1

P3.6 ใช้เป็นขาสัญญาณ \overline{WR} ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ \overline{RD} ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

ขารีเซต (Reset) ใช้ในการรีเซ็ตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซต สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมกซ์ไนเซคิล โดยที่วงจรมีการเกิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา \overline{ALE} / PROG (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนี้ขานี้ยังใช้เป็น

ขาสำหรับรับพัลส์ของการ โปรแกรมสำหรับ โปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

ขา $\overline{\text{PSEN}}$ (**Program store Enable**) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับ หน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำ โปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละเมกซีน ไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำภายใน ขานี้จะไม่มีสัญญาณใดๆออกมา

ขา $\overline{\text{EA}}/\text{Vpp}$ (**External Access enable/Programming voltage input**) ใช้สำหรับเลือกการ ติดต่อกับหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น "1"

เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายใน ไมโครคอนโทรลเลอร์ นอกจากนี้ ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการ โปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ สำหรับ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการ โปรแกรม +12 โวลต์

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตัลเพื่อสร้างสัญญาณนาฬิกาในการ กำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

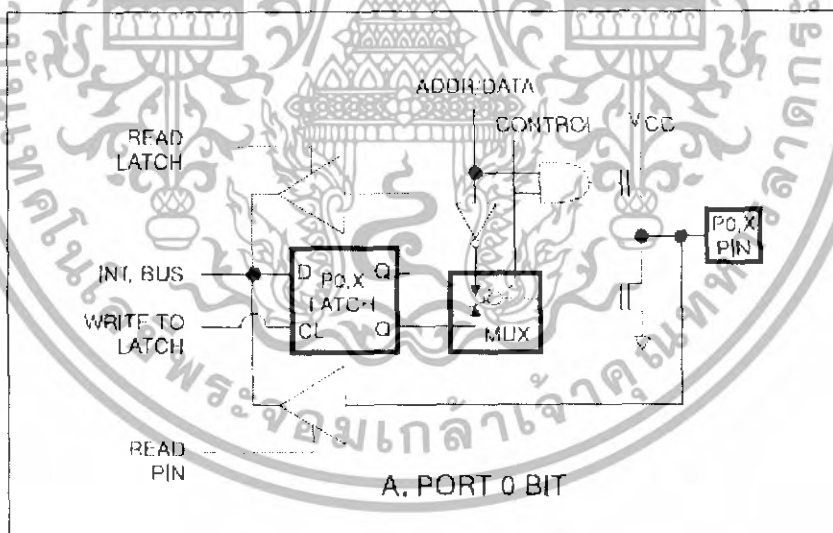
2.2.2 โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุต สำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงจรถอดและวงจรถับตลอดจนบัฟเฟอร์อินพุต ดัง แสดงให้เห็นในสถาปัตยกรรมรูปที่ 2.6

ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการ ติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ต และพอร์ต 1 บางขานอกจากจะใช้เป็น ขาพอร์ตอินพุตเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็น ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด ดังสรุปได้ในตารางที่ 2.1

ขา	เบอร์ของไมโครคอนโทรลเลอร์	หน้าที่พิเศษ
P1.0	AT89C52/AT89Sxx	ขา T2 เป็นขาอินพุตนับค่าของไทมเมอร์/เคาน์เตอร์ 2
P1.1	AT89C52/AT89Sxx	ขาอินพุตทริกเกอร์ของไทมเมอร์ 2
P1.4	AT89Sxx	ขา SS (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่มีไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟในระบบการติดต่อแบบ SPI
P1.5	AT89Sxx	ขา MOSI (Master data output , Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.6	AT89Sxx	ขา MISO (Master data input , Slave data output) ใช้ในการติดต่อกับพอร์ต SPI
P1.7	AT89Sxx	ขา SCK (Master clock output) เป็นสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

ตารางที่ 2.1 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmel



รูปที่ 2.7 วงจรภายในของพอร์ต 0

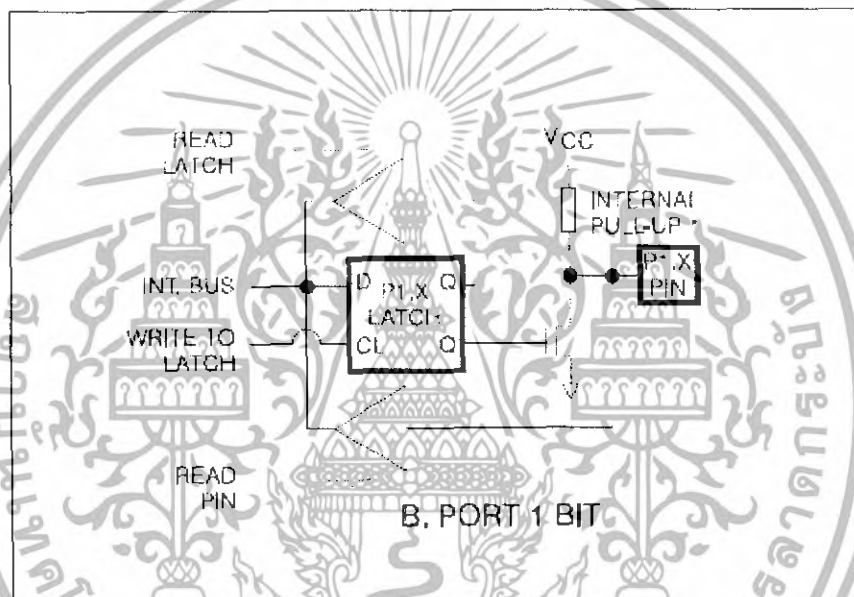
ในรูปที่ 2.7 เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิปฟล็อปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แลตซ์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟล็อป ในขณะที่ข้อมูลจะผ่านมาจากขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟล็อป

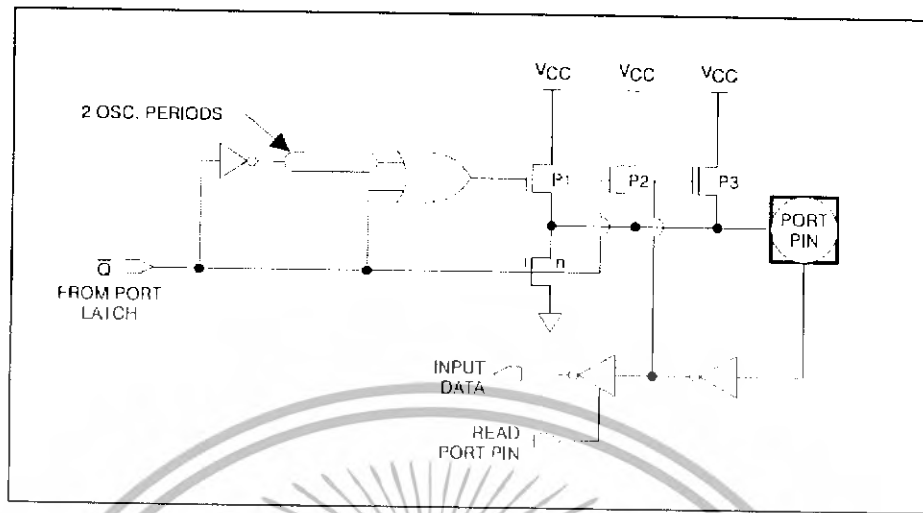
ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอก ไมโครคอนโทรลเลอร์

เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรวูล์อัปภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานวูล์อัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย

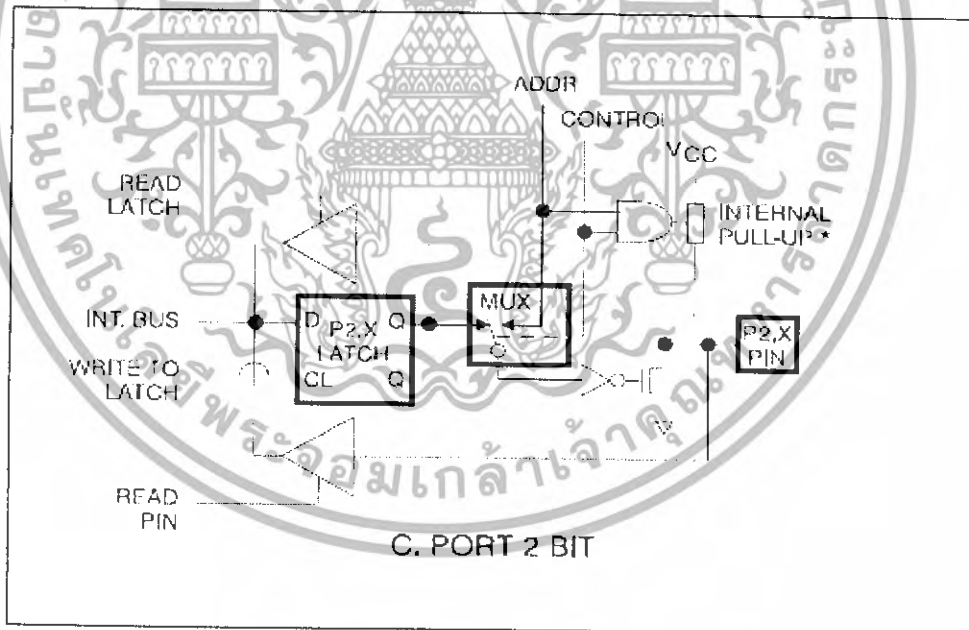


รูปที่ 2.8 วงจรภายในของพอร์ต 1

ในรูปที่ 2.8 เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้ไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรวูล์อัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำหรับรายละเอียดของวงจรวูล์อัปแสดงในรูปที่ 2.9

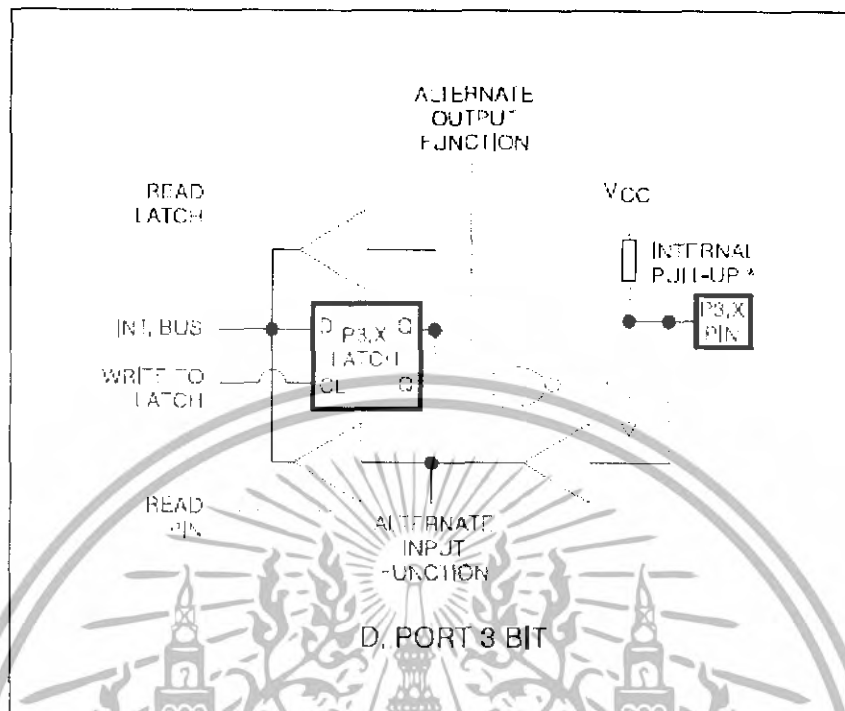


รูปที่ 2.9 วงจร พูลอัพภายในพอร์ตของไมโครคอนโทรลเลอร์



รูปที่ 2.10 วงจรภายในของพอร์ต 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 วงจรภายในของพอร์ต 3

ในรูปที่ 2.10 เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัพเพิ่มเติมเข้ามา ส่วนในรูปที่ 2.11 เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังก์ชันพิเศษเข้ามา เนื่องจากพอร์ต 3สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา

2.2.3 การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51แบบแฟรชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยั้งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟดที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัพภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น "1" สามารถรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควมกำหนดให้ทำงานในสถานะลอจิก “0” จะดีและสะดวกที่สุด(ซึ่งในปัจจุบัน อุปกรณ์ที่เชื่อมต่อกับไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก “0” แล้ว

2.2.4 การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูล ออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล “0” ออกไปทางเอาต์พุตก็ ให้เขียนข้อมูล “0” ไปยังวงจรถูก ซึ่งก็จะส่งต่อไปจับเฟด ทำในเฟดทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก “0” ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล “1” ออกไป ก็ให้ เขียนข้อมูล “1” ไปยังวงจรถูก วงจรจับก็จะหยุดทำงานทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรถูก ออ์พุตภายในเกิดเป็นลอจิก “1” ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่าน ข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีกรอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุต แต่ละขา(หรือแต่ละบิต)ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส (source current) ได้สูงสุด 10 mA และทุกขาารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาต์พุตเพื่อช่วยในการจับกระแสอีกทางหนึ่ง

2.2.5 การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรถูกของแต่ละพอร์ต

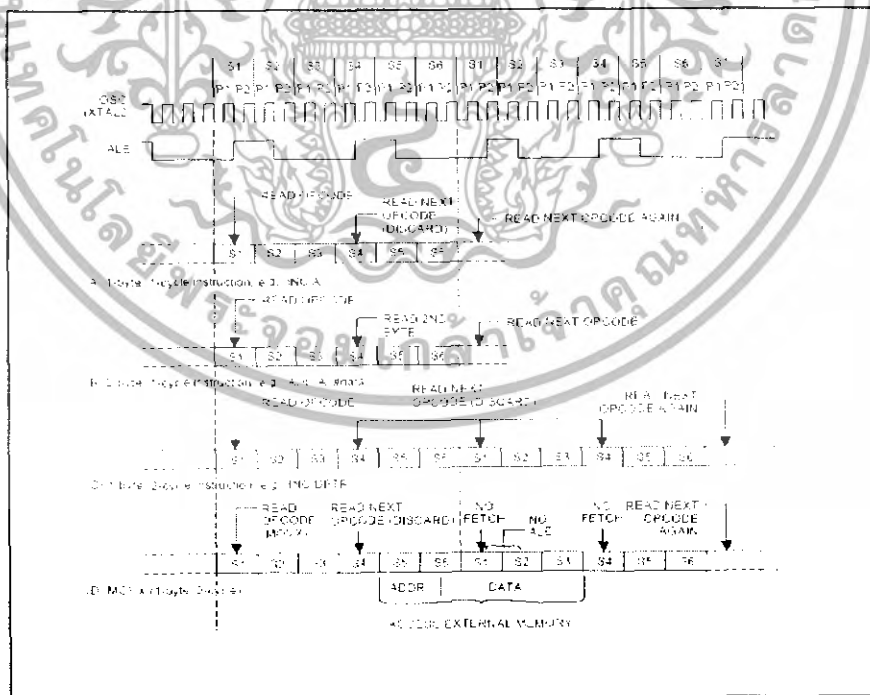
ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล “1” ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงานสถานะลอจิกที่ขาพอร์ตจะเป็น “0” เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเหมือนว่าขาพอร์ตนั้นถูกต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรถูก จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.6 จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะการทำงานของซีพียูและลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของซีพียูจะมีขั้นตอนหลักๆ 2 ขั้นตอนคือ กระบวนการเฟตช์ (fetch) เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผล ขั้นตอนต่อมาคือกระบวนการเอ็กซีคิวต์ (execute) เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นมาโดยกระบวนการก่อนหน้านี้นี้ เมื่อทำการเอ็กซีคิวต์คำสั่งเรียบร้อยแล้ว ก็จะไปเริ่มกระบวนการเฟตช์คำสั่งใหม่ต่อไป

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์จะเกิดการรีเซ็ตในลักษณะที่เรียกว่า เพาเวอร์อ้อนรีเซ็ต(Power on reset) ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม จังหวะการทำงานงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากรอบการทำงานหรือแมกซิมไซเคิล(machine cycle) ในรูป 2.12 เป็นไคอะแกรมเวลาแสดงจังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51 โดยใน 1 รอบการทำงานหรือแมกซิมไซเคิลจะแบ่งย่อยออกเป็น 6 สเตต (state) กำหนดชื่อเป็น S1 - S6 ในแต่ละสเตตมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 เมกเฮิรตซ์ จะมีคาบเวลาเท่ากับ 1ms คาบเวลาทั้งสองภายในหนึ่ง สเตตจะเรียกว่า เฟส 1(phase 1) และ เฟส 2 (phase 2)



รูปที่ 2.12 ไคเคิลการทำงานของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูปที่ 2.12 (a) และ (b) จะเป็นการเอ็ชคิวท์คำสั่งที่ใช้เวลา 1 ไชเคิล เริ่มต้นที่สเตต 1 จะเป็นการอ่านค่าออปโค้ด อันเป็นกระบวนการแลตซ์ค่าของออปโค้ดส่งไปให้รีจิสเตอร์คำสั่ง (Instruction Register : IR) การเฟตซ์ครั้งที่ 2 จะเกิดขึ้นที่สเตต 4 ภายในแมคซินไชเคิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตซ์ครั้งที่ 2 ภายในแมคซินไชเคิลเดียวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีการใช้เวลา 1 แมคซินไชเคิล จะสิ้นสุดการทำงานในสเตต 6 ของแมคซินไชเคิลเดียวกัน

ในกรณีที่คำสั่งใช้เวลา 2 ไชเคิล การทำงานของคำสั่งนั้นจะสิ้นสุดลงในสเตต 6 ของแมคซินไชเคิลที่สองดังในไดอะแกรมรูปที่ 2.12(c) สำหรับในการกระทำคำสั่ง MOVX ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไชเคิล จะไม่มีการเฟตซ์เกิดขึ้นในไชเคิลที่สองของคำสั่ง MOVX นี้ เนื่องจากซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอกดังแสดงในไดอะแกรมรูปที่ 2.12(d) จะเห็นได้ว่าเวลาในการเอ็ชคิวท์จะไม่ได้ขึ้นอยู่กับว่าทำการติดต่อกับหน่วยความจำโปรแกรมภายในหรือภายนอก

ในรูปที่ 2.13 แสดงสัญญาณและไดอะแกรมเวลาของการเข้าถึงหน่วยความจำโปรแกรมภายนอก โดยในรูปที่ 2.13(a) เป็นไดอะแกรมเวลาในขณะที่ยังไม่มีกรกระทำคำสั่ง MOVX สัญญาณที่ขา ALE และ PSEN จะเกิดการแอกทีฟ 2 ครั้งภายในหนึ่งแมคซินไชเคิล ในทุกครั้งที่ ALE เกิดการแอกทีฟที่พอร์ต 0 (P0) จะมีค่าของรีจิสเตอร์ PC ในไบต์ต่ำออกมา ในขณะที่พอร์ต 2 (P2) ก็จะมีค่าของ PC ในไบต์สูงเพื่อซีไปยังแอดเดรสต่อไปที่ต้องไปดำเนินการ สำหรับขา PSEN ก็จะมีการแอกทีฟเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ในกรณีที่กระทำคำสั่ง MOVX เพื่อเข้าถึงหน่วยความจำข้อมูลภายนอก ที่ขา PSEN จะไม่เกิดการแอกทีฟ 2 ครั้งภายใน 1 แมคซินไชเคิล เนื่องจากบัสแอดเดรสและบัสข้อมูลจะถูกใช้ในการติดต่อกับหน่วยความจำข้อมูลภายนอกแทน แต่สำหรับสัญญาณ ALE ยังคงแอกทีฟตามจังหวะการทำงานเหมือนเดิม จากไดอะแกรมเวลาสามารถสรุปได้ว่า ในการทำงาน 1 รอบหรือ 1 แมคซินไชเคิล ซีพียูในไมโครคอนโทรลเลอร์ MCS-51 จะใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไชเคิลมีค่าเท่ากับ 1ms หรือมีความเร็วในการทำงานภายใน 1 MHz ในกรณีที่ใช้ความถี่สัญญาณนาฬิกา 12 MHz ดังนั้นถ้าต้องการทราบความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถหาได้จาก ค่าความถี่สัญญาณนาฬิกาหารด้วย 12 และถ้าต้องการหาค่าเวลาของ 1 รอบ การทำงานหรือแมคซินไชเคิล สามารถทำได้โดยการหาส่วนกลับของความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถสรุปสูตรทางคณิตศาสตร์ได้ดังนี้

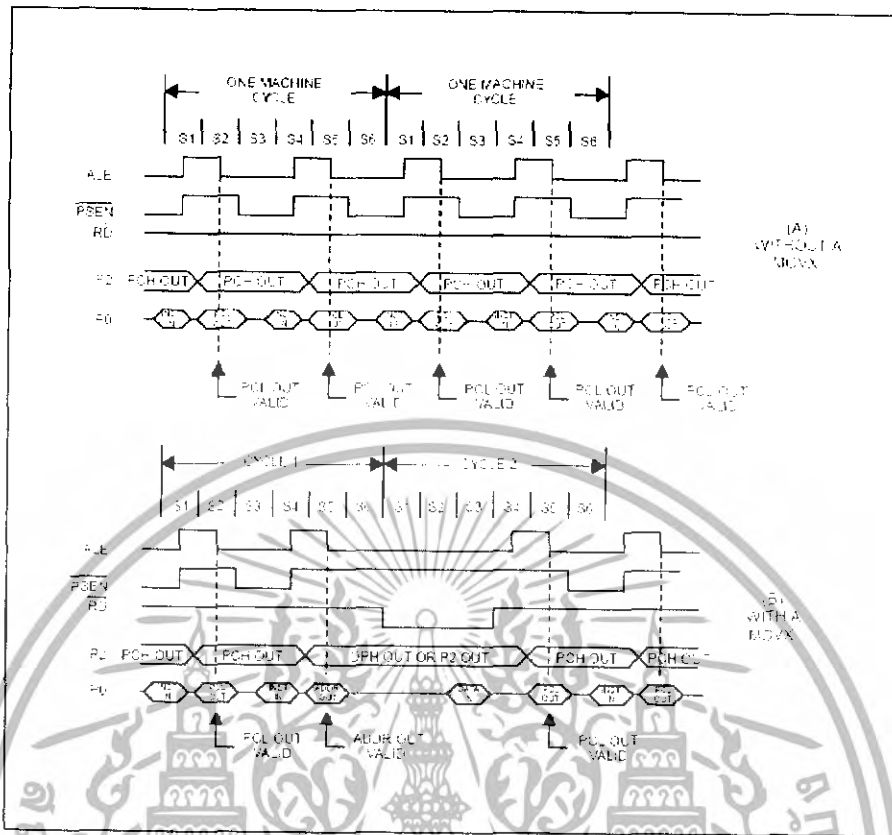
ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ

ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2) หารด้วย 12

เวลา 1 แมคซินไชเคิล = 1/ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง



รูปที่ 2.13 ไดอะแกรมเวลาแสดงการติดต่อและเข้าถึงหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์

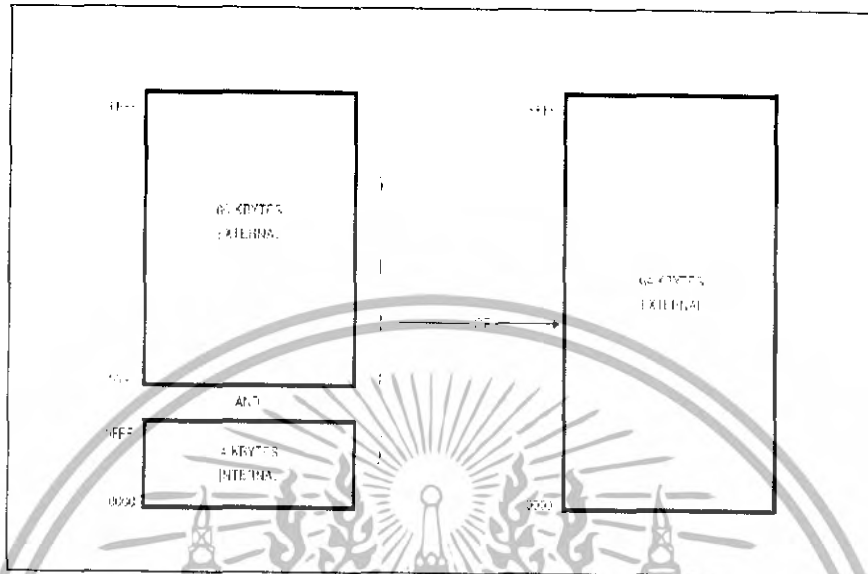
2.3 การจัดการหน่วยความจำของไมโครคอนโทรลเลอร์ MCS - 51แบบแฟลช

ในไมโครคอนโทรลเลอร์ MCS - 51แบบแฟลชมีหน่วยความจำภายในหลักๆ อยู่ 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์ ต่อไปนี้จะกล่าวถึงรายละเอียดของการจัดสรรหน่วยความจำภายในไมโครคอนโทรลเลอร์ MCS - 51 แบบแฟลช การเชื่อมต่อกับหน่วยความจำภายนอก และข้อมูลเบื้องต้นของรีจิสเตอร์ฟังก์ชันพิเศษที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS - 51 แบบแฟลช

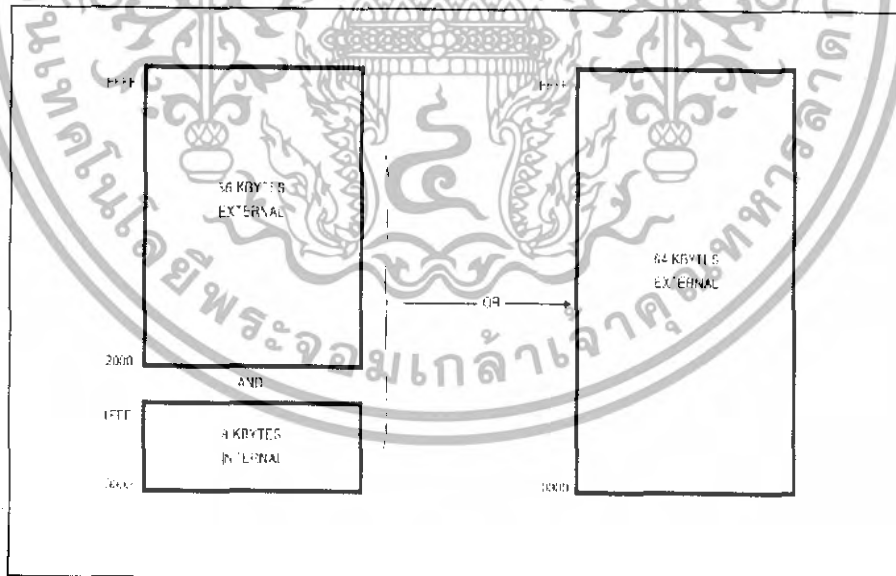
73164

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.1 หน่วยความจำโปรแกรม (Program memory)



รูปที่ 2.14 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ AT89C51



รูปที่ 2.15 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ AT89C52

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.14 และ 2.15 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS - 51 แบบแฟลชในบอร์ดต่างๆที่นิยมใช้งาน อันประกอบด้วยเบอร์ AT89C51 และ AT89C52 จะเห็นได้ว่า ทั้งสองบอร์ดสามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกหรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังในรูป 2.14 โดยภายใน AT89C51 มีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่ AT89C52 จะมีขนาด 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน หากใช้ AT89C51 ก็จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อกับหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปรแกรมใช้เก็บข้อมูลของโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หรือที่เรียกว่า โปรแกรมมอนิเตอร์(Monitor program) หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม (EPROM : Erasable Programmable Read-Only Memory) ซึ่งสามารถทำการอ่านได้เพียงอย่างเดียว หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซ็ตให้เริ่มต้นการทำงานจะต้องมาเริ่มต้นที่แอดเดรส 0000H นี้เสมอ อย่างไรก็ตาม ในพื้นที่ของหน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายในหรือภายนอกก็ตาม ต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 0003H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทมเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอกกำหนดไว้ที่แอดเดรส 0013H

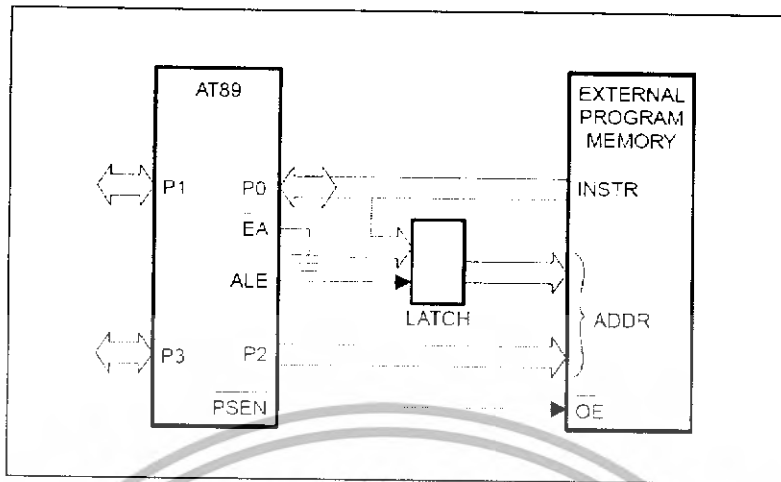
พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทมเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ของการสื่อสารแบบอนุกรม กำหนดไว้ที่แอดเดรส 0023H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทมเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่มีความจำโปรแกรมภายใน แต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วยสามารถทำได้โดย ต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายใน ของไมโครคอนโทรลเลอร์ ยกตัวอย่างไมโครคอนโทรลเลอร์ AT89C51 มีหน่วยความจำโปรแกรมขนาด 4 กิโลไบต์ มีแอดเดรสอยู่ระหว่าง 0000H - 0FFFH เมื่อต่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H - FFFFH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

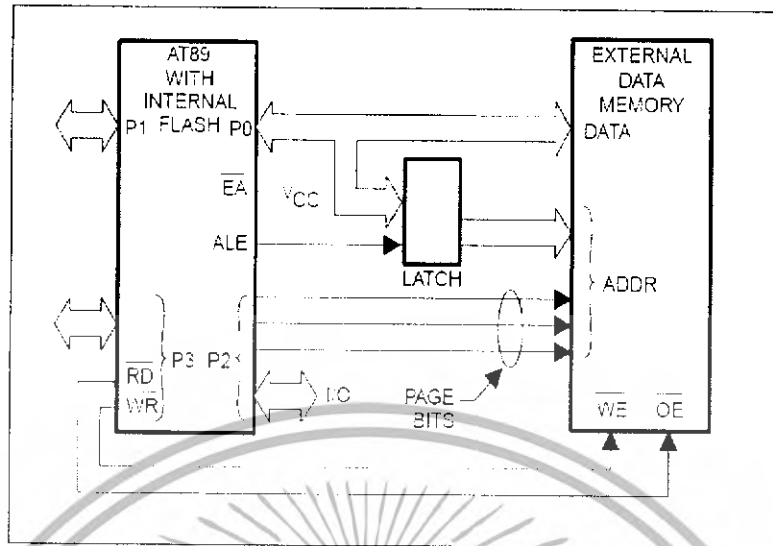


รูปที่ 2.16 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์

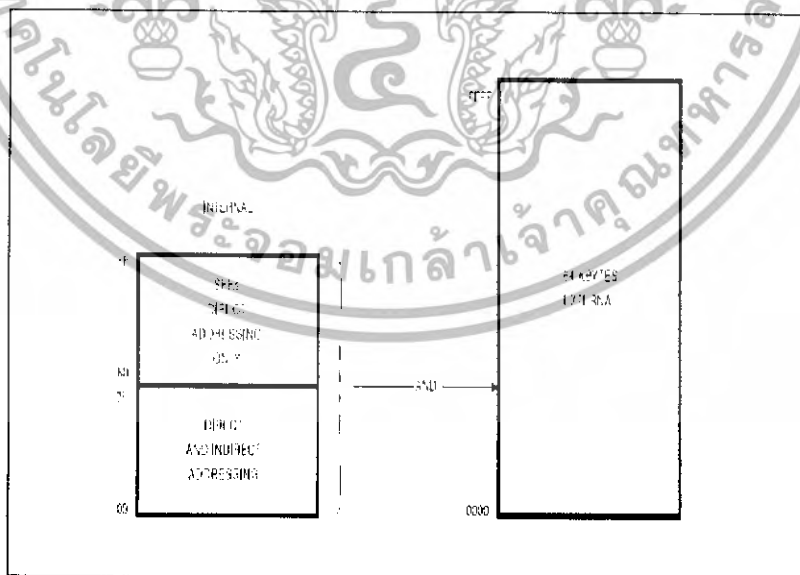
การต่อหน่วยความจำภายนอกแสดงดังรูป 2.16 ขาพอร์ต P0.0 – P0.7 ใช้เป็นขาข้อมูล D0 – D7 และขาแอดเรสไบต์ต่ำ โดยผ่านวงจรถ่ายที่ซึ่งปกติใช้ไอซีเบอร์ 74HC573 และใช้สัญญาณ ALE และ PSEN ในการเลือกใช้งานขา P0.0 – P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเรส ในขณะที่ขา P2.0 – P2.7 ใช้ในการเชื่อมต่อกับขาแอดเรสไบต์สูง A8 – A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ไมโครคอนโทรลเลอร์จะเหลือขาพอร์ตเพียง 16 บิต คือ ที่ขาพอร์ต P1.0 – P1.7 และ P3.0 – P3.7

2.3.2 หน่วยความจำข้อมูล (Data memory)

มีด้วยกัน 2 แบบคือ หน่วยความจำข้อมูลภายนอกและภายใน โดยไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก การติดต่อกับหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชแสดงในรูป จะเห็นได้ว่า มีลักษณะคล้ายกับการติดต่อกับหน่วยความจำโปรแกรมภายนอก แตกต่างกันที่มีสัญญาณที่ใช้สำหรับการอ่านและเขียนหน่วยความจำข้อมูลภายนอก นั่นคือ ขา \overline{RD} และ \overline{WR}



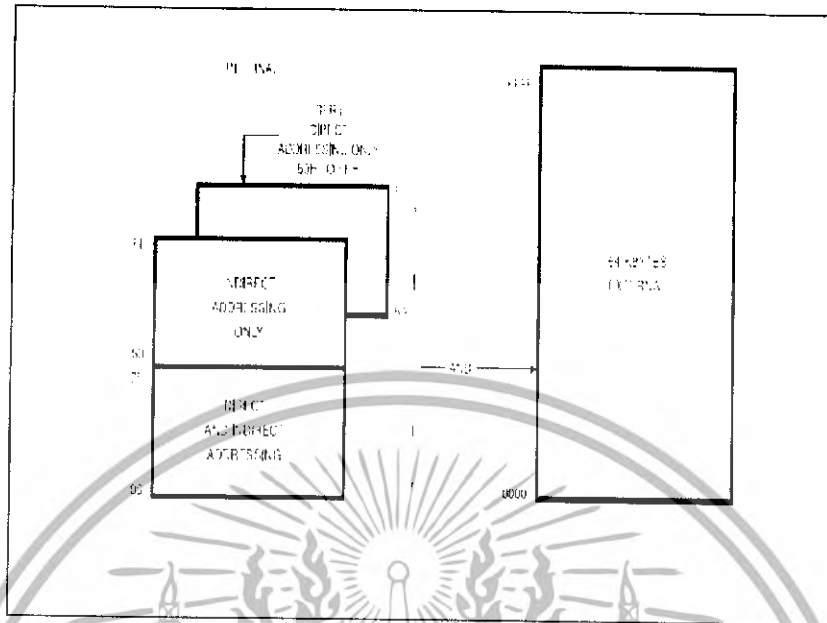
รูปที่ 2.17 การเชื่อมต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม (RAM : Random Access Memory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ ST89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำภายในแบ่งเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่าง (lower) ,ส่วนบน (upper) และรีจิสเตอร์ฟังก์ชันพิเศษ (SFR : Special Function Register) แต่ละส่วนมีขนาด 128 ไบต์ ดังแสดงในรูปที่ 2.18 และ 2.19



รูปที่ 2.18 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์เบอร์

AT89C51

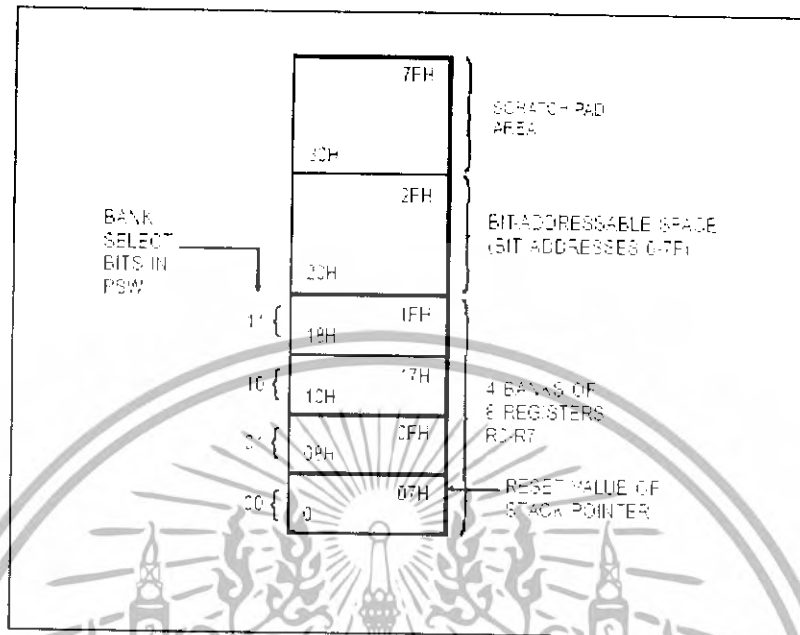
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.19 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์เบอร์ AT89C52

จะเห็นได้ว่า หน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชันพิเศษมีตำแหน่งทับซ้อนกัน แต่จะใช้การติดต่อที่แตกต่างกัน และในไมโครคอนโทรลเลอร์ MCS-51 บางเบอร์จะไม่มีหน่วยความจำส่วนบน

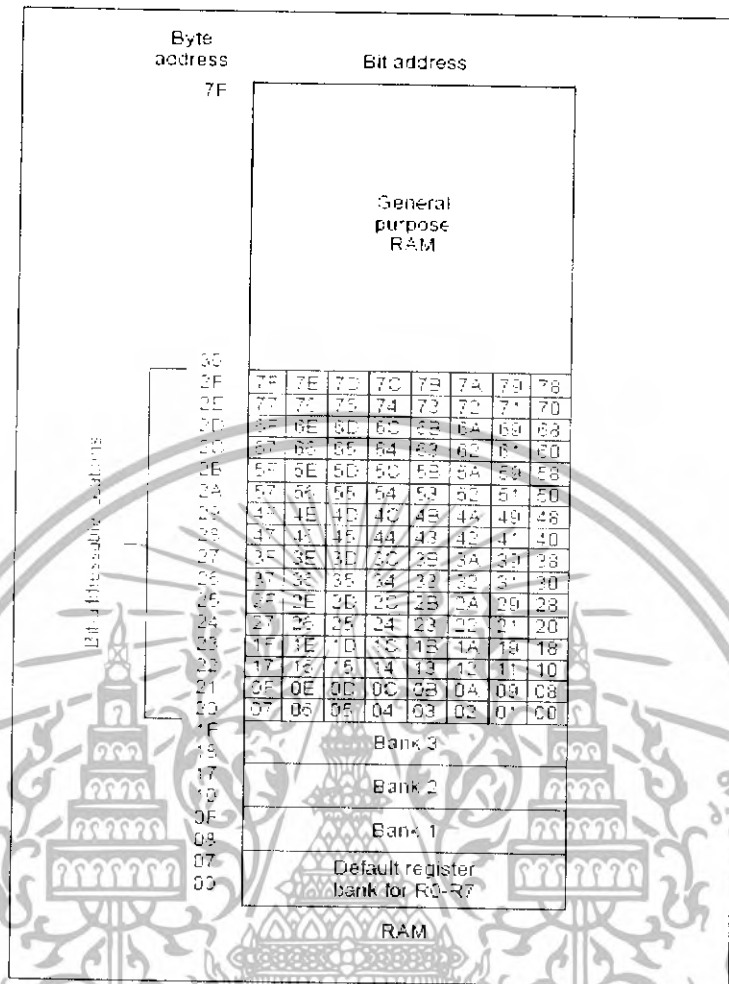
ขนาดของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยแท้จริงแล้วมีเพียง 256 ไบต์ แต่ด้วยการจัดการเข้าถึงที่แตกต่างกัน จึงดูเหมือนว่า ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำข้อมูลสูงสุดถึง 384 ไบต์ โดยในหน่วยความจำข้อมูลส่วนล่างขนาดขนาด 128 ไบต์ มีแอดเดรสอยู่ที่ 00H - 7FH สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128 ไบต์เช่นกัน มีแอดเดรสอยู่ที่ 80H - FFH สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H - FFH เช่นเดียวกับหน่วยความจำส่วนบน แต่สำหรับรีจิสเตอร์ SFR ใช้การเข้าถึงแบบโดยตรง ดังนั้นเพื่อความสะดวกและง่ายตลอดจนป้องกันความสับสนในการเขียนโปรแกรมสำหรับผู้เริ่มต้นจึงควรใช้หน่วยความจำข้อมูลภายในเพียง 128 ไบต์จากหน่วยความจำข้อมูลส่วนล่างร่วมกับรีจิสเตอร์ SFR



รูปที่ 2.20 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างของไมโครคอนโทรลเลอร์

ในรูปที่ 2.20 แสดงการจัดสรรหน่วยความจำข้อมูลส่วนล่าง หน่วยความจำ 32 ไบต์ต่ำสุดที่แอดเดรส 00H – 1FH แบ่งเป็น 4 กลุ่มเรียกว่า แบงก์ (bank) แต่ละแบงก์มีรีจิสเตอร์ 8 ตัวคือ R0 – R7 การติดต่อกับหน่วยความจำในแบงก์ใดให้กำหนดที่รีจิสเตอร์ PSW (Program Status Word register)

หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H -2FH เป็นพื้นที่สำหรับใช้งานทั่วไปสามารถเข้าถึงได้ในระดับบิต (Bit addressable) และหน่วยความจำข้อมูลที่เหลือ 80 ไบต์ จะต้องแบ่งส่วนหนึ่งสำรองไว้เป็นพื้นที่ของสแต็ค (stack : ที่พักข้อมูลชั่วคราวในกรณีที่ซอฟต์แวร์มีการกระโดดไปทำงานในโปรแกรมย่อย) การเข้าถึงหน่วยความจำในส่วนนี้ต้องเข้าถึงในระดับ ไบต์



รูปที่ 2.21 โครงสร้างของหน่วยความจำข้อมูลภายในส่วนบนของไมโครคอนโทรลเลอร์

ในรูปที่ 2.21 แสดงโครงสร้างของหน่วยความจำข้อมูลส่วนบน ซึ่งจะมีลักษณะที่คล้ายกับหน่วยความจำข้อมูลส่วนล่าง หากแต่ใน 80 ไบต์ บนไมโครคอนโทรลเลอร์ไม่จำเป็นต้องสำรองไว้สำหรับสแต็ก และต้องใช้การเข้าถึงในลักษณะโดยอ้อมเท่านั้น

2.3.3 รีจิสเตอร์ฟังก์ชันพิเศษ(Special Function Register : SFR)

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 มีด้วยกัน 22 ตัว สำหรับเบอร์ AT89C51 และ 28 ตัวในเบอร์ AT89C52 และอนุกรม AT89Sxx ทั้งนี้เนื่องจากใน AT89C52 และอนุกรม AT89Sxx มีจำนวนไทมเมอร์คอนโทรเลอร์มากกว่า AT89C51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Byte address	Bit address								
FF									
F9	F7	F6	F5	F4	F3	F2	F1	F0	B
EC	E7	E6	E5	E4	E3	E2	E1	EO	ACC
DO	D7	D6	D5	D4	D3	D2	—	D0	PSW
E8	—	—	—	EO	EB	EA	EB	EB	IP
8C	B7	B6	B5	B4	B3	B2	B1	EO	P3
A3	AF	—	—	AO	AB	AA	AO	A2	IE
AC	A7	A6	A5	A4	A3	A2	A1	AO	P2
99	not bit addressable								SBUF
96	9F	9E	9D	9C	9B	9A	99	98	SCON
90	8F	8E	8D	8C	8B	8A	89	80	P1
8D	not bit addressable								T41
8C	not bit addressable								T40
8B	not bit addressable								TL1
8A	not bit addressable								TL0
89	not bit addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit addressable								PCON
83	not bit addressable								DPH
82	not bit addressable								DPL
81	not bit addressable								SP
80	87	86	85	84	83	82	81	80	P0

Special Function Registers

รูปที่ 2.22 การจัดสรรพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ(SFR)

รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H - FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงได้โดยตรง (Direct addressing) ในรูป 2.22 แสดงการจัดสรรพื้นที่ของรีจิสเตอร์ SFR ในแต่ละตัวในหน่วยความจำข้อมูลส่วนบน สำหรับรายละเอียดเบื้องต้นของรีจิสเตอร์ SFR มีดังนี้

2.3.3.1 รีจิสเตอร์สถานะของโปรแกรม(Program Status Word : PSW)

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต จึงสามารถกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ มีแอดเดรสอยู่ที่ D0H ทำหน้าที่เก็บสถานะของการทำงานของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมในขณะนั้นจะเรียกสถานะต่างๆของโปรแกรมว่า แฟล็ก(flag) เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงนั้นจะปรากฏที่บิตต่างๆ ของรีจิสเตอร์ PSW

รายละเอียดของรีจิสเตอร์แสดงสถานะของโปรแกรม(PSW) มีดังต่อไปนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CY	AC	F0	RS1	RS0	OV	-	P

CY : แฟล็กทด (Carry flag) เป็น “1” เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิก แล้วค่าของแอกคิวมูลเตอร์เกิน 255(ฐานสิบ) หรือ FFH

AC : แฟล็กทดเสริม(Auxiliary Carry flag) เป็น “1” เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายัง บิต 4

F0 : แฟล็กใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่าจะกระทำคำสั่งใดๆที่บิตนี้จะไม่มีกรเปลี่ยนแปลง

RS1 : บิตเลือกรีจิสเตอร์เบงก์1 (Register Select 1) ใช้งานร่วมกับบิต RS0 เพื่อเลือกเบงก์ของรีจิสเตอร์ R0 – R7

RS0 : บิตเลือกรีจิสเตอร์เบงก์0 (Register Select 0) ใช้งานร่วมกับบิต RS1 เพื่อเลือกเบงก์ของรีจิสเตอร์ R0 – R7

OV : บิตเกิน(Overflow) เป็น “1” เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ทำให้เกิดการทดข้ามจากบิต 6 มายังบิต 7 ของแอกคิวมูลเตอร์ หรือแอกคิวมูลเตอร์มีค่าเกิน 127(ฐานสิบ) นอกจากนั้นยังใช้เป็นการแสดงค่าลบด้วย

- : บิตนี้ผู้ใช้งานสามารถกำหนดใช้งานได้อย่างอิสระ

P : บิตพาริตี(Parity) ใช้ในการตรวจสอบจำนวนค่า “1” ภายในแอกคิวมูลเตอร์ ถ้าหากในแอกคิวมูลเตอร์มีจำนวนบิตที่เป็น “1” รวมกันเป็นเลขคู่ บิตนี้จะป็น “0” ถ้ารวมกันเป็นเลขคี่ บิตนี้จะป็น “1”

จะเห็นได้ว่า นอกจากรีจิสเตอร์ PSW ถูกใช้ในการเก็บสถานะของโปรแกรมแล้ว ที่บิต RS0 และ RS1 ยังใช้ในการเลือกเบงก์ของหน่วยความจำส่วนล่าง ซึ่งเป็นพื้นที่ของรีจิสเตอร์ R0 – R7 ด้วย ดังมีรายละเอียดแสดงในตารางที่ 2.2 โดยปกติแล้วในการใช้งานรีจิสเตอร์ R0 – R7 มักนิยมใช้เบงก์ 0 เป็นอันดับแรก หากไม่เพียงพอจึงเลือกในเบงก์อื่นๆมาใช้ แต่ต้องระมัดระวังในการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดค่าและลำดับการติดต่อให้ดี มีเช่นนั้น อาจทำให้การเขียนโปรแกรมเกิดความสับสน ดังนั้น สำหรับผู้เริ่มต้นใช้งานไมโครคอนโทรลเลอร์ MCS-51 จึงควรเลือกใช้รีจิสเตอร์ R0 – R7 ในแบงก์ 0 เพียงแบงก์เดียวให้ชำนาญเสียก่อน

RS1	RS0	แบงก์ของรีจิสเตอร์	ช่วงแอดเดรส
0	0	แบงก์ 0	00H – 07H
0	1	แบงก์ 1	08H – 0FH
1	0	แบงก์ 2	10H – 17H
1	1	แบงก์ 3	18H – 1FH

ตารางที่ 2.2 การเลือกแบงก์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงก์ R0 – R7

การกำหนดค่าของรีจิสเตอร์ PSW เพื่อเลือกใช้งานรีจิสเตอร์ R0 – R7 ควรกำหนดไว้ที่ตอนต้นของโปรแกรมเสมอ เพื่อจะได้เขียนโปรแกรมติดต่อกับรีจิสเตอร์ R0 – R7 ได้อย่างสะดวกและไม่เกิดความผิดพลาด

2.3.3.2 แอควิวูเลเตอร์(Accumulator : ACC)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์ โดยเฉพาะอย่างยิ่งในการคำนวณทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้ให้แก่ซีพียูเพื่อทำการประมวลผลต่อไป อาจเรียกสั้นๆว่ารีจิสเตอร์ A หรือ ACC รีจิสเตอร์นี้สามารถเข้าถึงระดับบิตได้

2.3.3.3 รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง F0H มีหน้าที่พิเศษคือ หากต้องการคูณหรือหารทางคณิตศาสตร์ ต้องนำข้อมูลที่ต้องการการคูณหรือหารมาเก็บไว้ในรีจิสเตอร์ B แล้วจึงกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป

ในกรณีที่ 'ไม่' ด้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์ B นี้ในการเก็บข้อมูลทั่วไปได้เหมือนกับรีจิสเตอร์ปกติ และสามารถเข้าถึงในระดับบิตได้เช่นเดียวกับรีจิสเตอร์ A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.3.4 โปรแกรมเคาน์เตอร์(Program Counter : PC)

มีขนาด 16 บิต มีหน้าที่แจ้งแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งถัดไปที่ซีพียูจะต้องไปทำงาน รีจิสเตอร์ PC เป็นรีจิสเตอร์ตัวเดียวที่ไม่ได้จัดสรรไว้ร่วมกับรีจิสเตอร์ SFR ตัวอื่นๆ การเปลี่ยนแปลงค่าของรีจิสเตอร์ PC จะขึ้นอยู่กับผลของการกระทำคำสั่งภายในหน่วยความจำโปรแกรมที่ผู้เขียนโปรแกรมกำหนด

รีจิสเตอร์ PC มีความสำคัญมาก โดยเฉพาะอย่างยิ่งในการตรวจสอบการทำงานของโปรแกรมว่า ดำเนินไปตามลำดับขั้นตอนตามที่กำหนดไว้หรือไม่

2.3.3.5 สแต็กพอยน์เตอร์(Stack Pointer : SP)

สแต็กพอยน์เตอร์หรือรีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง 81H ใช้ในการเก็บค่าตำแหน่งของตัวชี้สแต็ก ซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อย หรือกระโดดจากโปรแกรมย่อยกลับมายังโปรแกรมหลัก เมื่อมีการรีเซตเกิดขึ้น (รีเซต คือ การกระทำที่ส่งผลให้ซีพียูต้องเริ่มต้นการทำงานใหม่ตั้งแต่ต้น) ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H ดังนั้นแอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H

2.3.3.6 รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์(Data Pointer : DPTR)

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไปต์สูง (DPH) และรีจิสเตอร์ชี้ข้อมูลไปต์ต่ำ (DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H สำหรับ DPL และ 83H สำหรับ DPH รีจิสเตอร์ DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไม่โครคอนโทรลเลอร์ต้องการติดต่อด้วย

2.3.3.7 รีจิสเตอร์พอร์ต(Port register)

เป็นรีจิสเตอร์ขนาด 8 บิตที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 มี 4 ตัวคือรีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิสเตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H, รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H และรีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหรือเขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องกระทำผ่านรีจิสเตอร์นี้ทุกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.3.8 รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม(Serial Data Buffer : SBUF)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ส่งออกหรือรับเข้าของ วงจรสื่อสารแบบอนุกรมที่อยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล(transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล(receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูลเพื่อส่งข้อมูลออกจาก ไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลแบบอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของ ไมโครคอนโทรลเลอร์ MCS-51

สำหรับรายละเอียดของรีจิสเตอร์ SBUF และวงจรสื่อสารแบบอนุกรมภายใน ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชจะกล่าวถึงในหัวข้อการสื่อสารผ่านพอร์ตอนุกรม

2.3.3.9 รีจิสเตอร์ไทมเมอร์(Timer register)

เป็นรีจิสเตอร์ขนาด 16 บิต แบ่งเป็นไบต์สูงและไบต์ต่ำเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทมเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ ภายในไมโครคอนโทรลเลอร์ เพื่อใช้ในการสร้างฐานเวลา, จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน บางทีเรียกรีจิสเตอร์ตัวนี้ว่า รีจิสเตอร์ไทมเมอร์/เคาน์เตอร์

ในไมโครคอนโทรลเลอร์เบอร์ AT89C51 มีรีจิสเตอร์ไทมเมอร์/เคาน์เตอร์ 2 ตัว แบ่งเป็น T0 หรือ Timer 0 และ T1 หรือ Timer 1 ในรีจิสเตอร์ยังแบ่งเป็นรีจิสเตอร์ไทมเมอร์ไบต์ต่ำ(TL) และ รีจิสเตอร์ไทมเมอร์ไบต์สูง(TH) เหมือนกัน โดยรีจิสเตอร์ TLO มีแอดเดรสอยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8BH ในขณะที่ TL1และ TH1 มีแอดเดรสอยู่ที่ 8CH และ 8DH สำหรับในเบอร์ AT89C52 และอนุกรม AT89Sxx จะมีรีจิสเตอร์ไทมเมอร์/เคาน์เตอร์ถึง 3 ตัว โดยมีรีจิสเตอร์ TL2 และ TH2 ซึ่งมีแอดเดรสอยู่ที่ 0CCH และ 0CDH เพิ่มเติมเข้ามา

2.3.3.10 รีจิสเตอร์แคปเจอร์(Capture register)

เป็นรีจิสเตอร์ขนาด 16 บิต มีเฉพาะในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx เท่านั้น เนื่องจากต้องใช้ร่วมกับไทมเมอร์/เคาน์เตอร์ 2 (Timer 2) โดยรีจิสเตอร์แคปเจอร์นี้มีชื่อเรียกอย่างย่อว่า รีจิสเตอร์ RCAP2 ซึ่งแบ่งออกเป็นไบต์ต่ำคือ RCAP2L มีแอดเดรสอยู่ที่ 0CAH และไบต์สูงคือ RCAP2H มีแอดเดรสอยู่ที่ 0CBH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รีจิสเตอร์แคปเจอร์จะถูกใช้งานเมื่อกำหนดให้ไทมเมอร์ 2 ทำงานในโหมดที่กำหนดให้ ไมโครคอนโทรลเลอร์ทำการตรวจจับการเปลี่ยนแปลงสถานะทางลอจิกที่ขา T2EX ทั้งนี้เพื่อประโยชน์ในการวัดคาบเวลา ความถี่ และการเปลี่ยนแปลงของสัญญาณพัลส์ที่ขา T2EX

2.3.3.11 รีจิสเตอร์ควบคุม(Control register)

- รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช
- รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช
- รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทมเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2CON ใช้สำหรับไทมเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx
- รีจิสเตอร์ TMOD และ T2MOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะการทำงานของไทมเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2MOD ใช้สำหรับไทมเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx
- รีจิสเตอร์ IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเทอร์รัปต์ (interrupt คือการขัดจังหวะการทำงานปกติของซีพียู) โดย IE เป็นรีจิสเตอร์สำหรับเอ็นเอเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเทอร์รัปต์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเทอร์รัปต์ว่า จะให้ซีพียูตอบสนองการเกิดอินเทอร์รัปต์ในลักษณะใดก่อนหรือหลัง

2.4 ไทมเมอร์/ เคาน์เตอร์

ไทมเมอร์/เคาน์เตอร์(Timer/counter) เป็นอีกส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์ เนื่องจากในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลา ทั้งนี้เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พัลส์ เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าของการนับ รวมไปถึงการกำหนดอัตราเร็วในการสื่อสารข้อมูลของพอร์ตอนุกรมด้วย

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 มีวงจรถ่ายโอน เคาท์เตอร์ ขนาด 16 บิต 2 ตัวโดยค่าของวงจรถ่ายโอน เคาท์เตอร์นี้จะเก็บไว้ในรีจิสเตอร์ขนาด 16 บิตที่ชื่อ ไทเมอร์ 0 (Timer 0) และ ไทเมอร์ 1 (Timer 1) เรียกสั้นๆว่า T0 และ T1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx จะมี ไทเมอร์ เคาท์เตอร์ถึง 3 ตัวคือมี ไทเมอร์ 2 (Timer 2) AT89C52 และอนุกรม AT89Sxx เพิ่มเติม โดยรีจิสเตอร์ ไทเมอร์ เคาท์เตอร์ ทั้งสามตัวสามารถกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือ ไทเมอร์ และตัวนับหรือ เคาท์เตอร์ ได้อย่างอิสระต่อกัน

2.4.1 การทำงานเป็นไทเมอร์

เมื่อกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือ ไทเมอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นในทุกๆเมกซ์ซินไซเคิล ดังนั้นเมื่อทำงานเป็น ไทเมอร์รีจิสเตอร์จะทำการนับค่าของเมกซ์ซินไซเคิลนั่นเอง และเนื่องจากเมกซ์ซินไซเคิลประกอบด้วยคาบเวลาของวงจรถ่ายโอนสัญญาณนาฬิกา 12 คาบเวลา ดังนั้นอัตราในการนับของรีจิสเตอร์จึงเท่ากับ $1/12$ ของความถี่สัญญาณนาฬิกา

2.4.2 การทำงานเป็นเคาน์เตอร์

เมื่อทำงานเป็นตัวนับหรือเคาน์เตอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิกจาก “1” เป็น “0” เกิดขึ้นที่ขาอินพุตทางฮาร์ดแวร์ของวงจรถ่ายโอน ไทเมอร์ เคาท์เตอร์ ซึ่งก็คือขา T0 (P3.5) และขา T1 (P3.6) สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 รวมทั้งขา T2 (P1.0) ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx โดยจะมีการสุ่มรับสัญญาณจากขาอินพุตในทุกๆคาบเวลาที่ 2 ของสเตตที่ 5 (S5P2) ในแต่ละเมกซ์ซินไซเคิล

เมื่อสัญญาณอินพุตเปลี่ยนแปลงจาก “1” เป็น “0” เป็นเวลาหนึ่งไซเคิลในไซเคิลต่อมาค่าของการนับจะเพิ่มขึ้นหนึ่งค่า และจะไปปรากฏในรีจิสเตอร์ภายในคาบเวลาที่ 1 ของสเตตที่ 3 (S3P1) ของเมกซ์ซินไซเคิลต่อไปหลังจากที่ตรวจจบการเปลี่ยนแปลงที่ขา ไทเมอร์อินพุตแล้ว เมื่อเป็นเช่นนั้นในกระบวนการตรวจจบการเปลี่ยนแปลงของสัญญาณอินพุตที่ขา ไทเมอร์จะต้องใช้ 2 เมกซ์ซินไซเคิล อัตราการนับของเคาน์เตอร์จึงเท่ากับ $1/24$ ของความถี่สัญญาณนาฬิกา ดังนั้น ความถี่สูงสุดของสัญญาณอินพุตที่ ไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถตรวจจับได้จึงเท่ากับความเร็วของสัญญาณนาฬิกาหารด้วย 24 ยกตัวอย่าง ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 สามารถใช้สัญญาณนาฬิกาได้สูงสุด 24 MHz ดังนั้นความเร็วสูงสุดของสัญญาณอินพุตที่ไทเมอร์/เคาน์เตอร์สามารถตรวจจับได้คือ 1 MHz

2.4.3 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1

ในการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีรีจิสเตอร์ที่ต้องเกี่ยวข้องเป็นพื้นฐานอยู่ 6 ตัว ดังมีรายละเอียดต่อไปนี้

2.4.3.1 รีจิสเตอร์ไทเมอร์

มีด้วยกัน 4 ตัว คือ TLO มีแอดเดรสอยู่ที่ 8AH, TH0 มีแอดเดรสอยู่ที่ 8CH, TL1 มีแอดเดรสอยู่ที่ 8BH และ TH1 มีแอดเดรสอยู่ที่ 8DH รีจิสเตอร์ทั้ง 4 ตัวจะอยู่ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ หรือ SFR รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้ร่วมกัน โดยจัดเป็นคู่คือ TLO กับ TH0 รวมกันเป็นรีจิสเตอร์ Timer0 ขนาด 16 บิต และ TL1 กับ TH1 รวมกันเป็นรีจิสเตอร์ Timer1 ขนาด 16 บิต โดยใน TLO และ TL1 เก็บข้อมูล 8 บิตล่าง ส่วน TH0 กับ TH1 เก็บข้อมูล 8 บิตบน รีจิสเตอร์ไทเมอร์ทั้ง 2 คู่ เมื่อนำมาใช้ร่วมกันจะสามารถเก็บค่าของการนับได้สูงสุด 65,536 หรือ FFFFH เมื่อนับถึงค่านี้อาจจะวนไปเริ่มนับ 0000H ใหม่ และเมื่อเกิดการนับรอบใหม่ บิต TFO หรือ TF1 ในรีจิสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์จะเกิดการเซต เพื่อแจ้งให้ทราบว่า นับเกินค่าสูงสุดแล้ว การเซตบิต TFO หรือ TF1 ขึ้นอยู่กับว่าเลือกใช้งานรีจิสเตอร์ไทเมอร์ตัวใด

● รีจิสเตอร์ควบคุมการทำงานของไทเมอร์/เคาน์เตอร์(Timer/Counter Control Register : TCON)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 88H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดดังนี้

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

TF1 (Timer 1 overflow) : เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจิสเตอร์ Timer 1 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางซอฟต์แวร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TR1 (Timer 1 run control bit) : ใช้ในการเปิดปิดการทำงานของไทมเมอร์ 1 (เอ็นเอเบิลหรือ ดิสเอเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทมเมอร์ 1 ทำงานต้อง เซตบิตนี้ให้เป็น “1”

TF0 (Timer 0 overflow) : เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจิสเตอร์ Timer 0 เกิดการนับเกินหรือ เกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางซอฟต์แวร์

TR1 (Timer 1 run control bit) : ใช้ในการเปิดปิดการทำงานของไทมเมอร์ 1 (เอ็นเอเบิลหรือ ดิสเอเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทมเมอร์ 1 ทำงานต้อง เซตบิตนี้ให้เป็น “1”

IE1 (External Interrupt 1 edge flag) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซต ได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จาก ภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 ได้ และจะสามารถเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น ซึ่งเป็นกระบวนการทางซอฟต์แวร์

IT1 (Interrupt 1 type control bit) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการ เลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุต อินเตอร์รัปต์ 1 (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการทางซอฟต์แวร์

“0” เลือกของขาสูงของสัญญาณ (falling edge)

“1” เลือกระดับลอจิกต่ำ (low level triggered)

IE0 (External Interrupt 0 edge flag) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซต ได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จาก ภายนอกที่ขาอินพุตอินเตอร์รัปต์ 0 ได้ และจะสามารถเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น ซึ่งเป็นกระบวนการทางซอฟต์แวร์

IT0 (Interrupt 0 type control bit) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการ เลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุต อินเตอร์รัปต์ 0 (INT0) การเซตและเคลียร์ทำได้ด้วยกระบวนการทางซอฟต์แวร์

“0” เลือกของขาสูงของสัญญาณ (falling edge)

“1” เลือกระดับลอจิกต่ำ (low level triggered)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

● รีจิสเตอร์เลือกโหมดการทำงานของไทมเมอร์/เคาน์เตอร์(Timer/Counter Mode Control

Register : TMOD)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 89H ในพื้นที่ของรีจิสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต แบ่งการทำงานออกเป็น 2 ส่วนคือ 4 บิตล่างใช้ในการเลือกโหมดการทำงานของไทมเมอร์ 0 และ 4 บิตบนใช้ในการเลือกโหมดการทำงานของไทมเมอร์ 1 ดังนั้นในการอธิบายการทำงานจะขออธิบายเพียงส่วนเดียวดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GATE	C/\bar{T}	M1	M0	GATE	C/\bar{T}	M1	M0
ไทมเมอร์ 1				ไทมเมอร์ 0			

GATE : ใช้เลือกลักษณะการควบคุมการทำงานของไทมเมอร์/เคาน์เตอร์

“0” ไทมเมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น “1” เรียกการควบคุมแบบนี้ว่าการควบคุมทางซอฟต์แวร์

“1” ไทมเมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น “1” และสถานะลอจิกที่ขาอินพุตอินเตอร์รัปต์ INT0 และ INT1 เป็น “1” เรียกการควบคุมแบบนี้ว่าการควบคุมทางฮาร์ดแวร์

C/\bar{T} (Timer or Counter selector) : ใช้เลือกลักษณะการทำงานของไทมเมอร์/เคาน์เตอร์

“0” เลือกให้ทำงานเป็นไทมเมอร์ โดยรับสัญญาณอินพุตจากสัญญาณสาฬิกาภายในไมโครคอนโทรลเลอร์

“1” เลือกให้ทำงานเป็นเคาน์เตอร์ โดยรับสัญญาณอินพุตทางขา T0 หรือ T1

M1,M0 (Mode selector) : ใช้เลือกโหมดการทำงานของไทมเมอร์/เคาน์เตอร์

“00” เลือกให้ทำงานในโหมดไทมเมอร์/เคาน์เตอร์ 13 บิต

“01” เลือกให้ทำงานในโหมดไทมเมอร์/เคาน์เตอร์ 16 บิต

“10” เลือกให้ทำงานในโหมดไทมเมอร์/เคาน์เตอร์ขนาด 8 บิตแบบตั้งค่าอัตโนมัติ

“11” สำหรับไทมเมอร์ 0 เลือกให้ทำงานในโหมดไทมเมอร์/เคาน์เตอร์แยกส่วน โดยแยกออกเป็นไทมเมอร์/เคาน์เตอร์ 8 บิต สองตัว รีจิสเตอร์ TLO จะได้รับการควบคุมการเปิดปิดจากบิต TR0 ในรีจิสเตอร์ TCON และรีจิสเตอร์ TH0 ซึ่งเป็นไทมเมอร์/เคาน์เตอร์ 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1 ในรีจิสเตอร์

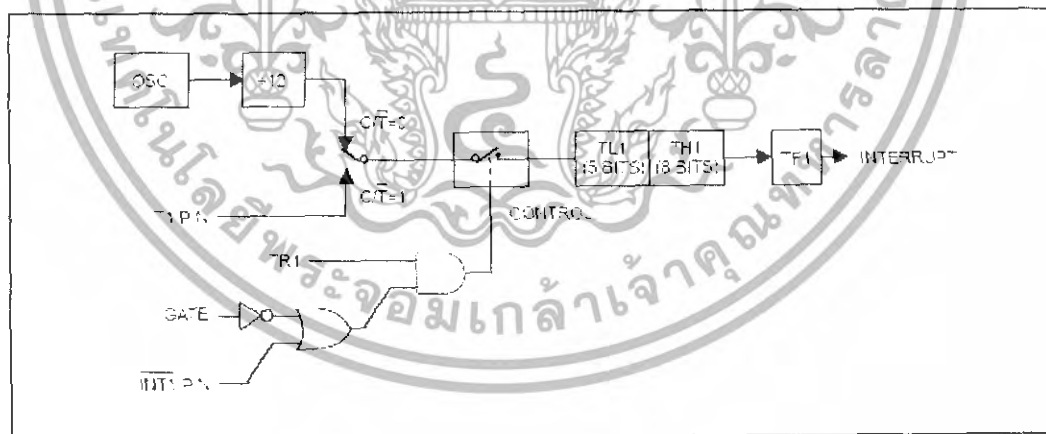
TCON ในกรณีของไทมเมอร์ 1 เป็นการสั่งให้ไทมเมอร์/เคาน์เตอร์ หยุดทำงาน (คิสเอเบิล)

2.4.4 โหมดการทำงานของไทมเมอร์/เคาน์เตอร์ 0 และ 1

ไทมเมอร์ 0 และ ไทมเมอร์ 1 สามารถเลือกโหมดการทำงานได้ 4 โหมด โดยการเลือกโหมดการทำงานของไทมเมอร์/เคาน์เตอร์ 0 และ 1 สามารถทำได้ทั้งรีจิสเตอร์ TCON และ TMOD ร่วมกัน โดย TCON ใช้ในการเอ็นเอเบิลหรือคิสเอเบิลไทมเมอร์ เคาน์เตอร์ ส่วน TMOD ใช้ในการเลือกโหมดและลักษณะการทำงาน ซึ่งอธิบายได้ดังนี้

2.4.4.1 การทำงานในโหมด 0 : ไทมเมอร์/เคาน์เตอร์ 13 บิต

ในโหมดนี้จะเป็นการกำหนดให้ใช้งานรีจิสเตอร์ TL1 เพียง 5 บิต และ TH1 ครบ 8 บิต โดย TL0 จะทำหน้าที่คล้ายกับเป็นปริสเกลเลอร์หาร 32 สัญญาณอินพุตสำหรับการนับจะเลือกจากสัญญาณนาฬิกาภายใน หรือภายนอกผ่านทางขา T1 ขึ้นอยู่กับการควบคุมของบิต C/T ในรีจิสเตอร์ TMOD, บิต TRI ในรีจิสเตอร์ TCON และสถานะของลอจิกที่ขาอินพุต INT1 เมื่อ TL1 นับครบ 32 คือ จาก 0 - 31 ก็จะส่งสัญญาณไปยัง TH1 เพื่อทำการเพิ่มค่า ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 13 บิต เมื่อทำการนับครบรอบก็จะทำการเซตบิต TFI ในรีจิสเตอร์ TCON

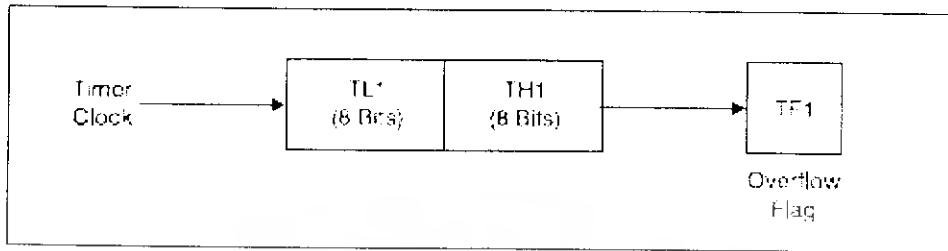


รูปที่ 2.23 ไคอะแกรมการทำงานในโหมด 0 ของไทมเมอร์/เคาน์เตอร์ 1

ส่วนการทำงานในโหมดนี้ของไทมเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่เปลี่ยนรีจิสเตอร์ และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทมเมอร์/เคาน์เตอร์ 0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

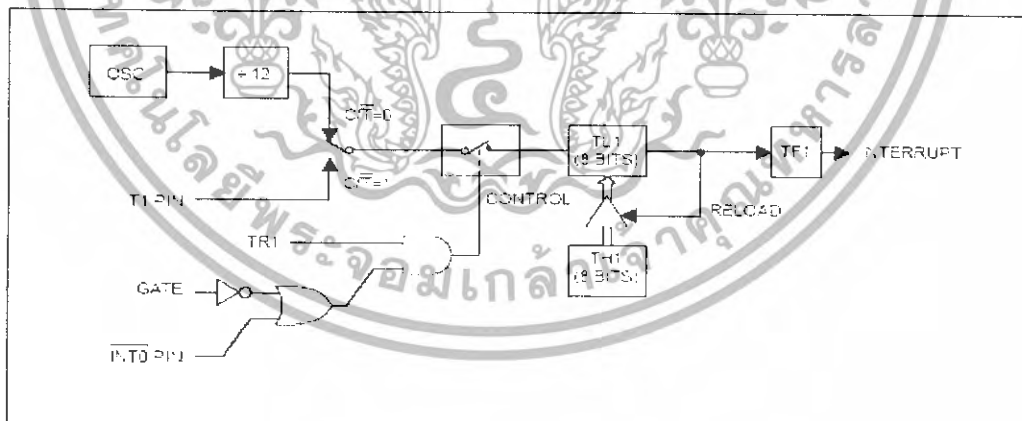
2.4.4.2 การทำงานในโหมด 1 : ไทเมอร์/เคาน์เตอร์ 16 บิต



รูปที่ 2.24 ไคอะแกรมการทำงานในโหมด 1 ของไทเมอร์/เคาน์เตอร์ 1

การทำงานในโหมดนี้จะใช้ไทเมอร์/เคาน์เตอร์ 1 ในการอธิบาย โดยในโหมดนี้จะคล้ายกับโหมด 0 แต่จะใช้งานรีจิสเตอร์ TL1 และ TH1 ครบ 8 บิต ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 16 บิต คือ 0000H – FFFFH เมื่อทำการนับครบรอบ ค่าของการนับเปลี่ยนจาก FFFFH เป็น 0000H ก็จะเซตบิต TF1 ในรีจิสเตอร์ TCON ส่วนการทำงานในโหมดนี้ของไทเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่เปลี่ยนรีจิสเตอร์ และ ขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์/เคาน์เตอร์ 0

2.4.4.3 การทำงานในโหมด 2 : ไทเมอร์/เคาน์เตอร์ 8 บิตแบบตั้งค่าอัตโนมัติ



รูปที่ 2.25 ไคอะแกรมการทำงานในโหมด 2 ของไทเมอร์/เคาน์เตอร์ 1

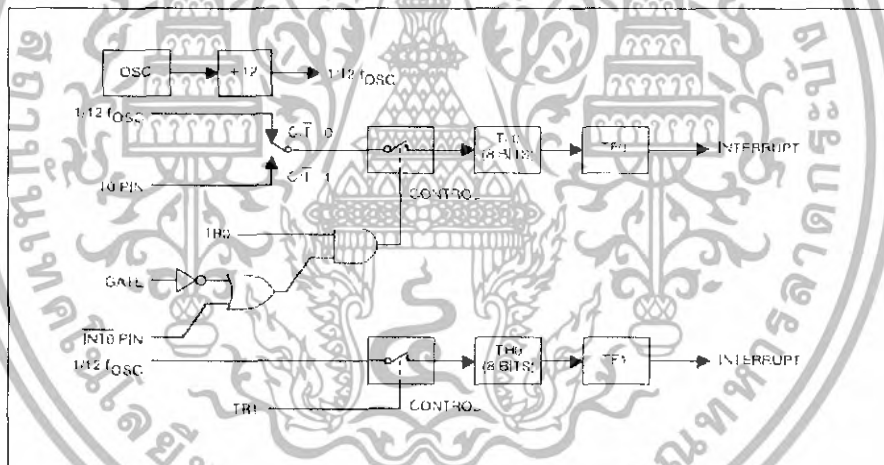
การทำงานในโหมดนี้จะใช้ไทเมอร์/เคาน์เตอร์ 1 ในการอธิบาย โดยในโหมดนี้จะแยกรีจิสเตอร์ไทเมอร์ออกเป็น 2 ตัว ตัวละ 8 บิต โดยรีจิสเตอร์ TL1 ทำหน้าที่เป็นตัวนับค่า ส่วน TH1 ใช้ในการเก็บค่าเริ่มต้นของการนับ เมื่อเริ่มต้นการทำงาน ค่าของรีจิสเตอร์ TH1 จะถูกส่งไปยัง

รีจิสเตอร์ TL1 ทำให้เมื่อเริ่มต้นการทำงานของรีจิสเตอร์ TL1 และ TH1 จะเหมือนกัน เมื่อ TL1 นับถึง FFH และจะเริ่มต้นการนับรอบใหม่ จะทำการเซตบิต TF1 พร้อมๆ กับการรับค่าการนับ เริ่มต้นจาก TH1 ใหม่โดยอัตโนมัติ หรือเรียกกระบวนการนี้ว่า รีโหลด(reload) แม้ว่าจะมีการส่งค่า เริ่มต้นไปยัง TL1 แล้วก็ตาม ค่าของข้อมูลในรีจิสเตอร์ TH1 ก็ยังคงเป็นค่าเดิม ไม่มีการเปลี่ยนแปลง จนกว่าจะมีการกำหนดค่าใหม่ด้วยกระบวนการทางซอฟต์แวร์

ส่วนการทำงานในโหมดนี้ของไทเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่ เปลี่ยนรีจิสเตอร์ และ ขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์/เคาน์เตอร์ 0

2.4.4.4 การทำงานในโหมด 3 : ไทเมอร์/เคาน์เตอร์แยกส่วนหรือไทเมอร์/เคาน์เตอร์ 8 บิต

ในโหมดนี้เป็นโหมดเดียวที่การทำงานของไทเมอร์ 0 และไทเมอร์ 1 ไม่เหมือนกัน ขออธิบายในส่วนของไทเมอร์ 1 ก่อน เมื่อเข้าสู่โหมดนี้ จะเป็นการสั่งให้ไทเมอร์/เคาน์เตอร์หยุดนับ ค่าของการนับก่อนหน้านี้จะถูกเก็บไว้ในรีจิสเตอร์ไทเมอร์ 1 มีลักษณะการทำงานเหมือนกับการ คิวสแควร์ไทเมอร์/เคาน์เตอร์ 1 ด้วยการเคลียร์บิต TR1 ในรีจิสเตอร์ TCON



รูปที่ 2.26 โค้ดแแกรมการทำงานในโหมด 3 ของไทเมอร์/เคาน์เตอร์ 1

ส่วนการทำงานของไทเมอร์/เคาน์เตอร์ 0 ในโหมดนี้จะแบ่งรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ 0 ออกเป็น 2 ตัว ตัวละ 8 บิต คือรีจิสเตอร์ TLO และ TH0 โดยแยกการทำงานออกจากกัน รีจิสเตอร์ TLO สามารถเลือกการทำงานได้เหมือนกับไทเมอร์/เคาน์เตอร์ตามปกติ ส่วนรีจิสเตอร์ TH0 สามารถทำงานในโหมดไทเมอร์ได้เพียงอย่างเดียว กล่าวคือสามารถรับสัญญาณอินพุตจากสัญญาณนาฬิกาภายในเพียงอย่างเดียวเท่านั้น แต่การแจ้งการนับเกินยังคงเหมือนเดิม หากแต่ TLO แจ้งผ่าน บิต TFO ในขณะที่ TH0 จะแจ้งผ่านทางบิต TF1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหมด	ฟังก์ชันของไทมเมอร์ 0	TMOD	
		การควบคุมจากภายใน	การควบคุมจากภายนอก
0	ไมเมอร์/เคาน์เตอร์ 13 บิต	00H	08H
1	ไทมเมอร์/เคาน์เตอร์ 16 บิต	01H	09H
2	8 บิตตั้งค่าอัตโนมัติ	02H	0AH
3	ไทมเมอร์/เคาน์เตอร์แยกส่วน	03H	0BH

หมายเหตุ

การควบคุมภายใน คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลงจาก "1" เป็น "0" ที่ขา $\overline{INT0}$ (P3.2) เมื่อ TR0 = "1"

ตารางที่ 2.3 แสดงข้อมูลจอร์จิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 0 ทำงานเป็นไทมเมอร์

โหมด	ฟังก์ชันของไทมเมอร์ 0	TMOD	
		การควบคุมจากภายใน	การควบคุมจากภายนอก
0	ไมเมอร์/เคาน์เตอร์ 13 บิต	04H	0CH
1	ไทมเมอร์/เคาน์เตอร์ 16 บิต	05H	0DH
2	8 บิตตั้งค่าอัตโนมัติ	06H	0EH
3	ไทมเมอร์/เคาน์เตอร์แยกส่วน	07H	0FH

หมายเหตุ

การควบคุมภายใน คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลงจาก "1" เป็น "0" ที่ขา $\overline{INT0}$ (P3.2) เมื่อ TR0 = "1"

ตารางที่ 2.4 แสดงข้อมูลจอร์จิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 0 ทำงานเป็นเคาน์เตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหมด	ฟังก์ชันของไทมเมอร์ 1	TMOD	
		การควบคุมจากภายใน	การควบคุมจากภายนอก
0	ไมเมอร์/เคาน์เตอร์ 13 บิต	00H	80H
1	ไทมเมอร์/เคาน์เตอร์ 16 บิต	10H	90H
2	8 บิตตั้งค่าอัตโนมัติ	20H	A0H
3	หยุดทำงาน	30H	B0H

หมายเหตุ

การควบคุมภายใน คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลงจาก "1" เป็น "0" ที่ขา $\overline{INT1}$ (P3.3) เมื่อ TR1 = "1"

ตารางที่ 2.5 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 1 ทำงานเป็นไทมเมอร์

โหมด	ฟังก์ชันของไทมเมอร์ 1	TMOD	
		การควบคุมจากภายใน	การควบคุมจากภายนอก
0	ไมเมอร์/เคาน์เตอร์ 13 บิต	40H	C0H
1	ไทมเมอร์/เคาน์เตอร์ 16 บิต	50H	D0H
2	8 บิตตั้งค่าอัตโนมัติ	60H	E0H
3	-	-	-

หมายเหตุ

การควบคุมภายใน คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก คือการควบคุมให้ไทมเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลงจาก "1" เป็น "0" ที่ขา $\overline{INT1}$ (P3.3) เมื่อ TR1 = "1"

ตารางที่ 2.6 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทมเมอร์ 1 ทำงานเป็นเคาน์เตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.5 ไทเมอร์/เคาน์เตอร์ 2 ในไมโครคอนโทรลเลอร์ MCS-51

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx ยังมี ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต ให้ใช้งานเพิ่มเติมอีก 1 ตัวคือ ไทเมอร์/เคาน์เตอร์ 2 (Timer/Counter 2) รีจิสเตอร์ฟังก์ชันพิเศษที่เกี่ยวข้องกับไทเมอร์/เคาน์เตอร์ 2 มีเพิ่มเติมอีก 5 ตัวคือรีจิสเตอร์ไทเมอร์ 2 ซึ่งประกอบด้วย TL2, TH2, รีจิสเตอร์ T2CON, รีจิสเตอร์ T2MOD มีแอดเดรสอยู่ที่ C9H และสุดท้ายคือรีจิสเตอร์แคปเจอร์ ซึ่งประกอบด้วย RCAP2L และ RCAP2H

ไทเมอร์/เคาน์เตอร์ 2 สามารถทำงานเป็นไทเมอร์ หรือตั้งเวลาและเคาน์เตอร์หรือตัวนับได้ เช่นเดียวกับไทเมอร์/เคาน์เตอร์ 0 และ 1 โดยการกำหนดที่บิต C/7 ในรีจิสเตอร์ T2CON เมื่อเทียบกับไทเมอร์/เคาน์เตอร์ 0 และ 1 ก็คือรีจิสเตอร์ TCON สำหรับโหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2 มีด้วยกัน 3 โหมดคือ โหมดแคปเจอร์หรือตรวจจับสัญญาณ (capture), โหมดตั้งค่าอัตโนมัติ (auto-reload) และโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมหรืออัตราบอด (baud rate generator)

2.4.5.1 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์ 2

ในการทำงานของไทเมอร์/เคาน์เตอร์ 2 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีรีจิสเตอร์ที่ต้องเกี่ยวข้องเป็นพื้นฐานอยู่ 5 ตัว ดังมีรายละเอียดต่อไปนี้

รีจิสเตอร์ไทเมอร์

มีด้วยกัน 2 ตัวคือ TL2 มีแอดเดรสอยู่ที่ CCH และ TH2 มีแอดเดรสอยู่ที่ CDH รีจิสเตอร์ทั้ง 2 ตัวจะอยู่ในพื้นที่ของฟังก์ชันพิเศษหรือ SFR รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้ รวมกันเป็นรีจิสเตอร์ไทเมอร์ 2 ขนาด 16 บิต โดยใน TL2 จะเก็บข้อมูล 8 บิตล่าง ส่วน TH2 เก็บข้อมูล 8 บิตบน รีจิสเตอร์ไทเมอร์ทั้งคู่เมื่อนำมาใช้งานร่วมกันจะสามารถเก็บค่าของการนับได้สูงสุด 65,536 ค่าหรือ FFFFH เมื่อนับถึงค่านี้แล้วก็จะวนไปเริ่มนับ 0000H ใหม่ และเมื่อเกิดการนับรอบใหม่ จะมีการเซตบิต TF2 ในรีจิสเตอร์ T2CON ที่ใช้ควบคุมการทำงานของไทเมอร์ เพื่อแจ้งให้ทราบว่าเกิดการนับเกินค่าสูงสุด

• รีจิสเตอร์ควบคุมการทำงานของไทมเมอร์/คาน์เตอร์ 2 (T2CON : Timer/Counter 2 Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ C8H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิตมีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/\bar{T}	$CP/\bar{RL}2$

TF2 (Timer 2 overflow flag) : จะเซตเมื่อค่าของรีจิสเตอร์ Timer2 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางซอฟต์แวร์ และจะไม่เกิดการเซตถ้าบิต RCLK หรือ TCLK เป็น "1"

EXF2 (Timer 2 external flag) : จะเกิดการเซตเมื่อเกิดการแคปเจอร์หรือรีโพลดขึ้นซึ่งเกิดจากการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขาอินพุต T2EX(ขา P 1.1) และ บิต EXEN2 เป็น "1" ในกรณีที่อินเทอร์รัปต์ในไทมเมอร์ 2 ได้รับการเอ็นเอเบิล และบิต EXF2 นี้เกิดเซต จะมีผลทำให้ซีพียูไปอ่านค่าแวกเตอร์ของการบริการอินเทอร์รัปต์ บิตนี้สามารถเคลียร์ด้วยกระบวนการทางซอฟต์แวร์เท่านั้น

RCLK (Receive clock flag) : ถ้าบิตนี้เกิดการเซต ทำให้วงจรถอดนุกรมภายในไมโครคอนโทรลเลอร์ ใช้พัลส์โอเวอร์โฟลวของไทมเมอร์ 2 เป็นสัญญาณนาฬิกาในการรับข้อมูลในโหมด 1 และ 2 ถ้าบิตนี้เป็น "0" จะใช้พัลส์จากไทมเมอร์ 1

TCLK (Transmit clock flag) : ถ้าบิตนี้เกิดการเซต ทำให้วงจรถอดนุกรมภายในไมโครคอนโทรลเลอร์ใช้พัลส์โอเวอร์โฟลวของไทมเมอร์ 2 เป็นสัญญาณนาฬิกาในการส่งข้อมูลในโหมด 1 และ 2 ถ้าบิตนี้เป็น "0" จะใช้พัลส์จากไทมเมอร์ 1

EXEN2(Timer 2 external enable flag) : เมื่อบิตนี้เซตเป็น "1" จะเป็นการเอ็นเอเบิลการแคปเจอร์หรือรีโพลดที่ขา T2EX ให้เกิดขึ้นได้ภายใต้เงื่อนไขว่า ไทมเมอร์ 2 ต้องไม่ถูกนำไปใช้ในการสื่อสารพอร์ตต่อนุกรมหรือบิต RCLK หรือ TCLK เป็น "1" ถ้าบิตนี้เป็น "0" ไทมเมอร์ 2 จะไม่สนใจเหตุการณ์ที่ขา T2EX

TR2 : บิตนี้จะใช้ในการควบคุมการเริ่มต้นและหยุดการทำงานของไทมเมอร์ 2 เป็นการควบคุมทางซอฟต์แวร์

"0" หยุดการทำงานของไทมเมอร์ 2 (STOP)

"1" เริ่มต้นการทำงานของไทมเมอร์ 2 (START)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

C/\bar{T} (Capture or Counter selector) : ใช้เลือกลักษณะการทำงานของไทเมอร์ เคา์เตอร์

“0” เลือกให้ทำงานเป็นไทเมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์

“1” เลือกให้ทำงานเป็นเคา์เตอร์ โดยรับสัญญาณอินพุตทางขา T2(P1.0)

$CP/RL2$ (Capture/Reload flag) : เป็นบิตแสดงสถานะการทำงานของไทเมอร์ 2

“0” มีการรีโหลดค่าของการนับแบบอัตโนมัติเกิดขึ้น เมื่อไทเมอร์ 2 เกิดโอเวอร์โฟลว หรือเกิดการเปลี่ยนระดับลอจิกแบบลบ(เปลี่ยนจาก “1” เป็น “0”)ที่ขา T2EX ในกรณีที่บิต EXEN2 เป็น “1” แต่ถ้าหากบิต RCLK หรือ TCLK เป็น “1” จะยอมให้การรีโหลดแบบอัตโนมัติเกิดขึ้นเมื่อไทเมอร์ 2 เกิดโอเวอร์โฟลวเท่านั้น

“1” แสดงว่ามีการแคปเจอร์เกิดขึ้น เมื่อเกิดการเปลี่ยนระดับลอจิกแบบลบ (เปลี่ยนจาก “1” เป็น “0”)ที่ขา T2EX ในกรณีที่บิต EXEN2 เป็น “1”

- รีจิสเตอร์เลือกโหมดการทำงานของไทเมอร์ เคา์เตอร์ 2 (T2MOD : Timer/Counter 2 Mode Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ C9H ในพื้นที่ของรีจิสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต มีบิตสำหรับกำหนดการทำงานเพียง 2 บิต คือ

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	-	-	-	-	T2OE	DCEN

T2OE(Timer 2 Output Enable bit) : ใช้เอ็นเอเบิลเอาต์พุตของไทเมอร์ 2

“0” ดิสเอเบิล

“1” เอ็นเอเบิล

DCEN(Timer 2 Counter Enable bit) : กำหนดให้ไทเมอร์ 2 ทำงานเป็นตัวนับหรือเคา์เตอร์แบบขึ้นและลง

“0” ดิสเอเบิล

“1” เอ็นเอเบิล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

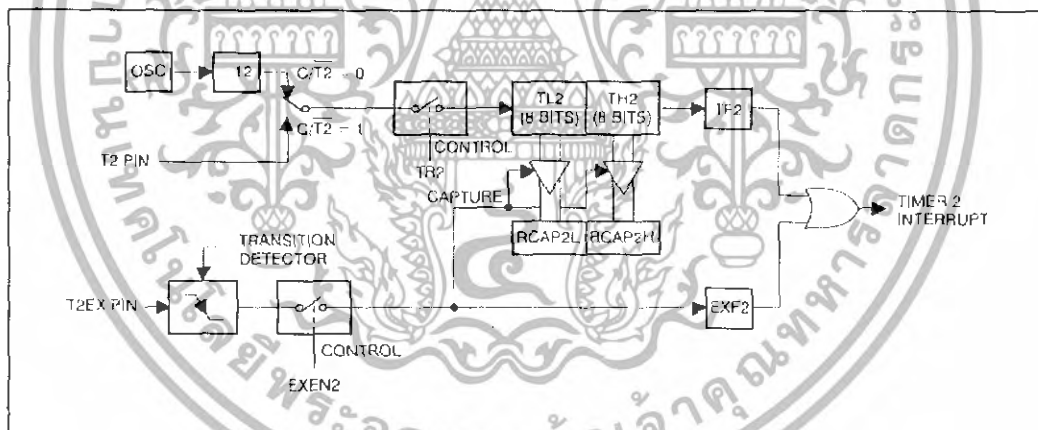
2.4.5.2 โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2

ไทเมอร์/เคาน์เตอร์ 2 สามารถเลือกโหมดการทำงานได้ 3 โหมด โดยการกำหนดที่บิต RCLK+TCLK, CP/RL2 และ TR2 ในรีจิสเตอร์ T2CON ดังแสดงรายละเอียดในตาราง..... โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2

RCLK+TCLK	CP/RL2	TR2	โหมด
0	0	1	16 บิตแบบตั้งค่าการนับอัตโนมัติหรือรีโหลด
0	1	1	แคปเจอร์หรือตรวจจับสัญญาณ
1	x	1	กำหนดอัตราบอด
x	x	0	คิสเอบีล

ตารางที่ 2.7 การเลือกโหมดทำงานของไทเมอร์/เคาน์เตอร์ 2

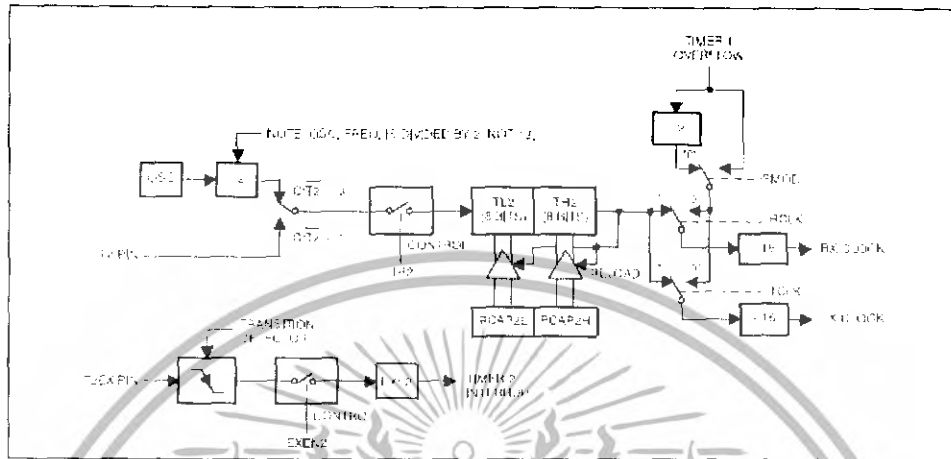
2.4.5.2.1 การทำงานในโหมดแคปเจอร์ หรือตรวจจับสัญญาณ(capture)



รูปที่ 2.27 ไดอะแกรมการทำงานในโหมดแคปเจอร์ของไทเมอร์/เคาน์เตอร์ 2

การทำงานในโหมดนี้จะเกิดขึ้นได้อย่างสมบูรณ์เมื่อกำหนดให้บิต EXEN2 เป็น “1” หลังจาก que เลือกให้ไทเมอร์ 2 ทำงานในโหมดแคปเจอร์แล้ว เมื่อเลือกให้ทำงานในโหมดนี้ขาอินพุตที่ทำการตรวจจับคือขาอินพุต T2EX ซึ่งตรงกับขา P1.1 ที่ขา T2EX จะมีวงจรตรวจจับการเปลี่ยนแปลงของระดับลอจิกจาก “1” เป็น “0” รีจิสเตอร์ TL2 และ TH2 จะทำการนับค่าเพิ่มขึ้นไปเรื่อยๆ จะกว่าที่ขา T2EX จะตรวจจับการเปลี่ยนแปลงได้ เมื่อตรวจจับได้ ค่าของรีจิสเตอร์ TL2 และ TH2 จะถูกส่ง

เกิดการเซต แต่จะไม่มีกรเรียกค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H เมื่อเป็นเช่นนี้ T2EX จึงสามารถใช้เป็นขาอินพุตสำหรับการอินเตอร์รัปต์ได้เป็นกรณีพิเศษ



รูปที่ 2.29 โค้ดแอมการทำงานในโหมดกำเนิดอัตราบอดในการสื่อสารข้อมูลผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ 2

เมื่อไมโครคอนโทรลเลอร์ 2 ทำงาน (บิต TR2 เป็น "1") จะถูกกำหนดให้ทำงานเป็นไมโครคอนโทรลเลอร์ จึงไม่สามารถอ่านหรือเขียนค่าที่รีจิสเตอร์ TL2 และ TH2 ได้ ค่าของรีจิสเตอร์ไมโครคอนโทรลเลอร์จะเพิ่มขึ้นทุกๆ สเตต (1 สเตตเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา) ค่าการนับของรีจิสเตอร์นำมาใช้สร้างสัญญาณนาฬิกาของการสื่อสารข้อมูลผ่านพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหมคการทำงานของ ไทเมอร์/เคาน์เตอร์ 2	T2CON	
	การควบคุมจากภายใน	การควบคุมจากภายนอก
16 บิตแบบตั้งค่าการนับอัตราโนมิติหรือรีโหลด	00H	08H
แคปเจอร์หรือตรวจจับสัญญาณ	01H	09H
กำหนดอัตราบอด โดยอัตรารับและส่งเท่ากัน	34H	36H
รับข้อมูลเท่านั้น	24H	26H
ส่งข้อมูลเท่านั้น	14H	16H

หมายเหตุ

การควบคุมจากภายใน คือ การแคปเจอร์และรีโหลดเกิดขึ้นจากการ โอเวอร์โพลของไทเมอร์

การควบคุมจากภายนอก คือ การแคปเจอร์และรีโหลดเกิดขึ้นจากการ โอเวอร์โพลของไทเมอร์และการเปลี่ยนระดับลอจิก “1” เป็น “0” ที่ขา T2EX ยกเว้นในกรณีที่กำหนดในการทำงานในการกำหนดอัตราบอดสำหรับพอร์ตอนุกรมในไมโครคอนโทรลเลอร์

ตารางที่ 2.8 แสดงข้อมูลของรีจิสเตอร์ T2CON เพื่อกำหนดให้ไทเมอร์/เคาน์เตอร์ 2 ทำงานเป็นไทเมอร์

โหมคการทำงานของ ไทเมอร์/เคาน์เตอร์ 2	T2CON	
	การควบคุมจากภายใน	การควบคุมจากภายนอก
16 บิตแบบตั้งค่าการนับอัตราโนมิติหรือรีโหลด	02H	0AH
แคปเจอร์หรือตรวจจับสัญญาณแบบ 16 บิต	03H	0BH

หมายเหตุ

การควบคุมจากภายใน คือ การแคปเจอร์และรีโหลดเกิดขึ้นจากการ โอเวอร์โพลของไทเมอร์

การควบคุมจากภายนอก คือ การแคปเจอร์และรีโหลดเกิดขึ้นจากการ โอเวอร์โพลของไทเมอร์และการเปลี่ยนระดับลอจิก “1” เป็น “0” ที่ขา T2EX ยกเว้นในกรณีที่กำหนดในการทำงานในการกำหนดอัตราบอดสำหรับพอร์ตอนุกรมในไมโครคอนโทรลเลอร์

ตารางที่ 2.9 แสดงข้อมูลของรีจิสเตอร์ T2CON เพื่อกำหนดให้ไทเมอร์/เคาน์เตอร์ 2 ทำงานเป็นเคาน์เตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.56 วอตซ์ด็อกไทมเมอร์(Watchdog Timer : WDT)

นอกเหนือไปจากไทมเมอร์/เคาน์เตอร์ทั้งสามตัวแล้วสำหรับในอนุกรม AT89Sxx ไม่ว่าจะเป็นเบอร์ AT89S8252, AT89LS8252 และเบอร์ AT89S53 ยังมีวงจรไทมเมอร์อีกแบบหนึ่งเพิ่มเติมเพื่อให้สามารถนำไปใช้สร้างสัญญาณรีเซ็ตในการที่ซีพียูภายในไมโครคอนโทรลเลอร์เกิดภาวะหยุดทำงานกะทันหันหรือแฮงก์(hang) นั่นคือ วอตซ์ด็อกไทมเมอร์(Watchdog Timer : WDT) โดยวอตซ์ด็อกไทมเมอร์ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชอนุกรม AT89Sxx จะเป็นวอตซ์ด็อกไทมเมอร์แบบโปรแกรมได้ โดยการกำหนดค่าของบิต PS0-PS2 ในรีจิสเตอร์ WDCON สำหรับเบอร์ AT89S8252 และรีจิสเตอร์ WCON สำหรับเบอร์ AT89S53

วอตซ์ด็อกไทมเมอร์จะได้รับการรีเซ็ตโดยอัตโนมัติเมื่อเริ่มต้นจ่ายไฟหรือเมื่อเกิดเพาเวอร์อนรีเซตและเมื่อไมโครคอนโทรลเลอร์เข้าสู่การทำงานในโหมดประหยัดพลังงานแบบลดพลังงานหรือเพาเวอร์ดาวน์ สำหรับการอื่นเอเบิลสามารถกระทำได้โดยกระบวนการทางซอฟต์แวร์ โดยการเซตบิต WDTEN ในรีจิสเตอร์ WDCON สำหรับเบอร์ AT89S8252 และรีจิสเตอร์ WCON สำหรับเบอร์ AT89S53

การทำงานของวอตซ์ด็อกไทมเมอร์จะเริ่มต้นขึ้นเมื่อมีการกำหนดคาบเวลาในการทำงานของวอตซ์ด็อกไทมเมอร์และมีการเอเบิลให้วอตซ์ด็อกไทมเมอร์เริ่มทำงาน วงจรนับภายในวอตซ์ด็อกไทมเมอร์จะนับค่าเวลาไปตามปกติ หากซีพียูทำงานอยู่เป็นปกติจะมีการส่งสัญญาณมารีเซ็ตหรือเคลียร์ค่าในวอตซ์ด็อกไทมเมอร์อยู่เป็นระยะเพื่อไม่ให้วอตซ์ด็อกไทมเมอร์ทำงานจนถึงค่าเวลาที่กำหนดหรือเกิดการไทม์เอาต์ (time out) ขึ้น แต่ถ้าหากซีพียูเกิดภาวะหยุดทำงานจนกะทันหันไม่ว่าจะด้วยสาเหตุใด ยกเว้นการรีเซ็ตและการเข้าสู่โหมดประหยัดพลังงานแบบลดพลังงานหรือเพาเวอร์ดาวน์ก็จะไม่มีสัญญาณมารีเซ็ตหรือเคลียร์ค่าในวอตซ์ด็อกไทมเมอร์ทำให้ในที่สุดวอตซ์ด็อกไทมเมอร์ก็จะทำงานถึงค่าเวลาที่กำหนดเกิดการไทม์เอาต์ วอตซ์ด็อกไทมเมอร์จะทำการสร้างสัญญาณรีเซ็ตส่งไปยังซีพียู เพื่อทำการรีเซ็ตซีพียู ส่งผลให้ซีพียูสามารถกลับมาเริ่มต้นทำงานใหม่ได้อีกครั้ง โดยที่ไม่จำเป็นต้องทำการรีเซ็ตซีพียูจากภายนอกหรือจ่ายไฟเลี้ยงใหม่ให้แก่ไมโครคอนโทรลเลอร์

ในไมโครคอนโทรลเลอร์ MCS-51 เบอร์พื้นฐานเช่น 8031, 8032, 8051 หรือแบบแฟลชเบอร์ AT89C51, AT89C52, AT89C55 จะไม่มีวอตซ์ด็อกไทมเมอร์อยู่ภายใน เมื่อมีความจำเป็นต้องตรวจสอบการทำงานของซีพียูจึงต้องใช้เทคนิคในการเขียนโปรแกรมเข้าช่วย โกรการใช้ไทมเมอร์ที่มีอยู่ในไมโครคอนโทรลเลอร์เบอร์พื้นฐานเหล่านั้น หรือใช้ไอซีที่ทำหน้าที่เป็นวงจรวอตซ์ด็อกโดยเฉพาะต่อเข้ากับขา RST ของไมโครคอนโทรลเลอร์ เพื่อสร้างสัญญาณรีเซ็ตขึ้น ในการที่ไมโครคอนโทรลเลอร์เกิดภาวะหยุดทำงานกะทันหันโดยไม่ทราบสาเหตุ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- รีจิสเตอร์ควบคุมการทำงานของวอตช์ด็อกไทมเมอร์(Watchdog & Memory Control register : WMCON และ Watchdog control register : WCON)

ในการทำงานของวอตช์ด็อกไทมเมอร์ในไมโครคอนโทรลเลอร์ MCS-51แบบแฟลชอนุกรม AT89Sxx มีรีจิสเตอร์ที่ใช้ในการควบคุมการทำงาน 1 ตัวคือ WDCON สำหรับเบอร์ AT89S8252 และรีจิสเตอร์ WCON สำหรับเบอร์ AT89S53 โดยมีหน้าที่ในการทำงานหลักเหมือนกันทุกประการ ส่วนแตกต่างกันมีเพียงจุดเดียวคือในเบอร์ AT89S8252 มีหน่วยความจำข้อมูลแบบอีอีพรอมอยู่ จึงต้องมีการควบคุมการทำงานเพิ่มเติมเข้ามา จึงใช้บิต 3 และ 4 ในรีจิสเตอร์ WMCON มาจัดการหน่วยความจำข้อมูลส่วนนี้ สำหรับบิตอื่นๆ จะเหมือนกันทุกประการ ดังนั้นในการอธิบายรายละเอียดของรีจิสเตอร์ควบคุมการทำงานของวอตช์ด็อกไทมเมอร์จะอธิบายรวมไปในคราวเดียวกัน โดยจะเพิ่มเติมรายละเอียดของบิตที่ใช้ควบคุมหน่วยความจำข้อมูลแบบอีอีพรอมเข้าไปด้วย

รีจิสเตอร์ควบคุมการทำงานของวอตช์ด็อกทั้ง WMCON และ WCON เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 96H ในพื้นที่ของ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0	
PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN	AT89S8252
PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN	AT89S53

PS2-PS0 (Prescaler bits) : ใช้ในการเลือกค่าเวลาให้แก่วอตช์ด็อกไทมเมอร์ มีรายละเอียดดังนี้

บิตปริสเกลเลอร์			ค่าเวลาของวอตช์ไทมเมอร์
PS2	PS1	PS0	
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1,024 ms
1	1	1	2,048 ms

ตารางที่ 2.10 การเลือกเวลาให้แก่วอตช์ด็อก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

EEMWE (EEPROM Data Memory Write Enable bit) : บิตนี้เป็นบิต 4 ของรีจิสเตอร์ WMCN ใช้ในการเอ็นเอเบิลการเขียนข้อมูลลงในหน่วยความจำข้อมูลแบบอีอีพรอมภายในไมโครคอนโทรลเลอร์เบอร์ AT89S8252 เมื่อต้องการเขียนข้อมูลต้องทำการเซตบิตนี้ก่อน หลังจากการเขียนข้อมูลเสร็จสิ้นลง ต้องทำการเคลียร์บิตนี้เสมอ สำหรับในรีจิสเตอร์ WCON ในไมโครคอนโทรลเลอร์ AT89S53 บิตนี้จะไม่มีการใช้งานต้องกำหนดให้เป็น “0”

EEMEM (Internal EEPROM Access Enable bit) : บิตนี้เป็นบิต 3 ของรีจิสเตอร์ WMCN ใช้ในการเอ็นเอเบิลการเข้าถึงหน่วยความจำข้อมูลแบบอีอีพรอมภายใน AT89S8252 แทนที่จะใช้หน่วยความจำข้อมูลภายนอก เมื่อต้องการเข้าถึงข้อมูลในหน่วยความจำส่วนนี้ โดยใช้คำสั่ง MOVX และรีจิสเตอร์ DPTR ต้องทำการเซตบิตนี้ก่อน ถ้าหากบิตนี้เป็น “0” จะเป็นการกำหนดให้ไมโครคอนโทรลเลอร์ใช้คำสั่ง MOVX และรีจิสเตอร์ DPTR ติดต่อกับหน่วยความจำข้อมูลภายนอกไมโครคอนโทรลเลอร์ สำหรับในรีจิสเตอร์ WCON ของ AT89S53 บิตนี้ไม่ได้ใช้งานกำหนดให้เป็น “0”

DPS (Data Pointer Register Select) : เนื่องจากในไมโครคอนโทรลเลอร์อนุกรม AT89Sxx จะมีรีจิสเตอร์สำหรับชี้ข้อมูลหรือ DPTR 2 ตัวคือ DPTR0 และ DPTR1 บิตนี้จึงมีหน้าที่ใช้ในการเลือกรีจิสเตอร์ DPTR ถ้าเป็น “0” จะเป็นการเลือกรีจิสเตอร์ DPTR 0 (ซึ่งก็คือ DP0L และ DP0H) ในกรณีที่บิตนี้เป็น “1” จะเป็นการเลือกรีจิสเตอร์ DPTR 1 (ซึ่งก็คือ DP1L และ DP1H)

WDTRST/RDY/BSY (Watchdog Timer Reset and EEPROM Ready/Busy flag) : บิตนี้มีความเกี่ยวข้องกับการทำงาน 2 ส่วนคือ การรีเซตของวอตช์ดีด็อกไทเมอร์และใช้เป็นบิตแสดงสถานะของหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์ AT89S8252 ในกรณีวอตช์ดีด็อกไทเมอร์ บิตนี้จะเซตโดยผู้ใช้งานด้วยกระบวนการทางซอฟต์แวร์ เป็นการสั่งให้วอตช์ดีด็อกไทเมอร์สร้างสัญญาณรีเซตส่งออกไปในกรณีที่วอตช์ดีด็อกไทเมอร์เกิดไทม์เอาต์ หลังจากที่ส่งสัญญาณรีเซตออกไปแล้ว บิตนี้จะเคลียร์ตัวเองเป็น “0” โดยอัตโนมัติในไซเคิลของการทำงานถัดไป ดังนั้นบิตนี้จึงเขียนได้เพียงอย่างเดียว(write-only)

ในกรณีของหน่วยความจำข้อมูลอีอีพรอม จะใช้บิตนี้ในการป้องกันการเขียนข้อมูลลงในหน่วยความจำ ถ้าบิตนี้เป็น “1” หน่วยความจำข้อมูลอีอีพรอมจะสามารถอ่านได้เพียงอย่างเดียว เมื่อมีความต้องการเขียนข้อมูลหลังจากเอ็นเอเบิลการเขียนที่บิต EEMWE แล้ว ในทำการเคลียร์บิตนี้เป็น “0” หลังจากทำการเขียนข้อมูลเสร็จสิ้นลง บิตนี้จะเซตเป็น “1” โดยอัตโนมัติ เป็นการแจ้งให้ทราบว่า การเขียนข้อมูลเสร็จสิ้นลงแล้ว

สำหรับใน AT89S53 ใช้งานบิตนี้ในการสร้างสัญญาณรีเซตของวอตช์ไทเมอร์เท่านั้น

WDTEN (Watchdog Timer Enable bit) : ใช้ในเอ็นเอเบิลการทำงานของวอตช์ไทมเมอร์

“0” คิสเอเบิลวอตช์ดีค็อกไทมเมอร์

“1” เอ็นเอเบิลวอตช์ดีค็อกไทมเมอร์

จากรายละเอียดของไทมเมอร์ เคนเตอร์ทั้งหมด จะเห็นได้ว่า ในการสร้างระบบควบคุมที่ดี ไทมเมอร์/เคนเตอร์เป็นสิ่งที่สำคัญและจำเป็นต้องมี ไม่ว่าจะเป็นการสร้างอัตราบอดสำหรับการสื่อสารข้อมูลผ่านพอร์ตอนุกรมเพื่อเชื่อมต่อกับคอมพิวเตอร์ หรือเพื่อสร้างฐานเวลา และยังสามารถใช้ในการวัดคาบเวลาและความถี่ของสัญญาณได้อีกด้วย นอกจากนี้ในไมโครคอนโทรลเลอร์อนุกรม AT89Sxx ที่มีวอตช์ดีค็อกไทมเมอร์ในตัวยังช่วยให้ระบบควบคุมมีเสถียรภาพมากขึ้น การมีวอตช์ดีค็อกไทมเมอร์ในตัวจึงได้ว่าเป็นคุณสมบัติขั้นพื้นฐานของไมโครคอนโทรลเลอร์สมัยใหม่ไปแล้วจึงส่งผลให้ไมโครคอนโทรลเลอร์ MCS-51 มีขีดความสามารถที่ทัดเทียมกับไมโครคอนโทรลเลอร์ยุคใหม่ และยังคงเป็นที่นิยมใช้งานกันต่อไป

2.5 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์(Full-duplex) 1 ชุด โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ เป็นแบบอะซิงโครนัสปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้วโดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

2.5.1 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรม

ในการทำงานของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว ดังนี้

- รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม(SBUF : Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ มีขนาด 8 บิต แบ่งออกเป็น 2 ส่วนคือรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรับข้อมูล (receiver buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม(SCON : Serial port Control Register)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์พิเศษ สามารถเข้าถึงได้
ในระดับบิต มีรายละเอียดการทำงานดังนี้

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

SM0,SM1 (Serial port mode bit 0-1): ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรม
ภายในไมโครคอนโทรลเลอร์ ดังมีรายละเอียดดังนี้

“00” โหมด0 Shift register

“01” โหมด1 8-bit UART

“10” โหมด2 9-bit UART

“11” โหมด3 9-bit UART

SM2 : ใช้ในการเอ็นเอเบิลการสื่อสารในแบบมัลติโพรเซสเซอร์(multiprocesser) ในการ
ทำงานของโหมด 2 และ3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ ถ้าบิตนี้เป็น “1” บิต RI จะไม่
แอกทีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น “0” ในการทำงานในโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอกทีฟถ้า
ยังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้จะไม่มีการใช้งาน

REN (Enable serial reception) : ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการ
เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซตบิตนี้ให้เป็น “1”

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงาน โหมด 2 และ 3 ของ
พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมใน
ไมโครคอนโทรลเลอร์ แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น “0” ข้อมูลที่
บิต RB8 คือข้อมูลของบิตหยุด สำหรับในการทำงานในโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้
สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจาก
พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการ
ส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานในโหมด 0 ส่วนในการทำงานในโหมดอื่น บิตนี้จะ
เซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้แสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ต
อนุกรม สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานในโหมด 0 ส่วนในการทำงานในโหมดอื่น บิตนี้จะเซตเมื่อสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

2.5.2 โหมดการทำงานของพอร์ตอนุกรม

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์สามารถเลือกการทำงานได้ถึง 4 โหมดคือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะซีฟรียูนิค
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

2.5.2.1 การทำงานในโหมด 0 ของวงจรถูกอนุกรม

มีไคอะแกรมการทำงานและไคอะแกรมเวลาแสดงในรูป ... ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล(Shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

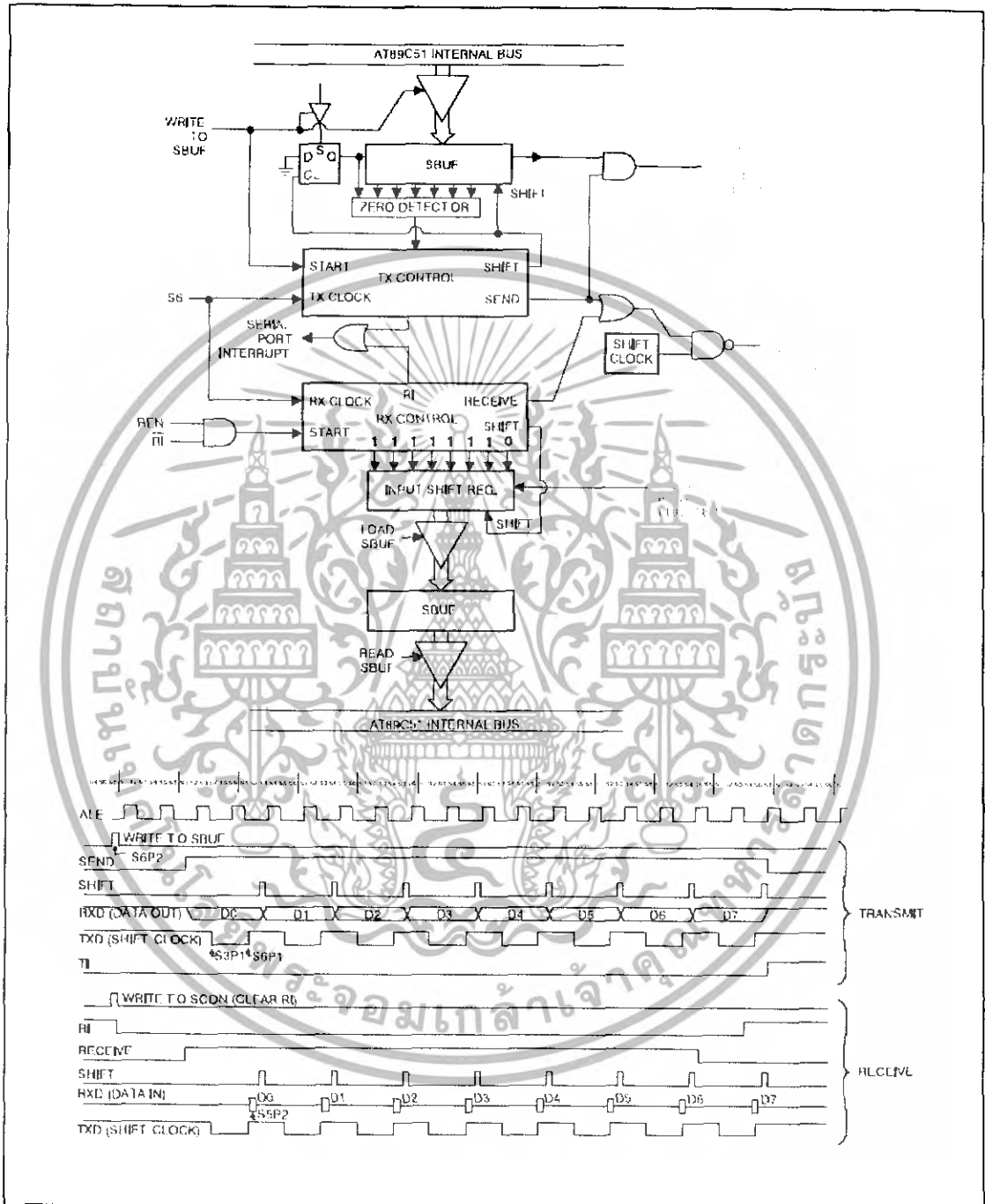
เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล แอคทีฟเป็น "1" ที่สแตต 6 เฟส 2(S6P2)ของแมคซินไซเคิล ส่งมายังวงจรถวลคุมการส่ง(TX control) ทำให้วงจรถวลคุมเริ่มต้นส่งข้อมูล สัญญาณ SEND จะกลายเป็น "1" ตลอดเวลาการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P3.0 หรือขา TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบขาลงของสัญญาณที่สแตต 3 เฟส 1 และมีของขาขึ้นของสัญญาณที่สแตต 6 เฟส 1 ของแต่ละแมคซินไซเคิลในกระบวนการส่งข้อมูลจนกระทั่งเมื่อส่งข้อมูลครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซต เป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะมีการอินเตอร์รัปต์ขึ้นในระบบ

เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูล เริ่มต้นด้วยการเซต REN ให้เป็น "1" และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อน ที่สแตต 6 เฟส 2 ของแมคซินไซเคิลถัดไป วงจรถวลคุมการรับ(RX control)

จะทำการเขียนข้อมูล 1111110 ไปยังชิพรีจิสเตอร์สำหรับรับข้อมูล และทำการแอกทีฟสัญญาณ RECEIVE ให้เป็น "1" ในสัญญาณนาฬิกาถูกลัดไป



รูปที่ 2.30 โค้ดอะแกรมการทำงานในโหมด 0 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อสัญญาณ RECEIVE แอคทีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล(shift register) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วงสแตต 3 เฟส 1 ถึง สแตต 6 เฟส 1 ของแต่ละแมคซินไซเคิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นในช่วงสแตต 5 เฟส 2 ในแมคซินไซเคิลเดียวกันกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเทอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะมีการอินเทอร์รัปต์ขึ้นภายในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ RECEIVE จะกลายเป็น “0” จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ จะใช้ในการเชื่อมต่อกับไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต คำไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรลเลอร์เองมีพอร์ตอยู่ค่อนข้างมาก และติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก

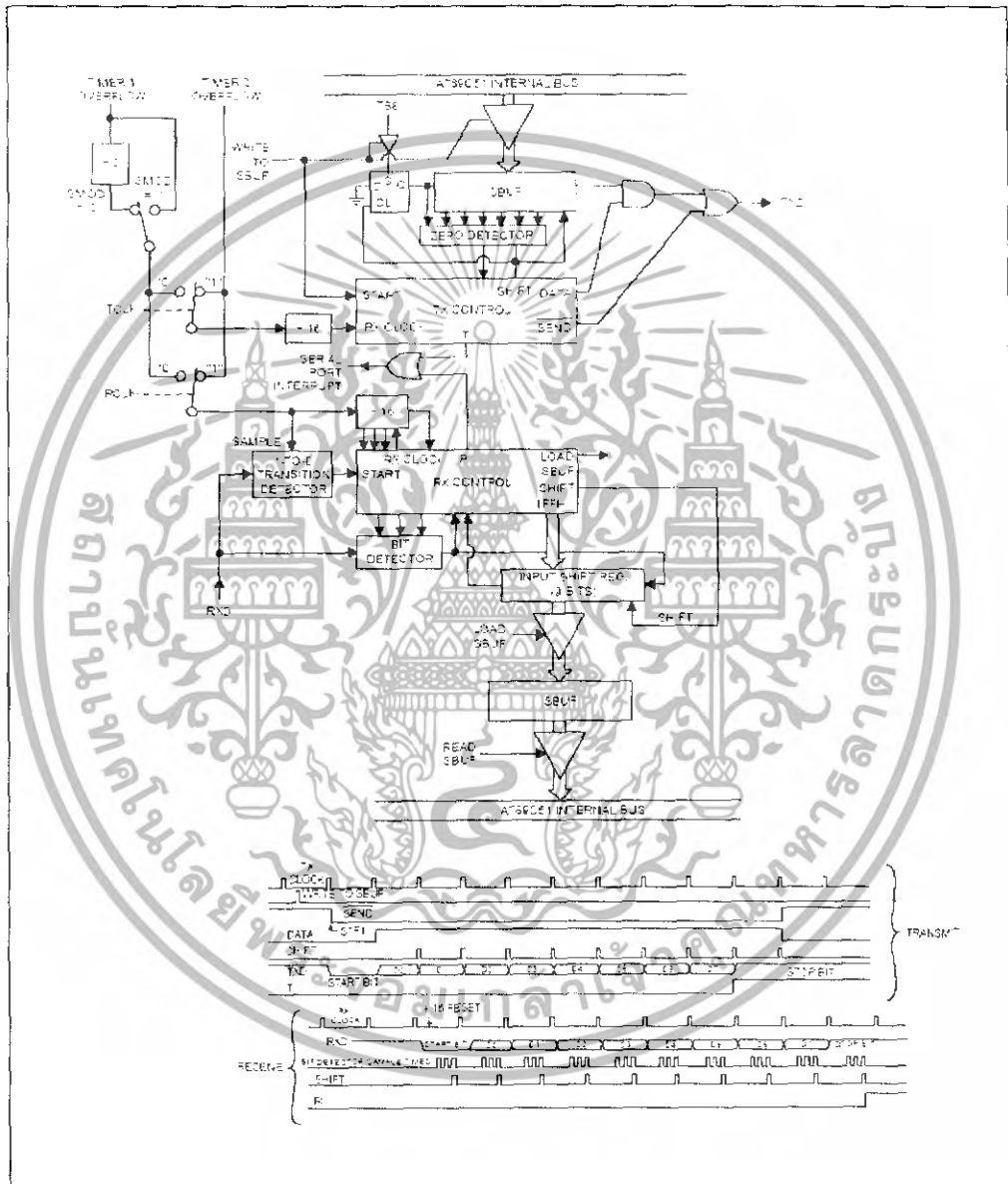
2.5.2.2 การทำงานในโหมด 1 ของวงจรถ่ายพอร์ตอนุกรม

มีไอซีแแกรมแสดงในรูปที่ ในโหมดนี้ใช้ในการนับส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตประกอบด้วยบิตเริ่มต้น(มีค่าเป็น “0”) 1 บิต บิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิต LSB ก่อน และบิตหยุดหรือบิตปิดท้าย(มีค่าเป็น “1”) ในการรับข้อมูลบิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทมเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลวของไทมเมอร์ 1 หรือไทมเมอร์ 2 ในการกำหนดอัตราบอดได้

กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอคทีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรถวนคุมการส่ง จากนั้นวงจรถวนคุมจะทำการแอคทีฟสัญญาณ SEND ที่สแตต 1 เฟส 1 ของแมคซินไซเคิลต่อมา โดยสัญญาณ SEND จะเป็น “0” ตลอดการส่งข้อมูล เมื่อสัญญาณ SEND แอคทีฟ จะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมคซินไซเคิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งจะถูกเรียงออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูล ในทุกๆบิตข้อมูลที่ทำการส่งออกไป จะเกิดสัญญาณพัลส์ เลื่อน(shift) ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลใช้สัญญาณนาฬิกาการส่ง(TX clock)เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จากไทมเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หยุดหรือบิตปิดท้าย 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว ต้องทำการเคลียร์บิต TX ก่อนเป็นอันดับแรก เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้



รูปที่ 2.31 โค้ดแอมแกรมการทำงานในโหมด 1 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

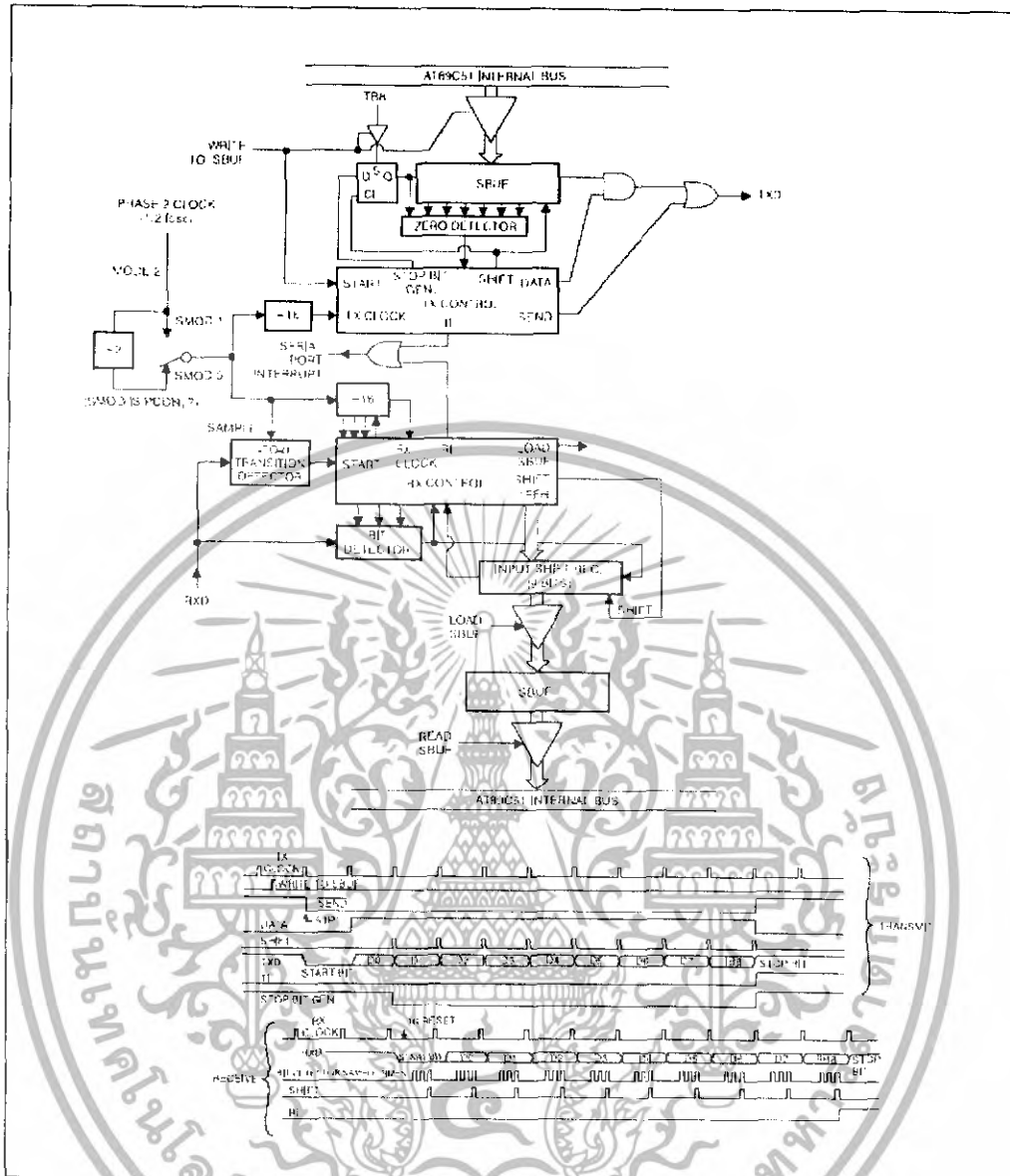
ด้านการรับข้อมูล จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก “0” เป็น “1” ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ $1/16$ เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดจะรีเซ็ตและทำการเขียนข้อมูล IFFH ไปยังซีพรีจิสเตอร์ ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น “0” หรือ “1” จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7, 8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความเป็นตามเสียงข้างมาก ยกตัวอย่าง สุ่มพบลอจิก “1” 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลที่ได้รับได้นั้นเป็น “1”

ลำดับของการรับข้อมูลมีลักษณะเกี่ยวกับการส่งข้อมูลคือ เริ่มต้นด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุกๆการรับข้อมูลได้ 1 บิต จะมีพัลส์เลื่อน(shift) เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล(RX clock) หลังจากสัญญาณนาฬิกาถูกสุ่มท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั้นเอง พร้อมกันนั้นยังทำการเซตบิต RI ในรีจิสเตอร์ SCON ด้วย หากการอินเตอร์รัปต์หรือการรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI ก่อน เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุดเนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

2.5.2.3 การทำงานในโหมด 2 และ 3 ของวงจรถอดอนุกรม

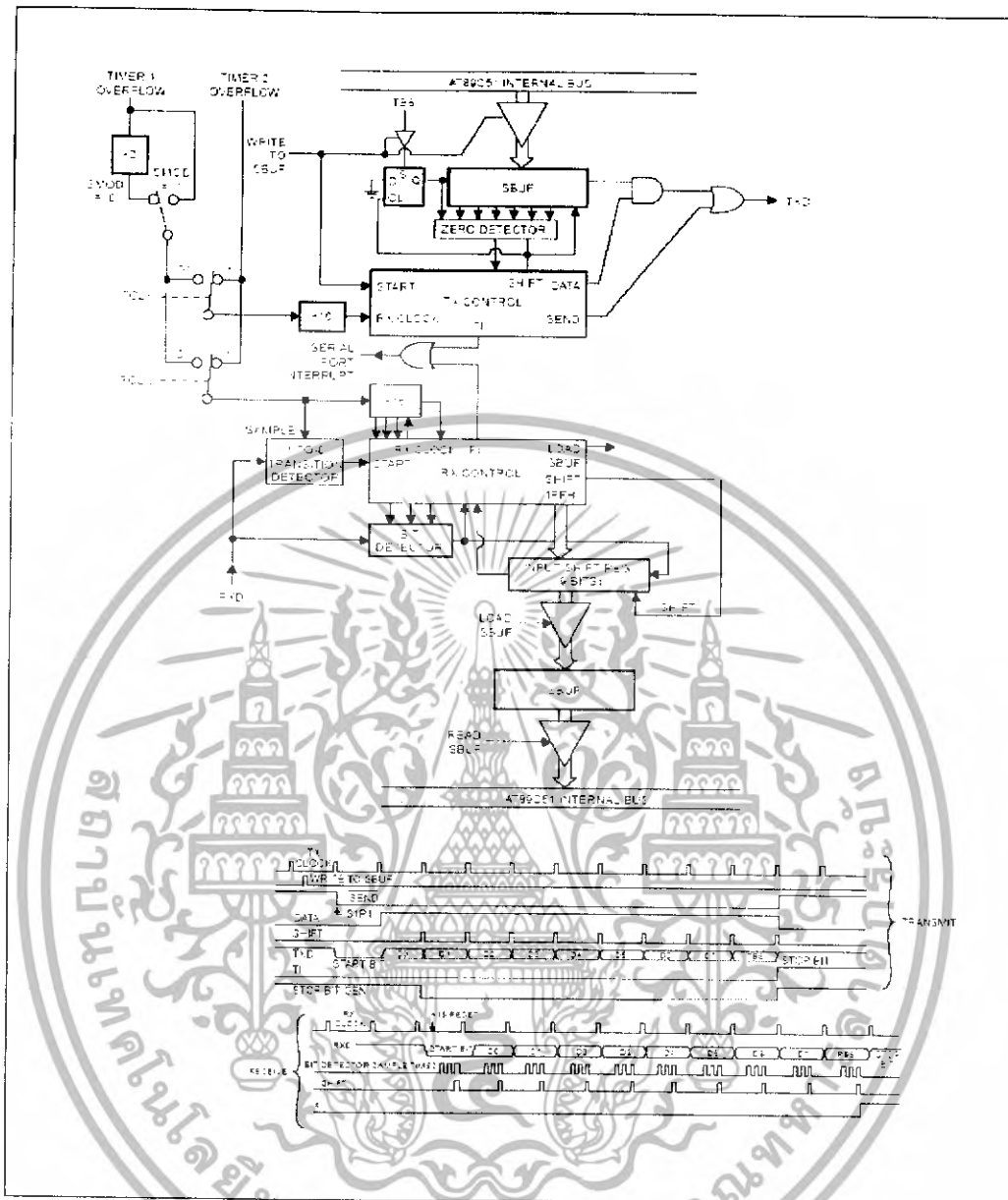
ในทั้งสองโหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น “0” จำนวน 1 บิต, บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน, บิตข้อมูลที่ 8 และบิตปิดท้ายมีค่าเป็น “1” จำนวน 1 บิต ในการส่งข้อมูล ข้อมูลที่ 9 จะได้รับการเก็บไว้ในบิต TB8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไปเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ $1/32$ หรือ $1/64$ ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1



รูปที่ 2.31 ไดอะแกรมการทำงานในโหมด 2 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

ในรูปที่ 3.31 และ 2.32 เป็นไดอะแกรมการทำงานและไดอะแกรมของการทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรม การทำงานโดยรวมจะคล้ายกับการทำงานในโหมด 1 ส่วนที่แตกต่างกันคือจำนวนบิตของข้อมูลที่ในโหมด 2 และ 3 จะมีเพิ่มมาอีก 1 บิต โดยส่วนใหญ่จะใช้เป็นบิตตรวจสอบพาริตี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.32 โค้ดแอมแกรมการทำงานในโหมด 3 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

2.5.3 อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์

- โหมด 0

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

$$\text{Baud rate} = \frac{\text{Clock}}{12} \quad (\text{บิตต่อวินาที})$$

โดยที่ Clock เป็นความถี่ของสัญญาณนาฬิกา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- โหมด 1 และ 3

เนื่องจากทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตรารอบคได้ 2 แหล่งคือ จากอัตรารอบคโอเวอร์โฟลวของไทมเมอร์ 1 และ 2 สำหรับอัตรารอบคเมื่อใช้การโอเวอร์โฟลวของไทมเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON มาพิจารณาประกอบด้วย โดยรีจิสเตอร์ PCON มีรายละเอียดดังนี้

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

SMOD(Double Baud rate bit) : ใช้ในการเพิ่มอัตรารอบคของการรับส่งข้อมูลของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ เป็นสองเท่า เมื่อบิตนี้เซตเป็น “1” และใช้ไทมเมอร์ 1 ในการสร้างรอบค โดยพอร์ตอนุกรมต้องถูกกำหนดให้ทำงานในโหมด 1, 2 หรือ 3

GF1 และ GF0(general-purpose flag bit) : เป็นแฟล็กสำหรับใช้งานทั่วไป เซตและเคลียร์ด้วยซอฟต์แวร์

PD(Power Down bit) : ใช้เอนเอเบิลการทำงานในโหมดประหยัดพลังงานแบบลดพลังงานหรือ พาวเวอร์ดาวน์(power down mode) สามารถเซตด้วยกระบวนการทางซอฟต์แวร์และเคลียร์ด้วยกระบวนการทางฮาร์ดแวร์ ถ้าต้องการให้ทำงานในโหมดประหยัดพลังงานแบบลดพลังงานต้องเซตบิตนี้ให้เป็น “1”

IDL (Idle mode bit) : ใช้เอนเอเบิลการทำงานในโหมดประหยัดพลังงานแบบไอดีล(idle mode) สามารถเซตด้วยกระบวนการทางซอฟต์แวร์และเคลียร์ด้วยกระบวนการทางฮาร์ดแวร์ ถ้าต้องการให้ทำงานในโหมดประหยัดพลังงานแบบไอดีลต้องเซตบิตนี้ให้เป็น “1”

จากข้างต้น สามารถคำนวณอัตรารอบคได้จาก

$$\text{Baud rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Clock}}{12 \times (256 - \text{TH1})}$$

โดยที่ SMOD เป็นค่าของบิตภายในรีจิสเตอร์ PCON ซึ่งอาจมีค่าเป็น 0 หรือ 1

TH1 เป็นค่าภายในรีจิสเตอร์ ซึ่งใช้ในสำหรับรีโหลด ค่าของการนับเวลา

Clock เป็นความถี่ของสัญญาณนาฬิกา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่ใช้ไทมเมอร์ 2 ในการกำหนดอัตราบอดโดยกำหนดให้ไทมเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด(baud rate generator) สามารถคำนวณหาอัตราบอดได้จาก

$$\text{Baud rate} = \frac{\text{Tmer 2 Overflow Rate}}{16} \quad (\text{บิตต่อวินาที})$$

โดยที่ Tmer 2 Overflow Rate เป็นอัตราการโอเวอร์โฟลวของไทมเมอร์ 2

ถ้าหากกำหนดให้ไทมเมอร์ 2 ทำงานในโหมดปกติ สามารถคำนวณหาอัตราบอดได้จาก

$$\text{Baud rate} = \frac{\text{Clock}}{32 \times (65536 - (\text{RCAP2H}, \text{RCAP2L}))}$$

โดยที่ (RCAP2H,RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิต ไม่คิดเครื่องหมาย

- โหมด 2

ในโหมดนี้อัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD มีค่าเป็น “0” อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น “1” อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสวการทางคณิตศาสตร์ได้ดังนี้

$$\text{Board rate} = \frac{2^{\text{SMOD}}}{64} \times \text{Clock} \quad (\text{บิตต่อวินาที})$$

โดยที่ Clock เป็นความถี่ของสัญญาณนาฬิกา

2.5.4 การกำหนดค่าของไทมเมอร์ 1 เพื่อเลือกอัตราบอด

การกำหนดค่าลงใน Timer 1 ทำได้โดยการโปรแกรมไปที่ TMOD ให้ทำงานแบบ 8-bit auto reload mode (โหมด 2) โดยเขียนค่าไปที่ TH1 ซึ่งโปรแกรมไปที่รีจิสเตอร์ TMOD ได้ดังนี้

```
MOV TMOD,#0010xxxxB
```

ค่า x หมายความว่า เป็นอะไรก็ได้เพราะบิตเหล่านี้ใช้ใน Timer 0

ถ้าต้องการอัตราบอดต่างๆ สามารถใช้ 16-bit mode ได้ โดยโปรแกรมเป็น TMOD = 0001xxxB ค่าของอัตราบอดที่ส่งออกมาจะมีค่าเท่ากับ ความถี่ของไทมเมอร์ 1 เกิด โอเวอร์โฟลวหารด้วย 32 (หรือหาร 16 ถ้า SMOD =1)

การคำนวณหาค่าอัตราบอดที่กำหนดด้วย ไทเมอร์ 1 สามารถทำได้โดยใช้สมการต่อไปนี้

$$\text{Baud rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Clock}}{12 \times (256 - \text{TH1})}$$

โดยที่ SMOD เป็นค่าของบิตภายในรีจิสเตอร์ PCON ซึ่งอาจมีค่าเป็น 0 หรือ 1

TH1 เป็นค่าภายในรีจิสเตอร์ ซึ่งใช้ในสำหรับรีโหลด ค่าของการนับเวลา

Clock เป็นความถี่ของสัญญาณนาฬิกา

รูปแบบทั่วไปของการหาอัตราบอดในโหมด 1 และ 3 สามารถหาได้ดังนี้

$$\text{Baud rate} = \frac{\text{Timer 1 Overflow Rate}}{32}$$

ถ้าต้องการอัตราบอดเท่ากับ 1,200 บิตต่อวินาที สามารถคำนวณค่าความถี่โอเวอร์โฟลวของไทเมอร์ 1 ได้ดังนี้

$$1,200 = \frac{\text{Timer 1 Overflow Rate}}{32}$$

จะได้ Timer 1 Overflow Rate เท่ากับ 38.4 kHz

ถ้าระบบใช้ความถี่สัญญาณนาฬิกาจากคริสตอล(Crystal) เท่ากับ 12 MHz ตัวไทเมอร์ 1 จะรับสัญญาณนาฬิกา(Clock) เท่ากับ 1 MHz ถ้าเราต้องการ Timer 1 Overflow เท่ากับ 38.4 kHz ดังนั้น ค่าอัตราโอเวอร์โฟลว มีค่าเท่ากับ $1 \text{ MHz} / 38.4 \text{ kHz} = 26.04 \text{ Clock}$ โดยค่าโอเวอร์โฟลวจะเกิดเมื่อเกิดการเปลี่ยนจาก FFH เป็น 00H ดังนั้นจะต้องให้ไทเมอร์ 1 นับไป 26 ครั้ง ดังนั้นค่าที่จะให้รีจิสเตอร์ TH1 มีค่าเท่ากับ -26 ซึ่งใช้เป็นค่ารีโหลด ดังนั้นเขียนคำสั่งได้ดังนี้

```
MOV TH1,#-26
```

ตัวโปรแกรมแอสเซมบลอร์ทั้งไปจะแปลงค่า -26 เป็น 0E6H เอง จากที่ผ่านมาจะเห็นว่าความถี่อัตราบอด จะมีความสัมพันธ์กับสัญญาณนาฬิกาที่ใช้ จากคริสตอลในตารางที่. จะเป็นค่าที่ต้องกำหนดในไทเมอร์ 1 เมื่อต้องการอัตราบอดต่างๆ จะเห็นว่าถ้าใช้ความถี่นาฬิกา 12 MHz ค่าอัตราบอดจะมีข้อผิดพลาด

Baud rate(bps)	Crystal(MHz)	SMOD	TH1	ค่า Baud rate ที่ได้(bps)	Error (%)
9,600	12	1	-7(F9H)	8,923	7
2,400	12	0	-13(F3H)	2,040	0.16
1,200	12	0	-16(E6H)	1,020	0.16
19,200	11.0592	1	-3(FDH)	19,200	0
9,600	11.0592	0	-3(FDH)	9,600	0
2,400	11.0592	0	-12(F4H)	2,400	0
1,200	11.0592	0	-24(E8H)	1,200	0

ตารางที่ 2.11 การเลือกอัตราบอดของวงจรถ่ายโอนข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51

2.6 การอินเทอร์รัปต์

การทำงานของไมโครคอนโทรลเลอร์ โดยทั่วไปมักมีอุปกรณ์ภายนอกต่อร่วมอยู่ ถ้าไมโครคอนโทรลเลอร์ต้องการทำงานกับอุปกรณ์ภายนอกจะต้องคอยตรวจสอบอุปกรณ์เหล่านั้นเสมอ ตัวอย่างเช่น ถ้าหากให้ไมโครคอนโทรลเลอร์พอร์ทหนึ่งต่ออยู่กับหลอด LED 7 ส่วน อีกพอร์ทหนึ่งต่อกับสวิทช์ ถ้าระบบของเราทำงานเป็นนาฬิกาแสดงทาง LED ถ้ามีการกดสวิทช์ให้นาฬิกาหยุดเดิน เราอาจเขียนโปรแกรมให้ทำงานเป็นนาฬิการะหว่างที่นาฬิกาเดินไปให้คอยตรวจสอบสวิทช์ด้วยว่ามีการกดหรือยัง การทำงานแบบนี้เรียกว่า Polling Method คือตัวไมโครคอนโทรลเลอร์จะต้องคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลาว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ ถ้ามีอุปกรณ์ภายนอกหลายตัวระบบต้องตรวจสอบอุปกรณ์ภายนอกหลายตัว ทำให้เสียเวลาในการทำงานหลักไป การทำงานอีกแบบหนึ่งจะให้ CPU ทำงานหลัก ถ้ามีการกดสวิทช์เมื่อไรให้นาฬิกาหยุดเดินทันที การทำงานในลักษณะนี้ CPU ไม่ต้องเสียเวลาในการตรวจสอบอุปกรณ์ภายนอก ถ้าอุปกรณ์ภายนอกต้องการติดต่อกับ CPU อุปกรณ์ภายนอกจะส่งสัญญาณมาบอก CPU เอง ระบบนี้เรียกว่า การอินเทอร์รัปต์ (Interrupt)

2.6.1 ขบวนการเกิดอินเทอร์รัปต์

ถ้าหากไมโครคอนโทรลเลอร์กำลังทำงานโปรแกรมหลักอยู่ เมื่อมีการอินเทอร์รัปต์เข้ามา ไมโครคอนโทรลเลอร์จะละทิ้งโปรแกรมหลัก แต่ไปทำงานโปรแกรมตอบสนองการอินเทอร์รัปต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Interrupt Service Routine) เมื่อทำโปรแกรมตอบสนองอินเทอร์รัปต์เสร็จไมโครคอนโทรลเลอร์จะกลับมาทำโปรแกรมเดิม

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m , $m+1$, $m+2$ ไปเรื่อยๆโดย PC จะชี้ตำแหน่งที่จะอ่านค่าคำสั่งถัดมา เมื่อโปรแกรมทำงานมาถึงตำแหน่งที่ $m+3$ แล้วเกิดการอินเทอร์รัปต์ขึ้น (ขณะนั้น PC อยู่ที่ $m+4$ โปรแกรมจะต้องทำงานโปรแกรมตอบสนองการอินเทอร์รัปต์ โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเทอร์รัปต์ จากนั้นจะเก็บค่า PC เดิมลงในหน่วยความจำสแตค เมื่อไมโครคอนโทรลเลอร์ทำงานโปรแกรมตอบสนองการอินเทอร์รัปต์เสร็จสิ้นลง จะคืนค่าใน สแตค ($m+4$) ให้กับ PC เพื่อให้ PC ทำโปรแกรมหลักต่อไป

2.6.2 สัญญาณอินเทอร์รัปต์

แหล่งกำเนิดสัญญาณอินเทอร์รัปต์ที่ใช้กับไมโครคอนโทรลเลอร์ มีสองชนิดคือ อินเทอร์รัปต์ภายใน และภายนอก โดยอินเทอร์รัปต์ภายในจะเกิดขึ้นจากภายในตัวไมโครคอนโทรลเลอร์เอง ได้แก่สัญญาณจาก ไทมเมอร์แฟลค 0(TF0) ไทมเมอร์แฟลค 1(TF1) และพอร์ตอนุกรม สำหรับอินเทอร์รัปต์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทางขา INTO และ INT1 เมื่อมีสัญญาณอินเทอร์รัปต์จากแหล่งต่างๆเข้ามา เราสามารถโปรแกรมจะให้ไมโครคอนโทรลเลอร์ยอมให้มีการอินเทอร์รัปต์ได้หรือไม่ โดยการโปรแกรมไปที่ รีจิสเตอร์ IE(Interrupt Enable)และมีสัญญาณอินเทอร์รัปต์มาจากแหล่งต่างๆ หลายแหล่งพร้อมกันเราสามารถจัดลำดับได้ว่า จะให้อินเทอร์รัปต์ใดเกิดก่อน โดยการโปรแกรมไปที่ อินเทอร์รัปต์ไพอริตี IP(Interrupt Priority) รีจิสเตอร์ทั้งสองตัวมีรายละเอียดดังนี้

2.6.3 รีจิสเตอร์เอ็นเอเบิลการอินเทอร์รัปต์ (IE :Interrupt Enables)

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเทอร์รัปต์จากแหล่งต่างๆ จะทำอินเทอร์รัปต์เหล่านั้นหรือไม่ โดยรายละเอียดของบิตต่างๆ มีดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA(Global enable/disable interrupt): ใช้เอ็นเอเบิลและดิสเอเบิลการตอบสนองการอินเทอร์รัปต์ทั้งหมด

“0” ดิสเอเบิลการอินเทอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์ไม่ตอบสนองการอินเทอร์รัปต์

“1” เอ็นเอเบิลการอินเทอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์สามารถตอบสนองการอินเทอร์รัปต์จากแหล่งกำเนิดชนิดต่างๆ

ถ้าต้องการให้ไมโครคอนโทรลเลอร์ตอบสนองการอินเทอร์รัปต์ไม่ว่าจะแหล่งกำเนิดใด จะต้องเซตบิตนี้ก่อนเสมอ

ET2(Timer 2 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวหรือการแคปเจอร์ในไทเมอร์/เคาน์เตอร์ 2 มีเฉพาะในเบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น

ES (Serial port interrupt enable bit) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากการรับหรือส่งข้อมูลทางพอร์ทอนุกรมภายในไมโครคอนโทรลเลอร์

ET1 (Timer1 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากโอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 1

EX1 (External interrupt 1 enable bit) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา $\overline{INT1}$

ET0 (Timer0 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 0

EX0 (External interrupt 0 enable bit) : ใช้ในการเอ็นเอเบิลการอินเทอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา $\overline{INT0}$

สำหรับขาบิต 6 ของรีจิสเตอร์ IE ไม่มีการใช้งาน ต้องกำหนดให้เป็น “0” เสมอ โดยทุกบิตสามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากข้างต้นสามารถสรุปโดยตารางที่ 2.12

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFH	ถ้าเซตยอมให้มีการอินเตอร์รัปต์
IE.6	-	AEH	ไม่ใช้งาน
IE.5	ET2	ADH	Enable อินเตอร์รัปต์จาก Timer 2
IE.4	ES	ACH	Enable อินเตอร์รัปต์จากพอร์ทอนุกรม
IE.3	ET1	ABH	Enable อินเตอร์รัปต์จาก Timer 1
IE.2	EX1	AAH	Enable อินเตอร์รัปต์จาก INT1
IE.1	ET0	A9H	Enable อินเตอร์รัปต์จาก Timer 0
IE.0	EX0	A8H	Enable อินเตอร์รัปต์จาก INTO

ตารางที่ 2.12 บิตและหน้าที่ต่างๆของรีจิสเตอร์ เอ็นเอเบิลการอินเตอร์รัปต์

2.6.4 รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปต์ (IP : Interrupt Priority)

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

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ใช้ในการจัดลำดับความสำคัญของการอินเตอร์รัปต์ซึ่งสามารถจัดได้สองลำดับ ถ้าเป็น "1" หมายความว่ามีความสำคัญสูงสุด ถ้าเป็น "0" หมายความว่ามีความสำคัญต่ำสุด ความหมายของบิตต่างๆ แสดงได้ดังตารางที่ . ถ้าหากกำหนดให้มีความสำคัญเป็น "1" เหมือนกันหมดไมโครคอนโทรลเลอร์ จะจัดลำดับความสำคัญใหม่ดังตารางที่ 2.13

PT2 (Timer 2 interrupt priority bit) : ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวหรือการแคปเจอร์ในไทเมอร์/แกนเตอร์ 2 จะมีเฉพาะในเบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น

PS (Serial port interrupt priority bit) : ใช้ในการกำหนดความสำคัญของการอินเทอร์รัปต์อันเนื่องมาจากการรับหรือการส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

PT1 (Timer 1 interrupt priority bit) : ใช้ในการกำหนดความสำคัญของการอินเทอร์รัปต์อันเนื่องมาจากโอเวอร์โฟลวในไทมเมอร์/เคาน์เตอร์1

PX1 (External interrupt 1 priority bit) : ใช้ในการกำหนดความสำคัญของการอินเทอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา $\overline{INT1}$

PT0 (Timer 0 interrupt priority bit) : ใช้ในการกำหนดความสำคัญของการอินเทอร์รัปต์อันเนื่องมาจากโอเวอร์โฟลวในไทมเมอร์/เคาน์เตอร์0

PX0 (External interrupt 0 priority bit) : ใช้ในการกำหนดความสำคัญของการอินเทอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา $\overline{INT0}$

ลำดับ	อินเทอร์รัปต์
1(สูงสุด)	IE0
2	TF0
3	IE1
4	TF1
5(ต่ำสุด)	Serial Port

ตารางที่ 2.13 ตารางแสดงการจำลำดับความสำคัญอัตโนมัติของไมโครคอนโทรลเลอร์

ทุกบิตสามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ และจากข้างต้นสามารถสรุปได้ดังตาราง...

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	BFH	ไม่ใช้งาน
IP.6	-	BEH	ไม่ใช้งาน
IP.5	PT2	BDH	ใช้กับอินเทอร์รัปต์จาก Timer 2
IP.4	PS	BCH	ใช้กับอินเทอร์รัปต์จาก พอร์ตอนุกรม
IP.3	PT1	BBH	ใช้กับอินเทอร์รัปต์จาก Timer 1
IP.2	PX1	BAH	ใช้กับอินเทอร์รัปต์จาก INT1
IP.1	PT0	B9H	ใช้กับอินเทอร์รัปต์จาก Timer 0
IP.0	PX0	B8H	ใช้กับอินเทอร์รัปต์จาก INT0

ตารางที่ 2.14 บิตและหน้าที่ต่างๆของรีจิสเตอร์รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเทอร์รัปต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.33 แสดงการอินเทอร์รัปต์จากแหล่งต่างๆ ที่มีผลกับไมโครคอนโทรลเลอร์ ถ้าเป็นเบอร์ AT89C51 จะถูกอินเทอร์รัปต์ได้ 5 แหล่ง ถ้าเป็นเบอร์ AT89C52 และในอนุกรม AT89Sxx จะถูกอินเทอร์รัปต์ได้ 6 แหล่ง โดยเพิ่มอินเทอร์รัปต์จาก Timer 2 ในรูปที่. จะแสดงให้เห็นว่า ถ้าไมโครคอนโทรลเลอร์จะถูกอินเทอร์รัปต์ได้จะต้องเซตค่า Global Enable ในรีจิสเตอร์ IE นอกจากนี้ยังกำหนดได้ว่าจะให้อินเทอร์รัปต์จากแหล่งใดเกิดได้ โดยการเซตค่า Interrupt Enable ของอินเทอร์รัปต์จากแหล่งต่างๆในรีจิสเตอร์ IE จากรูปยังแสดงให้เห็นอีกว่าเมื่อมีการอินเทอร์รัปต์เข้ามาจะมีผลต่อแฟล็กใด เช่นถ้า INTO เป็น “1” บิต IE0 จะเป็น “1” หมายความว่าถูกอินเทอร์รัปต์โดยแฟล็กต่างๆที่มีผลจากการถูกอินเทอร์รัปต์แสดงได้ดังตารางที่ ..

อินเทอร์รัปต์	แฟล็ก	ประกอบอยู่ในรีจิสเตอร์
External 0	IE0	TCON.1
External 1	IE1	TCON.3
Timer 1	TF1	TCON.7
Timer 0	TF0	TCON.5
Serial port	TI	SCON.1
Serial port	RI	SCON.0
Timer 2	TF2	T2CON.7(AT89C52)
Timer 2	EXF2	T2CON.6(AT89C52)

ตารางที่ 2.15 แฟล็กที่จะทำงานเมื่อถูกอินเทอร์รัปต์

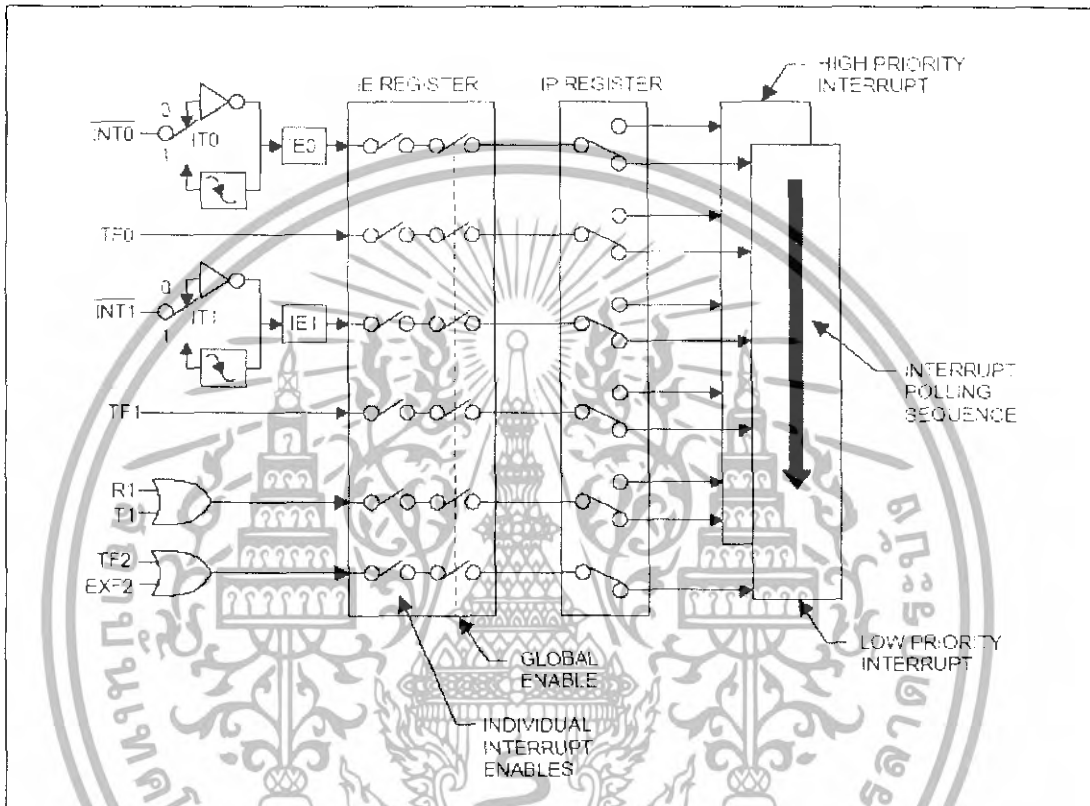
จากตาราง 2.15 จะเห็นว่า ถ้ามีการอินเทอร์รัปต์จากภายนอกเข้ามา ตัวที่จะอินเทอร์รัปต์ไมโครคอนโทรลเลอร์คือ บิตแฟล็ก IE0 ซึ่งอยู่ในรีจิสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมดแล้วจะอินเทอร์รัปต์ไมโครคอนโทรลเลอร์ ทางบิตแฟล็ก TI ถ้ารับข้อมูลหมดแล้วจะอินเทอร์รัปต์ไมโครคอนโทรลเลอร์ ทางบิต RI ซึ่งอยู่ในรีจิสเตอร์ SCON และถ้าใช้ Timer0 ในการนับเมื่อเกิด Overflow สามารถอินเทอร์รัปต์ไมโครคอนโทรลเลอร์ได้ทางบิต TF0

2.6.5 โครงสร้างระบบการอินเทอร์รัปต์ของไมโครคอนโทรลเลอร์

จากรายละเอียดทั้งหมดของการอินเทอร์รัปต์ในไมโครคอนโทรลเลอร์ สามารถแสดงเป็นโครงสร้างของระบบอินเทอร์รัปต์โดยรวมดังในรูปที่... โดยรีจิสเตอร์ IP แทนด้วยสวิทช์สองทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เพื่อกำหนดนัยสำคัญของการตอบสนองการอินเทอร์รัปต์ ในขณะที่รีจิสเตอร์ IE และบิตเอ็นเอเบิล การอินเทอร์รัปต์แบบต่างๆ จะแทนด้วยสวิตช์ต่อกัน นั่นคือถ้าไม่ต่อวงจรทั้งสองตัวก็จะไม่มีการอินเทอร์รัปต์แบบนั้นๆเกิดขึ้น นั่นหมายความว่า การอินเทอร์รัปต์จะเกิดขึ้นก็ต่อเมื่อเงื่อนไขในการเกิดเหตุการณ์เป็นจริง มีการเอ็นเอเบิลการอินเทอร์รัปต์ไว้อย่างถูกต้อง และกำหนดลำดับความสำคัญไว้อย่างเหมาะสม



รูปที่ 2.33 โครงสร้างของระบบอินเทอร์รัปต์ภายในไมโครคอนโทรลเลอร์

2.6.6 การทำงานของระบบหลังถูกอินเทอร์รัปต์

เมื่อไมโครคอนโทรลเลอร์ถูกอินเทอร์รัปต์ จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัปต์โดยตำแหน่งที่จะกระโดดไปเรียกว่า อินเทอร์รัปต์เวกเตอร์ (Interrupt Vectors) เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัปต์เรียบร้อยแล้ว ไมโครคอนโทรลเลอร์จะกระโดดมาทำงานยังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองอินเทอร์รัปต์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า PC ลงหน่วยความจำสแตคซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจิสเตอร์ SP เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัปต์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตคให้ PC ตามเดิม ค่าอินเทอร์รัปต์เวกเตอร์ของไมโครคอนโทรลเลอร์ แสดงได้ดังตารางที่ 2.16

อินเตอร์รัปต์	อินเตอร์รัปต์เวกเตอร์
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial port	0023H
Timer 2	002BH

ตารางที่ 2.16 อินเตอร์รัปต์เวกเตอร์ของอินเตอร์รัปต์ต่างๆ

จากตาราง 2.16 จะเห็นว่าถ้าระบบถูกอินเตอร์รัปต์จากภายนอกทาง INTO ตัวไมโครคอนโทรลเลอร์จะกระโดดไปทำงานที่ตำแหน่ง 0003H ถ้าระบบถูกอินเตอร์รัปต์จาก Timer0 จะกระโดดไปทำงานตำแหน่ง 000BH

2.6.7 การออกแบบโปรแกรมอินเตอร์รัปต์

ในการเขียนโปรแกรมหลัก(Main Program) จะต้องกำหนดค่าว่าจะให้ไมโครคอนโทรลเลอร์ถูกอินเตอร์รัปต์ด้วยอะไร และจะให้ไมโครคอนโทรลเลอร์ถูกอินเตอร์รัปต์ได้หรือไม่ โดยการโปรแกรมค่าต่างๆใน IE ดังนั้นในโปรแกรมหลักจะต้องมีการโปรแกรมต่อไปนี้

1. โปรแกรมค่าในรีจิสเตอร์ IE
2. โปรแกรมค่าในรีจิสเตอร์ IP

สำหรับโปรแกรมตอบสนองการอินเตอร์รัปต์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่ง แต่จะต้องจบโปรแกรมย่อยด้วยค่า RETI (Return Interrupt)

จากตารางที่ 2.16 จะเห็นว่าถ้ากด Reset หรือให้ระบบเริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง 0000H และจะเห็นว่า ตำแหน่งที่เก็บโปรแกรมหลักมีโอกาสอย่างมากที่จะทับกับหน่วยความจำโปรแกรมเก็บค่าอินเตอร์รัปต์ที่ตำแหน่ง 0003H ถ้าโปรแกรมยาวมากอาจจะไปทับตำแหน่ง 000BH ได้ซึ่งเป็นตำแหน่งของอินเตอร์รัปต์เวกเตอร์ของ Timer0 ดังนั้นในการเขียนโปรแกรมหลัก ภายใน 3 ตำแหน่งแรกคือ 0000H, 0001H และ 0002H จะต้องกระโดดไปที่อื่นก่อน เพื่อให้ข้ามอินเตอร์รัปต์เวกเตอร์ไปซึ่งอาจเขียนโปรแกรมได้ดังนี้

```

ORG    0000H    ; เริ่มต้น โปรแกรม
LJMP   MAIN    ; กระโดดไปโปรแกรมหลัก
.....        ; เพื่อหนีอินเทอร์รัปต์เวกเตอร์
.....
ORG    0030h    ; ตำแหน่งเริ่มต้นของโปรแกรม
MAIN:  .....   ; เริ่มต้น โปรแกรมหลัก
.....

```

จากตัวอย่างโปรแกรมจะเห็นว่า เมื่อเริ่มต้นโปรแกรมหรือระบบถูกรีเซต ระบบจะทำงานตำแหน่งแรกคือคำสั่งกระโดดไปโปรแกรมหลัก ซึ่งอยู่ต่อจากโปรแกรมตอบสนองการอินเทอร์รัปต์ที่อยู่ตำแหน่ง 0030H

• โปรแกรมตอบสนองการอินเทอร์รัปต์แบบสั้น

จากตารางอินเทอร์รัปต์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเทอร์รัปต์แต่ละแห่งห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเทอร์รัปต์จากแหล่งต่างๆ หลายๆ แหล่งและโปรแกรมตอบสนองการอินเทอร์รัปต์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์ จะทำให้โปรแกรมไปทับตำแหน่งของโปรแกรมตอบสนองการอินเทอร์รัปต์ของอินเทอร์รัปต์ถัดไป แต่ถ้าโปรแกรมตอบสนองการอินเทอร์รัปต์มียาวมากเกินไปเราสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```

ORG    0000H
LJMP   MAIN    ; กระโดดไปโปรแกรมหลัก

ORG    0000BH    ; ตำแหน่งเริ่มต้นของการอินเทอร์รัปต์ Timer0
TOISR: .....
.....
RETI    ; กลับไปโปรแกรมหลัก
MAIN:  .....   ; โปรแกรมหลัก
.....

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่างโปรแกรมจะใช้อินเทอร์รัปต์จาก Timer0 เมื่อระบบเริ่มทำงานจะทำตำแหน่ง 0000H โดยกระโดดไปโปรแกรมหลักซึ่งอยู่ที่ตำแหน่งต่อจากโปรแกรมตอบสนองการอินเทอร์รัปต์เมื่อมีการอินเทอร์รัปต์ Timer0 ระบบจะทำโปรแกรมตำแหน่งที่ 000BH ซึ่งเป็นอินเทอร์รัปต์เวกเตอร์ของ Timer0 โดยโปรแกรมตอบสนองการอินเทอร์รัปต์จะจบด้วยคำสั่ง RETI เพื่อกลับสู่โปรแกรมหลักต่อไป

- โปรแกรมตอบสนองการอินเทอร์รัปต์ขนาดใหญ่

ในกรณีที่มีการอินเทอร์รัปต์จากหลายแหล่ง และ โปรแกรมตอบสนองการอินเทอร์รัปต์แต่ละโปรแกรมยาวเกิน 8 ไบต์ เราไม่สามารถเขียนโปรแกรมตอบสนองการอินเทอร์รัปต์ไว้ที่ตำแหน่งของอินเทอร์รัปต์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเทอร์รัปต์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเทอร์รัปต์ที่เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่าง ต่อไปนี้

```

ORG 0000H ; เริ่มโปรแกรมของระบบ
LJMP MAIN ; กระโดดไปโปรแกรมหลัก
ORG 000BH ; ตำแหน่งของอินเทอร์รัปต์ Timer0
LJMP LED1 ; กระโดดไปโปรแกรมตอบสนองการอินเทอร์รัปต์ชื่อ LED1
ORG 0030H ; ตำแหน่งหลังอินเทอร์รัปต์เวกเตอร์
MAIN: ..... ; โปรแกรมหลัก
.....
LED1: ..... ; โปรแกรมตอบสนองการอินเทอร์รัปต์ Timer0
.....
RETI ; กลับสู่โปรแกรมหลัก

```

จากโปรแกรมจะเห็นว่า เมื่อระบบทำงาน จะต้องทำที่ตำแหน่ง 0000H โดยกระโดดไปทำโปรแกรมหลักที่ตำแหน่งต่อจาก 0030H เพราะตำแหน่งดังกล่าวข้ามอินเทอร์รัปต์เวกเตอร์จากแหล่งต่างๆ ไปแล้วเมื่อมีการอินเทอร์รัปต์จาก Timer0 โปรแกรมจะต้องทำงานที่ตำแหน่ง 000BH แต่โปรแกรมตอบสนองการอินเทอร์รัปต์ยาวมาก ที่ตำแหน่ง 000BH จึงให้ทำโปรแกรมกระโดดโดยกระโดดไปที่โปรแกรมตอบสนองการอินเทอร์รัปต์ชื่อ LED1 ซึ่งอยู่ท้ายโปรแกรม เมื่อจบโปรแกรมจะจบด้วยคำสั่ง RETI เพื่อกลับไปโปรแกรมหลักต่อไป

บทที่ 3

การสื่อสารข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรม จะยากกว่าการสื่อสารแบบขนาน การสื่อสารของอุปกรณ์ที่ต่อกับพอร์ตอนุกรม จะถูกเปลี่ยน (Convert) เป็น สัญญาณแบบขนาน ซึ่งจะใช้ UART (Universal Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) เป็นตัวทำหน้าที่นี้ ส่วนทางด้านซอฟต์แวร์ ก็มีรีจิสเตอร์ที่ต้องจัดการมากกว่าการสื่อสารแบบขนานอีกหลายตัว

3.1 หลักการและโครงสร้างของการสื่อสารแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมเป็นการสื่อสารที่การรับส่งข้อมูลจะใช้สายสัญญาณจำนวนน้อย ซึ่งปกติจะใช้เพียง 1 คู่ คือสายสัญญาณที่เป็นข้อมูล และสายกราวด์เปรียบเทียบกับ โดยข้อมูลที่ส่งออกหรือรับเข้าในลักษณะบิตต่อบิต ซึ่งเมื่อเราเปรียบเทียบกับการสื่อสารข้อมูลแบบขนานที่จำนวนข้อมูลและอัตราเร็วในการสื่อสารข้อมูลเท่ากันแล้ว การสื่อสารข้อมูลแบบอนุกรมจะต้องใช้เวลาในการรับ-ส่งข้อมูลมากกว่าอย่างแน่นอน แต่ข้อดีของการสื่อสารข้อมูลแบบอนุกรมนี้คือ การใช้สายสัญญาณน้อยกว่า และสามารถส่งสัญญาณได้ในระยะทางไกลกว่าเมื่ออัตราละทอน หรือ ผิดเพี้ยนของสัญญาณที่มาจากความยาวของสายสัญญาณจะมีค่าเท่ากับการสื่อสารข้อมูลแบบขนาน แต่การสื่อสารข้อมูลแบบอนุกรมมีวิธีการที่จะลดผลจากการลดทอนของสัญญาณนี้โดยอาศัยหลักการรับ-ส่งสัญญาณแบบดิฟเฟอเรนเชียล ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงเหมาะสำหรับใช้กับการสื่อสารข้อมูลระยะไกล หรือการสื่อสารที่ต้องการใช้สายหรือช่องสัญญาณในการรับ-ส่งข้อมูลจำนวนน้อย เช่น การสื่อสารข้อมูลโครงข่ายแบบท้องถิ่น(Local Area Network : LAN)

3.2 ข้อดีของการสื่อสารแบบอนุกรมคือ

3.1.1 ระยะของสายอนุกรมสามารถมีความยาวได้มากกว่าสายของแบบขนานมาก ทั้งนี้เพราะสัญญาณของพอร์ตอนุกรม ซึ่งส่วนใหญ่ใช้มาตรฐาน RS-232C จะมีค่า -3 โวลต์ ถึง -15 โวลต์ สำหรับลอจิก "1" หรือมาร์ก (Mark) และมีค่า $+3$ โวลต์ ถึง $+15$ โวลต์ สำหรับลอจิก "0" หรือ สเปซ (Space) (สำหรับช่วง $+3$ โวลต์ ถึง -3 โวลต์ เป็นช่วงไม่นิยาม) ส่วนสัญญาณของการสื่อสารแบบขนาน นั้น "1" จะมีค่า $+5$ โวลต์ และ ลอจิก "0" จะมีค่า 0 โวลต์ ทำให้สัญญาณของการสื่อสารแบบอนุกรม สามารถรับการสูญเสียของสาย (Cable loss) ได้มากกว่าสัญญาณของการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สื่อสารแบบขนาน ปรกติสาย พอร์ตขนานจะไปได้เพียง 5 ฟุต ส่วนสาย RS-232 จะไปได้ถึง 50 ฟุต ที่ความเร็ว สูงสุดของมัน

3.1.2 การสื่อสารแบบอนุกรมจะใช้จำนวนสายไฟน้อยกว่าแบบขนาน ถ้าต่อในลักษณะ โมเด็ม(Null Modem) จะใช้สายเพียง 3 เส้น ขณะที่แบบขนานจะต้องใช้สาย 19 ถึง 25 เส้น

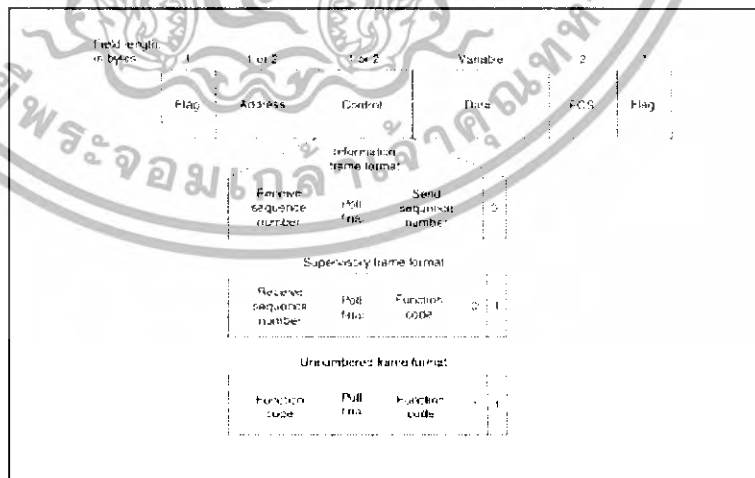
3.1.3 การสื่อสารแบบไร้สายเช่นการใช้อินฟราเรด การส่งพร้อมกันทีละ 8 บิต แบบขนาน จะทำให้ไม่สามารถแยกแยะได้ว่าบิต ไตเป็น บิต 0 หรือ บิต 1 เป็นต้น ปัจจุบันอุปกรณ์ IrDA มีความเร็วไม่ต่ำกว่า 115.2K Baud แต่มี Pulse length เพียง 3/16 ของ RS-232 เพื่อประหยัดพลังงาน เพราะส่วนมากใช้ในอุปกรณ์แบบพกพาเช่น โน้ตบุ๊ค หรือ ปาล์ม

3.1.4 ปัจจุบันไมโครคอนโทรลเลอร์ มักมีการผนวกพอร์ตอนุกรมไว้ด้วย เพราะใช้จำนวนขาน้อยกว่าแบบขนาน

3.3 รูปแบบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมสามารถแบ่งออกได้เป็น 2 แบบคือ การรับส่งข้อมูลแบบซิงโครนัสและอะซิงโครนัส ซึ่งมีรายละเอียดดังต่อไปนี้

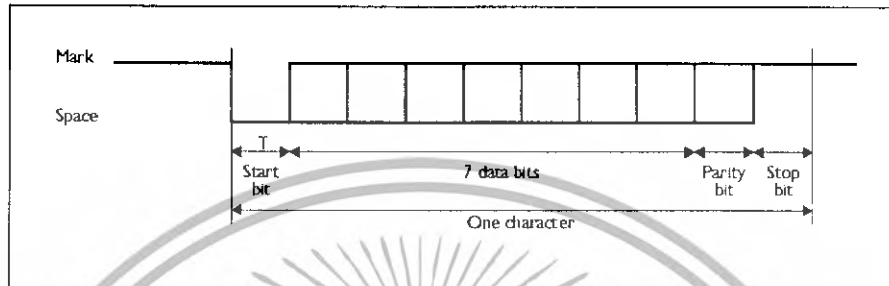
3.3.1 การรับส่งข้อมูลแบบซิงโครนัส คือการส่งข้อมูลที่ข้อมูลแต่ละเวิร์คถูกส่งออกไปตามเวลาที่แน่นอน โดยจะต้องมีการส่งสัญญาณนาฬิกาไปพร้อมๆ กับสัญญาณข้อมูล ในการส่งข้อมูลในระยะสั้นๆ สัญญาณนาฬิกาซึ่งใช้เป็นสัญญาณซิงค์ อาจจะส่งแยกไปในสายส่งสัญญาณอีกเส้นหนึ่งไม่รวมไปในสายข้อมูลก็ได้แต่ถ้าเป็นการสื่อสารระยะไกลๆ แล้ว สัญญาณนาฬิกาจะถูกเข้ารหัสส่งร่วมไปกับสัญญาณข้อมูลในสายเดียวกัน



รูปที่ 3.1 การส่งข้อมูลแบบซิงโครนัส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.2 การรับส่งข้อมูลแบบอะซิงโครนัส คือการรับส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิกาช่วยด้วยแต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกว่า อัตราบอด หรือบอดเรต(baud rate)มีหน่วยเป็น บิตต่อวินาที(bit per second : bps)



รูปที่ 3.2 การส่งข้อมูลแบบอะซิงโครนัส

รูปแบบของข้อมูลที่ใช้ในการรับส่งข้อมูลแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น(Start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาริตี(Parity bit) มีขนาด 1 บิต หรือ ไม่มี
4. บิตปิดท้ายหรือบิตหยุด(Stop bit) มีขนาด 1 บิต

รูปที่ 3.2 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขาข้อมูล (data)จะมีสถานะลอจิก “1” เรียกสถานะนี้ว่า สถานะหยุดรอ(waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขาข้อมูลมีลอจิก “0” ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น(start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด หรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วย บิตพาริตี ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้ายหรือบิตหยุด โดยจะเป็นการทำให้ขาข้อมูลมีสถานะลอจิก “1” อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิตเพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือค่าบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ตั้งแต่ 110 ถึง 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที

การตรวจสอบพาริตีสามารถกำหนดให้เป็นแบบคี่(odd),แบบคู่(even)หรือไม่มีการตรวจสอบพาริตีก็ได้ พาริตีคี่หรือคู่แสดงถึงจำนวนลอจิก “1” ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมกับบิตพาริตีว่ามีจำนวนเป็นเลขคู่หรือคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99 H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือ 10011001B จะเห็นว่าข้อมูลในไบนารีนี้มีจำนวนลอจิก “1” จำนวน 4 ตัว ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตีเป็นคู่ ค่าของบิตพาริตีจะต้องมีลอจิกเป็น “0” แต่ถ้ากำหนดพาริตีเป็นคี่ ค่าของบิตพาริตีจะต้องเป็น “1” เพื่อให้ข้อมูล 1 ไบนารีรวมทั้งบิตพาริตีเป็นคี่

บิตพาริตีถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART ซึ่งทางภาครับต้องกำหนดการตรวจสอบพาริตีที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตีคี่หรือคู่ จากนั้นทางภาครับของ UART จะทำการตรวจสอบค่าพาริตีที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่โดยการนับลอจิก “1” ทั้งหมดรวมทั้งบิตพาริตีด้วย ถ้ากำหนดค่าพาริตีไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้งาน กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ยากที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำให้การรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำให้การส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตีเป็น none นั้นทั้งภาครับและส่งจะไม่มีตรวจสอบพาริตี

นอกจากที่กล่าวไปแล้วการสื่อสารข้อมูลแบบอนุกรมยังสามารถที่จะแบ่งตามลักษณะของทิศทางในการสื่อสารข้อมูล ตาม โครงสร้างและความต้องการของระบบ ได้ดังนี้

1. การสื่อสารข้อมูลในทิศทางเดียวตลอดเวลา : Simplex
2. การสื่อสารข้อมูลใน 2 ทิศทางแต่สลับเวลา : Half-duplex
3. การสื่อสารข้อมูลใน 2 ทิศทางตลอดเวลา : Full-duplex

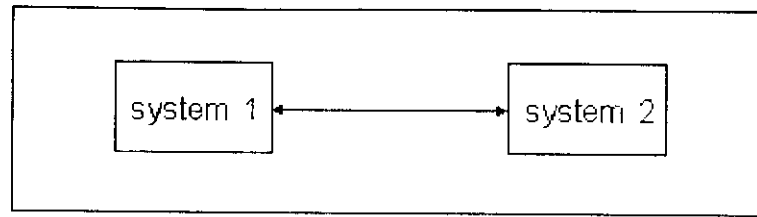
การสื่อสารในทิศทางเดียวตลอดเวลา(Simplex) : เป็นการสื่อสารในทิศทางใดก็ใช้ทิศทางนั้นตลอดเวลาไม่มีการเปลี่ยนแปลงทิศทางเช่นการส่งสัญญาณภาพจากสถานีไปยังเครื่องรับ หรือการส่งข้อมูลไปยังวิทยุติดตามตัว



รูปที่ 3.3 การสื่อสารในทิศทางเดียวตลอดเวลา

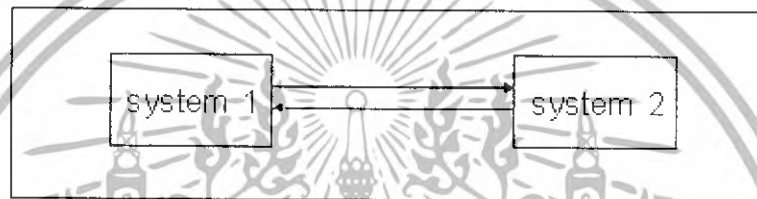
การสื่อสารสองทิศทางสลับเวลา(Half-duplex) : เป็นการสื่อสารข้อมูลใน 2 ทิศทาง แต่ในเวลาหนึ่งนั้นสัญญาณจะไปได้ในทิศทางเดียวเท่านั้น นั่นคือถ้าตัวหนึ่งตัวใดเป็นตัวส่ง อีกตัวจะต้องเป็นตัวรับข้อมูล จะส่งพร้อมกันไม่ได้ ดังนั้นอุปกรณ์แต่ละตัวที่จะเชื่อมต่อหรือสื่อสารข้อมูลในลักษณะนี้จะต้องเป็นได้ทั้งตัวรับหรือตัวส่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.4 การสื่อสารสองทิศทางสลับเวลา

การสื่อสารสองทิศทางตลอดเวลา(Full-duplex) : เป็นการสื่อสารข้อมูลที่คล้ายกับ half-duplex แต่เป็นการสื่อสารข้อมูลใน 2 ทิศทางตลอดเวลา



รูปที่ 3.5 การสื่อสารสองทิศทางตลอดเวลา

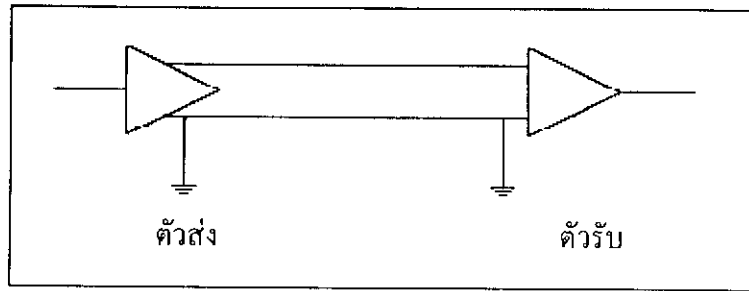
3.4 มาตรฐานของการสื่อสารข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมที่ใช้งานอยู่ในปัจจุบันนั้น ได้มีการกำหนดมาตรฐานการรับส่งข้อมูลไว้หลายแบบด้วยกัน ซึ่งมีดังต่อไปนี้

3.4.1 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

การสื่อสารข้อมูลแบบอนุกรมที่ใช้งานในปัจจุบันนั้น ได้มีการกำหนดมาตรฐานการรับส่งข้อมูลไว้หลายแบบด้วยกัน แต่ที่ได้รับความนิยมนำมาใช้งานเป็นอย่างมาก คือการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C และที่มาตรฐานนี้เป็นที่นิยมเนื่องจากเป็นระบบการสื่อสารข้อมูลที่ใช้ในเครื่องคอมพิวเตอร์ IBM PC ซึ่งเป็นคอมพิวเตอร์ที่มีการใช้งานอย่างแพร่หลายจากอดีตมาจนถึงปัจจุบัน มาตรฐาน RS-232C มีโครงสร้างการสื่อสารเป็นแบบจุดต่อจุดเท่านั้น โดยมีลักษณะสมบัติทางไฟฟ้าและทางกายภาพดังรูปที่ 3.6 และตารางที่ 3.1

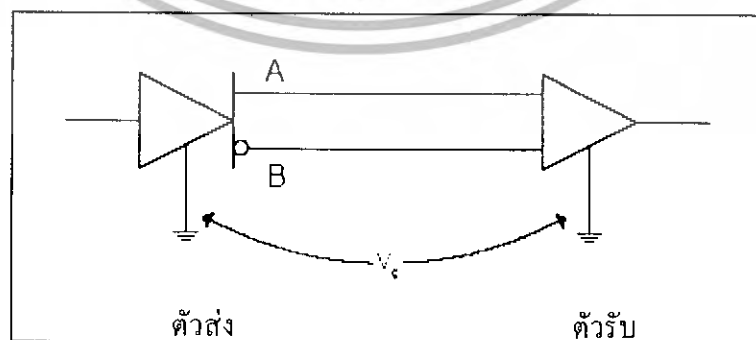
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.6 แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232

3.4.2 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-422A

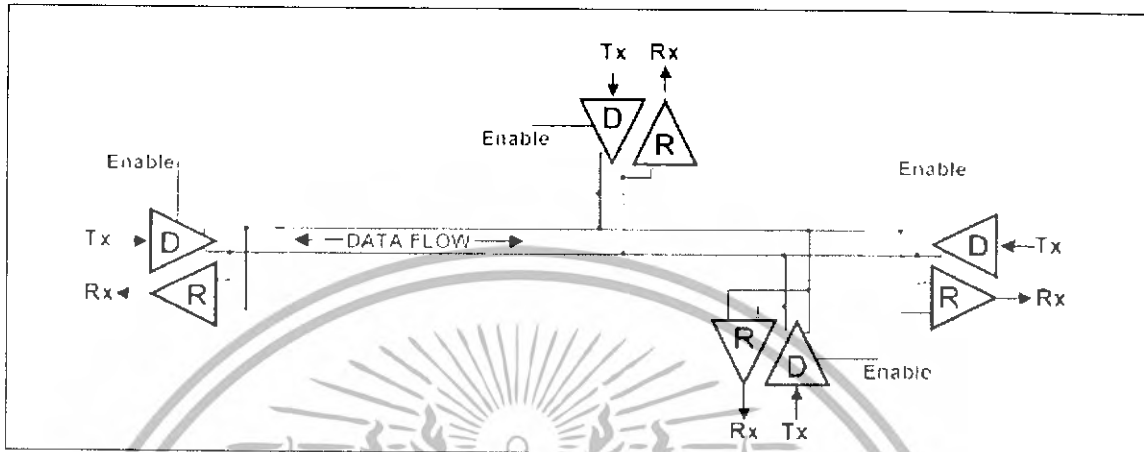
ในการออกแบบระบบการสื่อสารข้อมูลที่กำลังกล่าวมา ได้มีการพยายามที่จะออกแบบให้การสื่อสารข้อมูลได้รวดเร็วยิ่งขึ้น และมีระยะทางในการสื่อสารข้อมูลที่ยาวขึ้นด้วย ซึ่งที่ผ่านมาการสื่อสารข้อมูลตามมาตรฐาน RS-232C ได้ออกแบบมาเพื่อใช้เชื่อมโยงกับโมเด็มเท่านั้นจึงไม่ได้คำนึงถึงความเร็วและระยะทางในการสื่อสาร แต่ในปัจจุบันได้มีการออกแบบมาเพื่อรองรับความต้องการของการทำงานที่ต้องการให้รับและส่งข้อมูลได้ไกลขึ้น คือมาตรฐาน RS-422A ซึ่งจะใช้สัญญาณดิฟเฟอเรนเชียล ดังรูปที่ 3.7 หลักการคือ สัญญาณที่จะรับ-ส่งจะเป็นการเปรียบเทียบระหว่างสัญญาณ 2 เส้น เทียบกับมาตรฐาน RS-232C ที่สัญญาณจะเทียบกับกราวด์ ซึ่งในการสื่อสารในระยะไกลๆ แล้วสัญญาณจะถูกลดทอนไปและเมื่อสัญญาณถูกลดทอนถึงจุดๆ หนึ่งสัญญาณนั้นก็จะมีผิดพลาดไปจากความจริงก็จะทำให้การรับ-ส่งข้อมูลเกิดความผิดพลาดขึ้น แต่สำหรับสัญญาณดิฟเฟอเรนเชียลแล้วการลดทอนของสัญญาณจะไปลดทอนทั้งสองเส้นด้วยค่าที่เท่ากันหรือใกล้เคียงกัน และความแตกต่างของสัญญาณระยะสัญญาณทั้ง 2 เส้น จากตัวส่งไปยังตัวรับก็คงมีค่าเท่าเดิมหรือเปลี่ยนแปลงน้อย จึงทำให้ผลของการลดทอนต่อสัญญาณที่ระยะการสื่อสารไกลมีผลต่อสัญญาณดิฟเฟอเรนเชียลมีค่าน้อยกว่า การสื่อสารข้อมูลแบบนี้จึงสามารถส่งข้อมูลได้ไกลกว่าและอัตราการสื่อสารข้อมูลสูงกว่า ดังแสดงในตารางที่ 3.1



รูปที่ 3.7 แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-422A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.3 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-485



รูปที่ 3.8 แสดงโครงสร้างของการสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-485

การสื่อสารข้อมูลตามมาตรฐานที่กล่าวมาข้างต้นคือ RS-232C นั้นเป็นมาตรฐานการสื่อสารข้อมูลในแบบที่ใช้การระหว่างอุปกรณ์ หรือ จุดต่อจุด (point-to-point) ส่วน RS-422A นั้นเป็นมาตรฐานที่พัฒนามาจาก RS-232C ให้ได้ระยะทางไกลขึ้น และอัตราการสื่อสารเพิ่มขึ้น แต่ก็ยังเป็น การสื่อสารข้อมูลจากอุปกรณ์หนึ่งไปยังอุปกรณ์อื่นๆ ได้สูงสุด 10 ตัว เท่านั้น ไม่สามารถส่งย้อนกลับจากอุปกรณ์ 10 ตัวได้ หรือกล่าวได้ว่าการสื่อสารข้อมูลตามมาตรฐาน RS-422A นั้นเป็น การสื่อสารข้อมูลแบบ Simplex คือทิศทางข้อมูลเป็นแบบทางเดียวตลอดเวลา

ดังนั้นถ้าต้องการออกแบบระบบให้เป็นโครงข่ายข้อมูลก็ไม่สามารถทำได้ จึงมีการพัฒนา มาตรฐานการสื่อสารข้อมูลแบบใหม่เพื่อรองรับความต้องการนี้ คือมาตรฐาน RS-485 ซึ่งเป็น มาตรฐานที่อาศัยหลักการของสัญญาณดิฟเฟอเรนเชียลเช่นเดียวกับมาตรฐาน RS-422A แต่สามารถ สื่อสารข้อมูลได้ทั้งใน 2 ทิศทางในสายสัญญาณเพียงคู่เดียว ซึ่งการสื่อสารข้อมูลแบบ Half-duplex จากผลของการใช้สัญญาณในลักษณะดิฟเฟอเรนเชียลนี้ทำให้ระยะทางและความเร็วในการสื่อสาร ข้อมูลมีค่าสูงเช่นเดียวกับมาตรฐานการสื่อสารข้อมูล RS-422A แต่มาตรฐานการสื่อสาร RS-485 สามารถที่จะสื่อสารระหว่างอุปกรณ์ทั้งการรับและการส่งได้สูงสุด 32 ตัว หรืออาจกล่าวได้ว่า การ สื่อสารตามมาตรฐาน RS-485 เป็นการสื่อสารแบบหลายจุด(multipoint communication)โครงสร้าง การสื่อสารข้อมูลแบบ RS-485 แสดงได้ดังรูป 3.8

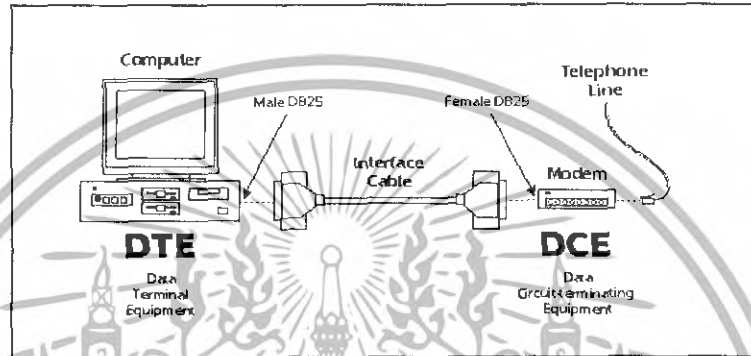
พารามิเตอร์	RS-232C	RS-423A	RS-422A	RS-485
โหมดการทำงาน	Single-ended	Single-ended	Differential	Differential
จำนวนของตัวรับ และตัวส่งที่ยอมรับได้	1 ตัวส่ง 1 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	32 ตัวรับ 32 ตัวส่ง
ความยาวของกลุ่มสาย สัญญาณรับส่งข้อมูล	50 ฟุต	4,000 ฟุต	4,000 ฟุต	4,000 ฟุต
อัตราการส่งข้อมูล สูงสุด(เปิดต่อวินาที)	20 k	100 k	10 M	10 M
แรงดันไฟฟ้าโหมด ร่วมสูงสุด	±2.5 V	±6 V	6 V -2.5 V	12 V -7V
Driver output	±5 V ต่ำสุด ±15 V สูงสุด	±3.6 V ต่ำสุด ±6.0 V สูงสุด	±2 V ต่ำสุด	±1.5 V ต่ำสุด
Driver load Ω	3 k ถึง 7 k	450 ต่ำสุด	100 ต่ำสุด	60 ต่ำสุด
Driver slew rate	30 V/ μ S	-	NA	NA
กระแสลิมิตเมื่อวงจร เอาต์พุตลัดวงจร	500 mA ลัดวงจรกับ Vcc หรือ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND 250 mA ลัดวงจรกับ 8 V หรือ 12 V
ค่าความต้านทาน เอาต์พุตของตัวส่ง Ω	NA – power on 300 – power off	NA – power on 60 k – power off	NA – power on 60 k – power off	120 k power on, off
ความต้านทานทาง อินพุตของตัวรับ Ω	3 k ถึง 4 k	4 k	4 k	12 k
ความไวของตัวรับ	± 3 V	± 200 mV	± 200 mV	± 200 mV

ตารางที่ 3.1 การเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 หัวต่อที่ใช้มาตรฐาน RS-232C (Conector)

มาตรฐาน RS-232C ได้ถูกตีพิมพ์โดย EIA (Electronic Industries Association) ในปี ค.ศ. 1969 ตัวอักษร RS แทน “Recommended Standard”, 232 แทนหมายเลขของมาตรฐาน ส่วนอักษร C แสดง โดยได้กล่าวถึงการสื่อสารข้อมูลระหว่าง Data Terminal Equipment (DTE) และ Data Communication Equipment (DCE) (แต่ในปัจจุบันด้วยย่อ DCE จะแทน data circuit terminating equipment) โดยมีคำจำกัดความดังจะกล่าวต่อไปนี้



รูปที่ 3.9 ตัวอย่างการเชื่อมต่อแบบอนุกรม มาตรฐาน RS-232

DCE : อุปกรณ์ที่มีฟังก์ชันการทำงานต่างๆที่ทำให้เกิดการเชื่อมต่อ,ทำให้การเชื่อมต่อยังคงดำรงต่อไป และยุติการเชื่อมต่อ นอกจากนี้ยังใช้เปลี่ยนลักษณะของสัญญาณและสร้างรหัสสัญญาณต่างที่จำเป็นต่อการสื่อสารข้อมูลระหว่าง DTE (data terminal equipment) และ data circuit โดย DCE อาจเป็นส่วนใดส่วนหนึ่งของคอมพิวเตอร์หรือไม่ก็ได้

DTE :

1. เป็นอุปกรณ์ที่ประกอบไปด้วยตัวส่งข้อมูล (data source) หรือตัวรับข้อมูล (data sink) หรือเป็นทั้งตัวส่งและตัวรับข้อมูลก็ได้
2. เป็นอุปกรณ์ที่ประกอบด้วย function unit ต่อไปนี้ control logic, buffer store และอุปกรณ์อินพุทหรือเอาต์พุตจำนวนหนึ่งหรือมากกว่าก็ได้ หรือรวมเครื่องคอมพิวเตอร์เข้าไปด้วยก็ได้ DTE อาจจะรวมส่วน error control, synchronization และความสามารถในการบ่งหรือระบุว่าการเกี่ยวข้องกับอุปกรณ์ตัวใด (station identification capability) เข้าไปด้วยก็ได้

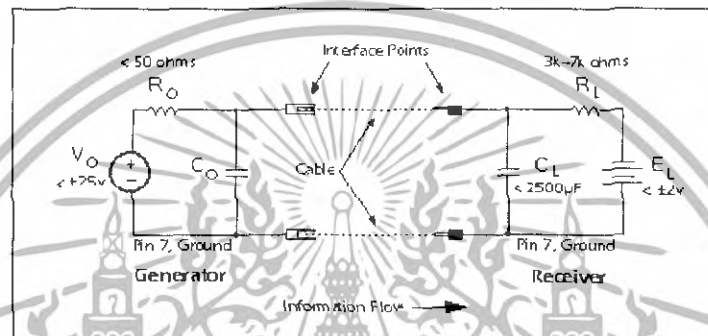
ลักษณะของ DCE และ DTE ที่ใช้ในการสื่อสารข้อมูลสามารถแสดงได้ดังรูป..... ตามลักษณะการทำงานที่ได้อธิบายไว้ข้างต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาตรฐาน RS-232 อัตราส่งข้อมูลจะถูกกำหนดให้อยู่ระหว่าง 0 ถึง 20,000 บิตต่อวินาที (bps) ในการประยุกต์ใช้งาน RS-232 อัตราเร็วสูงสุดที่ใช้ควรจะมีค่าไม่เกิน 19.2 kbps มีความยาวของสายเคเบิลที่ใช้ในการสื่อสารข้อมูลไว้ไม่เกิน 50 ฟุต (อาจยาวกว่านี้ก็ได้ ถ้าเรารู้สภาพแวดล้อมของสายเคเบิลและอยู่ในเงื่อนไขที่กำหนดไว้ในมาตรฐาน

3.5.1 คุณสมบัติของสัญญาณไฟฟ้า

คุณสมบัติของสัญญาณไฟฟ้าที่ใช้กับมาตรฐาน RS-232C มีดังนี้



รูปที่ 3.10 แสดงคุณสมบัติทางไฟฟ้าของมาตรฐาน RS-232

1. สัญญาณที่ขาทุกขาที่หัวต่อ(Connector) จะเป็นสถานะ(Status) โดสถานะหนึ่งในแต่ละคู่ต่อไปนี้

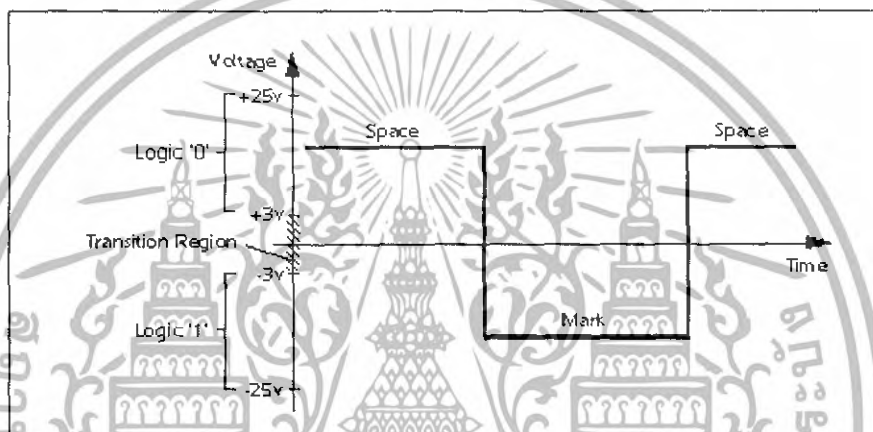
Mark/Space
On/Off
Logic 0/Logic 1

ความสัมพันธ์ระหว่างสถานะของสัญญาณคู่ต่างๆ กับระดับแรงดันได้แสดงได้แสดงไว้ในตารางที่ 3.2 จะเห็นว่า RS-232C ใช้ลจิกกลับ (negative logic) แทนระดับลอจิกต่าง(ลจิกกลับ คือวิธีการเปรียบเทียบแรงดันแบบหนึ่ง ถ้าระดับแรงดันหนึ่งมีค่าเป็นลบมากกว่าอีกระดับหนึ่ง ระดับแรงดันที่มีค่าลบมากกว่าจะเป็นลอจิก “สูง” ดังนี้ 1 = -V ส่วน 0 = +V) โดยแรงดันของระดับสัญญาณต่างๆจะถูกวัดเทียบกับกราวด์ นอกจากนี้ช่วงของระดับแรงดันระหว่าง -3 ถึง +3 โวลต์จะเป็นช่วงของการเปลี่ยนแปลงลอจิก ดังนั้นจึงไม่มีการระบุสถานะของสัญญาณในช่วงนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

ตารางที่ 3.2 ความสัมพันธ์ระหว่างสถานะคู่ต่างของสัญญาณ



รูปที่ 3.11 ขนาดของแรงดันในระดับลอจิก 0 และลอจิก 1

2. ในการแทนลอจิก 1 ตัวขับสัญญาณ(driver)ต้องจ่ายแรงดันระหว่าง -5 ถึง -15 โวลต์ ส่วนในลอจิก 0 ตัวขับสัญญาณ(driver)ต้องจ่ายแรงดันระหว่าง +5 ถึง +15 โวลต์

จากข้อ 1 และ 2 แสดงว่า RS-232C ขอมให้มี noise margin ได้ไม่เกิน 2 โวลต์ สำหรับความสัมพันธ์ระหว่างระดับแรงดันและสถานะของสัญญาณได้แสดงไว้ในรูปที่..... จากรูปจะเห็นว่า ถ้าตัวกำเนิดสัญญาณ(line driver) ต้องการส่งลอจิก 0 จะต้องจ่ายแรงดันระหว่าง +5 ถึง +15 โวลต์ ส่วนตัวรับสัญญาณ(line receiver) จะถือว่าแรงดันที่อยู่ภายในช่วง +3 ถึง +15 โวลต์ แทนลอจิก 0 จากการเปรียบเทียบระดับสัญญาณของตัวส่งและตัวรับ จะเห็นว่า RS-232C ขอมให้มีการลดลงของสัญญาณในช่วง 2 โวลต์ได้ สำหรับการส่งลอจิก 1 ก็เป็นเช่นเดียวกัน

สาเหตุที่ต้องใช้แรงดันระหว่าง ± 3 โวลต์ ถึง ± 15 โวลต์ แทนที่จะใช้ สถานะลอจิกแบบ TTL เพราะสถานะลอจิกแบบ TTLถูกรบกวนได้จากสัญญาณรบกวนต่างๆ ได้ง่าย นอกจากนี้ยังมีปัญหาเกี่ยวกับระยะทางที่สามารถทำการสื่อสารข้อมูลอีกด้วย และในขณะที่กำลังพัฒนา RS-232C ขึ้นนั้น ในวงจะคอมพิวเตอร์ต่างๆ โดยทั่วไปไม่มีการใช้ระดับแรงดันในช่วง ± 3 โวลต์ ถึง ± 15

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โวลต์ อยู่ หนึ่งทรานซิสเตอร์ที่มีขายกันทั่วไปนั้นสามารถทำงานได้ในช่วงแรงดันเหล่านี้ และยังทนต่อสัญญาณรบกวนต่างๆที่มีเข้ามาได้ นอกจากนี้ยังสามารถทำงานที่ความถี่สูงๆได้ สูงถึง 20,000 บิตต่อวินาที ยิ่งไปกว่านั้นสถานะมาร์ก(Mark) และ สเปส(Space) ยังถูกแทนด้วยการไหลของกระแสในทิศทางที่ตรงกันข้ามและมีความแตกต่างแรงดันกันถึง 6 โวลต์เป็นอย่างน้อย ข้อดีเหล่านี้ช่วยให้การส่งข้อมูลมีเสถียรภาพดี

3. ตัวเก็บประจุที่ต่อขนานกับอุปกรณ์รับข้อมูลปลายทางจะต้องมีค่าไม่เกิน 2,500 พิโกฟารัด (pF) โดยค่านี้ไม่รวมค่าความจุไฟฟ้าของสายเคเบิลเข้าไปด้วย(ตามที่ระบุนี้ ระยะทางที่สามารถใช้ทำการสื่อสารข้อมูลจะต้องไม่เกิน 50 ฟุต)

4.แรงดันขณะเปิดวงจรหรือขณะที่ไม่มีโหลด(V_o) จะต้องไม่เกิน 25 โวลต์ ซึ่งก็คือแรงดันใดๆ ในวงจรของการอินเทอร์เฟซแบบ RS-232C ต้องไม่เกิน 25 โวลต์

5. วงจรรับสัญญาณที่ใช้กับ RS-232C ต้องสามารถทนต่อการลัดวงจรที่เกิดขึ้นได้ โดยไม่ทำให้เกิดความเสียหายต่อตัวมันเองและอุปกรณ์ที่เกี่ยวข้องด้วย เช่น หัวต่อ, โมเด็ม, พอร์ต อินพุต/เอาต์พุต และอุปกรณ์ต่างๆที่ต่อเข้ากับเคเบิลที่ใช้ในการอินเทอร์เฟซแบบ RS-232C

3.5.2 หัวต่อ(Connector)

หัวต่อที่ใช้ยูนิตนั้นมีอยู่ 2 แบบคือ แบบ 25 ขา(DB-25 pin) และแบบ 9 ขา (DB-9 pin) ซึ่งทั้ง 2 แบบ จะเป็นชนิดตัวผู้ทางด้านของคอมพิวเตอร์ ดังนั้นอุปกรณ์ที่จะนำมาต่อกับคอมพิวเตอร์ จึงต้องใช้หัวต่อชนิดตัวเมีย



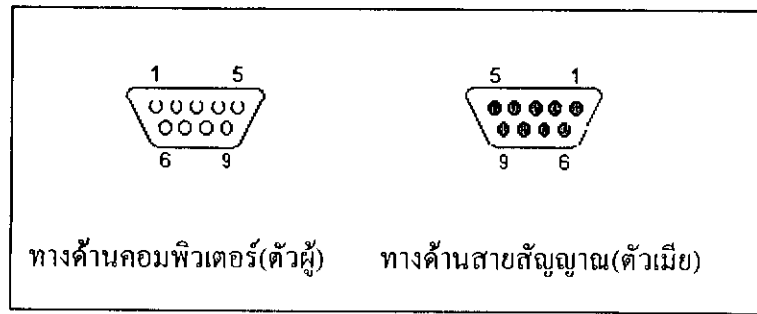
รูปที่ 3.12 หัวต่อ 25 ขา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขา	สัญลักษณ์	ทิศทางของสัญญาณ	ชื่อขาสัญญาณ
1	SHIELD	—	Shield or Protective Ground
2	TXD	→	Transmit Data
3	RXD	←	Receive Data
4	RTS	→	Request to Send
5	CTS	←	Clear to Send
6	DSR	←	Data Set Ready
7	GND	—	Signal Ground
8	CD	←	Carrier Detect
9	n/c	-	Reserved for Data Set Testing
10	n/c	-	Reserved for Data Set Testing
11	n/c	-	Unassigned
12		←	Secondary Carrier Detect
13		←	Secondary Clear to Send
14		→	Secondary Transmit Data
15		←	Transmit Signal Element Timing (DCE)
16		←	Secondary Receive Data
17		←	Receive Signal Element Timing (DCE)
18	n/c	-	Unassigned
19		→	Secondary Request to Send
20	DTR	→	Data Terminal Ready
21		←	Signal Quality Detect
22	RI	←	Ring Indicator
23		↔	Data Signal Rate Select (DCE/DTE)
24		→	Transmit Signal Element Timing (DTE)
25	n/c	-	Unassigned

ตารางที่ 3.3 แสดงขาต่างๆของหัวต่อตัวผู้ชนิด 25 ขา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.13 หัวต่อ 9 ขา

ขา	สัญลักษณ์	RS232	V.24	ทิศทางของสัญญาณ	ชื่อขาสัญญาณ
1	CD	CF	109	←	Carrier Detect
2	RXD	BB	104	←	Receive Data
3	TXD	BA	103	→	Transmit Data
4	DTR	CD	108.2	→	Data Terminal Ready
5	GND	AB	102	—	Signal Ground
6	DSR	CC	107	←	Data Set Ready
7	RTS	CA	105	→	Request to Send
8	CTS	CB	106	←	Clear to Send
9	RI	CE	125	←	Ring Indicator

ตารางที่ 3.3 แสดงขาต่างๆของหัวต่อตัวผู้ชนิด 9 ขา

หน้าที่ของสัญญาณต่าง ๆ มีดังนี้

โปรเทกทีฟกราวด์(Protective ground) เป็นจุดที่ต่อกับตัวเปลือกของอุปกรณ์ และไม่ต่อกับสัญญาณใด ๆ ในระบบ เพื่อใช้ต่อลงดิน เป็นการป้องกันอันตรายจากไฟลัดวงจร และใช้เป็น Shield ป้องกันการรบกวน

กราวด์ของสัญญาณ(Signal ground หรือ Common return) เป็นจุดสำคัญที่สุดที่ต้องมีในระบบ RS-232 เพราะเป็นจุดอ้างอิงของทุกสัญญาณ ยกเว้น Protective ground

ขาส่งข้อมูล(Transmit data) เป็นสัญญาณข้อมูลที่ส่งจาก DTE ไปยัง DCE ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็น Logic "1" หรือ "Mark" หรือ "Off" หรือ -5V ถึง -15V ที่ด้านส่ง (DTE) หรือ -3V ถึง -15V ที่ด้านรับ (DCE) การส่งข้อมูลจะเกิดขึ้นได้ต้องมีสัญญาณควบคุมที่เกี่ยวข้อง "On" ก่อนคือสัญญาณ RTS, CTS, (D)CD, DTR และ DSR

ขารับสัญญาณ(Recieve data) เป็นสัญญาณข้อมูลจาก DCE มายัง DTE ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็น ลอจิก "1" หรือ "Mark" หรือ "Off" หรือ -5V ถึง -15V ที่ด้านส่งหรือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-3V ถึง -15V ที่ด้านรับ ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็นลอจิก "1" หรือ "Mark" หรือ "Off" หรือ -5V ถึง -15V ที่ด้านส่ง (DCE) หรือ -3V ถึง -15V ที่ด้านรับ (DTE) กรณีที่เป็น การสื่อสารแบบฮาร์ฟดูเพล็กซ์(Half-duplex) สัญญาณ RD จะอยู่สถานะ "Off" ขณะที่สัญญาณ RTS อยู่ในสถานะ "On" และสัญญาณ RD จะยังคงอยู่ในสถานะ "Off" อีกชั่วระยะเวลาหนึ่งหลังจากที่สัญญาณ RTS เปลี่ยนจากสถานะ "On" มาเป็น "Off" แล้วเพื่อให้การรับ-ส่งข้อมูลเสร็จสมบูรณ์

สัญญาณร้องขอส่งข้อมูล(Request to send) เป็นสัญญาณจาก DTE ส่งไปให้ DCE เพื่อขอส่งข้อมูลไป ประกติจะอยู่ในสถานะ "Off" เมื่อต้องการส่งข้อมูลจะเปลี่ยนเป็นสถานะ "On" จนกว่า การส่งเสร็จสิ้นจึงเปลี่ยนกลับมาที่สถานะ "Off" ตามเดิม ทั้งนี้ทาง DTE ต้องได้รับสัญญาณ CTS จึงจะสามารถส่งข้อมูล TD ไปยัง DCE ได้ และสัญญาณ RTS ที่กลับสู่สถานะ "Off" จะไม่สามารถ เปลี่ยนเป็น "On" ใหม่ขณะที่สัญญาณ CTS อยู่ในสถานะ "On" ต้องรอจนกว่าสัญญาณ CTS เปลี่ยน มาอยู่ในสถานะ "Off" ก่อน เพื่อป้องกันการเกิด Overrun

ขาสัญญาณพร้อมรับส่งข้อมูล(Clear to send) เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่า พร้อมรับการส่งข้อมูลจาก DTE

ขาสัญญาณพร้อมที่จะติดต่อ(Data set ready) เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่า DCE สามารถเชื่อมโยงไปยังปลายทางและพร้อมที่จะติดต่อแล้ว

ขาสัญญาณพร้อมสื่อสาร(Data Terminal ready) เป็นสัญญาณที่ DTE ส่งให้ DCE เพื่อแจ้ง ว่า DTE พร้อมหรือต้องการจะติดต่อสื่อสาร ซึ่งสัญญาณ DTR นี้ต้องเกิดก่อนทาง DCE จึงจะทำการติดต่อไปยังปลายทางและเมื่อติดต่อได้แล้วจึงส่งสัญญาณ DSR มายัง DTE เพื่อแจ้งให้รู้ว่าพร้อม รับการสื่อสารแล้ว และถ้า DTR เปลี่ยนเป็น Off แปลว่า DTE ไม่ต้องการติดต่อสื่อสารแล้วทาง DCE ก็จะปิดช่องสื่อสารและเปลี่ยนสัญญาณ DSR เป็น Off ทั้งนี้ คู่สัญญาณระหว่าง RTS กับ CTS เป็นเรื่องของความพร้อมเกี่ยวกับช่องสื่อสารระหว่าง DTE กับ DCE ส่วนคู่สัญญาณ DTR กับ DSR เป็นเรื่องของความพร้อมเกี่ยวกับตัวอุปกรณ์ DTE กับ DCE

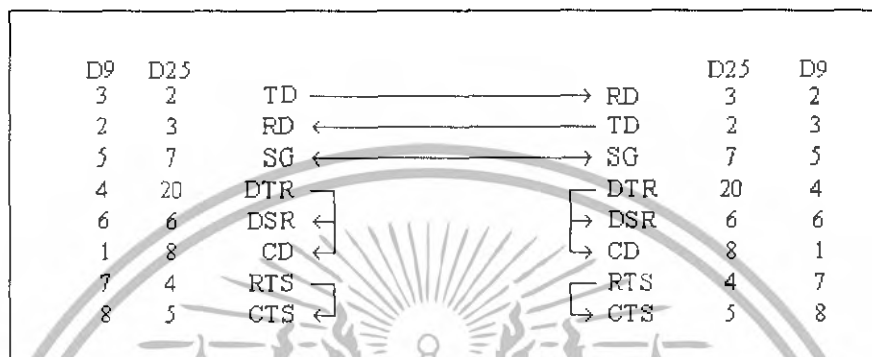
สัญญาณแจ้งผลการรับสัญญาณ(Data carrier detect) เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่าได้รับสัญญาณพาหะจาก DCE ที่อยู่อีกด้านหนึ่งของการสื่อสาร ซึ่งหมายความว่า ช่อง การสื่อสารระหว่าง DCE ทั้ง 2 ไม่ขาดตอน พร้อมที่จะทำการสื่อสารได้ ซึ่งอุปกรณ์ DTE หรือ Software ที่ควบคุมการสื่อสารมักจะตรวจสอบสัญญาณนี้ ถ้าไม่อยู่ที่ On แสดงว่าช่องการสื่อสารขาด ก็ จะไม่ทำการรับหรือส่งข้อมูล

สัญญาณเรียก(Ring Indicator) เป็นสัญญาณจาก DCE แจ้งให้ DTE รู้ว่ามีการเรียกจาก DCE ที่อยู่อีกด้านหนึ่งของการสื่อสาร ซึ่งมักจะใช้ในระบบ Automatic answering

3.5.3 ลักษณะการต่อ RS-232 ที่ไม่ใช่แบบมาตรฐาน

3.5.3.1 โมดโมเด็ม(Null Modems)

โมดโมเด็มใช้สำหรับเชื่อมโยงระหว่าง DTE 2 ตัวเข้าด้วยกันโดยตรงซึ่งโดยมากใช้ในการถ่ายข้อมูลระหว่าง Computer หรือใช้ในการพัฒนาระบบ Microprocessor หรือ Microcontroller ต่าง ๆ



รูปที่ 3.14 แสดงการต่อแบบโมดโมเด็ม

จะเห็นว่าใช้สายเพียง 3 เส้นคือ TD, RD และ SG และมีการต่อ Jump ที่หัวต่อแต่ละข้างอีกเล็กน้อยเท่านั้น เพื่อหลอกให้คอมพิวเตอร์คิดว่ามันกำลังคุยกับ DCE อยู่

หมายเหตุ TD ของคอมพิวเตอร์เครื่องหนึ่งจะต้องต่อกับ RD ของ คอมพิวเตอร์ อีกเครื่องหนึ่ง แต่ถ้าเป็นการต่อระหว่าง DTE กับ DCE จริง ๆ จะต่อขาสัญญาณตรงกันโดยไม่ต้องสลับสายเช่นขา 2 ของ DTE จะต่อกับขา 2 ของ DCE และขา 3 ของ DTE จะต่อกับขา 3 ของ DCE เป็นต้น

3.5.3.2 Loopback Plugs

สำหรับ Plug ตัวนี้มีวัตถุประสงค์เพื่อช่วยในการเขียน Program ที่ติดต่อกับ Serial Port แต่อาจจะไม่สามารถใช้กับ Program ที่ใช้วิเคราะห์หาจุดเสีย (Diagnostic program) เพราะแต่ละ Program อาจจะมีรูปแบบการเชื่อมต่อที่แตกต่างกันไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

LoopBack Plug		
D9	D25	
3	2	TD
2	3	RD
5	7	SG
4	20	DTR
6	6	DSR
1	8	CD
7	4	RTS
8	5	CTS

รูปที่ 3.15 แสดงการต่อแบบลูปแบ็กปลั๊ก

3.5.4 DTE/DCE Speeds

ความเร็วของการติดต่อระหว่าง DTE กับ DCE หรือระหว่างคอมพิวเตอร์กับโมเด็ม ซึ่งมักจะเรียกว่า เทอร์มินอล สปีด (Terminal speed) และความเร็วระหว่าง DCE กับ DCE หรือระหว่างโมเด็ม กับโมเด็มซึ่งมักจะเรียกว่าไลน์ สปีด (Line speed) ปกติ เทอร์มินอล สปีด มักจะเร็วกว่าไลน์ สปีด เช่น โมเด็ม มี มีความเร็วบอด 56 กิโลบิต (speed 56K baud) ขณะที่ติดต่อกับคอมพิวเตอร์ด้วย มีความเร็วบอด 56 กิโลบิต (speed 115.2Kbps) เป็นต้น ซึ่ง คอมพอร์ต (COM Port) ของคอมพิวเตอร์ ต้องเซต ให้เป็น 115.2Kbps ด้วย ปัจจุบัน UART เบอร์ 16550A จะรองรับได้ 115.2Kbps และเบอร์ 16C650 จะรองรับความเร็วได้ 230.4Kbps

3.5.5 Flow Control

การติดต่อระหว่าง DTE และ DCE จะมีการควบคุมการไหลของข้อมูล เพื่อไม่ให้เกิดการ โอเวอร์โฟลว(Overflow) ขึ้นได้ ซึ่งมีอยู่ 2 แบบคือ ฮาร์ดแวร์ โฟลว คอนโทรล(Hardware flow control) และ ซอฟต์แวร์โฟลว คอนโทรล(Software flow control) สำหรับ ซอฟต์แวร์โฟลว คอนโทรล มักจะเรียกว่า Xon/Xoff flow control ซึ่งใช้รหัส ASCII 17 เป็นสัญญาณ Xon และใช้รหัส ASCII 19 เป็นสัญญาณ Xoff หลักการทำงานที่ง่าย ๆ คือโมเด็ม จะมี บัฟเฟอร์ อยู่ เมื่อ โมเด็ม รับข้อมูลจาก คอมพิวเตอร์ จน บัฟเฟอร์ โดดเต็ม มันก็จะส่งสัญญาณ Xoff ไปให้ คอมพิวเตอร์ เพื่อให้คอมพิวเตอร์หยุดส่งข้อมูลให้มันชั่วคราว และเมื่อบัฟเฟอร์ มีที่ว่างถึงระดับหนึ่งโมเด็ม ก็จะส่งสัญญาณ Xon ไปให้คอมพิวเตอร์ เพื่อให้คอมพิวเตอร์ ส่งข้อมูลให้มันต่อ การควบคุมโดยวิธีนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประหยัดสายสัญญาณ เพราะรับ-ส่งผ่าน TD และ RD แต่อาจทำให้การสื่อสารช้าลงอย่างเห็นได้ชัด ในกรณีที่ใช้กับการสื่อสารที่มีความเร็วต่ำ เพราะแต่ละตัวอักษร ASCII ที่รับ-ส่งจะมีขนาด 10 Bit ส่วนฮาร์ดแวร์คอนโทรล มักจะเรียกว่า RTS/CTS flow control จะใช้สายสัญญาณของ พอร์ตอนุกรม ในการควบคุม ทำให้ไม่บั่นทอนความเร็วของข้อมูล หลักการทำงานคือ เมื่อ Modem มีที่ว่างเพื่อรับข้อมูล มันก็จะส่งสัญญาณ CTS ไปให้คอมพิวเตอร์ และเมื่อมันใกล้จะเต็ม มันก็จะหยุดส่งสัญญาณ CTS ไปให้คอมพิวเตอร์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

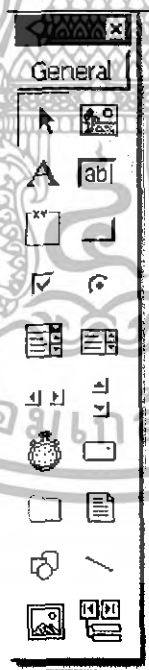
บทที่ 4

การใช้งาน Visual Basic

4.1 การใช้งานคอนโทรลในการสร้างอินเตอร์เฟซ

จุดเริ่มต้นของการพัฒนาแอปพลิเคชันด้วย VB ก็คือการนำคอนโทรลชนิดต่างๆ ที่ VB จัดเตรียมไว้นามาวาดอินเตอร์เฟซ แอปพลิเคชันจะมีหน้าต่าง เป็นอย่างไร ก็อยู่ในขั้นตอนนี้ ถ้าเริ่มต้นพัฒนาแอปพลิเคชันได้ดี โดยการออกแบบอินเตอร์เฟซที่ใช้งานง่าย เป็นมิตรกับผู้ใช้ มันจะส่งผลให้ระยะเวลา ในการพัฒนาแอปพลิเคชันหนึ่ง ลดลงไปได้มากที่สุด เพราะสิ่งที่เหลืออยู่คือการเขียนโค้ดเพื่อให้โปรแกรมทำงานให้สมบูรณ์มากที่สุด ผลงานก็จะออกมาเหมาะสมกับเวลาที่ ต้องเสียไปคอนโทรลแต่ละชนิดจะมีหน้าที่ หรือจุดประสงค์ในการนำไปใช้งานต่างกัน เช่น คอนโทรล Command Button ใช้สร้างปุ่มกดเพื่อตอบรับ, คอนโทรล Option Button ใช้สำหรับให้ ผู้ใช้มีทางเลือก, คอนโทรล TextBox ใช้รับและแสดงข้อมูลที่ผู้ใช้สามารถแก้ไขได้, คอนโทรล Label ใช้แสดงข้อมูลที่ผู้ใช้ไม่สามารถแก้ไขได้ เป็นต้น

ถ้ารับชื่อของคอนโทรลแต่ละตัว คุณสามารถเลื่อนเมาส์ไปบริเวณบนตัวคอนโทรลนั้นๆ แล้วจะมี ToolTip แสดงชื่อของคอนโทรลนั้นๆ ให้ผู้ใช้ทราบ



รูปที่ 4.1 แสดงกลุ่มคอนโทรลมาตรฐานของ VB

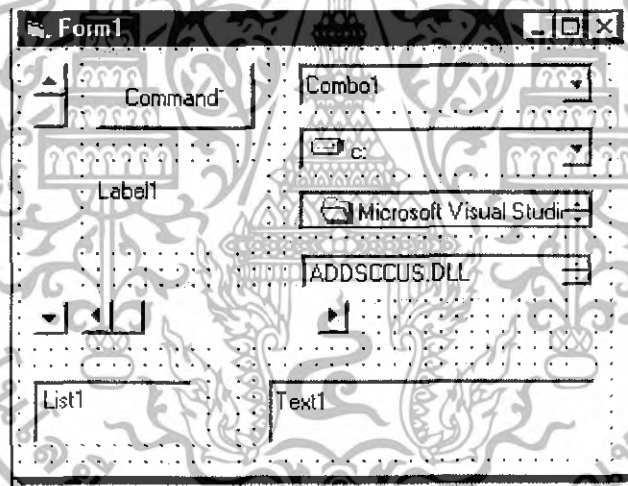
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 รูปแบบการปรากฏของคอนโทรลแต่ละชนิดบนฟอร์ม

คอนโทรลแต่ละชนิดมีรูปแบบการปรากฏตัว มีความแตกต่างกัน คอนโทรลบางตัวเมื่อนำมาใช้งานอาจจะไม่มีการปรากฏตัวขึ้นมา หรืออาจจะปรากฏขึ้นมาเลขก็ได้ ซึ่งจะขึ้นอยู่กับหน้าที่ของแต่ละคอนโทรล สำหรับคอนโทรลที่คุณจะต้องใช้งานในทุกๆ แอปพลิเคชันจะมี เช่น CommandButton, TextBox, CheckBox, OptionButton, ScrollBar เป็นต้น คอนโทรลพวกนี้ถือได้ว่าเป็น คอนโทรลคลาสสิก หมายถึง เป็นคอนโทรลที่พบเห็นได้ทั่วไปสำหรับรูปแบบการปรากฏตัวของคอนโทรลสามารถแยกออกได้เป็น 2 แบบ คือ

1. ปรากฏตัวทันทีเมื่อนำมาใช้งาน หมายถึง สามารถเห็นได้ในขณะที่ออกแบบ ที่เรียกว่า design time ซึ่งคอนโทรลส่วนใหญ่ จะมีลักษณะเช่นนี้

2. ไม่ปรากฏตัวเมื่อคุณนำมาใช้งาน หมายถึง คอนโทรลประเภทที่ไม่สามารถมองเห็นการทำงานของตัวมันในขณะที่ออกแบบได้ มันจะทำงานอยู่เบื้องหลัง แอปพลิเคชันของคุณ ซึ่งจะเป็นเวลาในช่วงของการรันแอปพลิเคชัน ที่เรียกว่า run time เช่น คอนโทรล Timer, CommonDialog เป็นต้น



รูปที่ 4.2 แสดงรูปแบบของคอนโทรลหลายๆชนิดบนฟอร์ม

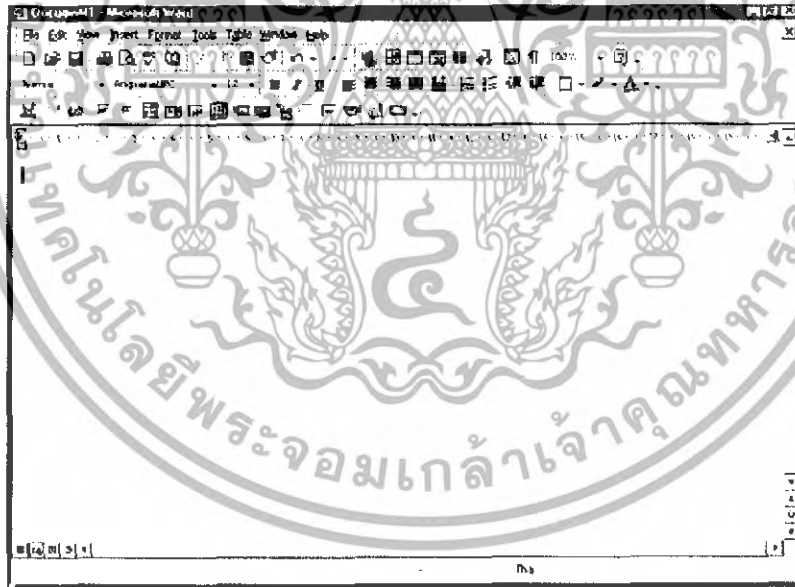
4.3 แนวทางในการออกแบบอินเตอร์เฟซแรก

สำหรับการออกแบบอินเตอร์เฟซ VB อนุญาตให้สามารถที่จะวาดคอนโทรลได้อย่างอิสระ ไม่มีกฎเกณฑ์ตายตัว ณ ตำแหน่งใดก็ได้ ที่อยู่ในบริเวณฟอร์ม สามารถสร้างสรรค์ให้แอปพลิเคชันให้มีหน้าตาเป็นอย่างไรก็ได้ แต่ขอให้ยึดถือไว้ว่า อินเตอร์เฟซที่ออกมา จะต้องเป็นมิตรกับผู้ใช้ มีรูปแบบเดียวกับแอปพลิเคชันต่างๆไป ไม่ทำให้ผู้ใช้สับสนกับแอปพลิเคชัน ยกตัวอย่างเช่น ถ้าออกแบบอินเตอร์เฟซให้ผู้ใช้ ต้องตอบคำถามมากมาย กว่าจะได้เซฟงาน, มีตัวเลือกมากจนกระทั่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

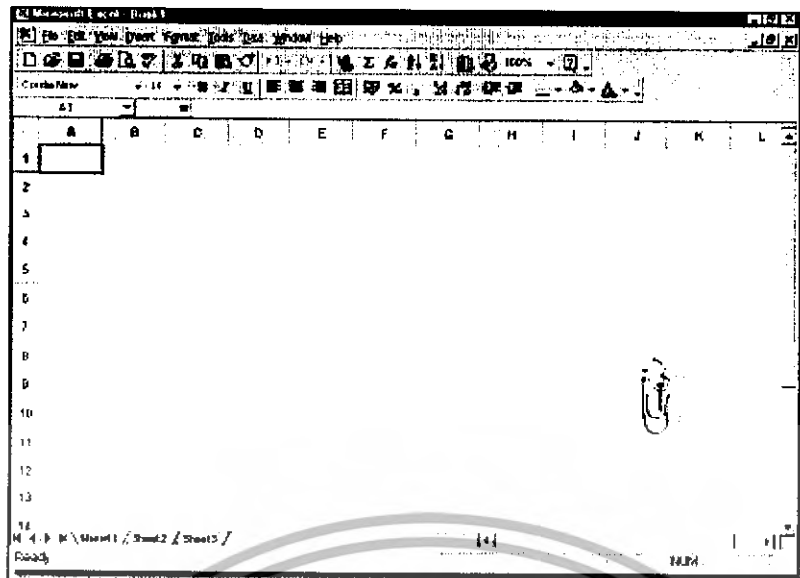
ผู้ใช้ไม่รู้ว่าจะเลือกอะไรดี, จัดวาง Lay out ไม่เป็นระเบียบ ไม่เป็นกลุ่ม, ทำให้ผู้ใช้สับสนในการใช้งาน จากแอปพลิเคชันต่างๆ ไปจะเห็นได้อย่างหนึ่งว่า จะพยายามเน้นที่การใช้งานง่าย รวดเร็ว และมีประสิทธิภาพ ซึ่งเป็นการอำนวยความสะดวกให้ผู้ใช้เป็นอันดับแรก แต่การต่างๆ จะอยู่ที่โปรแกรมเมอร์ต่างหาก สิ่งที่จะสร้างแนวความคิดให้ออกแบบอินเตอร์เฟซได้อย่างรวดเร็วก็คือ จะต้องมีประสิทธิภาพ ในการใช้งานแอปพลิเคชันในด้านต่างๆ ให้มากที่สุด เท่าที่จะเป็นไปได้ แล้วจะได้ concept ในการสร้างแอปพลิเคชันได้อย่างไม่ยากเย็น

ข้อสังเกตอีกประการหนึ่งคือ ในแอปพลิเคชันต่างๆ ไป เช่น Word, Excel รวมถึงผลิตภัณฑ์ที่ไม่ใช่ของไมโครซอฟท์ จะมีส่วนอินเตอร์เฟซที่มีลักษณะ คล้ายคลึงกันมาก ทำให้ผู้ใช้มีความคุ้นเคยสามารถเรียนรู้ ทำความเข้าใจได้ไม่ยากเย็นนัก อย่างน้อยที่สุด ทำให้ผู้ใช้งานมีความรู้สึกที่ว่า ไม่ต้องเริ่มต้นเรียนรู้ใหม่ มีความสนใจอยากที่จะเรียนรู้ ใช้งานแอปพลิเคชันนั้นๆ มากขึ้น ซึ่งเป็นสิ่งที่โปรแกรมเมอร์จะต้องสื่อออกมากับอินเตอร์เฟซให้ได้ข้อได้เปรียบที่เห็นได้ชัดเจนมากที่สุดในการพัฒนาแอปพลิเคชันด้วย VB นั่นก็คือ สามารถใช้ระยะเวลาสั้นๆ ในการออกแบบอินเตอร์เฟซได้อย่างรวดเร็ว ทำให้สามารถเปลี่ยนแปลงแก้ไขอินเตอร์เฟซให้ตรงกับความต้องการของผู้ใช้งาน ได้อย่างง่ายดาย ซึ่งเป็นการ programming ในสมัยใหม่ ซึ่งจะช่วยย่นระยะเวลาในการพัฒนาแอปพลิเคชันได้เป็นอย่างดี



รูปที่ 4.3 แสดงสภาพแวดล้อมของ Word 2000

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 แสดงสภาพแวดล้อมของ Excel 2000

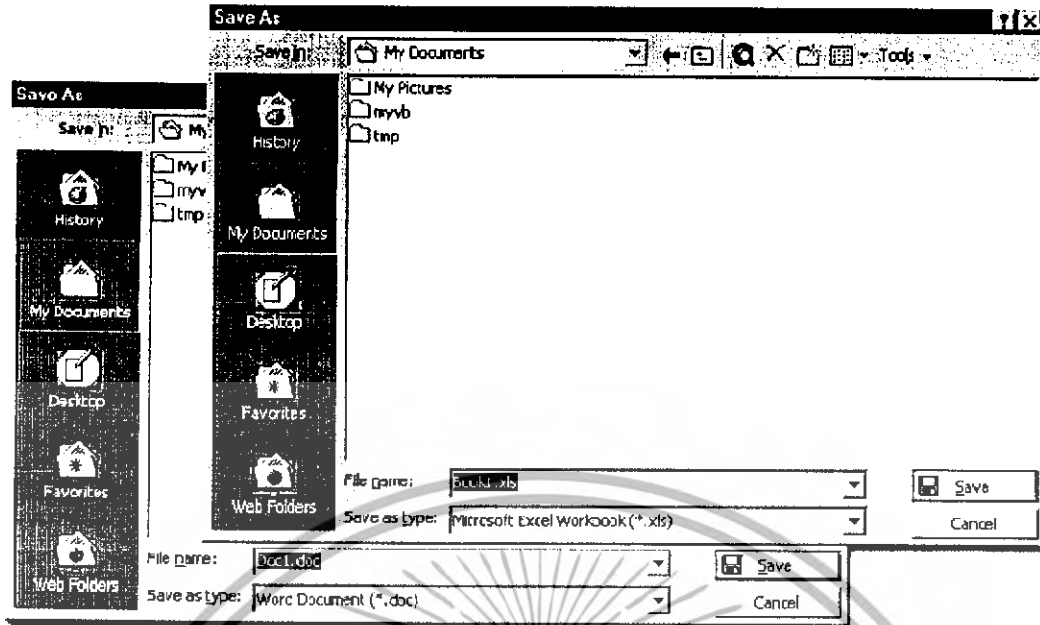
สำหรับแอปพลิเคชันที่ไม่ใช่ผลิตภัณฑ์ของไมโครซอฟท์ ก็ยังคงมีอินเทอร์เน็ตที่ใกล้เคียงกัน จะแตกต่างกันในแง่ของหน้าที่การใช้งานเท่านั้น



รูปที่ 4.5 แสดงสภาพแวดล้อมของ Netscape Composer

องค์ประกอบหลักๆ ของอินเทอร์เน็ต จะประกอบไปด้วย เมนู, ToolBar, dialog box ชนิดต่างๆ เช่น เปิดไฟล์, เซฟไฟล์, พิมพ์งาน เป็นต้น สิ่งเหล่านี้ VB เตรียมไว้ให้คุณแล้ว ที่เหลือก็ขึ้นอยู่กับว่า จะนำไปใช้งานให้เกิดประสิทธิภาพมากเพียงใด

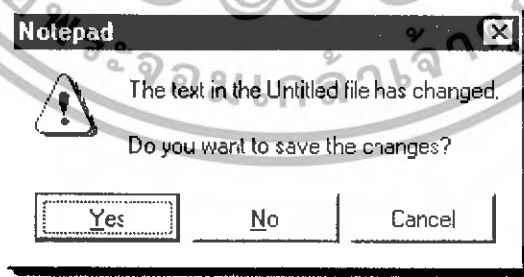
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.6 แสดง ไดอะล็อกบ็อกซ์เซฟไฟล์ของ Excel 2000 และ Word 2000

จะเห็นได้ว่ามีความคล้ายคลึงกัน เนื่องจากใช้งานกลุ่มฟังก์ชัน Windows API ชุดเดียวกัน มีผลทำให้ผู้ใช้มีความรู้สึก ว่าไม่ต้องเริ่มต้นเรียนรู้ใหม่ทั้งหมด และมีความรู้สึกต้องการใช้แอปพลิเคชันนั้นๆ ด้วย

ส่วนประกอบของอินเตอร์เฟซ เกิดจากการใช้คอนโทรลชนิดต่างๆ เข้าด้วยกัน เช่น คอนโทรล CommandButton, Frame, CheckBox เป็นต้น แต่ไม่จำเป็นที่จะต้องออกแบบอินเตอร์เฟซทุกอย่าง ในแอปพลิเคชันหนึ่งๆ VB ได้เตรียมกลุ่มฟังก์ชัน ที่ให้เขียนโค้ดเพียงบรรทัดเดียว ก็สามารถเรียกใช้งานได้แล้ว เช่น message box ที่เตือนให้ผู้ใช้เซฟงาน ก่อนออกจากโปรแกรม, ไดอะล็อกบ็อกซ์เซฟไฟล์, เปิดไฟล์ เป็นต้น



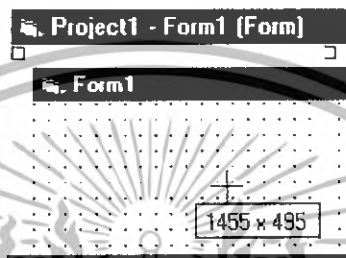
รูปที่ 4.7 แสดง message box ที่เตือนให้ผู้ใช้เซฟงาน ใน VB คุณเขียนโค้ดเพียงบรรทัดเดียวเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

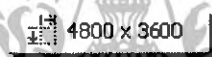
4.4 การนำคอนโทรลมาใช้งาน

สำหรับวิธีการนำคอนโทรลมาใช้งาน ควบคุมอินเตอร์เฟซบนฟอร์ม มี 2 วิธี คือ

1.คลิกที่ตัวคอนโทรลนั้นๆ บน ToolBox แล้วนำไปวางบนฟอร์ม สามารถกำหนดความกว้างและความยาวของคอนโทรลได้อย่างอิสระรวมถึงกำหนดตำแหน่งได้เช่นกัน VB จะแสดง ToolTip แสดงขนาดของคอนโทรล ให้อัตโนมติ ส่วนตำแหน่งของคอนโทรลดูได้จาก บน ToolBar โดยแสดงแบบพิกัด co-ordinate วัดจากมุมซ้ายบนของฟอร์มที่บรรจุอยู่ ถ้าคอนโทรลนั้นอยู่ในคอนโทรล Frame ก็จะวัดจากมุมซ้ายบนของคอนโทรล Frame แทน ดังรูปที่ 4.8 และรูปที่ 4.9



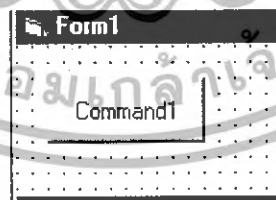
รูปที่ 4.8 แสดง Tooltip บอกขนาดของคอนโทรล



รูปที่ 4.9 แสดงขนาดของคอนโทรลบน Tool Bar

2. ดับเบิลคลิกที่ตัวคอนโทรลนั้นๆ แล้ว VB จะนำคอนโทรลไปวางบนฟอร์มให้โดยอัตโนมัติซึ่ง VB จะตั้งค่า default ไว้ให้ทั้งตำแหน่งและขนาดของคอนโทรล แล้วค่อยแก้ไขภายหลัง

สำหรับคอนโทรล CommandButton อาจใช้ขนาด default ที่ VB ตั้งมาให้ใช้งานได้ เพราะมีขนาดเหมาะสมอยู่แล้ว



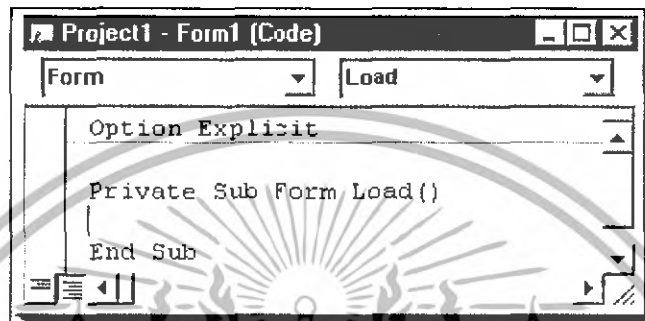
รูปที่ 4.10 แสดงคอนโทรล CommandButton แบบ default

สำหรับการปรับขนาดของคอนโทรล ให้คลิกที่คอนโทรลนั้นๆ เพื่อให้อยู่ในสถานะเอกทีฟ(active) แล้วจะมีปุ่มสี่เหลี่ยมทึบ 8 ปุ่มอยู่บนตัวคอนโทรล ให้เลื่อนเมาส์ไปบริเวณปุ่มทึบ(เคอร์เซอร์เปลี่ยนเป็นลูกศร 2 หัว) ก็สามารถปรับขนาดได้ตามต้องการ สำหรับการเลื่อนตำแหน่งของคอนโทรล สามารถ drag เมาส์ บนคอนโทรลนั้นๆ แล้วย้ายไปยังตำแหน่งที่ต้องการได้โดยตรง ซึ่งก็จะเหมือนกับ การใช้งานแอปพลิเคชันทั่วไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 พื้นฐานการเขียนโค้ด

สำหรับหัวข้อนี้จะเป็นขั้นตอนที่ต่อจากการออกแบบอินเตอร์เฟซ หลังจากที่ได้ออกแบบอินเตอร์เฟซเรียบร้อยแล้ว จะต้องเริ่มเขียนโค้ด สำหรับคอนโทรลแต่ละตัว เพราะเมื่อ ออกแบบอินเตอร์เฟซแล้ว มันยังไม่สามารถทำอะไรได้เลย ใน VB จะมี editor ไว้สำหรับให้เขียนโค้ด โดยเฉพาะ เตรียมไว้ให้แล้ว ดังรูปที่ 4.11



รูปที่ 4.11 แสดง editor ของ VB

มี 2 วิธีที่สามารถเรียก editor ขึ้นมาใช้งานคือ

1. ดับเบิลคลิกที่ตัวคอนโทรลนั้นๆ
2. คลิกที่คอนโทรลนั้น ให้อยู่ในสภาพเอกทิฟ (active) หรือได้รับความสนใจ (focus) แล้ว

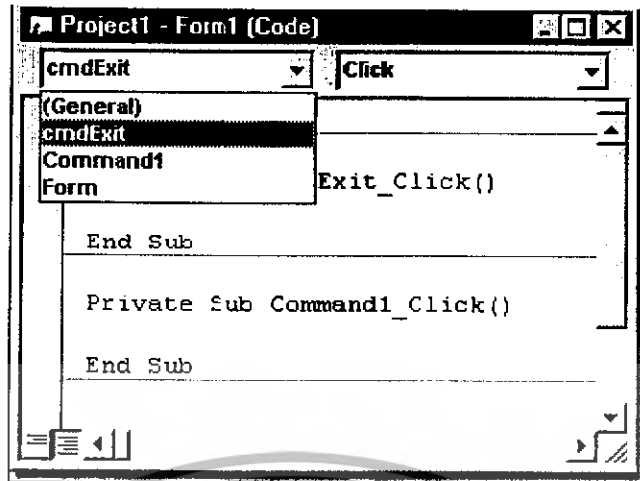
กด F7

4.6 การใช้งาน Editor

เครื่องมือตัวนี้ถือได้ว่าเป็นเครื่องมือที่มีความสำคัญมากที่สุด อีกตัวหนึ่งในบรรดาเครื่องมือที่ VB มี เพราะใช้สำหรับเขียนโค้ด ให้แอปพลิเคชันทำงานได้ (คำว่า editor จะหมายถึง editor ของ VB) เครื่องมือตัวนี้ใช้งานมากที่สุดในขบวนการพัฒนาแอปพลิเคชันด้วย VB การศึกษาสภาพแวดล้อมของ editor จึงมีความสำคัญเป็นอย่างยิ่ง ไม่ยิ่งหย่อนไปกว่าส่วนใดๆ ใน VBIDE การทำความเข้าใจ รูปแบบการใช้งาน และความหมายของส่วนต่างๆ ของ editor จะช่วยลดระยะเวลาในการเขียนโค้ดได้อีกระดับหนึ่ง จากรูปที่ 4.11 สามารถแยกส่วนต่างๆ ของ editor ออกได้เป็น 3 ส่วน ดังนี้

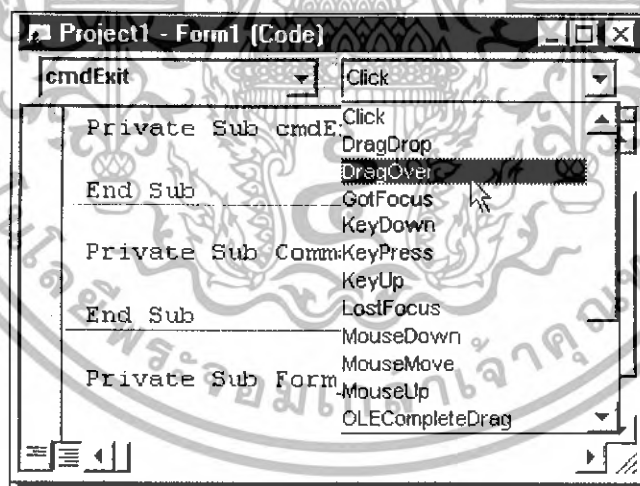
1. ส่วน object list box มีหน้าที่แสดงชื่อคอนโทรลหรืออ็อบเจกต์ที่นำมาใช้งาน เมื่อเพิ่มคอนโทรลเข้ามาในฟอร์ม รายชื่อของคอนโทรล จะถูกเพิ่มเข้ามาโดยอัตโนมัติ ถ้ามีการเปลี่ยนแปลงชื่อของคอนโทรล (คุณสมบัติ Name) ชื่อที่ปรากฏอยู่ใน object list box ก็จะเปลี่ยนไปตามชื่อที่ตั้งไว้เช่นกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.12 แสดงส่วน object list box และคอนโทรล CommandButton ชื่อปกติและ CommandButton ที่เปลี่ยนชื่อแล้ว

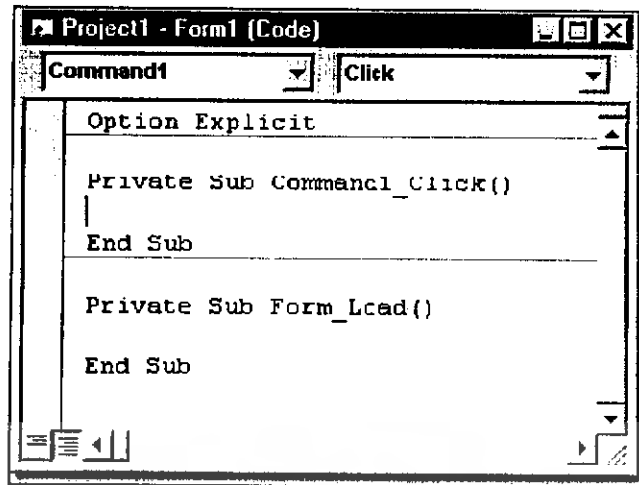
2. ส่วน event list box มีหน้าที่แสดงเหตุการณ์ (event) ของคอนโทรลที่ถูกเลือกใน object listbox สลับสับเปลี่ยนอยู่ มันจะเปลี่ยนแปลงโดยอัตโนมัติ เมื่อเปลี่ยนชนิดของคอนโทรลใน object list box ทั้ง 2 ส่วนนี้จะมีความสัมพันธ์กันตลอดเวลา



รูปที่ 4.13 แสดงส่วน event list box และรายการ events ที่คอนโทรล CommandButton สลับสับเปลี่ยน

3. ส่วนการเขียนโค้ด เมื่อเลือกคอนโทรลใน object list box และเลือกเหตุการณ์ใน event listbox แล้ว VB จะสร้างโพรซีเจอร์ (procedure) ว่างๆ ขึ้นมาให้โดยอัตโนมัติ ดังรูปที่ 4.14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.14 แสดง โพรซีเจอร์เหตุการณ์ Click ของคอนโทรล CommandButton

ความหมายในแต่ละคำของโพรซีเจอร์ สำหรับพื้นที่ในการเขียนโค้ดจะอยู่ระหว่าง 2 บรรทัดดังกล่าว (ที่เทอร์เซอร์กระพริบอยู่)

- **Private** เป็นคำสงวนที่กำหนดขอบเขตของโพรซีเจอร์
- **Sub** เป็นคำสงวนที่บอกชนิดของโพรซีเจอร์ กรณีนี้เป็นแบบซับรูทีน (ในการใช้งานจริงจะมีรูปแบบของโพรซีเจอร์หลายชนิด เช่น โพรซีเจอร์ทั่วไป ฟังก์ชันที่สร้างขึ้นมาเอง)
- **Command1** หมายถึงชื่อของอ็อบเจกต์ หรือคอนโทรล ถ้ามีการเปลี่ยนชื่อ (คุณสมบัติ Name) ส่วนนี้จะเปลี่ยนไปเป็นชื่อเดียวกับชื่อที่ตั้งไว้
- **_** เครื่องหมายอันเดอร์สกออร์ ใช้แบ่งชื่อของคอนโทรลและเหตุการณ์ออกจากกัน
- **Click** เหตุการณ์ประจำโพรซีเจอร์ หมายความว่าโพรซีเจอร์นี้จะทำงานเมื่อมีการคลิกปุ่มที่ชื่อว่าCommand1 เท่านั้น
- **()** ในวงเล็บ อาจจะมีรายการตัวแปรที่คุณจำเป็นต้องใช้ใน โพรซีเจอร์นี้ ซึ่งเรียกว่า อาร์กิวเมนต์(arguments) ซึ่งในการใช้งานจริง
- **End Sub** จบโพรซีเจอร์

4.7 โพรซีเจอร์ประจำคอนโทรล

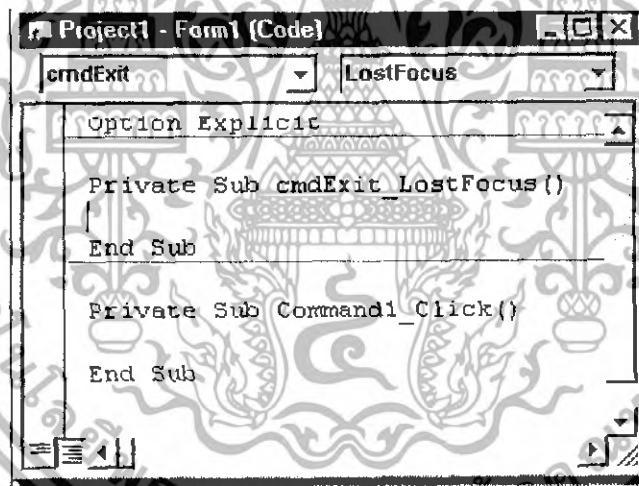
เมื่อนำคอนโทรลชนิดต่างๆ มาใช้งานแล้ว VB จะสร้างโพรซีเจอร์ต่างๆ พร้อมเหตุการณ์ประจำคอนโทรลนั้นๆ ขึ้นมาเสมอ โดยที่ VB จะใช้หน้าที่ของแต่ละคอนโทรลเป็นหลัก แล้วสร้างโพรซีเจอร์เหตุการณ์ที่เกี่ยวข้องกับคอนโทรลนั้นขึ้นมาโดยอัตโนมัติเช่นคอนโทรล

CommandButton มีหน้าที่เป็นปุ่มให้ผู้ใช้คลิกตอบรับ (เหตุการณ์ Click) VB ก็จะสร้างโพรซีเจอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Private Sub Command1_Click(), คอนโทรล TextBox มีหน้าที่รับหรือแสดงข้อมูล ที่ผู้ใช้สามารถเปลี่ยนแปลงได้ VB จะสร้างโพรซีเจอร์ Private Sub Text1_Change(), ฟอรัม เป็นตัวบรรจุคอนโทรลอื่นๆ และจะถูกโหลดเข้ามาในหน่วยความจำก่อนเสมอ VB จะสร้างโพรซีเจอร์ Private Sub Form_Load() เป็นต้น โพรซีเจอร์ต่างๆ ที่ VB สร้างขึ้นมา ไม่จำเป็นต้องใช้งานทั้งหมด หรือถ้าจะใช้ทั้งหมดก็ได้ ก็สามารถปล่อยให้โพรซีเจอร์ว่างๆได้ โดยไม่ปัญหาในการประมวลผลแต่อย่างใด แม้ว่า VB จะสร้างโพรซีเจอร์ประจำคอนโทรลให้โดยอัตโนมัติอยู่แล้ว แต่ในความเป็นจริง โพรซีเจอร์เหล่านี้ ยังไม่สามารถรองรับการทำงาน ของแอปพลิเคชันหนึ่งๆ ได้ทั้งหมด จะต้องเพิ่มเหตุการณ์ต่างๆขึ้นมา เพื่อรองรับการใช้งานของผู้ใช้ ดังนั้นสิ่งที่ต้องศึกษาต่อไปก็คือ จะมีเหตุการณ์อะไรบ้างที่อาจเกิดขึ้นกับคอนโทรล คอนโทรลแต่ละชนิดสนับสนุนเหตุการณ์อะไรบ้าง เหตุการณ์ที่เกิดขึ้นในแต่ละคอนโทรลมีความสัมพันธ์กันอย่างไร สำหรับวิธีการเพิ่มโพรซีเจอร์เหตุการณ์ของแต่ละคอนโทรล ดังนี้

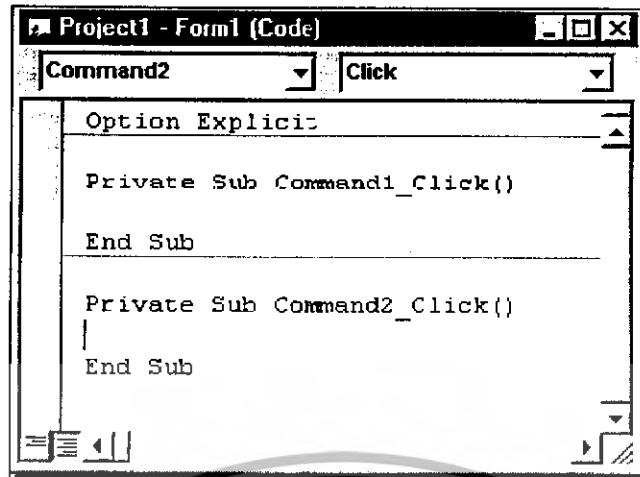
1. ให้เลือกชนิดของคอนโทรล หรืออ็อบเจกต์ใน object list box
2. เลือกเหตุการณ์ใน event list box
3. VB จะสร้างโพรซีเจอร์ใหม่ที่ตรงกับเหตุการณ์ที่เลือกไว้ กับคอนโทรลนั้นๆ



รูปที่ 4.15 แสดงโพรซีเจอร์เหตุการณ์ LostFocus ของคอนโทรล CommandButton ที่ถูกเพิ่มเข้ามา

และถ้าในกรณีที่เลือกคอนโทรลเหมือนกัน เหตุการณ์เหมือนกัน ในฟอร์มเดียวกัน VB จะสร้างโพรซีเจอร์เหตุการณ์ ให้กับคอนโทรลทั้ง 2 ตัว เพราะ VB ถือว่าเป็นคนละอ็อบเจกต์กัน เช่น ถ้าใช้งานคอนโทรล CommandButton 2 ตัว ซึ่งมีเหตุการณ์คลิกเหมือนกัน จึงมีเหตุการณ์คลิก 2 โพรซีเจอร์ สำหรับคอนโทรล Command1 และ Command2 ผู้ใช้คลิกปุ่ม Command1 อาจจะออกจากโปรแกรม ผู้ใช้คลิก Command2 อาจจะแสดงข้อความ Hello World ! ก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.16 แสดงโปรแกรมเหตุการณ์คลิกของคอนโทรล CommandButton 2 ตัว

สำหรับในส่วน object list box จะมีรายการพิเศษที่เขียนว่า (General) และใน event listbox จะเขียนว่า (declarations) จะเป็นพื้นที่พิเศษ ที่ใช้สำหรับประกาศ (declare) ตัวแปรต่างๆ, กลุ่มฟังก์ชัน Windows API เป็นต้น

4.8 โค้ดชุดแรก

หลังจากที่ VB ได้สร้างโปรแกรมต่างๆ ขึ้นมาประจำคอนโทรลแล้ว จะต้องเริ่มเขียนโค้ดเพื่อที่จะทำให้แต่ละโปรแกรมทำงานได้ซึ่งในแต่ละโปรแกรมจะรองรับเหตุการณ์เดียวเท่านั้น จะต้องเขียนโค้ดให้แต่ละโปรแกรมทำงานครบ และเสร็จสิ้นภายในโปรแกรมมันเอง เช่น เมื่อผู้ใช้คลิกปุ่ม OK แล้วจะเกิดอะไรขึ้นบ้าง, เมื่อฟอร์มถูกโหลดเข้ามาจะมีอะไรบ้าง, เมื่อผู้ใช้ดับเบิลคลิกที่ฟอร์ม จะเกิดอะไรขึ้นต้องเขียนโค้ดรองรับเหตุการณ์เหล่านี้ทั้งหมด เป็นต้น

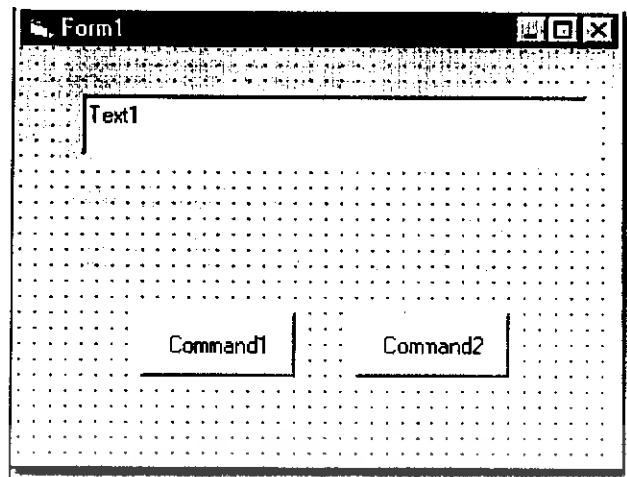
องค์ประกอบหลักของการเขียนโค้ดตอบรับคือจะต้องเชื่อมโยงโปรแกรมต่างๆ ให้สามารถทำงานสอดคล้องกันไปตามการทำงานของแอปพลิเคชันนั้นๆ ให้ได้ ตัวอย่างโปรแกรมที่ต้องใช้งานในทุกๆ แอปพลิเคชัน ก็คือ เมื่อผู้ใช้คลิกปุ่ม Command1 แล้วให้จบการทำงานของโปรแกรม

```
Private Sub Command1_Click()
```

```
End
```

```
End Sub
```

คำสั่ง End เป็นการบอกให้รู้ว่า เมื่อผู้ใช้คลิกปุ่ม Command1 แล้วให้จบการทำงาน จะยกตัวอย่างที่แสดงให้เห็น การเชื่อมโยง 2 โปรแกรมเข้าด้วยกัน ให้วาดอินเตอร์เฟซดังรูปและเขียนโค้ดต่อไปนี้



รูปที่ 4.17 แสดงอินเตอร์เฟซเพื่อสั่งให้คอนโทรล TextBox แสดงข้อความ hello world

```

Private Sub Command1_Click ()
    Text1.Text = "hello world"
End Sub
Private Sub Command2_Click ()
    Text2.Text = " "
End Sub

```

ให้ทดลองรันโปรแกรมโดยการคลิกปุ่ม บน Toolbar หรือคีย์ F5 แล้วลองคลิกที่ปุ่ม Command1 และ Command2 ดูผลที่เกิดขึ้น เมื่อคลิกที่ปุ่ม Command1 จะมีข้อความ helloworld ปรากฏในคอนโทรล TextBox และเมื่อคลิกที่ปุ่ม Command2 ข้อความดังกล่าวจะหายไปทำความเข้าใจกับโค้ดชุดแรก

ความหมายของโค้ดที่เขียนมีรายละเอียดดังนี้

- **Private Sub Command1_Click ()** หมายถึง โปรซีเจอร์นี้ทำงานเมื่อผู้ใช้คลิกที่ปุ่ม Command1 เท่านั้น
- **Text1.Text = "hello world"** หมายถึง สั่งให้คอนโทรล TextBox ที่ชื่อ Text1 แสดงข้อความ hello world ผ่านทางคุณสมบัติ Text เพราะว่าคุณสมบัติ Text นี้มีหน้าที่แสดงข้อความ ส่วนจุด . ใช้แยกชื่อคอนโทรลกับคุณสมบัติ (properties) หรือแยกชื่อคอนโทรลกับเมธอด (methods) กรณีนี้ Text เป็นคุณสมบัติของคอนโทรล TextBox
- **End Sub** หมายถึง จบโปรซีเจอร์ Command1_Click ()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Private Sub Command2_Click ()** หมายถึง โพรซีเดอร์นี้ทำงานเมื่อผู้ใช้คลิกที่ปุ่ม Command2 เท่านั้น
- **Text1.Text = " "** หมายถึง ตั้งให้คอนโทรล TextBox ที่ชื่อ Text1 ไม่ต้องแสดงข้อความ เครื่องหมาย " " แทนข้อความว่างเปล่า
- **End Sub** จบโพรซีเดอร์ Command2_Click ()

4.9 หมายเหตุในการเขียนโค้ด (Comment)

ในการเขียนโปรแกรม ไม่ว่าจะเป็ภาษาใดก็ตาม ควรที่จะต้องเขียนหมายเหตุไว้ เพื่อบอกความหมายของโค้ดบรรทัดต่อไป อาจเขียนไว้เพื่อบอกว่า ตัวแปรนี้ใช้แทนอะไร ข้อความบอกจุดประสงค์ของโค้ดบรรทัดนั้นๆ เพื่อเป็นประโยชน์ ในกรณีที่ต้องย้อนกลับมา ศึกษาโค้ดนี้อีกครั้ง รูปแบบการเขียนหมายเหตุก็คือ ควรจะเป็นข้อความสั้นๆ กระชับ แต่ได้ใจความ ที่บอกความหมายของโค้ดบรรทัดนั้น หรือบอกว่ตัวแปรนั้นแทนอะไร ไม่ควรใช้คำอธิบาย เพราะตัวหมายเหตุเองจะทำให้งงมากกว่าได้เสียอีก ในกรณีที่ต้องเขียนโค้ดมากๆ สำหรับการเขียนโค้ดที่มีการใช้งานตัวแปร

วิธีการเขียนหมายเหตุใน VB คือ จะใช้เครื่องหมาย ' (apostrophe) หน้าบรรทัดที่ต้องการทำหมายเหตุ แล้ว editor จะเปลี่ยนบรรทัดนั้น เป็นสีเขียว และจะไม่สนใจบรรทัดดังกล่าว ในการประมวลผลทันที อาจใช้หมายเหตุ เพื่อทดสอบที่ละโพรซีเดอร์ก็ได้ เช่น กรณีที่โพรซีเดอร์เกิด error บ่อยมาก :- (ก็หมายเหตุโพรซีเดอร์ ที่ error ไว้ก่อนก็ได้ อาจใช้คำสงวน Rem มาจากคำว่า Remark แทนเครื่องหมาย ' ก็ได้ โปรแกรมเมอร์ส่วนใหญ่ จะใช้หมายเหตุ เพื่อบอกความหมายของโค้ดเป็นช่วงๆ เสมอ)

4.10 คุณสมบัติและเมธอดของคอนโทรล

สำหรับคอนโทรลและอ็อบเจ็กต์ทุกชนิดใน VB จะมีคุณสมบัติและเมธอดประจำตัวของมันเองความหมายของทั้ง 2 อย่างมีดังนี้

- **คุณสมบัติ (properties)** หมายถึง รูปร่าง ลักษณะ ความกว้าง ความยาว สี ฯลฯ ของตัวคอนโทรลหรืออ็อบเจ็กต์ ขอให้มองคอนโทรลหรืออ็อบเจ็กต์ เป็นเสมือนวัตถุชิ้นหนึ่ง ที่มีความยาว ความกว้าง สีของตัวมันเอง คุณสามารถแก้ไขได้เหมือนดังเช่น คุณกำลังปรับแต่งวัตถุชิ้นหนึ่งอยู่ สำหรับในแต่ละคอนโทรล หรืออ็อบเจ็กต์ อาจจะมีคุณสมบัติที่เหมือนกัน หรือต่างกันก็ได้ ขึ้นอยู่กับหน้าที่ของแต่ละคอนโทรล คอนโทรลหรืออ็อบเจ็กต์หนึ่งๆ จะมีคุณสมบัติมากมาย หลายอย่าง ยิ่งคุณสามารถปรับแต่งคุณสมบัติ ให้ตรงกับความต้องการมากเพียงใด แอปพลิเคชันของก็จะมีประสิทธิภาพมากขึ้นเท่านั้น ซึ่งถือได้ว่าเป็นจุดเริ่มต้นได้ดี ในการพัฒนาแอปพลิเคชัน สามารถ

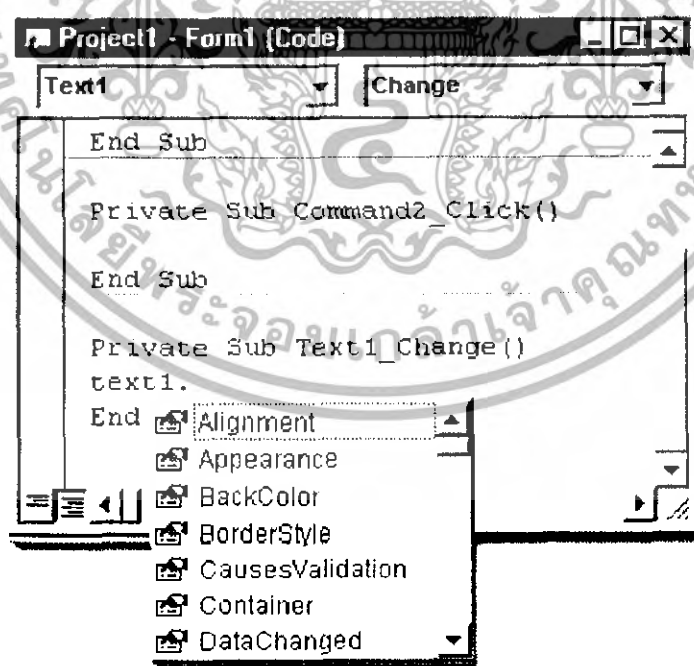
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปรับแต่ง คุณสมบัติได้จากหน้าต่าง Properties หรือปรับแต่งด้วยการเขียนโค้ดก็ได้ จะมีคุณสมบัติ บางตัว ที่ไมโครซอฟท์แนะนำให้ ปรับแต่งด้วยการเขียนโค้ด และบางตัวปรับแต่งด้วยการแก้ไขใน หน้าต่าง Properties ซึ่งจะอธิบายอย่างละเอียดอีกครั้ง ในการใช้งานแต่ละคอนโทรลและในทาง ปฏิบัติ ไม่จำเป็นต้องปรับแต่งทุกๆ คุณสมบัติ เพราะ VB ได้ตั้งค่า default ไว้ให้แล้ว ซึ่งก็สามารถ ใช้งานได้ในระดับหนึ่ง

- **เมธอด (methods)** หมายถึง พฤติกรรมของคอนโทรลหรืออ็อบเจกต์ อาจกล่าวได้ว่า เป็น การควบคุมการทำงานของคอนโทรลหรืออ็อบเจกต์นั่นเอง จะใช้จุดเป็นตัวคั่นระหว่างชื่อ คอนโทรลกับเมธอด ซึ่งจะเห็นได้ว่า คุณสมบัติและเมธอดมีความใกล้เคียงกันมาก เนื่องจากจะใช้ จุด . เป็นตัวแยกระหว่าง ชื่อคอนโทรลกับคุณสมบัติ หรือชื่อคอนโทรลกับเมธอด จะมีความ แตกต่างกันในแง่ของการควบคุมคอนโทรล หรืออ็อบเจกต์ ซึ่งคุณจะได้ศึกษาในหัวข้อต่อไป

4.11 ความสามารถพิเศษของ editor

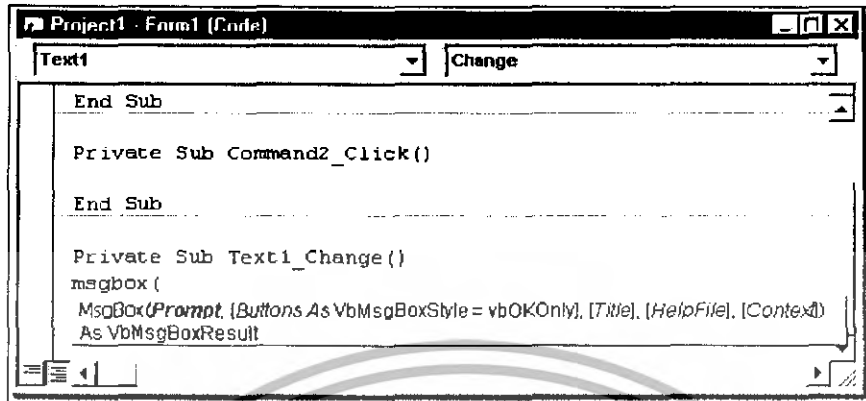
จากตัวอย่างข้างต้นจะพบว่า ขณะที่พิมพ์โค้ดให้กับแต่ละโพรซีเจอร์ เมื่อพิมพ์ . ตัว editor จะแสดง Tooltip ที่เป็นรายการคุณสมบัติ หรือรายการเมธอด ที่คอนโทรลนั้นสนับสนุนอยู่ขึ้นมาทันที มันช่วยให้คุณไม่ต้องไปจดจำว่า คอนโทรลนี้มีคุณสมบัติอะไรบ้าง มีเมธอดอะไร รวมถึง ป้องกันไม่ให้ พิมพ์ผิดอีกด้วย ดังรูปที่ 4.18



รูปที่ 4.18 แสดง Tooltip ของ editor ขณะเขียนโค้ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และถ้าในการเขียนโค้ดที่ต้องการมีการเรียกใช้งานฟังก์ชันมาตรฐานต่างๆ Tooltip ก็ จะแสดงรูปแบบไวยากรณ์อีกด้วย



```

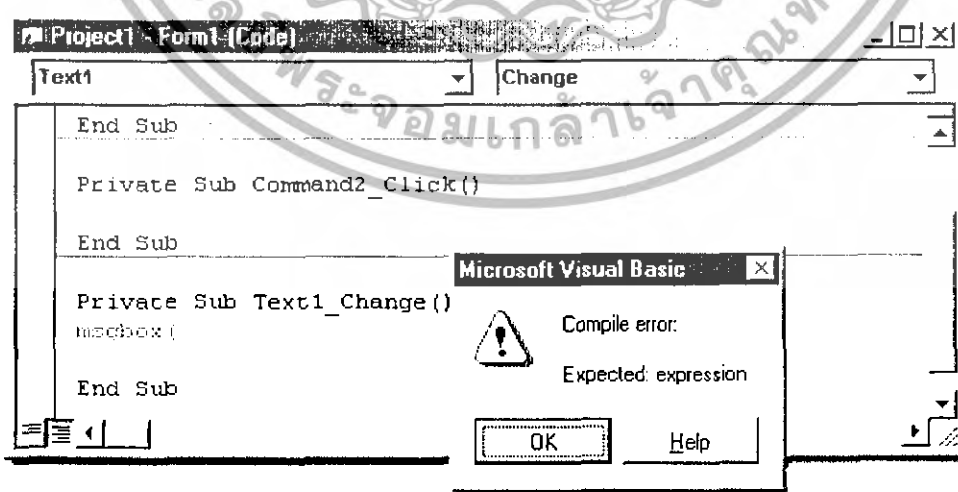
Project1 - Form1 (Code)
Text1 Change
End Sub
Private Sub Command2_Click()
End Sub
Private Sub Text1_Change()
msgbox (
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context])
As VbMsgBoxResult

```

รูปที่ 4.19 แสดงไวยากรณ์ประจำฟังก์ชันนั้นๆ

ความสามารถทั้ง 2 อย่างช่วยให้โปรแกรมเมอร์ย่นระยะเวลาในการเขียนโค้ดเป็นอย่างมาก เพราะจะทำให้ข้อผิดพลาดที่เกิดจากการใช้งานผิดไวยากรณ์ หรือพิมพ์ผิด หมดไปได้อย่างสิ้นเชิง และที่สำคัญคุณสามารถพัฒนาแอปพลิเคชันได้เต็มที่โดยไม่ต้องเสียเวลากับเรื่องที่ต้องท่องจำคุณสมบัติต่างๆ นานา รูปแบบการใช้งาน ที่มากมายเหลือเกิน คำหังแต่ความรู้ เทคโนโลยีใหม่ที่เกิดขึ้นมา ก็แทบเรียนรู้กันไม่หมดแล้ว

สิ่งที่น่าทึ่งอีกอย่างหนึ่งของ Editor ก็คือ มันสามารถตรวจสอบไวยากรณ์ (syntax) ตามโครงสร้างของภาษา VB ได้อีกด้วย ซึ่งเรียกว่า Auto Syntax Check ขณะที่พิมพ์โค้ดเข้าไป เมื่อกด Enter จบบรรทัด VB จะทำการตรวจสอบไวยากรณ์ทันที ถ้ามีข้อผิดพลาด ในการใช้งานไวยากรณ์เกิดขึ้น VB จะแสดงข้อความช่วยเหลือที่เกี่ยวข้องกับความผิดพลาดนั้นๆ ขึ้นมาทันที โดยการคลิกปุ่ม Help ทำให้แก้ไขความผิดพลาดนั้น ได้อย่างถูกต้อง



รูปที่ 4.20 แสดงข้อความช่วยเหลือ เมื่อคุณพิมพ์ผิดไวยากรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.20 เมื่อคลิกปุ่ม help VB จะแสดงข้อความที่เกี่ยวข้องกับความผิดพลาดที่เกิดขึ้นทันที สำหรับรูปแบบการแสดงโค้ด ที่ถูกพิมพ์ลงไปในแต่ละโปรซีเจอร์ VB จะแยกส่วนของโค้ดออกเป็น 3 กลุ่ม ด้วยสีของฟอนต์ คือ

- ตัวแปร,ชื่อคอนโทรล,เหตุการณ์จะมีสีดำ
- คำสั่ง,ฟังก์ชัน จะมีสีน้ำเงิน
- หมายเหตุ จะมีสีเขียว

เมื่อดู source code สามารถใช้กลุ่มสีของฟอนต์ ช่วยในการทำความเข้าใจโค้ดด้วยก็ได้ สิ่งหนึ่งที่จะทำให้เกิดข้อผิดพลาดได้ก็คือ ตัวแปรที่ประกาศใช้งานเป็นรูปแบบหนึ่ง เวลาพิมพ์อีกแบบหนึ่ง VB จะเปลี่ยนให้โดยอัตโนมัติ เช่น เมื่อประกาศตัวแปร iName AsString แต่เวลาที่นำไปใช้ พิมพ์ผิดเป็น iName VBจะทำการเปลี่ยนตัวแปรที่พิมพ์ไม่เหมือนกับต้นแบบให้เหมือนกันโดยอัตโนมัติและในกรณีที่มีการใช้คำสั่ง (statement) (หรือคำสั่งวงประเภทที่ VB ให้ใช้ ซึ่ง คำสั่งวงนั้นจะมีผลในการทำงานของโค้ด ใน โปรซีเจอร์ด้วย) VB จะเปลี่ยนอักษรตัวแรกให้เป็นตัวใหญ่เสมอเช่น เมื่อพิมพ์ end เข้าไป VB จะเปลี่ยนเป็น End หรือเมื่อพิมพ์ text1.text = "hello world" เข้าไป VB จะเปลี่ยนเป็น Text1.Text = "helloworld" ให้

จะเห็นได้ว่าไมโครซอฟท์พยายามทำให้ Editor มีบทบาทช่วยโปรแกรมเมอร์ ในการพัฒนาแอปพลิเคชันมากที่สุด ยังมีความสามารถอื่นๆ อีกของ Editor ที่ไม่ได้กล่าวไว้ เทคนิคที่ใช้ใน Word 97 บางอย่าง สามารถนำมาใช้งานกับ editor ได้เช่นกัน

4.12 คุณสมบัติประจำตัวของคอนโทรล

จากหัวข้อที่ผ่านมาทำให้ทราบว่าคอนโทรลแต่ละชนิดล้วนแล้วแต่มีทั้งคุณสมบัติและเมธอดทั้งสิ้นยากแก่การจดจำ(และไม่ต้องจำ)แต่ในทุกๆคอนโทรลจะมีคุณสมบัติอยู่ตัวหนึ่งที่ใช้สำหรับเก็บค่า แสดงค่า หรืออ่านค่าของตัวเอง เช่น คอนโทรล TextBox มีหน้าที่สำหรับ รับหรือแสดงข้อมูล ที่ผู้ใช้สามารถแก้ไขได้ ทั้ง 2 กรณีตัวคอนโทรล TextBox จะเก็บข้อมูลไว้ที่คุณสมบัติ Text จึงถือได้ว่า คุณสมบัติ Text เป็นคุณสมบัติประจำตัวของคอนโทรล TextBox, คอนโทรล CommandButton มีหน้าที่เป็นปุ่มกดรับคำสั่งของผู้ใช้ เช่น ปุ่ม OK , Cancel เป็นต้น คุณสมบัติที่จะแสดงชื่อปุ่มคือ คุณสมบัติ Caption, คอนโทรล Label มีหน้าที่แสดงข้อมูลให้ผู้ใช้ทราบ โดยที่ผู้ใช้ไม่สามารถแก้ไขได้ ก็จะมีคุณสมบัติ Caption เป็นตัวเก็บค่า เป็นต้น ซึ่งคุณสมบัติที่กล่าวมาจะต้องเข้าไปใช้งานอย่างแน่นอน ตารางต่อไปนี้ เป็นคุณสมบัติประจำตัวของแต่ละคอนโทรล

คอนโทรล	คุณสมบัติประจำตัว
CheckBox	Value
ComboBox	Text
CommandButton	Caption
CommonDialog	Action
Data	Caption
DBCombo(Data-Bound ComboBox)	Text
DBGrid(Data-Bound Grid)	Text
DBList(Data-Bound ListBox)	Text
DirListBox	Path
DriveListBox	Drive
FileListBox	FileName
Frame	Caption
HScrollBar	Value
Image	Picture
Label	Caption
Line	Visible
ListBox	Text
OptionButton	Value
PictureBox	Picture
Shape	Shape
TextBox	Text
Timer	Enabled
VScrollBar	Value

ตารางที่ 4.1 แสดงคุณสมบัติประจำตัวแต่ละคอนโทรล

ในการเขียนโค้ดกับคุณสมบัติประจำตัวของคอนโทรล ไม่จำเป็นต้องระบุคุณสมบัติก็ได้ โดยที่ไม่ผิดไวยากรณ์แต่อย่างใด สำหรับในการใช้งาน จริงๆ แล้ว โปรแกรมเมอร์ทุกๆ ไปจะไม่นิยมใช้เพราะว่า มันจะทำให้คุณสับสนเป็นอย่างยิ่ง เนื่องจากเฉพาะโค้ดของแอปพลิเคชันหนึ่ง จะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประกอบไปด้วย จำนวนโพรซีเจอร์มากมาย ซึ่งในแต่ละโพรซีเจอร์ยังมีจำนวนบรรทัดของโค้ดอีกหลายสิบ หลายร้อยบรรทัด ทำให้ไม่นิยมใช้ เช่น

รูปแบบที่ 1 Text1.Text ="hello world"

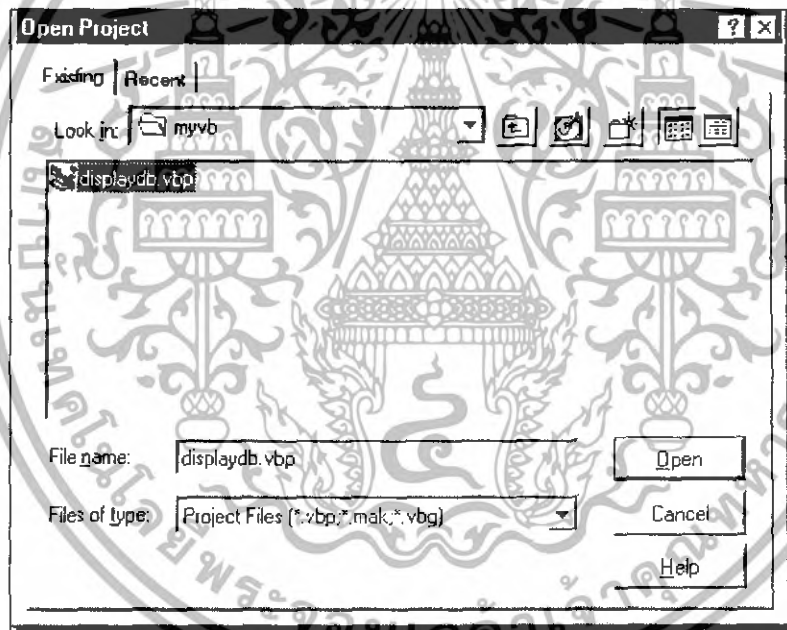
รูปแบบที่ 2 Text1 = "hello world"

รูปแบบที่ 1 เป็นการเขียนโค้ดแบบปกติ ส่วนรูปแบบที่ 2 เป็นการเขียนโค้ดกับคุณสมบัติประจำตัวคอนโทรลดังนั้นจึงไม่ต้องระบุคุณสมบัติ Text ก็ได้ ซึ่งไม่ผิดไวยากรณ์แต่อย่างใด ที่กล่าวถึง เนื่องจากว่า เป็นการแสดงความยืดหยุ่นที่ไม่โครซอฟต์แวร์เตรียมไว้ให้เท่านั้น

4.13 การจัดการโปรเจกต์

- การเปิดโปรเจกต์ที่มีอยู่แล้ว

1. ให้เลือกเมนู File/Open Project หรือคลิกที่ปุ่ม ที่ทูลบาร์ แล้ว VB จะแสดงไดอะล็อกบ็อกซ์ Open Project ดังรูปที่ 4.21



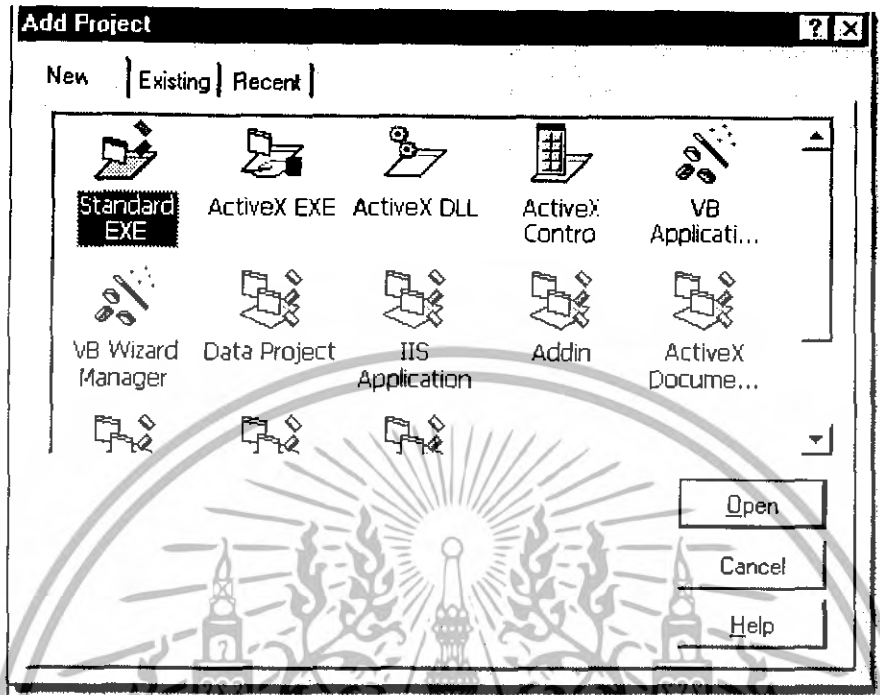
รูปที่ 4.21 แสดง ไดอะล็อกบ็อกซ์ Open Project

2. แท็บ Existing หมายถึง เมื่อต้องการเปิดโปรเจกต์ที่มีอยู่ โดยที่ไม่สนใจว่าจะเป็นโปรเจกต์ที่สร้างมาเมื่อใด ส่วนแท็บ Recent หมายถึง ต้องการเปิดโปรเจกต์ที่คุณพัฒนาครั้งล่าสุด ซึ่งจะแสดงไครฟ์ และโฟลเดอร์ที่เก็บโปรเจกต์นั้นๆอยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การเพิ่มโปรเจกต์ เข้าไปในสภาพแวดล้อม

1. ให้คุณเมนู Project/Add Project แล้ว VB จะแสดงไดอะล็อกบ็อกซ์ Add Project ดังรูปที่ 4.22



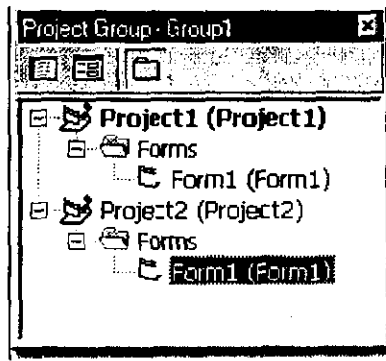
รูปที่ 4.22 แสดงไดอะล็อกบ็อกซ์ Add Project

2. สามารถเลือกชนิดของโปรเจกต์ที่ต้องการ ได้จาก ไดอะล็อกบ็อกซ์ดังกล่าว แท็บ Existing หมายถึง ต้องการเพิ่มโปรเจกต์เดิมที่มีอยู่แล้ว โดยที่ไม่สนใจว่า จะเป็นโปรเจกต์ที่สร้างมาตอนไหน ส่วนแท็บ Recent หมายถึง ต้องการเพิ่มโปรเจกต์ที่พัฒนาครั้งล่าสุด ซึ่งจะแสดงไครฟ และ โพลเดอร์ที่จัดเก็บโปรเจกต์นั้นๆ อยู่

- การถอดโปรเจกต์ออกจากสภาพแวดล้อม

จะใช้สำหรับกรณีที่มีการเพิ่ม โปรเจกต์เข้ามา แล้วต้องการถอดโปรเจกต์นั้นๆ ออกจากสภาพแวดล้อม ต้องทำดังนี้

1. ให้เลือกโปรเจกต์ที่ต้องการถอด จากหน้าต่าง Project Explorer ดังรูปที่ 4.23

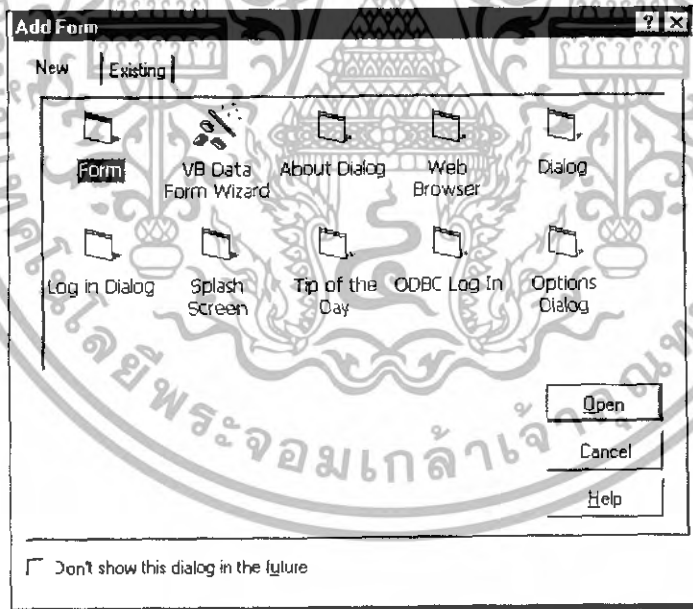


รูปที่ 4.23 แสดงการเลือกโปรเจกต์ที่คุณต้องการถอดออกจากสภาพแวดล้อม

2. เลือกคำสั่ง File/Remove Project

- การเพิ่ม Form หรือโมดูลอื่นๆ เข้ามาในสภาพแวดล้อม












1. ให้เลือกเมนู Project จากนั้นให้เลือกชนิดของโมดูลที่ต้องการเพิ่มเข้ามาในสภาพแวดล้อม และถ้าในกรณีที่ต้องการเพิ่ม Form เข้ามาในโปรเจกต์ สามารถคลิกที่ปุ่ม  บนทูลบาร์เลยก็ได้



รูปที่ 4.24 แสดงการเพิ่ม Form เข้ามาในโปรเจกต์

แท็บ Existing หมายถึง ต้องการเพิ่มฟอร์มเดิมที่มีอยู่แล้ว ในกรณีนี้เป็นการเพิ่มฟอร์มว่างๆ เข้ามาในสภสแวดล้อม สำหรับไอคอนต่างๆ มีความหมายดังนี้

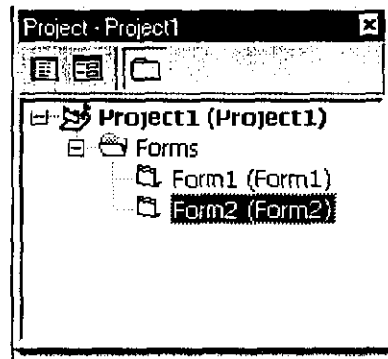
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	เป็นฟอร์มกว้างๆ แบบปกติ
	
	เป็นการใช้ความสามารถของ VB Data Form Wizard เพื่อสร้างแบบฟอร์ม
VB Data Form Wizard	รายงาน ที่ติดต่อกับฐานข้อมูลโดยผ่านคอนโทรล Data
	เป็นแบบฟอร์ม About สำเร็จรูป คุณอาจจะต้องปรับแต่งเพื่อให้ได้ฟอร์ม
About Dialog	About ตามที่คุณต้องการ
	เป็นฟอร์มที่มีลักษณะเหมือนกับ browser คล้ายๆ กับ Internet Explorer
Web Browser	หรือ Netscape Communicator
	ไดอะล็อกบ็อกซ์สำเร็จรูป ถ้าคุณเบื่อๆ ไดอะล็อกบ็อกซ์แบบเก่าๆ นี่เป็นได
Dialog	อะล็อกบ็อกซ์ที่คุณออกแบบเอง
	ถ้าคุณสร้างแอปพลิเคชันที่ต้องการตรวจสอบ หรือจำกัดสิทธิการใช้งานผู้
Log in Dialog	ใช้ในแต่ละระดับ แบบฟอร์ม Log in สำเร็จรูปช่วยคุณได้ ซึ่งจะต้องมีการใส่
	ชื่อผู้ใช้ และ password ด้วย
Splash Screen	เป็นจอภาพเริ่มต้นก่อนเข้าแอปพลิเคชัน ทุกๆ แอปพลิเคชันต้องมี
	คุณคงเคยเห็นไดอะล็อกบ็อกซ์ที่แสดง Tip ตอนเข้าสู่แอปพลิเคชัน นี่เป็น
Tip of the Day	ไดอะล็อกบ็อกซ์ที่แสดง Tip สำเร็จรูป ซึ่งคุณต้องเพิ่มเทคนิคต่างๆ เข้าไป
	เป็นแบบฟอร์มที่ใช้สำหรับติดต่อกับฐานข้อมูลโดยผ่านไดร์เวอร์ ODBC
ODBC Log In	
	เป็นแบบฟอร์มที่มีลักษณะเป็นตัวเลือกเพื่อกำหนดหัวข้อข้อมูลต่างๆ ตาม
Options Dialog	ความต้องการของผู้ใช้ ซึ่งคุณจะพบเห็นได้ในแอปพลิเคชันทั่วๆ ไป

ตารางที่ 4.2 ความหมายของไอคอนในการสร้างฟอร์มชนิดต่างๆ

2. ให้สังเกตที่หน้าต่าง Project Explorer จะมีรายการแสดงเพิ่มขึ้นมา ซึ่งอยู่ภายใต้ Project เดียวกัน จากรูปจะเห็นว่า Project1 ประกอบด้วย Form 2 ฟอร์ม

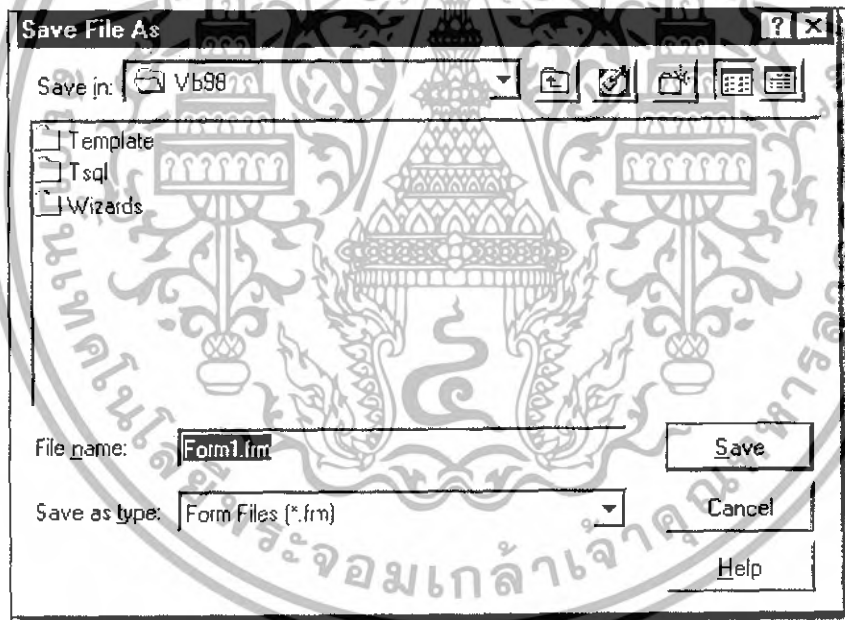
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.25 แสดง Form ที่ถูกเพิ่มเข้ามาในโปรเจกต์

- การเซฟโปรเจกต์

1. ให้เลือกเมนู File/Save Project Group As... หรือคลิกที่ปุ่ม บน Tool Bar แล้ว VB จะแสดงไดอะล็อกบ็อกซ์ เซฟโปรเจกต์ ดังรูปที่ 4.26 ซึ่ง VB จะแสดงรายการส่วนประกอบต่างๆ ที่อยู่ในโปรเจกต์นั้นๆ ให้ทราบด้วย








รูปที่ 2.26 แสดงไดอะล็อกบ็อกซ์เซฟโปรเจกต์

2.คลิกปุ่ม

ในกรณีที่คุณเลือกคำสั่ง Save Form As... หมายถึง เป็นการเซฟเฉพาะฟอร์มที่ใช้งานอยู่เท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

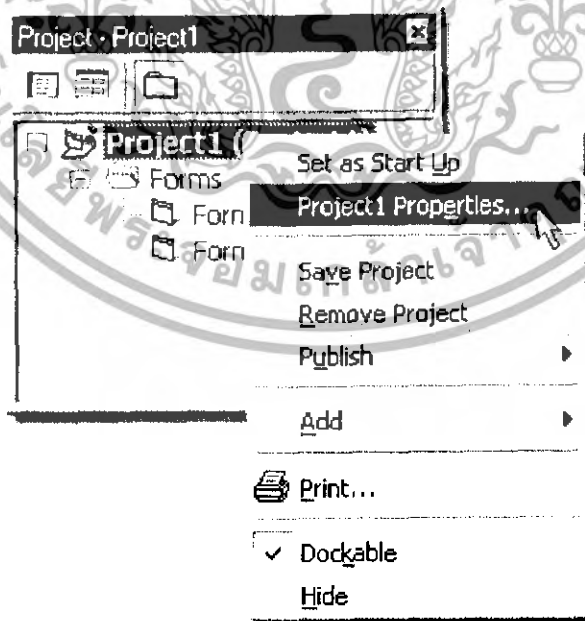
4.14 การทดสอบ (Run) โปรเจกต์

- 1.เลือกเมนู Run/Start หรือกดปุ่ม FS ที่คีย์บอร์ด หรือกดปุ่ม  ที่ทูลบาร์
- 2.ที่ฟอร์มจะมีการเปลี่ยนรูปแบบ ไม่มีเส้นกริด และจะปรากฏหน้าต่าง Immediate ด้านล่างของสภาพแวดล้อม
- 3.ถ้ามีการสั่งให้พิมพ์โดยผ่านทางคำสั่ง Debug.Print ที่หน้าต่าง Immediate จะแสดงผลจากการประมวลผลด้วย
- 4.จะสังเกตเห็นว่าที่บนทูลบาร์ จะมีปุ่ม  และ  มีสถานะ Active อยู่ โดยมีความหมายดังนี้
 - ปุ่ม  หมายถึง ให้หยุดการรันโปรเจกต์
 - ปุ่ม  หมายถึง ให้หยุดรันโปรเจกต์ชั่วคราวเพื่อ แก้ไขโค้ด

4.15 การกำหนดฟอร์มเริ่มต้น

ในการพัฒนาแอปพลิเคชันส่วนใหญ่แล้ว มักจะประกอบไปด้วยฟอร์มหลายฟอร์ม คุณจะต้องมีการกำหนดให้ฟอร์มใดฟอร์มหนึ่ง เป็น ฟอร์มเริ่มต้น(Start Up Form)เสมอ ในกรณีที่โปรเจกต์มีเพียงฟอร์มเดียว Form1 จะถูกกำหนดให้เป็นฟอร์มเริ่มต้นเสมอ แต่ถ้าโปรเจกต์มีมากกว่า 1 ฟอร์ม จะต้องกำหนดให้ฟอร์มใด ฟอร์มหนึ่งเป็นฟอร์มเริ่มต้นเสมอ ซึ่งมีขั้นตอนดังนี้

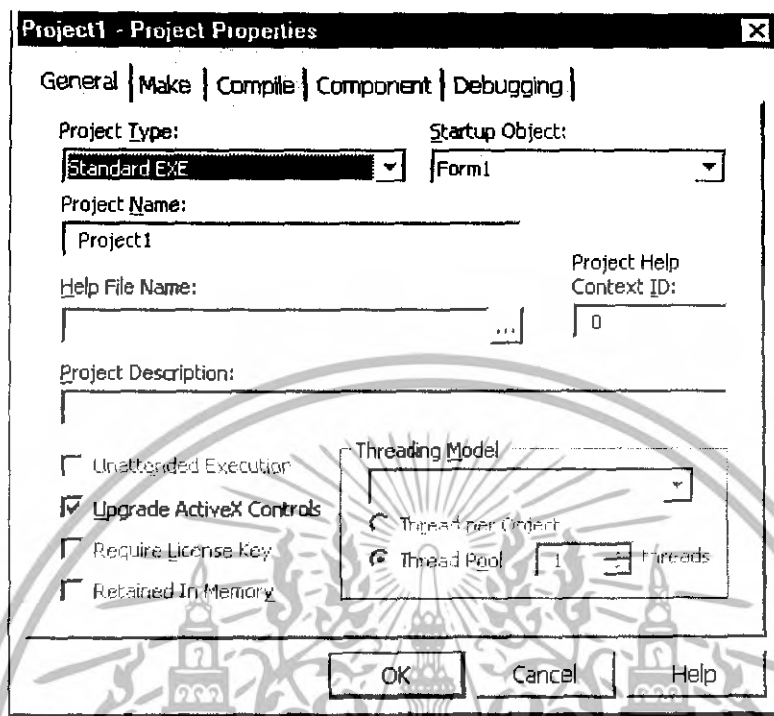
1. เลือกคำสั่ง Project/Project1 Properties... หรือเลื่อนเมาส์ ไปที่ไปที่ไอคอน Project1 (Project1) ในหน้าต่าง Project Explorer แล้วคลิกขวา ตามรูปที่ 4.26



รูปที่ 4.27 แสดง Pop-up เมนู เมื่อคุณคลิกขวา

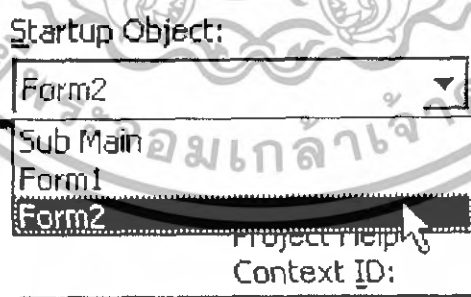
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือกคำสั่ง Project1 Properties... ก็จะได้พบกับ ไดอะล็อกบ็อกซ์ Project Properties ดังรูป



รูปที่ 4.28 แสดง ไดอะล็อกบ็อกซ์ Project Properties

2. ในช่อง Startup Object: ให้เลือกฟอร์มที่ต้องการให้เป็นฟอร์มเริ่มต้น ตามรูป สมมติให้ฟอร์ม 2 เป็นฟอร์มเริ่มต้น จากนั้นคลิกปุ่ม OK เมื่อรันโปรแกรมทุกครั้ง ฟอร์ม 2 ก็จะเป็นฟอร์มเริ่มต้น



รูปที่ 4.29 แสดงการกำหนดให้ฟอร์ม2 เป็นฟอร์มเริ่มต้น

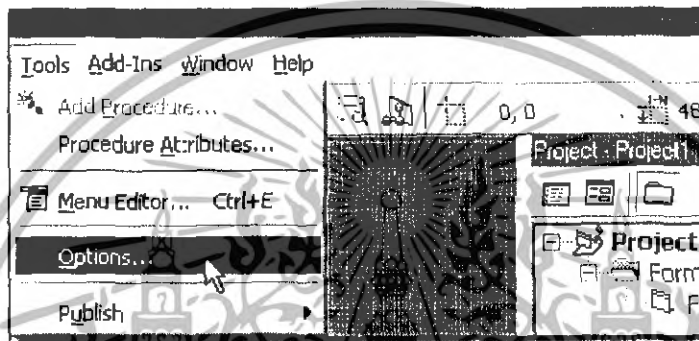
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้拿去ไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับตัวเลือก Sub Main จะใช้ในกรณีที่ต้องการสร้างแอปพลิเคชันในเชิง Client/Server ซึ่งจะต้องมีการ login เข้ามาก่อนที่จะใช้งานแอปพลิเคชัน จะต้องสร้างโปรซีเจอร์ Sub Main () ด้วยเช่นกัน

4.16 วิธีการกำหนดให้ Editor ของ VB สามารถแสดงผลภาษาไทยได้

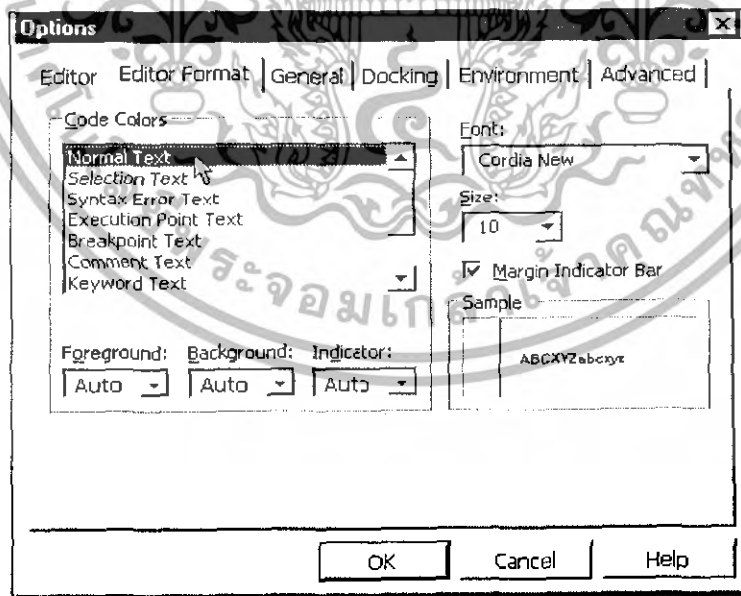
โดยปกติแล้ว Editor ของ VB จะไม่สามารถแสดงผลภาษาไทยได้ ทำให้เราไม่สามารถเขียนหมายเหตุ หรือข้อความภาษาไทยได้ ซึ่ง Editor จะแสดงเป็นตัวอักษรที่อ่านไม่รู้เรื่อง แต่เราสามารถกำหนดให้ Editor สามารถแสดงผลภาษาไทยได้ ซึ่งมีขั้นตอนดังนี้

1.เลือกคำสั่ง Tools/Options... ดังรูป



รูปที่ 4.30 เลือกคำสั่ง Options...

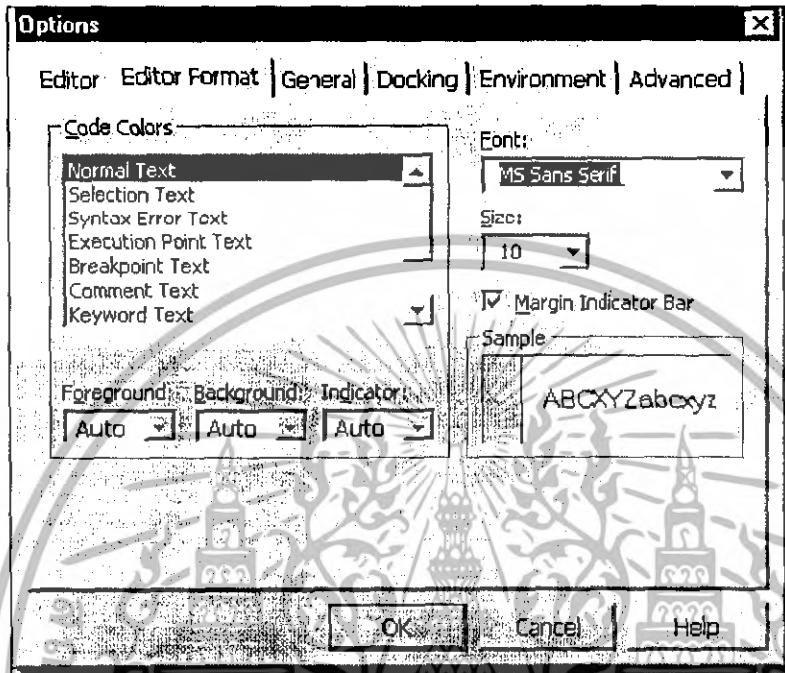
2.จะได้พบกับ ไดอะล็อกบ็อกซ์ Options ให้คลิกเลือกที่แท็บ Editor Format ดังรูป



รูปที่ 4.31 ไดอะล็อกบ็อกซ์ Options

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรายการ Normal Text ให้เปลี่ยนชื่อฟอนต์ในช่อง Font เป็น MS Sans Serif ขนาด 10 หรือฟอนต์ที่ลงท้ายด้วย UPC ขนาด แนะนำให้เลือก MS Sans Serif ขนาด 10 ซึ่งมีลักษณะใกล้เคียงกับแบบปกติ ดังรูปที่ 2-36 จากนั้น คลิกที่ปุ่ม OK



รูปที่ 4.32 กำหนดให้ editor ใช้ฟอนต์ MS Sans Serif ขนาด 10



รูปที่ 4.33 แสดงรูปแบบของฟอนต์ MS Sans Serif ขนาด 10 ใน editor

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

การออกแบบและการทำงานของวงจร

โครงการนี้แบ่งออกเป็น 3 ส่วน ซึ่งประกอบด้วย ส่วนของสวิทช์ , ส่วนที่เป็นตัวควบคุมสวิทช์ และสุดท้ายส่วนที่เป็นการติดต่อระหว่างผู้ใช้งานกับระบบ

5.1 ส่วนของสวิทช์

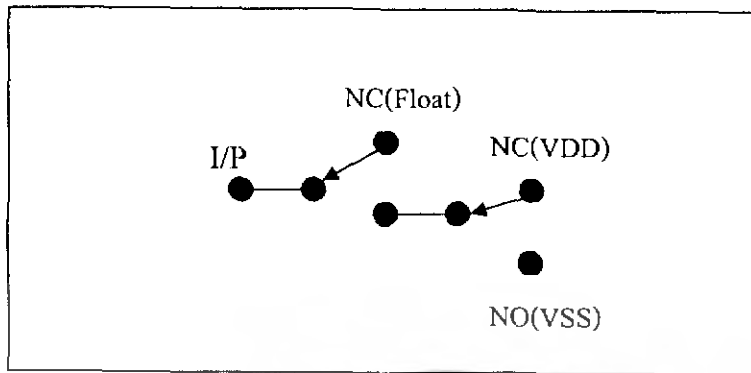
สวิทช์เป็นส่วนที่ใช้สำหรับเลือกสัญญาณที่จะป้อนให้กับตัวอุปกรณ์ที่จะใช้ในการทดสอบ โดยคุณสมบัติที่สำคัญของสวิทช์คือมีแรงดันตกคร่อมภายในที่ต่ำที่สุดที่จะเป็นไปได้เพื่อให้ผลที่ได้ผิดพลาด โดยต้องมีคุณสมบัติทางกายภาพดังต่อไปนี้

1. มีความต้านทานภายในไม่เกิน 0.4 โอห์ม
2. สามารถนำกระแสได้สองทิศทาง
3. ทนกระแสและแรงดันได้ 0.5 แอมแปร์ และ 5 โวลต์ตามลำดับ
4. สามารถเลือกสัญญาณได้ 3 ทิศทาง

จากคุณสมบัติที่ต้องการดังกล่าวและจากการวิเคราะห์คุณสมบัติของสวิทช์ชนิดต่างที่มีอยู่ในท้องตลาด แล้วนำมาเปรียบเทียบผู้จัดทำเห็นว่า รีเลย์มีคุณสมบัติเหมาะสมที่สุด โดยมีรีเลย์ที่ใช้คือเบอร์ 5Y-5-K ซึ่งมีคุณสมบัติทางกายภาพดังนี้

1. ความต้านทานภายใน 0.1 โอห์ม
2. สามารถนำกระแสได้ทั้งสองทิศทาง
3. สามารถทนกระแสได้ 2 แอมแปร์
4. แรงดันมากกว่า 100 โวลต์
5. เป็นสวิทช์สองทิศทาง

สามารถทำเป็นสวิทช์ 3 ทิศทางได้โดยการนำรีเลย์สองตัวมาต่ออนุกรมกันทำให้ได้สวิทช์ 3 ทิศทางได้ดังรูปที่ 5.1



รูปที่ 5.1 แสดงการต่อรีเลย์เพื่อทำเป็นสวิตช์สามทาง

5.2 ส่วนที่ใช้ในการควบคุมสวิตช์

ในการควบคุมสวิตช์ผู้จัดทำเลือกใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C51 เนื่องจากความง่ายและหลากหลายในการใช้งาน หาซื้อง่าย ราคาไม่แพง ทำงานร่วมกับวงจรต่างๆ

5.2.1 หลักการทำงานของส่วนควบคุมสวิตช์

ไมโครคอนโทรลเลอร์ทำหน้าที่รับคำสั่งจากผู้ใช้ผ่านทางคอมพิวเตอร์ และทำการส่งค่าอาห์พทไปที่ ชิพรีจิสเตอร์เพื่อขั้วรีเลย์ในการเลือกสัญญาณไปให้กับตัวชิ้นงาน(แพลชเมมโมรี่) ที่เราต้องการทดสอบ โดยไมโครคอนโทรลเลอร์จะรับข้อมูลที่ได้รับมาจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม โดยใช้ความเร็วในการสื่อสารแบบอนุกรม 19200 บิตต่อวินาที ไม่มีบิตตรวจสอบ โดยใช้การทำงานของไทม์เมอร์ ในโหมด 2 เป็นตัวกำหนดความเร็วในการสื่อสารข้อมูล เมื่อไมโครคอนโทรลเลอร์รับข้อมูลจากคอมพิวเตอร์จะนำข้อมูลไปเก็บไว้ที่บัฟเฟอร์ซึ่งมีตำแหน่งเริ่มต้นอยู่ที่ 30H ทำการเก็บข้อมูลจนถึงค่าสุดท้ายซึ่งเป็น ไบต์ปิดของข้อมูล ไมโครคอนโทรลเลอร์ จะทำการเอ็นเอเบิล ชิพรีจิสเตอร์ที่เหมาะสมกับจำนวนพินที่ผู้ใช้เลือก โดยใช้วงจรสำหรับเอ็นเอเบิลดังรูปที่ 5.3 ซึ่งจะอธิบายรายละเอียดในหัวข้อถัดไป แล้วส่งข้อมูลออกทางพอร์ต 0 ไปให้วงจรเรียงข้อมูลและวงจรจับสวิตช์แสดงในรูปที่ 5.4 เพื่อทำการเรียงข้อมูลให้เป็นขบวน ซึ่งรายละเอียดของวงจรในหัวข้อถัดไป แล้วทำการจับข้อมูลออกจากวงจรแลตที่อยู่ใน ชิพรีจิสเตอร์ซึ่งต่ออยู่กับวงจรเอ็นพีเอ็น คาลิงตัน(NPN Darlington) โดยใช้ไอซีเบอร์ ULN 2803 ซึ่งเป็นไอซีสำหรับขั้วรีเลย์ที่ทำหน้าที่เป็นสวิตช์ ที่ใช้เลือกสัญญาณป้อนให้แก่ตัวชิ้นงานที่เราต้องการทดสอบ ซึ่งในส่วนสวิตช์รีเลย์แสดงในรูปที่ 5.5 เมื่อทำการจับข้อมูลทั้งหมดเพื่อขั้ววงจรของสวิตช์รีเลย์ที่เรียบร้อยแล้ว ไมโครคอนโทรลเลอร์ จะทำการส่งข้อมูลเดิมกลับไปให้คอมพิวเตอร์โดยส่งตั้งแต่ไบต์แรกจนถึงไบต์สุดท้าย เพื่อให้คอมพิวเตอร์นำข้อมูลที่ส่งมากลับข้อมูลที่ไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่บนสื่อออนไลน์ การค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่งค่ากลับไปให้เพื่อป้องกันการผิดพลาดที่เกิดจากสายสัญญาณ และรับข้อมูลชุดต่อไป เมื่อข้อมูลชุดต่อไปเข้ามาการทำงานก็จะวนเช่นเดิม

หากผู้ใช้งานต้องการเคลียร์ข้อมูล ไมโครคอนโทรลเลอร์จะทำการส่งคิสมัลติบรีจิสเตอร์ทั้งหมด และทำการกำหนดค่าเริ่มต้นของวงจรใหม่ทั้งหมด

5.2.2 วงจรและการทำงานของวงจร

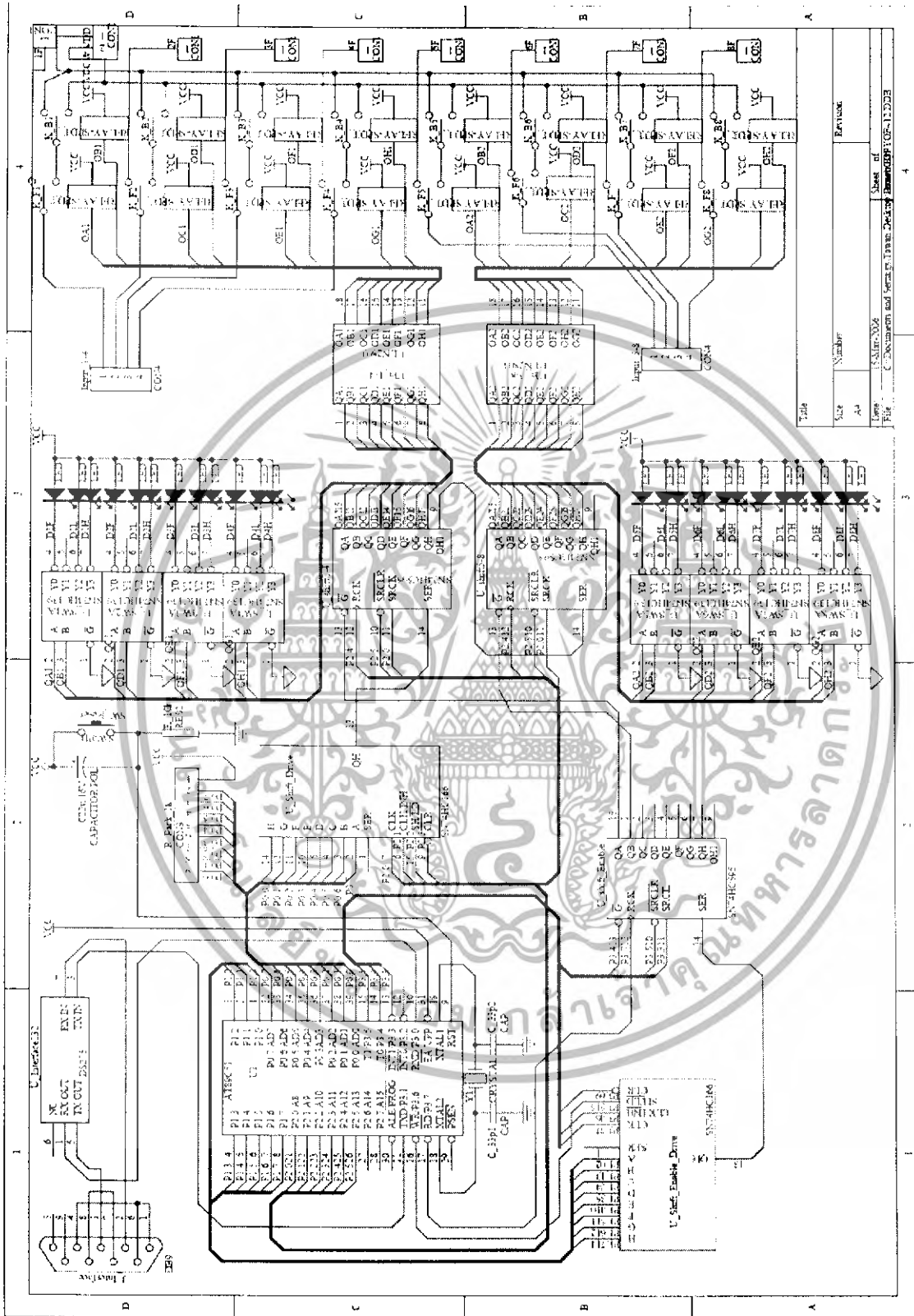
วงจรทั้งหมดสามารถแบ่งออกเป็นส่วนๆตามการทำงานได้ดังนี้

- วงจรในส่วนของไมโครคอนโทรลเลอร์
- วงจรสำหรับเอ็นเอเบิลชิปรีจิสเตอร์
- วงจรสำหรับวงจรจัดเรียงข้อมูลและขับสวิทช์
- วงจรสวิทช์
- วงจรส่วนของการแสดงผล

ซึ่งจะแสดงรายละเอียดดังต่อไปนี้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

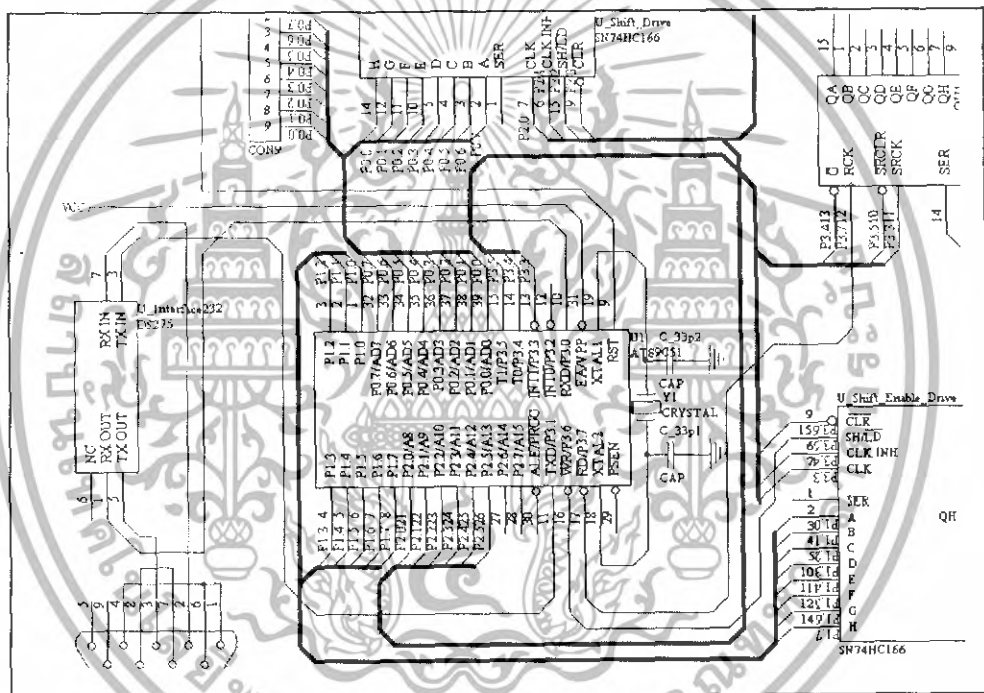


รูปที่ 5.2 ภาพวงจรรวม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่วารณใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. วงจรในส่วนของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์จะทำการรับส่งข้อมูลแบบอนุกรมกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ใช้ไอซีเบอร์ DS275 ซึ่งจะทำหน้าที่แปลงสัญญาณจากมาตรฐาน RS232 ให้เป็นสัญญาณดิจิทัล TTL รับส่งข้อมูลผ่านทางขา Rx(P3.0) และTx (P3.1) เมื่อทำการรับข้อมูลจากคอมพิวเตอร์จนถึงไบต์ที่ใดแล้ว ไมโครคอนโทรลเลอร์จะทำการส่งข้อมูลที่ใช้ในการเอ็นเอเบิลชิปรีจิสเตอร์ผ่านทางพอร์ท 1 ให้กับวงจรเอ็นเอเบิลชิปรีจิสเตอร์ โดยมีสัญญาณควบคุมจากพอร์ท 3 ตั้งแต่บิต 4 ถึง 7 เมื่อทำการเอ็นเอเบิลชิปรีจิสเตอร์แล้วไมโครคอนโทรลเลอร์จะส่งข้อมูลออกไปที่พอร์ท 0 และสัญญาณควบคุมผ่านทางพอร์ท 2 บิต 0 ถึงบิต 5 ไปให้วงจรเรียงข้อมูลและขับสวิทช์ จากข้างต้นแสดงได้ดังรูปที่ 5.3



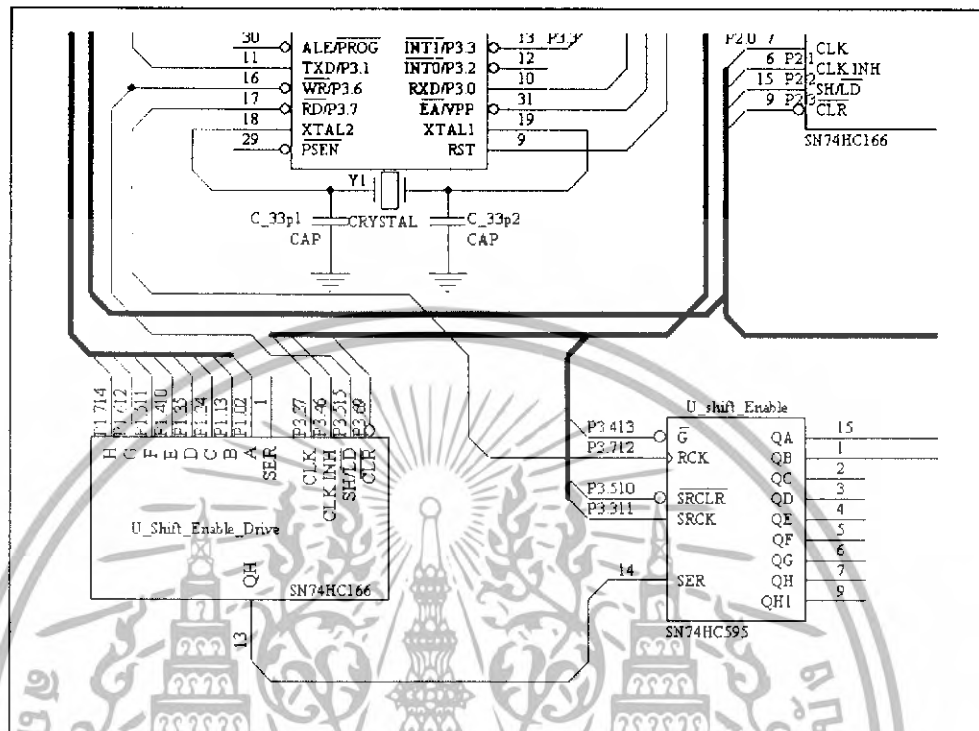
รูปที่ 5.3 แสดงการต่อวงจรในส่วนของไมโครคอนโทรลเลอร์

2. วงจรเอ็นเอเบิลชิปรีจิสเตอร์

วงจรนี้ประกอบด้วยไอซีเบอร์ 74HC166 และ 74HC595 โดยไอซีเบอร์ 74HC166 จะรับข้อมูลสำหรับเอ็นเอเบิลชิปรีจิสเตอร์จากไมโครคอนโทรลเลอร์ผ่านทางพอร์ท 1 และสัญญาณควบคุมผ่านทางพอร์ท 3 และทำการเปลี่ยนจากข้อมูลแบบขนานให้เป็นแบบอนุกรมแล้วส่งไปยังไอซีเบอร์ 74HC595 ซึ่งจะเรียงต่อกัน โดยแต่ละตัวจะสามารถเอ็นเอเบิลชิปรีจิสเตอร์ได้ 8 ตัว กรณี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ชิปรีจิสเตอร์ มากกว่า 8 ตัวก็ต้องต่อไอซี 74HC595 มากกว่าหนึ่งตัวมาต่อกันสามารถแสดงได้ดังรูปที่ 5.4

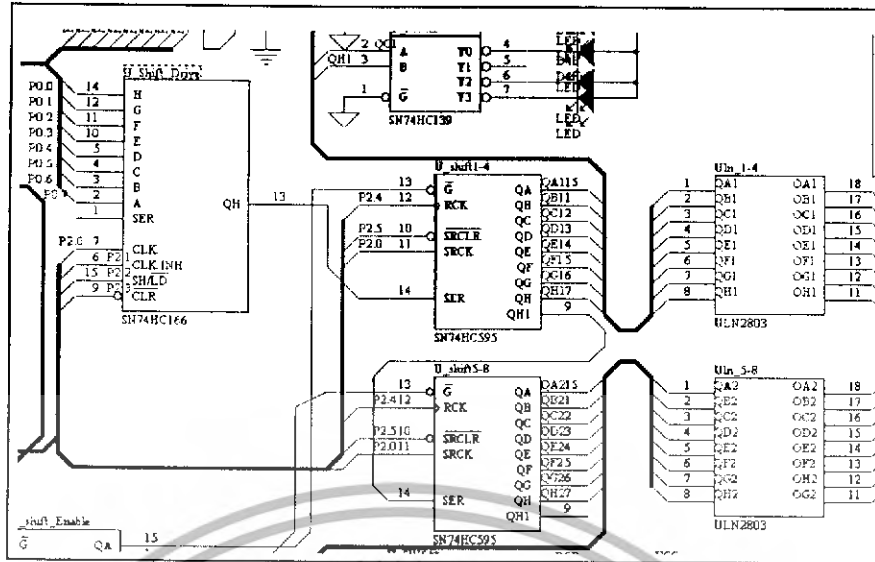


รูปที่ 5.4 แสดงวงจรในส่วนการเอ็นเอเบิลชิปรีจิสเตอร์

3. วงจรสำหรับวงจรจัดเรียงข้อมูลและขับสวิทช์

วงจรในส่วนของการขับสวิทช์จะประกอบด้วยไอซีเบอร์ 74HC166, 74HC595 และ ULN2803 โดยไอซีเบอร์ 74HC166 จะรับข้อมูลจากพอร์ท 0 และสัญญาณควบคุมและสัญญาณนาฬิกาจากพอร์ท 2 แล้วทำการส่งข้อมูลแบบอนุกรมไปยังไอซีเบอร์ 74HC595 ซึ่งได้รับสัญญาณเอ็นเอเบิลจากวงจรเอ็นเอเบิลและสัญญาณนาฬิกา รวมทั้งสัญญาณควบคุมส่วนอื่นๆจากพอร์ท 2 เมื่อส่งข้อมูลจนครบทุกพินแล้วไมโครคอนโทรลเลอร์จะทำการส่งสัญญาณขับข้อมูลทั้งหมดออกพร้อมกันผ่านทางพอร์ท 2 ชิปรีจิสเตอร์ก็จะส่งข้อมูลเพื่อไปขับรีเลย์โดยใช้ไอซีเบอร์ ULN2803 ซึ่งเป็นวงจรคาร์ดินันแบบเปิดคอลเล็กเตอร์ โดยไอซีเบอร์ ULN2803 สามารถขับกระแสสูงสุดได้ถึง 500 มิลลิแอมป์ ซึ่งสามารถขับรีเลย์ได้อย่างสบาย วงจรส่วนนี้แสดงดังรูปที่ 5.4

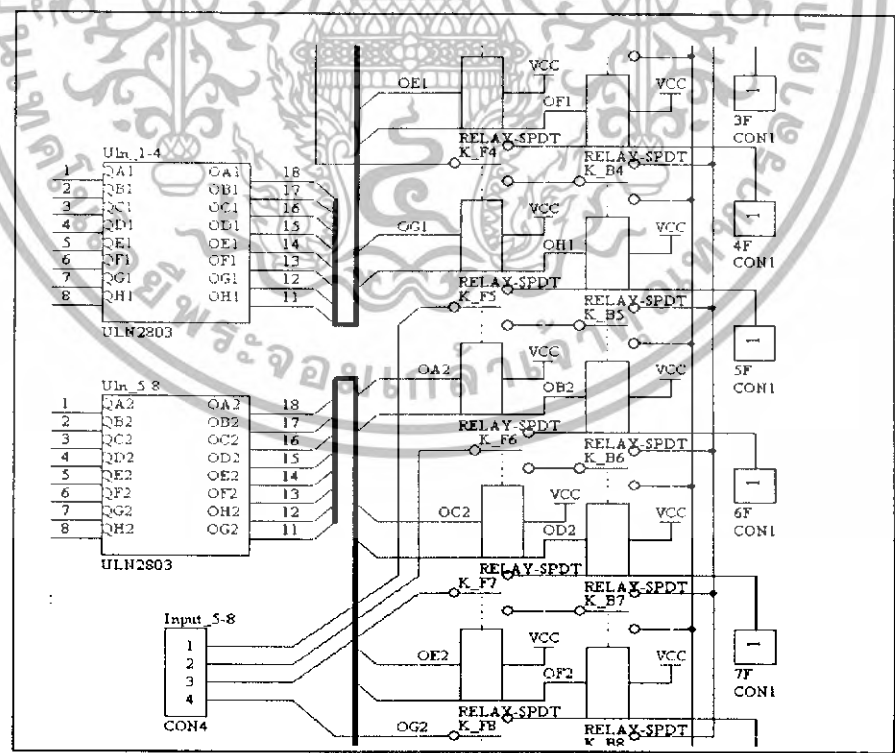
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.5 แสดงส่วนของวงจรทำหน้าที่เป็นวงจรสำหรับวงจรจัดเรียงข้อมูลและขับสวิทซ์

4. วงจรสวิทซ์

วงจรในส่วนนี้จะป็นรีเลย์ต่ออนุกรมกันเหมือนดังการออกแบบข้างต้นพร้อมทั้งทำการต่อสายสัญญาณสำหรับสัญญาณที่ป้อนสัญญาณให้แก่ชิ้นงานและสายสัญญาณที่ต่อเข้าไปที่ขาของชิ้นงานวงจรส่วนนี้แสดงในรูปที่ 5.5



รูปที่ 5.6 วงจรในส่วนกับสวิทซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. วงจรส่วนของการแสดงผล

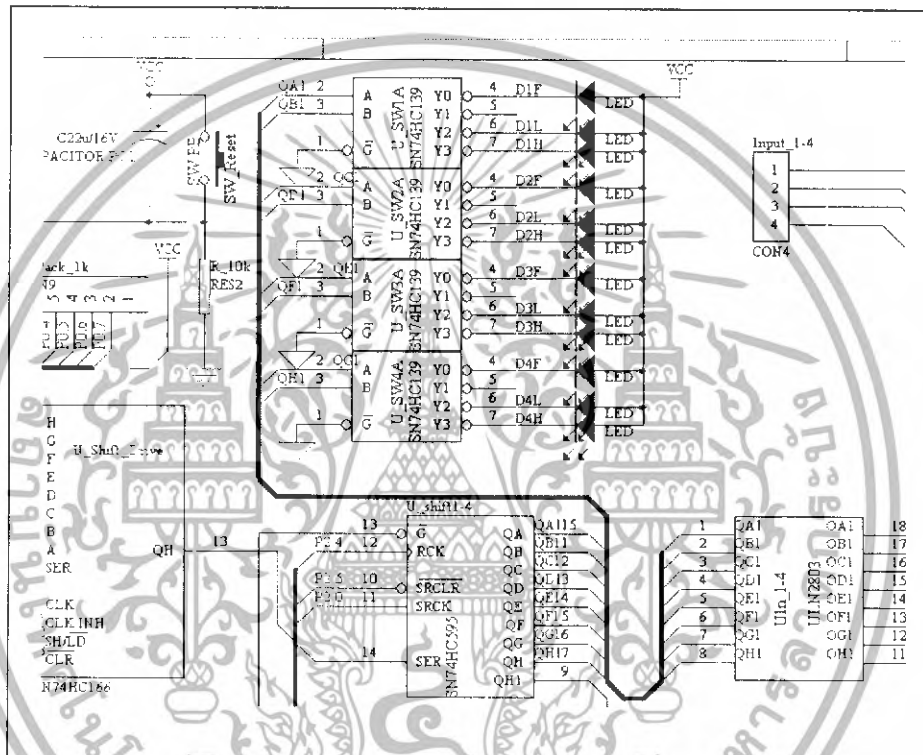
วงจรในส่วนนี้จะประกอบด้วย ไอซีเบอร์ 74HC139 ซึ่งเป็นวงจรมัลติเพล็กซ์ ใช้สำหรับขับ LED เพื่อเป็นการแจ้งการเลือกสวิตช์ของรีเลย์โดยสัญญาณที่ส่งมาขั้วรีเลย์จะมีค่า

00 เป็นสัญญาณ ปล่อยลอย

10 เป็นสัญญาณ VDD

11 เป็นสัญญาณ VSS

ซึ่งสามารถแสดงได้ดังรูปที่ 5.6

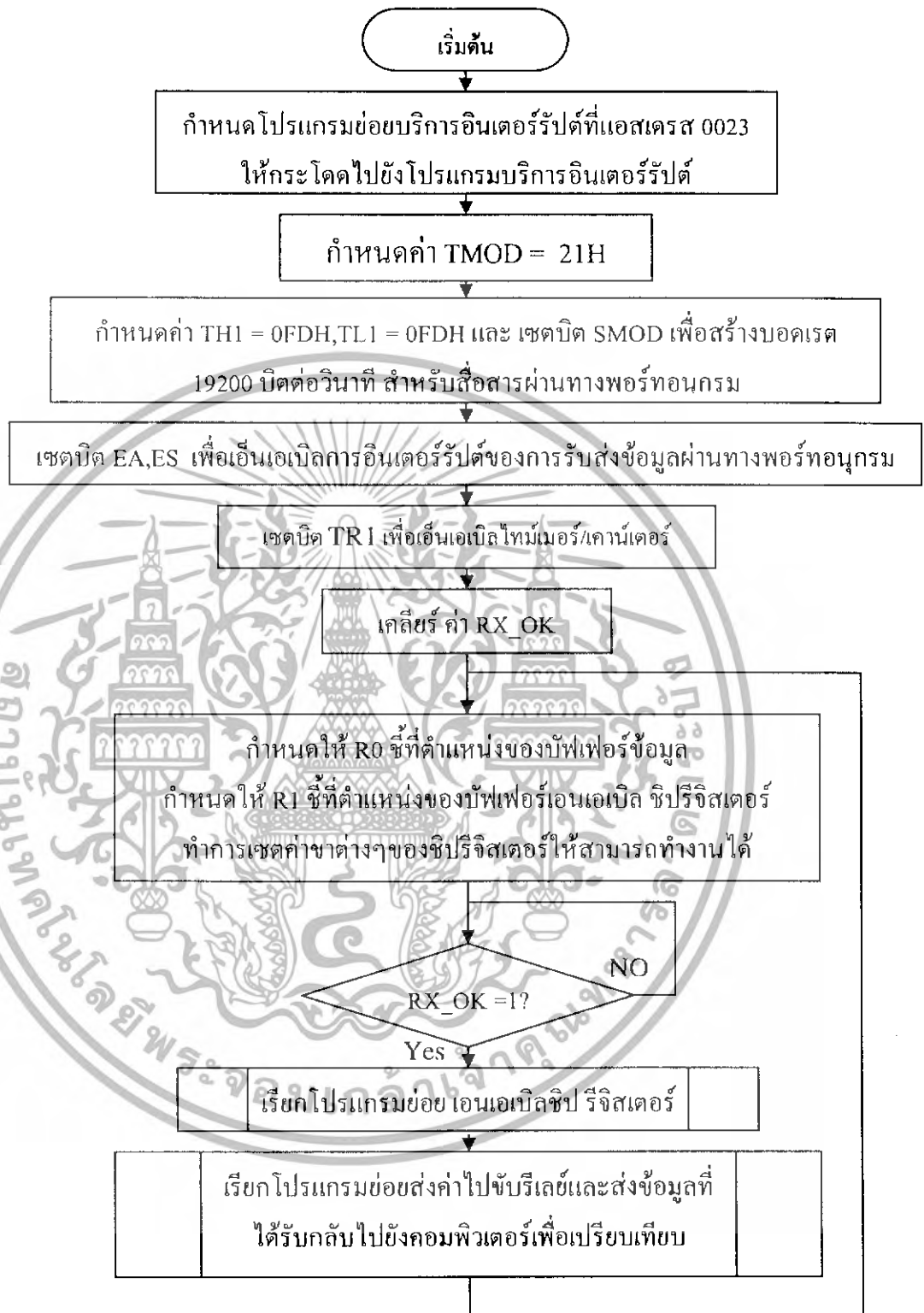


รูปที่ 5.7 แสดงวงจรในส่วนของการแสดงผล

5.2.3 แผนผังการทำงานของโปรแกรมในไมโครคอนโทรลเลอร์

การทำงานของไมโครคอนโทรลเลอร์มีขั้นตอนการทำงานเป็นดังแผนผังการทำงาน (Flow chart) ได้ดังนี้

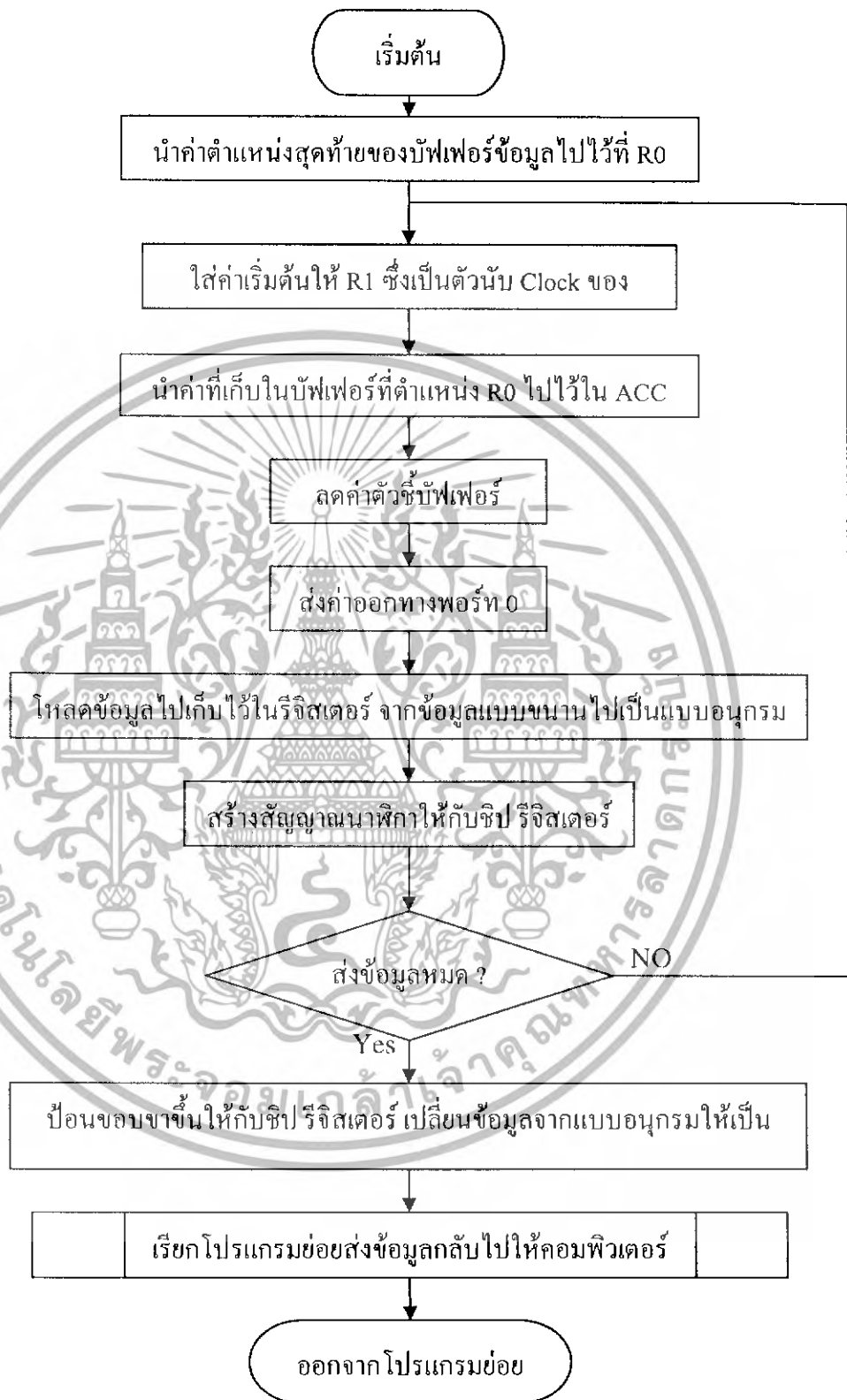
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.8 โฟลวชาร์ตการทำงานของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

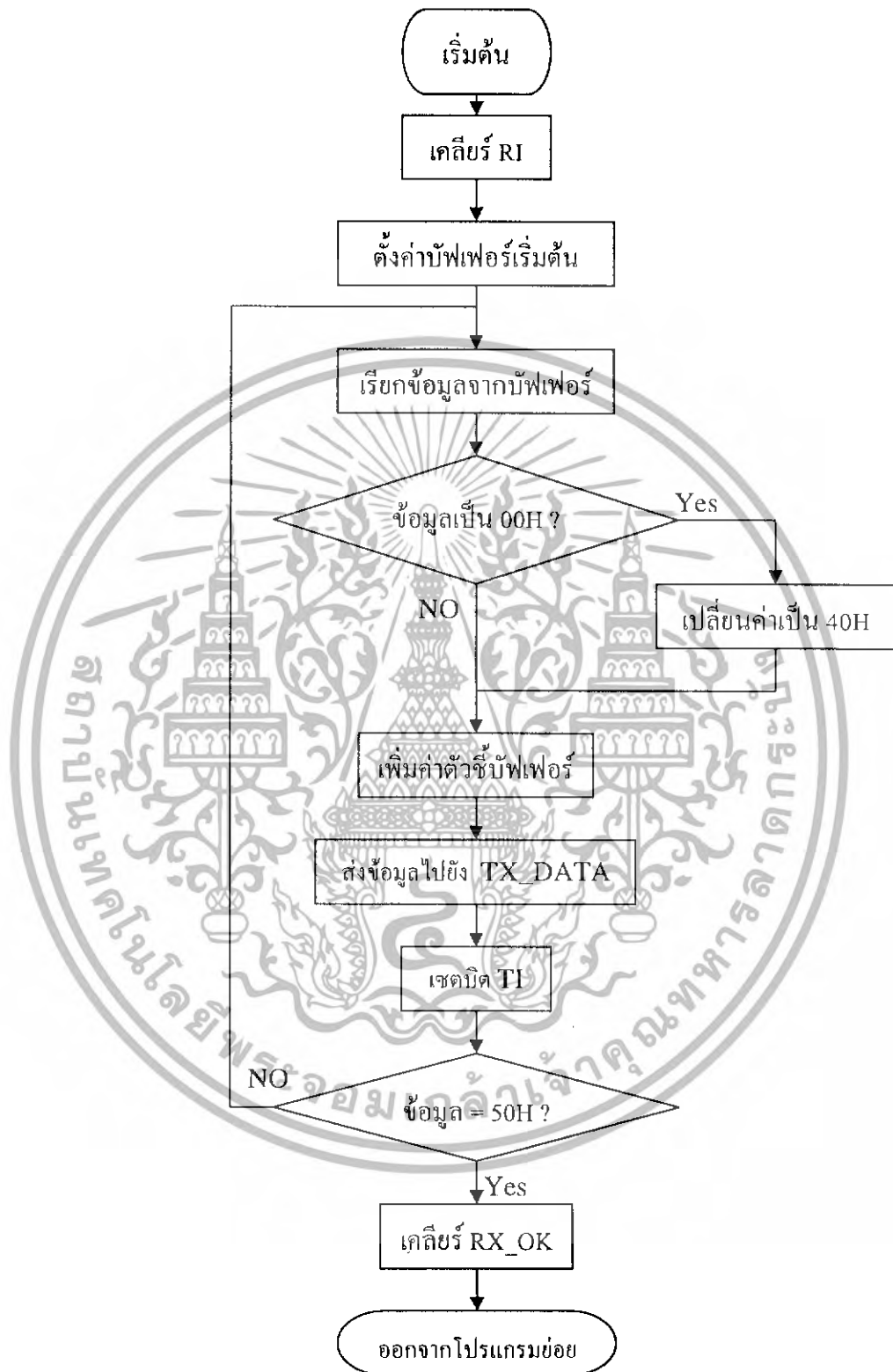
โปรแกรมย่อยส่งค่าเอาต์พุต



รูปที่ 5.9 โฟลวชาร์ทโปรแกรมย่อยส่งค่าเอาต์พุต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

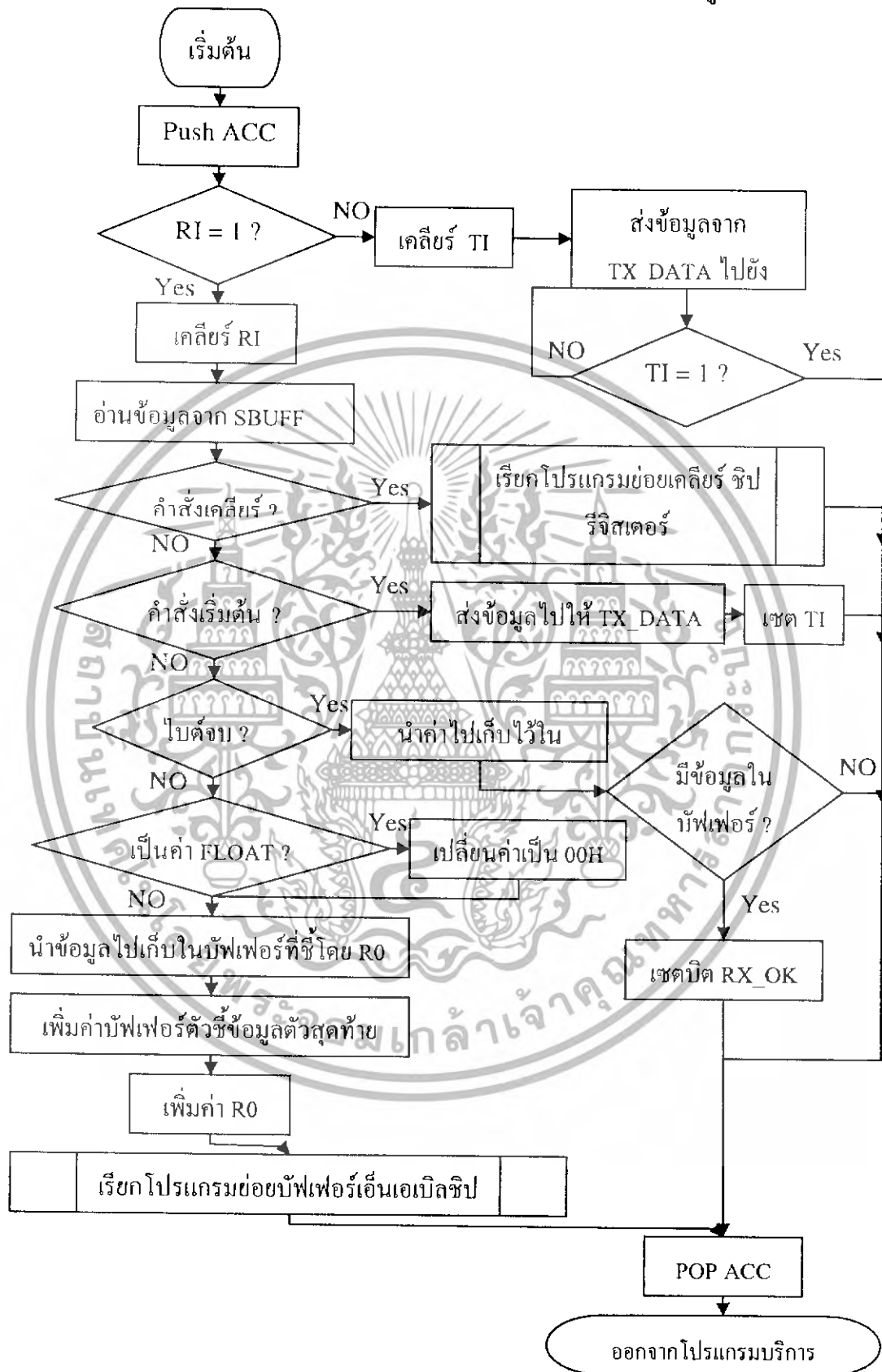
โปรแกรมย่อยส่งค่ากลับไปให้คอมพิวเตอร์



รูปที่ 5.10 โฟลวชาร์ตโปรแกรมย่อยส่งค่ากลับไปให้คอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

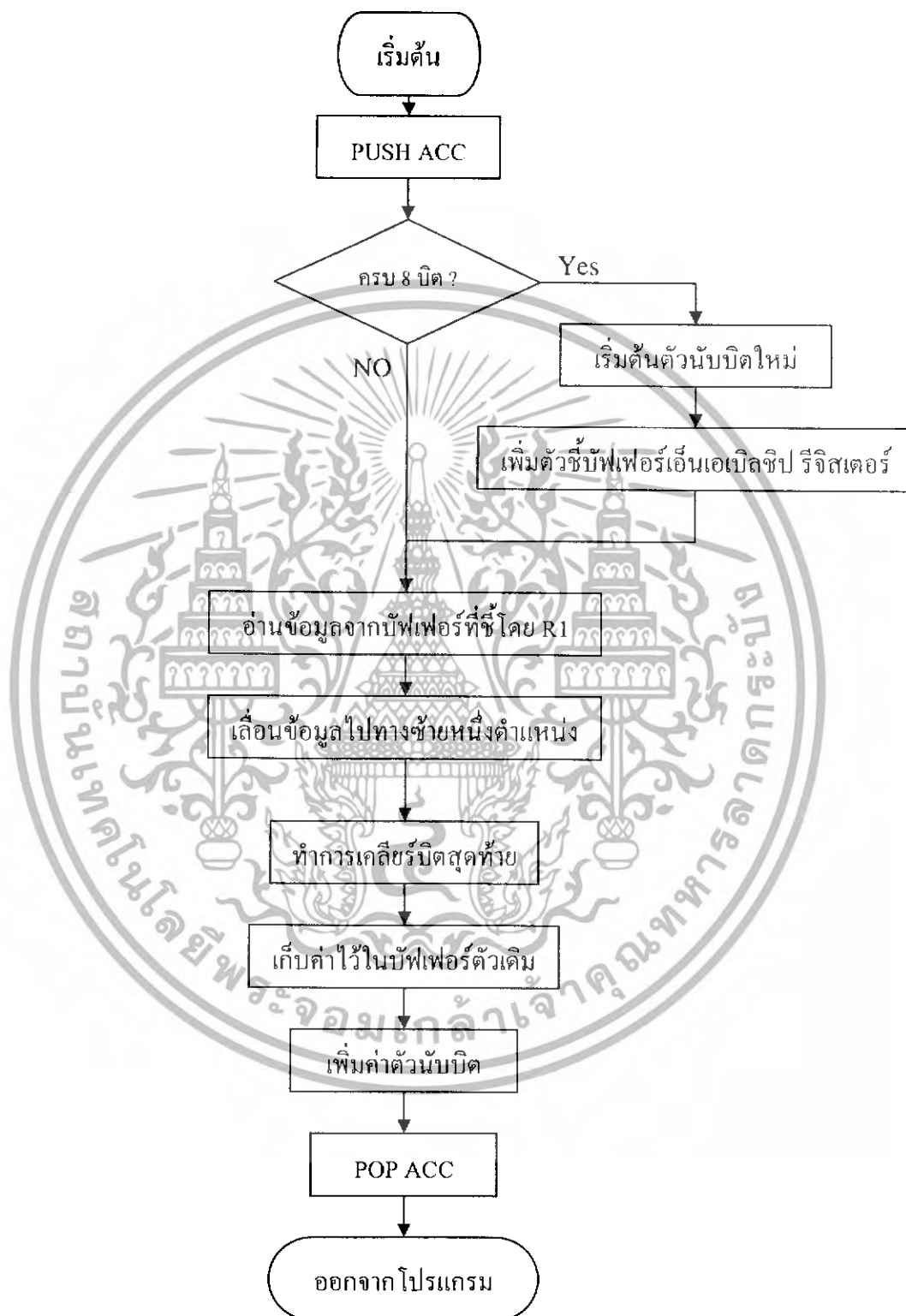
โปรแกรมย่อยบริการอินเทอร์เน็ตรับ-ส่งข้อมูล



รูปที่ 5.11 โฟลวชาร์ตโปรแกรมย่อยบริการอินเทอร์เน็ตรับ-ส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

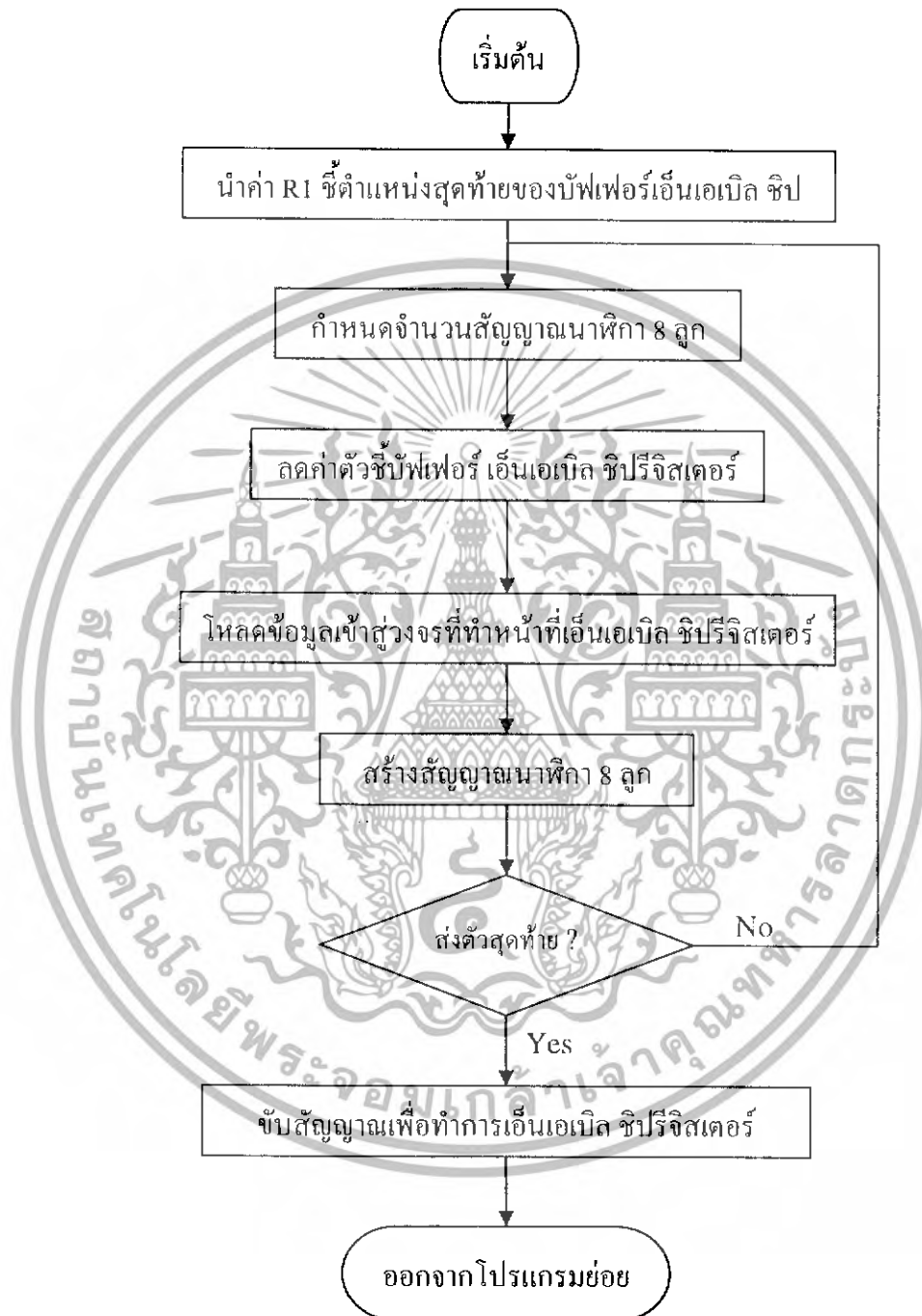
โปรแกรมย่อยบัพเฟอร์เอนเอเบิล ชิพ รีจิสเตอร์



รูปที่ 5.12 โฟลวชาร์ต โปรแกรมย่อยบัพเฟอร์เอนเอเบิล ชิพ รีจิสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยเอ็นเอเบิลชิปริจิสเตอร์



รูปที่ 5.13 โฟลวชาร์ต โปรแกรมย่อยเอ็นเอเบิลชิปริจิสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยเคลียร์ข้อมูล



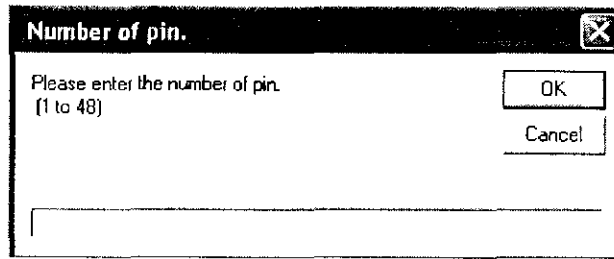
รูปที่ 5.14 โฟลวชาร์ต โปรแกรมย่อยเคลียร์ข้อมูล

5.3 ส่วนที่เป็นการติดต่อระหว่างผู้ใช้งานกับระบบ(User Interface)

5.3.1 หลักการทำงานของโปรแกรม

โปรแกรมที่เลือกใช้ในส่วนของคอมพิวเตอร์ที่เป็นตัวรับคำสั่งจากผู้ใช้งาน คือ โปรแกรม วิซิวัล เบสิก (Visual Basic) เพราะเป็นโปรแกรมที่ได้รับความนิยม มีการใช้งานง่าย โดยเมื่อเปิดโปรแกรม ตัวโปรแกรมจะรับค่าเป็นจำนวนขาของชิ้นงานที่เราต้องการทดสอบ โดยใช้ฟังก์ชัน Message Box ในการออกแบบ

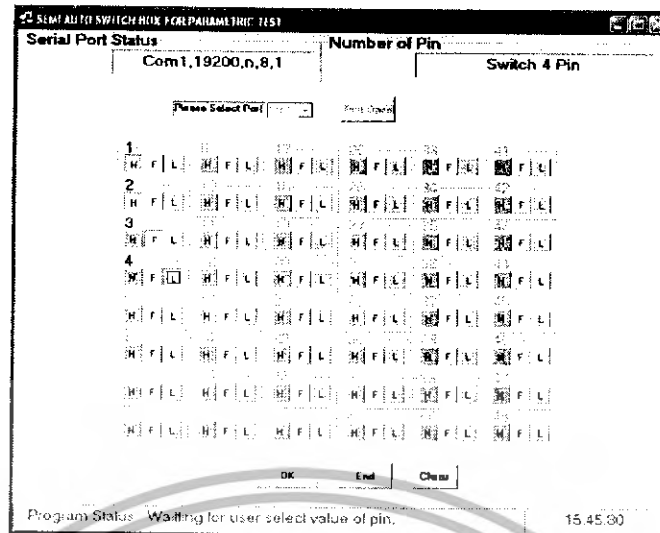
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



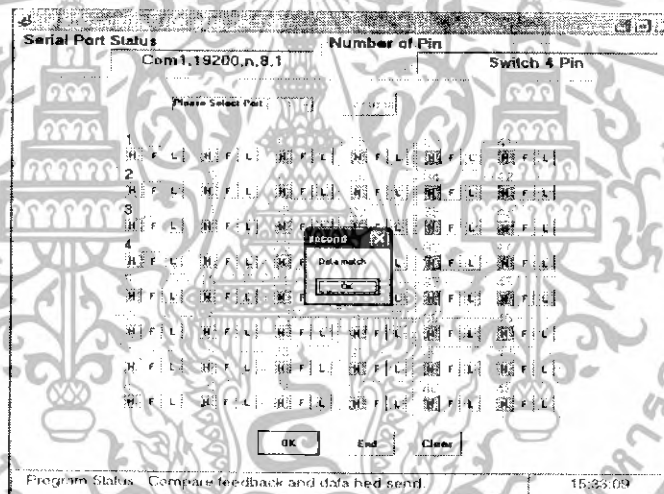
รูปที่ 5.15 แสดงหน้าจอของโปรแกรมเพื่อรับจำนวนขาของชิ้นงาน

โปรแกรมจะทำการตรวจสอบค่าที่ได้รับซึ่งต้องอยู่ในช่วงของตัวเลขจำนวนนับ ตั้งแต่ 1 ถึง 48 เท่านั้น หากนอกเหนือจากนี้โปรแกรมจะแสดงข้อความเตือนผู้ใช้งานเป็นกรณีต่างๆคือ กรณีป้อนตัวเลขนอกเหนือจาก 1-48 หรือ ป้อนตัวอักษร รวมถึงกรณีที่ไม่ได้ป้อนข้อมูลด้วย เมื่อทำการตรวจสอบค่าที่ได้รับเรียบร้อยแล้วโปรแกรมจะทำการขอให้ผู้ใช้งานป้อนหมายเลขพอร์ตที่ต้องการใช้ในการสื่อสาร แล้วทำการเซตค่าเริ่มต้นของพอร์ตให้สามารถใช้งานได้ โดยความเร็วที่ใช้คือ 19200 บิตต่อวินาที ไม่มีโหมดตรวจสอบ ความยาวของข้อมูล 8 บิต และความยาวของบิตหยุด 1 บิต หากพอร์ตนั้นมีปัญหา โปรแกรมจะแสดงข้อผิดพลาดนั้นให้ผู้ใช้งานทราบ เมื่อทำการเปิดพอร์ตสำเร็จแล้ว โปรแกรมจะทำการส่งค่า "0" ออกไปให้ไมโครคอนโทรลเลอร์ แล้วขอให้ไมโครคอนโทรลเลอร์ส่งค่า "0" กลับมา เพื่อเป็นการทดสอบการเชื่อมต่อของสายสัญญาณระหว่างคอมพิวเตอร์และไมโครคอนโทรลเลอร์ ว่าสามารถติดต่อกันได้หรือไม่ หากการติดต่อกันมีปัญหา โปรแกรมจะแสดงข้อผิดพลาดให้ผู้ใช้งานทราบ เมื่อทำการติดต่อกับไมโครคอนโทรลเลอร์เรียบร้อยแล้ว โปรแกรมจะรอรับการเลือกสวิตซ์จากผู้ใช้งานดังรูปที่ 5.8 เมื่อผู้ใช้งานทำการเลือกสัญญาณที่ต้องการได้แล้วทำการกดปุ่ม OK เพื่อทำการส่งข้อมูล โปรแกรมจะทำการส่งข้อมูลไปให้ไมโครคอนโทรลเลอร์พร้อมทั้งรอข้อมูลที่ส่งกลับมาเพื่อมาเปรียบเทียบกับข้อมูลที่ส่งไปว่าเป็นข้อมูลเดียวกันหรือไม่ แล้วทำการแสดงผลการเปรียบเทียบค่าของข้อมูล แล้วโปรแกรมจะกลับมาสู่หน้าต่างรอรับการเลือกสวิตซ์จากผู้ใช้งานครั้งต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.16 แสดงหน้าจอของโปรแกรม Visual Basic เมื่อเข้าสู่สถานะรอรับค่าจากผู้ใช้งาน



รูปที่ 5.17 แสดงตัวอย่างผลการเปรียบเทียบค่าที่ส่งไปให้ไมโครคอนโทรลเลอร์กับค่าที่ได้รับกลับมา

เมื่อผู้ใช้งานต้องการเคลียร์ทุกพินให้อยู่ในตำแหน่งปล่อยลอย สามารถทำได้โดยกดที่ปุ่ม Clear โปรแกรมจะทำการส่งคำสั่งที่เป็นคำสั่งเคลียร์ไปยังไมโครคอนโทรลเลอร์

และถ้าต้องการออกจากโปรแกรมสามารถทำได้โดยการกดปุ่ม End โปรแกรมจะทำการปิดโปรแกรมให้โดยกลับไปสู่หน้า Desktop ของคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.2 ลำดับการทำงานของโปรแกรมที่ใช้ใน Visual Basic

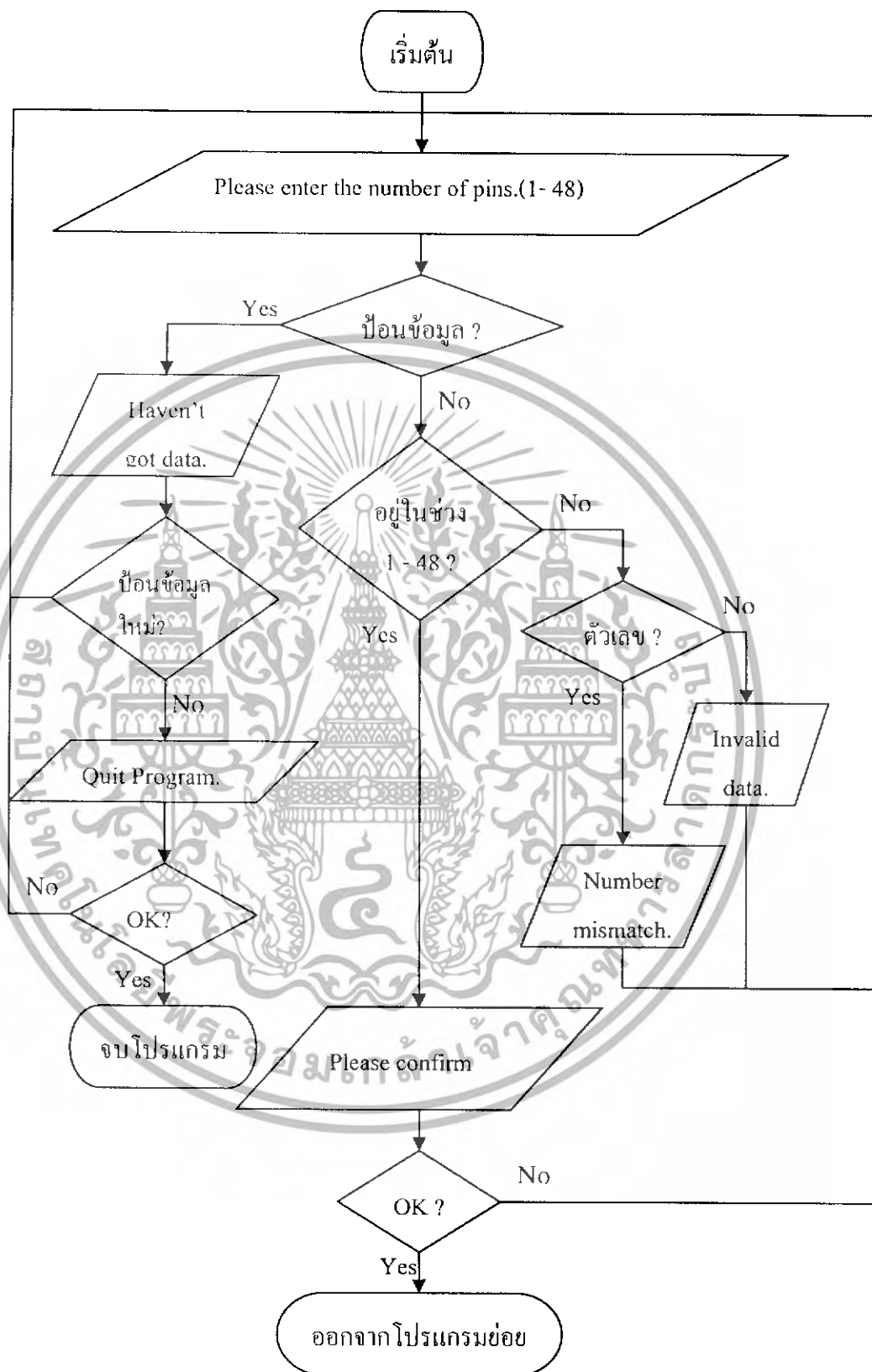
โปรแกรม Visual Basic ใช้ในการสร้างโปรแกรมเพื่อใช้ติดต่อกับผู้ใช้งาน โดยมีโฟลวชาร์ตดังต่อไปนี้



รูปที่ 5.18 โฟลวชาร์ตการทำงานของโปรแกรม Visual Basic

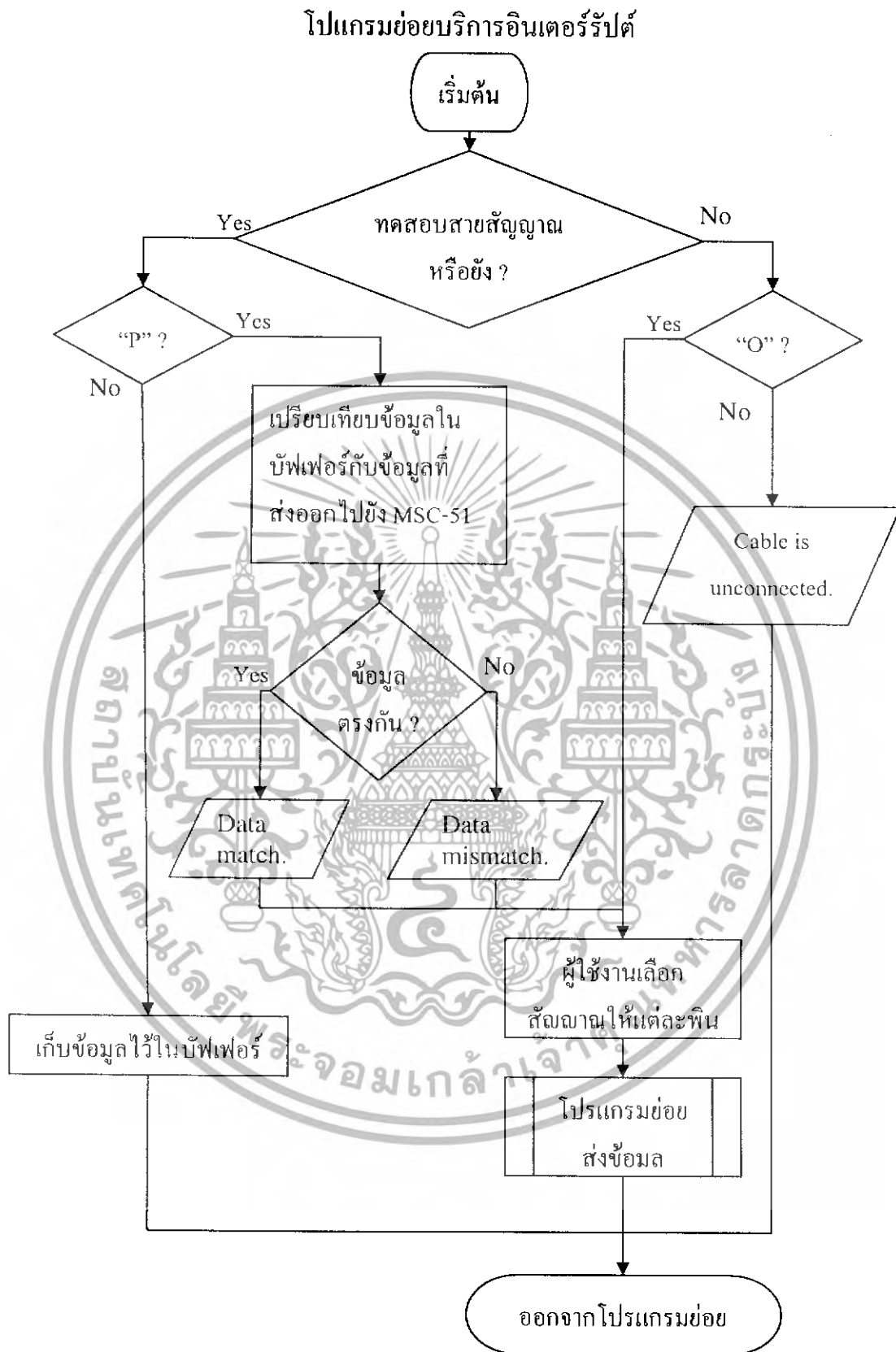
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยรับค่าจำนวนพิน



รูปที่ 5.19 โฟลวชาร์ต โปรแกรมย่อยรับค่าจำนวนพิน

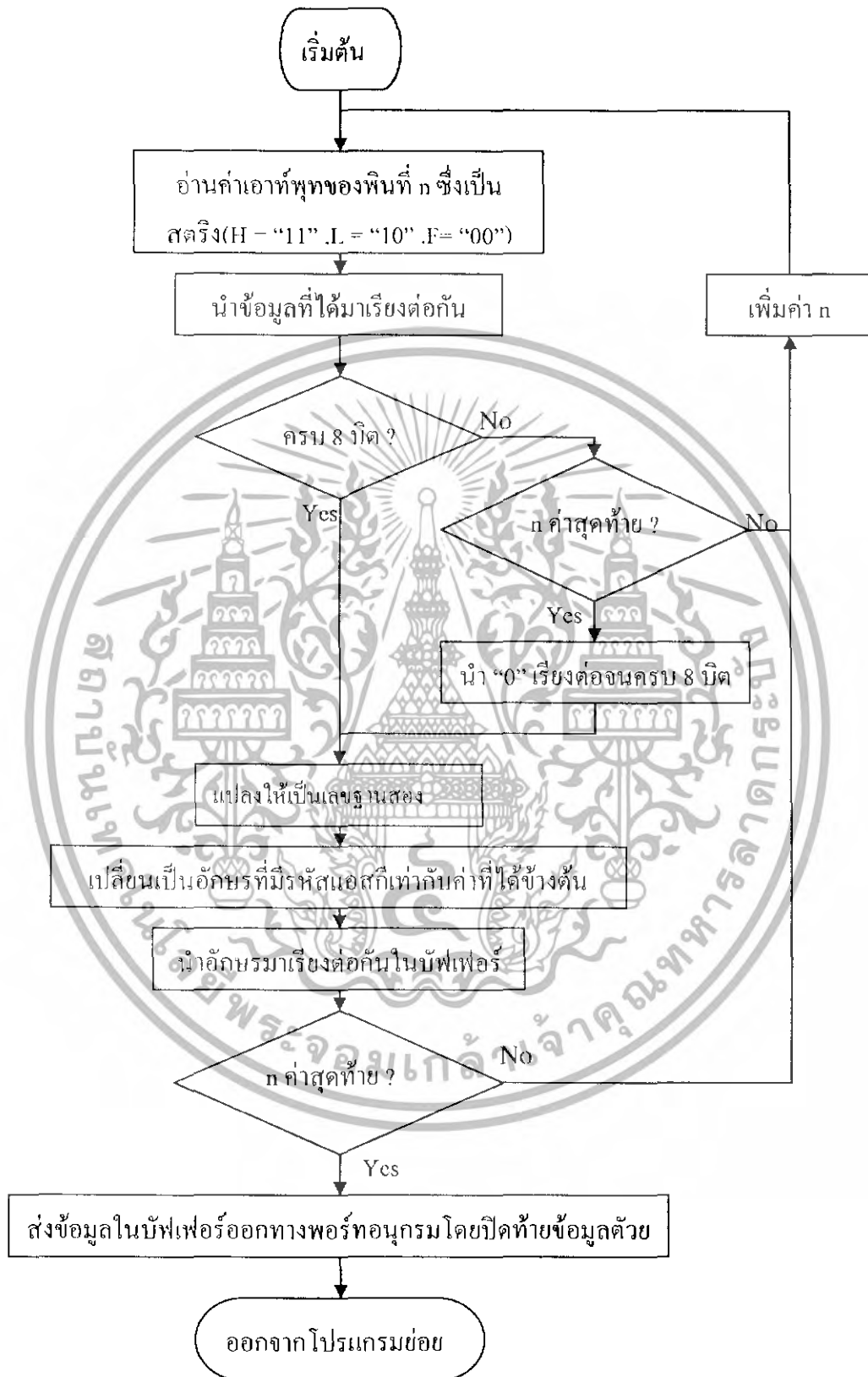
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.20 โปรแกรมย่อยบริการอินเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยส่งข้อมูล



รูปที่ 5.21 ฟLOWชาร์ตโปรแกรมย่อยส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

การทดลองและผลการทดลอง

6.1 โปรแกรมที่เขียนจากโปรแกรม Visual Basic

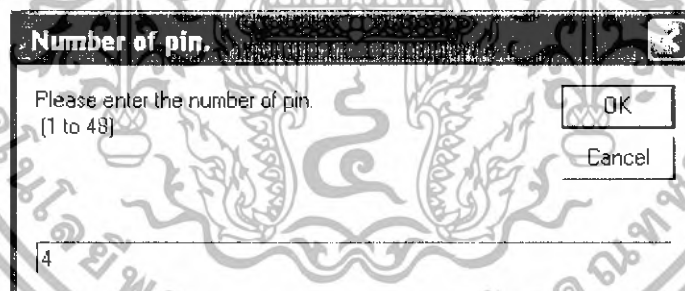
6.1.1 ทดสอบการรับค่าจำนวนขาของตัวชันงาน

- ทำการคลิกที่รูปโปรแกรมบนหน้าจอจะปรากฏจอภาพเป็นหน้าต่างรับค่าจำนวนสวิทช์ของอุปกรณ์ที่จะทดสอบ ดังรูป ที่ 6.1

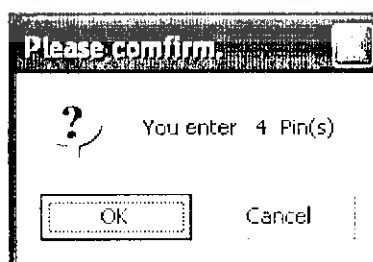


รูปที่ 6.1 แสดงหน้าจอรับจำนวนขาของอุปกรณ์

- ทำการใส่จำนวนขาของชันงานที่ต้องการทดสอบ ทดลองใส่เลข 4 แล้วกด OK จะได้ผลดังรูป ที่ 6.2 และ 6.3



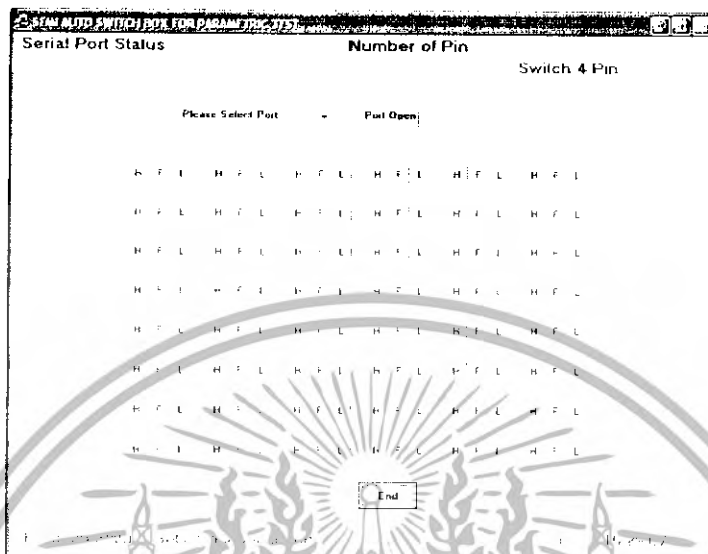
รูปที่ 6.2 แสดงการใส่จำนวนขาของชันงาน



รูปที่ 6.3 แสดงผลของโปรแกรมให้ยืนยันจำนวนขา

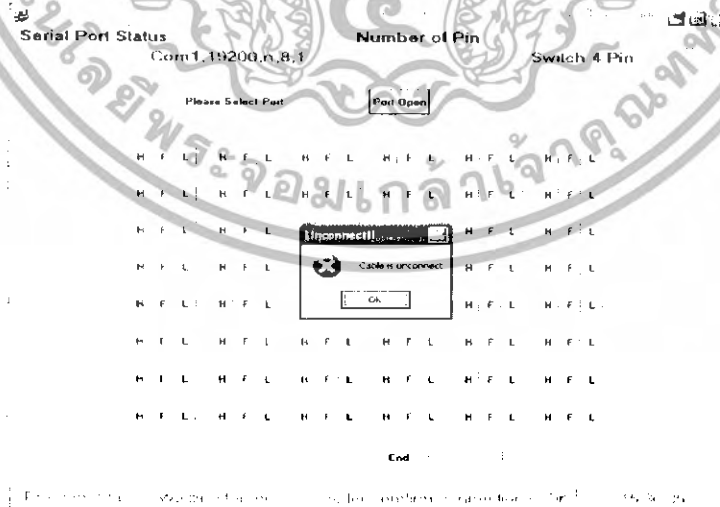
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้ากด Cancel โปรแกรมจะทำการกลับไปรับค่าใหม่
- ถ้ากด OK โปรแกรมจะทำในส่วนต่อไป โดยจะขึ้นหน้าจอของสวิตช์ พร้อมกับให้ไดคอมพอร์ทที่ต้องการใช้รับส่งข้อมูล ดังรูป ที่ 6.4



รูปที่ 6.4 แสดงหน้าต่างของโปรแกรมเพื่อเลือกพอร์ทสำหรับ รับ ส่งข้อมูล

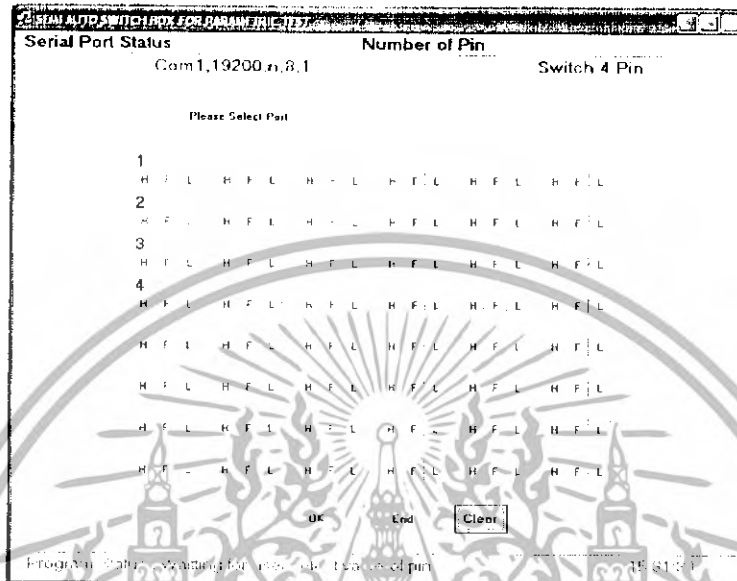
เมื่อเลือกพอร์ทที่จะทำการรับ ส่งข้อมูล โปรแกรมจะทำการทดสอบส่งค่าไปให้ไมโครคอนโทรลเลอร์ เพื่อตรวจสอบการเชื่อมต่อถ้าการเชื่อมต่อมีปัญหาจะขึ้นการผิดพลาด ดังรูป ที่ 6.5



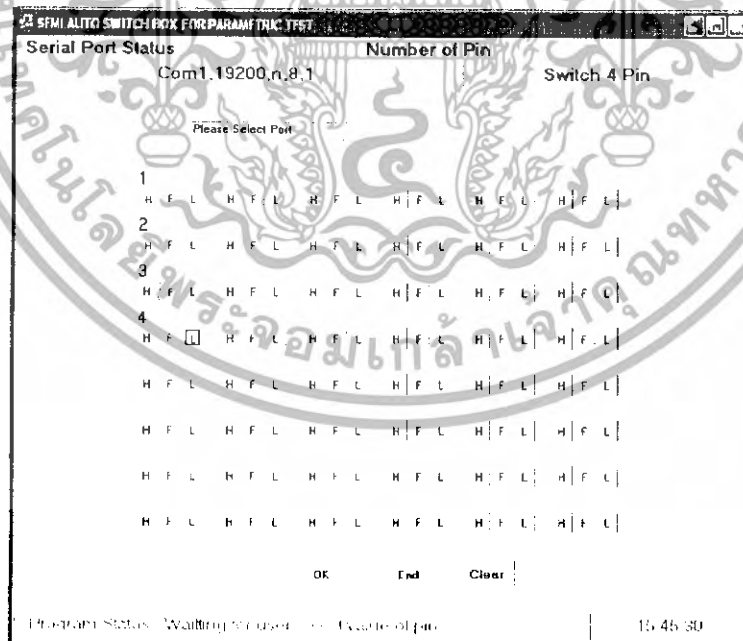
รูปที่ 6.5 แสดงการเกิดการผิดพลาดเนื่องจากไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ไม่สามารถติดต่อกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้าไมโครคอนโทรลเลอร์และคอมพิวเตอร์สามารถติดต่อกันได้ โปรแกรมจะทำการ เอ็นเอเบิลสวิตช์ตามที่เราระบุค่าในตอนแรกและทำการ เอ็นเอเบิลปุ่ม OK แสดงสถานะ รอกำลัง ดังรูป ที่ 6.6 และเมื่อทำการเลือกปุ่ม จะทำให้ปุ่มนั้น Active ดังรูปที่ 6.7



รูปที่ 6.6 แสดงหน้าต่างรอกำลัง

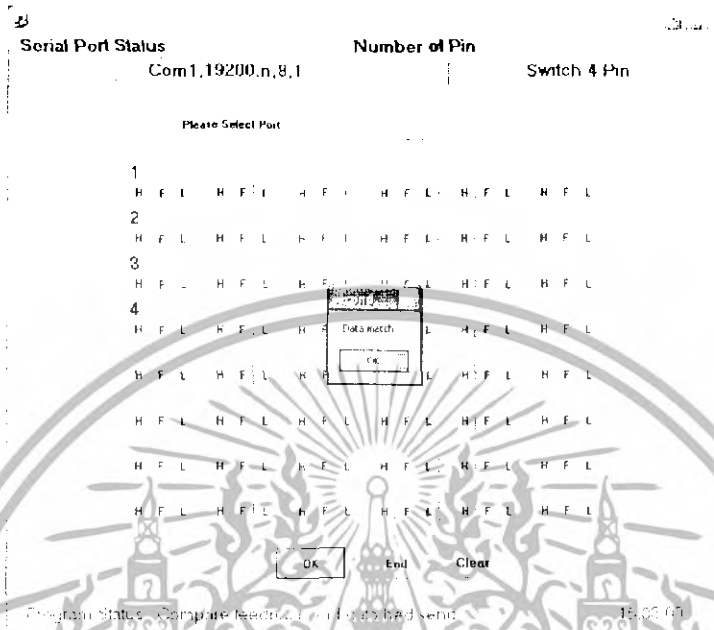


รูปที่ 6.7 เมื่อทำการเลือกสัญญาณ ปุ่มที่เลือกจะ Active

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

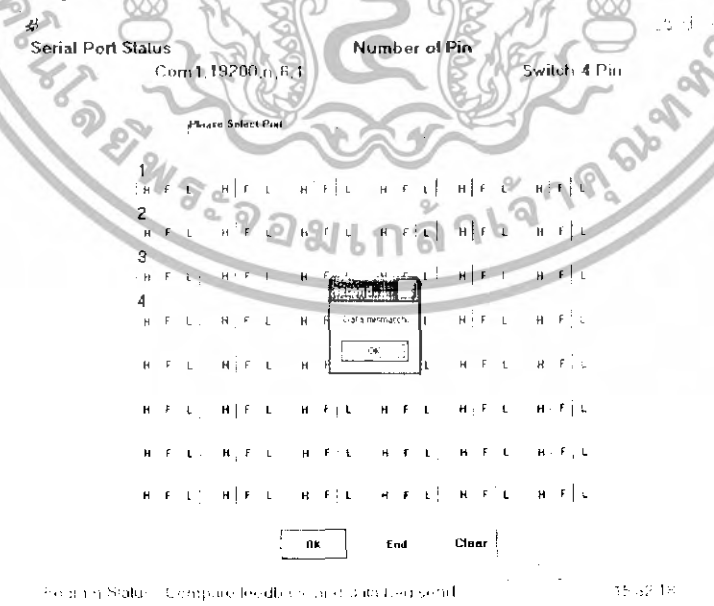
- เมื่อกดปุ่ม OK โปรแกรมจะทำการส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ พร้อมรับค่าที่ไมโครคอนโทรลเลอร์ส่งกลับมา แล้วทำการเปรียบเทียบสัญญาณที่ส่งกับสัญญาณที่รับ

○ ถ้าสัญญาณทั้งสองตรงกัน จะมีหน้าต่างแสดงข้อความ data match ดังรูปที่ 6.8



รูปที่ 6.8 แสดงหน้าต่างเมื่อข้อมูลที่ได้รับมาตรงกัน

○ ถ้าสัญญาณทั้งสองไม่ตรงกัน จะมีหน้าต่างแสดงข้อความ data mismatch ดังรูปที่ 6.9

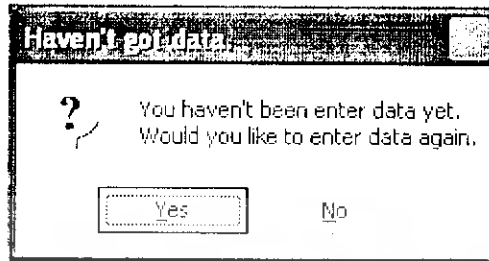


รูปที่ 6.9 แสดงหน้าต่างเมื่อข้อมูลที่ได้รับมาไม่ตรงกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2 ทดสอบการเช็คเมื่อผู้ทดสอบไม่ได้ใส่ค่าใดๆ

- ในกรณีที่ไม่ได้ใส่ค่าใดๆก่อนกดปุ่ม OK โปรแกรมจะทำการแจ้งการผิดพลาดบอกแก่ผู้ใช้งานว่ายังไม่ได้ใส่ข้อมูลที่ป็นตัวเลข ดังรูปที่ 6.10



รูปที่ 6.10 แสดงการผิดพลาดกรณีที่ผู้ใช้ไม่ได้ใส่ข้อมูลให้แก่วงจร

- ถ้าตอบ Yes โปรแกรมจะกลับไปสู่หน้าต่างให้ใส่จำนวนขาของอุปกรณ์ใหม่
- ถ้าตอบ No โปรแกรมจะถามว่าต้องการออกจากโปรแกรมหรือไม่ ดังรูปที่ 6.11



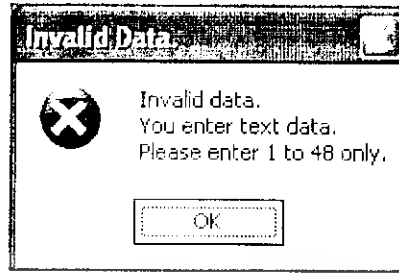
รูปที่ 6.11 แสดงหน้าต่างถามว่าจะออกจากโปรแกรมหรือไม่

- ถ้าตอบ Cancel โปรแกรมจะกลับไปหน้ารับจำนวนขาของอุปกรณ์ใหม่
- ถ้าตอบ OK โปรแกรมจะทำการออกจากโปรแกรมทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.3 ทดสอบการเช็คเมื่อใส่ข้อความที่ไม่ใช่ตัวเลข

ทดสอบโปรแกรมโดยการป้อนข้อมูลไปให้โคบใส่ข้อมูลที่ไม่ใช่ตัวเลขจะได้ผลดังรูปที่ 6.12



รูปที่ 6.12 หน้าต่างเมื่อเอนข้อมูล Tawan



รูปที่ 6.13 หน้าต่างของโปรแกรมเมื่อเอนข้อมูล 123T

- ถ้าตอบว่า Yes โปรแกรมจะกลับไปรับข้อมูลใหม่
- ถ้าตอบว่า No โปรแกรมจะถามว่าจะออกจากโปรแกรมหรือไม่ เหมือนกับข้อ 1.2

6.1.4 ทดสอบเมื่อใส่ค่าที่ไม่ใช่ค่าที่อยู่ระหว่าง 1 – 48

ทำการทดสอบการตรวจเช็คข้อมูล เมื่อป้อนข้อมูลเกินช่วง 1 ถึง 48 ว่าโปรแกรมจะมีการแสดงผลอย่างไร โดยเมื่อทำการป้อนค่าดังกล่าวโปรแกรมจะแสดงหน้าต่างดังรูปที่ 6.14



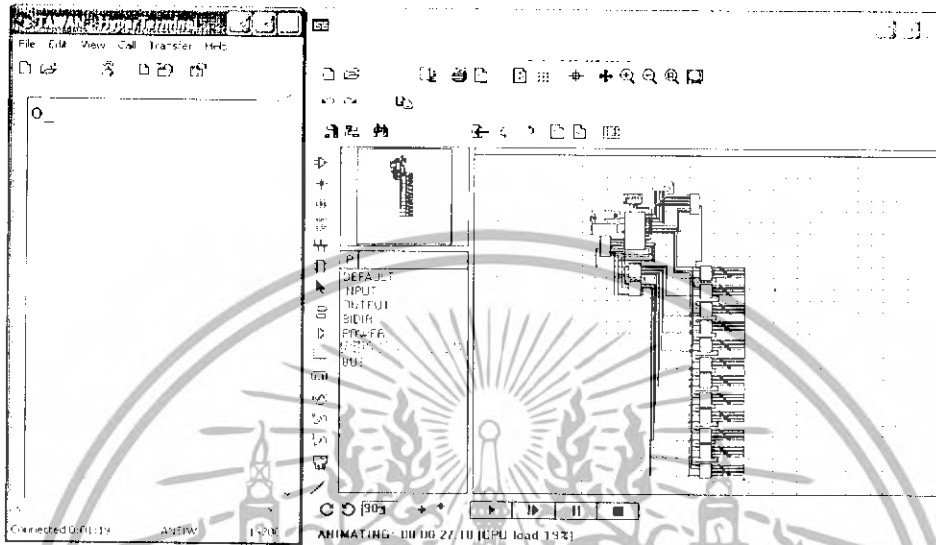
รูปที่ 6.14 แสดงหน้าต่างเมื่อใส่จำนวนข่าไม่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 ไมโครคอนโทรลเลอร์

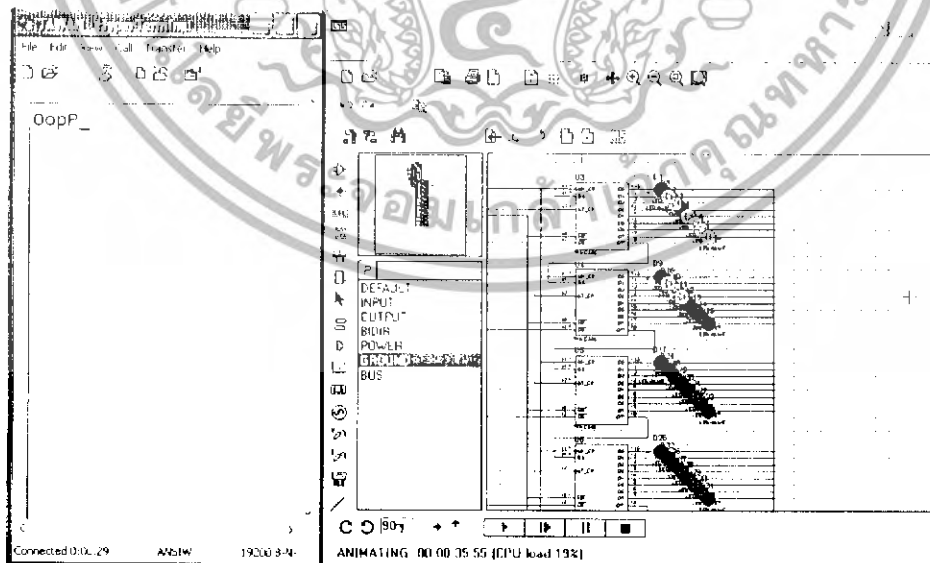
6.2.1 ทดสอบการทำงานในโปรแกรม Proteus

- ทดลองส่งค่า “0” ให้ไมโครคอนโทรลเลอร์เพื่อทดสอบสายสัญญาณจะได้รับค่า “0” กลับ แสดงว่าสายสัญญาณไม่มีปัญหา ดังรูปที่ 6.15



รูปที่ 6.15 แสดงการทดลองส่งค่า “0” ไปยังไมโครคอนโทรลเลอร์ เพื่อทดสอบสายสัญญาณว่ามีปัญหาหรือไม่

- ทดลองส่ง ข้อมูล 2 ตัวอักษร แล้วตามด้วยตัวอักษร “P” โดย อักษร P เป็นไบต์ปิดของข้อมูล จะได้ผลดังรูปที่ 6.16

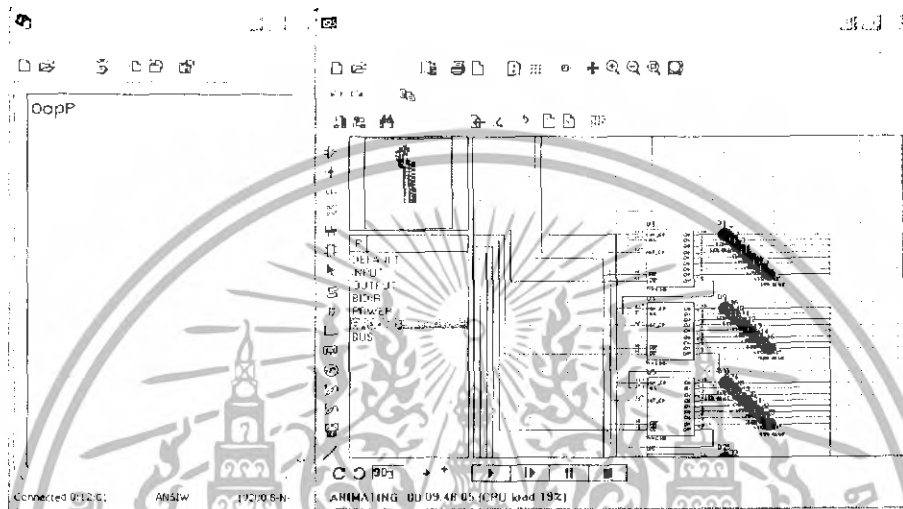


รูปที่ 6.16 เมื่อส่งข้อมูลไปยัง MCS- 51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

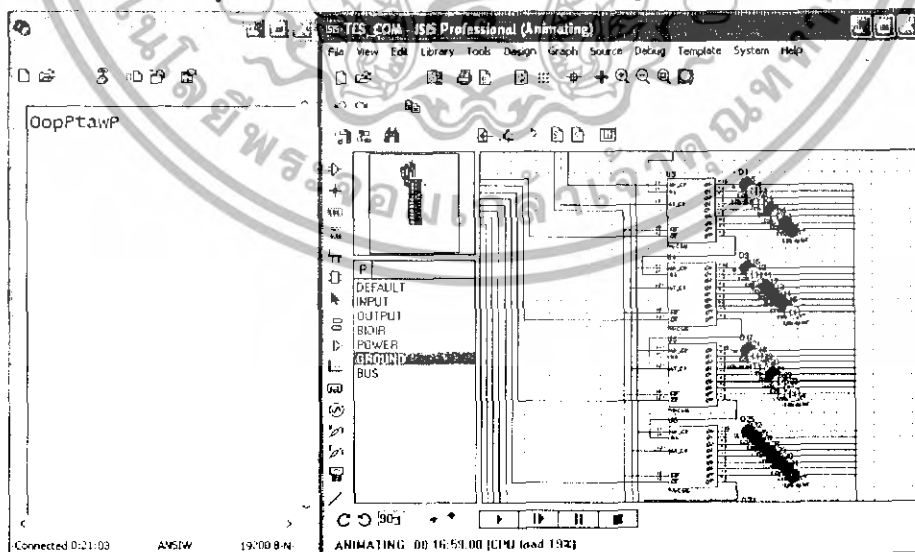
เมื่อส่งข้อมูลไปยัง MCS-51 จะทำให้ 74595 ทำงาน โดยส่งค่าเป็นรหัสแอสกี ของตัวอักษรที่เราส่งไป จะเห็นว่าเมื่อส่ง ตัวอักษร op ซึ่งมีรหัสแอสกี 6F และ 70 ในเลขฐาน 16 ตามลำดับ จากรูปจะเห็นว่า LED สว่าง ตามค่าแอสกีดังกล่าวโดย LED ที่อยู่บนสุดจะแสดงค่าบิตสูงสุด

- ทดลองส่งอักษร "U" ซึ่งกำหนดให้เป็นไบต์เคลียร์ ไอซี 74HCT595 จะได้ผลดังรูปที่ 6.17



รูปที่ 6.17 แสดงผลเมื่อทำการส่ง "U" ไปยัง MCS-51
เมื่อทำการส่ง "U" ไปให้ MCS-51 จะเป็นคำสั่งให้ทำการเคลียร์ เอพท

- ทดลองส่งข้อมูลไปยัง MCS-51 อีกครั้ง จะได้ผลดังรูปที่ 6.18



รูป 6.18 แสดงผลเมื่อทำการส่งข้อมูลไปยัง MCS-51 อีกครั้ง

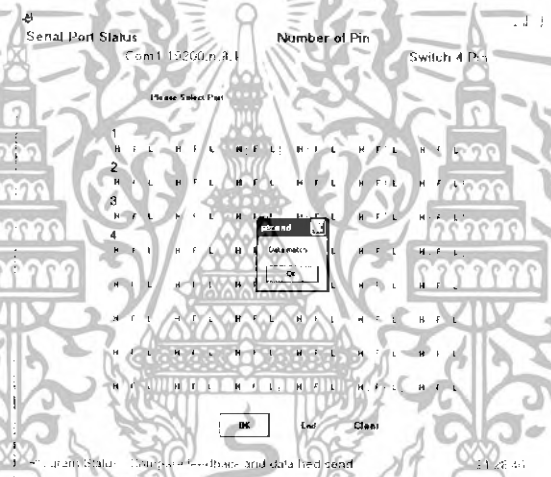
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อทำการส่งข้อมูล ไปยัง MCS-51 อีกครั้งหลังจากทำการส่งค่าเคลียร์ โดยส่งข้อมูลเพิ่ม เป็น 3 ไบต์คือ low ซึ่งมีค่ารหัสแอสกีคือ 74, 73 และ 77 ในเลขฐาน 16 ตามลำดับ จากรูปจะเห็น LED ทำงานตามรหัสแอสกี

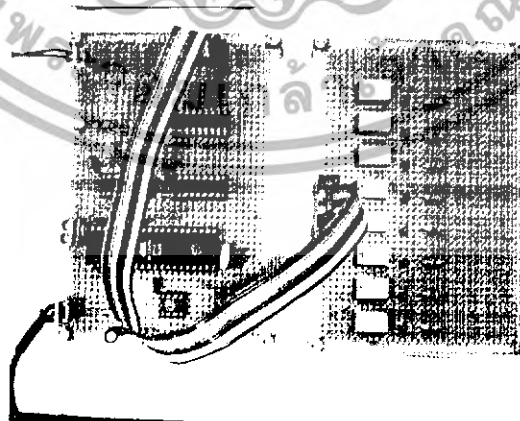
6.2.2 การทดลองทางเครื่องต้นแบบ

เป็นการทดลองโดยการต่อเครื่องต้นแบบกับพอร์ตอนุกรมที่ควบคุมโดยโปรแกรมที่เขียน จากโปรแกรม Visual Basic โดยจากเครื่องต้นแบบ LED ที่อยู่ด้านบนจะเป็นบิตสูงสุด และด้านล่าง จะเห็นบิตต่ำสุด โดยแต่หนึ่จะใช้ 2 บิต พินที่หนึ่งอยู่ด้านบน เรียงกันจนถึงพินที่สี่ ซึ่งจะ ได้ผลดังนี้

- เมื่อป้อนสัญญาณ Float ทั้งสี่พิน โดยการคลิก F ทั้ง 4 พินดังรูปที่ 6.19 จะได้ผลที่ ออกทางเครื่องต้นแบบ ดังรูปที่ 6.20



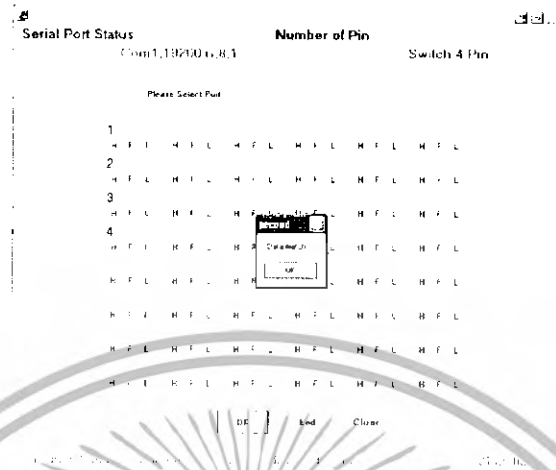
รูปที่ 6.19 เมื่อคลิกปุ่ม F ทั้ง 4 พินแล้วคลิก OK



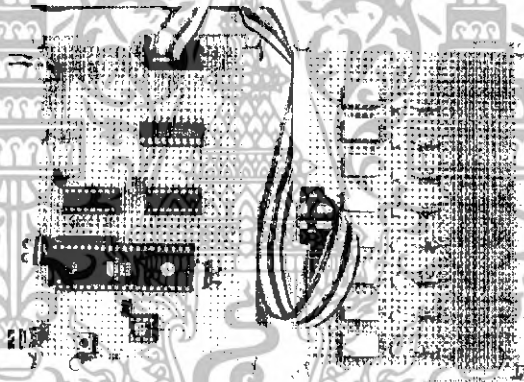
รูปที่ 6.20 ผลที่ออกที่เครื่องต้นแบบเมื่อป้อน Float ทั้ง 4 พิน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อป้อนสัญญาณ High ทั้ง 4 พิน โดยการคลิกที่ปุ่ม H ทั้ง 4 พินดังรูปที่ 6.21 จะได้ผลที่ออกทางเครื่องต้นแบบดังรูปที่ 6.22



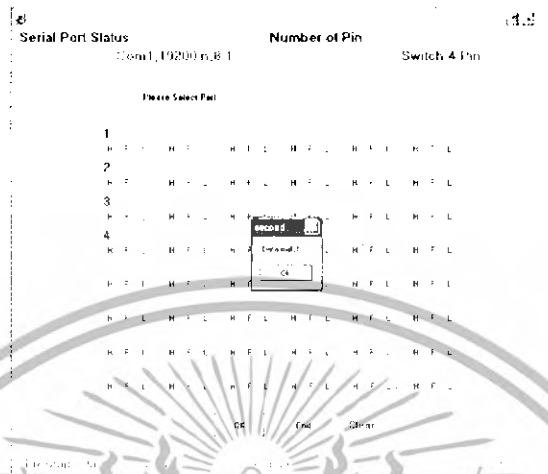
รูปที่ 6.21 เมื่อคลิก H ทั้ง 4 พินแล้วคลิก OK



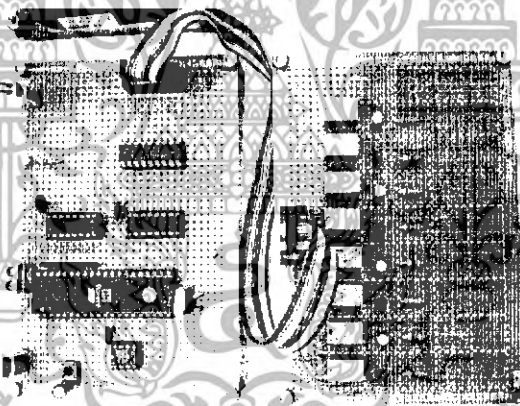
รูปที่ 6.22 ผลที่ออกที่เครื่องต้นแบบเมื่อป้อน High ทั้ง 4 พิน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อไอออนสัญญาณ Low ทั้ง 4 พิน โดยการคลิกที่ปุ่ม L ทั้ง 4 พินดังรูปที่ 6.23 จะ
ได้ผลที่ออกทางเครื่องต้นแบบดังรูปที่ 6.24



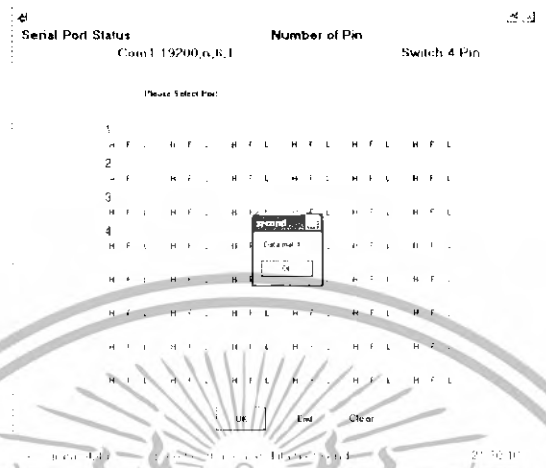
รูปที่ 6.23 เมื่อคลิก L ทั้ง 4 พินแล้วคลิก OK



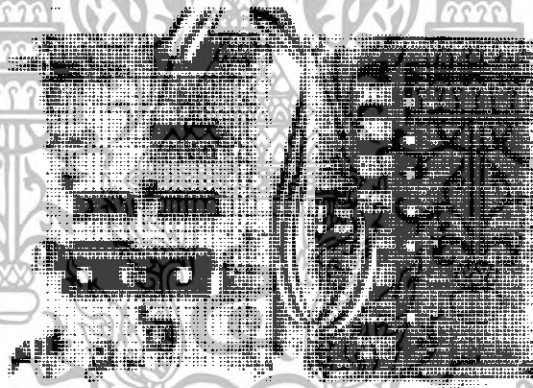
รูปที่ 6.24 ผลที่ออกที่เครื่องต้นแบบเมื่อไอออน Low ทั้ง 4 พิน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อป้อนสัญญาณ 2 พินแรกเป็น High และ สองพินสุดท้ายเป็น Low ดังรูป 6.25 และจะได้ผลดังรูปที่ 6.26



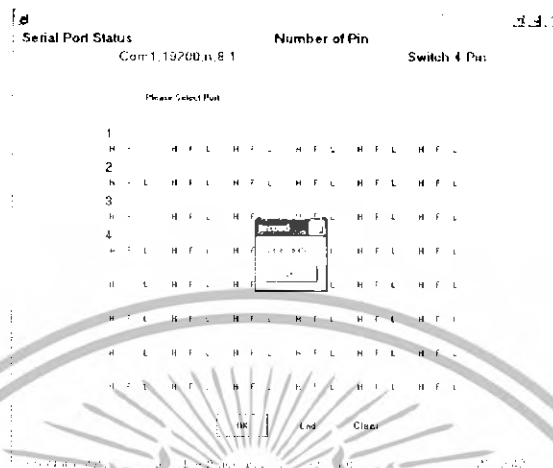
รูปที่ 6.25 เมื่อ 2 พินแรกคลิก H และ L ใน 2 พินหลัง



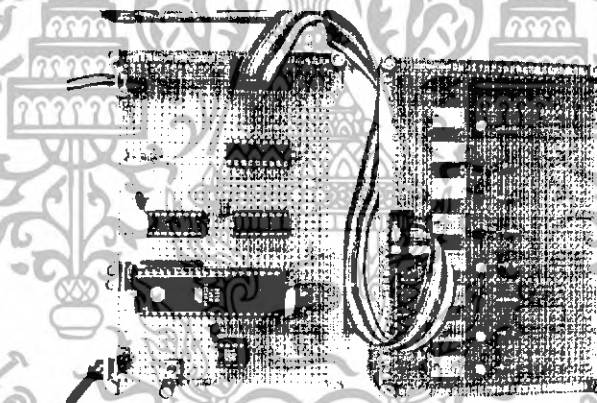
รูปที่ 6.26 ผลเมื่อ ป้อน High 2 พินแรก และ Low 2 พินหลัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อป้อนสัญญาณ Low , Float ,Low และ High ตามลำดับ ดังรูป 6.27 และจะได้ผลดังรูปที่ 6.28



รูปที่ 6.27 เมื่อคลิก L,F,L และ H ตามลำดับ



รูปที่ 6.27 ผล เมื่อป้อน แต่ละพินเป็น Low,Float,Low และ High ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

บทสรุป

ในการออกแบบแผงวงจรสวิทช์จะเริ่มที่การเลือกชนิดของสวิทช์ที่จะนำมาใช้งาน ต้องมีการทดสอบสัญญาณที่ได้รับมาจากตัวอุปกรณ์น้อยที่สุด ควรสามารถทนกำลังไฟที่จะเกิดขึ้นจากการสูญเสียภายในในการทดสอบชิ้นงานได้ โดยไม่ทำให้เกิดความเสียหายแก่ตัวชิ้นงาน การออกแบบวงจรที่ใช้ร่วมกับเครื่องทดสอบแบบพารามตริก สำคัญอยู่ที่ส่วนที่เป็นสวิทช์ควรสามารถในการถ่ายทอดสัญญาณที่ทดสอบได้เป็นอย่างดี และสวิทช์ไม่ควรสร้างสัญญาณรบกวนแก่ระบบหรือมีสัญญาณรบกวนน้อยที่สุด

การควบคุมสวิทช์เมื่อเลือกสวิทช์ที่มีคุณสมบัติที่ต้องการได้ ปัญหาต่อมาคือการควบคุมสวิทช์ ไม่ว่าจะเป็นสวิทช์แบบแมคคานิคหรืออิเล็กทรอนิกส์ ถ้าควบคุมให้ทำงานไม่ได้ตามที่เรารต้องการก็ไม่สามารถเลือกสัญญาณที่ต้องการได้ การใช้ไมโครคอนโทรลเลอร์ในการควบคุมสวิทช์นั้นมีข้อดีตรงที่สามารถออกแบบการใช้งานง่าย มีคำสั่งไม่มากนัก มีพอร์ตให้ใช้งานมากมาย ในการสร้างวงจรสำหรับขับสวิทช์ สิ่งที่สำคัญคือจะต้องให้ทุกสวิทช์ทำงานได้พร้อมกัน

ส่วนที่เป็นการติดต่อกับผู้ใช้งานนั้น การใช้คอมพิวเตอร์เป็นสิ่งที่เหมาะสมในการใช้งานในปัจจุบันเพราะสามารถสร้างจากโปรแกรมเกือบทุกภาษา และมีข้อจำกัดในการใช้งานน้อย การที่เลือกใช้โปรแกรม Visual basic เพราะเป็นโปรแกรมที่สามารถศึกษาการใช้งานได้ง่ายไม่ยุ่งยากสำหรับผู้เริ่มการเขียนโปรแกรมใหม่ๆ ไม่ว่าจะเป็นการทำความเข้าใจหรือว่าจะเป็นการนำไปพัฒนาต่อไป

จากการทดลองจะเห็นว่าโปรแกรมที่สร้างจากโปรแกรม Visual basic มีความสามารถในการสื่อสารได้กับไมโครคอนโทรลเลอร์ได้อย่างมีประสิทธิภาพ การที่ให้มีการส่งข้อมูลจากไมโครคอนโทรลเลอร์กลับไปยังคอมพิวเตอร์ ถึงแม้จะทำให้เสียเวลาพอสมควรแต่ก็สามารถรับประกันได้ว่าระหว่างการส่งสัญญาณผ่านสายสัญญาณ ไม่มีความผิดพลาด หากมีความผิดพลาดในการส่งสัญญาณก็สามารถตรวจสอบได้ง่าย

ปัญหาที่เกิดขึ้นในการทำโครงการชิ้นนี้คือ การสร้างวงจรบนแผ่นปริ้นท์ไปปลา การจัดเรียงตัวอุปกรณ์ การเชื่อมต่อสายสัญญาณเป็นสิ่งที่สำคัญมาก หากเชื่อมต่อไม่ดีเช่นการที่สายสัญญาณขนานกันจำนวนมากจะเกิดสัญญาณรบกวนจนทำให้วงจรไม่ทำงานตามที่ออกแบบไว้ ปัญหานี้เกิดขึ้นได้เกือบทุกวงจร ดังนั้นการแก้ปัญหาที่ดีที่สุดคือการต่อวงจรลงบนแผ่นลายวงจร ซึ่งจะลดสัญญาณรบกวนได้เป็นอย่างมาก อนึ่งการต่อวงจรบนลายวงจรจะป้องกันปัญหาการช็อตกันของสายสัญญาณที่อยู่ใกล้เคียงกันได้มาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. กิตติ ภัคคีวัฒนะกุล, “Visual Basic6 ฉบับโปรแกรมเมอร์” ,เคทีพี คอมพ์ แอนด์ คอนซัลท์ ,621 หน้า,2547
2. จิรศักดิ์ เหลืองอุไร, “คัมภีร์การใช้งานการสื่อสารอนุกรมบน PC”. ซีเอ็ดยูเคชั่น, 372 หน้า, 2538
3. ชูชัย ธนสารตั้งเจริญ, “การสื่อสารข้อมูล”, ฟิสิกส์เซ็นเตอร์, 203 หน้า
4. ชีรวัฒน์ ประกอบผล, “การประยุกต์ใช้งานไมโครคอนโทรลเลอร์”, สมาคมส่งเสริมเทคโนโลยี(ไทย – ญี่ปุ่น), 235 หน้า, 2544
5. วรพจน์ กรแก้ววัฒนกุล, “เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51”, อินโนเวติฟ เอ็ดเจอร์เมนท์, 399 หน้า
6. สัจจะ จรัสรุ่งรวีร, “คู่มือโปรแกรม Visual Basic 6 ฉบับผู้เริ่มต้น”, ไอดีซี อินโฟ คิสทริบิวเตอร์ เซ็นเตอร์,432 หน้า ,2548
7. อภิชาติ ภูพัตถ์ , “เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic” , อินโฟเพรส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้