กิตติกรรมประกาศ

การศึกษาวิจัยครั้งนี้สำเร็จได้นั้นข้าพเจ้าขอขอบคุณหน่วยงานคือ บริษัท กสท. โทรคมนาคม จำกัด (มหาชน) ที่ให้ความรู้ ประสบการณ์จากการทำงานด้านสื่อสารโทรคมนาคมมาโดยตลอด ขอขอบคุณเพื่อนร่วมงานที่ให้คำแนะนำให้ความร่วมมือและให้ความช่วยเหลือต่าง ๆ จนทำให้งานนี้สำเร็จลุล่วงด้วยดี

กราบขอบพระคุณ ครู อาจารย์ทุกท่านที่ประสิทธิ์ประสาทความรู้ให้ตั้งแต่เริ่มต้นการศึกษาจนถึงบัดนี้ ที่ให้คำปรึกษา คำแนะนำและให้แนวคิดต่าง ๆ ตลอดจนคณาจารย์บุคลากร ในภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ ทุกท่านที่ให้ความอนุเคราะห์และช่วยเหลือในด้านต่าง ๆ เป็นอย่างดี โดยเฉพาะอาจารย์ ผู้ช่วยศาสตราจารย์ ดร. มนูญ ศรีวิรัตน์ อาจารย์ที่ปรึกษาที่คอยช่วยเหลือให้คำแนะนำและข้อคิดเห็นในการศึกษาในครั้งนี้

ขอขอบคุณ นางลลิตา รัตนะ พนักงานพัสดุ งานพัสดุ ศูนย์หัวใจสิริกิติ์ ภาคตะวันออกเฉียงเหนือ มหาวิทยาลัยขอนแก่น ที่ช่วยจัดรูปเล่มและตรวจทานเอกสารต่าง ๆ ให้จนสำเร็จ

และสุดท้ายนี้ ขอขอบคุณมิตรภาพอันดีจากเพื่อนๆ นักศึกษาปริญญาโทวิทยาศาสตร์มหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศทุก ๆ ท่าน ที่ให้ความช่วยเหลือ ให้กำลังใจ จนสามารถทำการศึกษาวิจัยครั้งนี้ได้สำเร็จ

(นายธวัชชัย สิงหัษฐิต)
ผู้วิจัย

บทคัดย่อ

ชื่อเรื่อง : ระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง
โดย : ธวัชชัย สิงหัษฐิต
ชื่อปริญญา : ปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชา : เทคโนโลยีสารสนเทศ
ประธานกรรมการที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. มนูญ ศรีวิรัตน์

ศัพท์สำคัญ : เฝ้าตรวจสอบ ไมโครคอนโทรลเลอร์ อุปกรณ์ไฟฟ้ากำลัง พอร์ตอนุกรม กราฟิก

เป็นการออกแบบและสร้างระบบการเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง (Main power Monitoring and Logging System) เพื่อใช้ตรวจสอบการทำงานของอุปกรณ์เหล่านี้ หากเกิดการทำงานผิดปกติหรือเมื่อเกิดเหตุการณ์ไฟฟ้าขัดข้องเจ้าหน้าผู้ปฏิบัติงานจะสามารถแก้ปัญหาได้อย่างถูกต้องและรวดเร็ว ในการทำงานนั้นจะใช้ไมโครคอนโทรลเลอร์ในการรับสัญญาณจากตัว Sensor ที่ใช้ตรวจสอบแรงดันไฟฟ้า แล้วเปลี่ยนเป็นสัญญาณอนุกรม RS-232C ผ่านสายโทรศัพท์ธรรมดา 1 คู่สาย เข้าไปยังห้องควบคุม (Control room) ต่อเข้าพอร์ตอนุกรม RS-232C ของไมโครคอมพิวเตอร์

เขียนโปรแกรมการทำงานของอุปกรณ์ไฟฟ้ากำลังเสมือนจริง มีหน้าต่างแสดงสภาวะการทำงานของอุปกรณ์ไฟฟ้ากำลัง เป็นรูปภาพแบบกราฟิกและมีหน้าต่างที่ใช้บันทึกเหตุการณ์ต่าง ๆ ที่เกิดขึ้นเพื่อใช้ตรวจสอบและใช้อ้างอิงในภายหลังได้ โดยจะใช้โปรแกรม Visual Basic 6 ในการพัฒนา

ABSTRACT

TITLE : MAIN POWER MONITORING AND LOGGING SYSTEM

BY : TAWATCHAI SINGHUSSATIT

DEGREE : MASTER OF SCIENCE

MAJOR : INFORMATION TECHNOLOGY

CHAIR : ASST. PROF. MANOON SRIVIRAT, Ph.D.

KEYWORDS : MONITORING / MICROCONTROLLER / MAIN POWER / SERIAL PORT/ GRAPHIC USER INTERFACE (GUI)

This thesis is about the designing and prototyping of the main power monitoring and logging system. The system regularly checks the power system functionality and, in case of system failure, indicates the system errors to reduce the system downtime. The sensors send the detected voltages to the microcontroller in order to format RS-232C digital signal. The serial port output is then transmitted via regular telephone cable to destination serial port of the microcomputer in control room.

The Graphic User Interface (GUI) software installed on the microcomputer is developed by using Microsoft Visual Basic 6 as the programming language. It illustrates the virtual hardware connections including the hardware-status windows. These statuses can be recorded and reviewed later on.

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากสถานีดาวเทียมสิรินธร จังหวัดอุบลราชธานี เป็นสถานีดาวเทียมที่ใช้ติดต่อสื่อสารกับประเทศต่าง ๆในย่านแถบมหาสมุทรแปซิฟิกมีการติดต่อสื่อสารกัน 24 ชั่วโมง ดังนั้น ระบบไฟฟ้าที่จ่ายให้อุปกรณ์ดาวเทียมภายในสถานีดาวเทียมสิรินธรจะต้องดี มีเสถียรภาพสูง จึงจะสามารถทำให้ระบบการสื่อสารดำเนินไปด้วยดีตลอดเวลา สถานีดาวเทียมสิรินธรมีห้องควบคุม ที่ใช้เป็นจุดศูนย์กลางในการควบคุมและเฝ้าดูการทำงานของอุปกรณ์ดาวเทียมที่อยู่ตามห้อง ตามอาคารต่าง ๆ ภายในสถานีดาวเทียมสิรินธร หากอุปกรณ์ดาวเทียมทำงานผิดปกติระบบจะรายงานมายังคอมพิวเตอร์ที่ใช้ควบคุมอุปกรณ์เหล่านั้นที่ห้องควบคุม เพื่อให้เจ้าหน้าที่ช่างเทคนิคที่เฝ้าดูการทำงานของอุปกรณ์ต่าง ๆ ที่อยู่ในห้องควบคุม สามารถทราบได้ว่าอุปกรณ์ตัวใดเสียทั้งยังสามารถช่วยในการแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นได้อย่างรวดเร็ว ถูกต้อง แม่นยำ ส่วนระบบไฟฟ้ากำลังที่จ่ายให้อุปกรณ์ดาวเทียมประจำสถานีดาวเทียมสิรินธรนั้น ถึงแม้ว่าจะเป็นระบบอัตโนมัติ เช่น เมื่อกระแสไฟฟ้าจากการไฟฟ้าส่วนภูมิภาคเกิดดับหรือขัดข้อง ระบบ ATS (Automatic Transfer System) ก็จะสั่งให้ติดเครื่องยนต์เพื่อปั่นเครื่องกำเนิดไฟฟ้าอย่างอัตโนมัติและทำการจ่ายกระแสไฟฟ้าแทนการไฟฟ้าส่วนภูมิภาค แต่ระบบไฟฟ้าเหล่านี้ยังไม่มีระบบที่ใช้สำหรับเฝ้าตรวจสอบและบันทึกผลการทำงาน หากอุปกรณ์ไฟฟ้าเหล่านี้เกิดทำงานผิดปกติหรือระบบไฟฟ้าขัดข้อง อาจจะก่อให้เกิดความเสียหาย หรืออาจทำให้ เจ้าหน้าที่ผู้ปฏิบัติงานเสียเวลาในการตรวจสอบหาสาเหตุ และแก้ไขปัญหาต่าง ๆ ได้อย่างล่าช้า ดังนั้น จึงได้คิดพัฒนาระบบเฝ้าตรวจสอบและบันทึกการทำงานของอุปกรณ์ไฟฟ้ากำลัง (Main power monitoring and logging system) ขึ้นมา เพื่อแก้ปัญหาดังกล่าวข้างต้น

1.2 วัตถุประสงค์

1.2.1 เพื่อให้เจ้าหน้าผู้ปฏิบัติงานสามารถตรวจสอบสถานะการทำงานของระบบไฟฟ้ากำลังได้ทันท่วงทีและเฝ้าติดตามการทำงานของอุปกรณ์ได้ตลอดเวลา

1.2.2 เพื่อให้เจ้าหน้าที่ผู้ปฏิบัติงานสามารถบันทึกเหตุการณ์ที่เกิดขึ้นตามเวลาจริง เมื่อสถานะการทำงานของอุปกรณ์ไฟฟ้ากำลัง เกิดการเปลี่ยนแปลง และใช้เป็นข้อมูลในการอ้างอิง ตรวจสอบสถานะการทำงานของอุปกรณ์ไฟฟ้ากำลังได้

1.2.3 เพื่อช่วยเจ้าหน้าที่ผู้ปฏิบัติงานแก้ไขปัญหาได้ตรงจุด รวดเร็ว แม่นยำ หากมีอุปกรณ์ไฟฟ้ากำลัง หรือระบบไฟฟ้าขัดข้อง

1.2.4 เพื่อใช้เป็นข้อมูลในการบริหารจัดการระบบไฟฟ้ากำลังเช่น จัดทำรายงานข้อมูลด้านสถิติจำนวนครั้งที่ไฟฟ้าดับ หรือขัดข้องในแต่ละเดือน สามารถวิเคราะห์ช่วงเวลาที่ระบบไฟฟ้ากำลังขัดข้องได้

1.2.5 พื่อสามารถลดขั้นตอนการปฏิบัติงานและแบ่งเบาภาระให้กับของเจ้าหน้าที่โดยไม่จำเป็นจะต้องเฝ้าดูอุปกรณ์ไฟฟ้ากำลังจริงตลอดเวลา

1.2.6 เพื่อช่วยเพิ่มประสิทธิภาพการทำงานให้มีความน่าเชื่อถือและทันสมัยมากยิ่งขึ้น

1.3 ขอบเขตของการค้นคว้าอิสระ

การพัฒนาระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลังในครั้งนี้ จะเป็นการวิเคราะห์ออกแบบและพัฒนาเองทั้งหมดและนำไปใช้งานจริงที่สถานีดาวเทียมสิรินธร (บริษัท กสท. โทรคมนาคม จำกัด) อำเภอสิรินธร จังหวัดอุบลราชธานี โดยจะประกอบด้วย 2 ส่วน คือส่วนกล่องอินเตอร์เฟส และส่วนติดต่อกับผู้ใช้ (Graphic User Interface)

1.3.1 กล่องอินเตอร์เฟส (Interface Box) จะเป็นการวิเคราะห์ออกแบบและสร้างเองทั้งหมดโดย ฮาร์ดแวร์ จะใช้ไมโครคอนโทรลเลอร์ PIC16F84 ซึ่งเป็นไมโครคอนโทรลเลอร์ของบริษัท Microchip [6] ส่วนSOFTWARE นั้น จะใช้ภาษา ASSAMBLY ในการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ PIC16F84 [1] และใช้ CASE TOOL MP-LAB ของบริษัทMicrochip โดยในการพัฒนาส่วนการเชื่อมต่อสัญญาณระหว่างไมโครคอนโทรลเลอร์ที่อยู่ภายในอาคารไฟฟ้ากำลัง กับตัวไมโครคอมพิวเตอร์ที่อยู่ในห้องควบคุม (Control Room) ซึ่งมีระยะทางห่างกันนั้นจะใช้ IC MAX232N [7] แปลงสัญญาณ TTL ให้เป็นสัญญาณ RS-232C [4] เพื่อส่งสัญญาณได้ไกลขึ้นที่ความเร็ว 9600 bps. กล่องอินเตอร์เฟสนั้นจะติดตั้งที่ อาคารการไฟฟ้ากำลัง สถานีดาวเทียมสิรินธร

1.3.2 ส่วนติดต่อกับผู้ใช้ (Graphic User Interface) จะเป็นโปรแกรมที่เขียนขึ้นมาใหม่เพื่อใช้สำหรับอ่านข้อมูลอนุกรม (RS-232C) [3] ที่ส่งมาจากกล่องอินเตอร์เฟส ที่ใช้สำหรับตรวจสอบอุปกรณ์ไฟฟ้าที่อาคารไฟฟ้ากำลัง (Main power building) โปรแกรมส่วนติดต่อกับผู้ใช้จะทำการอ่านค่าสถานะของอุปกรณ์ไฟฟ้าเหล่านั้น ประมวลผลตีความหมาย จากนั้นโปรแกรม

จะแสดงสถานะจากการตีความหมายออกมาที่หน้าจอเสมือนเป็นอุปกรณ์ฟ้ากำลังจริงที่อยู่ในอาคารไฟฟ้ากำลัง โดยการเขียนภาพแสดง Block diagram ของอุปกรณ์ไฟฟ้ากำลัง ในส่วนโปรแกรมนั้นจะประกอบด้วย STATUS WINDOWS ใช้สำหรับดูสถานะของอุปกรณ์และ LOGGING WINDOWS ใช้สำหรับบันทึกเหตุการณ์ต่าง ๆ ที่เกิดขึ้น โดยจะใช้โปรแกรมภาษา Visual Basic 6 ในการพัฒนา และใช้โปรแกรม Microsoft Access 2003 ในการจัดการแฟ้มข้อมูล ส่วนติดต่อกับผู้ใช้ (Graphic User Interface) นั้นจะติดตั้งที่ห้องควบคุม สถานีดาวเทียมสิรินธร

1.4 ขั้นตอนในการดำเนินงาน

ตารางที่ 1 ระยะเวลาดำเนินงาน

การดำเนินงาน	สัปดาห์ที่								
	1	2	3	4	5	6	7	8	9
1. ศึกษา Interface RS232-C	←	→							
2. ศึกษา Microcontroller (PIC 16F84)	←	→							
3. ออกแบบระบบ		←	→						
4. สร้างกล่องอินเตอร์เฟส				←	→				
5. เขียนโปรแกรมควบคุม Microcontroller PIC16F84					←	→			
6. เขียนโปรแกรมติดต่อกับผู้ใช้งาน					←	→	→		
7. ทดสอบ						←	→	→	
8. แก้ไขถ้ามีข้อผิดพลาด								← →	
9. ส่งงาน									← →

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 หากระบบไฟฟ้ากำลังขัดข้อง จะได้รับสารสนเทศเกี่ยวกับปัญหา ที่เกิดขึ้นที่จุดใด ซึ่งเป็นการแก้ไขปัญหาได้ตรงจุดถูกต้อง รวดเร็ว และแม่นยำ

1.5.2 สามารถค้นหาประวัติการทำงานของอุปกรณ์ไฟฟ้ากำลังย้อนหลังได้

1.5.3 ช่วยในการบริหารจัดการระบบไฟฟ้ากำลัง ให้ได้รับความสะดวกมากยิ่งขึ้น

1.5.4 มีไฟฟ้าจ่ายกระแสให้กับอุปกรณ์ต่าง ๆ อย่างต่อเนื่องไม่ขาดช่วง

1.6 คำนิยามศัพท์เฉพาะ

1.6.1 UPS : Uninterruptible Power Supply หมายถึง เครื่องจ่ายไฟฟ้าสำรองไม่ขาดช่วง

1.6.2 MDB : Main Distribution Board หมายถึง ตู้จ่ายไฟฟ้าสำหรับไฟฟ้าจาก กฟภ. Load ที่ใช้ไฟฟ้าจากตรงนี้จะขาดช่วงเมื่อไฟฟ้าจาก กฟภ. ขัดข้อง

1.6.3 EMDB : Emergency Main Distribution Board หมายถึงตู้จ่ายไฟฟ้าสำหรับไฟฟ้าจากเครื่องกำเนิดไฟฟ้า Load ที่ใช้ไฟฟ้าจากตรงนี้จะขาดช่วงไม่เกิน 10 วินาที่ เมื่อไฟฟ้าดับ

1.6.4 ATS : Automatic Transfer System หมายถึง ระบบสำหรับทำหน้าที่ในการถ่ายโหลดไฟฟ้า เช่น ไฟฟ้าจาก กฟภ. ไปที่โหลด หรือ ไฟฟ้าจากเครื่องกำเนิดไฟฟ้าไปที่โหลดซึ่งที่ใช้ไฟฟ้าจาก ATA นี้จะเป็นโหลดแบบ EMDB ซึ่งหมายความว่า ไฟฟ้าจะขาดช่วงไม่เกิน 10 วินาที เมื่อไฟฟ้าจาก กฟภ. ดับ

1.6.5 ACB-MDB : Air Circuit Breaker – MDB หมายถึง ตัวป้องกันไฟฟ้า ของ MDBเมื่อเกิดการลัดวงจร ซึ่งมีขนาดใหญ่ และทดกระแสไฟฟ้าได้สูง ภายในจะมีห้องป้องกันการเกิดประกายไฟของหน้า Contract เมื่อตอนสัมผัสกัน ซึ่งภายในบรรจุก๊าซเฉื่อยไว้

1.6.6 ACB-MAIN : Air Circuit Breaker – MAIN หมายถึง ตัวป้องกันไฟฟ้า เมื่อเกิดการลัดวงจร และเป็น Switch ตัดต่อไฟฟ้าทางด้าน ไฟฟ้าจาก กฟภ. สำหรับการถ่ายโหลดในระบบ ATS ซึ่งมีขนาดใหญ่ทดกระแสไฟฟ้าได้สูง ภายในจะมีห้องป้องกันการเกิดประกายไฟของหน้าสัมผัส เมื่อตอนสัมผัสกัน ซึ่งภายในบรรจุก๊าซเฉื่อยไว้

1.6.7 ACB-GEN : Air Circuit Breaker – GEN หมายถึง อุปกรณ์ป้องกันไฟฟ้า เมื่อเกิดการลัดวงจร และเป็น Switch ตัดต่อไฟฟ้าทางด้าน ไฟฟ้าจาก เครื่องกำเนิดไฟฟ้า(Generator) สำหรับการถ่ายโหลดในระบบ ATS ซึ่งมีขนาดใหญ่ทดกระแสไฟฟ้าได้สูง ภายในจะมีห้องป้องกันการเกิดประกายไฟของหน้าสัมผัสt เมื่อตอนสัมผัสกัน ซึ่งภายในบรรจุก๊าซเฉื่อยไว้

1.6.8 YU หมายถึง ขดลวด Under Voltage ใน ACB (Air Circuit Breaker) ถ้าหากไม่มีไฟฟ้าไปเลี้ยงที่ขดลวด อุปกรณ์ ACB. จะไม่ทำงาน (Open circuit) หน้าสัมผัสจะไม่ต่อ

1.6.9 PEA: Provincial Electricity Authority หมายถึง อักษรย่อของ การไฟฟ้าส่วนภูมิภาค

1.6.10 UUV หมายถึง อุปกรณ์ตรวจสอบแรงดันไฟฟ้าจาก กฟภ. ว่าปกติหรือไม่ ซึ่งจะมีชุดตรวจสอบแรงดันด้านต่ำ (Under Voltage) และชุดตรวจสอบแรงดันด้านสูง (Over Voltage) ซึ่งด้านต่ำจะตรวจสอบ Voltage ต่ำกว่า 360 โวลท์ และ ด้านสูงจะตรวจสอบสูงกว่า 440 โวลท์ แรงดันไฟฟ้าจาก กฟภ. ค่าปกติจะมีค่าระหว่าง 360 โวลท์ ถึง 440 โวลท์ ในระบบ 3 เฟส

1.6.11 A1 BLD หมายถึง อาคารจานสายอากาศขนาด 32 เมตร ภายในมีอุปกรณ์ดาวเทียมเช่น เครื่องรับ-ส่ง สัญญาณดาวเทียม และระบบควบคุมจานสายอากาศ

1.6.12 A2 BLD หมายถึง อาคารจานสายอากาศ ขนาด 21 เมตร โดยภายในมีอุปกรณ์ดาวเทียม เช่น เครื่องรับ-ส่ง สัญญาณดาวเทียมและระบบควบคุมจานสายอากาศ

1.6.13 CONTROL BLD หมายถึง อาคารควบคุม ภายในมีอุปกรณ์ดาวเทียม

1.6.14 POWER BLD หมายถึง อาคารไฟฟ้ากำลัง ซึ่งมีอุปกรณ์ไฟฟ้ากำลังอยู่ด้านใน เช่น Generator, UPS, ATS, ตู้ควบคุมไฟฟ้า เป็นต้น

1.6.15 DB1-CB หมายถึง Circuit Breaker ของอาคารควบคุม ใช้กับอุปกรณ์ไฟฟ้าที่ไม่สำคัญโดยโหลดที่ต่อจากจุดนี้กระแสไฟฟ้าจะขาดช่วงเมื่อไฟฟ้าจาก กฟภ. ขัดข้อง

1.6.16 EDB1-CB หมายถึง Circuit Breaker ของอาคารควบคุม ใช้สำหรับกับอุปกรณ์ไฟฟ้าที่สำคัญไฟฟ้าจะขาดช่วงไม่เกิน 10 วินาที เมื่อไฟฟ้าจาก กฟภ. ขัดข้อง จะใช้ไฟฟ้าจากเครื่องกำเนิดไฟฟ้าแทน

1.6.17 EDB2-CB หมายถึง Circuit Breaker ของอาคารจานสายอากาศ 32 เมตร สำหรับอุปกรณ์ไฟฟ้าที่สำคัญไฟฟ้าจะขาดช่วงไม่เกิน 10 วินาที เมื่อไฟฟ้าจาก กฟภ. ขัดข้อง จะใช้ไฟฟ้าจากเครื่องกำเนิดไฟฟ้าแทน

1.6.18 EDB3-CB หมายถึง Circuit Breaker ของอาคารจานสายอากาศ 21 เมตร สำหรับอุปกรณ์ไฟฟ้าที่สำคัญไฟฟ้าจะขาดช่วงไม่เกิน 10 วินาที เมื่อไฟฟ้าจาก กฟภ. ขัดข้อง จะใช้ไฟฟ้าจากเครื่องกำเนิดไฟฟ้าแทน

1.6.19 UPS-CB หมายถึง Circuit Breaker สำหรับจ่ายกระแสไฟฟ้าให้กับ Charger เพื่อ Charge Battery ในระบบ UPS. ไฟฟ้าจะขาดช่วงไม่เกิน 10 วินาที เมื่อไฟฟ้าจาก กฟภ. ขัดข้อง และจะใช้ไฟฟ้าสำรองจากเครื่องกำเนิดไฟฟ้าแทน

1.6.20 VDU : Visual Display Unit หมายถึง ชุดคอมพิวเตอร์ที่ติดตั้งโปรแกรมติดต่อกับผู้ใช้งาน GRAPHIC USER INTERFACE แสดงสถานะการทำงานเสมือนมีอุปกรณ์ไฟฟ้ากำลังจริง

บทที่ 2
ความรู้พื้นฐานที่เกี่ยวข้อง

ระบบการเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง (Main power monitoring and logging system) ที่พัฒนาขึ้นนี้อาศัยทฤษฎี และหลักการที่เกี่ยวข้อง ดังนี้

2.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ PIC16F84

2.2 พื้นฐานการสื่อสารแบบอนุกรม

2.3 การเขียนโปรแกรมติดต่อและควบคุมพอร์ตอนุกรม ด้วย Visual Basic

2.4 พื้นฐานอิเล็กทรอนิกส์

2.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ PIC16F84

PIC16F84 เป็นไมโครคอนโทรลเลอร์ในตระกูล PIC (Peripheral Interface Controller) ของไมโครชิปเทคโนโลยี (Microchip Technology) ไมโครคอนโทรลเลอร์ในตระกูล PIC มีด้วยกันหลายเบอร์ แต่ละเบอร์ก็จะมีขีดความสามารถแตกต่างกันไป สำหรับในด้านการเรียนรู้เบื้องต้น เบอร์ที่มีความเหมาะสมมากที่สุดคือ PIC16F84 เนื่องจากภายใน PIC16F84 มีหน่วยความจำโปรแกรม (Program memory) เป็นแบบแฟลซ (Flash) ซึ่งเป็นหน่วยความจำที่สามารถเขียนและลบได้ด้วยสัญญาณไฟฟ้านับพันครั้ง เมื่อผู้เรียนหรือผู้ใช้งานต้องการพัฒนาโปรแกรมก็สามารถทำได้สะดวก

ไมโครคอนโทรลเลอร์ PIC16F84 เป็นไมโครคอนโทรลเลอร์สมัยใหม่ที่จัดอยู่ในกลุ่มของไมโครโปรเซสเซอร์แบบ RISC (Reduced Instruction Set Computer)
กล่าวคือไมโครคอนโทรลเลอร์ตระกูลนี้จะมีชุดคำสั่งน้อยมากเพียง 33 คำสั่งพื้นฐานเท่านั้น และทุกคำสั่งสามารถทำงานให้เสร็จได้ด้วยการใช้สัญญาณนาฬิกาเพียงลูกเดียว ทั้งยังทำงานในลักษณะไปป์ไลน์ (Pipe Line) เหมือนกับไมโครโปรเซสเซอร์ในสมัยใหม่ ความเร็วในการทำงานจึงสูงมาก เมื่อเทียบกับไมโครคอนโทรลเลอร์เบอร์อื่นที่ความถี่ของสัญญาณนาฬิกาเท่ากัน [1]

2.1.1 คุณสมบัติทางเทคนิคของ PIC16F84

สามารถแบ่งออกเป็น 3 ส่วน คือ หน่วยประมวลผลกลาง (Central Processing Unit CPU) ส่วนของเพอริเฟอรัล (Peripheral) และคุณสมบัติพิเศษอื่น ๆ

2.1.2 คุณสมบัติทางเทคนิคของหน่วยประมวลผลกลางภายใน PIC16F84 มีดังนี้

2.1.2.1 มีคำสั่งเพียง 33 คำสั่ง ขนาด 14 บิต

2.1.2.2 ทุกคำสั่งใช้เวลาในการประมวลผลเพียง 1 ไซเกิลของสัญญาณนาฬิกา หรือประมาณ 400 นาโนวินาทีที่สัญญาณนาฬิกาความถี่ 10 MHz ยกเว้นชุดคำสั่งกระโดดจะใช้เวลาประมาณ 2 ไซเกิลของสัญญาณนาฬิกา

2.1.2.3 ประมวลผลข้อมูลขนาด 8 บิต

2.1.2.4 มีรีจีสเตอร์ฟังก์ชั่นพิเศษ 15 ตัว

2.1.2.5 มีสแต็ก 8 ระดับ

2.1.2.6 มีโหมดการอ้างอิงแอดเดรส 3 โหมดคือ แบบโดยตรง (direct) แบบโดยอ้อม (indirect) และแบบสัมพัทธ์ (relative)

2.1.2.7 มีแหล่งกำเนิดอินเตอร์รัปต์ 4 แหล่ง ได้แก่

1) รับสัญญาณจากภายนอก โดยป้อนสัญญาณอินเตอร์รัปต์เข้าที่ขาอินพุต RB0/INT

2) จาก TMR0 ไทเมอร์โอเวอร์โฟลว์

3) เมื่อเกิดการเปลี่ยนแปลงที่พอร์ต B

4) เมื่อการเขียนข้อมูลลงในหน่วยความจำ EEPROM เสร็จสิ้นสมบูรณ์

2.1.2.8 หน่วยความจำข้อมูล (Data memory) เป็นแบบ EEPROM สามารถลบและเขียนใหม่ได้ประมาณล้านครั้งและเก็บข้อมูลได้นาน 40 ปี

2.1.2.9 ขนาดหน่วยความจำโปรแกรม เป็นแบบแฟลชมี ขนาด 1 กิโลไบท์ (1 ไบท์ ของ PIC16F84 มีขนาด 14 บิต) หน่วยความจำ EEPROM ภายใน 64 ไบท์ และหน่วยความจำแรม 68 ไบท์ ซึ่งใช้เป็นรีจิสเตอร์

2.1.3 คุณสมบัติทางเทคนิคของเพอริเพอริเฟอรัลใน PIC16F84 มีขาอินพุต-เอาต์พุต 13 ขา สามารถกำหนดเป็นขาอินพุตหรือเอาต์พุตได้อย่างอิสระ

2.1.3.1 กระแสซิงก์/ซอร์ส ของแต่ละขาอินพุตเอาต์พุตสูงพอที่จะขับ LED ได้

2.1.3.2 กระแสซิงก์ สูงสุด 25 mA. ต่อขา

2.1.3.3 กระแสซอร์ส สูงสุด 20 mA. ต่อขา

2.1.3.4 มีไทเมอร์/เคาน์เตอร์ ขนาด 8 บิต คือ TMR0 พร้อมกับ ปรีสเกลเลอร์ขนาด 8 บิต ที่สามารถโปรแกรมได้

2.1.4 สถาปัตยกรรมของ PIC16F84

2.1.4.1 ไมโครคอนโทรลเลอร์ PIC16F84 ได้รับการบรรจุหน่วยประมวลผล หน่วยความจำ หน่วยคำนวณทางคณิตศาสตร์และหน่วยอินพุต-เอาต์พุตไว้พร้อมสรรพ ทั้งยัง

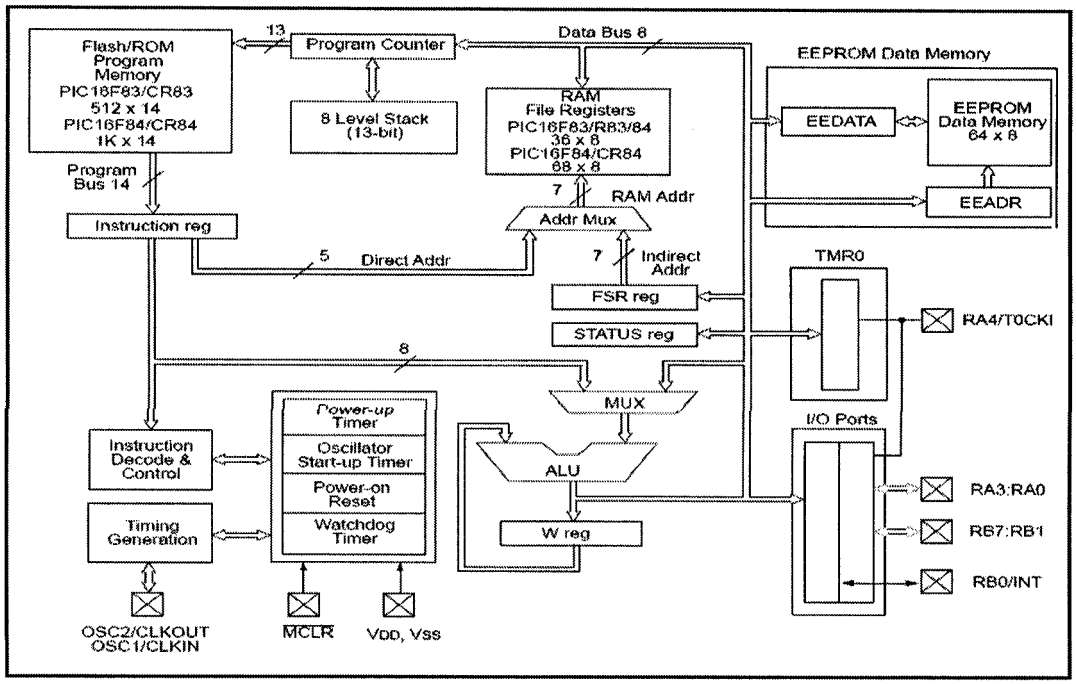
มีไทเมอร์และวอตซ์ด็อก ครบถ้วนสมบูรณ์ ดังแสดงในภาพที่ 1 แสดงสถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC16F84 ดังนี้

1) หน่วยความจำโปรแกรมมีโครงสร้างเป็นหน่วยความจำแบบแฟลซ ขนาด 1 กิโลไบท์ โดยใน 1 ไบท์ ของ PIC16F84 มีขนาด 14 บิต

2) หน่วยความจำข้อมูลจำกัดข้อมูลเป็นหน่วยความจำแบบอีอีพรอม ขนาด 64 ไบท์

3) หน่วยความจำแรมได้รับการกำหนดให้ทำงานเป็นรีจีสเตอร์กำหนดแฟ้มข้อมูลหรือรีจิสเตอร์ไฟล์ขนาด 68 ไบท์

4) การเข้าถึงหน่วยความจำทั้งหมดของหน่วยประมวลผลกลาง หรือซีพียูภายในไมโครคอนโทรลเลอร์นี้ สามารถทำได้ทั้งในลักษณะตรง โดยอ้อม และแบบสัมพัทธ์ โดยมีรีจีสเตอร์ FSR (File Select Register) ทำหน้าที่ในการควบคุมการเข้าถึงหน่วยความจำโดยอ้อม



ภาพที่ 1 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC16F84

2.1.4.2 เมื่อไมโครคอนโทรลเลอร์ทำงานตามคำสั่งที่กำหนดให้ข้อมูลของชุดคำสั่งจะถูกนำไปเก็บไว้ในรีจีสเตอร์คำสั่ง (Instruction register) จากนั้นจะถูกส่งต่อไปยังวงจรถอดรหัสเพื่อทำการควบคุมไทเมอร์ ทั้งหมดและจะถูกส่งต่อไปยังวงจรถอดรหัสเพื่อทำการควบคุมไทเมอร์ ทั้งหมดภายในตัวไมโครคอนโทรลเลอร์ นอกจากนั้นยังส่งไปควบคุมหน่วยคำนวณทางคณิตศาสตร์ โดยผ่านวงจรมัลติเพล็กซ์ด้วย

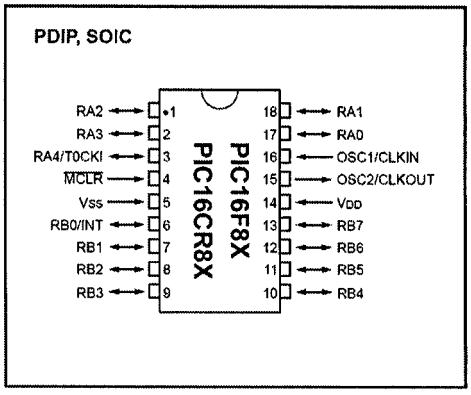
2.1.4.3 ใน PIC16F84 มี ไทเมอร์/เคาน์เตอร์ ขนาด 8 บิต 1 ตัวคือ TMR0 ภายในไทเมอร์/เคาน์เตอร์ ตัวนี้มี ปรีสเกลเลอร์ขนาด 8 บิต ที่สามารถโปรแกรมได้ โดยมีขาต่อใช้งาน 1 ขาคือ RA4/TOCKI

2.1.4.4 หน่วยคำทางคณิตศาสตร์ (Arithmetic Logic Unit: ALU) มีขนาด 8 บิต จะทำการคำนวณ บวก ลบ เลื่อนข้อมูลและประมวลผลทางลอจิก โดยใช้ฟังก์ชั่นบูลีนในการทำงาน ALU จะต้องมีรีจิสเตอร์ W ช่วย ซึ่งซีพียูจะไม่สามารถเข้าถึงรีจีสเตอร์ W นี้ได้โดยตรง เมื่อ ALU ทำงานจะมีผลต่อบิตทด (carry bit) บิตหลักทด (digit carry) และบิตศูนย์ (zero bit) ในรีจิสเตอร์ STATUS

2.1.4.5 ในส่วนของพอร์ตอินพุตใน PIC16F84 มีด้วยกัน 2 พอร์ต คือ พอร์ต A และพอร์ต B โดยในพอร์ต A มี 5 บิต คือ RA0 – RA4 สำหรับ RA4 ยังใช้เป็นขาอินพุตสำหรับรับสัญญาณนาฬิกาจากภายนอกให้แก่ TMR0 ด้วยส่วนพอร์ต B มี 8 บิตคือ RB0 – RB7 สำหรับขา RB0 ยังใช้เป็นขาอินพุตสำหรับสัญญาณอินเตอร์รัปต์ด้วย

2.1.4.6 การกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์ PIC16F84 เป็นหน้าที่ของส่วนกำเนิดจังหวะการทำงาน (Timing generation) ซึ่งต้องทำงานสัมพันธ์กับไทเมอร์ทั้ง 3 ตัว คือ เพาเวอร์อัปไทเมอร์ ออสซิลเลเตอร์สตาร์ทอัปไทเมอร์ และวอตช์ด็อกไทเมอร์ สำหรับPIC16F84 สามารถใช้สัญญาณนาฬิกาจากภายนอก โดยต่อสัญญาณเข้าที่ขา OSC1 และ OSC2 หรือจะใช้สัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ก็ได้

2.1.5 การจัดขาของ PIC16F84



ภาพที่ 2 การจัดขาของไมโครคอนโทรลเลอร์ PIC16F84

ไมโครคอนโทรลเลอร์ PIC16F84 บรรจุอยู่ในตัวถัง 2 แบบ คือ PDIP (Plastic Dual – In Line package) ซึ่งมีลักษณะเดียวกับไอซีแบบตีนตะขาบที่พบเห็นโดยทั่วไป และแบบ

SOIC เป็นตัวถังแบบที่ใช้ติดตั้งบนผิวหน้าของแผ่นวงจรพิมพ์ ตัวถังทั้งสองแบบของ PIC16F84 มีขาต่อใช้งานทั้งสิ้น 18 ขา ดังแสดงในภาพที่ 2 ซึ่งสามารถจัดขาต่อใช้งานของ PIC16F84 เป็น 4 กลุ่ม คือ

2.1.5.1 กลุ่มสัญญาณนาฬิกา มี 2 ขา คือ OSC1/CLKIN (ขา 16) และ OSC2/CLKOUT (ขา 15)

2.1.5.2 กลุ่มขาควบคุมมี 1 ขาคือ MCLR (ขา 4)

2.1.5.3 กลุ่มขาพอร์ตอินพุตเอาต์พุต มี 13 ขา แบ่งเป็นขาพอร์ต A 5 ขา ได้แก่ RA0-RA4 (ขา 17, 18, 1, 2 และ 3) และขาพอร์ต B ได้แก่ ขา RB0-RB7 (ขา 6 ถึง ขา 13)

2.1.5.4 กลุ่มขาไฟเลี้ยง มี 2 ขา คือ ขา VSS (ขา 5) หรือขาต่อกราวด์และขา VDD (ขา 14) หรือขาต่อไฟเลี้ยง โดยปกติใช้ +5 V สำหรับรายละเอียดโดยสรุปของขาต่อใช้งานทั้งหมดดังแสดงในตารางที่ 2

ตารางที่ 2 รายละเอียดของขาต่อใช้งานทั้งหมดของ PIC16F84

Pin Name	DIP No.	SOIC No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	I	ST/CMOS (3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
					PORTA is a bi-directional I/O port.
RA0	17	17	I/O	TTL	
RA1	18	18	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RA4/T0CKI	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	I/O	TTL/ST (1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	Interrupt on change pin.
RB5	11	11	I/O	TTL	Interrupt on change pin.
RB6	12	12	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming clock.
RB7	13	13	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming data.
Vss	5	5	P	—	Ground reference for logic and I/O pins.
VDD	14	14	P	—	Positive supply for logic and I/O pins.

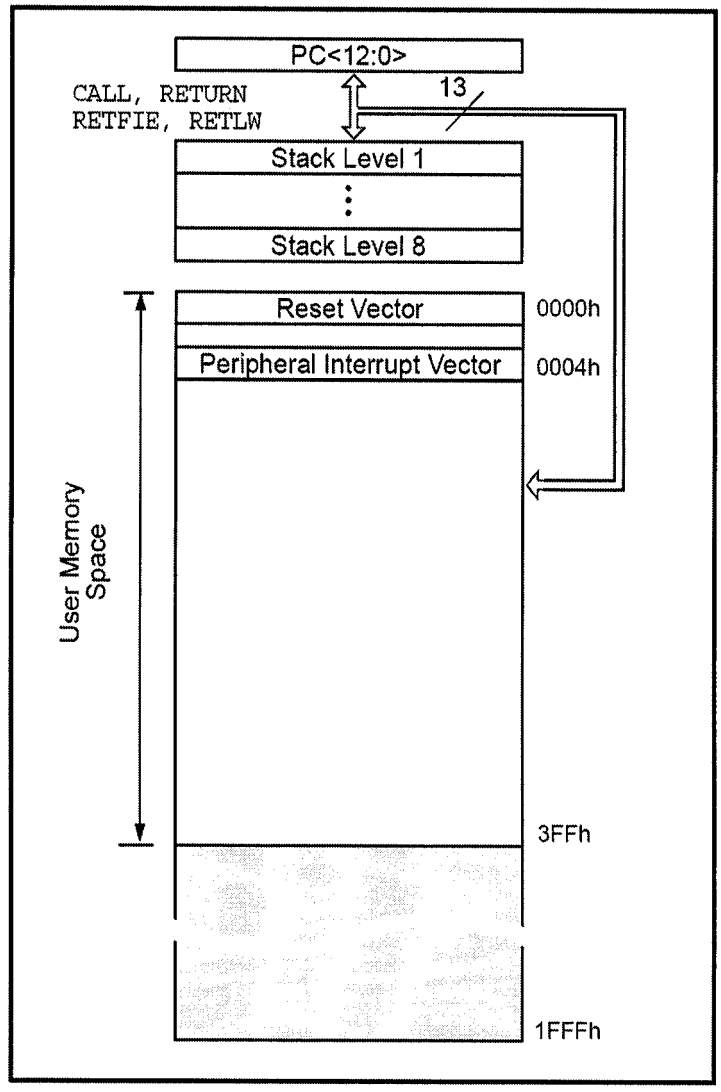
Legend: I= input O = output I/O = Input/Output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

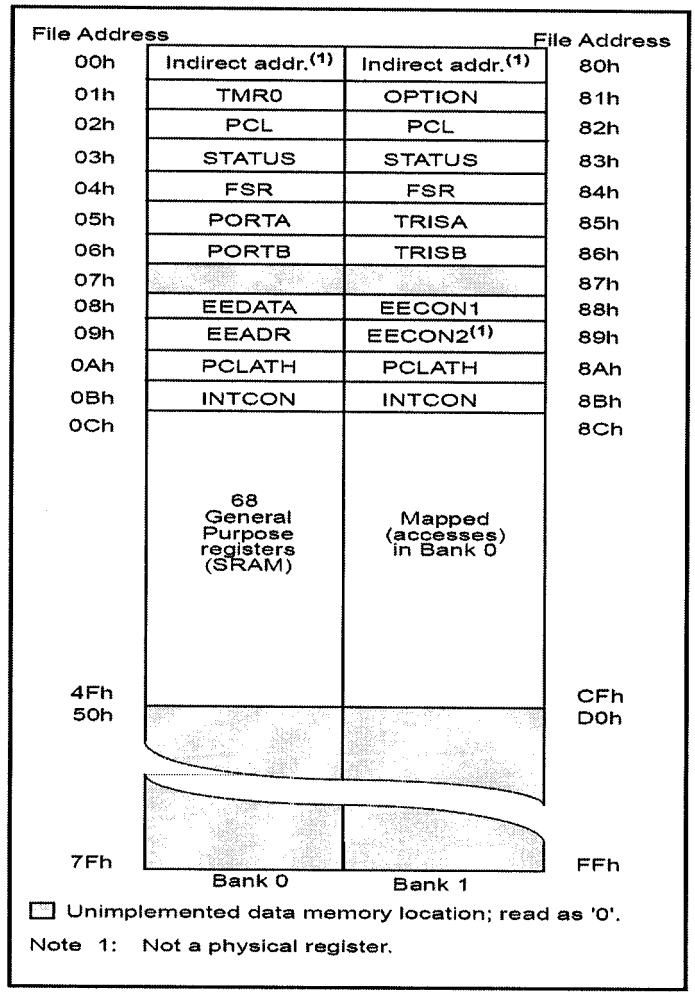
2.1.6 การจัดสรรหน่วยความจำใน PIC16F84

การจัดสรรหน่วยความจำภายใน PIC16F84 แบ่งออกเป็น 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

2.1.6.1 หน่วยความจำโปรแกรม เป็นหน่วยความจำที่ใช้ในการโปรแกรมควบคุมการทำงานของระบบ หรือใช้เป็นที่เก็บโปรแกรมมอนิเตอร์นั้นเอง หน่วยความจำโปรแกรมใน PIC16F84 เป็นหน่วยความจำแบบแฟลซ ในขณะที่ PIC16F84 ทำงานปกติ หน่วยความจำโปรแกรมนี้จะสามารถอ่านได้เพียงอย่างเดียวเท่านั้นและสามารถเขียนหรือแก้ไขได้ก็ต่อเมื่อ PIC16F84 อยู่ในโหมดของการโปรแกรมเท่านั้น



ภาพที่ 3 การจัดสรรหน่วยความจำโปรแกรมใน PIC16F84



ภาพที่ 4 การจัดสรรหน่วยความจำข้อมูลใน PIC16F84

หน่วยความจำโปรแกรมมีขนาด 1 กิโลไบท์ (จำนวนบิตต่อ 1 ไบท์ของหน่วยความจำโปรแกรมของ PIC16F84 นี้เท่ากับ 14 บิต) ได้รับการจัดสรรอยู่ในตำแหน่ง 0000H – 03FFH สามารถเขียนข้อมูลเก็บลงในพื้นที่ของหน่วยความจำนี้ได้ทุกแอดเดรสเว้นแต่แอดเดรส 0000H ซึ่งสงวนไว้สำหรับเก็บค่าเวกเตอร์ของการรีเซต หรือรีเซตเวกเตอร์ (Reset Vector) และแอดเดรส 0004H ซึ่งในการเก็บค่าเวกเตอร์ของการอินเตอร์รัปต์ หรืออินเตอร์รัปต์เวกเตอร์ (Interrupt Vector) ดังแสดงในภาพที่ 4 ซึ่งแสดงการจัดสรรพื้นที่ของหน่วยความจำโปรแกรมในไมโครคอนโทรลเลอร์ PIC16F84

2.1.6.2 หน่วยความจำข้อมูล พื้นที่ของหน่วยความจำข้อมูลได้รับการจัดสรรเป็น 2 ส่วน คือ พื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Registers: SFR) และพื้นที่ของรีจิสเตอร์ใช้งานทั่วไป (General Purpose Registers: GPR) จะจัดแบ่งเป็นแบงก์ (Bank)โดยในแต่ละแบงก์ของ SFR จะเป็นรีจีสเตอร์ที่ทำหน้าที่แตกต่างกันออกไป การติดต่อกับหน่วยความจำ

ข้อมูลในแต่ละแบงก์ จะต้องกำหนดค่าของบิตควบคุม RP0 และ RP1 ในรีจิสเตอร์ STATUS ดังแสดงในภาพที่ 5 เป็นการจัดสรรหน่วยความจำข้อมูลของ PIC16F84

จะเห็นได้ว่า หน่วยความจำข้อมูล จะแบ่งออกเป็น 2 แบงก์ เพื่อเก็บค่าของ SFR และ GPR หน่วยความจำในแบงก์ "0" จะถูกเลือกเมื่อทำการเคลียร์บิต RP0 (บิต 5 ของรีจิสเตอร์ Status) และถ้าหากบิตนี้เซตเป็น "1" จะเป็นการเลือกหน่วยความจำในแบงก์ 1 ในแต่ละแบงก์จะมีขนาด 128 ไบท์ (0x07F) และในทุก ๆ 12 ตำแหน่งแรกของแต่ละแบงก์จะถูกสำรองไว้สำหรับเก็บค่าของ SFR การเข้าถึงหน่วยความจำข้อมูลสามารถทำได้โดยตรง (Direct) โดยกำหนดแฟ้มข้อมูลที่เก็บรีจิสเตอร์นั้น ๆ หรือโดยอ้อม (indirect) โดยผ่านทางรีจีสเตอร์เลือกแฟ้มข้อมูล (File Select Register: FSR)

2.1.7 รีจีสเตอร์ควบคุมของ PIC16F84

ใน PIC16F84 มีรีจีสเตอร์ที่ใช้ในการควบคุมอยู่ 6 ตัว คือ STATUS, OPTION, INTCON, PCL, PCLATH และ W

ตารางที่ 3 รายละเอียดของรีจิสเตอร์ไฟล์ทั้งหมดของ PIC16F84

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.
Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.
2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.
3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

2.7.1.1 รีจิสเตอร์ STATUS

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
0 = Bank 0, 1 (00h - FFh)
1 = Bank 2, 3 (100h - 1FFh)
The IRP bit is not used by the PIC16F8X. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F8X. RP1 should be maintained clear.

bit 4: $\overline{TO}$: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3: $\overline{PD}$: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions) (For $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

ภาพที่ 5 รายละเอียดของบิตต่าง ๆ ของรีจีสเตอร์ STATUS

จากภาพที่ 5 ในรีจิสเตอร์ STATUS เป็นรีจีสเตอร์ที่ใช้แสดงสถานะทางคณิตศาสตร์ของ ALU. สถานะการทำงานของ PIC16F84 และใช้เป็นตัวกำหนดการเลือกแบงก์ของหน่วยความจำข้อมูล การเข้าถึงรีจีสเตอร์ STATUS เพื่ออ่านและเขียนข้อมูลก็สามารถกระทำได้ด้วยวิธีการเดียวกับการอ่านและเขียนรีจีสเตอร์ตัวอื่น ๆ และถ้าหากมีการกระทำคำสั่งเกี่ยวกับคณิตศาสตร์และลอจิก บิต Z, DC และ C จะเกิดการเปลี่ยนแปลงค่าตามผลการทำงานที่เกิดขึ้นโดยไม่คำนึงถึงค่าเดิมที่มีการเขียนหรือกำหนดมาก่อนหน้านี้

รีจีสเตอร์ STATUS มีแอดเดรสอยู่ที่ตำแหน่ง 0x003 และ 0x083 ในหน่วยความจำข้อมูลสำหรับความหมายและการกำหนดค่าในแต่ละบิตของรีจีสเตอร์ STATUS ดังแสดงในภาพที่ 5 อย่างไรก็ตามในบางคำสั่งที่จัดการข้อมูลระดับบิต เช่น btfss หรือ btfsc จะไม่มีผลต่อคำของรีจีสเตอร์ STATUS เช่นเดียวกับคำสั่ง bcf bsf swapf และ movwf ซึ่งเมื่อกระทำตามตัวอย่างดังกล่าวแล้ว บิตที่ใช้แสดงสถานะจะไม่เกิดการเปลี่ยนแปลงแต่อย่างใด

2.1.8 คำสั่งไมโครคอนโทรลเลอร์ PIC16F84

คำสั่งที่ใช้ในการทำงานไมโครคอนโทรลเลอร์ตระกูล PIC มีชุดคำสั่งหลักที่เหมือนกัน 35 คำสั่ง สำหรับ PIC16F84 จะมีเพิ่มเติมอีก 2 คำสั่งเป็น 37 คำสั่ง ตัวอย่างคำสั่งที่ใช้ในการพัฒนาระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลังมีดังนี้

2.1.8.1 คำสั่ง btfss (bit test register, skip if set)

เป็นคำสั่งกระโดดอย่างมีเงื่อนไขแบบหนึ่ง ที่จะทำการตรวจสอบบิตที่กำหนดในรีจีสเตอร์เสียก่อน หากตรงตามเงื่อนไขก็จะกระโดดข้ามแอดเดรสถัดไป 1 แอดเดรส แล้วจึงทำงานต่อที่แอดเดรสถัดจากนั้น เงื่อนไขการตรวจสอบคือบิตที่ตรวจนั้นเป็น “1” หรือไม่ ถ้าใช่ก็จะกระโดดไม่ทำงานในแอดเดรสถัดไป แต่ถ้าไม่ใช่นั้นคือค่าเป็น “0” ซีพียูก็จะทำงานในแอดเดรสต่อไปตามปกติ การทำงานในกรณีหลังนี้จะให้ผลการทำงานเหมือนกับคำสั่ง nop

สำหรับจำนวนไซเกิลที่ใช้ หากตรวจสอบตรงตามเงื่อนไขจะใช้ 2 ไซเกิล แต่ถ้าไม่ตรงจะใช้เพียงไซเกิลเดียว คำสั่ง btfss มีลักษณะการตรวจสอบเงื่อนไขของการกระโดดที่ตรงข้ามกับคำสั่ง btfsc

2.1.8.2 คำสั่ง goto (go to address)

เป็นคำสั่งที่กำหนดให้ซีพียูกระโดดไปทำงานยังแอดเดรสที่กำหนดให้ในหน่วยความจำโปรแกรมโดยไม่มีเงื่อนไขค่าที่กำหนดให้กระโดดไปนี้ต้องไม่เกิน 2,047

2.1.8.3 คำสั่ง call (subroutine call)

เป็นคำสั่งที่กำหนดให้ซีพียูกระโดดไปทำงานที่โปรแกรมย่อย โดยไม่มีเงื่อนไขและจะกลับมายังโปรแกรมหลักก็ต่อเมื่อพบคำสั่ง return หรือ retlw เมื่อกระทำคำสั่งนี้ซีพียูจะทำการเก็บค่า PC ไว้ในสแต็กโดยอัตโนมัติ หลังจากซีพียูทำงานในโปรแกรมย่อยเสร็จ ก็จะกลับมาอ่านค่า PC จากสแต็ก แล้วกลับไปทำงานต่ออย่างถูกต้อง

2.1.8.4 คำสั่ง return (return from subroutine)

เป็นคำสั่งที่แจ้งให้ซีพียูทราบว่าสิ้นสุดการทำงานของโปรแกรมย่อยแล้ว เมื่อซีพียูกระทำคำสั่งแล้ว จะไปอ่านค่า PC จากสแต็กที่นำไปเก็บไว้ชั่วคราวก่อนหน้าที่กระทำคำสั่ง call จากนั้นกระโดดไปยังแอดเดรสที่กำหนดโดย PC ในการเขียนโปรแกรมย่อยทุกครั้งต้องมีคำสั่ง return หรือ retlw ต่อท้ายเสมอ

2.1.8.5 คำสั่ง bcf (clear bit in register)

เป็นคำสั่งเคลียร์บิตในรีจีสเตอร์ไฟล์ที่กำหนดให้เป็น "0"

2.1.8.6 คำสั่ง bsf (set bit in register)

เป็นคำสั่งเซตบิตในรีจีสเตอร์ไฟล์ที่กำหนดให้เป็น"1"

2.1.8.7 คำสั่ง movwf (move data from W register to file register)

เป็นคำสั่งที่ให้ผลตรงข้ามกับ wovf นั้นคือ เป็นคำสั่งโอนย้ายข้อมูลจากรีจิสเตอร์ W มาเก็บไว้ในรีจีสเตอร์ไฟล์ที่กำหนด

2.1.8.8 คำสั่ง movlw (move data to W register)

เป็นคำสั่งที่ใช้ในการกำหนดค่าคงที่ 8 บิตของในรีจิสเตอร์ W ของเซตของค่าคงที่ที่ใช้ได้ระหว่าง 0-255

2.1.8.9 คำสั่ง decfsz : decrement register , skip if zero

เป็นคำสั่งกระโดดอย่างมีเงื่อนไขแบบหนึ่ง ที่จะทำการลดค่าในรีจีสเตอร์ลงหนึ่งแล้วตรวจสอบว่าเป็น "0" หรือไม่ หากตรงตามเงื่อนไขก็จะกระโดดข้ามไปในแอดเดรสถัดไป 1 แอดเดรสแล้วจึงสามารถทำงานต่อที่แอดเดรสถัดจากนั้น ค่าของรีจิสเตอร์ที่ลดค่าลงแล้วจะถูกเก็บไว้ในรีจีสเตอร์ W หรือรีจีสเตอร์ตัวเดิมก็ได้ ทั้งนี้ขึ้นอยู่กับการกำหนดข้อมูลที่ต่อจากรีจีสเตอร์

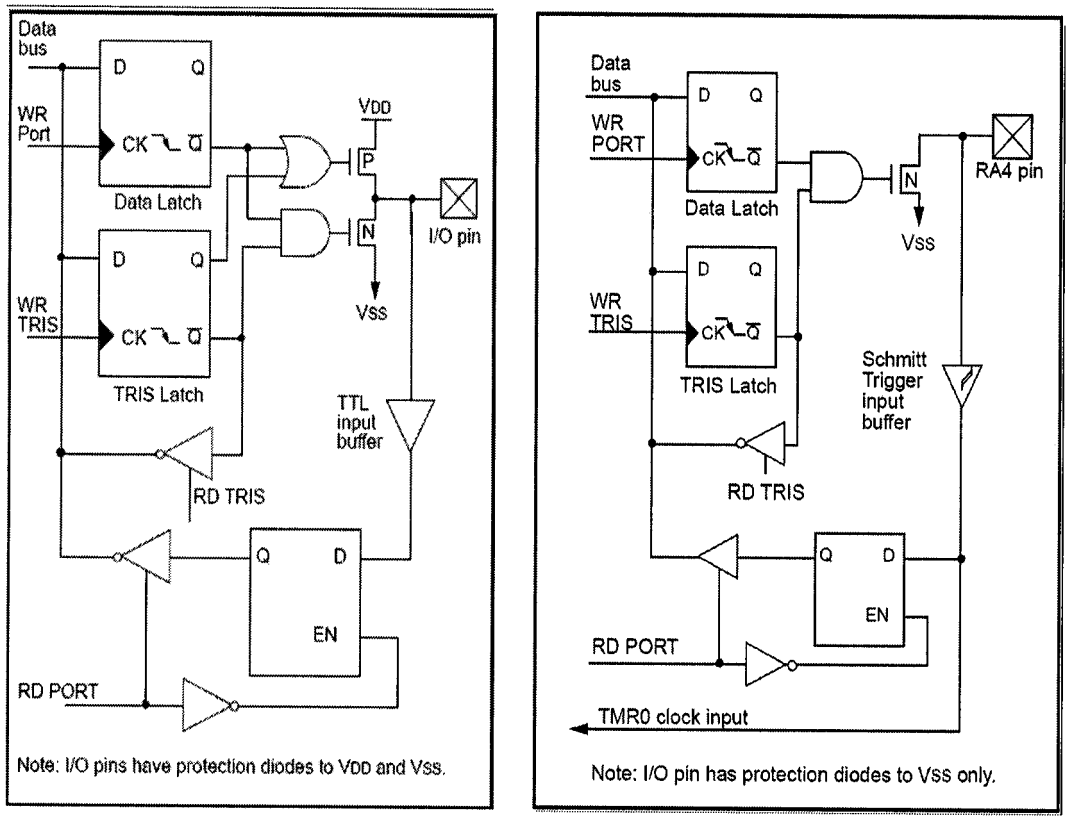
สำหรับจำนวนไซเกิลที่ใช้ หากตรวจสอบตรงตามเงื่อนไขจะใช้ 2 ไซเกิล แต่หากไม่ตรงตามเงื่อนไขจะใช้เพียงไซเกิลเดียว

2.1.9 พอร์ตของ PIC16F84

ไมโครคอนโทรลเลอร์ PIC16F84 มีพอร์ตสำหรับติดต่อกับอุปกรณ์ภายนอก 2 พอร์ตคือ พอร์ต A และพอร์ต B โดยพอร์ต A มีด้วยกัน 5 บิต คือ RA0-RA4 ในขณะที่พอร์ต B มี 8 บิต คือ RB0-RB7

2.1.9.1 พอร์ต A (โครงสร้างทางฮาร์ดแวร์) ดังแสดงใน ภาพที่ 6 วงจรภายในพอร์ต A 4 บิตแรก คือ RA0-RA3 จะเห็นได้ว่ามี ฟลิปฟลอปเป็นส่วนประกอบหลัก ทำให้ RA0 RA3 สามารถแลตซ์ข้อมูลได้พร้อมกันนั้นยังมีไว้คอยป้องกันแรงดันย้อนกลับทั้งจากไฟเลี้ยงบวกและ ลบด้วย ทุกขาของพอร์ต A สามารถรับสัญญาณอินพุทในระดับทีทีแอล (0-5 โวลต์) เมื่อ

ทำงานเป็นอินพุต และสามารถขับอุปกรณ์ซีมอสทางเอาต์พุตได้เมื่อกำหนดให้ทำงานเป็นขาเอาต์พุต



ภาพที่ 6 วงจรภายพอร์ต A 4 บิตแรก คือ RA0-RA3

รีจิสเตอร์ที่เกี่ยวข้องกับพอร์ต A การทำงานของพอร์ต A จะมีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว คือ รีจิสเตอร์สำหรับเก็บข้อมูลของพอร์ต A หรือ PORTA และรีจิสเตอร์กำหนดทิศทางการถ่ายทอดข้อมูลพอร์ต A หรือ TRISA โดยรีจิสเตอร์ PORTA มีขนาด 8 บิต แต่ใช้เพียง 5 บิตโดย 3 บิตบน คือ PORTA5-PORTA7 จะต้องกำหนดให้เป็น “0” ทั้งหมดมีแอดเดรสอยู่ที่ 05 H รีจิสเตอร์ PORTA บิต 0 ใช้เก็บข้อมูลของพอร์ต A บิต 0 หรือ RA0 ไล่เรียงตามลำดับ จนถึงรีจิสเตอร์ PORTA บิต 4 หรือ PORTA4 ใช้เก็บข้อมูลของพอร์ต A บิต 4 หรือ RA4 สำหรับรีจิสเตอร์ TRISA มีแอดเดรสอยู่ที่ 85H ใช้กำหนดทิศทางการถ่ายทอดข้อมูลให้ขาสัญญาณของพอร์ต A เป็นขาอินพุตหรือเอาต์พุต ถ้าต้องการให้เป็นเป็นอินพุตต้องเขียนข้อมูล “1” ไปยังบิต TRISA

ตารางที่ 4 สรุปการทำงานของพอร์ต A

Name	Bit0	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0. Output is open drain type.

Legend: TTL = TTL input, ST = Schmitt Trigger input

ตารางที่ 5 สรุปการกำหนดค่าของรีจิสเตอร์ที่เกี่ยวข้องกับพอร์ต A ทั้งหมด

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are unimplemented, read as '0'

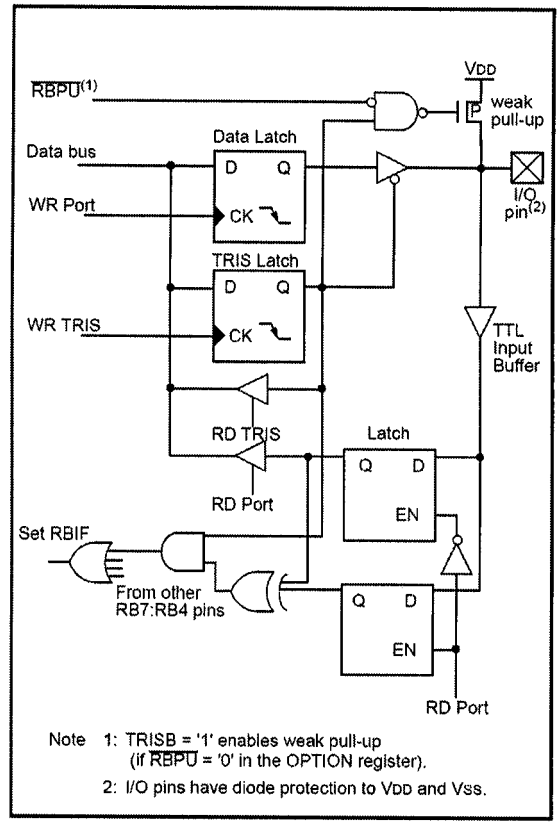
จากตารางที่ 5 แสดงการกำหนดค่าของรีจิสเตอร์ที่เกี่ยวข้อง โดยในการกำหนดพอร์ต A ให้เป็นอินพุต และถ้าหากต้องการกำหนดให้เป็นเอาต์พุตให้เขียนข้อมูล “0” ไปยังบิตนั้นโดย TRISA บิต 0 (TRISA0) ใช้กำหนดการทำงานของ RA0 ไล่เรียงตามลำดับจนถึง TRISA4 ใช้กำหนดการทำงานของ RA4 สำหรับ 3 บิตบนของ TRISA ไม่มีการใช้งานเช่นเดียวกับรีจิสเตอร์ PORTA เมื่อทำการอ่านจะได้ค่าเป็น “0” ทั้งหมด

ตัวอย่าง การกำหนดคำเริ่มต้นของพอร์ต A ในโปรแกรมที่ 1 เป็นโปรแกรมย่อยตัวอย่างที่เขียนขึ้นเพื่อกำหนดค่าเริ่มต้นหรือ อินิเชียล (Initializing) พอร์ต A โดยต้องกำหนดให้ RA0-RA3 เป็นอินพุต และ RA4 เป็นเอาต์พุต

โปรแกรมอินิเชียล พอร์ต A

```
clrf    PORTA       ; เคลียร์ค่าของรีจีสเตอร์พอร์ต A
bst     STATUS, RPO ; เลือกรีจีสเตอร์แบงก์ 1
movlw   0x0F        ; กำหนดค่าเพื่อเตรียมกำหนดทิศทางการ
                    ; ถ่ายทอดข้อมูลของพอร์ต A
movwf   TRISA       ; เขียนคำสั่ง 0FH ไปยังรีจีสเตอร์ TRISA เป็น
                    ; การกำหนดให้ RA3-RA0 เป็นอินพุต และ
                    ; RA4 เป็นเอาต์พุต ส่วนบิต TRISA 7-TRISA5
                    ; กำหนดให้เป็น "0"ทั้งหมด
```

2.1.9.2 พอร์ต B (โครงสร้างทางฮาร์ตแวร์) ในภาพที่ 7 และ 8 แสดงวงจรภายในของพอร์ต B โดยแบ่งเป็น 2 ส่วน คือ ในภาพที่ 7 เป็นวงจรภายในของพอร์ต B บิต 7 – บิต4 หรือ RB7-RB4 ส่วนในภาพที่ 8 เป็นวงจรภายในของ RB3-RB0



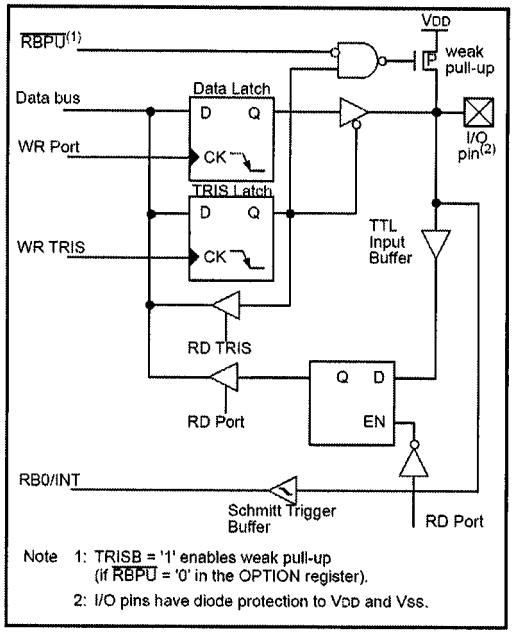
ภาพที่ 7 วงจรภายในของพอร์ต B บิต7 – บิต4 (RB7-RB4)

ตารางที่ 6 สรุปการทำงานของพอร์ต B

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST(2)	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST(2)	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger.
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.

จากตารางที่ 6 แสดงการทำงานของพอร์ต B ทุกสัญญาณของพอร์ต B จะมีการพูลอัป (ทำให้มีสภาวะเป็น High) ด้วยการต่อตัวต้านทานค่าน้อย ๆ ได้ เมื่อกำหนดที่ขาพอร์ต B เป็นอินพุต โดยต้องมีการเคลียร์บิต RBPU ในรีจิสเตอร์ OPTION เมื่อแจ้งให้ทราบว่าขณะนี้พอร์ต B ถูกกำหนดให้เป็นอินพุตแล้ว และเมื่อกำหนดให้พอร์ต B เป็นเอาต์พุต การพูลอัปจะถูกยกเลิกโดยอัตโนมัตินอกจากนั้นการพูลอัปจะถูกยกเลิกเมื่อเกิดเพาเวอร์ ออนรีเซต



ภาพที่ 8 วงจรภายในของพอร์ต B บิต 3-บิต 0 (RB3-RB0)

ส่วนขา RB0 – RB3 เป็นขาสัญญาณอินพุตเอาต์พุต ซึ่งได้รับการกำหนดทิศทางรีจิสเตอร์ TRISB ดังแสดงในตารางที่ 7 เป็นตารางสรุปการทำงานของพอร์ต B

รีจิสเตอร์ที่เกี่ยวข้องกับ พอร์ต B เช่นเดียวกับพอร์ต A พอร์ต B จะมีรีจิสเตอร์ที่เกี่ยวข้อง 3 ตัว คือ PORTB, TRISB และ OPTION โดยรีจิสเตอร์ PORTB มีแอดเดรสอยู่ที่ 06H , TRISB มีแอดเดรสที่ 86H และ OPTION มีแอดเดรสที่ 81H การกำหนดลักษณะการทำงานของรีจิสเตอร์เป็นไปในลักษณะเดียวกับรีจิสเตอร์ PORTA

โปรแกรมอินิเชียล พอร์ต B

```
clrf PORTB              ; เคียร์ค่าของรีจีสเตอร์พอร์ต B
bsf STATUS,RP0          ; เลือกรีจีสเตอร์แบงก์ 1
movlw 0xcf              ; กำหนดค่าเพื่อกำหนดทิศทางการทำงานของ
                        ; พอร์ต B
movwf TRISB             ; เขียนค่า 0xCF ไปยัง TRISB เพื่อกำหนดให้
                        ; RB3-RB0 เป็นอินพุตและ RB4, RB5 เป็น
                        ; เอาต์พุต ส่วน RB6 กับ RB7 เป็นอินพุต
```

ตารางที่ 7 สรุปการกำหนดค่าของรีจิสเตอร์ที่เกี่ยวข้องกับพอร์ต B ทั้งหมด

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

จากตารางที่ 7 เป็นการสรุปรายละเอียดของรีจิสเตอร์ที่เกี่ยวข้องกับพอร์ต B และในโปรแกรมย่อยตัวอย่างข้างบนเป็นการอินิเชียล พอร์ต B โดยกำหนดให้ RB0 –RA3, RB6 และ RB7 เป็นอินพุต ในขณะที่ RB4 และ RB5 เป็นเอาต์พุต

พอร์ตอินพุตและเอาต์พุตของไมโครคอนโทรลเลอร์เป็นส่วนที่มีความสำคัญมาก เพราะจะเป็นส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอก เพื่อทำการรับสัญญาณเมื่อเป็นอินพุต และใช้ส่งสัญญาณออกไปควบคุมการทำงานเมื่อทำงานเป็นเอาต์พุต ถ้าจะเปรียบเทียบกับมนุษย์ พอร์ตอินพุตเอาต์พุตก็เหมือนกับแขน ขา ตา จมูก ปาก นั้นเอง ดังนั้นการเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์จำเป็นอย่างยิ่งที่ต้องเข้าใจเรื่องของการควบคุมและใช้งานพอร์ตอินพุตเอาต์พุต

2.1.10 การเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ PIC16F84

สิ่งที่จำเป็นต้องรู้อันดับแรกก่อนการเขียนโปรแกรม คือไมโครคอนโทรลเลอร์ PIC และไมโครคอนโทรลเลอร์ส่วนใหญ่ จะทำงานตามคำสั่งที่เก็บอยู่ในหน่วยความจำโปรแกรมเรียงลำดับไล่กันไปเรื่อย ๆ ดังนั้น เมื่อต้องการนำไมโครคอนโทรลเลอร์ไปควบคุมระบบจะต้องคำนึงถึงธรรมชาติของการทำคำสั่งของโปรแกรมที่จะต้องไล่ตามลำดับไปเรื่อย ๆ สำหรับผู้ออกแบบโปรแกรมที่ดีควรเริ่มต้นการเขียนโปรแกรมด้วยการเขียนผังงาน เพราะจะช่วยให้ผู้เขียนโปรแกรมไม่สับสน สามารถเขียนโปรแกรมได้อย่างเป็นระบบ ทำความเข้าใจและตรวจสอบได้ง่าย

การเขียนโปรแกรมสำหรับข้อมูลที่เกิดขึ้นซ้ำ ๆ นั้นสามารถเขียนโดยวิธีการต่อข้อมูลออกไปเรื่อย ๆ แต่บางทีวิธีนี้ก็ไม่ถูกต้องนัก เนื่องจากหน่วยความจำภายในที่ไมโครคอนโทรลเลอร์มีอยู่นั้น มีขนาดไม่มากนัก หากบรรจุข้อมูลของโปรแกรมที่ซ้ำ ๆ ดูจะเป็นการสิ้นเปลือง นอกจากนั้นยังทำให้การเขียนโปรแกรมดูซับซ้อน

ดังนั้น การใช้วิธีการวนลูปจึงเป็นเทคนิคที่มีความสำคัญและมักจะพบมากในระบบควบคุมที่ใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมหลัก ทั้งนี้เนื่องจากในไมโครคอนโทรลเลอร์มีคำสั่งสำหรับการกระโดดไปทำงานจุดต่าง ๆ ได้ การวนลูปให้ผลลัพธ์ดังนี้

2.1.10.1 ถ้ามีการวนลูป โดยไม่มีการกำหนดเงื่อนไขให้หยุดโปรแกรมจะทำงานวนไปเรื่อย ๆ จนกว่าจะหยุดจ่ายไฟให้กับมัน

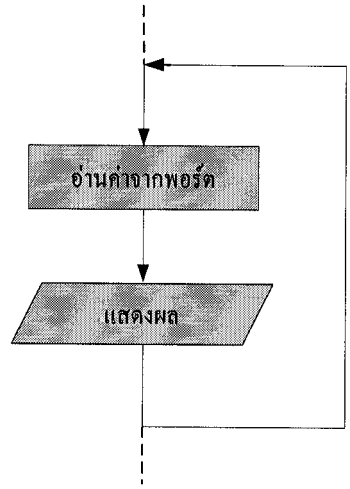
2.1.10.2 ถ้ามีการวนลูปโดยใช้เคาน์เตอร์ เป็นตัวกำหนดจำนวนรอบของการทำงาน โปรแกรมจะทำงานเท่ากับจำนวนที่กำหนดไว้ในเคาน์เตอร์เท่านั้น เมื่อค่าในเคาน์เตอร์มีค่าเท่ากับค่าที่กำหนดโปรแกรมก็จะออกจากการวนรอบ

2.1.10.3 ถ้าการวนลูปมีการตรวจจับค่าอินพุต โปรแกรมจะออกจากการวนลูปก็ต่อเมื่อมีการป้อนค่าอินพุตเข้าสู่ระบบ

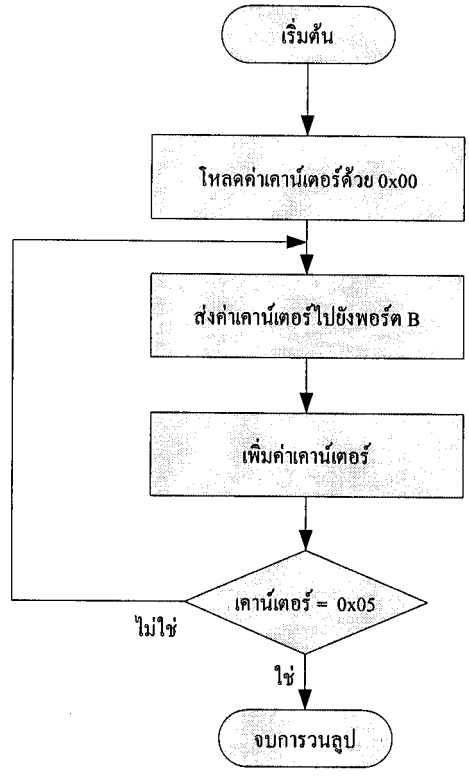
ผังงานง่าย ๆ ของการวนรอบ ดังแสดงในภาพที่ 9 โดยโปรแกรมจะเริ่มทำการอ่านค่า แล้วนำข้อมูลที่อ่านได้นี้ ไปแสดงผลออกสู่พอร์ตเอาต์พุต แล้วจึงวนกลับมาอ่านค่าอีกครั้ง

สำหรับผังงานของโปรแกรมลูปที่ใช้เคาน์เตอร์ แสดงในภาพที่ 10 ค่าของเวลาที่ใช้ในการวนลูปกำหนดได้ที่ตัวแปร N

เทคนิคอีกอย่างหนึ่งที่ช่วยให้ขนาดของโปรแกรมเล็กลง คือ การใช้โปรแกรมย่อย หรือ Subroutine ในบางครั้งมีการกระทำกลุ่มคำสั่งใด ๆ มากกว่าหนึ่งครั้ง ผู้เขียนโปรแกรมควรจะเขียนกลุ่มคำสั่งเหล่านั้นเพียงครั้งเดียว แล้วเก็บกลุ่มคำสั่งนั้นไว้ที่ใดที่หนึ่ง เมื่อต้องการใช้งานกลุ่มคำสั่งนี้ ก็ให้เรียกใช้ด้วยคำสั่ง CALL กลุ่มคำสั่งนี้ก็คือโปรแกรมย่อยนั้นเอง



ภาพที่ 9 ผังงานของโปรแกรมลูปอย่างง่าย



ภาพที่ 10 ผังงานของโปรแกรมลูปที่ใช้เคาน์เตอร์

2.1.11 การแอสเซมเบลอร์

การแอสเซมเบลอร์[1]ในภาษาแอสเซมบลี หมายถึง การแปลความหมายจากไฟล์เซอร์สโค้ดที่เขียนโดยภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง โปรแกรมภาษาแอสเซมบลีที่เขียนขึ้นจะขอเรียกว่า ไฟล์ซอร์สโค้ด เมื่อเขียนขึ้นด้วยเท็กซ์อิดิเตอร์ตัวใดตัวหนึ่งเรียบร้อย เช่น Notepad หรือ Microsoft Word แล้วจะต้องบันทึกลงในไฟล์ที่มีนามสกุลเป็น .ASM ทั้งนี้ เพราะโปรแกรมแอสเซมเบลอร์ จะแอสเซมเบลอร์ไฟล์ที่มีนามสกุลเป็น .ASM เท่านั้น โดยที่โปรแกรมที่ใช้ในการแอสเซมเบลอร์ เช่น MPASM ซึ่งได้อธิบายการใช้โปรแกรม MPASM ในภาคผนวก ง

หลังจากที่ทำการแอสเซมเบลอร์ โปรแกรมเรียบร้อยแล้ว โปรแกรมแอสเซมเบลอร์จะสร้างไฟล์ขึ้นมาอย่างน้อย 2 ไฟล์จากซอร์สโค้ดที่ผ่านการแอสเซมเบลอร์ โดยที่มีชื่อไฟล์เหมือนกับซอร์สโค้ดแต่จะแตกต่างกันที่มีนามสกุล ซึ่งจะเป็นไฟล์นามสกุล .LST และ .HEX สำหรับไฟล์นามสกุล .LST นั้น จะบรรจุรายละเอียดของโปรแกรม ทั้งนีโมนิก และออปโค้ดซึ่งเป็นเลขฐานสิบหก ตลอดจนข้อความที่ใช้อธิบายการทำงานของโปรแกรม ส่วนไฟล์นามสกุล .HEX นั้น จะเป็นข้อมูลเลขฐานสิบหกที่ใช้สำหรับเขียนลงในหน่วยความจำโปรแกรมบนไมโครคอนโทรลเลอร์ PIC16F84 ซึ่งในภาคผนวก จ ได้อธิบายการใช้โปรแกรม ICPROG ในการโปรแกรมไฟล์นามสกุล .HEX ลงไมโครคอนโทรลเลอร์

2.1.12 การเขียนโปรแกรมหน่วงเวลาโดยใช้รีจีสเตอร์

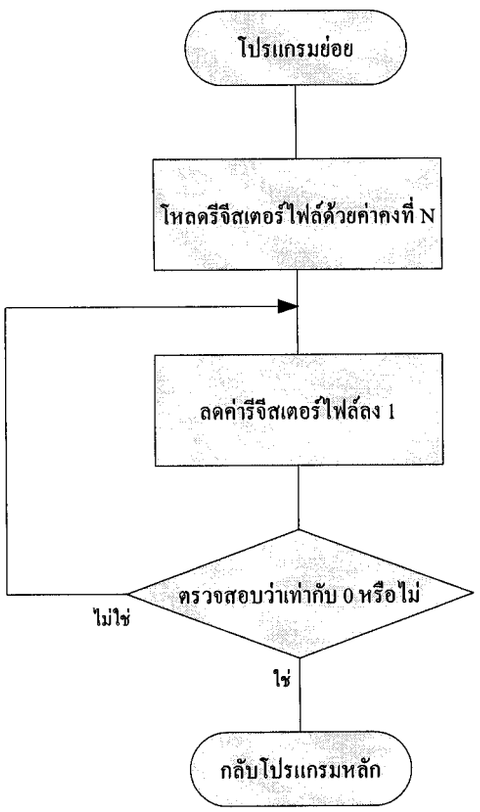
การหน่วงเวลาโดยใช้ซอฟต์แวร์นั้น จะใช้วิธีการวนลูปเพื่อนับสัญญาณนาฬิกาจนกระทั้งครบจำนวนที่ตั้งไว้ โดยความเที่ยงตรงของช่วงเวลานั้นจะขึ้นอยู่กับความเที่ยงตรงของสัญญาณนาฬิกาของ PIC16F84 จำเป็นอย่างยิ่งที่จะต้องใช้วงจรกำเนิดสัญญาณนาฬิกาแบบ คริสตอล ในกรณีที่ต้องการให้ค่าการหน่วงเวลามีความเที่ยงตรงสูง

สมมุติว่าถ้าปกติใช้วงจรกำเนิดสัญญาณนาฬิกาแบบคริสตอลความถี่ 4 เมกะเฮิรตซ์ค่าความถี่ของสัญญาณนาฬิกาภายในจะถูกหารด้วย 4 ดังนั้นสัญญาณนาฬิกาภายใน PIC16F84 เท่ากับ 1 เมกะเฮิรตซ์ หรือกำหนดเป็นช่วงเวลาได้เท่ากับ 1 ไมโครวินาที

คำสั่งส่วนใหญ่ของ PIC16F84 จะใช้สัญญาณนาฬิกาเพียง 1 ไซเกิล แต่จะมีคำสั่งบางคำสั่งที่จะต้องอ่านค่าโปรแกรมเคาน์เตอร์มาเก็บไว้ จำเป็นต้องใช้สัญญาณนาฬิกา 2 ไซเกิลดังแสดงในตารางที่ 8 จะแสดงให้เห็นถึงคำสั่งที่มักใช้กับการเขียนโปรแกรมหน่วงเวลาและจำนวนสัญญาณนาฬิกาที่ใช้ และภาพที่ 11 เป็นผังงานของโปรแกรมหน่วงเวลา

ตารางที่ 8 คำสั่งที่ใช้ในการเขียนโปรแกรมหน่วงเวลา

คำสั่ง	จำนวนสัญญาณนาฬิกา
movlw	1
movwf	1
decfsz	1, 2 ถ้าผลลัพธ์เป็น 0
goto	2
call	2
return	2



ภาพที่ 11 ผังงานของโปรแกรมหน่วงเวลา

จากผังงานของโปรแกรมหน่วงเวลาโดยใช้รีจีสเตอร์[1] ดังแสดงในภาพที่ 11 สามารถเขียนเป็นโปรแกรมได้ ดังนี้

จำนวนสัญญาณนาฬิกา

1			movlw	N	; load counter
1			movwf	COUNT	
(1xN) + 1	LOOP	*	decfsz	COUNT	; decrement counter
2xN			goto	LOOP	
2			return		; done

* ถ้าผลลัพธ์จากการลดค่าเป็น 0 จะใช้สัญญาณนาฬิกา 2 ลูก

ดังนั้น ช่วงเวลาทั้งหมดที่ใช้ในการหน่วงเวลาสามารถคำนวณได้ ดังนี้

ช่วงเวลาทั้งหมด = 1 + 1+ N + 1 + 2N + 2 = 3N +5

* ถ้าใช้คริสตอล 4 เมกะเฮิรตซ์ สัญญาณนาฬิกา 1 ลูก ใช้เวลา 1 ไมโครวินาที ถ้าป้อนค่าตัวแปร N = 33 (ฐานสิบ) จะได้ช่วงเวลาในการหน่วงทั้งหมดเท่ากับ

ช่วงเวลาทั้งหมด = 3*(33) + 5

= 104 ไมโครวินาที (ความเร็ว 9600 bps.)

2.2 พื้นฐานการสื่อสารแบบอนุกรม

แม้ว่าการสื่อสารแบบอนุกรมในเครื่องคอมพิวเตอร์นั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน ทั้งนี้ก็เพราะว่าการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ครั้งละหลาย ๆ บิตพร้อมกัน ซึ่งส่งผลให้การสื่อสารข้อมูลแบบอนุกรมมีความเร็วต่ำกว่าแบบขนาน[3]

2.2.1 การส่งข้อมูลแบบอนุกรม

ในการส่งข้อมูลแบบอนุกรมนั้น มีข้อดีเหนือกว่าการส่งข้อมูลแบบขนาน คือ สามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังมีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วย การสื่อสารแบบอนุกรมสามารถแบ่งออกเป็น 3 รูปแบบ ดังนี้

2.2.1.1 Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว

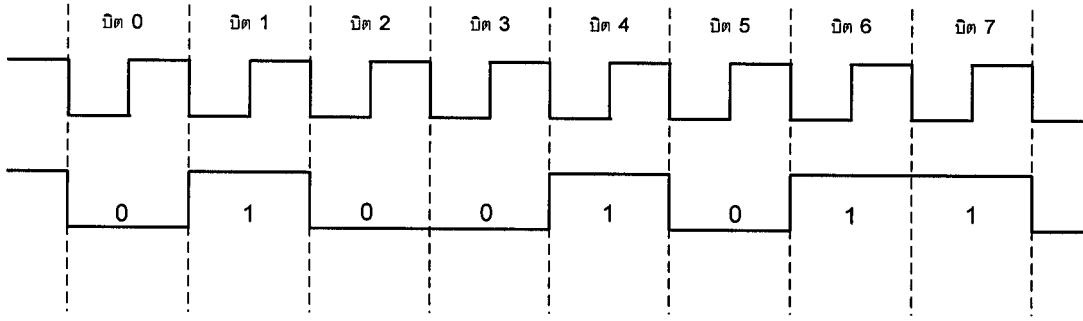
2.2.1.2 Half-Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลได้ในเวลาเดียวกัน

2.2.1.3 Full-Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้ ยังสามารถแบ่งประเภทของการสื่อสารแบบอนุกรม ตามลักษณะสัญญาณในการส่งได้อีก 2 แบบ คือ

2.2.2 การสื่อสารแบบซิงโครนัส (Synchronous)

สำหรับการสื่อสารแบบซิงโครนัสจะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณเช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และมีสายกราวด์ด้วย) ดังภาพที่ 12



ภาพที่ 12 ลักษณะสัญญาณของการสื่อสารแบบซิงโครนัส

การสื่อสารแบบซิงโครนัสนี้ เหมาะสำหรับการทำงานในระยะใกล้ ดังนั้นข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

2.2.3 การสื่อสารแบบอะซิงโครนัส (Asynchronous)

สำหรับการสื่อสารแบบอะซิงโครนัส จะใช้สายข้อมูลเพียงตัวเดียวแต่จะใช้รูปแบบการส่งข้อมูล หรือ Bit Pattern เป็นตัวกำหนดว่าส่วนใดเป็นส่วนเริ่มต้นข้อมูล, ส่วนใดเป็นตัวข้อมูล, ส่วนใดจะเป็นส่วนตรวจสอบความถูกต้องข้อมูล และส่วนใดเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่ง และภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ชื่อว่า UART หรือ Universal Asynchronous Receiver / Transmitter คอยควบคุม การรับ และส่งข้อมูล

ภาพที่ 13 UART อุปกรณ์ควบคุมการรับส่งข้อมูลแบบอะซิงโครนัส

สำหรับมาตรฐานของการส่งข้อมูลแบบอนุกรมอีกแบบที่ได้รับความนิยมอย่างสูงตั้งแต่อดีตถึงปัจจุบัน โดยใช้งานกันอย่างแพร่หลายทั้งการสื่อสาร และการควบคุมทางอุตสาหกรรมนั่นก็คือ มาตรฐาน RS-232C

2.2.4 มาตรฐาน RS-232C

มาตรฐาน RS-232C เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ ต่อพ่วงจากผู้ผลิตต่างกันสามารถใช้ร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวางมากที่สุดคือ มาตรฐาน RS-232C ซึ่งถูกประกาศใช้ในปี 1969 โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ในยุคแรกๆ การอินเตอร์เฟสแบบ RS-232C ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอล (DTE : Data Terminal Equipment) กับโมเด็ม (DCE : Data Communication Equipment) ทั้งนี้ก็เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเส้นเดียวกัน มาตรฐาน RS-232 ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้ คือ

2.2.4.1 อุปกรณ์ DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (เอาต์พุต)

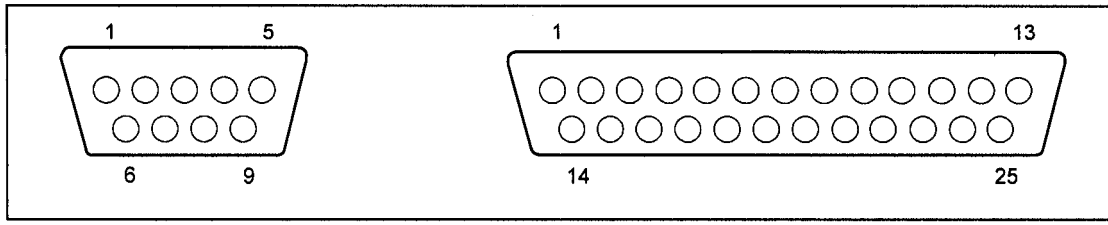
2.2.4.2 อุปกรณ์ DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล

ตามมาตรฐาน RS-232C แล้ว คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DEC จะเป็นตัวเมีย ซึ่งคอนเน็กเตอร์ที่นิยมใช้กันอยู่จะเป็นชนิด D-Type แบบ 9 ขา และแบบ 25 ขา โดยจะติดตั้งอยู่หลังเครื่องคอมพิวเตอร์ ระดับแรงดันจะมีค่าระหว่าง -3 V ถึง -15 V

สำหรับลอจิก High และลอจิก Low จะมีระดับแรงดันระหว่าง $\pm$3 V ถึง $\pm$15 V สามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุด 50 ฟุต หรือ 15 เมตร แต่ถ้าเราต้องการสื่อสารกับอุปกรณ์อื่นที่อยู่ห่างกันมาก ๆ เราจำเป็นต้องใช้อุปกรณ์อื่น ๆ เข้าช่วย เช่น การใช้โมเด็มเป็นต้น

2.2.5 ลักษณะของคอนเน็กเตอร์ แบบ D-Type

หัวต่อแบบ D-Type ที่ใช้ในการสื่อสารแบบอนุกรมของเครื่องคอมพิวเตอร์นั้นจะมี 2 ลักษณะ คือ แบบ 9 ขา และแบบ 25 ขา บางครั้งเราจะเรียกว่า DB9 หรือ DB25 ซึ่งหัวต่อทั้งสองชนิดจะมีลักษณะการทำงานของสัญญาณเหมือนกัน แต่การจัดเรียงไม่เหมือนกัน ดังภาพที่ 14



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

ภาพที่ 14 แผนผังคอนเน็กเตอร์ ของ RS-232C

2.2.6 รายละเอียดของสายสัญญาณ ประกอบด้วย

2.2.6.1 Transmit Data : TD ใช้สำหรับส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์

2.2.6.2 Receive Data : RD ใช้สำหรับรับข้อมูลอนุกรมเข้ามายังคอมพิวเตอร์

2.2.6.3 Request To Send : RTS ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทาง เพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา

2.2.6.4 Clear To Send : CTS ใช้สำหรับตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อด้วย พร้อมที่จะรับข้อมูลหรือไม่ โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะสามารถทำการส่งข้อมูลออกทางขา TD

2.2.6.5 Data Set Ready : DSR ใช้สำหรับตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง จะใช้คู่กับขา DTR

2.2.6.6 Signal Ground : SG เป็นกราวด์ของระบบ

2.2.6.7 Carrier Detect : CD ขานี้ Active เมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม

2.2.6.8 Data Terminal Ready : DTR ใช้สำหรับบอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง

2.2.6.9 Ring Indicator : RI ขานี้จะ Active เมื่อโมเด็มได้รับสัญญาณเรียกเข้าจากสายโทรศัพท์

2.2.7 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารข้อมูลแบบอะซิงโครนัสนั่นคือ ต้องใช้สายสัญญาณเส้นเดียวทำหน้าที่ทั้งรับและส่ง ส่วนที่เป็นข้อมูลและส่วนที่ใช้ควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิต จากการส่งแบบอนุกรม จึงต้องถูกแยกออกว่า ใช้สำหรับวัตถุประสงค์ใด โดยสามารถแบ่งได้เป็น 4 ส่วน คือ

(1) Start Bit ขนาด 1 บิต

(2) บิตข้อมูล (Data Character) ขนาด 7 บิต หรือ 8 บิต

(3) Parity Bit ขนาด 1 บิต

(4) Stop Bit ขนาด 1 บิต หรือ 2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วย บิตเริ่มต้น บิตข้อมูล บิตพาริตี้ (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราจะพอจะสรุปหน้าที่ของแต่ละส่วนได้ ดังนี้

2.2.7.1 Start Bit หรือบิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

2.2.7.2 Data Character หรือบิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่มๆ โดยทั่วไปจะส่งเป็น 7 หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง ASCII WORD

2.2.7.3 Parity Bit หรือบิตพาริตี้ ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตี้เข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตี้แบบไหนในการส่งข้อมูลซึ่งหลักการในการกำหนดบิตพาริตี้มีหลายแบบ ดังนี้

1) พาริตี้คู่ (Even Parity) ค่าของพาริตี้นี้เมื่อรวมกับทุก ๆ บิตของข้อมูลแล้วจะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้น บิตพาริตี้จะเป็น 0

2) พาริตี้คี่ (Odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุก ๆ บิตของข้อมูลแล้วจะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 1

3) ไม่มีพาริตี้ (None) ถ้าตั้งบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี้

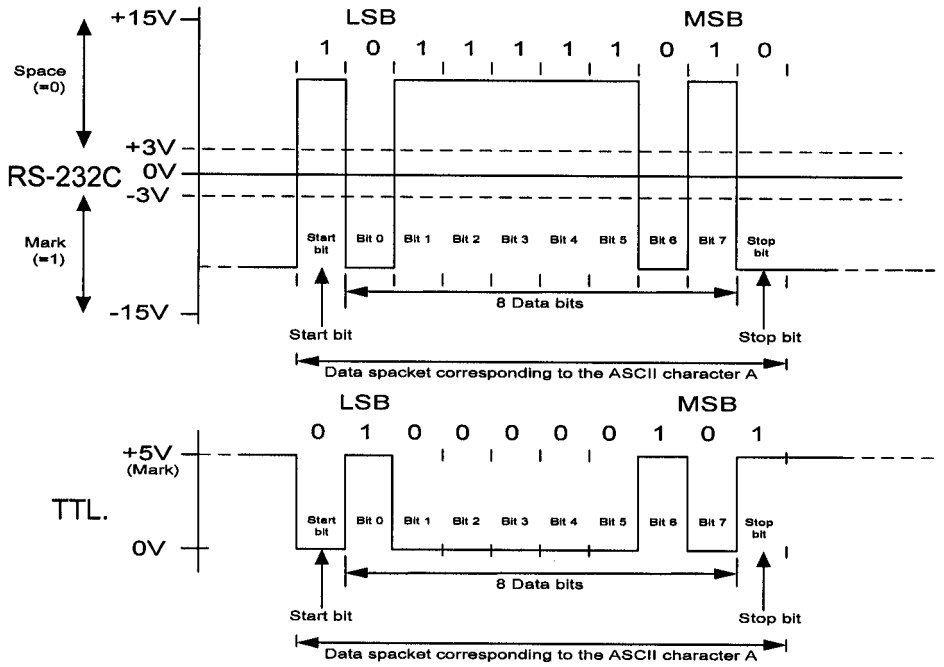
2.2.7.4 Stop Bit หรือบิตจบ เป็นบิตที่ส่งมาปิดท้ายข้อมูล

2.2.8 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัส คือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรม สำหรับมาตรฐาน RS-232C นั้นมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

2.2.9 สัญญาณทางไฟฟ้า

มาตรฐาน RS-232C ได้กำหนดลักษณะของสัญญาณทางไฟฟ้าที่ถูกใช้ในการเชื่อมต่อแบบ อนุกรมนี้มี 2 ลักษณะคือ SPACE แสดงถึง logic '1' และ MARK หมายถึง logic '0' โดย SPACE จะเป็นแรงดันไฟฟ้าบวก (+3 ถึง +15 V) MARK จะเป็นแรงดันไฟฟ้าลบ (-3 ถึง -15 V)



ภาพที่ 15 สัญญาณ RS-232C เมื่อเทียบกับสัญญาณ TTL. เมื่อ รับ-ส่ง อักขระตัว "A"

จากภาพที่ 15 ข้างบนจะเห็นว่าสัญญาณ RS-232C[4] กับ สัญญาณ TTL. (Transistor Transistor Logic ซึ่งมีระดับสัญญาณ 0 ถึง 5 โวลท์ ที่ใช้ในคอมพิวเตอร์ทั่วไป) มีความแตกต่างกันไม่ว่าเฟสของสัญญาณที่ตรงกันข้าม หรือ ระดับโวลท์ที่ใช้ในการทำงาน โดยถ้าหากไม่มีการส่งข้อมูลสัญญาณจะเป็น Mark ตลอด จากภาพจะเป็นการส่งอักขระตัว "A" การส่งจะต้องเริ่มด้วย Start Bit (ส่ง Space ไป 1 bit) ตามด้วย Data ขนาด 8 bits ซึ่งอักขระตัว "A" นั้น ค่า Data เป็นรหัสแอสกี้ 41 ในฐาน 16 หรือ 0100 0001 ในฐาน 2 แต่ในการส่งนั้นจะต้องส่ง bit ที่มีค่า

น้อย หรือ LSB (Least Significant Bit) จะอยู่ทางขวามือไปก่อนตามลำดับ คือ 10000010 เสร็จแล้วจะตามด้วยบิตจบ (ส่ง Mark ไป 1 bit) จากภาพข้างบนจะเป็นภาพของสัญญาณ RS-232C และ TTL. ที่ใช้ 1 บิตเริ่มต้น 8 บิตข้อมูล ไม่มีบิตตรวจสอบ(พาริตี้บิต) และใช้บิตจบ 1 บิต

ส่วนอัตราเร็วในการรับส่งข้อมูลหรือ ค่าบอดเรต (Baud Rate) นั้น จะเป็นค่าที่กำหนดเวลาของบิตแต่ละบิต ถ้าค่าบอตเรตมาก คาบเวลาของแต่ละบิตจะน้อย ถ้าหากค่าบอตเรตน้อย ค่าของคาบเวลาจะมากเช่น ค่าบอดเรต = 9600 จะมีค่าคาบเวลาของแต่ละบิต = 104 ไมโครวินาที แต่ถ้าค่าบอตเรต = 2400 จะมีค่าคาบเวลาของแต่ละบิต = 417 ไมโครวินาที ดังภาพที่ 16 โดยใช้สูตรในการคำนวณ ดังนี้

$$T = 1/F$$

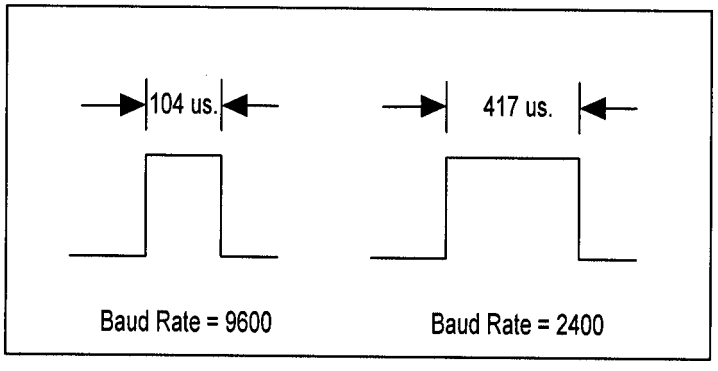
T = Time หรือคาบเวลาของแต่ละบิต

F = Frequency หรือความถี่แต่ในที่นี้คือความเร็ว หรือ บอตเรต (Baud Rate)

ตัวอย่าง

$1/9600 = 104 * 10^{-6}$ หรือ 104 ไมโครวินาที

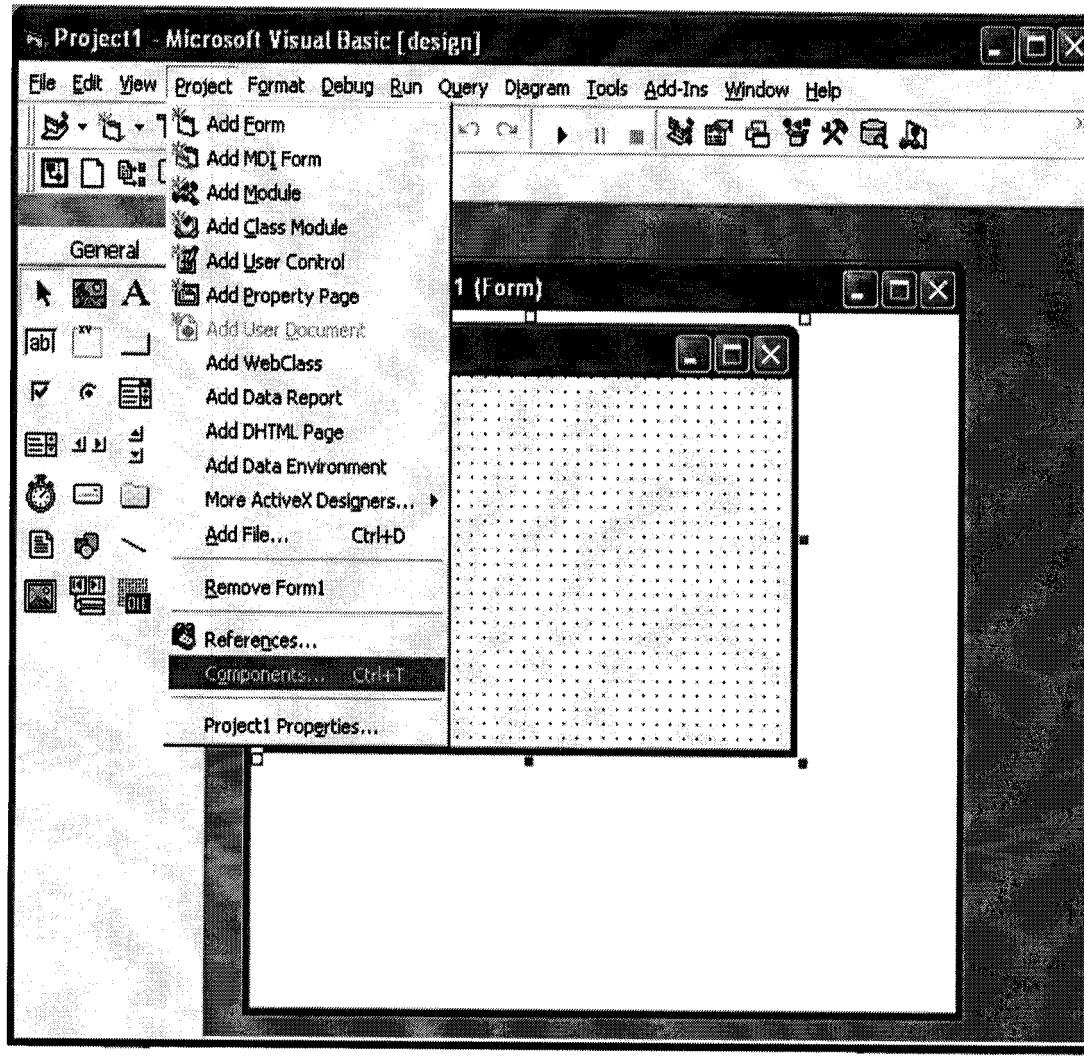
$1/2400 = 417 * 10^{-6}$ หรือ 417 ไมโครวินาที



ภาพที่ 16 คาบเวลาของแต่ละบิต เมื่อเทียบกับบอตเรต (Baud Rate)

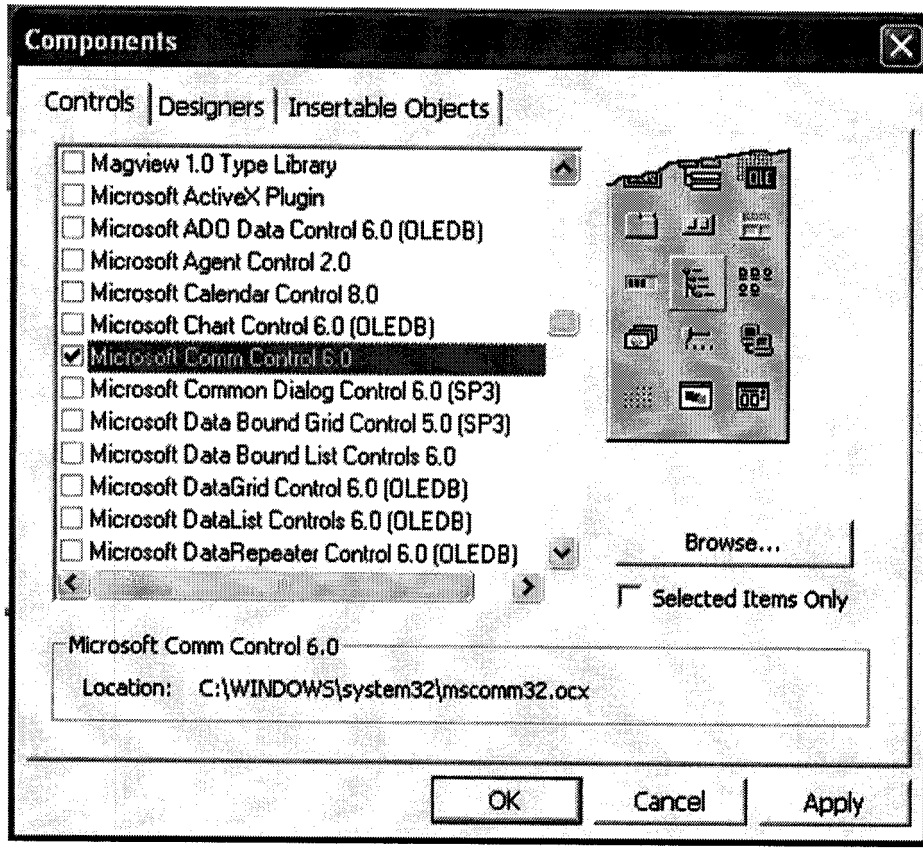
2.3 การเขียนโปรแกรมติดต่อ และควบคุม Serial Port ด้วย Visual Basic

คอนโทรลที่สำคัญในการทำให้ Visual Basic สามารถสื่อสารผ่านพอร์ตอนุกรมได้นั้น ก็คือคอนโทรล MSComm ซึ่งไม่ใช่คอนโทรลมาตรฐาน[3] ดังนั้นถ้าเราต้องการใช้งาน MSComm เราจะต้องทำการเพิ่มคอนโทรลนี้เข้าไปใน Tool Box ซึ่งสามารถกระทำได้โดยคลิกขวาที่ Toolbox แล้วเลือกเมนู Components ดังแสดงในภาพที่ 17



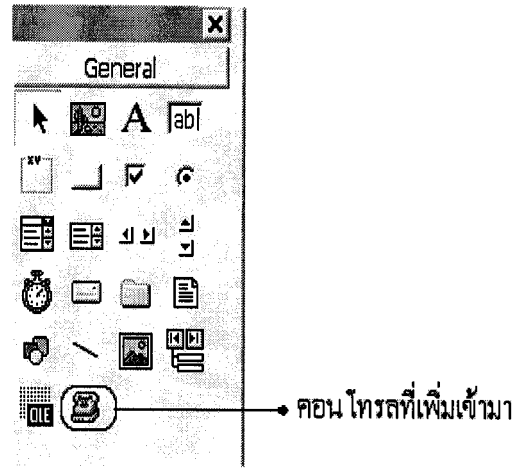
ภาพที่ 17 การเพิ่มคอมโพเนนต์ MSComm

จากนั้นจะปรากฏไดอะล็อก Components ขึ้นมา จากนั้นให้คลิกเลือกที่ Microsoft Comm Control 6.0 แล้วคลิกปุ่ม OK ดังภาพที่ 18



ภาพที่ 18 การเลือกที่รายการ MSComm

จากนั้นก็จะปรากฏภายใน Toolbox จะมีไอคอนรูปโทรศัพท์ ซึ่งเป็นไอคอนของคอนโทรล MSComm ปรากฏขึ้นมาให้เราเลือกใช้งาน



ภาพที่ 19 แถบเครื่องมือควบคุม MSComm พร้อมทำงาน

2.3.1 พร็อพเพอร์ตี้ที่สำคัญในการใช้งาน MSComm

2.3.1.1 CommPort ใช้ในการกำหนดหมายเลขของพอร์ตอนุกรมที่เราต้องการจะติดต่อ โดยมีรูปแบบของการใช้งาน ดังนี้

```
object.CommPort [ = value ]
```

ตัวอย่าง เรากำหนดให้การเขียนโปรแกรมติดต่อกับพอร์ต Com1 จะเขียนเป็น

```
MSComm1.CommPort = 1
```

2.3.1.2 Settings ใช้ในการกำหนดอัตราบอด (Baud Rate) หรือความเร็วในการส่งข้อมูล มีหน่วยเป็นบิตต่อวินาที, พาริตี้, จำนวนของบิตข้อมูล, จำนวนของบิตปิดท้าย โดยมีรูปแบบของการใช้งาน ดังนี้

```
object.Settings [ = value ]
```

ตัวอย่าง เรากำหนดให้มีการเขียนโปรแกรมใช้แกรมใช้งานที่ Baud Rate = 9600 บิตต่อวินาที ไม่มีพาราริตี้ จำนวนบิตข้อมูลเท่ากับ 8 บิต และมีบิตปิดท้าย 1 บิต

```
MSComm1.Settings = "9600, N, 8, 1"
```

2.3.1.3 PortOpen ใช้สำหรับและปิดการใช้งานพอร์ตอนุกรม โดยมีรูปแบบของการทำงาน ดังนี้

```
object.PortOpen = true
```

ตัวอย่าง เราจะเปิดใช้งานพอร์ตอนุกรม ให้กำหนดค่า Value เป็น True เขียนโค้ด ดังนี้

```
MSComm1.PortOpen = True
```

แต่ถ้าต้องการปิดพอร์ตอนุกรม ให้กำหนดค่า Value เป็น False เช่น

```
MSComm1.PortOpen = False
```

2.3.1.4 InBuffersize เป็นการกำหนดขนาด Buffer ในการรับข้อมูลเข้ามา โดยมีรูปแบบการทำงาน ดังนี้

```
object.InBuffersize [ = value ]
```

2.3.1.5 OutBuffersize เป็นการกำหนดขนาดของ Buffer ในการส่งข้อมูลออกไป โดยมีรูปแบบการกำหนดค่า ดังนี้

```
object.OutBuffersize [ = value ]
```

2.3.1.6 Inputlen เป็นการกำหนดค่าของข้อมูลที่อ่านจาก Buffer ภาครับ โดยมีรูปแบบการกำหนดค่า ดังนี้

```
object.Inputlen [ = value ]
```

2.3.1.7 InputMode เป็นการกำหนดค่าชนิดของข้อมูลที่รับเข้ามา โดยมีรูปแบบการกำหนดค่า ดังนี้

```
object.InputMode [ = value ]
```

โดยที่เราสามารถเลือกชนิดของข้อมูลได้ 2 ประเภท คือ

(1) comInputModeText ข้อมูลที่รับเข้ามาเป็นข้อความปกติเราสามารถตั้งค่าให้อยู่ในโหมดนี้ได้โดยกำหนด value ให้เป็น “0”

(2) comInputModeBinary ข้อมูลที่รับเข้ามาเป็นข้อมูลไบนารี่ เราสามารถตั้งค่าให้อยู่ในโหมดนี้ได้โดยการกำหนดค่า value ให้เป็น “1”

2.3.1.8 Input ใช้ในการอ่านค่าข้อมูลจากพอร์ตอนุกรม โดยมีรูปแบบการอ่านค่าดังนี้

```
object.Input
```

ตัวอย่าง เราอ่านค่าจาก Buffer ของพอร์ตอนุกรม แล้วนำมาเก็บไว้ในตัวแปรที่ชื่อว่า Data เราจะเขียนโค้ด ดังนี้

```
Data = MSComm1.Input
```

2.3.1.9 Output ใช้ในการส่งข้อมูลออกไปจากพอร์ตอนุกรม โดยมีรูปแบบของการเขียน ดังนี้

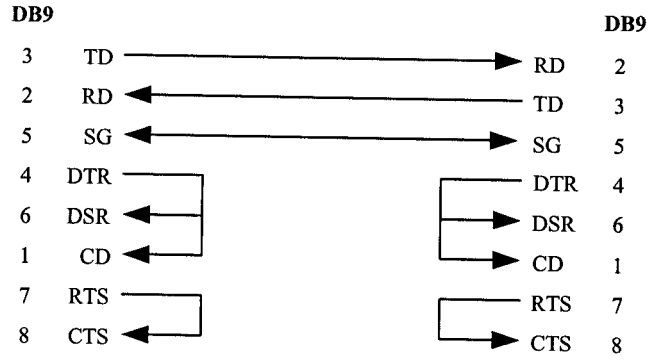
```
object.Output [ = value ]
```

2.3.1.10 EOFEnable เป็นการบอกว่าสิ้นสุดของไฟล์ End of File [EOF] โดยมีรูปแบบการใช้งาน ดังนี้

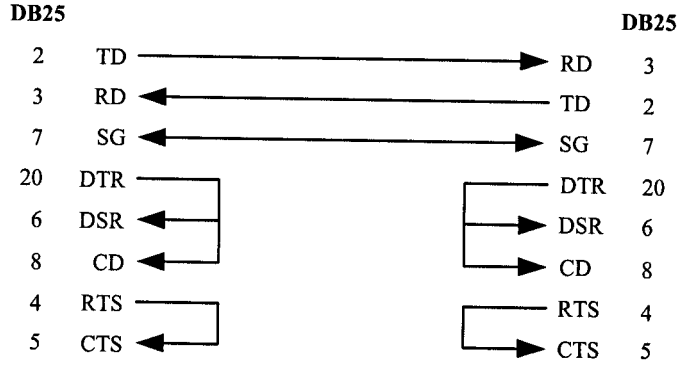
```
object.EOFEnable [ = value ]
```

2.3.2 การเชื่อมต่อระหว่าง PC กับ PC ด้วยพอร์ตอนุกรม

ถ้าเราต้องการทำการแลกเปลี่ยนข้อมูลกันระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง ผ่านพอร์ตอนุกรม ซึ่งเป็นการเชื่อมต่อระหว่าง DTE กับ DTE เข้าด้วยกัน ในลักษณะเช่นนี้เราจะต้องทำการเชื่อมต่อสายขาที่ 3 ซึ่งเป็นสายสำหรับส่งข้อมูลบนอุปกรณ์ตัวแรกเข้ากับสายขาที่ 2 ซึ่งเป็นสายสำหรับรับข้อมูลบนอุปกรณ์ตัวที่สอง และเชื่อมต่อสายขาที่ 2 บนอุปกรณ์ตัวแรกเข้ากับสายขาที่ 3 บนอุปกรณ์ตัวที่สอง ส่วนสายอื่น ๆ ก็ต้องถูกไขว้ด้วย ดังวงจรสำหรับการเชื่อมต่อคอนเน็กเตอร์แบบ D-Type 9 ขา ดังแสดงในภาพที่ 20 และ แบบ D-Type 25 ขา ดังภาพที่ 21



ภาพที่ 20 การเชื่อมต่อคอนเน็กเตอร์ แบบ D-Type 9 ขา



ภาพที่ 21 การเชื่อมต่อคอนเน็กเตอร์ แบบ D-Type 25 ขา

เมื่อทราบถึงวิธีการเชื่อมต่อระหว่าง PC กับ PC แล้วให้ผู้อ่านทำการสร้างสายเชื่อมต่อนี้ไว้สำหรับทดลองโปรแกรมได้เลย สายอินเตอร์เฟสนี้จะใช้คอนเน็กเตอร์แบบ Female 2 ตัว หรือจะซื้อสายที่ทำสำเร็จรูปมาเลยก็ได้ ซึ่งมีขายตามร้านขายอุปกรณ์อิเล็กทรอนิกส์ และร้านขายอะไหล่ เครื่องคอมพิวเตอร์ทั่วไป โดยบอกกับทางร้านว่าซื้อสายที่ใช้ต่อกับพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

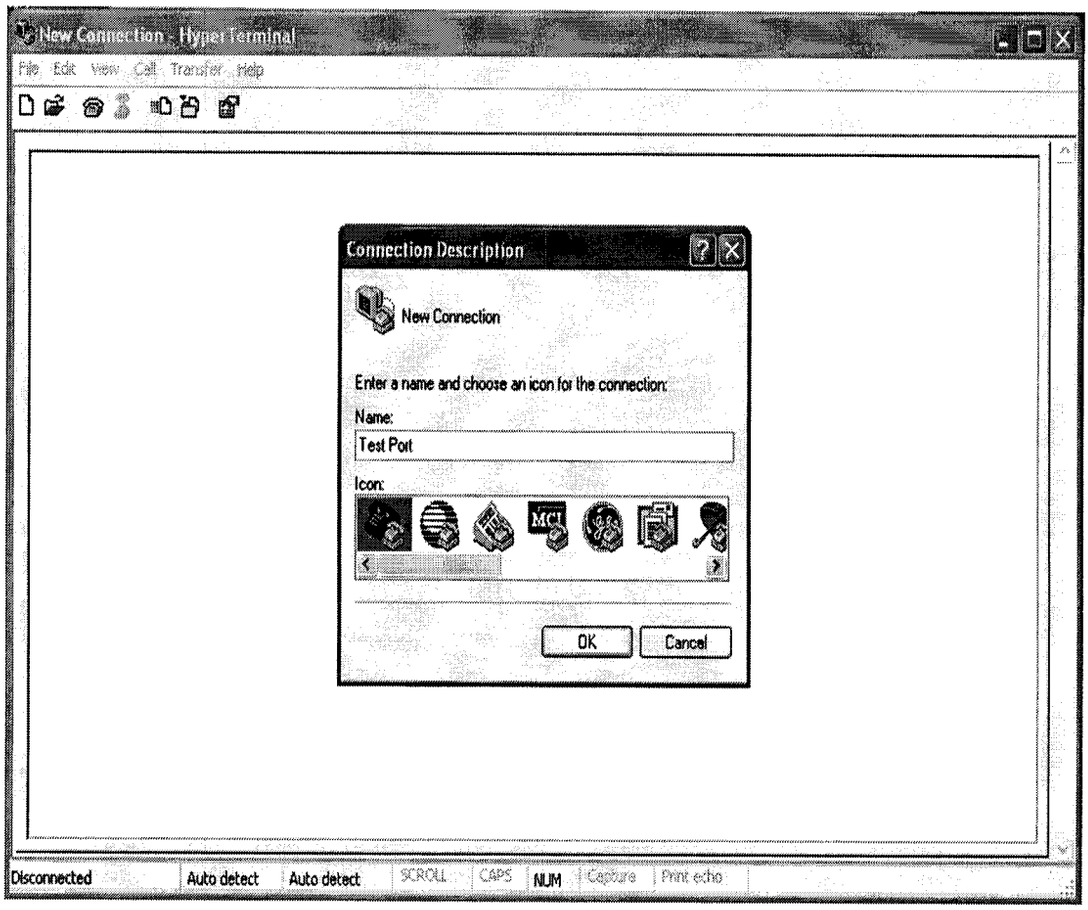
2.3.3 การทดสอบสายเชื่อมโยง

เมื่อเราทำการสร้างสายเชื่อมโยงเสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปเราจะทำการทดสอบสายเชื่อมโยงโดยใช้โปรแกรม HyperTerminal ซึ่งเป็นโปรแกรมสำเร็จ ที่มาพร้อมกับระบบWindows ใช้สำหรับการติดต่อสื่อสารผ่านพอร์ตอนุกรม [3]

2.3.3.1 การเรียกใช้งานโปรแกรม Hyperterminal

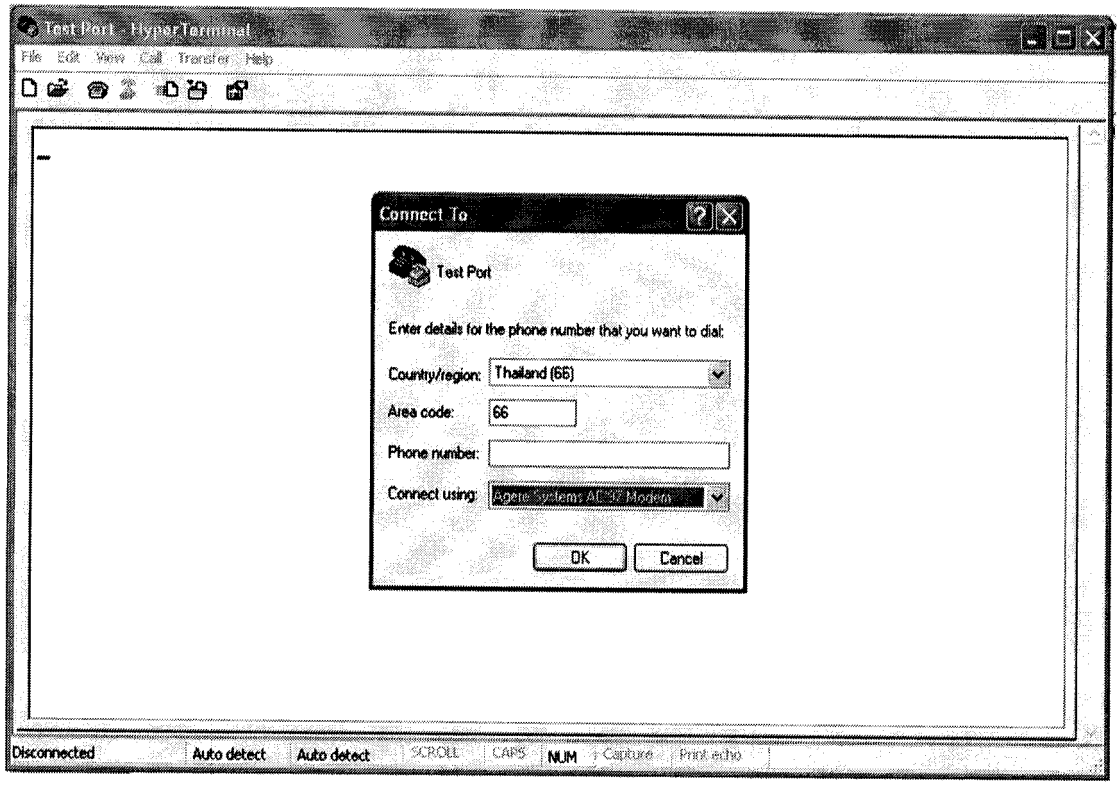
ให้คลิกที่ Start > Program > Accessories > Communication >

HyperTerminal จากนั้นให้เลือก Hypertrm.exe เมื่อเปิดโปรแกรมขึ้นมาโปรแกรมจะแสดงหน้าต่างของ Connection Description เพื่อให้เราทำการสร้าง Connection ขึ้นใหม่ ในที่นี้เราจะสร้าง Connection ที่ชื่อว่า Test Port ดังแสดงในภาพที่ 22



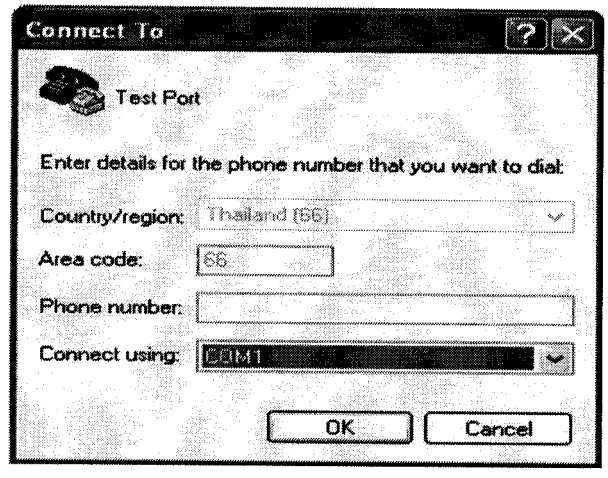
ภาพที่ 22 การสร้าง Connection ใหม่

จากนั้นให้คลิกปุ่ม OK ก็จะปรากฏไดอะล็อก Connection To ดังแสดงในภาพที่ 23



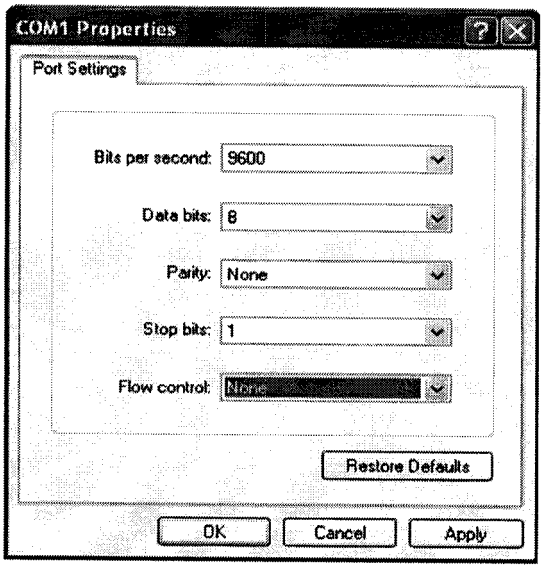
ภาพที่ 23 การกำหนดรายละเอียดของ Connection

จากนั้นให้เลือกที่หัวข้อ Connect using : เป็น Direct to Com1 ภาพที่ 24



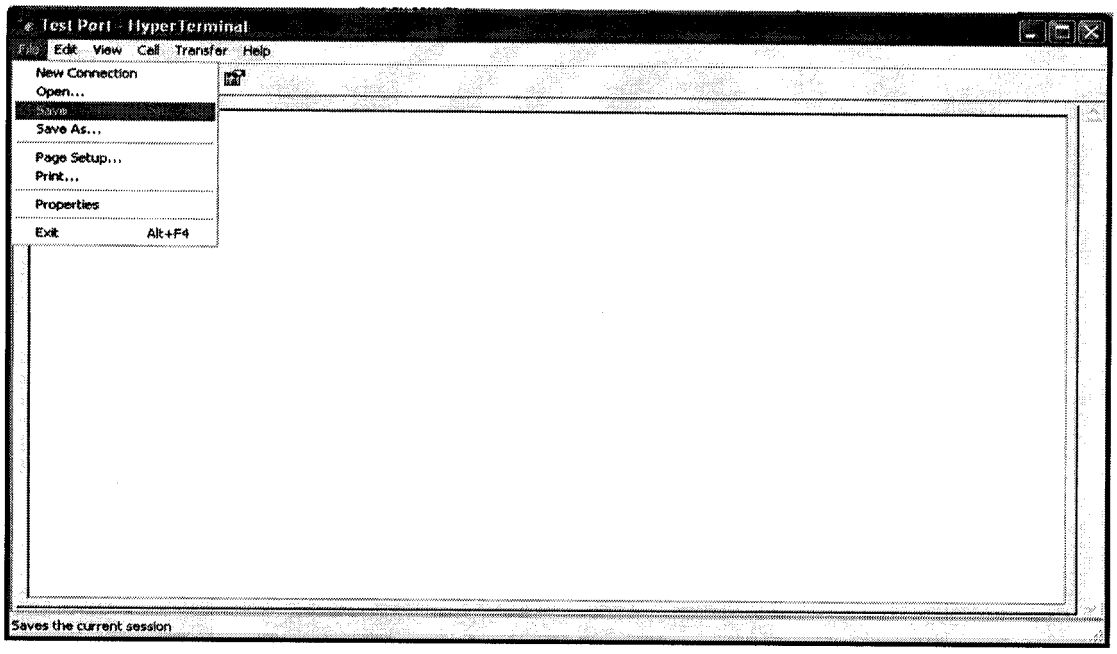
ภาพที่ 24 การ Connect using โดยการเลือก COM1

เมื่อคลิกปุ่ม OK ก็จะปรากฏไดอะล็อก Com1 Properties ให้ทำการกำหนดค่า Port Settings ดังภาพที่ 25



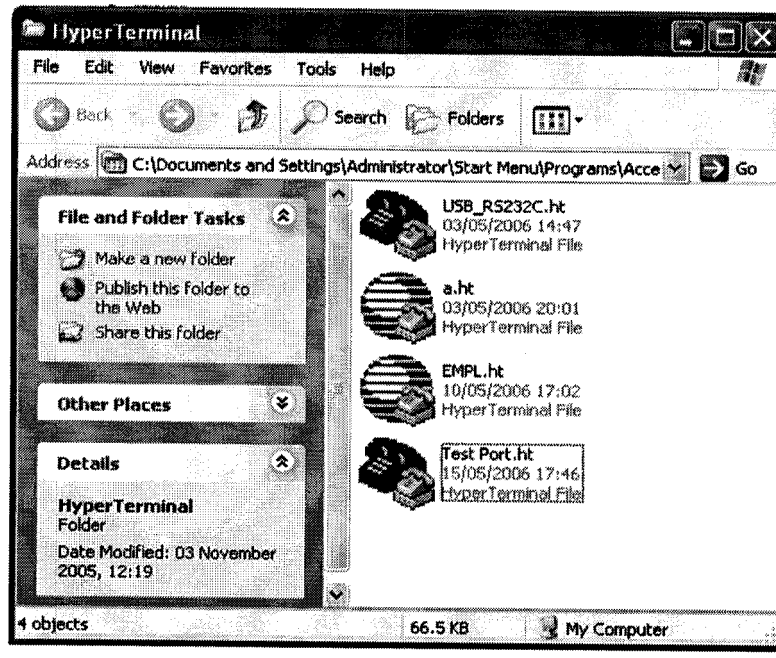
ภาพที่ 25 การกำหนดคุณสมบัติจุดเชื่อมต่อให้กับ Connection

เมื่อเสร็จแล้วให้คลิกปุ่ม OK โปรแกรมก็จะทำการเชื่อมต่อระบบให้ จากนั้นให้เราบันทึก Connection ที่สร้างไว้ ดังภาพที่ 26 เพื่อให้โปรแกรมทำการสร้างไอคอนของ Connection ที่ชื่อ Test port เสร็จแล้วปิดโปรแกรม



ภาพที่ 26 การบันทึก Connection ที่สร้างขึ้น

เมื่อต้องการเปิดโปรแกรมเพื่อใช้งานใหม่ เราก็สามารถดับเบิ้ลคลิกที่ไอคอนชื่อว่า Test Port โดยไม่ต้องเปิดโปรแกรม HyperTerminal ขึ้นมาเพื่อสร้าง Connection ใหม่



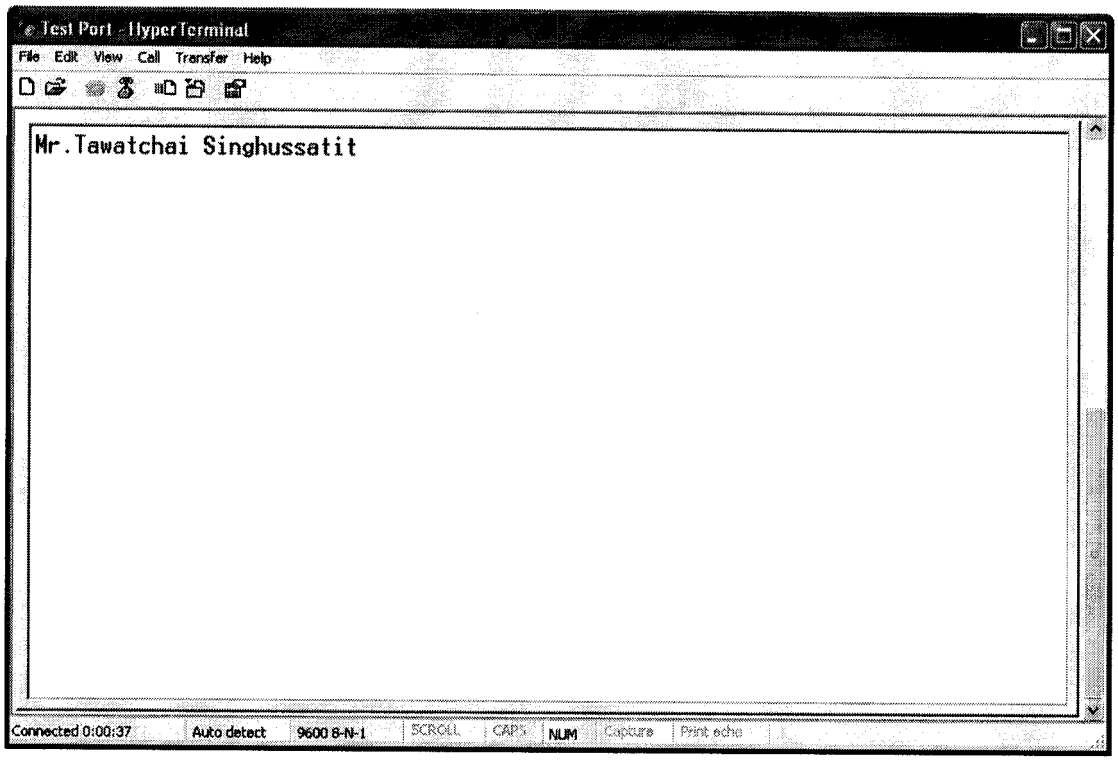
ภาพที่ 27 ไอคอนของ Connection ที่สร้างขึ้น

2.3.3.2 การทดสอบสายสัญญาณ

ต่อไปเราจะใช้โปรแกรม HyperTerminal เพื่อทดสอบสายสัญญาณที่เราได้สร้างขึ้นมา ในการทดสอบสายสัญญาณนี้เราจะใช้คอมพิวเตอร์เพียงเครื่องเดียวในการรับและส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม ดังนั้นเราจะต้องทำการจั๊มสายสัญญาณเส้นที่ 2 และเส้นที่ 3 ที่ปลายด้านใดด้านหนึ่งเข้าด้วยกัน

ส่วนอีกด้านหนึ่งให้ต่อเข้ากับพอร์ตอนุกรม (Com1) ของเครื่องคอมพิวเตอร์ จากนั้นให้เปิด Connection ที่ชื่อว่า Test Port ที่เราสร้างไว้แล้วก่อนหน้านี้ โดยให้คลิกที่ Start > Program > Accessories > Communication > Hyperterminal จากนั้นให้คลิกที่ Icon ของ Test Port

เมื่อโปรแกรมเปิดขึ้นมาแล้วให้ทดลองพิมพ์อะไรก็ได้ ตัวอักษรที่เราพิมพ์ไปนั้นก็จะปรากฏที่ตัวโปรแกรม

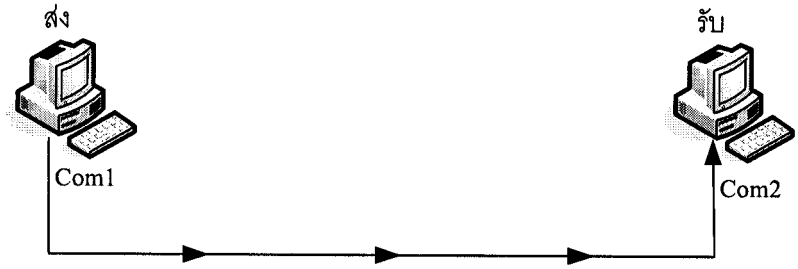


ภาพที่ 28 การทดสอบการทำงานของ Connection

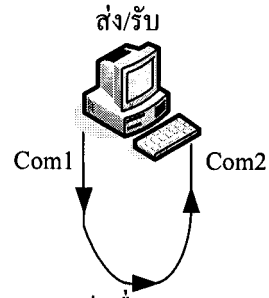
จากนั้นทดลองดึงสายสัญญาณออก แล้วลองพิมพ์อะไรก็ได้อีกครั้ง สังเกตตัวอักษรที่พิมพ์ไปจะไม่ปรากฏที่ตัวโปรแกรม ที่เป็นเช่นนี้ก็เพราะว่าโปรแกรมได้ทำการส่งข้อมูลออกไปจากพอร์ตอนุกรมแล้ว แต่ไม่สามารถรับข้อมูลเข้ามาเพื่อแสดงได้เพราะไม่มีตัวเชื่อมโยงสัญญาณนั่นเอง

2.3.4 การเขียนโปรแกรมเพื่อติดต่อและควบคุมผ่านพอร์ตอนุกรม

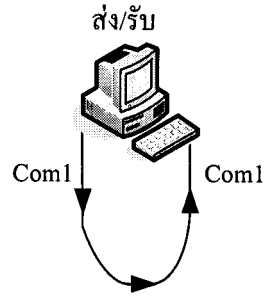
ตัวอย่างการเขียนโปรแกรมควบคุมผ่านพอร์ตอนุกรม ด้วย Visual Basic 6 ในการทดลองนั้น จำเป็นต้องมีคอมพิวเตอร์ 2 เครื่อง เพื่อทำการรับและส่งข้อมูลระหว่างกัน โดยเครื่องหนึ่งจะเป็นตัวส่งข้อมูล และอีกเครื่องหนึ่งจะเป็นตัวรับข้อมูลเพื่อทำการแสดงผล แต่หากมีคอมพิวเตอร์เพียงแค่ 1 เครื่องก็สามารถทำการทดลองได้เช่นกัน สำหรับผู้ที่มีคอมพิวเตอร์เพียงเครื่องเดียวซึ่งโปรแกรมที่ใช้ทดลองด้วยคอมพิวเตอร์เพียงเครื่องเดียวนั้น จะเป็นการเขียนโปรแกรมเพื่อมิให้คอมพิวเตอร์สามารถรับและส่งข้อมูลผ่านพอร์ตอนุกรมได้ในตัวเดียวกัน แต่จะต้องทำการต่อที่ขา 2 และขา 3 ของปลายสายสัญญาณที่จะใช้ในการอินเตอร์เฟสผ่านพอร์ตอนุกรมเข้าด้วยกัน แต่หากคอมพิวเตอร์ที่ใช้ในการทดลองมีพอร์ตอนุกรม 2 พอร์ตในเครื่องเดียวกันก็สามารถทดลองโปรแกรมได้ทั้งหมด แต่ต้องเปลี่ยนแปลงโปรแกรมบางส่วน
ดูภาพที่ 29 ประกอบ



กรณีคอมพิวเตอร์คนละเครื่อง



กรณีคอมพิวเตอร์เครื่องเดียว 2 พอร์ตอนุกรม



กรณีคอมพิวเตอร์เครื่องเดียว 1 พอร์ตอนุกรม

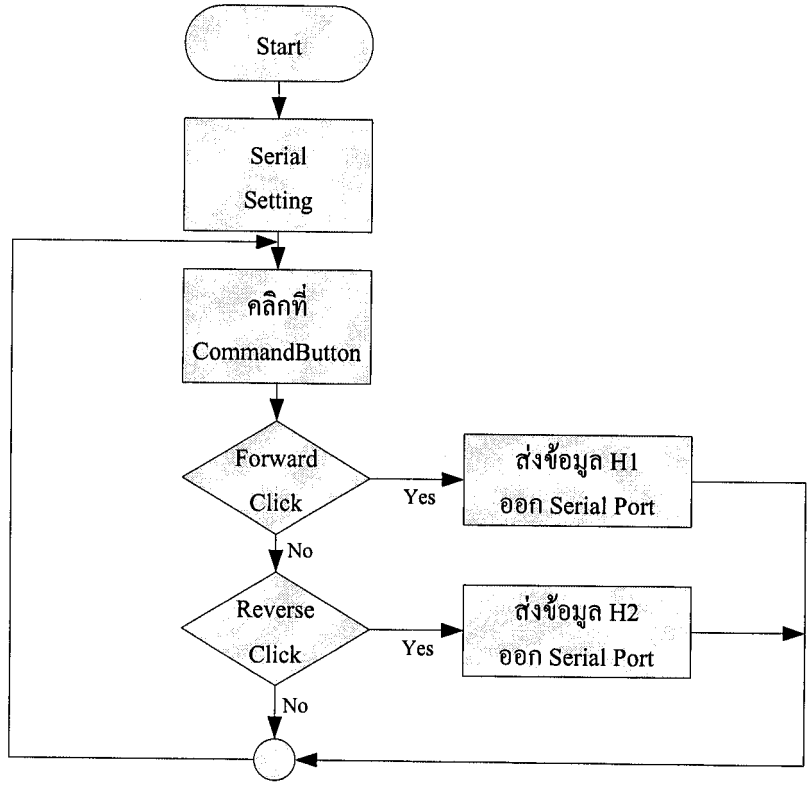
ภาพที่ 29 แนวคิดในการทดลอง

2.3.4.1 การรับและส่งข้อมูลผ่านพอร์ตอนุกรม

โปรแกรมที่นำมาเสนอนี้แบ่งออกเป็น 2 ส่วนคือ ส่วนรับข้อมูลผ่านทางพอร์ตอนุกรมเพื่อนำมาแสดงผล และอีกส่วนหนึ่งเป็นส่วนที่ใช่ส่งข้อมูลออกไปเพื่อให้ส่วนที่รับข้อมูล รับข้อมูลไปประมวลผล

2.3.4.2 สร้างโปรแกรมส่วนที่ส่งข้อมูล

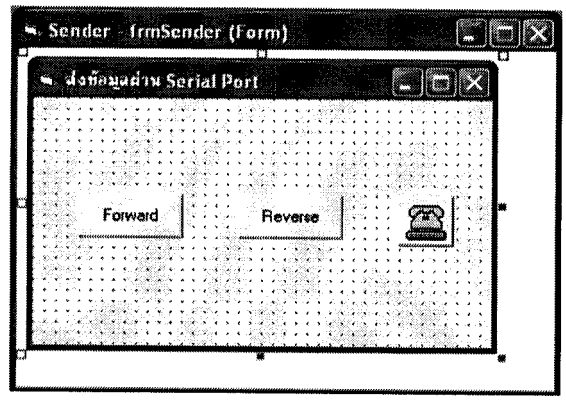
การจะส่งข้อมูลออกทางพอร์ตอนุกรม โดยการคลิกที่ปุ่มจากผู้ใช้งาน ซึ่งมีรายละเอียดการทำงานดังผังงาน ดังต่อไปนี้



จากผังงานข้างต้นนั้น สามารถเขียนโปรแกรมด้วยโปรแกรม Visual Basic โดยการใช้คอนโทรล MSComm เข้ามาทำหน้าที่ส่งข้อมูลไปยังพอร์ตอนุกรม ทำตามขั้นตอนดังต่อไปนี้

1) ให้ทำการเปิด Project ใหม่โดยคลิกที่ File > New Project ให้เลือก Standard EXE เพื่อเปิดฟอร์มขึ้นมา

2) เมื่อมีฟอร์มปรากฏขึ้นมาแล้วให้นำคอนโทรลต่าง ๆ มาจัดวางดังแสดงในภาพที่ 30



ภาพที่ 30 หน้าตาของโปรแกรมด้านส่ง

(3) กำหนดพร็อพเพอร์ตี้ให้กับคอนโทรลต่าง ๆ ดังต่อไปนี้

คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
Form	Name	frmSender
	Caption	ส่งข้อมูลผ่าน Serial
CommandButton	Name	cmdForward
	Caption	Forward
ConmandButton	Name	cmdReverse
	Caption	Reverse
MSComm	Name	mscSender
	InBufferSize	1024

(4) เขียนโค้ดคำสั่งเพื่อควบคุมการทำงานของโปรแกรม ดังนี้

```
Private Sub Form_Load()
mscSender.CommPort =1
mscSender.Setting = "9600, n , 8, 1"
mscSender.PortOpen = True
End Sub
```

```
Private  Sub cmdForward()
mscSender.Output = Chr(&H1)
End Sub
```

```
Private Sub cmdReverse_Click()
mscSender.Output = Chr(&H2)
End Sub
```

จากโค้ดควบคุมการทำงานนั้น เมื่อฟอร์มถูกโหลดขึ้นมาจะมีการกำหนดคุณสมบัติของคอนโทรล MSComm ดังรายละเอียดต่อไปนี้

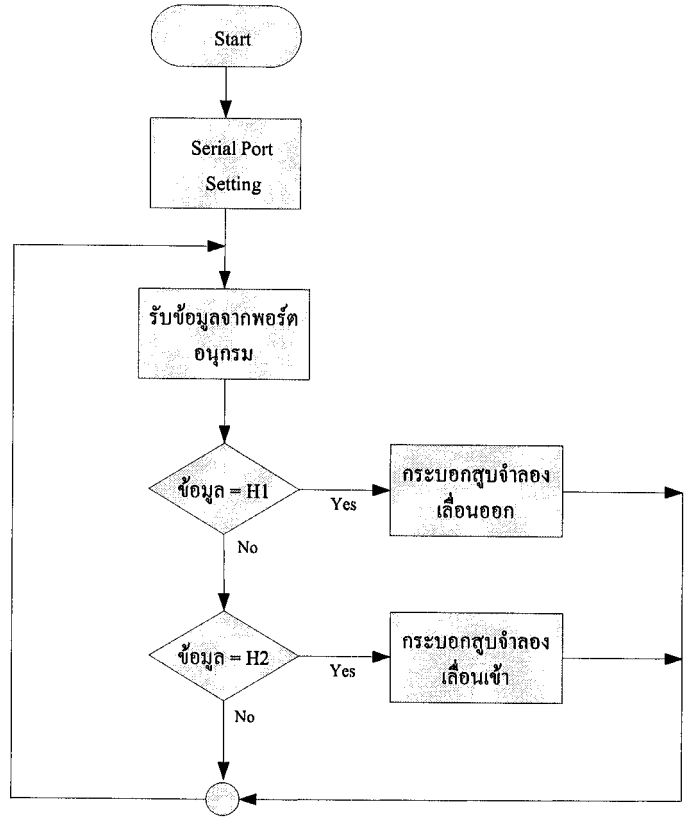
```
mscSender.ComPort = 1                    'ติดต่อกับพอร์ต Com1
mscSender.Settings ="9600, n, 8, 1,"     'กำหนดคุณสมบัติให้พอร์ต
mscSender.PortOpen = True                'เปิดการใช้งานพอร์ตอนุกรม
```

ในคำสั่งแรกจะเป็นการติดต่อกับพอร์ต Com1 ส่วนคำสั่งที่สองจะกำหนดอัตราบอด (Baud Rate) เท่ากับ 9600 bps ไม่มีพาริตี จำนวนบิตข้อมูลเท่ากับ 8 และมีบิตปิดท้าย 1 บิต สำหรับคำสั่งสุดท้ายจะเป็นการเปิดใช้งานพอร์ตอนุกรม

จากนั้นเมื่อผู้ใช้งานได้คลิกปุ่ม Forward หรือ ปุ่ม Reverse ก็จะมีการส่งข้อมูลออกไปยังพอร์ตอนุกรม Com1 ซึ่งคลิกหนึ่งครั้งก็จะส่งออกไปหนึ่งครั้ง

2.3.4.3 สร้างโปรแกรมส่วนรับข้อมูล

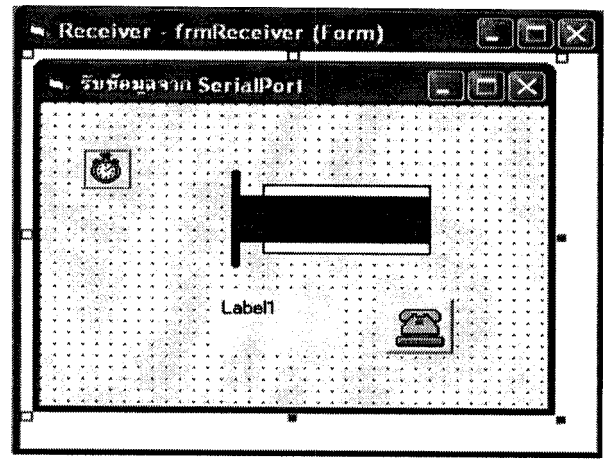
หลังจากสร้างส่วนที่ทำหน้าที่ส่งข้อมูลแล้ว ต่อไปเราจะสร้างส่วนที่รับข้อมูล โดยจะใช้งานคอนโทรล MSComm เช่นกัน โดยที่เมื่อรับข้อมูลจากโปรแกรมที่ทำหน้าที่ส่งข้อมูลแล้ว จะไปสั่งให้กระบอกสูบจำลองมีการทำงานสอดคล้องกับการคลิกปุ่ม Forward หรือ Reverse จากตัวอย่างที่แล้ว โปรแกรมจะตรวจสอบข้อมูลที่รับเข้า หากเป็น H1 กระบอกสูบจำลองจะเลื่อนออก แต่ถ้าหากเป็น H2 กระบอสูบจำลองจะเลื่อนเข้า ดังผังงานโปรแกรมต่อไป



การเขียนโปรแกรมในส่วนการรับข้อมูลจากพอร์ตอนุกรม ดังนี้

(1) ให้ทำการเปิด Project ใหม่โดยคลิกที่ File > New Project ให้เลือก Standard EXE เพื่อเปิดฟอร์มขึ้นมา

(2) เมื่อมีฟอร์มปรากฏขึ้นมาให้นำคอนโทรลต่าง ๆ มาจัดวาง ดังแสดงในภาพที่ 31



ภาพที่ 31 หน้าตาของโปรแกรมด้านรับ

(3) กำหนดค่าพร็อพเพอร์ตี้ให้กับคอนโทรลต่าง ๆ

คอนโทรล	พรอพเพอร์ตี้	ค่าที่กำหนด
Form	Name	frmReceiver
	Caption	รับข้อมูลจาก Serial Port
Shape	Name	Shape1
	BackColor	&H80000005& (สีขาว)
	BackStyle	1 – Opaque
	Shape	0 – Rectangle
Shape	Name	Shape2
	BackColor	&H000000FF& (สีแดง)
	BackStyle	1 – Opaque
	Shape	0 – Rectangle
Line	Name	Line1
	BorderColor	&H80000008& (สีดำ)
	BorderWidth	5
Label	Name	lblStatus
MSComm	Name	mscReceiver
Timer	Name	Timer1

2.4 พื้นฐานอิเล็กทรอนิกส์

การเขียนโปรแกรมเกี่ยวกับฮาร์ดแวร์นั้น สิ่งหนึ่งที่จำเป็นต้องมีคือพื้นฐานความรู้เกี่ยวกับไฟฟ้าและอิเล็กทรอนิกส์เบื้องต้น ทั้งนี้ เพราะการจะเขียนโปรแกรมได้นั้นจำเป็นต้องรู้พื้นฐานของการทำงานภายในของโปรแกรม ซึ่งต้องอาศัยความรู้ในด้านนี้ประกอบด้วย [3] หากต้องการปรับปรุงหรือพัฒนาความรู้เกี่ยวกับไฟฟ้าและอิเล็กทรอนิกส์นับว่าเป็นเรื่องที่จำเป็นและสำคัญ ซึ่งจะทำให้สามารถเขียนโปรแกรมและใช้งานได้อย่างปลอดภัยมากยิ่งขึ้น โดยมีรายละเอียดดังนี้

2.4.1 ความรู้เบื้องต้นเกี่ยวกับไฟฟ้าและอิเล็กทรอนิกส์ โดยมีนิยามศัพท์เกี่ยวกับอิเล็กทรอนิกส์เบื้องต้นดังนี้

2.4.1.1 กระแสไฟฟ้า (Electrical Current) คือ จำนวนอิเล็กตรอนที่ไหลผ่านตัวนำในวงจรไฟฟ้ามีหน่วยเป็นแอมแปร์ (Amperes) กระแสไฟฟ้า 1แอมแปร์เท่ากับแรงเคลื่อนไฟฟ้า 1 โวลท์ (Volt) ไหลผ่านความต้านทาน 1 โอห์ม (Ohm)

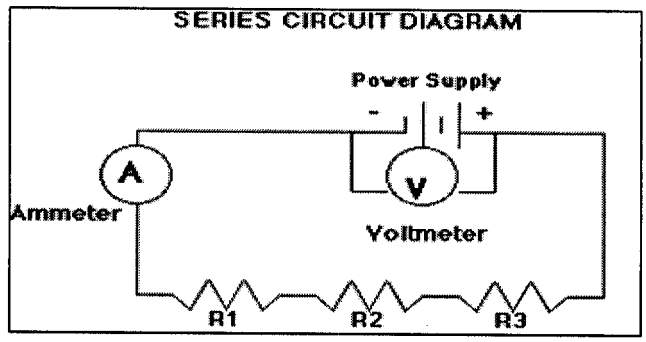
2.4.1.2 แรงเคลื่อนไฟฟ้า (Electrical Voltage) คือ แรงผลักดันที่ทำให้อิเล็กตรอนเคลื่อนตัวไป มีหน่วยเป็นโวลท์ (Volt) แรงเคลื่อน 1 โวลท์ หมายถึงแรงเคลื่อนไฟฟ้าที่ทำให้กระแสไฟฟ้า 1 แอมแปร์ไหลผ่านความต้านทาน 1 โอห์ม

2.4.1.3 ความต้านทาน (Resistance) คือ สิ่งที่ต้านการไหลของอิเล็กตรอน หรือกระแสไฟฟ้าที่ไหลผ่านตัวนำไปในวงจรไฟฟ้า มีหน่วยเป็นโอห์ม ถ้าความต้านทานมีค่ามากกระแสไฟฟ้าจะไหลผ่านได้น้อย แต่ถ้าความต้านทานมีค่าน้อยกระแสไฟฟ้าก็จะไหลผ่านได้มาก

จากความหมายของนิยามข้างต้นอาจจะทำให้เกิดความไม่เข้าใจว่าแรงเคลื่อนไฟฟ้า หรือ Voltage ก็คือ ความพยายามที่จะทำให้เกิดกระแสไฟฟ้าไหลผ่านเข้าไปในวงจรได้อย่างสมบูรณ์ แต่ก็เป็นไปได้ที่จะมีแรงเคลื่อนไฟฟ้า โดยที่ไม่มีกระแสเพราะอุปกรณ์บางอย่าง เช่น แบตเตอรี่หรือเซลล์ไฟฟ้าจะมีแรงเคลื่อน แต่วงจรไม่ได้ต่ออย่างสมบูรณ์จึงไม่มีกระแส

ส่วนกระแส หรือ Current นั้น คืออัตราการไหลของประจุผ่านอุปกรณ์ ซึ่งการไหลของกระแสก็เหมือนกับการปล่อยน้ำไหลไปตามท่อจนกลับมาบรรจบกันที่จุดเริ่มต้น วงจรไฟฟ้า สามารถแบ่งได้เป็น 3 รูปแบบ คือ วงจรอนุกรม วงจรขนาน และ วงจรผสม

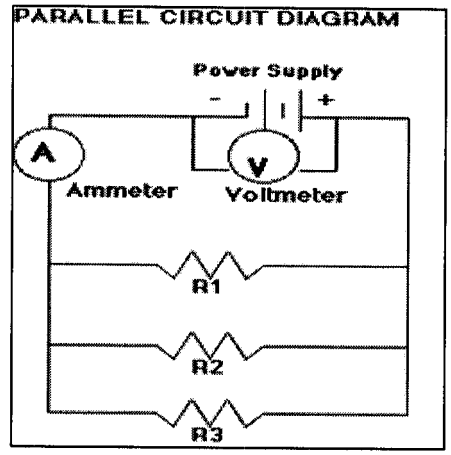
2.4.1.4 วงจรอนุกรม (Series Circuit) เป็นการต่ออุปกรณ์ทีละตัวในลักษณะโยงกันไปเรื่อย ๆ เหมือนการต่อเชือกแล้วมาบรรจบกับจุดเริ่มต้นเป็นวงกลม ดังภาพที่ 32



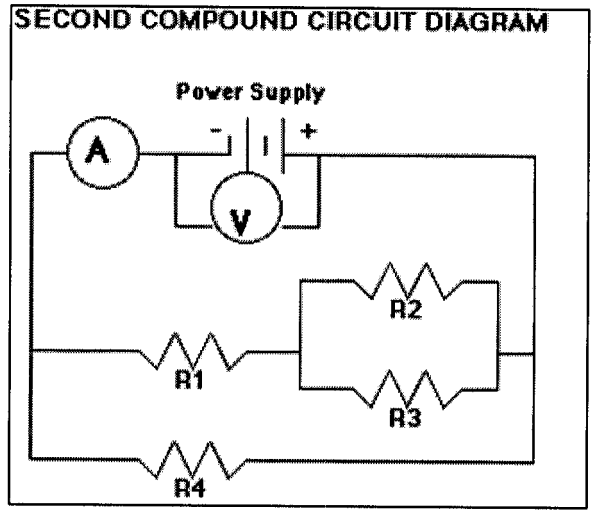
ภาพที่ 32 การต่อวงจรแบบอนุกรม

2.4.1.5 วงจรขนาน (Parallel Circuit) เป็นการต่ออุปกรณ์ในลักษณะคร่อมกัน โดยมีจุดเริ่มต้นและจุดสุดท้ายของอุปกรณ์แต่ละชนิดจะต่อร่วมกัน ดังภาพที่ 33

2.4.1.6 วงจรผสม (Compound Circuit) เป็นการต่อวงจรที่มีการต่อทั้งแบบอนุกรม และแบบขนานในวงจรเดียวกัน ดังภาพที่ 34



ภาพที่ 33 การต่อวงจรแบบขนาน



ภาพที่ 34 การต่อวงจรแบบผสม

2.4.2 กฎของโอห์ม (Ohm's Law)

เป็นกฎแรกที่เราจะต้องเรียนรู้เกี่ยวกับไฟฟ้า โดยที่กฎจะบอกถึงความสัมพันธ์ของ 3 สิ่งต่อไปนี้

2.4.2.1 กระแสไฟฟ้า (Electrical Current) ใช้สัญลักษณ์ I มีหน่วยเป็นแอมแปร์ (Amperes : A)

2.4.2.2 แรงเคลื่อนไฟฟ้า (Electrical Voltage) ใช้สัญลักษณ์ V มีหน่วยเป็นโวลท์ (Volt : V)

2.4.2.3 ความต้านทาน (Resistance) ใช้สัญลักษณ์ R มีหน่วยเป็นโอห์ม (Ohm : Ω)

ความสัมพันธ์สามารถเขียนเป็นสมการทางคณิตศาสตร์ง่าย ๆ คือ กระแสไฟฟ้าจะเท่ากับค่าความต่างศักย์ (แรงเคลื่อนไฟฟ้า) ระหว่างวงจรหารด้วยความต้านทาน ดังนี้

กระแสไฟฟ้า = แรงเคลื่อนไฟฟ้า / ความต้านทาน

ในการคำนวณเราต้องทราบค่าของปริมาณไฟฟ้าเพียง 2 ใน 3 ปริมาณ เช่น ทราบแรงเคลื่อนไฟฟ้า และกระแสไฟฟ้า ปริมาณที่ 3 คือ ความต้านทานก็สามารถหาค่าได้ โดยอาศัยกฎของโอห์มโดยกฎของโอห์มกล่าวไว้ โดยสามารถเขียนเป็นสมการได้ว่า

$$I = E / R \qquad \text{หรือ}$$

$$E = I \times R \qquad \text{หรือ}$$

$$R = E / I$$

2.4.2.4 กำลังไฟฟ้า (Electrical Power)

(1) กำลังไฟฟ้า คือ อัตราของกำลังเพื่อออกแรงให้วัตถุเคลื่อนที่ไปหรือแรงเคลื่อนที่ไปตามวงจร มีหน่วยเป็นวัตต์ (Watt)

(2) แรงเคลื่อนหรือความต่างศักย์ (ในทางทฤษฎีทางไฟฟ้า) หมายถึง การใช้งาน 1 จูล ในการเคลื่อนย้ายประจุ 1 คูลอมบ์และกระแสหมายถึง อัตราการไหลของประจุ 1 คูลอมบ์ในเวลา 1 วินาที โดยกำลังในทางไฟฟ้าสามารถเขียนเป็นสมการ ได้ดังนี้

กำลัง 1 วัตต์ = แรงเคลื่อน 1 โวลท์ × กระแส 1 แอมแปร์

หรือ เขียนเป็นสมการได้ว่า

$P = E \times I$

โดยที่ P คือ กำลังไฟฟ้า (Power) มีหน่วยเป็นวัตต์

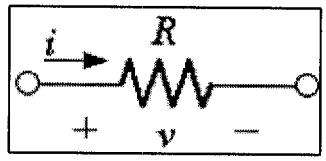
E คือ แรงเคลื่อนไฟฟ้า มีหน่วยเป็นโวลท์

I คือ กระแสไฟฟ้า มีหน่วยเป็นแอมแปร์

2.4.3 อุปกรณ์อิเล็กทรอนิกส์เบื้องต้น

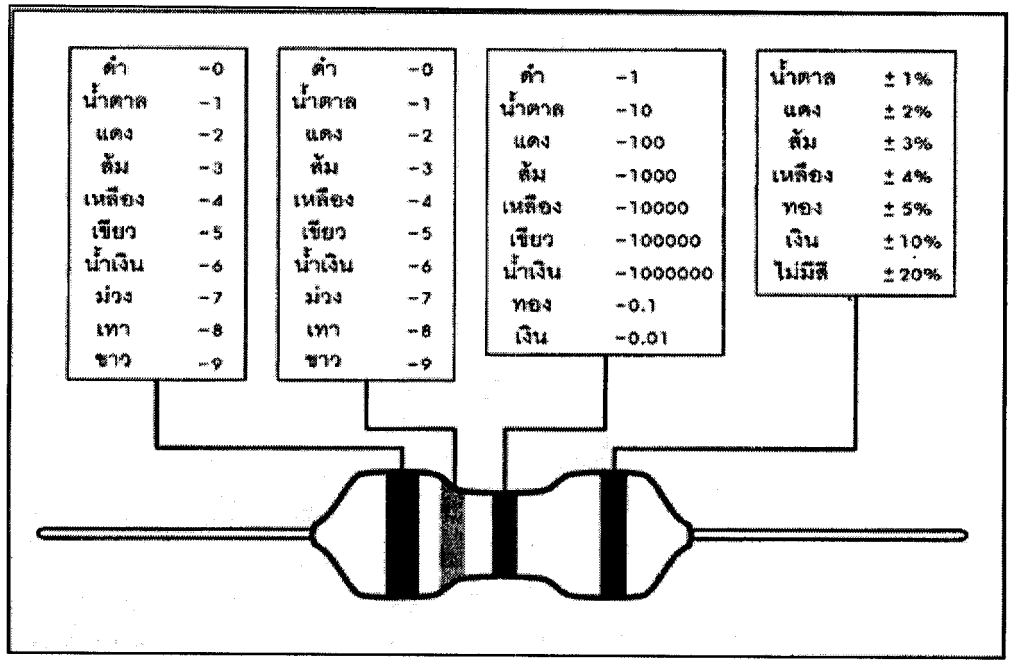
ในการทำงานเกี่ยวกับอิเล็กทรอนิกส์และไฟฟ้านั้น เราจำเป็นต้องเรียนรู้เกี่ยวกับอุปกรณ์อิเล็กทรอนิกส์ที่จำเป็นแต่ละชนิด ดังรายละเอียดต่อไปนี้

2.4.3.1 ตัวต้านทาน (Resistors) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่มีคุณสมบัติในการต้านการไหลของกระแสไฟฟ้ามีหน่วยเป็นโอห์ม ถ้าตัวต้านทานมีค่ามากกระแสไฟฟ้าจะไหลผ่านได้น้อย แต่ถ้าตัวต้านทานมีค่าน้อยกระแสไฟฟ้าจะไหลผ่านได้มาก ดังภาพที่ 35



ภาพที่ 35 สัญลักษณ์ของตัวต้านทาน

ถ้าตัวต้านทานมีค่ามาก (มีค่า Resistance สูง) ก็จะทำให้กระแสไหลผ่านได้น้อยลง ซึ่งต้องออกแบบให้กระแสที่ไหลผ่านอุปกรณ์และวงจรมีค่าที่เหมาะสมเพราะอุปกรณ์บางอย่างบอบบาง ถ้าได้รับกระแสมากเกินไปจะทำให้อุปกรณ์หรือวงจรเสียหายได้ ตัวต้านทานนั้นมีให้เลือกอยู่หลายค่า โดยสามารถอ่านค่าแถบสีที่อยู่บนตัวต้านทานได้ว่ามีค่าเท่าใดเพื่อการเลือกใช้ที่เหมาะสม ซึ่งแถบสีนั้นจะมีอยู่ 4 แถบ ซึ่งจะต้องอ่านค่าของสีแต่ละค่าแล้วนำมาคำนวณค่าความต้านทาน ดังแสดงในภาพที่ 36



ภาพที่ 36 แถบสีบนตัวต้านทาน

แถบที่ 1 คือ หลักที่หนึ่ง

แถบที่ 2 คือ หลักที่สอง

แถบที่ 3 คือ จำนวนเลข 0 ที่ต่อท้าย (แต่ถ้าเป็นสีเงิน คูณ 0.01 และสีทอง คูณด้วย 0.1)

แถบที่ 4 คือ ค่าความคลาดเคลื่อน ถ้าเป็นสีเงินจะมีค่าความคลาดเคลื่อน 10% สีทองจะมีค่าความคลาดเคลื่อน 5% สีแดงมีค่าความคลาดเคลื่อน 2%

ตารางที่ 9 รหัสสี และค่าที่อ่านบนตัวต้านทาน

สี	ค่าที่อ่าน
ดำ	0
น้ำตาล	1
แดง	2
ส้ม	3
เหลือง	4
เขียว	5
น้ำเงิน	6
ม่วง	7
เทา	8
ขาว	9

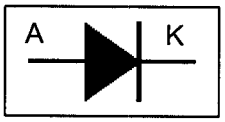
(1) ส้ม ม่วง เหลือง ทอง ค่าความต้านทานคำนวณจาก ส้ม (3) ม่วง (7) เหลือง (0 อีก 4 ตัว) และสีทองคือ ค่าความคลาดเคลื่อน 10% ดังนั้น ค่าความต้านทานจะมีค่าเท่ากับ 370000 หรือ 370 กิโลโอห์ม (kilo Ohm)

(2) น้ำตาล ม่วง เขียว เงิน ค่าความต้านทานคำนวณจาก น้ำตาล (1) ม่วง (7) เขียว (0 อีก 5 ตัว) และสีเงินคือ ค่าความคลาดเคลื่อน 5% ดังนั้นค่าความต้านทานจะมีค่าเท่ากับ 1700000 หรือ 1.7 เมกกะโอห์ม (Mega Ohm)

(3) เขียว แดง ทอง (ไม่มีแถบที่ 4) ค่าความต้านทานคำนวณจาก เขียว (5) แดง (2) ทอง (คูณด้วย 0.1) ดังนั้น ค่าความต้านทานจะมีค่า $52 \times 0.1 = 5.2$ โอห์ม

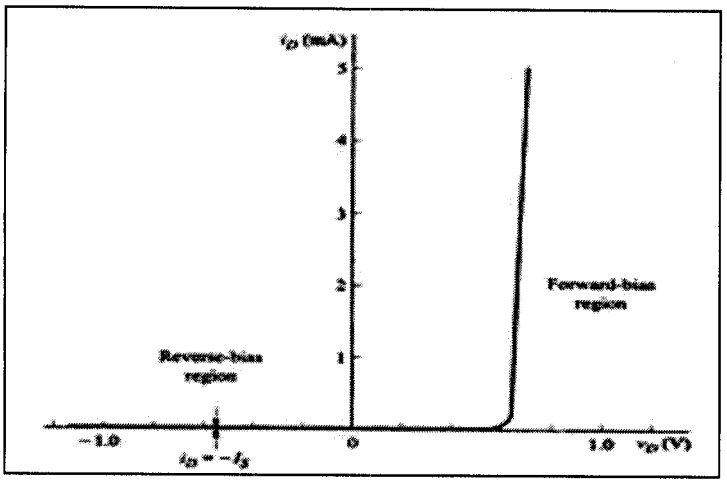
2.4.3.2 ไดโอด (Diode)

ไดโอดเป็นอุปกรณ์อิเล็กทรอนิกส์ที่ยอมให้กระแสไหลผ่านได้เพียงทิศทางเดียว (จาก A ไป K) ซึ่งเรามักจะใช้งานไดโอดในการกำหนดทิศทางการไหลของกระแสไฟฟ้า กล่าวคือถ้าเราทำการป้อนแรงดันที่ขั้วตรงกับขั้วของไดโอด หรือการไบอัสตรง กระแสไฟฟ้าจะสามารถไหลผ่านตัวมันได้ แต่ถ้าแรงดันมีขั้วไม่ตรงกันหรือการไบอัสกลับ กระแสไฟฟ้าก็จะไหลผ่านตัวมันไม่ได้ สำหรับสัญลักษณ์ของไดโอด ดังแสดงในภาพที่ 37



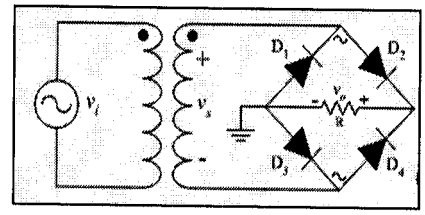
ภาพที่ 37 สัญลักษณ์ไดโอด

การกำหนดแรงเคลื่อนไฟฟ้าระดับหนึ่ง เพื่อให้กระแสไหลผ่านไดโอดได้ เช่น ไดโอดที่ทำจากซิลิคอนจะต้องใช้แรงเคลื่อน 0.7 โวลท์ในทิศทางที่ถูกต้องก็จะเกิดกระแสไฟผ่านไดโอดได้ ขณะเดียวกันถ้ากลับทิศทางของแรงเคลื่อนไฟฟ้าก็จะไม่มีกระแสไหลผ่าน ยกเว้นจะมีแรงเคลื่อนไฟฟ้าที่มากพอ ซึ่งเป็นระดับที่ไม่สามารถต้านทานได้โดยเรียกว่า Breakdown ดังแสดงในภาพที่ 38



ภาพที่ 38 กราฟแสดงพฤติกรรมของไดโอด

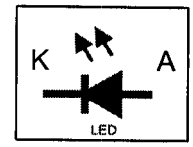
ส่วนใหญ่จะเห็นไดโอดในวงจรเร็กติไฟเออร์ (Rectifier Circuit) หรือวงจรแปลงแรงดันไฟฟ้ากระแสสลับให้เป็นกระแสตรง ส่วนมากจะใช้เป็นแบบ Bridge Rectifier ดังภาพที่ 39



ภาพที่ 39 วงจร Bridge Rectifier

2.4.3.3 LED (Light Emitting Diode)

LED เป็นไดโอดชนิดหนึ่ง ซึ่งสามารถเปล่งแสงออกมาได้เมื่อมีการป้อนแรงดันที่มีขั้วถูกต้องให้กับตัวอุปกรณ์ ส่วนมากจะนำไปใช้สำหรับแสดงผลของวงจร สำหรับสัญลักษณ์ของ LED แสดงดังภาพที่ 40

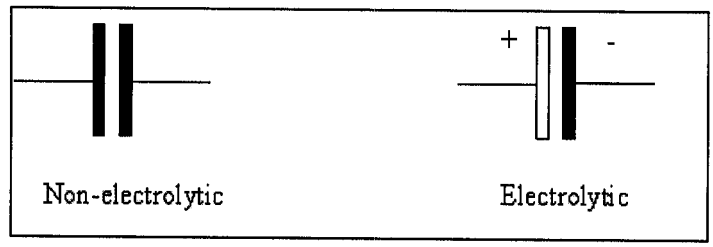


ภาพที่ 40 สัญลักษณ์ของ LED

สำหรับ LED นั้นมีอยู่หลายสีให้เลือกใช้งานทั้งแดง ส้ม เขียว น้ำเงิน ขาว ซึ่งสีที่ได้เกิดจากวัสดุสารกึ่งตัวนำที่นำมาผลิต โดยสีน้ำเงิน และสีขาวจะมีราคาแพงที่สุด ขณะเดียวกันก็มี LED ที่มีสามสีอยู่ในหลอดเดียวกันได้ด้วย เช่น มีสีเขียว และสีแดงอยู่ในหลอดเดียวกัน ส่วนสีที่สามเกิดจากการผสมระหว่างเขียว และแดงออกมาเป็นเหลือง เป็นต้น ซึ่งเราเรียกว่า Tri-Color LED

2.4.3.4 ตัวเก็บประจุ (Capacitors)

ตัวเก็บประจุ หรือ Capacitor ในวงจรอิเล็กทรอนิกส์สามารถนำตัวประจุไปใช้ประโยชน์ได้หลาย ๆ หน้าที่ เช่น การกรองความถี่ เชื่อมโยงสัญญาณและปรับแรงดันไฟฟ้าให้เรียบ เป็นต้นสำหรับสัญลักษณ์ของตัวเก็บประจุ ดังแสดงในภาพที่ 41



ภาพที่ 41 สัญลักษณ์ตัวเก็บประจุ

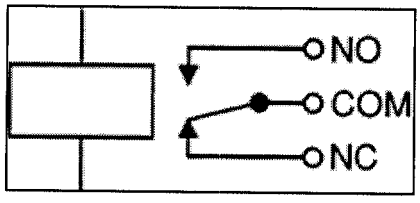
โดยปกติค่าความจุจะเป็นฟารัด (Farad : F) โดยค่าที่ใช้งานส่วนมากจะอยู่ในระดับไมโครฟารัด (1×10^{-6} F) และพิโคฟารัด (1×10^{-12} F) ซึ่งเราสามารถอ่านค่าได้จากตัวถังของตัวเก็บประจุ

2.4.3.5 สวิทซ์ (Switch)

สวิทซ์ใช้เป็นตัวตัดต่อกระแสไฟฟ้าที่ป้อนให้แก่วงจรในการเลือกใช้สวิทซ์ต้องคำนึงถึงกระแสไฟฟ้าที่ไหลผ่านเช่น 5 Amperes ดังนั้น สวิทซ์ที่จะนำมาใช้ต้องสามารถทนกระแสได้ไม่น้อยกว่า 5 Amperes

2.4.3.6 รีเลย์ (Relay)

รีเลย์เป็นอุปกรณ์ที่ใช้ในการตัดต่อกระแสไฟฟ้าเหมือนกับสวิทซ์ แต่ลักษณะการใช้งานจะต่างกัน คือ รีเลย์จะทำหน้าที่ตัดหรือต่อวงจรก็ต่อเมื่อมีการป้อนแรงดันไฟฟ้าเข้าไปในขดลวดรีเลย์ ซึ่งในการเลือกใช้งานรีเลย์นั้นจะต้องคำนึงถึงความสามารถในการทนกระแสของหน้าสัมผัสรีเลย์เป็นหลักเหมือนกับการเลือกใช้งานสวิทซ์ นอกจากนี้ยังต้องคำนึงถึงแรงดันที่จะป้อนให้กับขดลวดเพื่อให้รีเลย์ทำงานด้วย สำหรับสัญลักษณ์ของรีเลย์ ดังแสดงในภาพที่ 42



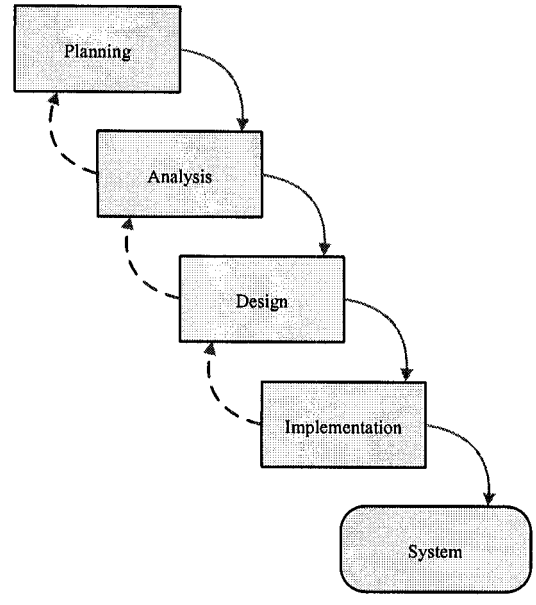
ภาพที่ 42 สัญลักษณ์รีเลย์

บทที่ 3
การวิเคราะห์และการออกแบบ

การดำเนินการค้นคว้าอิสระในครั้งนี้ ผู้ดำเนินการค้นคว้าอิสระได้ทำการสร้างฮาร์ดแวร์และโปรแกรมที่ใช้ในตรวจจับการทำงานของอุปกรณ์ไฟฟ้ากำลังแล้วเปลี่ยนเป็นสัญญาณอนุกรมส่งไปยังห้องควบคุมและยังได้สร้างโปรแกรมติดต่อกับผู้ใช้งานแบบ Graphic User Interface โดยใช้วิธีในการพัฒนาได้ดำเนินการตามวงจรการพัฒนาระบบ (System Development Life Cycle: SDLC) ซึ่งวิธีการ SDLC มีหลากหลายวิธีการ ในงานวิจัยนี้ ได้เลือกใช้วิธี Structured System Analysis and Design Methodology (SSADM) เป็นส่วนหนึ่งในวิธีการของ SDLC [2]

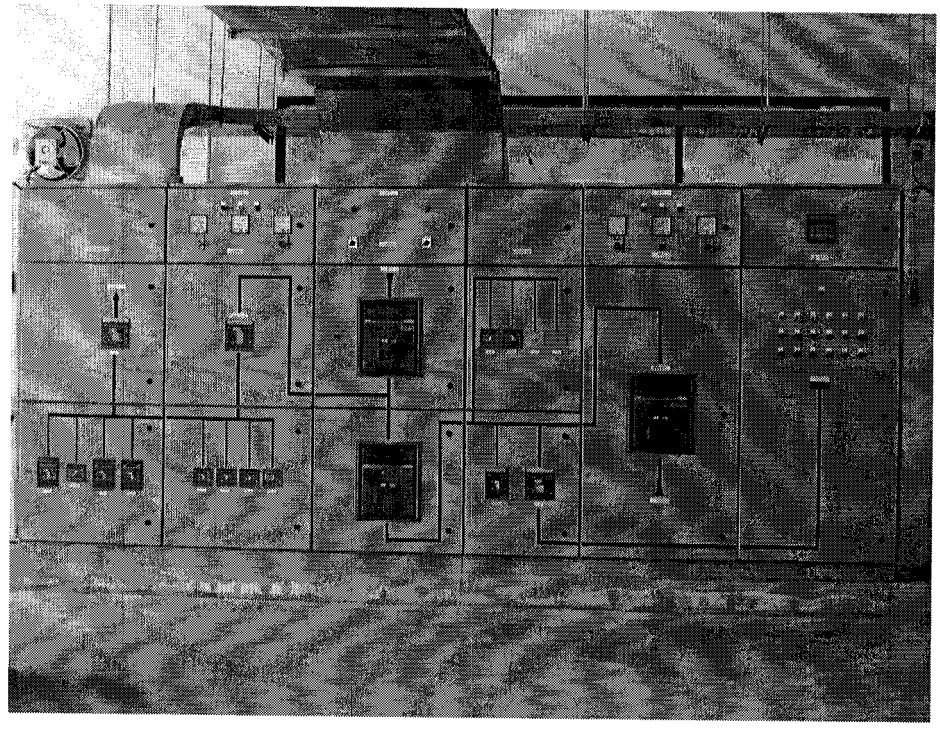
Structured System Analysis and Design Methodology (SSADM) เป็นลำดับขั้นจากขั้นตอนหนึ่งไปสู่ขั้นตอนต่อไป และมีการใช้แบบจำลองที่เป็นแผนภาพ เพื่ออธิบายขั้นตอนการทำงานและข้อมูลทั้งหมดของโปรแกรมที่จะพัฒนา

ในบทนี้ ผู้วิจัยได้ทำในส่วนขั้นตอนการวิเคราะห์และออกแบบตามวิธีการของ SSADM

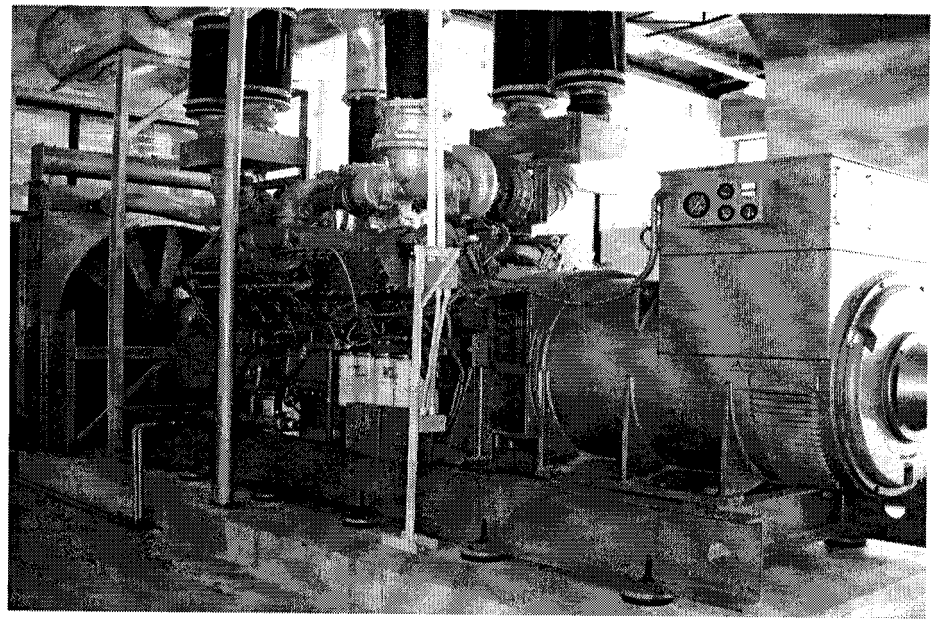


ภาพที่ 43 Structured System Analysis and Design Methodology (SSADM) ที่ใช้ใน SDLC แบบ Waterfall

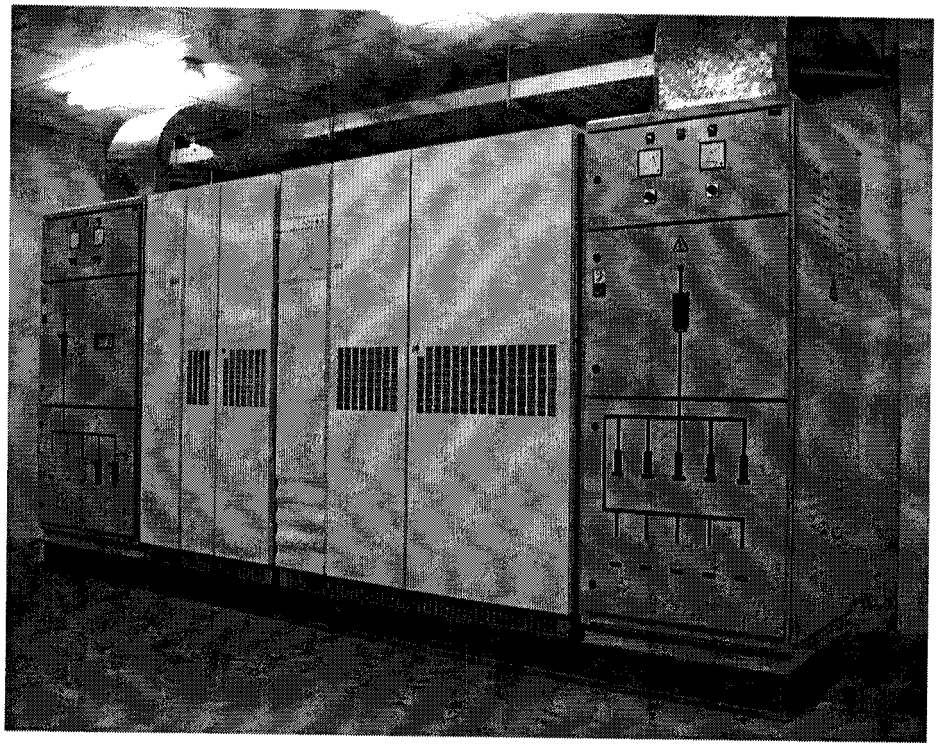
3.1 วิเคราะห์ระบบ



ภาพที่ 44 ระบบถ่ายโหลดไฟฟ้าของสถานีดาวเทียมสิรินธร

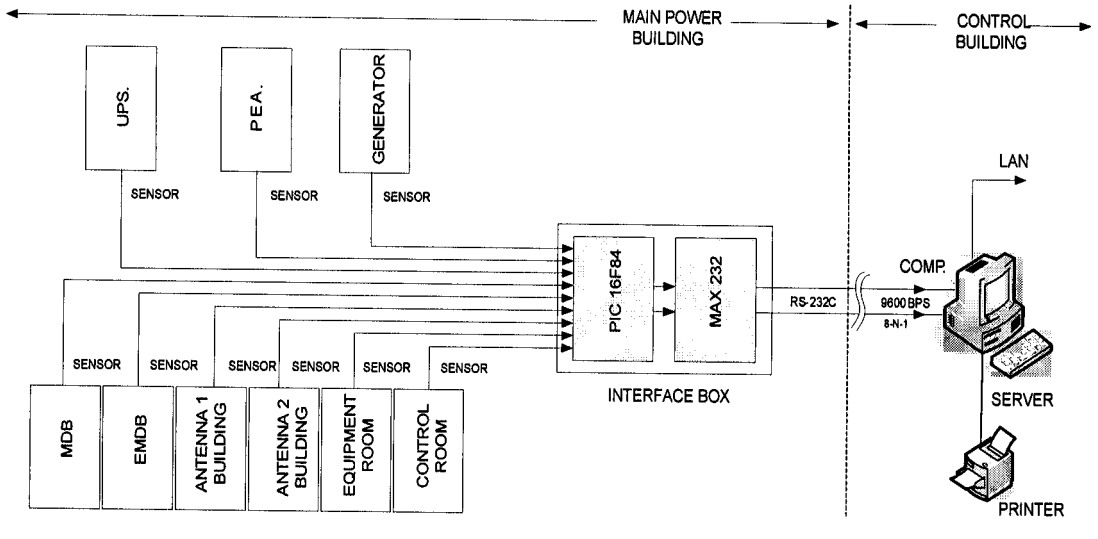


ภาพที่ 45 เครื่องกำเนิดไฟฟ้าขนาด 1250 KVA



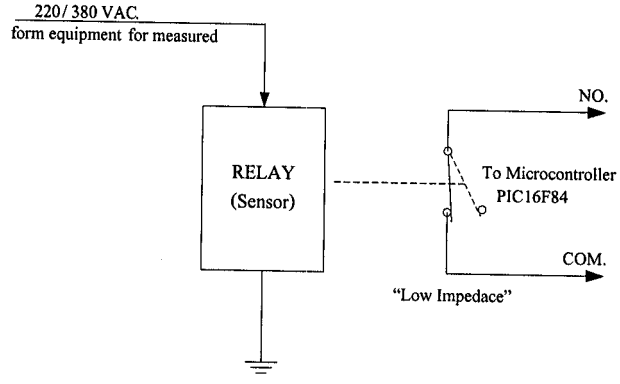
ภาพที่ 46 UPS. ขนาด 280 KVA

จากภาพที่ 44 ถึง 46 นั้น จะเป็นอุปกรณ์ของระบบไฟฟ้ากำลังที่ต้องการจะตรวจสอบ โดยจะเลือกตรวจสอบอุปกรณ์ที่สำคัญๆ ว่าอุปกรณ์เหล่านั้นทำงานเป็นปกติหรือไม่ และได้วิเคราะห์ออกมาเขียนเป็นแผนภาพบล็อก (Block diagram) ได้ดังแสดงในภาพที่ 47



ภาพที่ 47 แผนภาพบล็อก ระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง

การจะตรวจสอบว่าอุปกรณ์ไฟฟ้ากำลังที่จะตรวจสอบว่าปกติหรือไม่นั้น จะใช้วิธีการตรวจสอบแรงดัน (Voltage) ของอุปกรณ์เหล่านั้น โดยใช้ Relay เป็นตัวตรวจสอบ (Sensor) ดังแสดงในภาพที่ 48

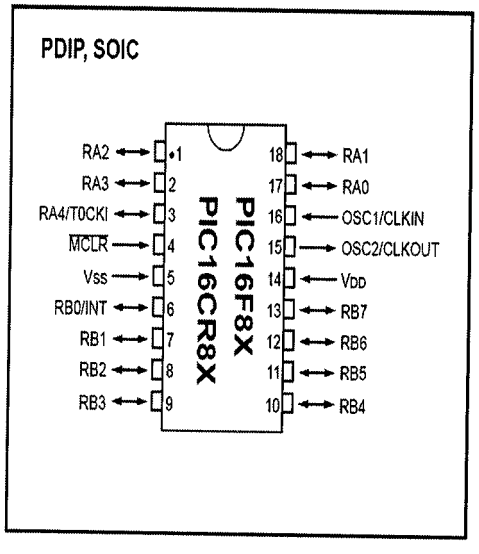


ภาพที่ 48 การใช้ Relay ในการตรวจสอบแรงดัน (Sensor)

นำผลที่ได้จากการตรวจสอบโดยใช้ Relay ซึ่งเป็นหน้า Contract ส่งเข้าไปประมวลผลโดยใช้ ไมโครคอนโทรลเลอร์ เบอร์ PIC16F84 ซึ่งเป็นของบริษัท Microship [6] เป็นตัวประมวลผลโดยการตรวจสอบ Relay ที่ใช้เป็นตัว Sensor อุปกรณ์ไฟฟ้ากำลังตามจุดต่าง ๆ ดูภาพที่ 52 วงจรกล่องอินเตอร์เฟส แ ละตารางที่ 10 ป ระกอบ โดยถ้าหาก Relay ที่ใช้ตรวจสอบนั้นมีแรงดันไฟฟ้าก็จะทำให้ขดลวดของ Relay มีอำนาจแม่เหล็ก ดูดหน้า Contract ของ Relay ขา COM. เข้ากับ ขา NO. ต่อถึงกัน ดังภาพที่ 48 พอร์ตของไมโครคอนโทรลเลอร์ซึ่งเป็นขาเดียวกันกับ Contract ของ Relay ก็จะเปลี่ยนจากสถานะ High Impedance (ตรรกะ 1) เป็นสถานะ Low Impedance (ตรรกะ 0) ไมโครคอนโทรลเลอร์ก็จะส่งข้อมูลค่าประจำพอร์ตออกไป เช่น ถ้าเป็นพอร์ต 1 ก็จะส่ง Character 01 ออกที่ขา Tx.Data(RA1) และถ้าเป็นพอร์ต 2 ก็จะส่ง Character 02 ออกไปที่ขา Tx.Data(RA1) ซึ่งเป็นขาที่ 18 เข้าไปยังโปรแกรมติดต่อกับผู้ใช้ในห้องควบคุมภายในอาคารสถานีดาวเทียม โปรแกรมติดต่อกับผู้ใช้ก็จะตรวจสอบข้อมูลที่รับเข้ามาหากตรวจสอบแล้ว พบว่า ค่าประจำพอร์ตที่ถูก Sensor ก็จะแสดงว่าอุปกรณ์ไฟฟ้ากำลังตัวที่ถูก Sensor อยู่นั้นมีสถานะปกติ แต่ในทางตรงกันข้าม หากตรวจสอบแล้วไม่พบค่าประจำพอร์ตที่ส่งเข้ามาก็แสดงว่าสถานะอุปกรณ์ไฟฟ้ากำลังตัวที่ Sensor อยู่นั้นมีปัญหาหรือผิดปกตินั้นเอง

ส่วนภาพที่ 49-50 ได้แสดงรายละเอียดของขาต่อใช้งานทั้งหมดของไมโครคอนโทรเลอร์ PIC16F84 ซึ่งมีพอร์ตใช้งานทั้งหมด 13 พอร์ตหรือ 13 ขา คือพอร์ต RA จำนวน 5 ขา และ พอร์ต RB จำนวน 8 ขา รวม 13 พอร์ต ตามที่ได้กล่าวมา สามารถกำหนดให้เป็นได้ซึ่งทั้งอินพุตพอร์ต หรือเอาต์พุตพอร์ตก็ได้

ส่วนตารางที่ 10 และ ภาพที่ 52 เป็นการกำหนด พอร์ต และ ตัว Sensor ในการทำงาน ไม่ว่าจะเป็นส่วนของอินพุต ก็คือส่วนที่ใช้รับสัญญาณจากตัว Sensor ที่ใช้ตรวจสอบการทำงานอุปกรณ์ไฟฟ้ากำลัง คือ ขา RB0-RB7, RA2-RA4, RA0 และส่วน เอาต์พุต ก็คือส่วนที่ใช้ส่งข้อมูลที่ได้จากการประมวลผลของไมโครคอนโทรลเลอร์ออกไปยังโปรแกรมติดต่อกับผู้ใช้ในห้องควบคุมอุปกรณ์ ภายในอาคารสถานีดาวเทียมตามที่ได้ออกแบบไว้ คือขาที่ 18 (RA1) ของไมโครคอนโทรลเลอร์ ซึ่งเรียกว่าขา Tx.Data



ภาพที่ 49 การจัดขาของไมโครคอนโทรลเลอร์ PIC16F84

Pin Name	DIP No.	SOIC No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	I	ST/CMOS (3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
					PORTA is a bi-directional I/O port.
RA0	17	17	I/O	TTL	
RA1	18	18	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RA4/T0CKI	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	I/O	TTL/ST (1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	Interrupt on change pin.
RB5	11	11	I/O	TTL	Interrupt on change pin.
RB6	12	12	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming clock.
RB7	13	13	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming data.
Vss	5	5	P	—	Ground reference for logic and I/O pins.
VDD	14	14	P	—	Positive supply for logic and I/O pins.

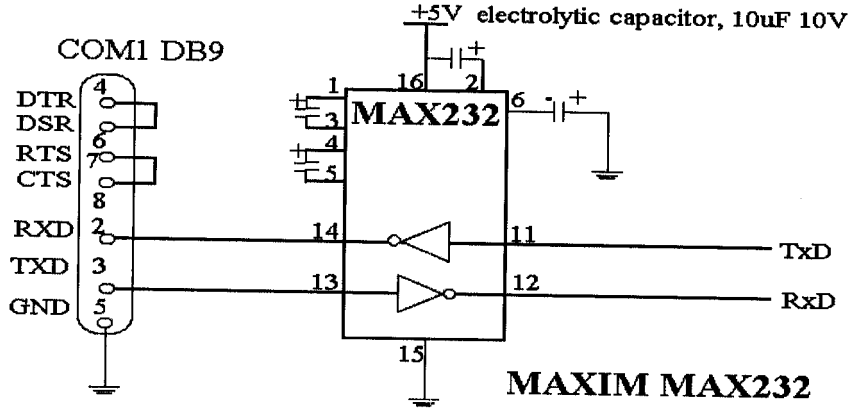
Legend: I= input O = output I/O = Input/Output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

ภาพที่ 50 รายละเอียดของขาต่อใช้งานทั้งหมดของ PIC16F84

จากภาพที่ 47 แสดงแผนภาพบล็อกของระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง ไมโครคอนโทรลเลอร์ PIC16F84 จะคอยเฝ้าตรวจสอบอุปกรณ์ไฟฟ้ากำลังอยู่ตลอดว่าอุปกรณ์เหล่านั้นมีสถานะเป็นอย่างไร ปกติหรือไม่ จากนั้นไมโครคอนโทรลเลอร์ ก็จะทำการแปลง (Convert) ข้อมูลที่ตรวจสอบได้ ซึ่งเป็นข้อมูลแบบขนาน (Parallel) ให้เป็นข้อมูลแบบอนุกรม (Serial) แต่ระดับสัญญาณที่ได้ยังต่ำอยู่ซึ่งอยู่ในระดับสัญญาณ

TTL. (0 โวลต์ กับ +5 โวลต์) ดังนั้น จึงต้องปรับระดับของสัญญาณให้สูงขึ้นอีก เพื่อที่จะส่งสัญญาณที่ตรวจจับได้ จากตัว Sensor ส่งผ่านสายโทรศัพท์ ไปยังอีกห้องควบคุม (Control Room) ซึ่งอยู่ห่างกันมีระยะทางประมาณ 60-80 เมตร ได้อย่างไม่มีปัญหา ดังนั้น จึงต้องปรับระดับสัญญาณที่ได้จากไมโครคอนโทรลเลอร์จากระดับสัญญาณ TTL. ให้เป็นระดับสัญญาณ RS-232C (±15 โวลต์) [4] โดยจะใช้ IC. MAX232 ของบริษัท Texas instruments [7] ในการแปลงสัญญาณ ดังแสดงในภาพที่ 51



ภาพที่ 51 MAX232 ในการแปลงระดับสัญญาณ จาก TTL. เป็น RS-232C

3.2 การออกแบบระบบ

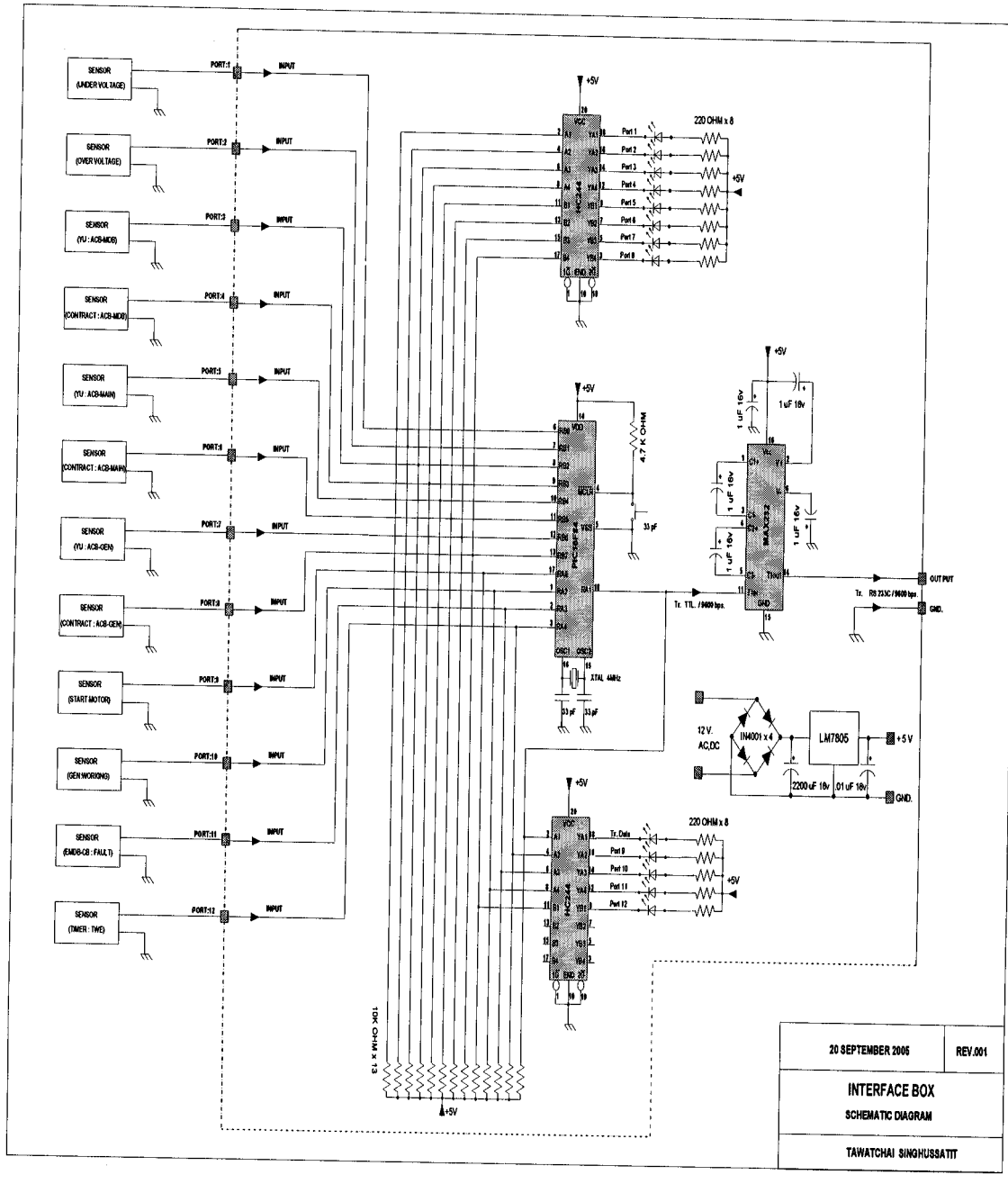
การออกแบบระบบจะแบ่งเป็น 2 ส่วนคือ ส่วนกล่องอินเตอร์เฟส และ ส่วนโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE)

3.2.1 การออกแบบกล่องอินเตอร์เฟส

3.2.1.1 ออกแบบฮาร์ดแวร์กล่องอินเตอร์เฟส โดยเป็นการสำรวจและกำหนดจุดที่จะติดตั้ง Sensor อุปกรณ์ไฟฟ้ากำลังที่จะทำการตรวจสอบ และการกำหนดพอร์ตใช้งานต่าง ๆ ของไมโครคอนโทรลเลอร์ รวมทั้งวัสดุอุปกรณ์ที่จะใช้ในการสร้างกล่องอินเตอร์เฟส โดยการเขียนเป็นตาราง และ เป็นวงจรออกมา ดังมีรายละเอียดตามตารางที่ 10 และภาพที่ 52

ตารางที่ 10 การกำหนด Port และ ตัว Sensor ในกล่องอินเตอร์เฟส

Port Sensor	Pin Name (PIC)	Pin No (PIC)	Equipment	หมายเหตุ
1	RB0	6	UUV-1	ตรวจสอบ Under Voltage
2	RB1	7	UUV-1	ตรวจสอบ Over Voltage
3	RB2	8	ACB-MDB	ตรวจสอบขดลวด YU ของ ACB-MDB
4	RB3	9	ACB-MDB	ตรวจสอบหน้าสัมผัส ACB-MDB
5	RB4	10	ACB-MAIN	ตรวจสอบขดลวด YU ของ ACB-MAIN
6	RB5	11	ACB-MAIN	ตรวจสอบหน้าสัมผัส ACB-MAIN
7	RB6	12	ACB-GEN	ตรวจสอบขดลวด YU ของ ACB-GEN
8	RB7	13	ACB-GEN	ตรวจสอบหน้าสัมผัส ACB-GEN
9	RA0	17	Starter Motor	ตรวจสอบ Starter Motor ของ Generator
10	RA2	1	Generator	ตรวจสอบการทำงานของ Generator
11	RA3	2	EMDB-CB	ตรวจสอบ EMDB-CB ปกติหรือไม่
12	RA4	3	TWE	ตรวจสอบ Timer ตั้งเวลาการทำงานของ Generator
13	RA1	18	Tx.(Data)	ใช้เป็นขาส่ง Data ไปยังห้องควบคุม

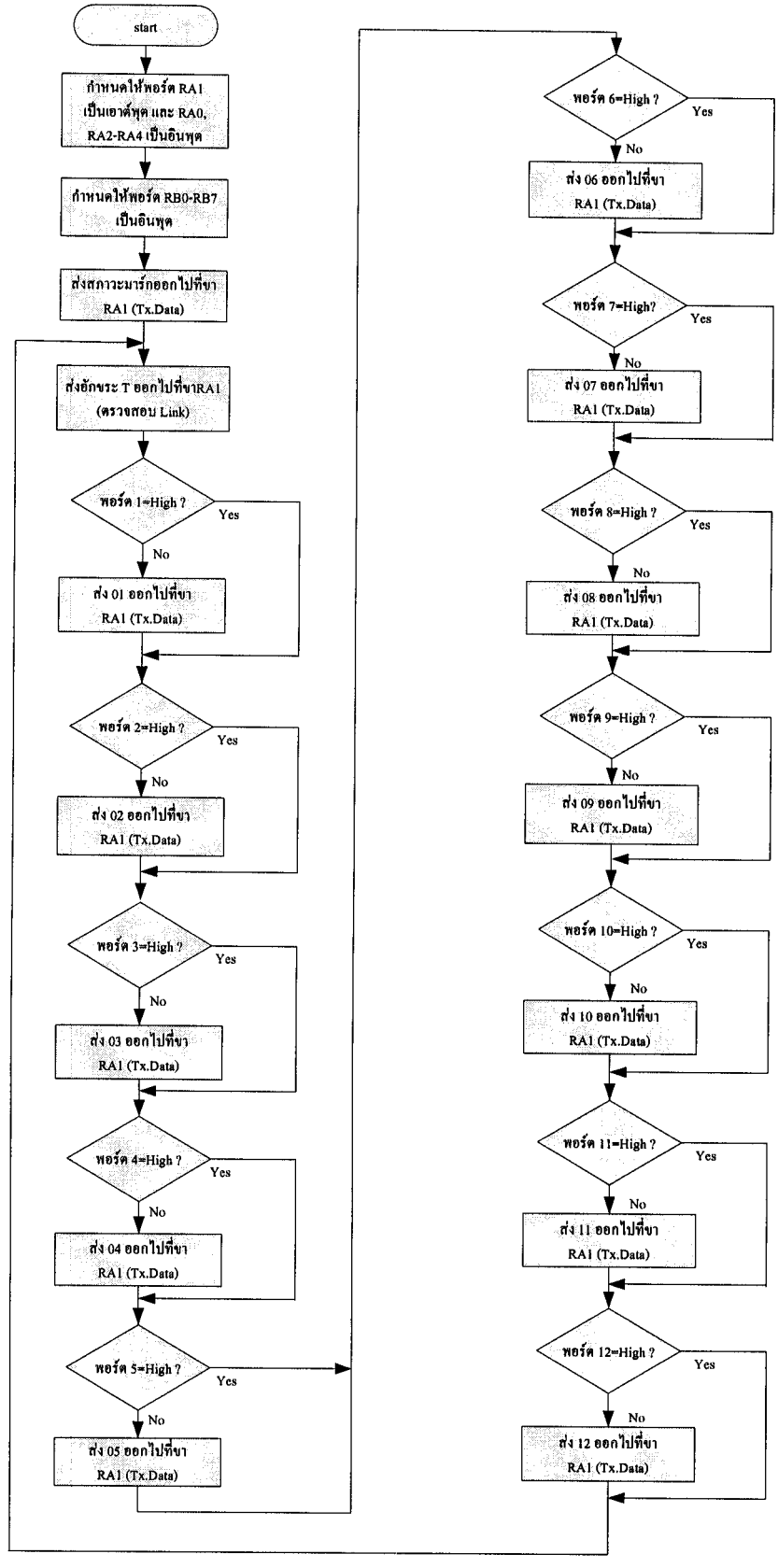


ภาพที่ 52 วงจรกล่องอินเตอร์เฟส

3.2.1.2 การออกแบบส่วนการเขียนโค้ดดิ่งโปรแกรม ของไมโครคอนโทลเลอร์ในกล่องอินเตอร์เฟส ซึ่งใช้ภาษาแอสเซมบลี ในการพัฒนาโดยมีลำดับขั้นตอนในการออกแบบใช้ผังงานโปรแกรม (Program Flowchart) แสดงการทำงานของแต่ละคำสั่งโดยละเอียด เป็นการแสดงลำดับตรรกะ (Logic) ขั้นตอน ของโปรแกรมหรือขั้นตอนการทำงานของโปรแกรม โดยใช้ภาพต่าง ๆ แสดงการทำงานในแต่ละขั้นตอนการใช้ผังงานซึ่งสามารถใช้ได้ง่ายและสามารถจะทำความเข้าใจได้ง่ายในการเขียนโปรแกรมอย่างมีโครงสร้าง (Structured Programming) จะใช้

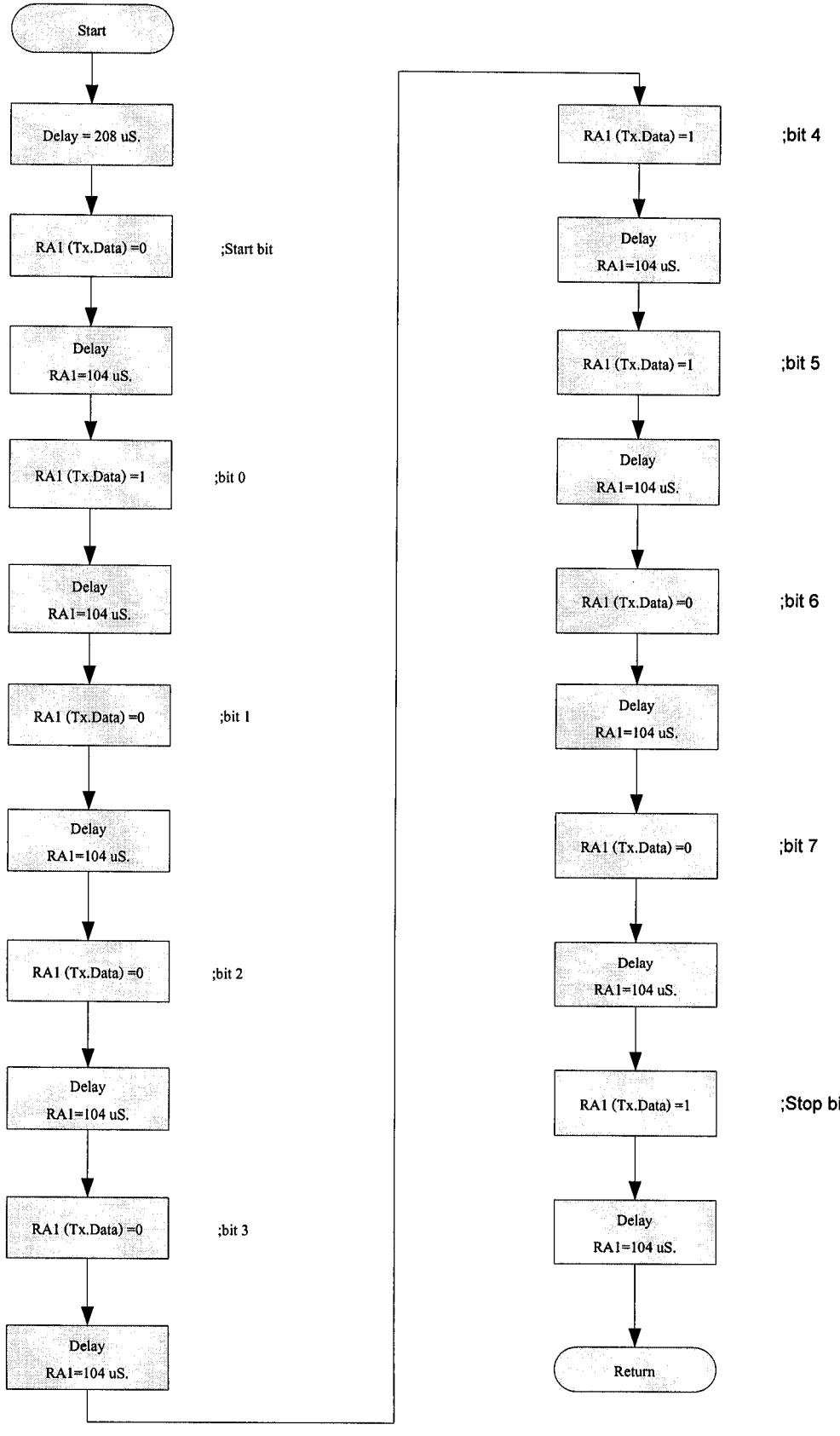
โครงสร้างพื้นฐานการเขียนโปรแกรมในรูปแบบตามลำดับ รูปแบบเลือก และรูปแบบการวนรอบ ดังผังโปรแกรมตามภาพที่ 53 เป็นแสดงผังงานการทำงานของกล่องอินเตอร์เฟส (โปรแกรมหลัก) ส่วนซอร์สโค้ดทั้งหมดนั้นจะอยู่ใน
ภาคผนวก ข

จากผังงานภาพที่ 53 เมื่อโปรแกรมเริ่มทำงานอันแรกเลยจะต้องไปกำหนดค่าเริ่มต้นหรืออินิเชียล (Initializing) พอร์ตก่อนโดยกำหนดให้พอร์ต A โดยขา RA1 กำหนดให้เป็นเอาต์พุทพอร์ต เพื่อใช้เป็นขาส่งข้อมูลไปยังโปรแกรมติดต่อกับผู้ใช้ในห้องควบคุม ส่วน พอร์ต A ที่เหลืออีก 4 พอร์ตกำหนดให้เป็น อินพุตพอร์ตเพื่อใช้ตรวจสอบตัว Sensor ที่ใช้ตรวจสอบการทำงานอุปกรณ์ไฟฟ้ากำลัง คือขา RA0 และขา RA2 ถึงขา RA4 เมื่อกำหนดค่าเริ่มต้นให้พอร์ต A เรียบร้อยแล้ว ต่อมาก็มากำหนดค่าเริ่มต้นให้กับพอร์ต B โดยกำหนดให้ขา RB0 ถึงขา RB7 รวม 8 ขา หรือ 8 พอร์ต เพื่อใช้ตรวจสอบตัว Sensor ที่ใช้ตรวจสอบการทำงานอุปกรณ์ไฟฟ้ากำลัง จากนั้นก็สั่งให้ขา 18 หรือขา RA1(Tx.Data) มีสภาวะเป็น MARK หรือตรรกะ 1 (Logic 1) คือ ในสภาวะไม่มีการส่งข้อมูลจะให้เป็นตรรกะ 1 จากนั้นส่งรหัสแอสกี้ “T” ออกไปที่ขา RA1(Tx.Data) เพื่อใช้ในการตรวจสอบการทำงานของกล่องอินเตอร์เฟส ต่อจากนั้นจะเริ่มตรวจสอบพอร์ตต่าง ๆ ตามลำดับ โดยเริ่มตรวจสอบที่พอร์ตที่ 1 ก่อนว่ามีสภาวะเป็น ตรรกะ 1 (Logic 1) หรือ ตรรกะ 0 (Logic 0) โดยถ้าหากตรวจสอบแล้วพบว่าเป็นเป็นตรรกะ 1 หรือสภวาวะ High ก็จะกระโดดข้ามคือไม่ส่งข้อมูลออกที่ขา RA1(Tx.Data) แล้วจะไปตรวจสอบพอร์ตต่อไป แต่ถ้าหากตรวจสอบแล้วพบว่าพอร์ตที่ 1 มีสภาวะเป็น ตรรกะ 0 (Logic 0) หรือสภาวะ Low ก็จะสั่งให้ส่งข้อมูลซึ่งเป็นค่าประจำพอร์ต 1 คือ รหัสแอสกี้ ” 0” และ
รหัสแอสกี้ “1” ออกที่ขา RA1 (Tx.Data) ส่วนพอร์ต 2 และพอร์ตอื่น ๆ ที่เหลือก็มีลักษณะการตรวจสอบเหมือนๆ กัน โดยจะตรวจสอบไล่ตามลำดับจากน้อยไปมาก จนถึงพอร์ตสุดท้ายคือพอร์ตที่ 12 ก็จะวนไปตรวจสอบพอร์ตที่ 1 ใหม่อีก



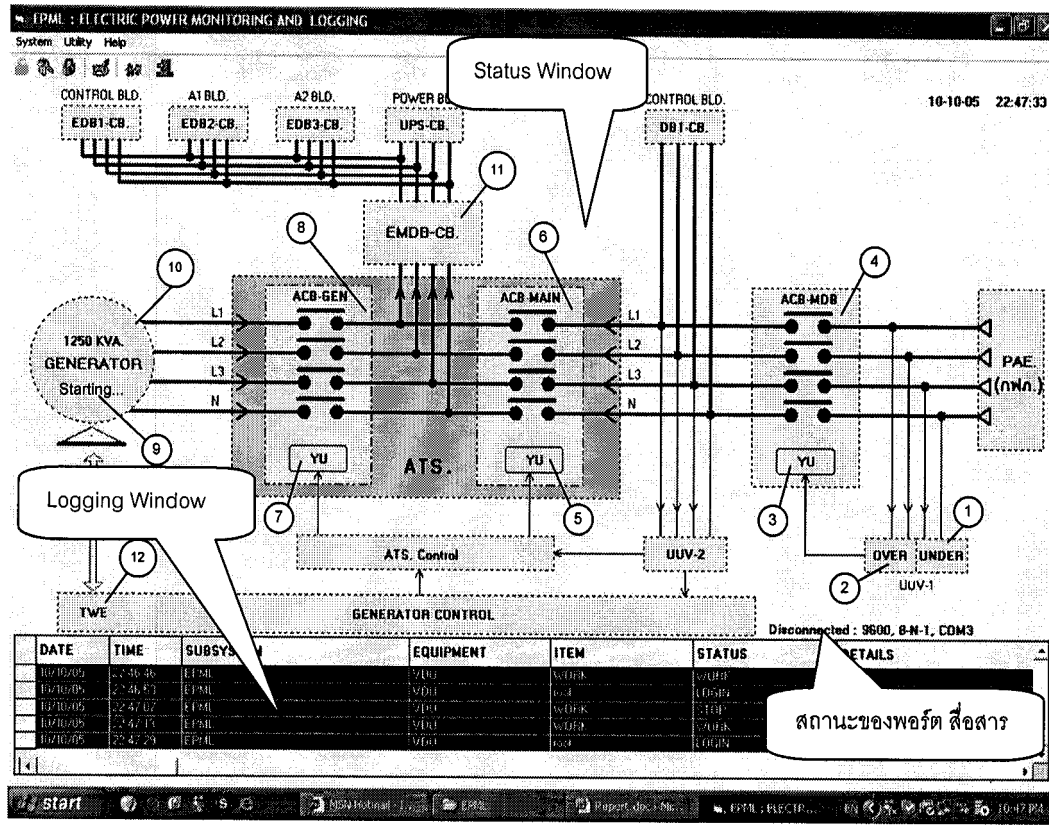
ภาพที่ 53 ผังงานการทำงานของกล่องอินเตอร์เฟส (โปรแกรมหลัก)

จากภาพที่ 54 เป็นผังงานการทำงานโปรแกรมย่อยในกล่องอินเตอร์เฟส เมื่อส่งอักษร "1" ออกไป จากผังงาน เมื่ออุปกรณ์ไฟฟ้า ทำงานตัว Sensor ก็จะตรวจสอบอุปกรณ์ไฟฟ้าเหล่านั้น และผลจากการตรวจสอบจะมีสภาวะเป็นตรรกะ 0 (Logic 0) หรือ ."Low" ในโปรแกรมหลัก ดังภาพที่ 53 จากโปรแกรมหลักก็จะกระโดดมาทำงานที่โปรแกรมย่อยในภาพที่ 54 โปรแกรมย่อยก็จะหน่วงเวลาประมาณ 208 ไมโครวินาที จากนั้นก็จะเริ่มส่งข้อมูลออกที่ขา18 (RA1) ซึ่งเป็นขา Tx.Data โดยส่งบิตเริ่มต้น(Start bit) ไปก่อนเพื่อบอกให้อุปกรณ์ปลายทางที่รับสัญญาณจากเราให้รู้ว่าเราจะเริ่มส่งข้อมูลแล้วนะ จากนั้นก็ส่งข้อมูลออกไปซึ่งเป็น 8 bits จากผังงานในภาพที่ 54 เป็นตัวอย่างการส่งรหัสแอสกี้ "1" ออกไปที่ขา RA1 (Tx.Data) ซึ่ง รหัสแอสกี้ "1" คือค่า 31 ในเลขฐานสิบหก หรือ 0011 0001 ในเลขฐานสอง ดังตาราง ASCII Code ที่ให้ไว้ในภาคผนวก ฉ การส่งข้อมูลนั้นจะส่งเป็นแบบเลขฐานสอง คือ 0 กับ 1 เท่านั้นโดยการส่ง จะส่งบิตที่ 0-1-2-3-4-5-6-7 ตามลำดับ คือส่งจากทางด้านขวามือไปซ้ายมือ จากรหัสแอสกี้ "1" คือ 00110001 ฐานสอง ก็จะเป็น 10001100 ฐานสอง เมื่อส่งในแต่ละบิตก็จะหน่วงเวลาเท่ากับ 104 ไมโครวินาที (ใช้ความเร็ว 9600 บิตต่อวินาที) 104 ไมโครวินาทีได้มาจาก เวลาที่ต้อง หารด้วยความเร็ว คือ 1 หารด้วย 9600 จะได้ 104 ไมโครวินาที เมื่อส่งรหัสแอสกี้เสร็จ คือส่งรหัสแอสกี้ที่เป็นเลขฐานสองครบ 8 บิตแล้ว ต่อมาก็จะต้องส่งบิตจบ (Stop bit) เพื่อบอกให้อุปกรณ์ปลายทางรู้ว่าเสร็จสิ้นการส่งอักขระ 1 ตัวแล้วนะ ต่อมาก็จะเจอคำสั่ง Return โปรแกรมก็จะกลับมา ที่โปรแกรมหลักแล้วทำงานที่คำสั่งต่อไปก่อนที่จะกระโดดไปยังโปรแกรมย่อย ส่วนโค้ดโปรแกรมไมโครคอนโทรลเลอร์ในกล่องอินเตอร์เฟสทั้งหมดนั้นจะอยู่ในภาคผนวก ข



ภาพที่ 54 ผังงานการทำงานโปรแกรมย่อยในกล่องอินเตอร์เฟส เมื่อส่งอักษร "1" ออกไป

3.2.2 การออกแบบโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE)



ภาพที่ 55 ตำแหน่ง Sensor ต่าง ๆ ที่ใช้ตรวจสอบอุปกรณ์ไฟฟ้ากำลัง

ส่วนโปรแกรมในห้องควบคุม (Control Room) ในภาพที่ 55 นั้น จะเขียนโปรแกรมอ่านค่าข้อมูลที่ถูกส่งมาจากกล่องอินเตอร์เฟสที่ใช้ตรวจสอบอุปกรณ์ไฟฟ้ากำลัง โดยใช้วิธีการตรวจสอบตัว Sensor ที่ได้ติดไว้ตามจุดต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลัง ซึ่งส่งเข้ามาเป็นระดับสัญญาณ RS-232C ความเร็ว 9600 bps. [4] โปรแกรมก็จะแปลงค่าของสถานะของอุปกรณ์เหล่านั้นจากการตรวจสอบของ Sensor และแสดงผลออกมาที่หน้าจอ เสมือนเป็นการทำงานของอุปกรณ์ไฟฟ้ากำลังจริง ๆ โดยจะออกแบบโปรแกรมติดต่อกับผู้ใช้ (GRAPHIC USER INTERFACE) เป็นแผนภาพบล็อก (Block diagram) ซึ่งจะประกอบด้วย 2 หน้าต่างหลัก ดังนี้

3.2.2.1 STATUS WINDOW เพื่อใช้แสดงสถานะการทำงานของอุปกรณ์

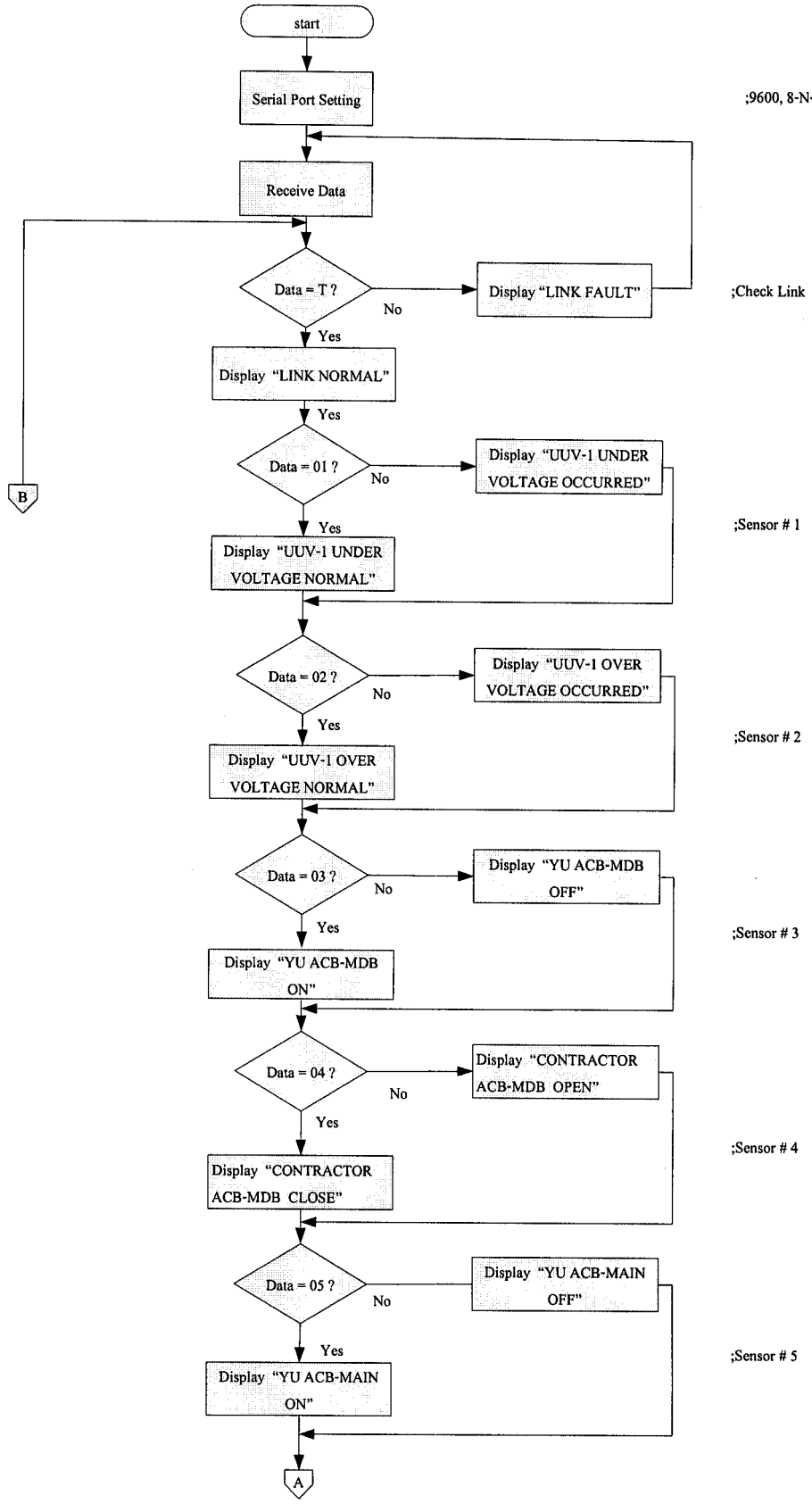
1) สีเขียว หมายถึง อุปกรณ์ปกติ

2) สีแดง หมายถึง อุปกรณ์ผิดปกติ

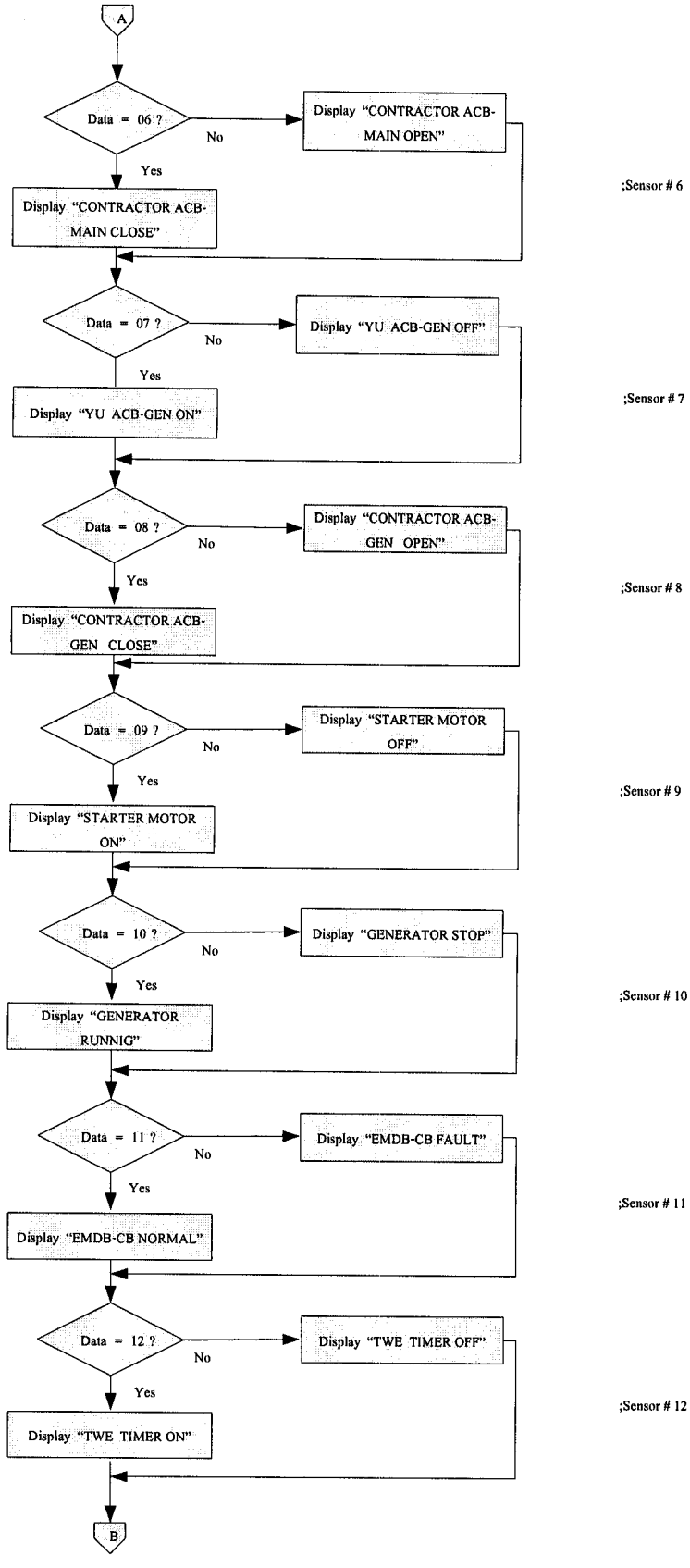
3) สีเทา หมายถึง อ่านข้อมูลไม่ได้หรือ Link ขาด

3.2.2.2 LOGGING WINDOW จะใช้บันทึกเหตุการณ์ สถานะการทำงานต่าง ๆ ที่เกิดขึ้นของอุปกรณ์ไฟฟ้ากำลัง โดยจะบันทึก วัน เวลา สถานะ ของอุปกรณ์นั้น ๆ ไว้ในแฟ้มข้อมูล

3.2.2.3 การออกแบบส่วนโปรแกรมโต้ดดิ่งของโปรแกรมติดต่อกับผู้ใช้งาน จากภาพที่ 56 และภาพที่ 57 เป็นการแสดงผังงานการทำงานโปรแกรมติดต่อกับผู้ใช้งาน ซึ่งใช้ภาษา Visual Basic 6 ในการพัฒนา ส่วนโค้ดโปรแกรมติดต่อกับผู้ใช้ (Graphic User Interface) นั้นอยู่ที่ภาคผนวก ค การทำงานของโปรแกรมเริ่มแรกจะต้องกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรมก่อนเช่น การกำหนดพอร์ตในการใช้งาน เลือกความเร็วเท่ากับ 9600 บิตต่อวินาที ข้อมูลใช้แบบ 8 บิต ไม่มีพาริตี้บิต ใช้บิตจบ 1 บิต ใช้ค่าเหล่านี้เพื่อให้ตรงกับค่าที่กล่องอินเตอร์เฟสซึ่งเป็นต้นทางส่งข้อมูลเข้ามา ถ้าหากค่าต่าง ๆ ที่กำหนดไม่ตรงกัน (คือที่กล่องอินเตอร์เฟส และคอมพิวเตอร์ที่ใช้กับโปรแกรมติดต่อกับผู้ใช้) จะไม่สามารถสื่อสารกันได้ข้อความที่รับได้นั้นจะอ่านไม่ออก จากนั้นโปรแกรมก็จะอ่านข้อมูลที่รับเข้ามาจาก Buffer จากนั้นก็จะตรวจสอบว่ามีรหัสแอสกี้ "T" หรือไม่ ถ้าไม่มีรหัส "T" จะหมายความว่า โปรแกรมติดต่อกับผู้ใช้ไม่สามารถติดต่อกับกล่องอินเตอร์เฟสได้ จะเป็นการบอกให้ทราบว่ากล่องอินเตอร์เฟสมีปัญหา หรืออาจจะเป็นที่สายเชื่อมต่อระหว่างกล่องอินเตอร์เฟสกับส่วนที่ติดต่อกับผู้ใช้ขาดก็ได้ ถ้าหากตรวจสอบแล้วไม่พบรหัสแอสกี้ "T" โปรแกรมติดต่อกับผู้ใช้ก็จะแสดงสถานะเป็น Link Fault มีข้อความเตือนพร้อมบันทึกเรคอร์ดเก็บค่าต่าง ๆ ที่สำคัญไว้ตรวจสอบในภายหลัง แล้วก็วนไปอ่านข้อมูลที่รับเข้ามาใหม่ จนกว่าจะพบรหัสแอสกี้ "T" ถึงจะเปลี่ยนสถานะเป็น Link Normal จากนั้นโปรแกรมก็จะทำการตรวจสอบพอร์ตและ Sensor ส่วน Sensor หมายเลขใดเข้าพอร์ตไหน และตรวจสอบอะไรนั้นตรวจสอบได้ที่ตารางที่ 10 และภาพที่ 52 (วงจรกล่องอินเตอร์เฟส) โปรแกรมจะตรวจสอบทุกพอร์ตเริ่มตั้งแต่พอร์ตที่ 1 จนถึงพอร์ตที่ 12 โดยเรียงลำดับจากน้อยไปหามาก โปรแกรมจะทำการตรวจสอบพอร์ต ถ้าตรวจสอบแล้วไม่เห็นค่าประจำพอร์ตนั้น ก็จะหมายความว่าอุปกรณ์ไฟฟ้ากำลัง ที่ Sensor ต่ออยู่กับพอร์ตนั้นตรวจสอบแล้วพบว่าอุปกรณ์ไฟฟ้ากำลังอุปกรณ์นั้นไม่ทำงานหรือผิดปกติ โปรแกรมก็จะแสดงสถานะบอกให้ทราบว่าพอร์ตนั้นไม่ปกติเช่นถ้าเป็นพอร์ต 1 จะเป็นพอร์ตใช้สำหรับตรวจสอบ Under Voltage ก็จะแสดงสถานะการเกิดเหตุการณ์ Under Voltage ให้ทราบโดยการเปลี่ยนสีพื้นของอุปกรณ์ตัวนั้นเป็นสีแดง และมีการบันทึกเหตุการณ์ไว้ตรวจสอบในภายหลัง โดยในเวลาต่อมาถ้าตรวจสอบแล้วพบว่าพอร์ต 1 กลับมาปกติโปรแกรมก็จะเปลี่ยนสถานะของอุปกรณ์ตัวนั้นเป็นปกติโดยการเปลี่ยนสีพื้นของอุปกรณ์ตัวนั้นเป็นสีเขียวพร้อมมีการบันทึกเหตุการณ์ไว้ตรวจสอบภายหลัง จากนั้นโปรแกรมก็จะไปตรวจสอบพอร์ตอื่นตามลำดับต่อไปจนครบทั้ง12 พอร์ต เมื่อครบแล้วก็จะวนมาตรวจสอบพอร์ต 1 ใหม่อีกเป็นอย่างนี้ไปเรื่อยๆ ดังภาพที่ 56 และ ภาพที่ 57



ภาพที่ 56 ผังงานการทำงานโปรแกรมติดต่อกับผู้ใช้งาน



ภาพที่ 57 ผังงานการทำงานโปรแกรมติดต่อกับผู้ใช้งาน (ต่อ)

3.2.3 การออกแบบแฟ้มข้อมูล

ระบบที่พัฒนาขึ้นมานี้มีแฟ้มข้อมูลที่ใช้สำหรับรวบรวมข้อมูลต่าง ๆ การทำงานในระบบ มีอยู่ 2 แฟ้มข้อมูลซึ่งแต่ละแฟ้มข้อมูลจะประกอบไปด้วยหลายตาราง

3.2.3.1 แฟ้มข้อมูล CONFIG ใช้สำหรับเก็บค่า Configuration ซึ่งมี 4 ตาราง ดังนี้

1) ตาราง PortConfig ใช้เก็บค่า Configuration ต่าง ๆ ในแต่ละพอร์ตของกล่องอินเตอร์เฟส ซึ่งรายละเอียดจะแสดงอยู่ในตารางที่ 11

2) ตาราง ComPort ใช้เก็บค่าตำแหน่ง ของพอร์ตอนุกรม โดยรายละเอียดจะแสดงในตารางที่ 12

3) ตาราง SecUserReg ใช้เก็บบัญชีรายชื่อผู้ใช้งาน และระดับการเข้าถึงข้อมูล ซึ่งรายละเอียดแสดงในตารางที่ 13

4) ตาราง Sound ใช้เก็บตำแหน่งที่อยู่ของไฟล์เสียง รายละเอียดแสดงในตารางที่ 14

3.2.3.2 แฟ้มข้อมูล LOGGING ใช้สำหรับเก็บเหตุการณ์และสถานะการทำงานต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลังในแต่ละวันและใช้สำหรับพิมพ์รายงาน ซึ่งมีอยู่ 2 ตาราง ดังนี้

1) ตาราง Power ใช้เก็บ Logging (เหตุการณ์สถานะการทำงานของอุปกรณ์ไฟฟ้ากำลังในแต่ละวัน) ซึ่งรายละเอียดแสดงในตารางที่ 15

2) ตาราง PrintLogging ใช้เก็บข้อมูลต่าง ๆ ชั่วคราว ที่ได้จากการค้นหามาเพื่อพิมพ์รายงาน ซึ่งรายละเอียดแสดงในตารางที่ 16

ตารางที่ 11 รายละเอียดในตาราง PortConfig

No.	Field Name	Field Type	Field Size	Description
1	PortID	Number	Integer	หมายเลข ID ของ Port
2	PortNo	Number	Integer	หมายเลข Port
3	Subsystem	Text	32	ชื่อระบบย่อย
4	Equipment	Text	32	ชื่ออุปกรณ์
5	Item	Text	32	ส่วนประกอบของอุปกรณ์
6	Status	Text	32	สถานะการทำงานของอุปกรณ์
7	Remark	Text	50	หมายเหตุของแต่ Port
8	PortStatus	Text	10	สถานะของ Port

ตารางที่ 12 รายละเอียดในตาราง ComPort

No.	Field Name	Field Type	Field Size	Description
1	ComID	Number	Integer	หมายเลข ID ของ Port (RS-232C)
2	ComName	Text	10	ชื่อ Port (RS-232C)

ตารางที่ 13 รายละเอียดในตาราง SecUserReg

No.	Field Name	Field Type	Field Size	Description
1	UserID	Autonumber	Long Integer	หมายเลข ID ของผู้ใช้
2	GroupNo	Number	Integer	กลุ่มในการเข้าถึงข้อมูล 1) เจ้าหน้าเวรปฏิบัติการ 2) ผู้ควบคุมระบบ
3	Username	Text	32	ชื่อในการใช้งาน
4	FullName	Text	32	ชื่อเต็ม
5	Password	Text	16	รหัสของผู้ใช้งาน
6	Description	Text	50	คำอธิบาย

ตารางที่ 14 รายละเอียดในตาราง Sound

No.	Field Name	Field Type	Field Size	Description
1	SoundId	Number	Integer	หมายเลข ID ของไฟล์เสียง
2	SoundFileName	Text	50	ที่อยู่ไฟล์เสียง

ตารางที่ 15 รายละเอียดในตาราง Power

No.	Field Name	Field Type	Field Size	Description
1	ID	AutoNumber	Long Integer	หมายเลข ID ของ Logging
2	DATE	Date/Time	-	วันที่บันทึก Logging
3	TIME	Date/Time	-	เวลาบันทึก Logging
4	SUB	Text	32	ชื่อระบบย่อย
5	EQUIP	Text	32	ชื่ออุปกรณ์
6	ITEM	Text	32	ส่วนประกอบของอุปกรณ์
7	STATUS	Text	32	สถานะการทำงานของอุปกรณ์
8	REMARK	Text	50	หมายเหตุ

ตารางที่ 16 รายละเอียดในตาราง PrintLogging

No.	Field Name	Field Type	Field Size	Description
1	ID	AutoNumber	LongInteger	หมายเลข ID ของ Logging
2	DATE	Date/Time	-	วันที่บันทึก Logging
3	TIME	Date/Time	-	เวลาบันทึก Logging
4	SUB	Text	32	ชื่อระบบย่อย
5	EQUIP	Text	32	ชื่ออุปกรณ์
6	ITEM	Text	32	ส่วนประกอบของอุปกรณ์
7	STATUS	Text	32	สถานะการทำงานของอุปกรณ์
8	REMARK	Text	50	หมายเหตุ

3.2.3.3 ระดับสิทธิในการเข้าไปใช้งาน

1) จัดระดับสิทธิในการเข้าไปใช้งาน โดยจัดแบ่งเป็น 3 กลุ่มตามการ Login ซึ่งเป็นการจัดระดับการเข้าถึงข้อมูลตามความเหมาะสมในการปฏิบัติงานซึ่งจะแตกต่างกัน ดังนี้

2) ไม่สามารถเข้าไปใช้งานได้นอกจากดู Status และ Logging ที่หน้าจอได้เท่านั้น

3) มีความสามารถเข้าไปใช้งานโดยมีความสามารถทำงานได้ ดังนี้

(1) ค้นหา (Find) ข้อมูลเหตุการณ์ การทำงานต่าง ๆ ของอุปกรณ์ต่าง ๆ ได้

(2) ลบ (Delete) ข้อมูลเหตุการณ์ การทำงานต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลัง ที่ได้บันทึกไว้ในแฟ้มข้อมูลในอดีตตามเงื่อนไขที่ต้องการได้

(3) คัดลอก (Copy) ข้อมูลเหตุการณ์ การทำงานต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลัง ที่ได้บันทึกไว้ในแฟ้มข้อมูลในอดีตตามเงื่อนไขที่ต้องการได้ โดยจะเก็บเป็นไฟล์นามสกุล .CSV สามารถใช้ Program Excel เปิดอ่านได้ และเก็บเป็น ไฟล์ html ได้

(4) พิมพ์ (Print) ข้อมูลเหตุการณ์ การทำงานต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลัง ที่ได้บันทึกไว้ในแฟ้มข้อมูลในอดีตตามเงื่อนไขที่ต้องการได้

3.2.3.4 มีความสามารถเพิ่มเติมคือ (สำหรับผู้ควบคุมระบบ)

(1) เพิ่ม User ได้

(2) ลบ User ได้

(3) เปลี่ยนคุณสมบัติต่าง ๆ ของ User ได้ เช่น ระดับการเข้าถึงข้อมูล

(4) เปลี่ยนหมายเลขพอร์ต สำหรับการสื่อสารได้

(5) เปลี่ยนเสียง Alarm เมื่อมีเหตุการณ์ Alarm ได้

(6) เปลี่ยนข้อความต่าง ๆ ใน Logging เมื่อสถานะการทำงานต่าง ๆ ของอุปกรณ์ไฟฟ้ากำลัง ที่กำลังตรวจจับเกิดการเปลี่ยนแปลง

บทที่ 4

การทดลองและผลการทดลอง

ผลที่ได้จากการดำเนินการเขียนโปรแกรมติดต่อและควบคุมพอร์ตอนุกรมด้วยโปรแกรมVisual Basic 6 ระบบการเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง (Main power monitoring and logging system) สามารถตรวจสอบสถานะการทำงานของระบบไฟฟ้ากำลังได้ และเฝ้าติดตามการทำงานของอุปกรณ์ได้ตลอดเวลา และสามารถบันทึกเหตุการณ์ที่เกิดขึ้นตามเวลาจริง เมื่อสถานะการทำงานของอุปกรณ์ไฟฟ้ากำลังที่เฝ้าติดตามอยู่เกิดการเปลี่ยนแปลงซึ่งใช้เป็นข้อมูลในการอ้างอิง การตรวจสอบสถานะการทำงานของอุปกรณ์ไฟฟ้ากำลังในภายหลังได้ การทดลองและผลจากการทดลอง มีองค์ประกอบที่สำคัญต่าง ๆ ดังนี้

4.1 เครื่องมือและวัสดุอุปกรณ์ แบ่งออกเป็น 3 ประเภท คือ

4.1.1 เครื่องมือวัดและอุปกรณ์ที่ใช้ในการทดลอง

4.1.1.1 Protocol analyzer	1 ชุด
4.1.1.2 Digital oscilloscope	1 ชุด
4.1.1.3 Pic & eeprom programmer	1 ชุด
4.1.1.4 NX-84 PIC16F84 Training	1 ชุด

4.1.2 อุปกรณ์ด้านฮาร์ดแวร์ที่ต้องการ

4.1.2.1 Micro Computer ที่มีพอร์ต RS-232C	1 ชุด
4.1.2.2 Printer LQ 300	1 ชุด
4.1.2.3 Power Supply +12V ขนาด 1A	1 ชุด
4.1.2.4 สายโทรศัพท์	1 ม้วน
4.1.2.5 Connector DB9	2 ตัว
4.1.2.6 Connector DB9	2 ตัว
4.1.2.7 Resistor 10 kΩ	30 ตัว
4.1.2.8 Resistor 220 Ω	12 ตัว
4.1.2.9 Resistor 4.7k	4 ตัว
4.1.2.10 Capacitor 1uF. 16 Volt	5 ตัว

4.1.2.11 Capacitor 1000 uF. 16 Volt 1 ตัว

4.1.2.12 Capacitor 47 pF. 2 ตัว

4.1.2.13 LED สีเขียว 1 ตัว

4.1.2.14 LED สีแดง 12 ตัว

4.1.2.15 Bridge Diode 1 Amp. 1 ตัว

4.1.2.16 Reset Switch (กดดับปล่อยติด) 1 ตัว

4.1.2.17 Dip Switch 8 bit 2 ตัว

4.1.2.18 แผ่นปริ้นเอนกประสงค์ 3" X 5" 1 แผ่น

4.1.2.19 Fuse 500 mA. 1 ชุด

4.1.2.20 IC. 16F84 1 ตัว

4.1.2.21 74LS244N 2 ตัว

4.1.2.22 IC. MAX 232 1 ตัว

4.1.2.23 IC. LM7805 1 ตัว

4.1.2.24 Crystal 4.00 MHz 1 ตัว

4.1.3 อุปกรณ์ซอฟต์แวร์ที่ต้องการ ดังนี้

4.1.3.1 โปรแกรม Microsoft Windows XP Professional 1 ชุด

4.1.3.2 โปรแกรม Microsoft Visual basic 6.0 1 ชุด

4.1.3.3 โปรแกรม Microsoft Office Access 2003 1 ชุด

4.1.3.4 โปรแกรม MP-LAB ของบริษัท Microship 1 ชุด

4.1.3.5 โปรแกรม ICPROG 1 ชุด

4.1.3.6 โปรแกรม PIP-02 Device Programmer 1 ชุด

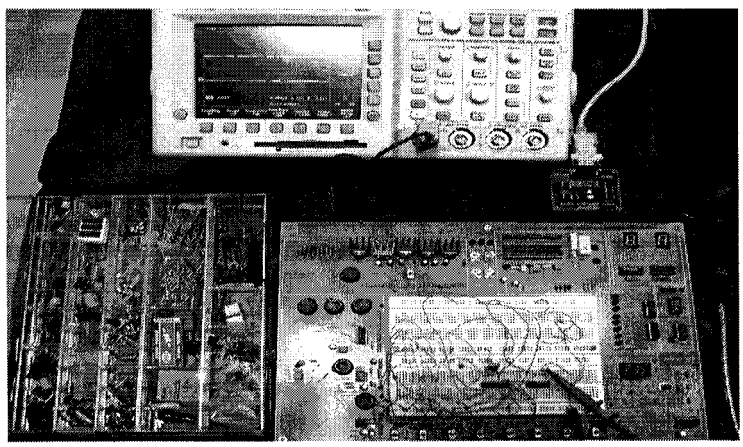
4.2 ขั้นตอนการสร้างระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง

4.2.1 ทดสอบการทำงานของกล่องอินเตอร์เฟสในเบื้องต้นจากที่ได้ออกแบบไว้ว่าสามารถทำงานได้จริงหรือไม่ ก่อนที่จะประกอบวงจรจริง โดยการเขียนโปรแกรมภาษาแอสเซมบลีตรวจสอบการทำงานของพอร์ต จากนั้นทำการแอสเซมเบลอร์ เขียนลงในไมโครคอนโทรลเลอร์และทดสอบการทำงาน ดังนี้

```
                    list  P=16F84
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Interface Box  xxxxxxxxxxxxxxxxxxxxxxxxxxxx;
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Config  Define xxxxxxxxxxxxxxxxxxxxxxxxxxxx;
CP_ON       equ     0x000f          ; Code Protect ON
CP_OFF      equ     0x3fff          ; Code Protect OFF
PWT_ON      equ     0x3fff          ; Power Up Timer ON
PWT_OFF     equ     0x3ff7          ; Power Up Timer OFF
WDT_ON      equ     0x3fff          ; Watch Dog Timer ON
WDT_OFF     equ     0x3ffb          ; Watch Dog Timer OFF
LP_OSC      equ     0x3ffc          ; LP Oscilator
XT_OSC      equ     0x3ffd          ; XT Oscilator
HS_OSC      equ     0x3ffe          ; HS Oscilator
RC_OSC      equ     0x3fff          ; RC Oscilator
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
;                   Code Protect OFF
;                   Power Up Timer ON
;                   Watch Dog Timer OFF
;                   XT Oscilator
                    __config CP_OFF&PWT_ON&WDT_OFF&XT_OSC
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Byte Define xxxxxxxxxxxxxxxxxxxxxxxxxxxx;
STATUS      equ     0x03
PORTA       equ     0x05
PORTB       equ     0x06
SEND        equ     0x0c
COUNT       equ     0x0d
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Bit Define xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
RP0             equ  5
F               equ  1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Test xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
                Org     0x000
                bsf     STATUS,RP0      ; Bank 1
                movlw   b'11111101'     ; RA1, =Output (TxData)
                movwf    PORTA
                movlw    b'11111111'    ; PortB = Input
                movwf   PORTB
                bcf     STATUS,RP0      ; Bank 0
start           bsf     PORTA,1         ; Mark bit   (ขา Tx.Data)
                call    Delay96
                btfss   PORTB,0
                call    L1
                goto    start
L1              call    Delay96
                call    Delay96
                bcf     PORTA,1         ; Start bit (1=31h)
                call    Delay96
                bsf     PORTA,1         ;Digit 1
                call     Delay96
                bcf     PORTA,1         ;Digit 2
                call    Delay96
                bcf     PORTA,1         ;Digit 3
                call    Delay96
                bcf     PORTA,1         ;Digit 4
                call    Delay96
                bsf     PORTA,1         ;Digit 5
                call    Delay96
                bsf     PORTA,1         ;Digit 6
```

```
                call    Delay96
                bcf     PORTA,1         ;Digit 7
                call    Delay96
                bcf     PORTA,1         ;Digit 8
                call    Delay96
                bsf     PORTA,1         ; Stop bit
                call    Delay96
                goto    L1
;xxxxxxxxxxxxxxxxxxxxxxxxx Delay Buadrate   104 Microsecond xxxxxxxxxxxxxxxxx;
Delay96         movlw   0x21
                movwf   COUNT
Delay1          decfsz  COUNT,F
                goto    Delay1
                return
                end
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
```

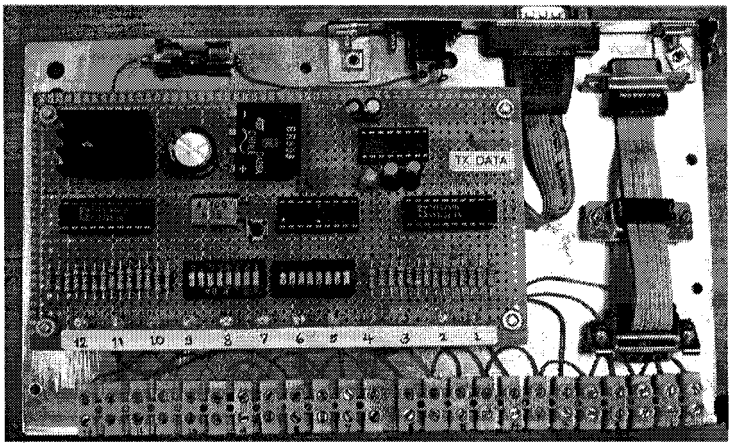


ภาพที่ 58 การทดสอบไมโครคอนโทรลเลอร์ว่าทำงานตามที่ได้ออกแบบไว้หรือไม่

จากการทดลองเบื้องต้นดังภาพที่ 58 ได้ทดสอบพอร์ตที่ 1 และส่งอักษร "1" ออกที่พอร์ต Tx.Data พอสรุปได้ว่าอุปกรณ์ที่จะใช้สร้าง กล่องอินเตอร์เฟส สามารถทำงานตามที่ได้ออกแบบไว้ และสามารถส่งข้อมูลจากอาคารไฟฟ้ากำลังไปยังห้องควบคุมอุปกรณ์ในอาคารควบคุมได้ จากนั้นจึงจะประกอบชุดกล่องอินเตอร์เฟสขึ้นจริง

4.2.2 การสร้างกล่องอินเตอร์เฟส

ทำการสร้างกล่องอินเตอร์เฟส โดยการประกอบอุปกรณ์ตามวงจรที่ได้วิเคราะห์และออกแบบไว้ ดังแสดงในภาพที่ 59



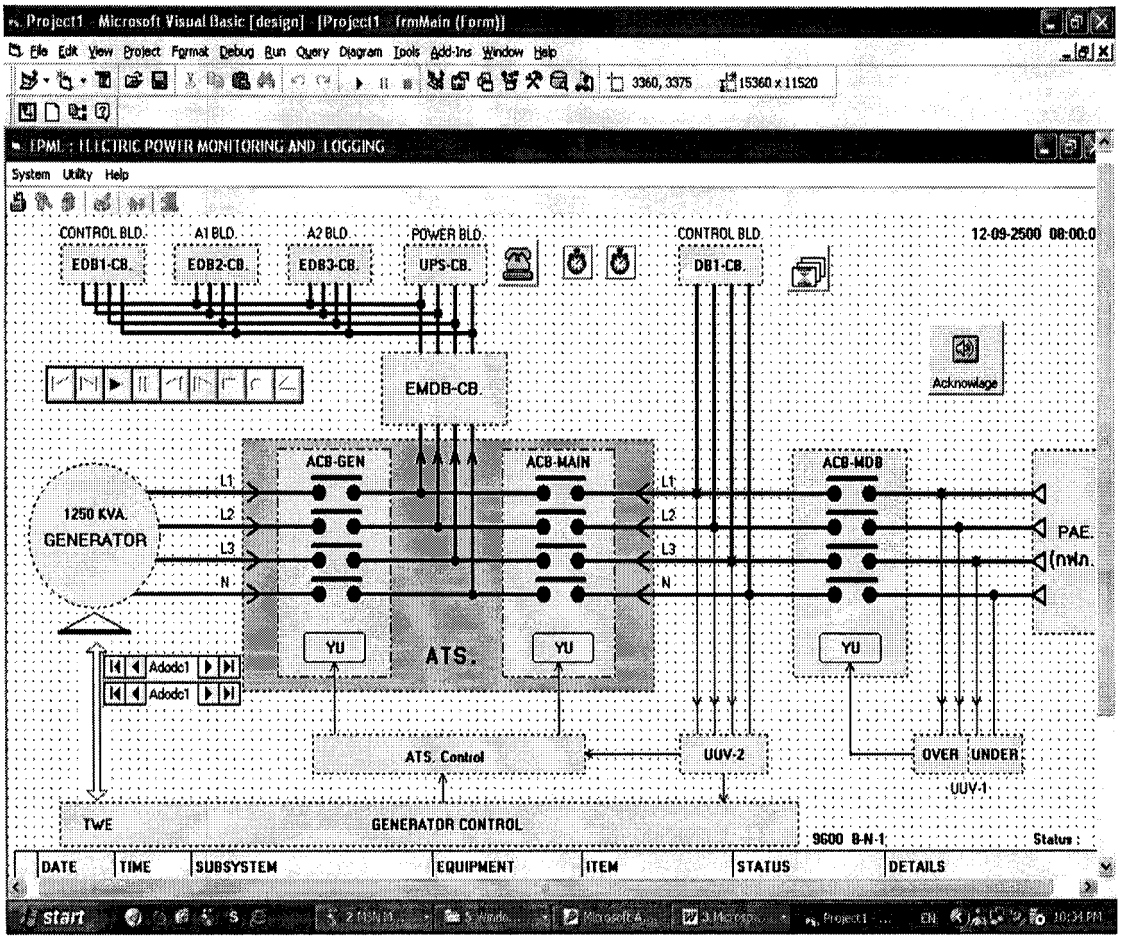
ภาพที่ 59 กล่องอินเตอร์เฟสที่สร้างขึ้น

4.2.3 เขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ ในกล่องอินเตอร์เฟส

ในส่วนของกล่องอินเตอร์เฟสนั้น เมื่อสร้างอุปกรณ์ Hardware เสร็จเรียบร้อยแล้วขั้นตอนต่อไป จะต้องเขียนโปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์ เบอร์ PIC16F84 ซึ่งเป็นของบริษัท Microship ให้ทำการตรวจสอบพอร์ตอินพุต ต่าง ๆ ที่ต่อกับ Sensor ที่ใช้วัดอุปกรณ์ไฟฟ้ากำลังต่าง ๆ ว่ามีสถานะเป็นอย่างไรแล้วก็จะส่งค่าข้อมูลที่อ่านได้เป็นสัญญาณอนุกรม Speed 9600 bps. ,1 bit start , 8 bit data ออกไปยังห้องควบคุม ภายในห้องควบคุม ก็จะมีโปรแกรมอ่านค่าสัญญาณจากกล่องอินเตอร์เฟส อีกทีหนึ่ง เพื่อแสดงสถานะ การทำงานของอุปกรณ์นั้น ๆ จากสัญญาณที่ถูกส่งมาจากกล่องอินเตอร์เฟส พร้อมทั้งบันทึกค่าเหตุการณ์ต่าง ๆ ที่ได้เปลี่ยนแปลงสถานะลงในแฟ้มข้อมูล เช่น วันที่ เวลา ชื่ออุปกรณ์ และสถานะการทำงานของอุปกรณ์ที่เกิดขึ้นเป็นต้น ตัวโปรแกรมภาษาแอสเซมบลีในกล่องอินเตอร์เฟสจะอยู่ในภาคผนวก ข และวิธีการเขียนโปรแกรมลงบนตัวไมโครคอนโทรลเลอร์ ก็จะอธิบายการใช้โปรแกรม ICPROG ซึ่งอยู่ในภาคผนวก จ

4.2.4 เขียนโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE)

เป็นการเขียนโปรแกรมติดต่อกับผู้ใช้งานที่อ่านค่าสัญญาณที่ส่งมาจากกล่องอินเตอร์เฟสในอาคารการกำลังที่ได้ออกแบบไว้ในบทที่ 3 โดยโปรแกรมนี้จะถูกติดตั้งที่ห้องควบคุมในอาคารควบคุมของสถานีดาวเทียมสิรินธร ซึ่งอาคารทั้ง 2 แห่งจะอยู่ห่างกันประมาณ 60 ถึง 80 เมตร เพื่อให้เจ้าหน้าที่ฝ่ายปฏิบัติการ ได้ทราบสถานะการทำงานของอุปกรณ์ไฟฟ้ากำลังได้ตลอดเวลา โดยได้ใช้โปรแกรม Visual Basic 6.0 เป็นเครื่องมือในการพัฒนา ซึ่งในโปรแกรม Visual Basic 6.0 จะมีพอร์ต RS-232C สำหรับสื่อสารที่มีชื่อว่า MSComm เป็นเครื่องมือในการอ่านสัญญาณจากกล่องอินเตอร์เฟสและแฟ้มข้อมูลใช้โปรแกรม Microsoft Office Access ในการจัดการแฟ้มข้อมูล ส่วนตัวโปรแกรมซึ่งพัฒนาโดยใช้ภาษา Visual Basic จะอยู่ในภาคผนวก ค ส่วนภาพที่ 60 เป็นการแสดงตัวโปรแกรมติดต่อกับผู้ใช้ที่พัฒนาขึ้นมาเรียบร้อยแล้ว

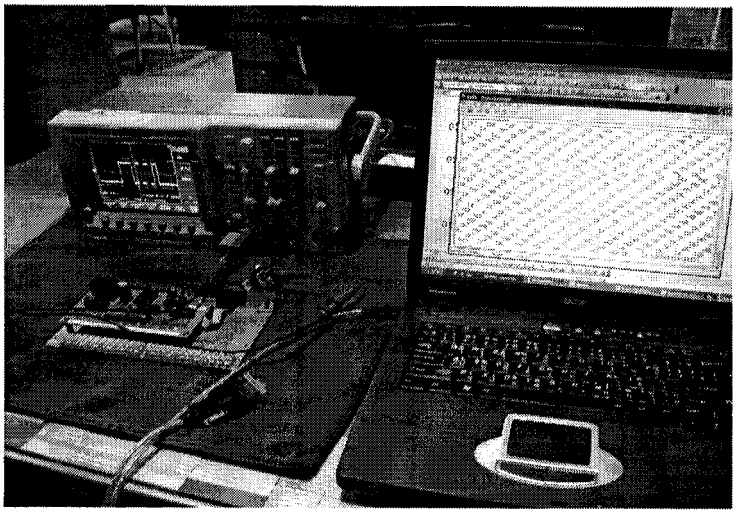


ภาพที่ 60 โปรแกรมติดต่อกับผู้ใช้ที่พัฒนาขึ้นมา

4.3 ทดสอบการใช้งาน

4.3.1 ทดสอบกล่องอินเตอร์เฟส

ก่อนจะนำไปใช้งานจริง เมื่อประกอบวงจรเรียบร้อยแล้ว ใช้ Digital oscilloscope และ โปรแกรม Hyper Terminal ตรวจสอบสัญญาณจากไมโครคอนโทรลเลอร์ ขา Tx.Data (ขา 18) แล้วทดลองตรวจสอบขาอินพุต ทั้ง 12 พอร์ต ที่จะต่อเข้ากับ Sensor ว่าทำงานได้ถูกต้องหรือไม่ โดยใช้วิธีการตรวจสอบ หากพอร์ตใดทำงาน ก็จะส่งค่าประจำพอร์ต ออกไป เช่น พอร์ต 1 ทำงาน ก็จะส่งรหัสแอสกี้ 01 ออกไป ดังภาพที่ 61 ซึ่ง พอร์ตอินพุต ทั้ง 12 พอร์ต ทำงาน

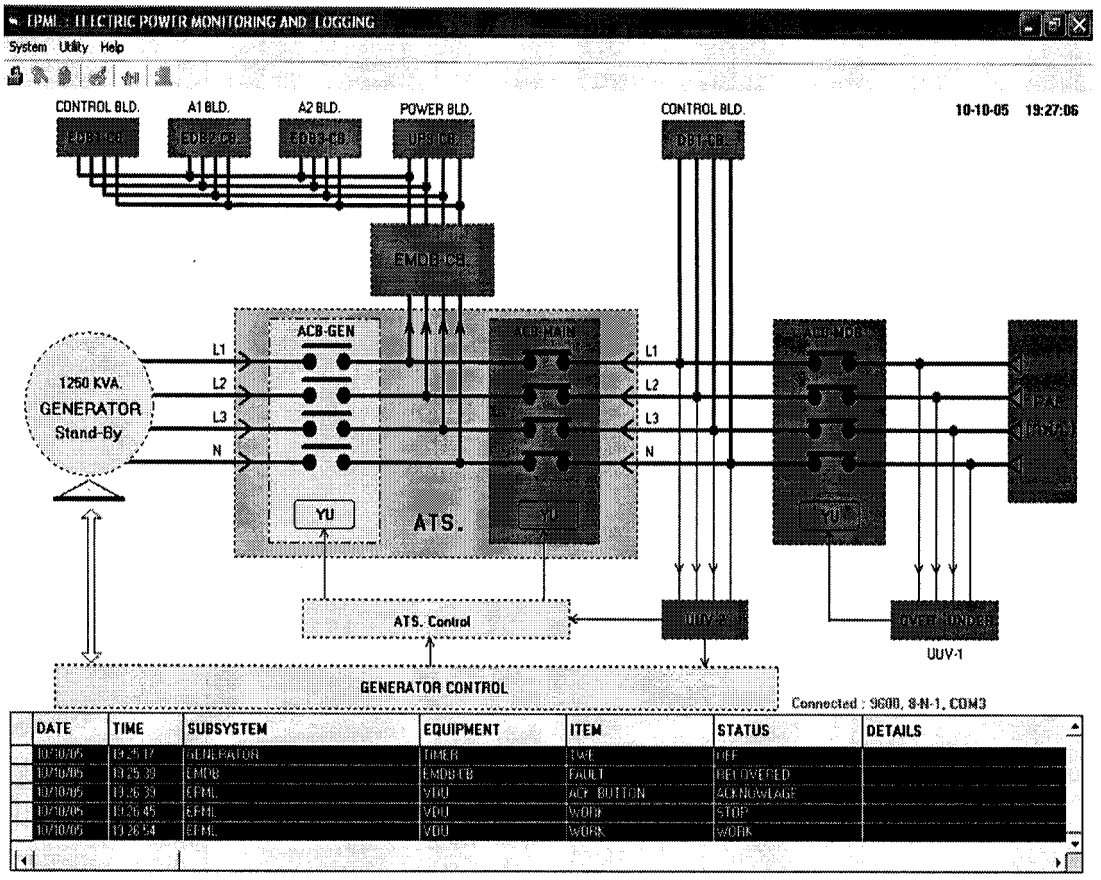


ภาพที่ 61 การใช้ Osciloscope และโปรแกรม Hyper Terminal ตรวจสอบการทำงาน กล่องอินเตอร์เฟส

4.3.2 ทดสอบโปรแกรมติดต่อกับผู้ใช้งาน

ในการทดสอบนั้นจะกระทำในระหว่างการเขียนโปรแกรมว่า ผลที่ได้ตรงกับที่ได้ออกแบบไว้ในบทที่ 3 หรือไม่ ถ้าไม่ตรงก็จะแก้ไขในขั้นตอนนี้เลย วิธีการทดสอบทำได้โดย การต่อกล่องอินเตอร์เฟส เข้ากับ โปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE) ทางพอร์ตอนุกรม (RS-232C) ทดสอบโดยการต่อ Sensor ซึ่งเป็น Relay เข้า พอร์ต ต่าง ๆ ในกล่องอินเตอร์เฟส จำลองให้ Sensor ต่าง ๆ เหล่านั้นทำงาน และตรวจสอบดูว่า เมื่อ Sensor เหล่านั้นทำงาน กล่องอินเตอร์เฟส สามารถส่ง Data ไปเปลี่ยนแปลงสถานะการทำงานในโปรแกรมติดต่อกับผู้ใช้งานตามที่ได้ออกแบบไว้หรือไม่

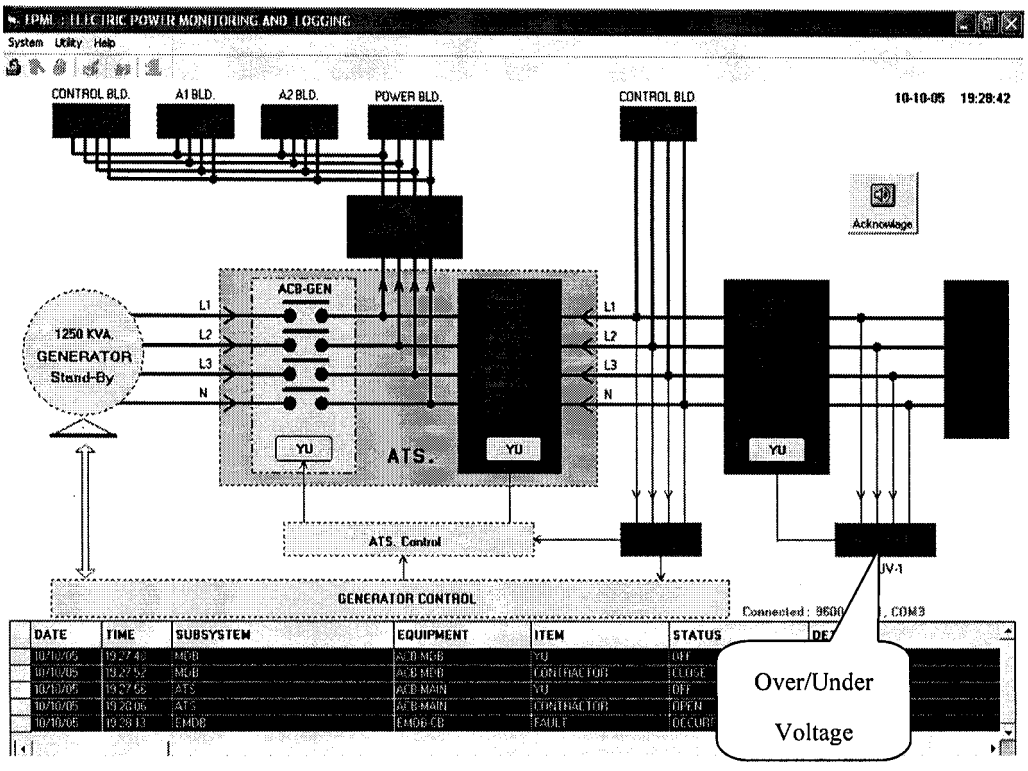
ภาพที่ 62 หน้าแรกของโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE)



ภาพที่ 63 ไฟฟ้าจาก กฟภ. ปกติ

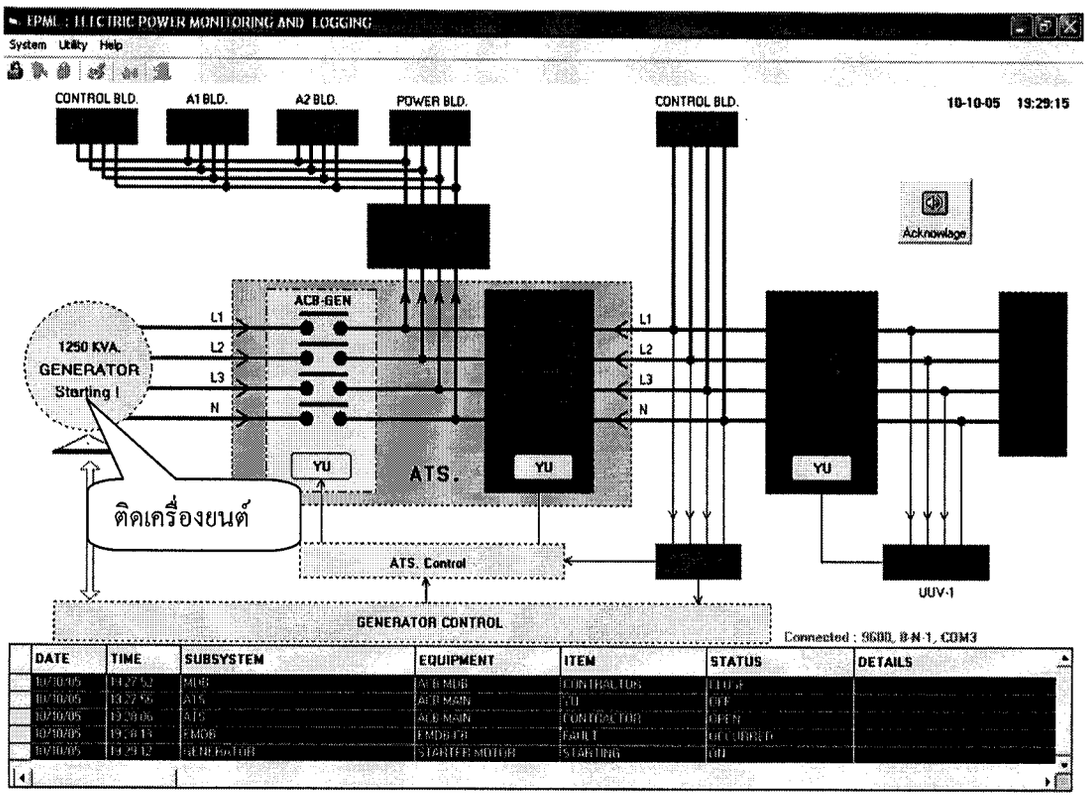
จากภาพที่ 63 แสดงโปรแกรมติดต่อกับผู้ใช้ จากภาพจะเห็นว่า ไฟฟ้าจาก กฟภ. มีแรงดัน(Voltage) ปกติ (360 – 440 VAC. 3ϕ) ซึ่งแรงดันที่ได้จาก กฟภ. นั้นจะถูกตรวจสอบแรงดันโดยอุปกรณ์ UUV-1 ซึ่งจะตรวจสอบทั้ง แรงดันเกินกว่า 440 V. (Over Voltage) และแรงดันต่ำกว่า 360 V. (Under Voltage) โดยถ้าหากผิดปกติ คือ มีแรงดันต่ำกว่า หรือสูงกว่าค่าที่

กำหนด ตัว UUV-1 จะตัดกระแสไฟฟ้าไม่ให้ไปเลี้ยง Coil YU โดยถ้าหากขดลวด (Coil) ของ YU ไม่มีไฟฟ้าไปเลี้ยง ตัว ACB-MDB ซึ่งเป็น Air Circuit Breaker ก็จะตก (หน้า Contract จะไม่ต่อถึงกัน) ทำให้ไม่มีแรงดันจาก กฟภ. ผ่านออกไป แต่ในภาพก่อนหน้านี้ จะเห็นว่าแรงดันจาก กฟภ. ปกติ จะมีแรงดันผ่านหน้า Contract ของ ACB-MDB ไปยัง ACB-MAIN และ UUV-2 ซึ่ง UUV-2 จะเป็น Under/Over Voltage ของ ระบบ ATS.(Automatic Transfer System) ซึ่งจะใช้ตรวจสอบแรงดันก่อนที่จะเข้าระบบ ATS. ว่าเป็นปกติหรือไม่ โดยถ้าหากแรงดันเป็นปกติ จะมีค่าอยู่ระหว่าง 360 – 440 V. ในระบบ 3 Phase ก็จะมีแรงดันออกไปที่ระบบควบคุม ATS. ซึ่งระบบควบคุม ATS. จะคอยควบคุมการทำงานของ ACB.(Air Circuit Breaker) ทั้งสอง คือ ACB-MAIN กับ ACB-GEN ซึ่งเป็นตัวตัดต่อกระแสไฟฟ้าไม่ให้ทำงานพร้อมกัน (โดยทั้งคู่จะทำงานสลับกัน) เพื่อไม่ให้เกิดการชนกันทางไฟฟ้า (ไฟฟ้าจาก กฟภ. กับไฟฟ้าจากเครื่องกำเนิดไฟฟ้า) ซึ่งอันตรายมากจะมีวิธีป้องกันในทางไฟฟ้า และทางกลไกล จากภาพข้างบนจะเห็นว่า UUV-2 ตรวจสอบแรงเป็นปกติ จึงมีแรงดันไปบอก ระบบควบคุม ATS. ให้ต่อแรงดันที่ได้จาก กฟภ. ส่งไปให้ Load ต่าง ๆ เช่น อาคารควบคุม, อาคารจานสายอากาศ 21 เมตร, อาคารจานสายอากาศ 32 เมตร, อาคารการกำลัง ซึ่งแรงดันไฟฟ้าที่ได้จากส่วนนี้เมื่อไฟฟ้าดับ จะมีแรงดันขาดช่วงประมาณไม่เกิน 10 วินาที โดยโหลดที่ใช้ไฟฟ้าจากจุดนี้เรียกว่า EMDB (Emergency Main Distribution Board) ซึ่งไฟฟ้าจะขาดช่วงในตอนที่ขณะเครื่องกำเนิดไฟฟ้าเริ่มทำงานเท่านั้น



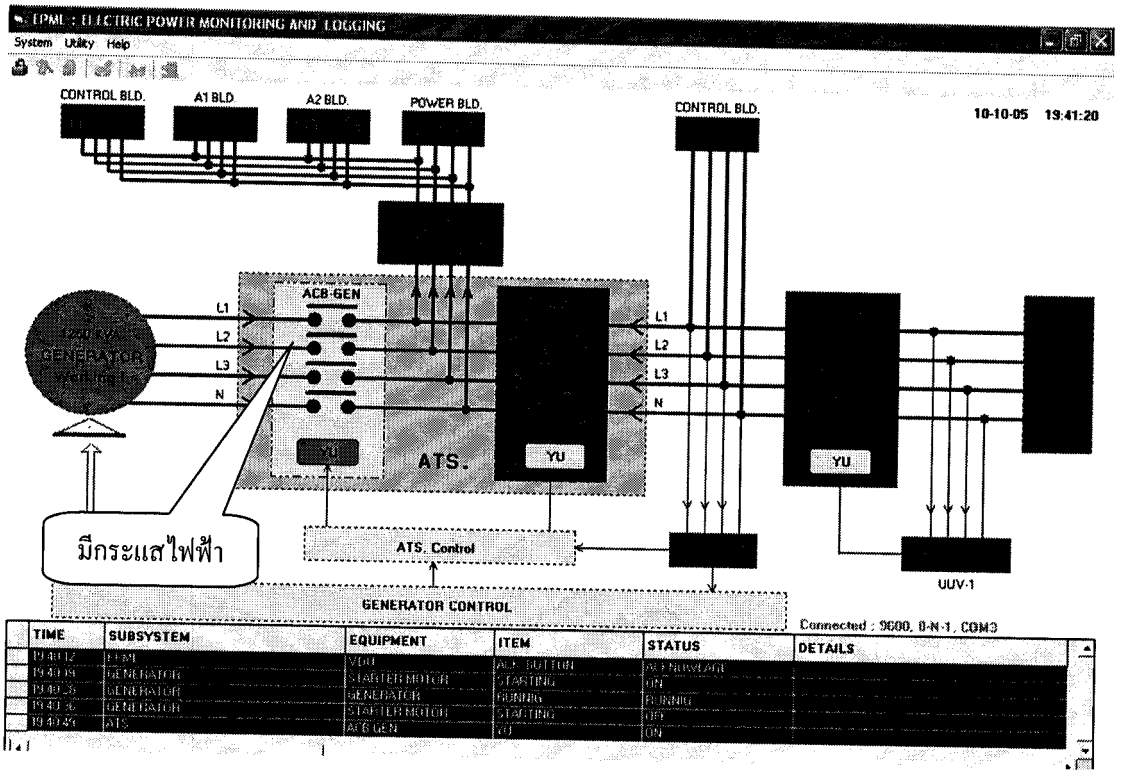
ภาพที่ 64 ไฟฟ้าจาก กฟภ. เกิด Over/Under Voltage

จากภาพที่ 64 แสดงสถานะกรณีไฟฟ้าจาก กฟภ. เกิด Over/Under Voltage หรือผิดปกติโดยจะมี UUV-1 คอยตรวจสอบแรงดันไฟฟ้าจาก กฟภ.ที่จ่ายเข้ามายัง สถานีดาวเทียม หากผิดปกติ (ไม่อยู่ในช่วง 360 – 440 VAC.) ก็จะสั่งให้ ACB-MDB หยุดทำงาน (Open Circuit) มีสัญลักษณ์สีแดง มีเสียง Alarm และ ปุ่ม Acknowledge จะ pop-up ขึ้นมา พร้อมทั้งบันทึกเหตุการณ์การทำงานของอุปกรณ์ต่าง ๆ ขึ้นที่หน้าต่าง Logging Window ดังภาพที่ 64 จะเห็นว่าไม่มีแรงดันไปจ่ายโหลดตามอาคารต่าง ๆ



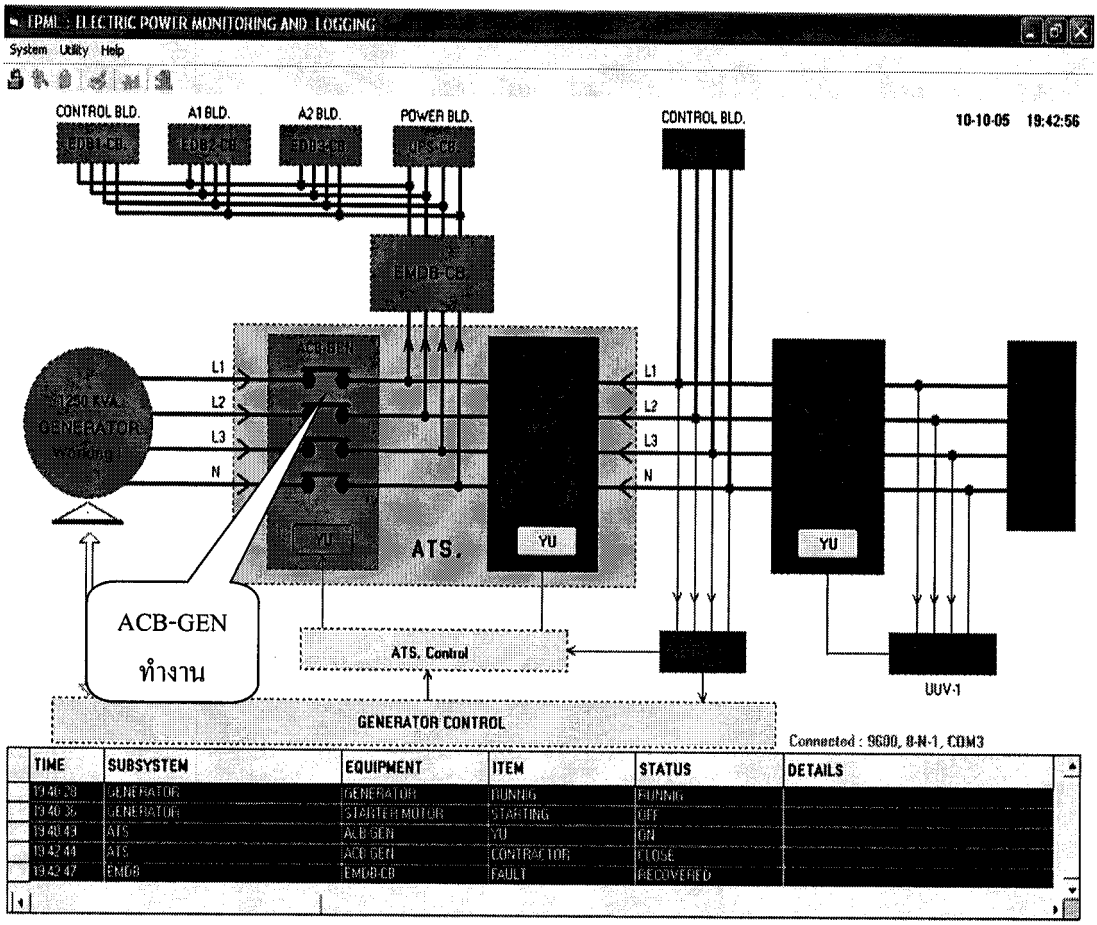
ภาพที่ 65 เครื่องกำเนิดไฟฟ้า (Generator) เริ่มทำงาน

จากภาพที่ 65 แสดงการทำงานของเครื่องกำเนิดไฟฟ้า (Generator) เริ่มทำงาน UUV-2 ซึ่งจะทำการตรวจสอบแรงดันไฟฟ้าจาก กฟภ. หากไม่มีแรงดันมา UUV-2 ก็จะสั่งให้ชุดควบคุมเครื่องกำเนิดไฟฟ้า ติดเครื่องยนต์เพื่อปั่นเครื่องกำเนิดไฟฟ้า (Generator)



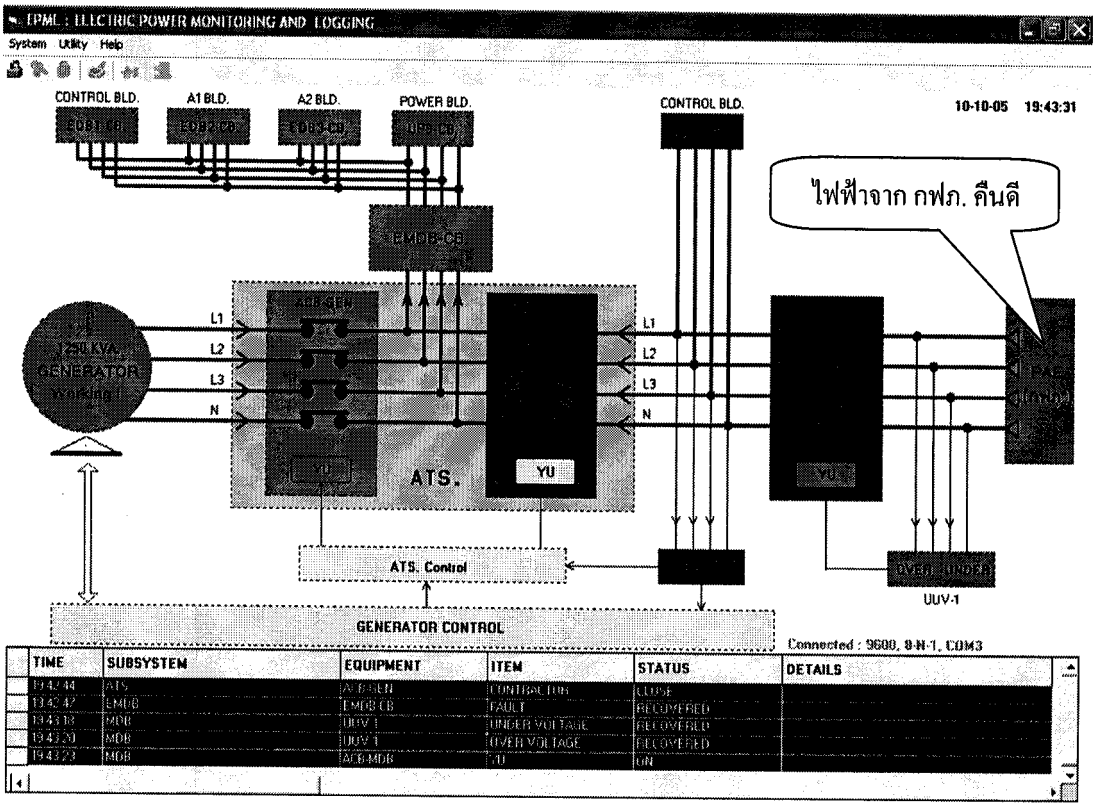
ภาพที่ 66 เครื่องกำเนิดไฟฟ้าจ่ายกระไฟฟ้ามาที่ ACB-GEN

จากภาพที่ 66 เมื่อเครื่องกำเนิดไฟฟ้าทำงานก็จะมีแรงดันมาที่ ACB-GEN และ ชุดควบคุม ATS. เพื่อจะ ถ่ายโหลด (Transfer) ไฟฟ้าที่ได้จากเครื่องกำเนิดไฟฟ้าไปยังโหลดตามอาคารต่าง ๆ โดย ACB-GEN ในระบบ ATS.



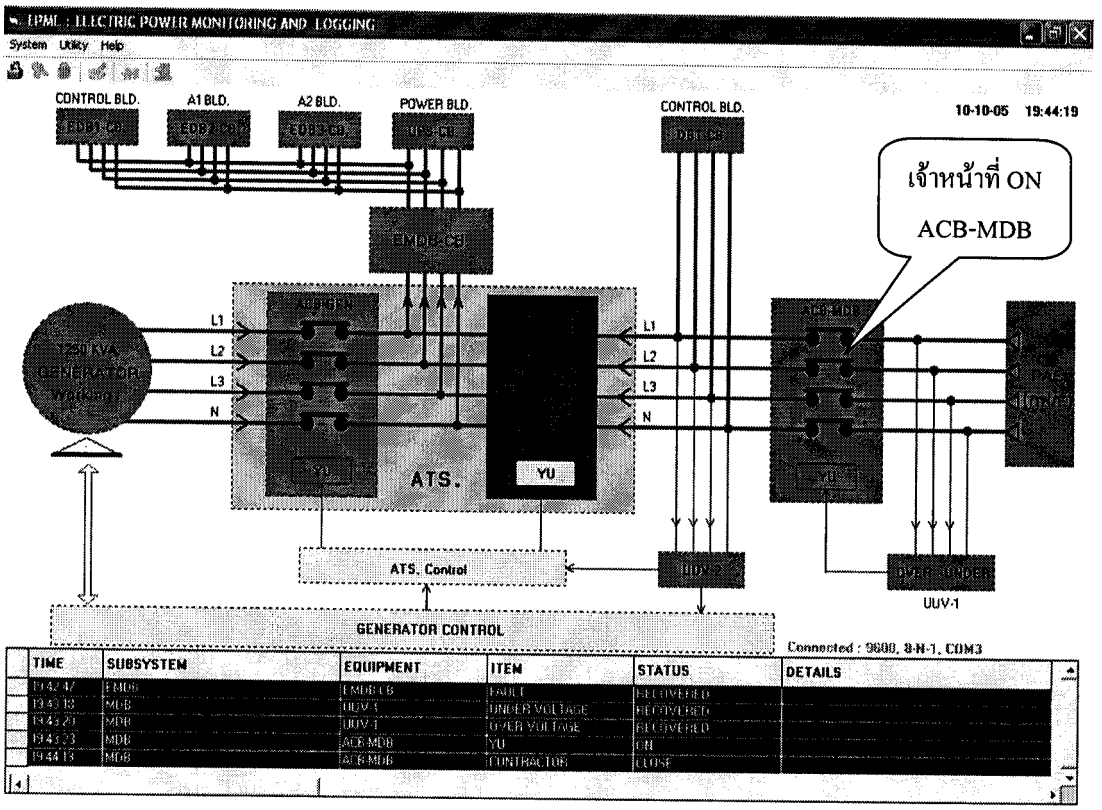
ภาพที่ 67 ACB-GEN ทำงาน (Close Circuit)

จากภาพที่ 67 ระบบ ATS. โดย ACB-GEN ถ่ายโหลด (Transfer) จากเครื่องกำเนิดไฟฟ้าไปยังโหลด ตามอาคารต่าง ๆ เรียบร้อยแล้วโดยผ่าน EMDB-CB ซึ่งเป็น Circuit Breaker ซึ่งเป็นอุปกรณ์ป้องกันในกรณีที่เกิดการลัดวงจร



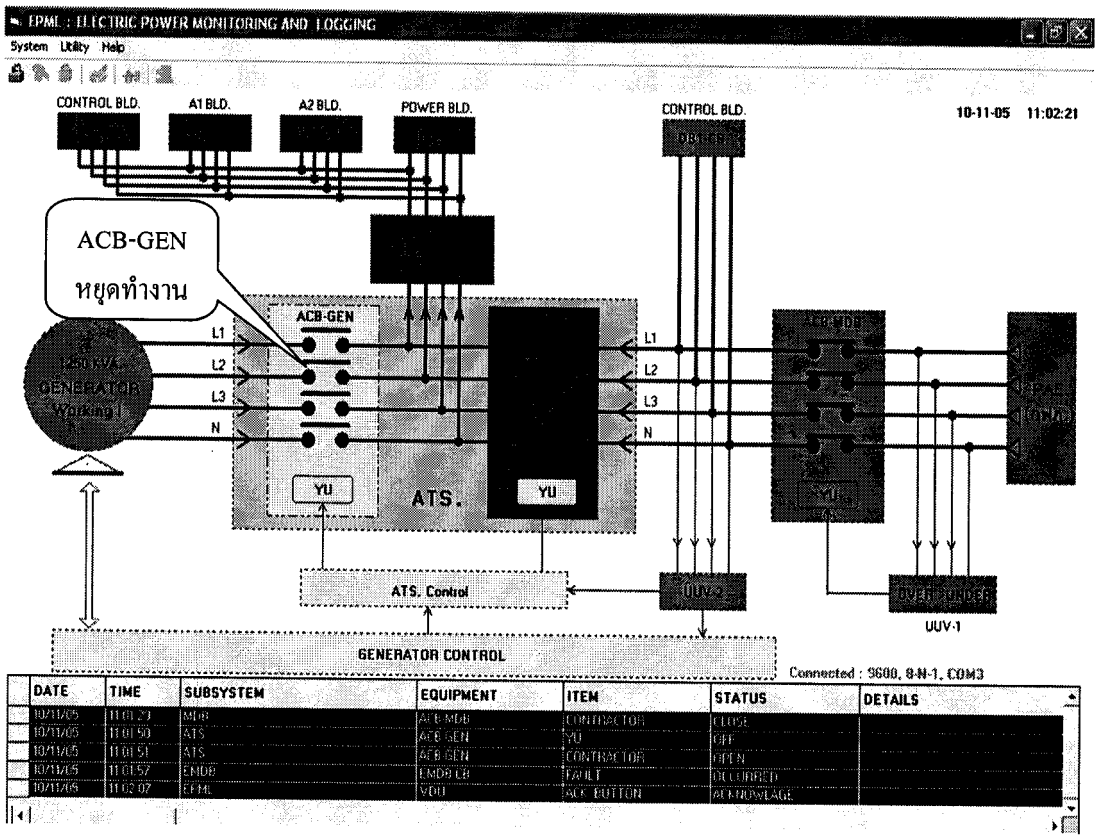
ภาพที่ 68 ไฟฟ้าจาก กฟภ. กลับมาเป็นปกติในขณะที่ยังใช้ไฟฟ้าจากเครื่องกำเนิดไฟฟ้า

จากภาพที่ 68 แสดงสถานะการใช้ไฟฟ้าจากเครื่องกำเนิดไฟฟ้า และในเวลาต่อมาไฟฟ้าจาก กฟภ. กลับคืนมาเป็นปกติ ซึ่งสังเกตได้จากสี ซึ่งจะเป็นสีเขียวพร้อมกับบันทึกเวลาคืนดีลงในหน้าต่าง Logging Window และจะมีแรงดันไฟฟ้าไปจ่ายให้ขดลวด YU ของ ACB-MDB



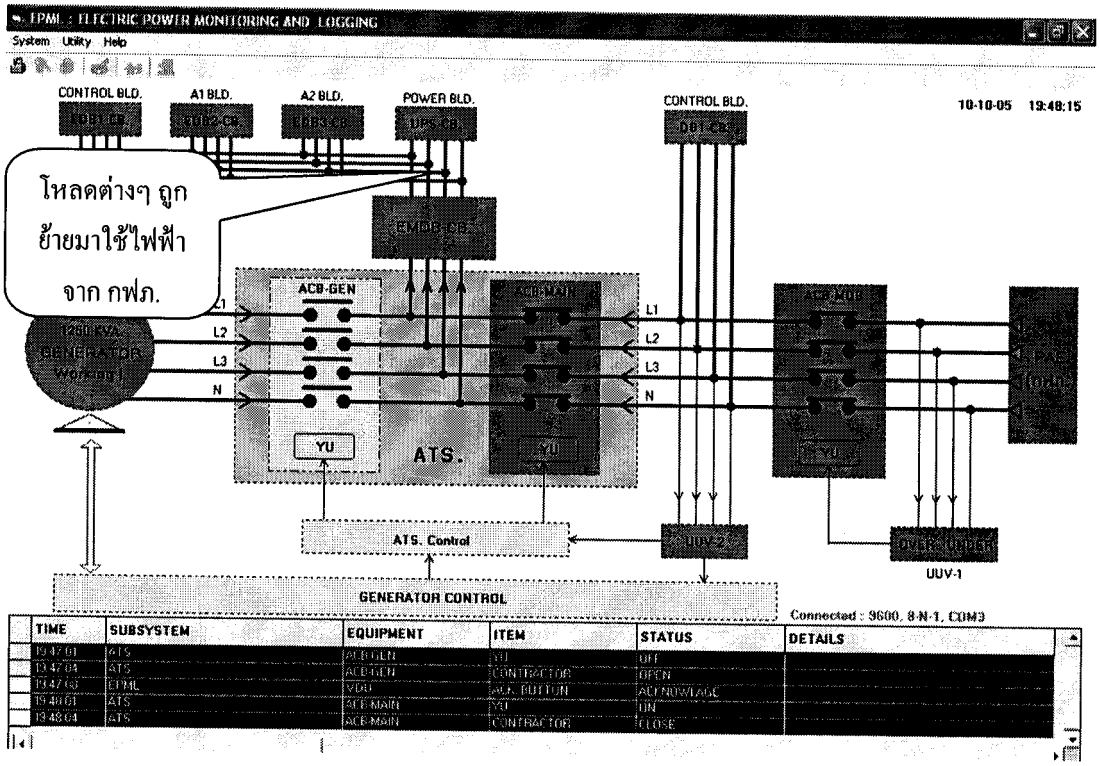
ภาพที่ 69 ACB-MDB ทำงาน (Close Circuit)

จากภาพที่ 69 เมื่อเจ้าหน้าที่ประจำสถานีดาวเทียม ตรวจสอบความเรียบร้อยแล้ว หากพบว่าไม่มีผิดสิ่งปกติใด ๆ เจ้าหน้าประจำสถานีก็จะทำการ ON ACB-MDB เพื่อจ่ายกระแสไฟฟ้าจาก กฟภ. เข้าไปในระบบให้ DB1-CB (เป็น Circuit Breaker ของโหลดที่ไม่สำคัญ เช่น ไฟฟ้าแสงสว่างซึ่งโหลดที่ได้จากตรงนี้จะขาดช่วงเมื่อตอนไฟฟ้า จาก กฟภ. ผิดปกติ) และจะมี UUV-2 คอยตรวจสอบแรงดันจาก ACB-MDB ว่ามีไฟฟ้ามาหรือไม่ ถ้าหากเจ้าหน้าประจำสถานีไป ON ACB-MDB เรียบร้อยแล้วก็จะมีแรงดันออกมาที่ตัว UUV-2 ๆ ก็จะสั่งให้ตัวควบคุม ATS. ทำงาน



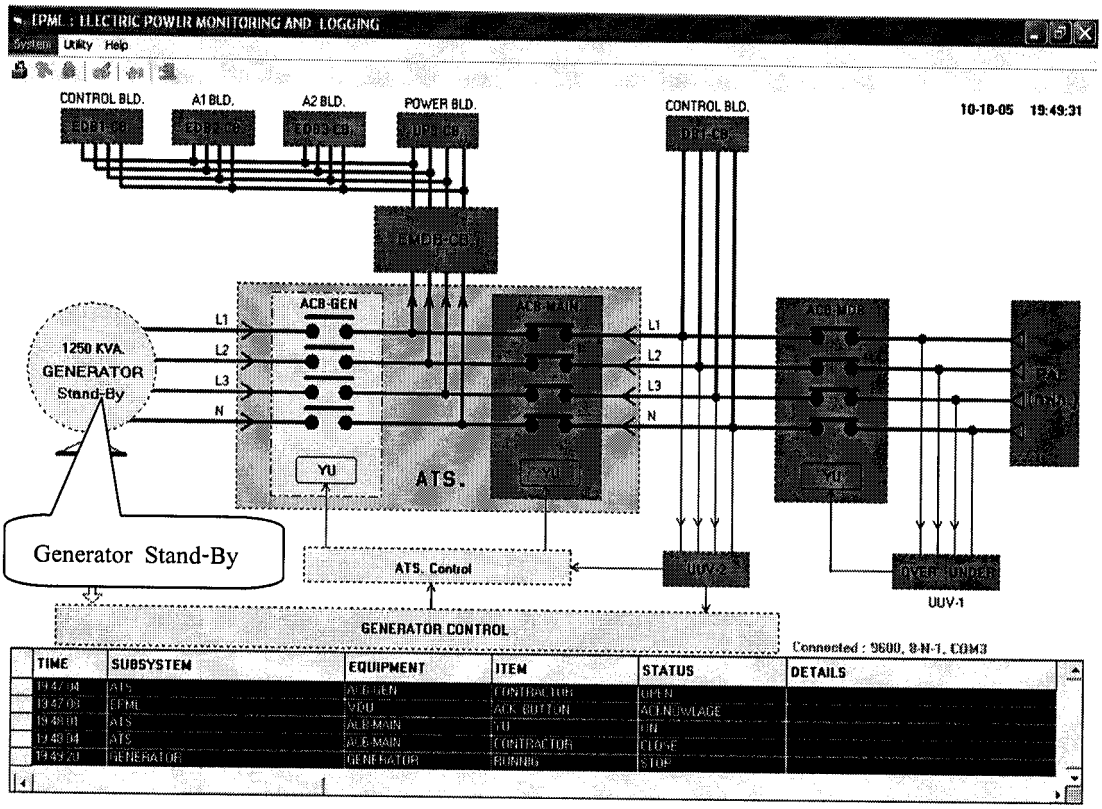
ภาพที่ 70 ระบบ ATS. สั่งให้ ACB-GEN หยุดทำงาน (Open Circuit)

จากภาพที่ 70 จะเห็นได้ว่าตัว UUV-2 ทำงานสั่งให้ ACB-GEN หยุดทำงาน (Open Circuit) เพื่อป้องกันไฟฟ้าจาก กฟภ. กับไฟฟ้าที่ได้จากเครื่องกำเนิดไฟฟ้าชนกันซึ่งจะอันตรายมาก



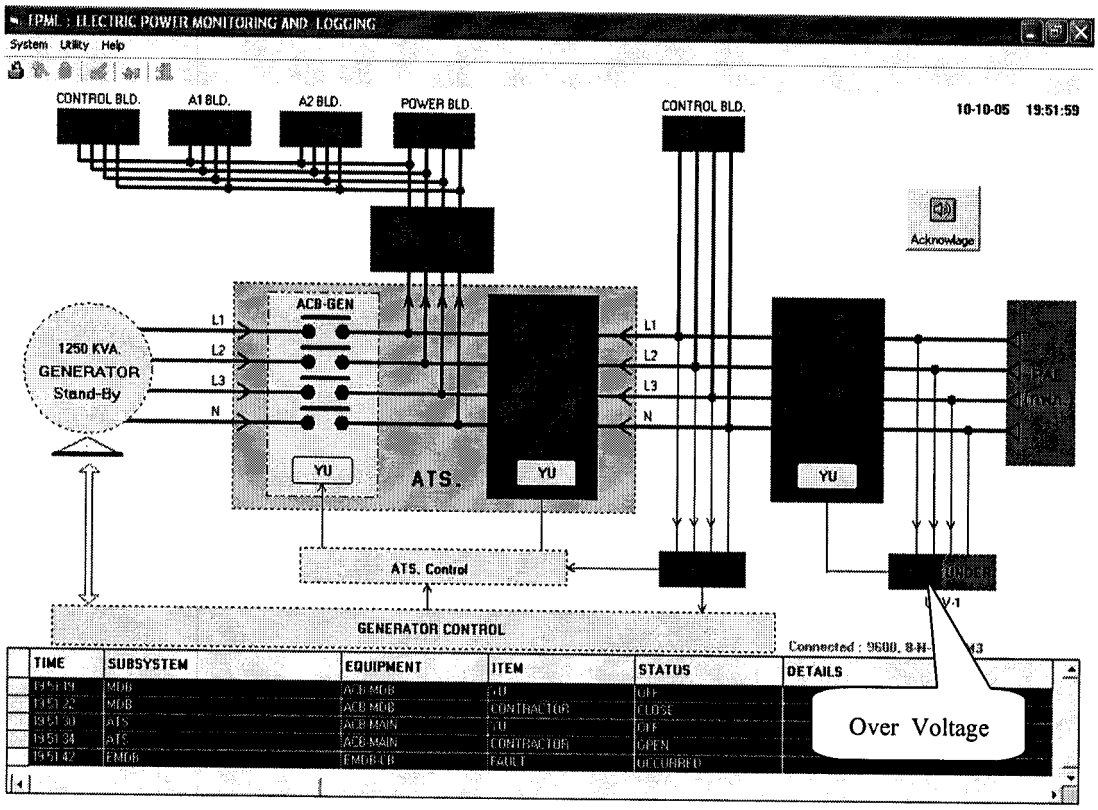
ภาพที่ 71 โหลดที่ถูกถ่ายไปยัง กฟภ. เรียบร้อยแล้ว

จากภาพที่ 71 ตัวควบคุม ATS. จะหน่วงเวลาประมาณ 1-2 วินาที แล้วตัว ควบคุม ATS. จะสั่งให้ ACB-MAIN ทำงาน (Close Circuit) ทำให้มีกระแสไฟฟ้าจาก กฟภ. จ่ายไปยังโหลดในอาคารต่าง ๆ โดยผ่าน EMDB-CB ซึ่งเป็น Circuit Breaker (เป็นตัวป้องกันหากเกิดเหตุการณ์ Short Circuit ขึ้นมา) และเมื่อ ATS. Switch ถ่ายโหลดไฟฟ้าจากเครื่องกำเนิดไฟฟ้าไปยังไฟฟ้าจาก กฟภ.เรียบร้อยแล้ว เครื่องยนต์ก็ยังทำงานต่อไปเพื่อระบายความร้อนออกจากเครื่องยนต์ซึ่งเรียกว่า Run Down ใช้เวลาประมาณ 4 นาที



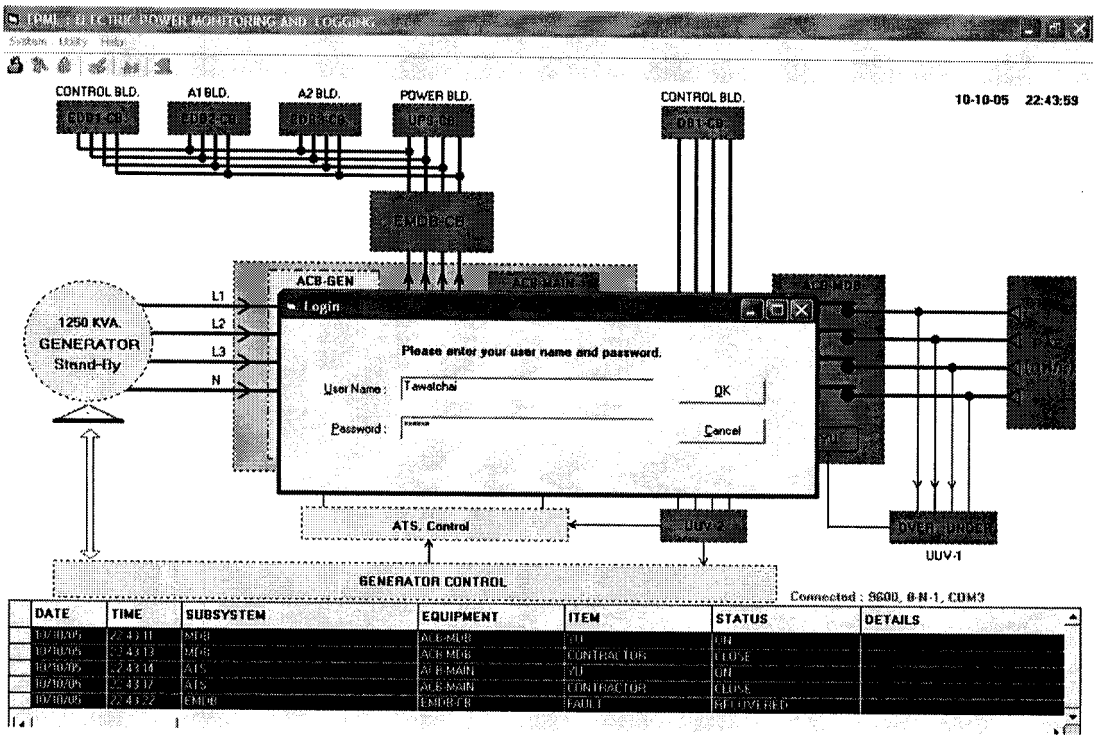
ภาพที่ 72 เครื่องกำเนิดไฟฟ้าหยุดทำงานและ Stand-By

จากภาพที่ 72 เมื่อเครื่องยนต์ Run Down เพื่อระบายความร้อนออกจากเครื่องยนต์เรียบร้อยแล้ว เครื่องยนต์ก็จะหยุดทำงานและ พร้อมที่จะทำงานในครั้งต่อไปที่เรียกว่า Stand-By



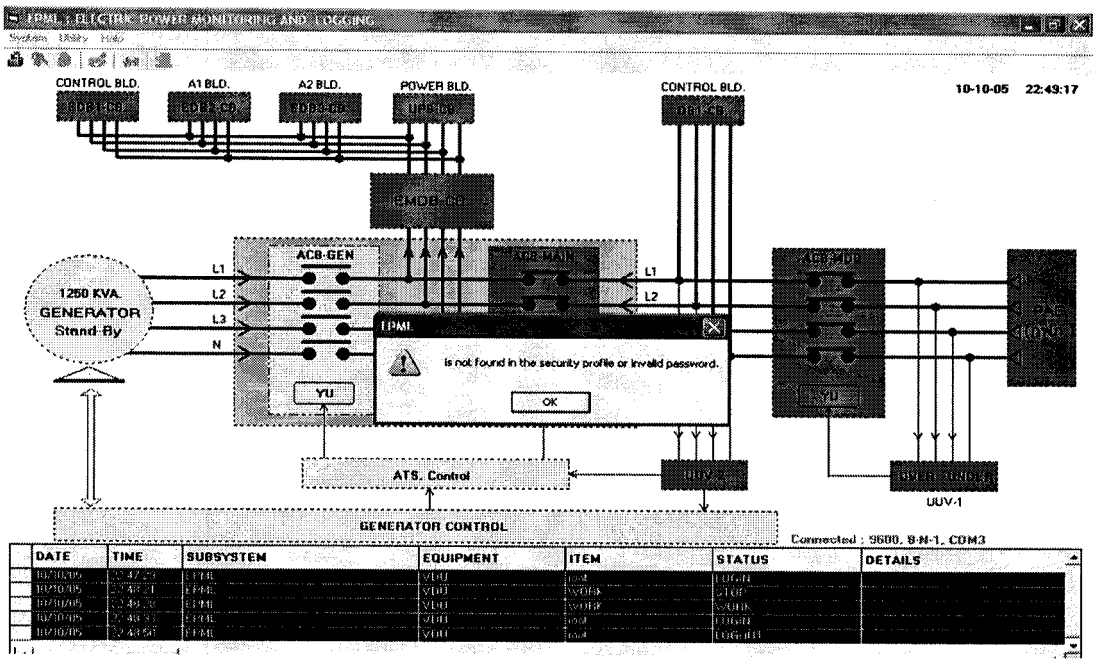
ภาพที่ 73 การเกิดแรงดันไฟฟ้าจาก กฟภ. มีแรงดันเกิน (Over Voltage)

จากภาพที่ 73 หากไฟฟ้าจาก กฟภ. เกิดสิ่งผิดปกติเช่น แรงดันไฟฟ้าเกิน 440 Volt ในระบบ 3 Phase อุปกรณ์ UUV-1 ก็จะทำสั่งให้ MCB-MDB หยุดทำงานตัดกระแสไฟฟ้าจาก กฟภ. ออกจากระบบ เพื่อไม่ให้อุปกรณ์ดาวเทียมและอุปกรณ์ไฟฟ้าต่าง ๆ ภายในสถานีดาวเทียม เกิดความเสียหาย และก็จะสั่งให้เครื่องยนต์ทำงานและปั่นเครื่องกำเนิดไฟฟ้าจ่ายกระแสไฟฟ้าทดแทน



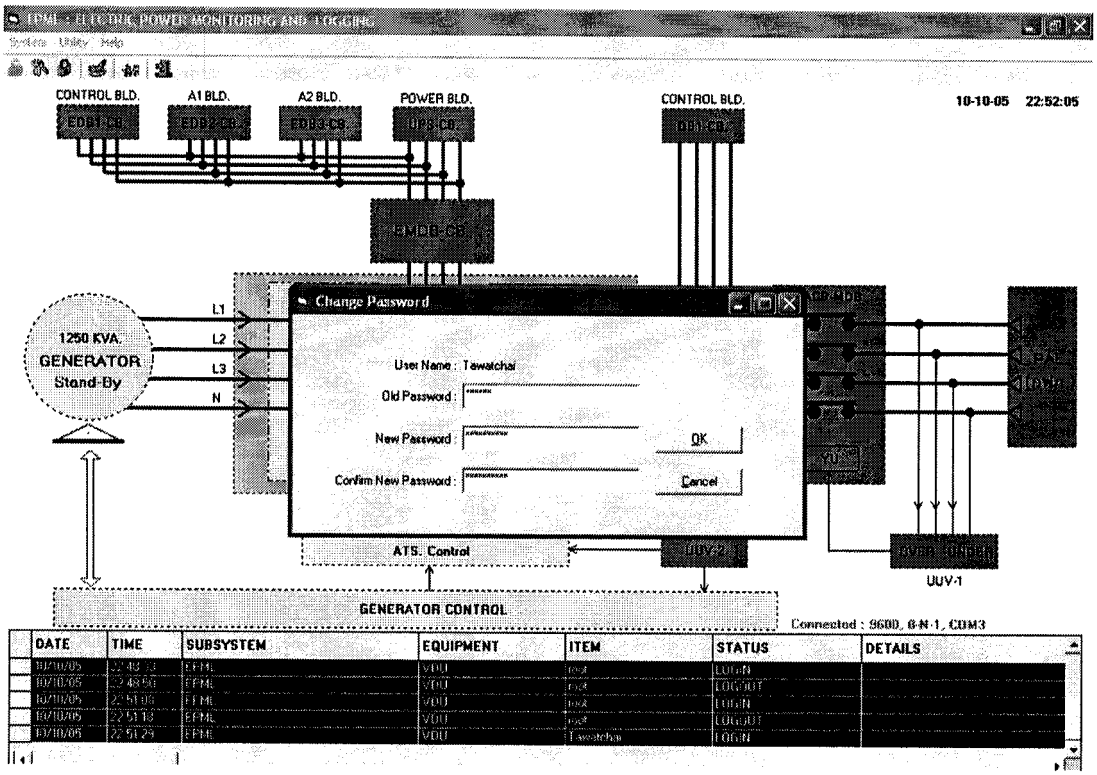
ภาพที่ 74 หน้า Login

จากภาพที่ 74 เป็นหน้าสำหรับ Login เพื่อตรวจสอบสิทธิเข้าไปใช้งานโปรแกรม



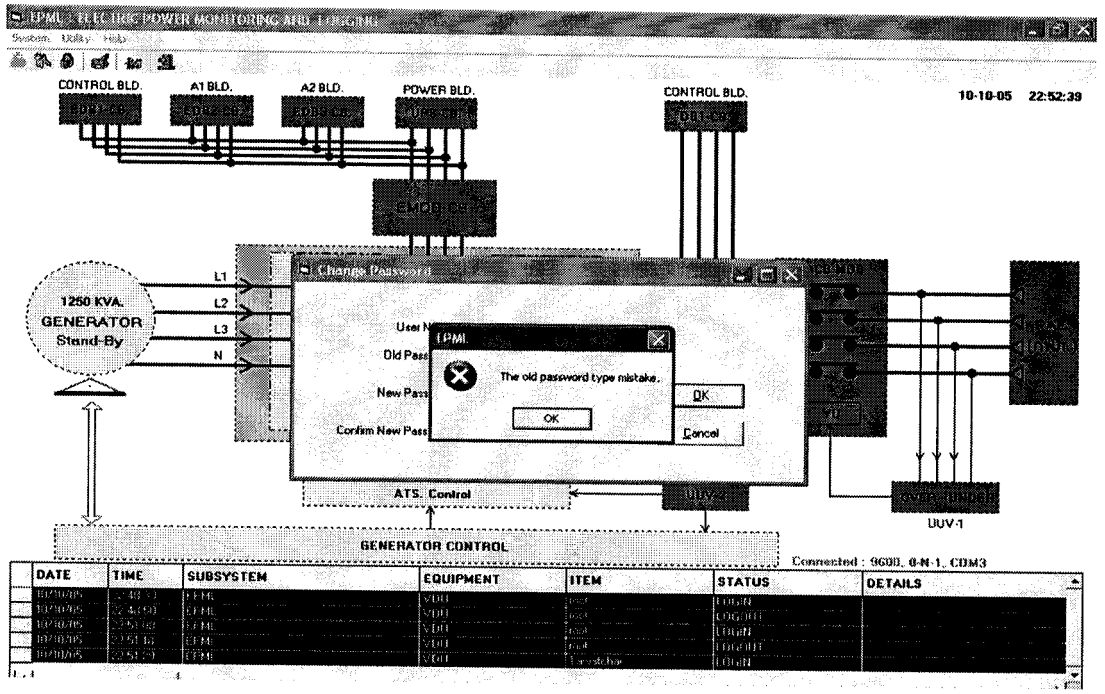
ภาพที่ 75 กรณีการพิมพ์ User Name และ Password ผิด

จากภาพที่ 75 เมื่อจะใช้งานเพื่อต้องการเข้าไปแก้ไขหรือค้นหาข้อมูลต่าง ๆ ผู้ใช้จะต้อง Login เข้าไป ถ้าหากถูกต้องและมีรายชื่ออยู่ในบัญชีผู้ใช้งาน ก็จะสามารถใช้ Menu bar หรือ Tool bar ตามระดับความสามารถในการใช้งานซึ่งจะถูกกำหนดโดยผู้ควบคุมระบบ (Admin) แต่ถ้าหาก Login ผิด หรือไม่มีข้อมูลอยู่ในบัญชีรายชื่อผู้ใช้งานโปรแกรมก็จะรายงานให้ทราบ ดังแสดงในภาพที่ 75

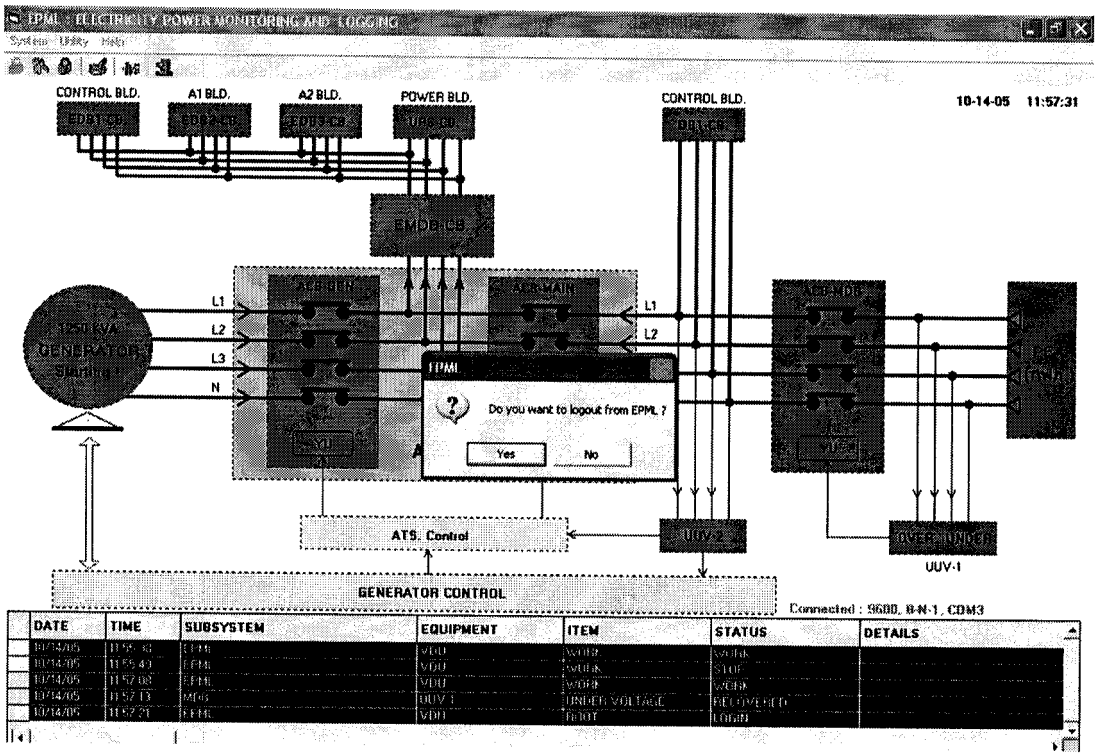


ภาพที่ 76 การเปลี่ยน Password

จากภาพที่ 76 ด้านบนเป็นหน้าสำหรับใช้เปลี่ยน Password โดยผู้ใช้งานที่มี Account แล้ว สามารถที่จะเปลี่ยน Password ด้วยตัวเอง เมื่อเห็นว่า Password เริ่มไม่ปลอดภัยแล้ว โดยจะต้องใส่ Password เก่าให้ถูกต้องเสียก่อน ถ้าใส่ Password เก่าไม่ถูกจะมีข้อความเตือน ดังภาพที่ 77

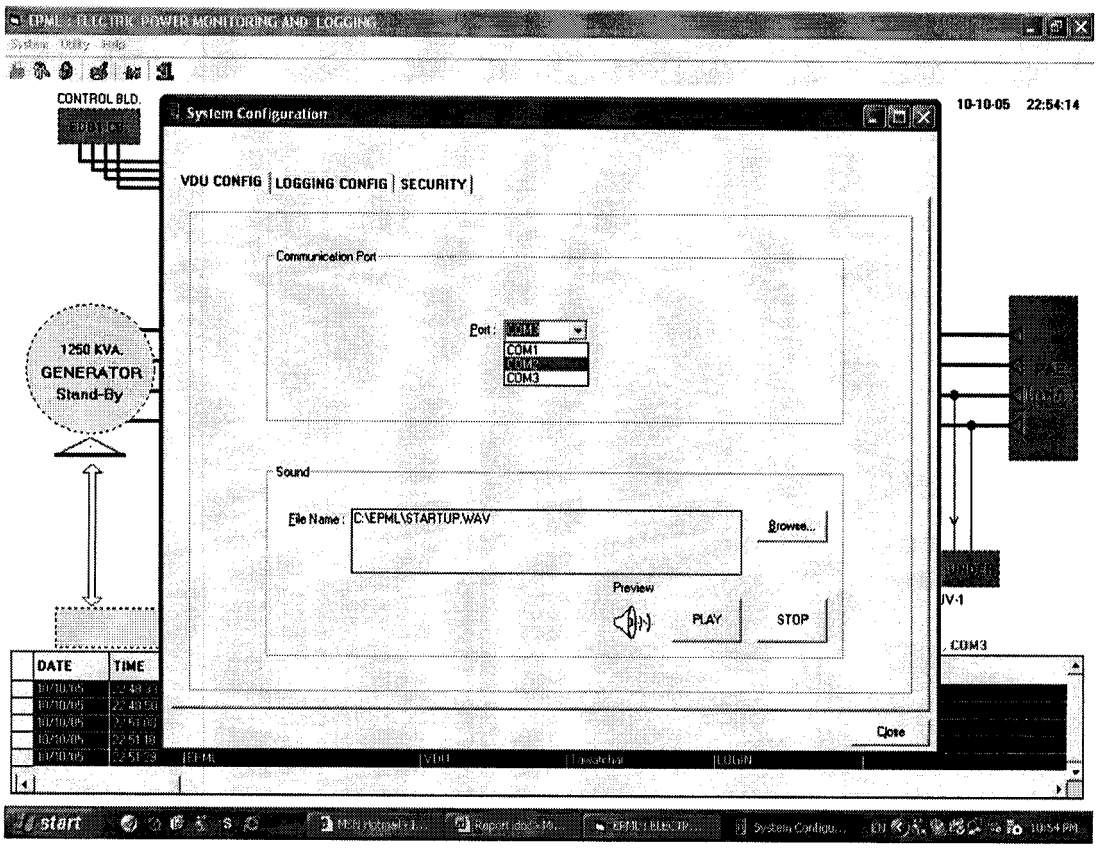


ภาพที่ 77 กรณีผู้ใช้งานพิมพ์ Password เก่าผิด



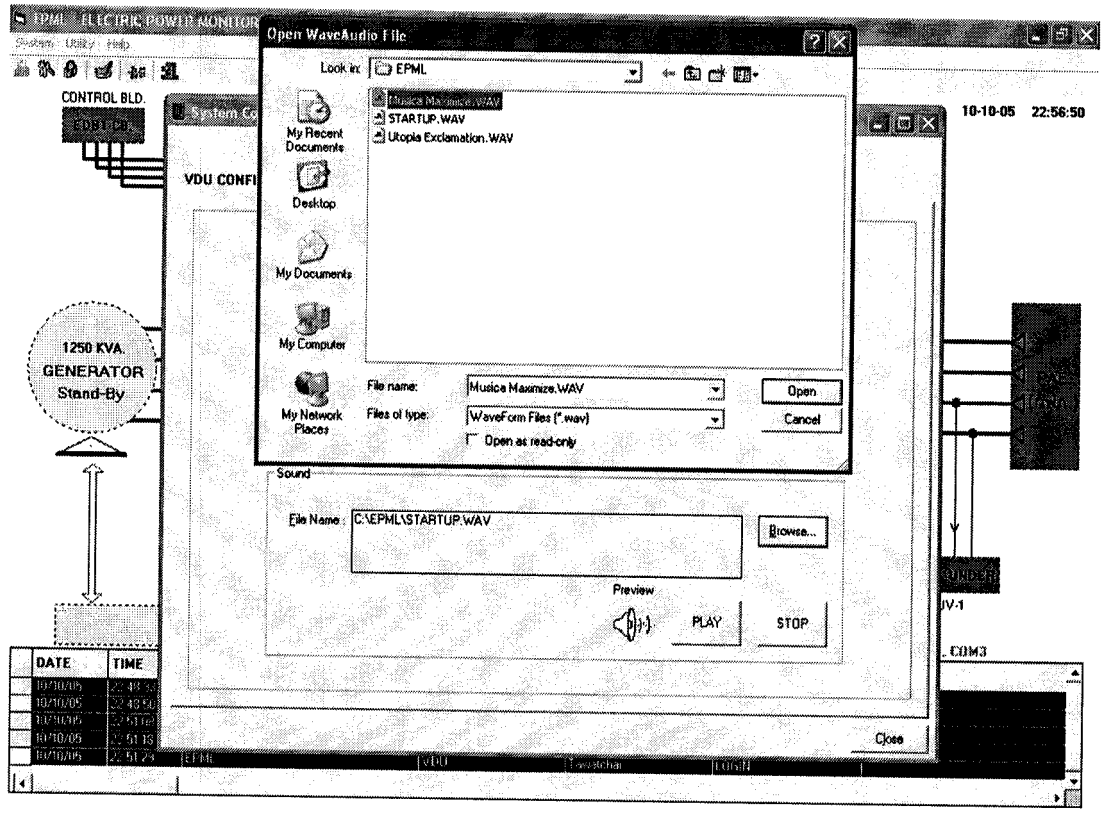
ภาพที่ 78 การ Log out

กรณีต้องการ Logout ออกจากระบบ ให้ Click ที่ปุ่ม Logout ใน Tool bar หรือ ใน Menu bar โปรแกรมก็ จะถามเพื่อยืนยันการ Logout ออกจากระบบอีกครั้งเพื่อความมั่นใจ ดังแสดงในภาพที่ 78

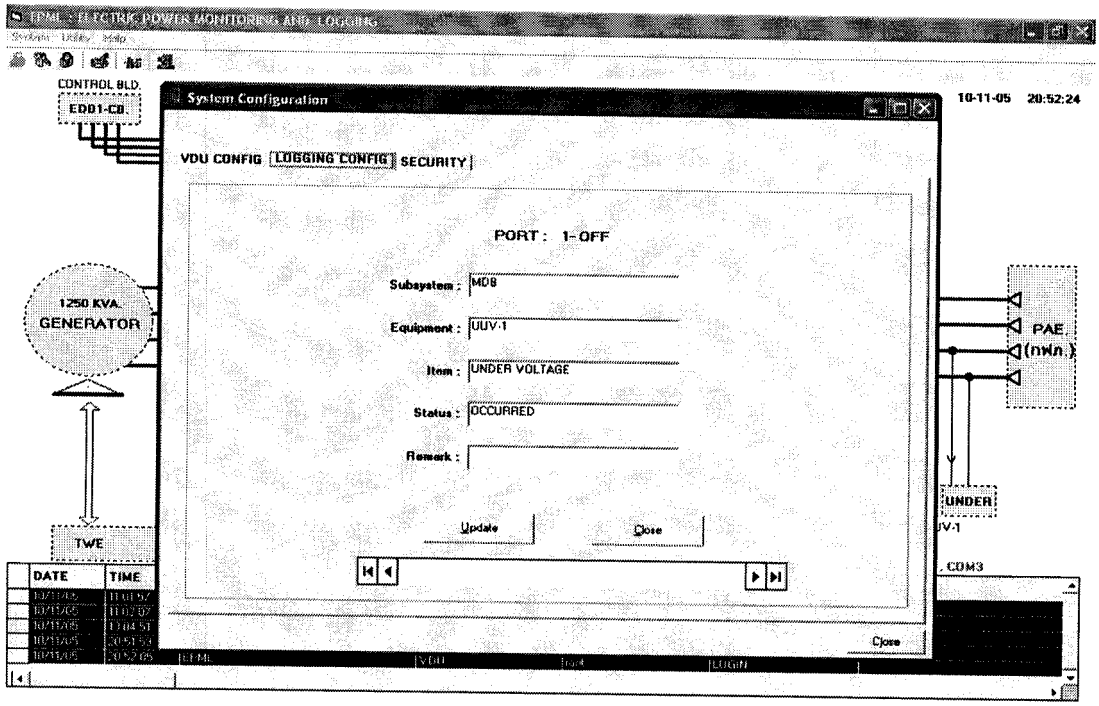


ภาพที่ 79 การกำหนดค่าต่าง ๆให้กับโปรแกรม

จากภาพที่ 79 เป็นหน้า System Configuration จะเป็นส่วนสำหรับ ผู้ควบคุมระบบเท่านั้นเข้าไปใช้งาน เช่นการเปลี่ยนพอร์ต สำหรับสื่อสารซึ่งเป็น RS-232C ว่าจะเลือกใช้พอร์ตไหน เลือกเสียง Alarm อะไร ตอนเมื่อมีเหตุการณ์ต่าง ๆ เกิดขึ้น จะมีเสียง Alarm เกิดขึ้นเพื่อให้เจ้าหน้าที่ ๆ เข้าเวร ได้ทราบว่ามีเหตุการณ์ผิดปกติ เสียง Alarm นี้สามารถเปลี่ยนแปลงได้ โดยใช้ไฟล์ที่มีนามสกุลเป็น .WAV ดังแสดงในภาพที่ 80

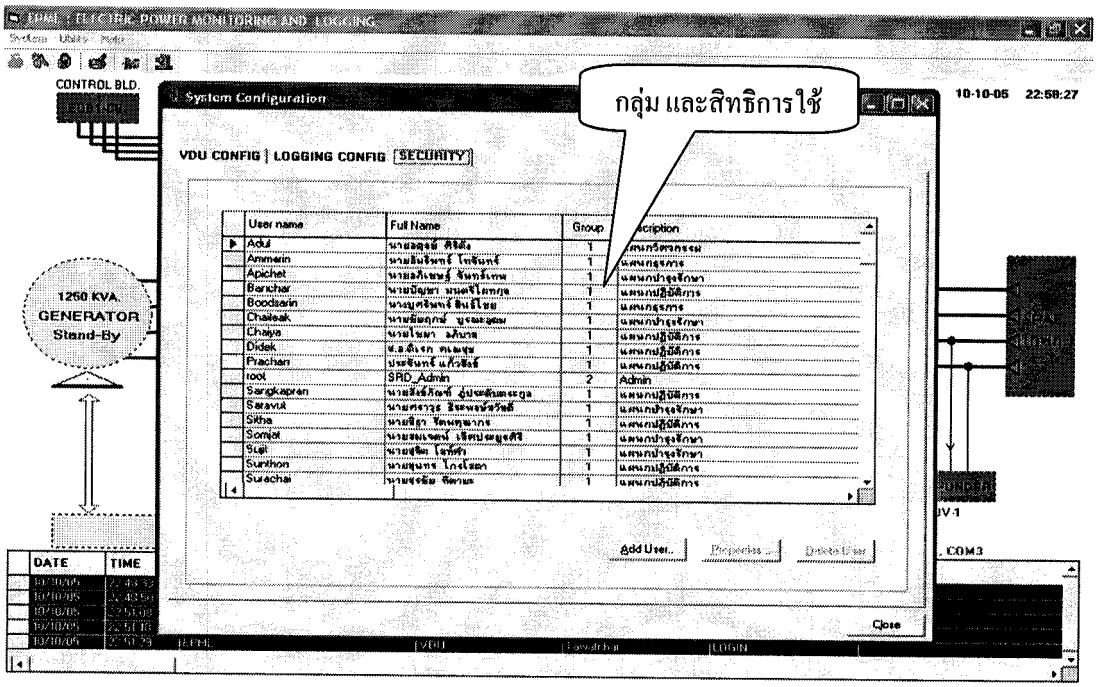


ภาพที่ 80 การหาไฟล์เสียงที่มีนามสกุล WAV ที่ต้องการ



ภาพที่ 81 การกำหนดค่า Logging ให้กับโปรแกรม

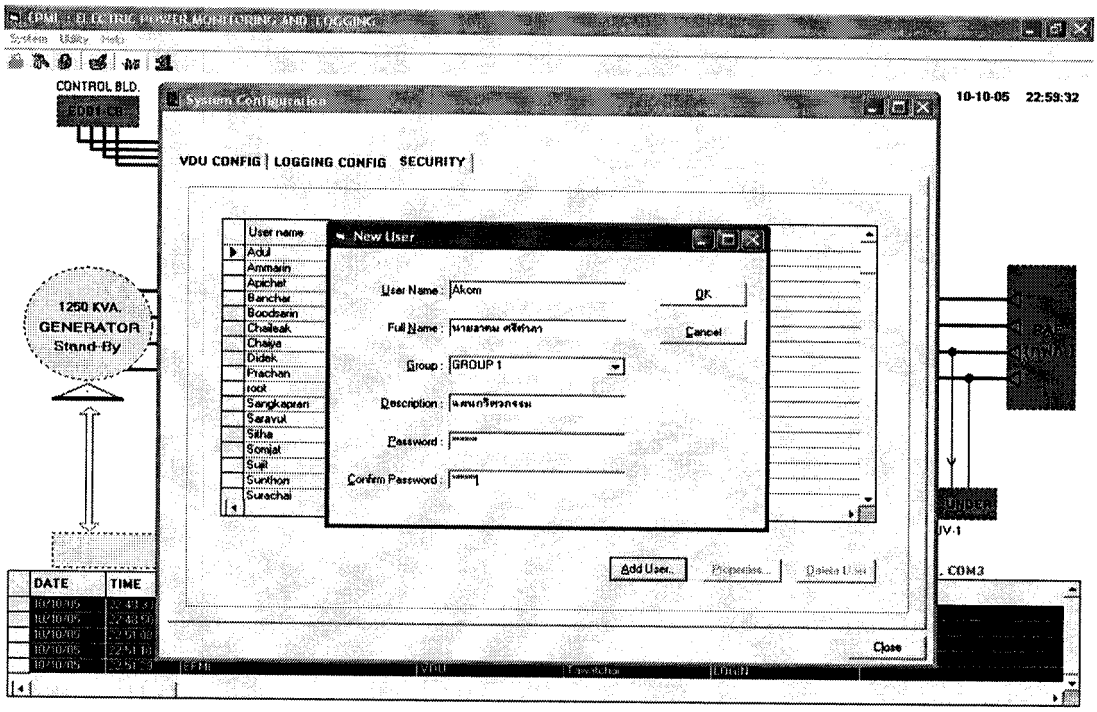
จากภาพที่ 81 ผู้ควบคุมระบบ (Admin) สามารถเปลี่ยนแปลงแก้ไขข้อความ Logging ได้ ซึ่งข้อความ Logging จะเป็นข้อความที่ใช้บันทึกลงแฟ้มข้อมูลเมื่อเกิดเหตุการณ์ที่อุปกรณ์ที่เฝ้าติดตามเกิดการเปลี่ยนแปลงสถานะการทำงาน ซึ่งที่ตัวกล่องอินเตอร์เฟส จะมี Port อยู่ทั้งหมด 12 Port สามารถที่จะกำหนดค่า Loggingได้เมื่อ Port ON และ เมื่อ Port OFF จากภาพตัวอย่างข้างบนเป็นการกำหนดค่า Config ของ Port ที่1 เมื่อ Port1 OFF จากตัวอย่างในภาพที่ 81 เป็นการกำหนดให้ Port1 เป็น ระบบย่อย (Subsystem) ของ MDB และ อุปกรณ์ (Equipment) ที่เฝ้าติดตามที่ได้ติดตัว Sensor ไว้คือ UUV-1 ซึ่งเป็นอุปกรณ์ที่จะคอยตรวจสอบแรงดันด้านต่ำ (Under Voltage) จึงกำหนดให้รายการ (Item) เท่ากับ Under Voltage ตัว UUV-1 จะตรวจสอบแรงดันไฟฟ้าจาก กฟภ. ว่าแรงดันต่ำกว่า 360 VAC. 3ϕ หรือไม่ และถ้าหากไม่มีแรงดันไฟฟ้าจาก กฟภ. หรือแรงดันไฟฟ้าต่ำกว่า 360 VAC. 3ϕ แล้ว Status (สถานะ) ที่ได้กำหนดไว้แล้วจะปรากฏเป็น OCCURRED (Port1 มีสภาวะเป็น OFF) และจะบันทึกค่าต่าง ๆ ที่ได้กำหนดไว้ทั้งเรคอร์ด ในภาพที่ 81 ลงในแฟ้มข้อมูล Logging.MDB



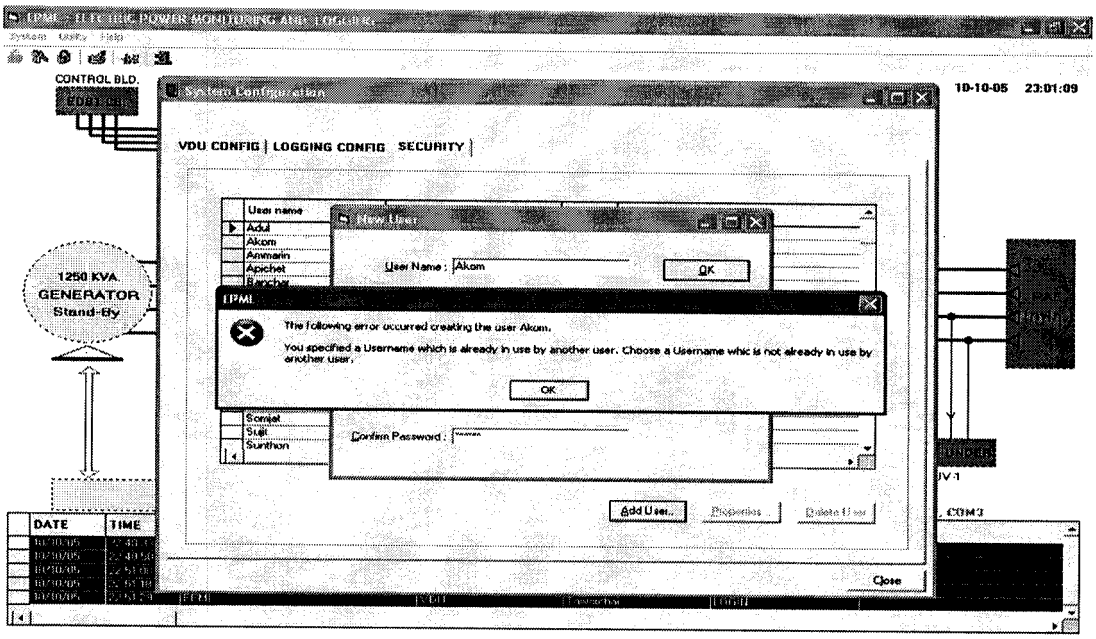
ภาพที่ 82 กลุ่ม และระดับสิทธิในการเข้าไปใช้งาน

จากภาพที่ 82 เป็นการแสดงกลุ่ม และระดับสิทธิในการเข้าไปใช้งาน ซึ่งส่วนนี้ผู้ดูแลระบบจะเป็นคนจัดการ เช่นเปลี่ยนค่าต่าง ๆ ของ User ที่มีรายชื่ออยู่ในบัญชีผู้ใช้งาน เช่นเปลี่ยน Group ซึ่งระดับการเข้าถึงข้อมูลเปลี่ยน User Nameเปลี่ยน Full Name,เปลี่ยน Description เปลี่ยน

Password และ ลบ User ออกจากบัญชีรายชื่อผู้ใช้งาน ส่วนภาพที่ 83 จะเป็นหน้าต่างสำหรับการเพิ่ม User ใหม่

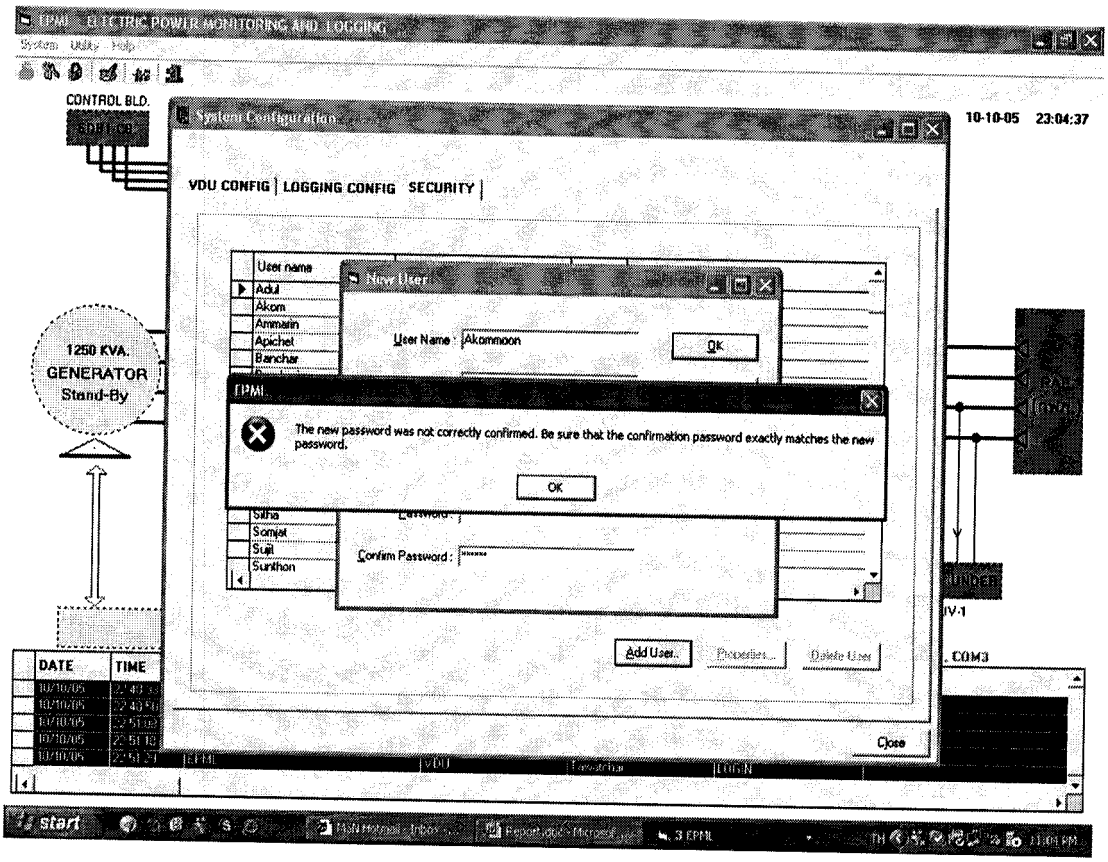


ภาพที่ 83 การเพิ่ม Userใหม่



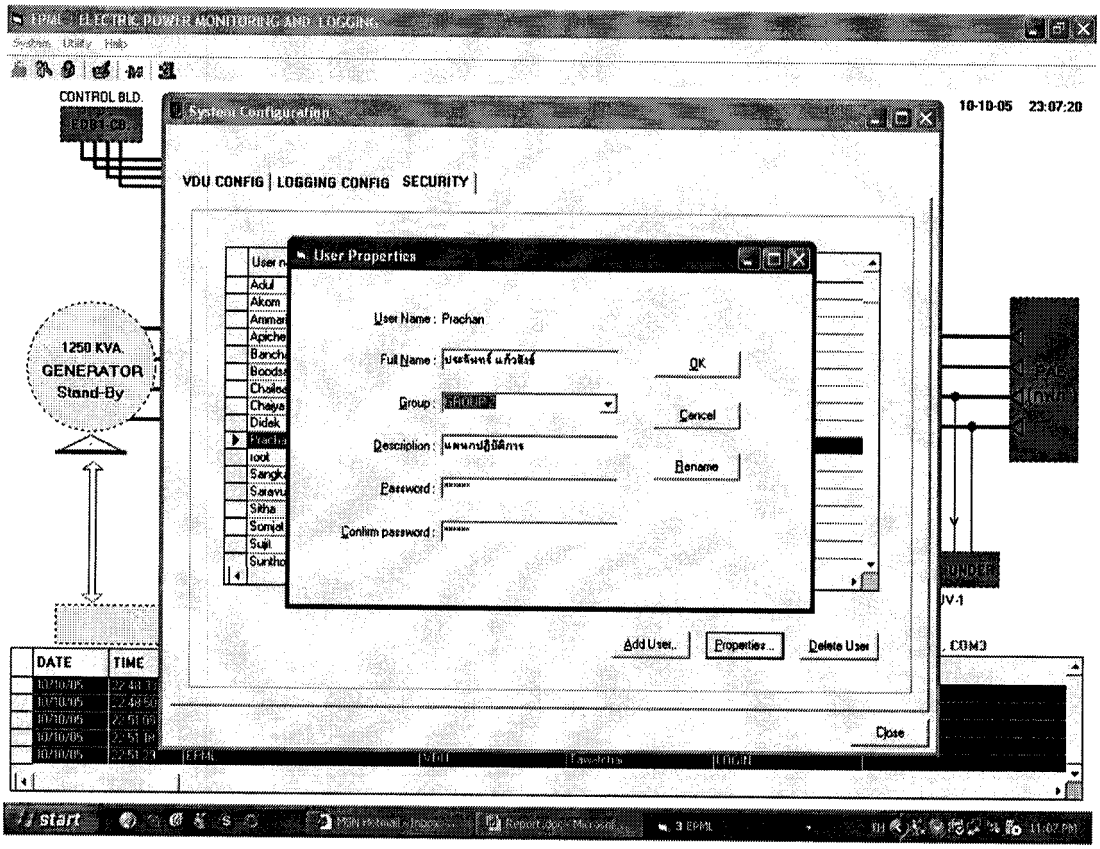
ภาพที่ 84 กรณีสร้าง User ซ้ำกันจะมีข้อความเตือน

จากภาพที่ 84 ถ้าหากสร้าง User ใหม่ที่มีชื่อซ้ำ User เก่าระบบจะไม่ยอม และข้อความเตือนดังภาพ



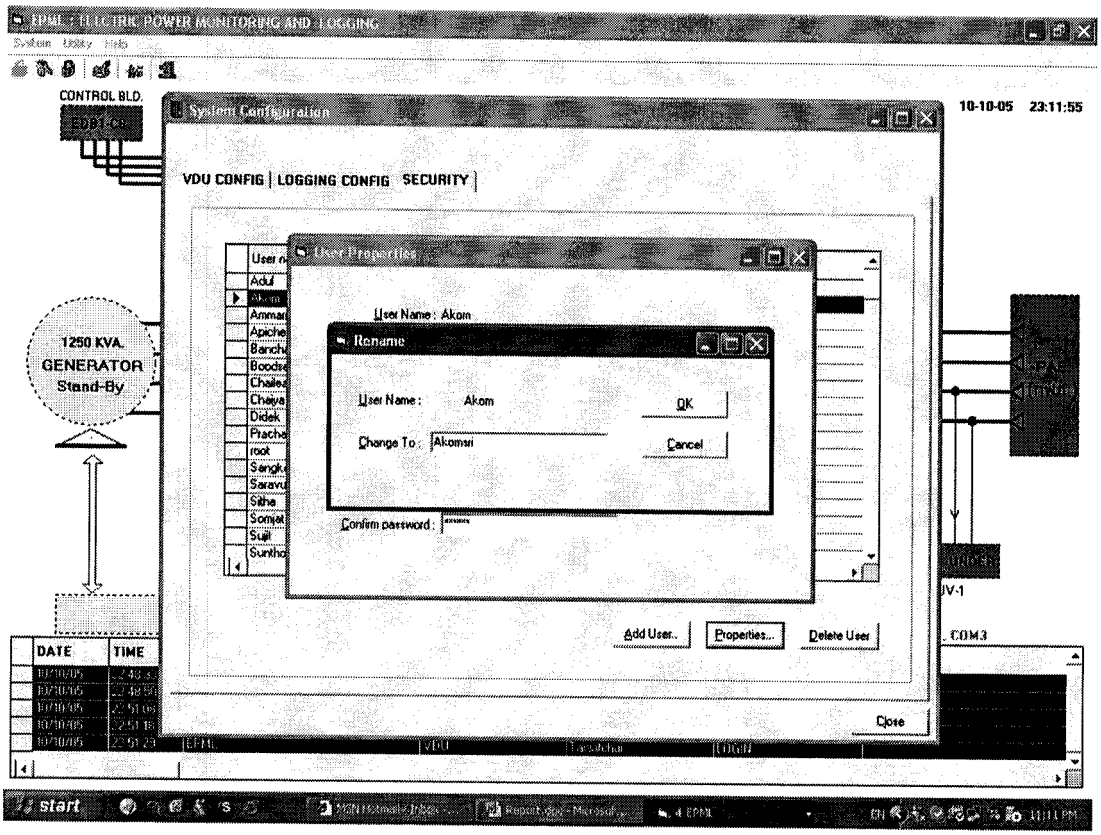
ภาพที่ 85 การกำหนดค่า Confirm Password ไม่ถูกต้อง

จากภาพที่ 85 หากป้อนข้อความที่ช่อง Password และ ช่อง Confirm Password ไม่เหมือนกัน ระบบจะไม่ยอมให้ทำงานต่อไป และจะมีข้อความเตือนขึ้น



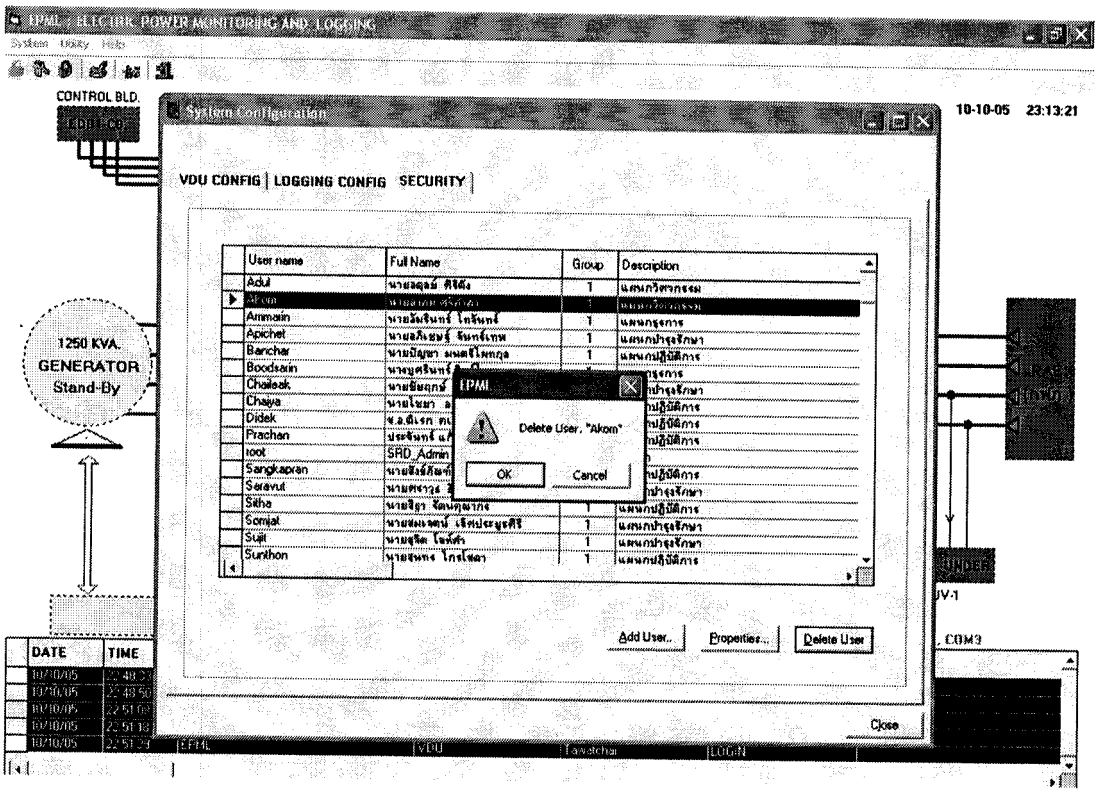
ภาพที่ 86 การแก้ไขรายละเอียดของ User

จากภาพที่ 86 หากต้องการแก้ไขรายละเอียด (Properties) ของแต่ละ User ก็สามารถทำได้โดยการ Click ที่ปุ่ม Properties ในแท็บ Security ในหน้าต่าง System Configuration ก็จะเปิดหน้าต่างใหม่ดังภาพด้านบน สามารถแก้ไขค่าต่าง ๆ ได้ เช่น ชื่อเต็ม (Full Name) กลุ่ม (Group) ซึ่งเป็นระดับการเข้าถึงข้อมูล, การอธิบายรายละเอียด (Description) แต่ละ User และ Password



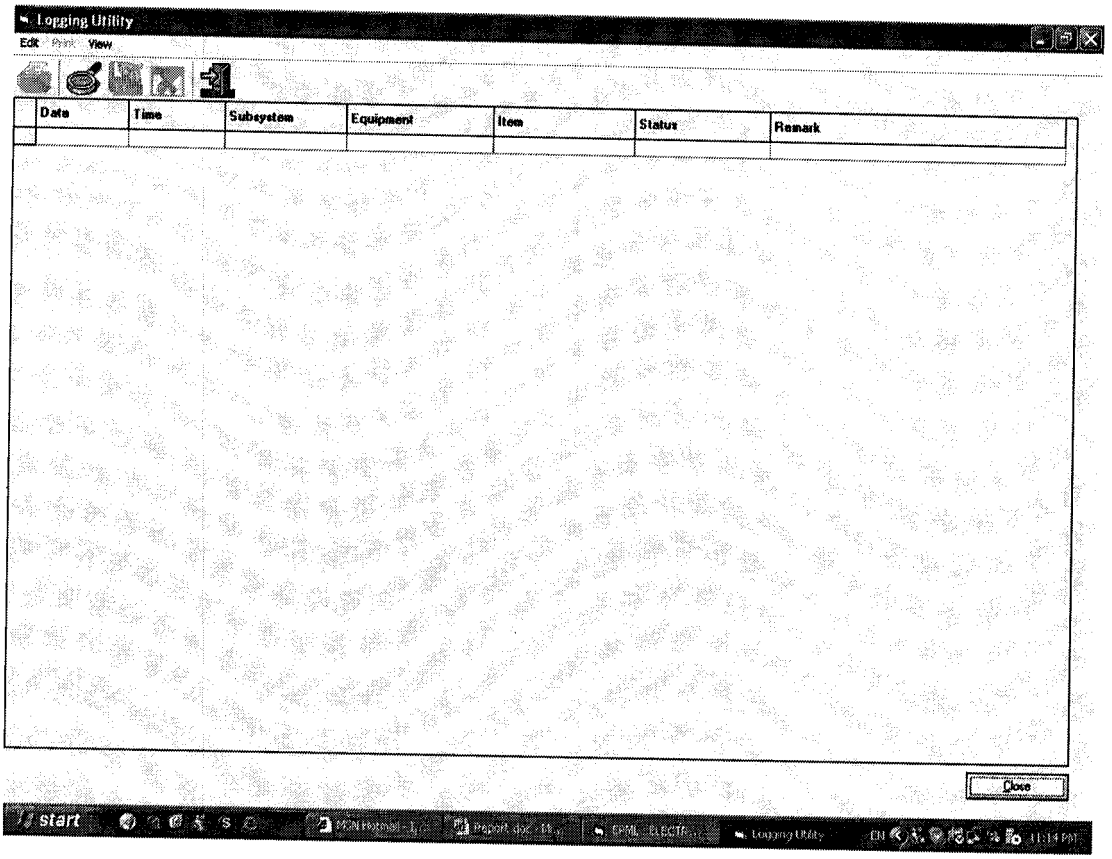
ภาพที่ 87 การเปลี่ยน User Name

จากภาพที่ 87 หากต้องการเปลี่ยน User Name ก็สามารถทำได้โดยกดปุ่ม Rename ในหน้าต่าง User Properties แล้วพิมพ์ชื่อใหม่ที่ต้องการตามภาพข้างบน



ภาพที่ 88 การลบ User

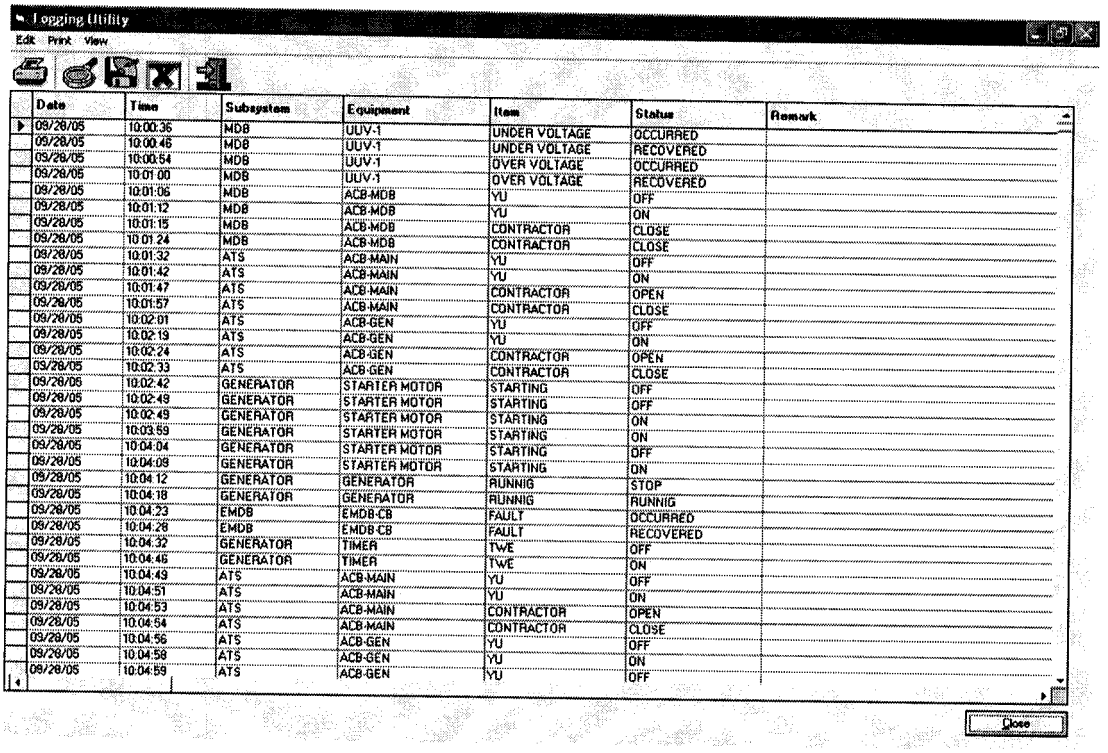
จากภาพที่ 88 หากต้องการลบ User ออกจากบัญชีรายชื่อผู้ใช้งานก็สามารถทำได้โดยให้ Click รายชื่อ User ที่ต้องการลบในตาราง ใน แท็บ Security และให้ Click ที่ปุ่ม Delete User ระบบจะถามเพื่อความแน่ใจที่จะลบอีกครั้ง ดังภาพด้านบน



ภาพที่ 89 การค้นหา Logging Utility

จากภาพที่ 89 Logging Utility เป็นหน้าสำหรับใช้บริหารจัดการกับข้อมูลต่าง ๆ ที่ได้บันทึกลงแฟ้มข้อมูลเรียบร้อยแล้วซึ่งเป็น Logging เหตุการณ์ของสถานะ การทำงานของอุปกรณ์ไฟฟ้ากำลังต่าง ๆ ที่เกิดขึ้นในอดีตที่ผ่านมา เช่น การค้นหาข้อมูลตามเงื่อนไข ตามช่วง วัน และเวลา การกำหนดระบบย่อยของอุปกรณ์ในการค้นหา การกำหนดอุปกรณ์ในการค้นหา การกำหนดรายการย่อยของอุปกรณ์ในการค้นหา การกำหนดสถานะของอุปกรณ์ในหารค้นหา เงื่อนไขเหล่านี้สามารถกำหนดค่าได้พร้อมกันทั้งหมดก็ได้ การคัดลอกข้อมูลจากการค้นหานำไปเปิดในโปรแกรม Microsoft Office Excel ก็ได้ การลบข้อมูลจากการค้นหาก็ได้ และ การพิมพ์รายงานจากการค้นหาก็ได้

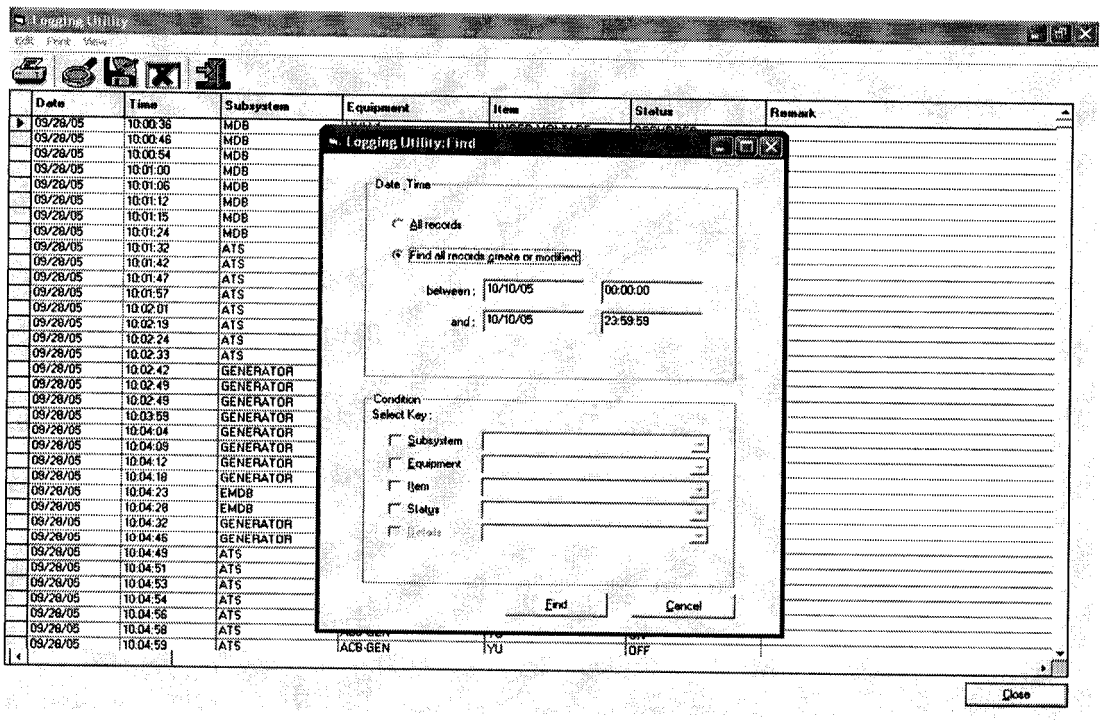
หากต้องการใช้งาน Logging Utility ให้ Click ที่ปุ่ม Logging Utility ใน Menu bar หรือTool bar สำหรับผู้ใช้ที่ Login เข้ามาเท่านั้นถึงจะใช้งานได้โดยหน้าแรกที่เข้ามาจะเป็นดังในภาพที่ 89



Date	Time	Subsystem	Equipment	Item	Status	Remark
09/26/05	10:00:36	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
09/26/05	10:00:46	MDB	UUV-1	UNDER VOLTAGE	RECOVERED	
09/26/05	10:00:54	MDB	UUV-1	OVER VOLTAGE	OCCURRED	
09/26/05	10:01:00	MDB	UUV-1	OVER VOLTAGE	RECOVERED	
09/26/05	10:01:06	MDB	ACB-MDB	YU	OFF	
09/26/05	10:01:12	MDB	ACB-MDB	YU	ON	
09/26/05	10:01:15	MDB	ACB-MDB	CONTRACTOR	CLOSE	
09/26/05	10:01:24	MDB	ACB-MDB	CONTRACTOR	CLOSE	
09/26/05	10:01:32	ATS	ACB-MAIN	YU	OFF	
09/26/05	10:01:42	ATS	ACB-MAIN	YU	ON	
09/26/05	10:01:47	ATS	ACB-MAIN	CONTRACTOR	OPEN	
09/26/05	10:01:57	ATS	ACB-MAIN	CONTRACTOR	CLOSE	
09/26/05	10:02:01	ATS	ACB-GEN	YU	OFF	
09/26/05	10:02:19	ATS	ACB-GEN	YU	ON	
09/26/05	10:02:24	ATS	ACB-GEN	CONTRACTOR	OPEN	
09/26/05	10:02:33	ATS	ACB-GEN	CONTRACTOR	CLOSE	
09/26/05	10:02:42	GENERATOR	STARTER MOTOR	STARTING	OFF	
09/26/05	10:02:49	GENERATOR	STARTER MOTOR	STARTING	OFF	
09/26/05	10:02:49	GENERATOR	STARTER MOTOR	STARTING	ON	
09/26/05	10:03:59	GENERATOR	STARTER MOTOR	STARTING	ON	
09/26/05	10:04:04	GENERATOR	STARTER MOTOR	STARTING	OFF	
09/26/05	10:04:09	GENERATOR	STARTER MOTOR	STARTING	ON	
09/26/05	10:04:12	GENERATOR	GENERATOR	RUNNIG	STOP	
09/26/05	10:04:18	GENERATOR	GENERATOR	RUNNIG	RUNNIG	
09/26/05	10:04:23	EMDB	EMDB-CB	FAULT	OCCURRED	
09/26/05	10:04:28	EMDB	EMDB-CB	FAULT	RECOVERED	
09/26/05	10:04:32	GENERATOR	TIMER	TWE	OFF	
09/26/05	10:04:46	GENERATOR	TIMER	TWE	ON	
09/26/05	10:04:49	ATS	ACB-MAIN	YU	OFF	
09/26/05	10:04:51	ATS	ACB-MAIN	YU	ON	
09/26/05	10:04:53	ATS	ACB-MAIN	CONTRACTOR	OPEN	
09/26/05	10:04:54	ATS	ACB-MAIN	CONTRACTOR	CLOSE	
09/26/05	10:04:56	ATS	ACB-GEN	YU	OFF	
09/26/05	10:04:58	ATS	ACB-GEN	YU	ON	
09/26/05	10:04:59	ATS	ACB-GEN	YU	OFF	

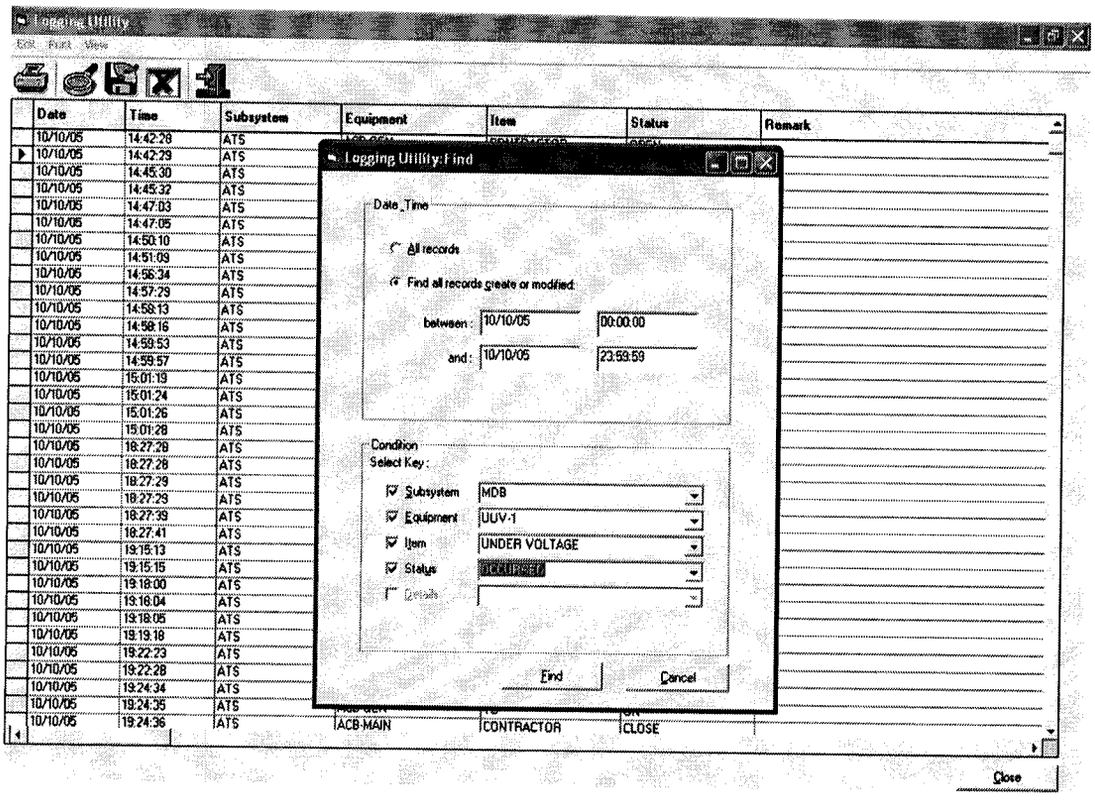
ภาพที่ 90 ผลจากการค้นหา All Records

ภาพที่ 90 แสดงผลจากการค้นหาข้อมูลทั้งหมดที่ถูกบันทึกในแฟ้มข้อมูลโดยไม่มีเงื่อนไขในการค้นหา



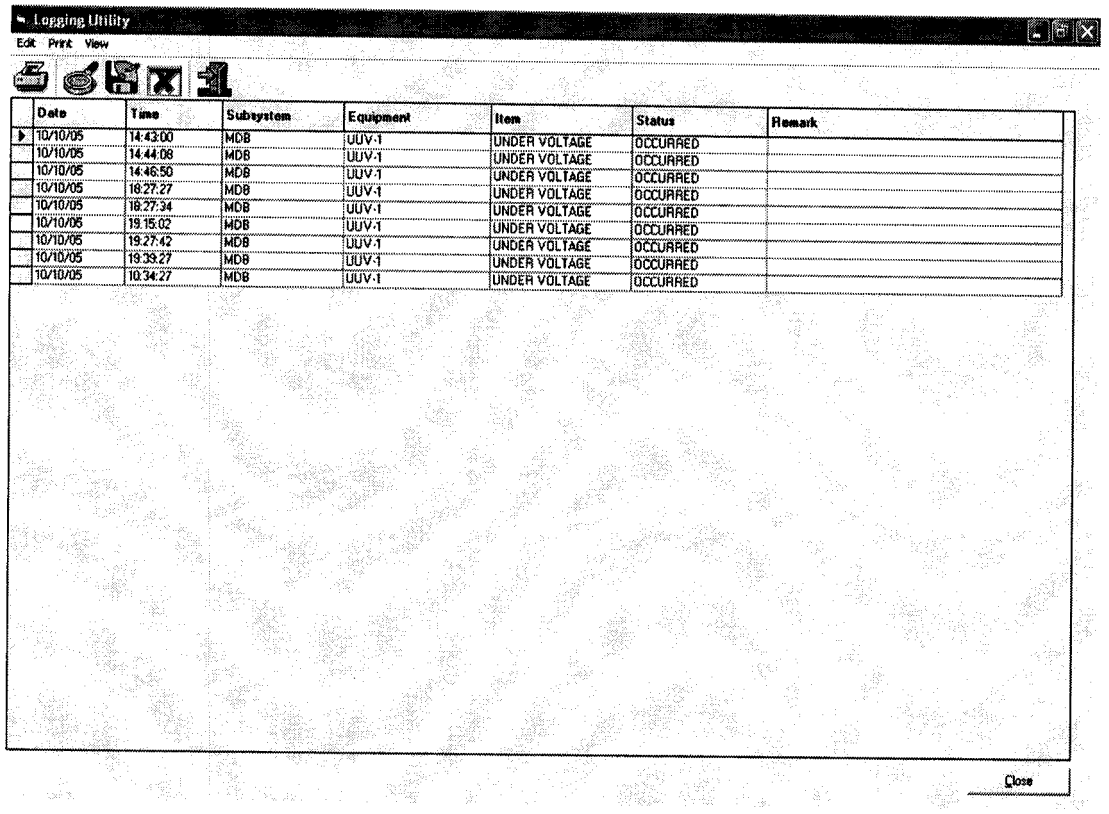
ภาพที่ 91 การค้นหา Logging ตามช่วง วัน และ เวลาที่ต้องการ

จากภาพที่ 91 หากต้องการค้นหาข้อความ Logging ในแฟ้มข้อมูล ให้เลือก Find all records create or modified จากภาพตัวอย่างข้างบนเป็นการค้นหาเหตุการณ์สถานะการทำงานของอุปกรณ์ทั้งหมดในวันที่ 10 ตุลาคม 2005 ทั้งวัน คือ จากช่วงเวลา 00:00:00 ถึง 23:59:59



ภาพที่ 92 การค้นหาตามเงื่อนไข

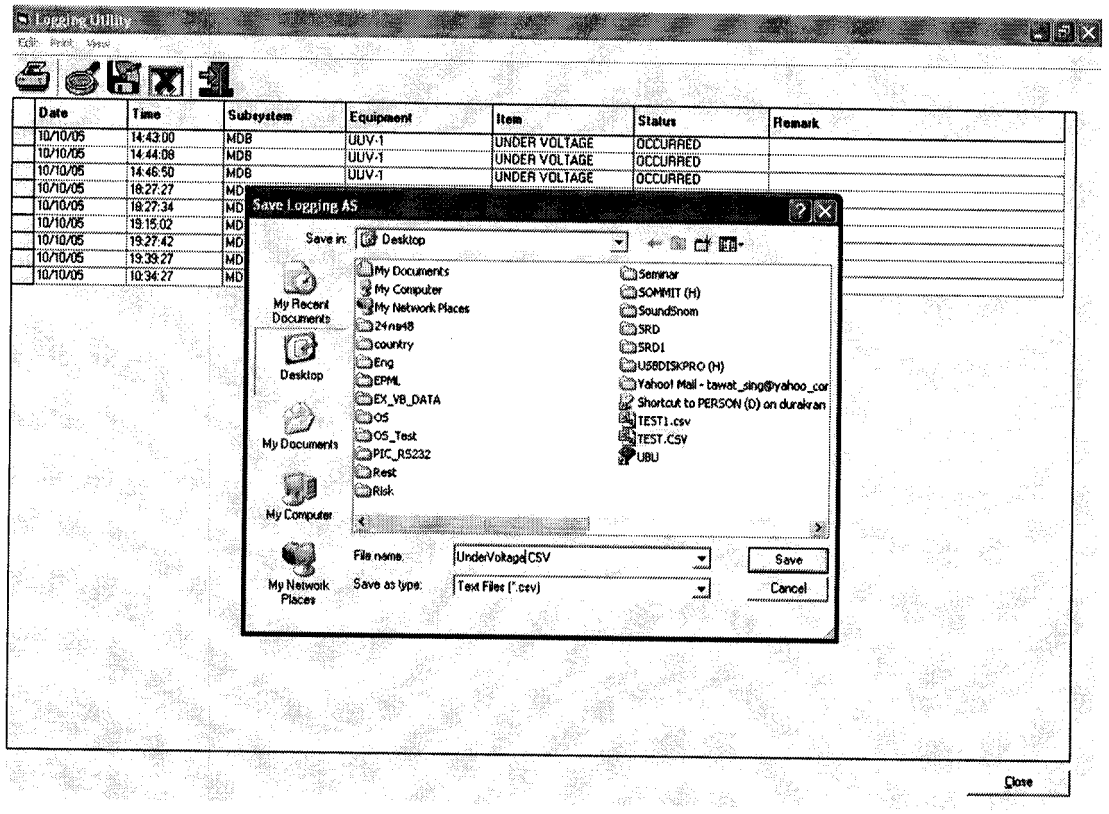
จากภาพที่ 92 หากต้องการค้นหาเหตุการณ์สถานะการทำงานของอุปกรณ์ตามเงื่อนไขที่ต้องการซึ่งเป็นการหา Subsystem = MDB , Equipment = UUV-1, Item = UNDER VOLTAGE, Status = OCCURRE วันที่ 10 ตุลาคม 2005 และเวลาระหว่าง 00:00:00 ถึง 23:59:59



Date	Time	Subsystem	Equipment	Item	Status	Remark
10/10/05	14:43:00	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	14:44:08	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	14:46:50	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	16:27:27	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	16:27:34	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	19:15:02	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	19:27:42	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	19:39:27	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	
10/10/05	10:34:27	MDB	UUV-1	UNDER VOLTAGE	OCCURRED	

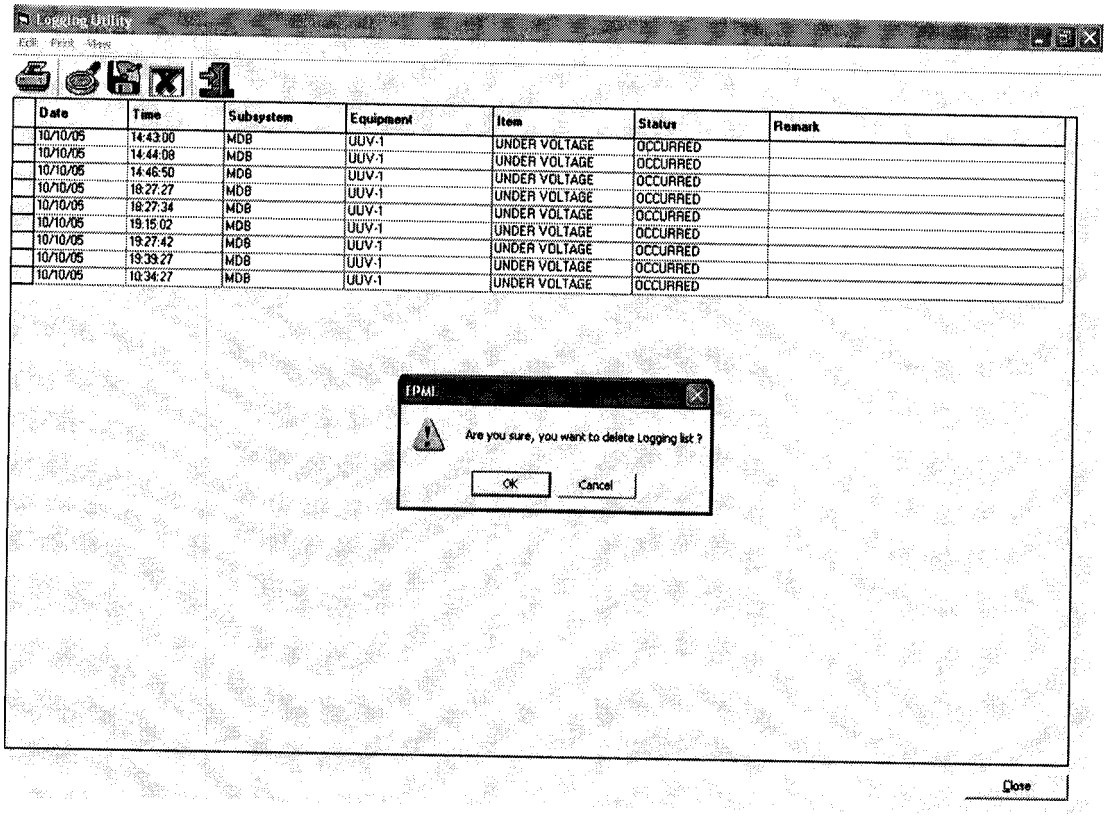
ภาพที่ 93 ผลจากการค้นหาตามเงื่อนไข

จากภาพที่ 93 เมื่อป้อนข้อมูลตามเงื่อนไขที่ต้องการครบถ้วนแล้วให้กดปุ่ม Find ระบบก็จะทำการค้นหาข้อมูลในแฟ้มข้อมูลตามเงื่อนไขต่าง ๆ ตามที่ได้กรอกไว้และแสดงผลการค้นหาออกมา ดังภาพที่ 93



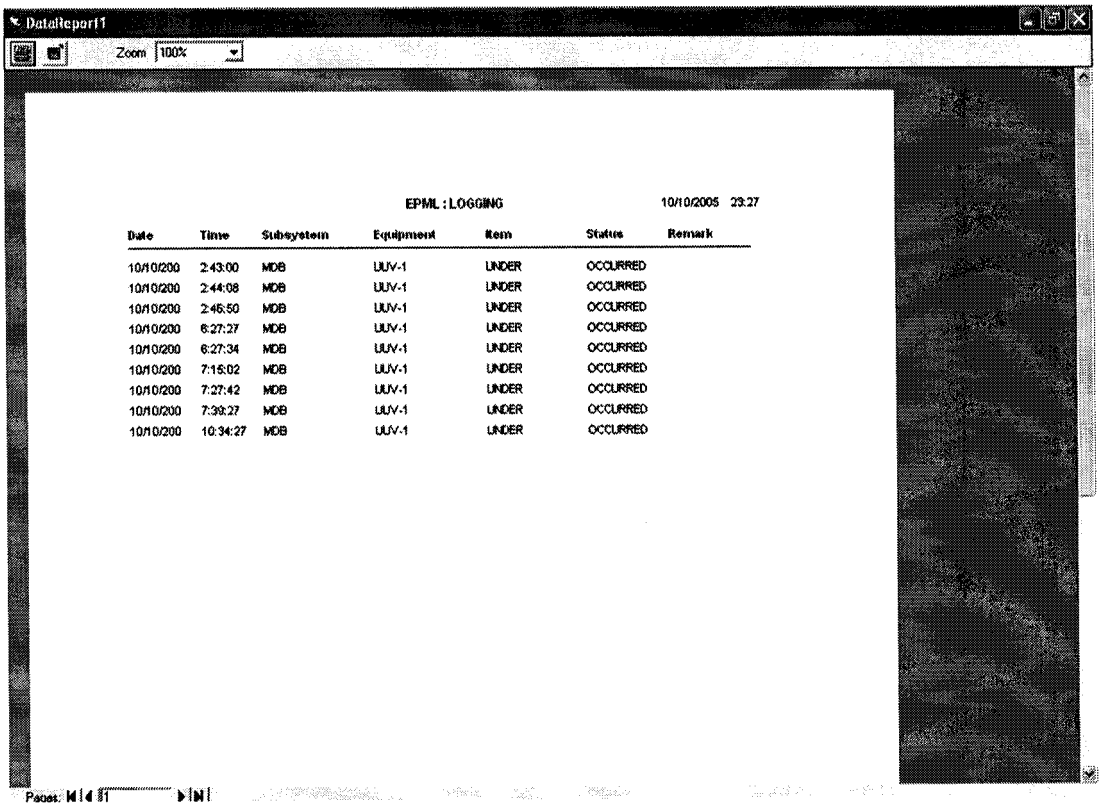
ภาพที่ 94 ผลการ Copy ผลจากที่ได้ค้นหามาลงใน Disk เพื่อนำไปเปิดใน Excel

จากภาพที่ 94 หลังจากค้นหาข้อมูลตามเงื่อนไขที่ต้องการแล้ว ยังสามารถนำข้อมูลที่ได้จากการค้นหาไปใช้ในวัตถุประสงค์อื่นได้อีกเช่น นำไปทำรายงานต่าง ๆ ไปทำข้อมูลในการทำสถิติเป็นต้น ซึ่งข้อมูลที่ได้จากการค้นหาเมื่อทำการ Copy และ Save ไปยัง Drive ที่ต้องการจะมีนามสกุลเป็น .CSV ซึ่งสามารถใช้ Program Excel เปิดได้



ภาพที่ 95 ผลการลบ Records ที่ได้จากการค้นหา

จากภาพที่ 95 หากต้องการลบ Records ที่ได้จากการค้นหามา ก็สามารถทำได้โดยการ Click ที่ปุ่ม Delete ของ Tool bar หรือ Menu bar ในหน้าต่าง Logging Utility ระบบจะมีข้อความเตือนเพื่อยืนยันอีกครั้งเพื่อความมั่นใจ



ภาพที่ 96 ผลการพิมพ์รายงานออกทางเครื่องพิมพ์

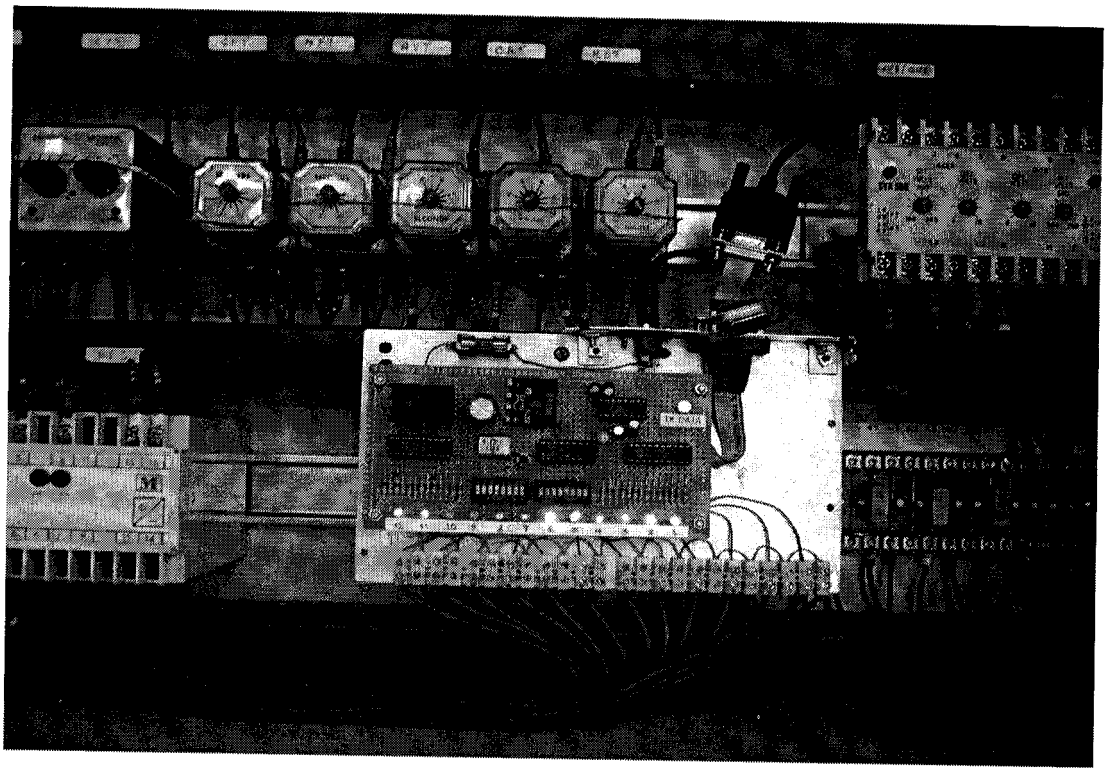
จากภาพที่ 96 ผลจากการค้นหาตามเงื่อนไข หากต้องการพิมพ์ออกทางเครื่องพิมพ์ ก็สามารถทำได้ โดย Click ที่ปุ่ม Print ใน Tool bar หรือ Menu bar ก็ได้โปรแกรมก็จะ พิมพ์ Records ต่าง ๆ ที่ได้จากการค้นหาออกกระดาษ

4.3.3 ติดตั้งใช้งานจริง

เมื่อตรวจสอบข้อบกพร่องต่าง ๆ และแก้ไขเรียบร้อยแล้ว ขั้นตอนต่อมาเป็นการติดตั้งทดลองใช้งานจริง

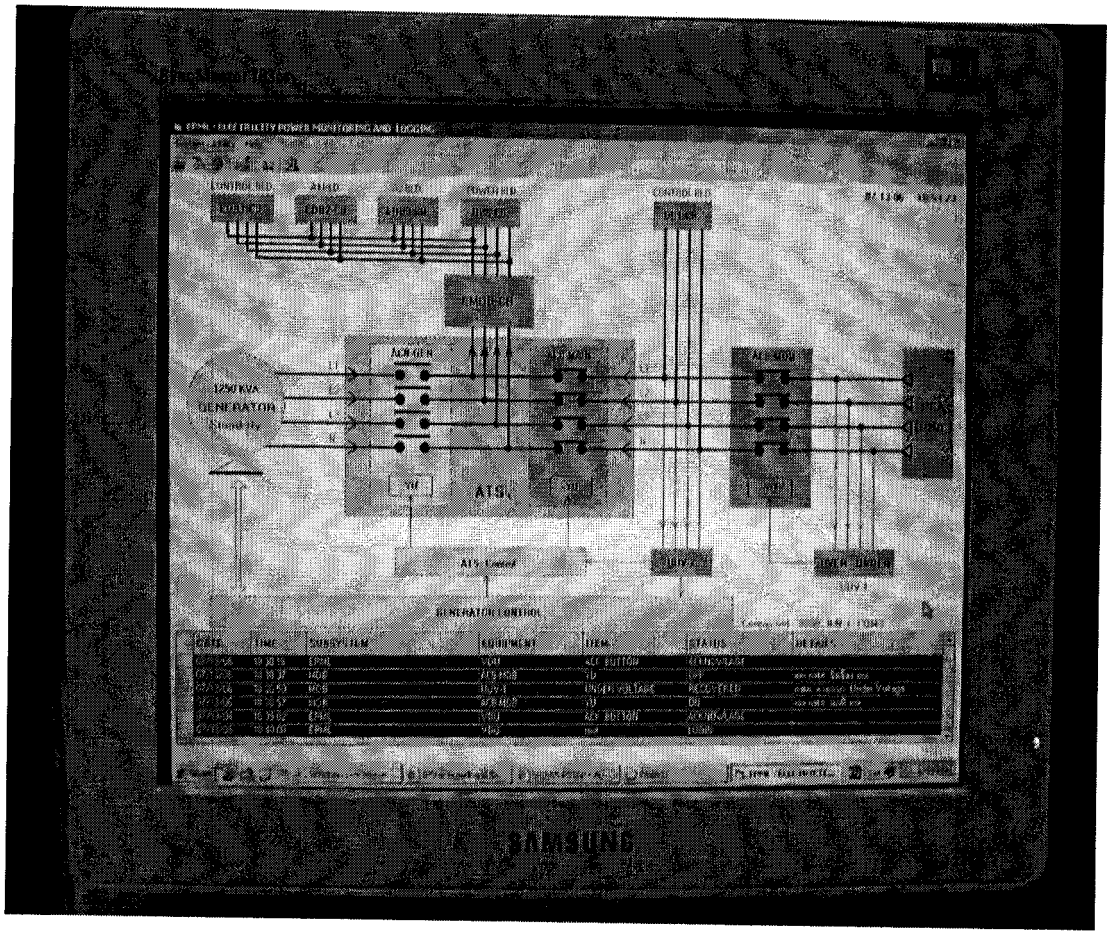
4.3.3.1 เดินสายโทรศัพท์ 1 คู่สาย ระหว่างกล่องอินเตอร์เฟส ในอาคารไฟฟ้ากำลัง ไปยังห้องคอนโทรล ภายในอาคารสถานีดาวเทียมฯ ซึ่งมีระยะทางประมาณ 60 – 80 เมตร เพื่อเชื่อมเข้ากับพอร์ต RS-232C ของคอมพิวเตอร์ที่ใช้เปิดโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE) โดยให้ต่อขา 3 (Transmit Data) ของกล่องอินเตอร์เฟส เข้ากับ
ขา 2 (Receive Data) ของคอมพิวเตอร์ที่ใช้เปิดโปรแกรมติดต่อกับผู้ใช้งานในห้องควบคุม ส่วนขา 5 ซึ่งเป็นขา Ground ของทั้งสองให้ต่อถึงกัน

4.3.3.2 ติดตั้งชุดกล่องอินเตอร์เฟส ที่อาคารการกำลัง ฯ และต่อ Sensor ตามจุดต่าง ๆ ตามที่ได้ออกแบบไว้ในบทที่ 3 เข้ากับพอร์ตต่าง ๆ ใน กล่องอินเตอร์เฟสดังภาพที่ 97

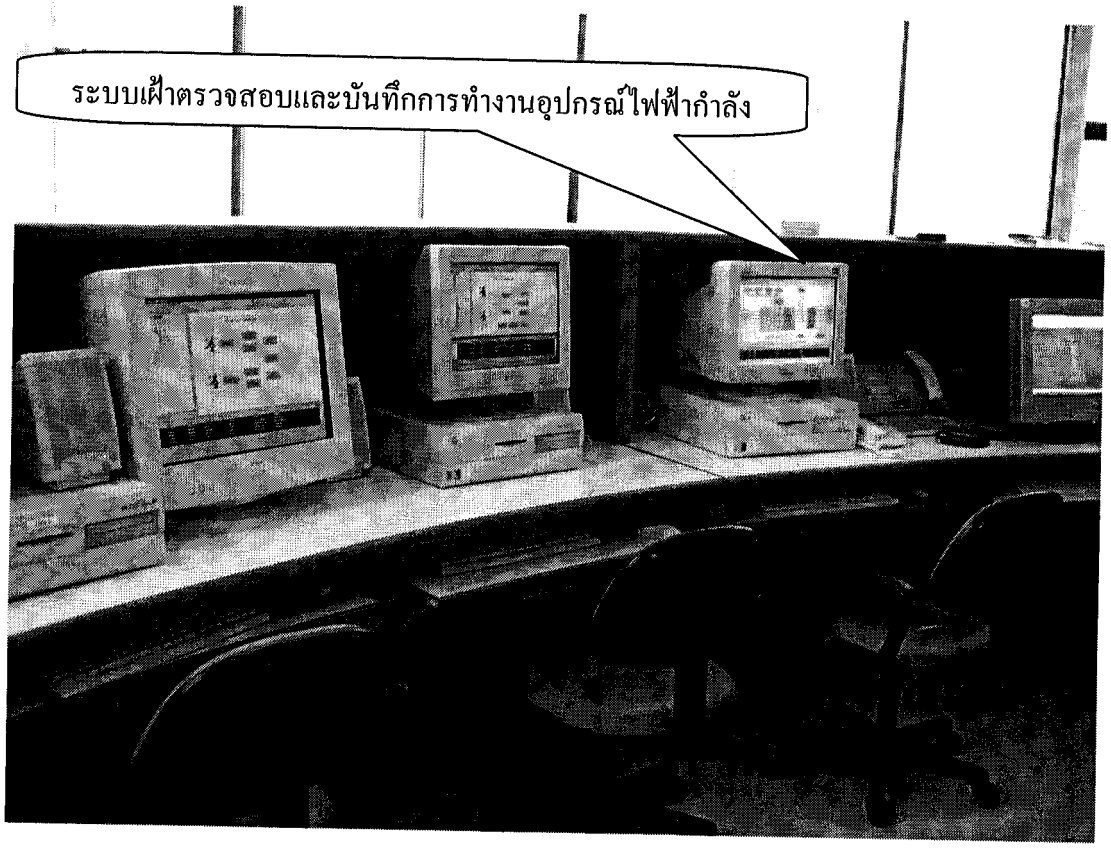


ภาพที่ 97 การติดตั้งกล่องอินเตอร์เฟส

4.3.3.3 ติดตั้งโปรแกรมติดต่อกับผู้ใช้งาน(GRAPHIC USER INTERFACE) โดยในห้องคอนโทรล ภายในอาคารสถานีดาวเทียมฯ พร้อมทั้งต่อสายสัญญาณที่มาจาก กล่องอินเตอร์เฟส ให้อาคารการกำลัง โดยต่อขา 3 (Transmit Data.) จาก กล่องอินเตอร์เฟส เข้า ขา 2 (Receive Data.) พอร์ตอนุกรม RS-232C ของคอมพิวเตอร์ที่เปิดโปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE) เมื่อต่อเรียบร้อยแล้ว อาจจะตรวจสอบความถูกต้องโดยการกดปุ่ม Test Swith ในกล่องอินเตอร์เฟส ที่อาคารไฟฟ้ากำลัง โปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE) ในห้องคอนโทรล ภายในอาคารสถานีดาวเทียมฯ ก็จะเปลี่ยนแปลงสถานะตามที่เรากดปุ่ม Test Swith ที่กล่องอินเตอร์เฟส ถ้าหากไม่เกิดข้อผิดพลาดใด ๆ



ภาพที่ 98 โปรแกรมติดต่อกับผู้ใช้งาน (GRAPHIC USER INTERFACE) ในห้องควบคุม



ภาพที่ 99 ระบบเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้ากำลัง เป็นส่วนหนึ่งของสถานีดาวเทียมสิรินธร

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

เป็นการพัฒนาระบบการเฝ้าตรวจสอบและบันทึกการทำงานอุปกรณ์ไฟฟ้าใหม่ทั้งหมดเพื่อให้เจ้าหน้าประจำสถานีดาวเทียมได้ใช้งาน เพื่อเพิ่มประสิทธิภาพการทำงานให้มีความน่าเชื่อถือมากยิ่งขึ้น ง่ายต่อการควบคุมอุปกรณ์ต่าง ๆ ภายในสถานีดาวเทียม ซึ่งมีห้องควบคุมอุปกรณ์ (Control room) เป็นศูนย์กลางในการควบคุมอุปกรณ์ดาวเทียม และอุปกรณ์ต่าง ๆ ภายในสถานี และได้ทดลองใช้งานในเบื้องต้นที่สถานีดาวเทียมสิรินธรแล้ว ได้ช่วยอำนวยความสะดวกต่อการปฏิบัติงาน ของเจ้าหน้าที่ประจำสถานีดาวเทียม โดยเฉพาะเจ้าหน้าที่เวรปฏิบัติการเมื่อเกิดเหตุการณ์ผิดปกติเกี่ยวกับระบบไฟฟ้ากำลัง สามารถตรวจสอบการทำงานของอุปกรณ์ที่หน้าจอมอนิเตอร์ได้ภายในห้องควบคุมอุปกรณ์เพียงจุดเดียว ทำให้รู้สาเหตุของปัญหา และสามารถช่วยในการตัดสินใจ ในการแก้ไขปัญหา หรือประสานงานกับหน่วยงานอื่น ที่เกี่ยวข้องได้รวดเร็วและถูกต้อง ในการพัฒนาครั้งนี้ใช้งบประมาณที่น้อยมากเพียงไม่กี่ร้อยบาท

5.2 ข้อเสนอแนะ

จากการทดสอบและใช้งาน ระบบสามารถทำงานได้ตรงตามขั้นตอนและเงื่อนไขตามที่ได้ออกแบบไว้ แต่ถ้าหากต้องการให้ระบบนี้ดีขึ้นกว่านี้อีก ควรมีการพัฒนาปรับปรุงแก้ไขดังต่อไปนี้

5.2.1 ระบบควรบอกรายละเอียดการทำงานอุปกรณ์ไฟฟ้ากำลังได้มากกว่านี้ เนื่องจากข้อจำกัดของไมโครคอนโทรลเลอร์ PIC16F84 ซึ่งเป็น ไมโครคอนโทรลเลอร์ขนาดเล็ก โดยมีพอร์ตให้ใช้งานเพียง 13 พอร์ต เท่านั้น โดย 1 พอร์ต ใช้สำหรับส่งข้อมูล (Tx.Data) จึงเหลือพอร์ต ใช้ในการตรวจสอบอุปกรณ์ไฟฟ้ากำลังแค่ 12 พอร์ต ผู้พัฒนาจึงเลือกตรวจสอบอุปกรณ์จุดที่สำคัญ ๆ เท่านั้น หากต้องการรายละเอียดมากกว่านี้อีก ควรเพิ่มไมโครคอนโทรลเลอร์ PIC16F84 เข้าไปอีก เพื่อเพิ่มพอร์ตในการตรวจสอบ หรือเปลี่ยนเป็น ไมโครคอนโทรลเลอร์เบอร์อื่นที่มี พอร์ต ให้ใช้งานมากกว่านี้ เช่นเบอร์ PIC16F87 แต่ PIC16F84 ยังมีข้อดีอยู่ คือราคาถูกมาก

5.2.2 ระบบนี้ควรพัฒนาด้วย Microsoft Visual Studio .NET ซึ่งจะเป็นพัฒนาแบบ OOP. อย่างสมบูรณ์แต่ใน Microsoft Visual Studio .NET ยังไม่มีเครื่องมือใช้ในการพัฒนา พอร์ต

อนุกรม (RS-232C) ดังนั้น ผู้พัฒนาจึงเลือกใช้ Microsoft Visual Basic 6.0 ในการพัฒนาซึ่งมีเครื่องมือในการพัฒนา พอร์ต RS-232C อยู่เรียบร้อยแล้ว ใน Version ต่อไป ของMicrosoft Visual Studio .NET น่าจะเพิ่มเครื่องมือ พอร์ตอนุกรมเข้ามา

5.2.3 เนื่องจากเวลามีจำกัดระบบที่พัฒนาขั้นมานี้ยังไม่มีในส่วนให้ผู้ใช้งานสามารถติดต่อผ่านระบบ Internet ได้ ดังนั้น ควรจะทำการพัฒนาส่วนที่เหลือต่อไปในภายหลัง

5.2.4 เนื่องจากเวลามีจำกัดอีกอันหนึ่งที่ยังไม่ได้ทำคือ ส่วนที่ใช้สำหรับจัดทำรายงานสรุป ของข้อมูลต่าง ๆ ที่เก็บเข้ามาในแฟ้มข้อมูล เช่น รายงานจำนวนครั้งที่ไฟฟ้าจากการไฟฟ้าส่วนภูมิภาคขัดข้องในแต่ละเดือน หรือ เวลารวมทั้งหมดที่ไฟฟ้าจากการไฟฟ้าส่วนภูมิภาคขัดข้องในแต่ละเดือนเป็นต้น ซึ่งส่วนที่เหลือที่ยังไม่ได้ทำ ผู้พัฒนาจะทำการพัฒนาส่วนที่เหลือทั้งหมดนี้ในโอกาสต่อไปในภายหลังเพื่อให้ระบบนี้สมบูรณ์มากยิ่งขึ้น

เอกสารอ้างอิง

เอกสารอ้างอิง

[1] กฤษดา ใจเย็น และ ชัยวัฒน์ ลิ้มพรจิตรวิไล. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ PIC16F84. กรุงเทพ ฯ : อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, ม.ป.ป.

[2] กิติ ภักดีวัฒนะกุล และ พนิดา พานิชกุล. คัมภีร์การวิเคราะห์และออกแบบระบบ. กรุงเทพ ฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2546.

[3] อภิชาติ ภู่พลับ. เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic. นนทบุรี : Infopress Developer Book, 2546.

[4] DCE Company. RS232 Data Interface. http://www.arcelect.com/index.htm, July, 2005.

[5] Hostito. ASCII table - Format of standdard characters. http://www.ascii.cl/, April, 2005.

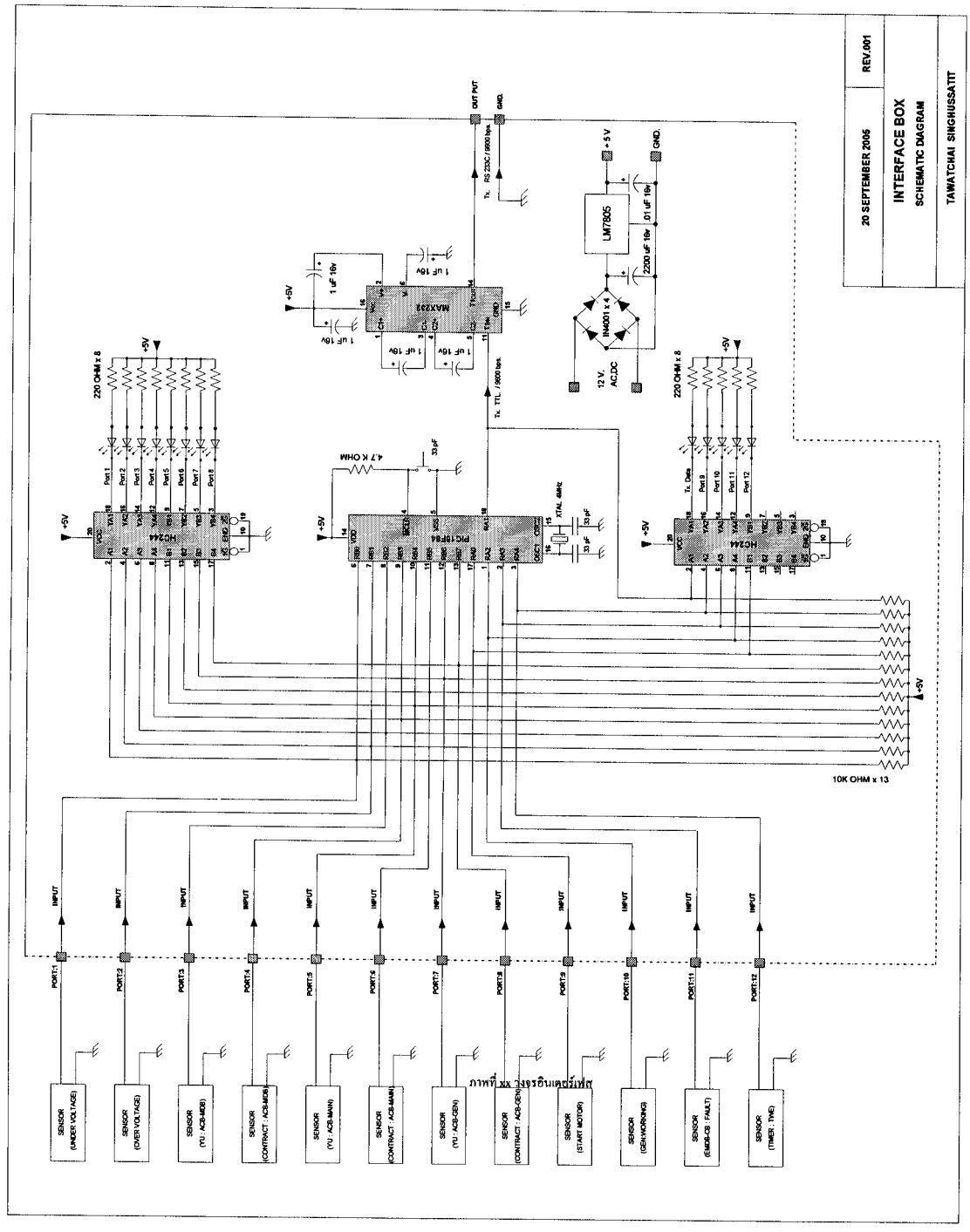
[6] Microship. 18-pin Flash/EEPROM 8-Bit Microcontrollers. http://ww1.microchip.com/downloads/en/DeviceDoc/30430c.pdf, January, 2005.

[7] Texas instruments, MAX232 data sheet. http://www.datasheetcatalog.com, March, 2005.

ภาคผนวก

ภาคผนวก ก
วงจรกล่องอินเตอร์เฟส

วงจรกล่องอินเตอร์เฟส



ภาพที่ 100 วงจรกล่องอินเตอร์เฟส

ภาคผนวก ข

โค้ดโปรแกรมไมโครคอนโทรลเลอร์ในกล่องอินเตอร์เฟส

โค้ดโปรแกรมไมโครคอนโทรลเลอร์ในกล่องอินเตอร์เฟส

```
                        list  P=16F84
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Interface Box  xxxxxxxxxxxxxxxxxxxxxxxx;
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Config  Define xxxxxxxxxxxxxxxxxxxxxxxx;
CP_ON       equ     0x000f      ; Code Protect ON
CP_OFF      equ     0x3fff      ; Code Protect OFF
PWT_ON      equ     0x3fff      ; Power Up Timer ON
PWT_OFF     equ     0x3ff7      ; Power Up Timer OFF
WDT_ON      equ     0x3fff      ; Watch Dog Timer ON
WDT_OFF     equ     0x3ffb      ; Watch Dog Timer OFF
LP_OSC      equ     0x3ffc      ; LP Oscilator
XT_OSC      equ     0x3ffd      ; XT Oscilator
HS_OSC      equ     0x3ffe      ; HS Oscilator
RC_OSC      equ     0x3fff      ; RC Oscilator
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
;                   Code Protect OFF
;                   Power Up Timer ON
;                   Watch Dog Timer OFF
;                   XT Oscilator
                    __config CP_OFF&PWT_ON&WDT_OFF&XT_OSC
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Byte Define xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
STATUS      equ     0x03
PORTA       equ     0x05
PORTB       equ     0x06
SEND        equ     0x0c
COUNT       equ     0x0d
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Bit Define xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
RP0             equ  5
F               equ  1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Test xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
                Org     0x000
                bsf     STATUS,RP0      ; Bank 1
                movlw   b'11111101'     ; RA1, =Output (TxData)
                movwf   PORTA
                movlw   b'11111111'     ; PortB = Input
                movwf   PORTB
                bcf     STATUS,RP0      ; Bank 0
start           bsf     PORTA,1         ; Mark bit  (ป1 Tx.Data)
                call    Delay96
                btfss   PORTB,0
                call    L1
                goto    start
L1              call    Delay96
                call    Delay96
                bcf     PORTA,1         ; Start bit (1=31h)
                call    Delay96
                bsf     PORTA,1         ;Digit 1
                call    Delay96
                bcf     PORTA,1         ;Digit 2
                call    Delay96
                bcf     PORTA,1         ;Digit 3
                call    Delay96
                bcf     PORTA,1         ;Digit 4
                call    Delay96
                bsf     PORTA,1         ;Digit 5
                call    Delay96
                bsf     PORTA,1         ;Digit 6
```

```
                call    Delay96
                bcf     PORTA,1         ;Digit 7
                call    Delay96
                bcf     PORTA,1         ;Digit 8
                call    Delay96
                bsf     PORTA,1         ; Stop bit
                call    Delay96
                goto    L1
;xxxxxxxxxxxxxxxxxxxxxxxx Delay Buadrate  104 Microsecond xxxxxxxxxxxxxxxx;
Delay96         movlw   0x21
                movwf   COUNT
Delay1          decfsz  COUNT,F
                goto    Delay1
                return
                end
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
```

ภาคผนวก ค

โค้ดโปรแกรมติดต่อกับผู้ใช้ (Graphic User Interface)

โค้ดโปรแกรมติดต่อกับผู้ใช้ (Graphic User Interface)

```
Option Explicit
Dim cn As ADODB.Connection
Dim cnnString As String
Dim Conn As New ADODB.Connection
Dim rs As New ADODB.Recordset
Dim rs1 As New ADODB.Recordset
Dim SQL As String
Dim Test As String
Dim N As Integer
Dim Date_F, Time_F, DateTime_F As Date
Dim Status As String
Dim SFlag, Tck, SxTO, SxTR As Boolean
Dim S01O, S01R, S02O, S02R, S03O, S03R, S04O, S04R, S05O, S05R  As Boolean
Dim S06O, S06R, S07O, S07R, S08O, S08R, S9O, S9R, S10O, S10R  As Boolean
Dim Aon, Bon, Con, Non, Aoff, Boff, Coff, Noff As Integer
Private Sub Change_Password_Click()
    frmChangeword.Show
End Sub
Private Sub cmdACK_Click()
    cmdACK.Visible = False
    ADO1.Recordset.AddNew
    ADO1.Recordset.Fields("DATE").Value = Date
    ADO1.Recordset.Fields("TIME").Value = Time
    ADO1.Recordset.Fields("SUB").Value = "EPML"
    ADO1.Recordset.Fields("EQUIP").Value = "VDU"
    ADO1.Recordset.Fields("ITEM").Value = "ACK. BUTTON"
    ADO1.Recordset.Fields("STATUS").Value = "ACKNOWLAGE"
    ADO1.Recordset.Update
```

```
        ADO1.Refresh
        ADO1.Recordset.MoveLast
    End Sub
    Private Sub Form_Activate()
        Screen.MousePointer = 0
    End Sub
    Private Sub Form_Load()
    On Error GoTo Err
    Dim R, s, T, N As Integer
    Dim ComStr As String
    Dim ComNum As Integer
    Dim SQLStr As String
    Dim ErrStr As String
        Screen.MousePointer = 11
        Me.Left = (Screen.Width - Me.Width) / 2
        Me.Top = (Screen.Height - Me.Height) / 2
        Call FlagON
        Tck = True
        N = 0
    ' xxxxxxxxxxxx Position Contractor ACB xxxxxxxxxxxxxxxxxx
        Aon = LAG.Y1 - 120
        Bon = LBG.Y1 - 120
        Con = LCG.Y1 - 120
        Non = LNG.Y1 - 120
        Aoff = LAG.Y1 - 200
        Boff = LBG.Y1 - 200
        Coff = LCG.Y1 - 200
        Noff = LNG.Y1 - 200
```

```
'xxxxxxxxxxxxxxxxxxx Open Config.mdb xxxxxxxxxxxxxxxxxx
    With Conn
        If .State = adStateOpen Then .Close
        .ConnectionString = strConn & ";Data Source=c:\EPML\CONFIG.MDB"
        .ConnectionTimeout = 90
        .Open
    End With
    SQL = "SELECT * FROM SOUND"
    With rs
        If .State = adStateOpen Then .Close
        .ActiveConnection = Conn
        .CursorType = adOpenForwardOnly
        .CursorLocation = adUseClient
        .Open SQL
    End With
    SQL = rs.Fields("soundfilename").Value
    MMControl1.Command = "close"
    MMControl1.DeviceType = "WaveAudio"
    MMControl1.FileName = rs.Fields("soundfilename").Value
    MMControl1.Command = "open"
' xxxxxxxxxxxxxxxxxxx Select Table CommPort xxxxxxxxxxxxxx
    SQL = "SELECT * FROM CommPort"
    With rs1
    If .State = adStateOpen Then .Close
        .ActiveConnection = Conn
        .CursorType = adOpenForwardOnly
        .CursorLocation = adUseClient
        .Open SQL
    End With
        ADO1.Recordset.MoveLast
        ComStr = rs1.Fields("ComName").Value
```

```
            If ComStr = "COM1" Then ComNum = 1
            If ComStr = "COM2" Then ComNum = 2
            If ComStr = "COM3" Then ComNum = 3
            If ComStr = "COM4" Then ComNum = 4
            If ComStr = "COM5" Then ComNum = 5
' xxxxxxxxxxxxxxx  Port Setting xxxxxxxxxxxxxxxxxxxxxxxxx
            mscReceiver.CommPort = ComNum
            mscReceiver.Settings = "9600,n,8,1"
            mscReceiver.PortOpen = True
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            Timer1.Interval = 800
            Timer1.Enabled = True
            lblStatus1.Caption = ""
            Exit Sub
Err:
            ErrStr = Error & " Please change new port "
            MsgBox ErrStr, 48, "EPML"
            Resume Next
End Sub
Private Sub Form_Unload(Cancel As Integer)
Dim Reply As Integer
    Beep
    Reply = MsgBox("Do you want to terminate EPML ?", 36, "EPML")
    If Reply = 7 Then
        Cancel = 1
        Exit Sub
    End If
        ADO1.Recordset.AddNew
        ADO1.Recordset.Fields("DATE").Value = Date
        ADO1.Recordset.Fields("TIME").Value = Time
        ADO1.Recordset.Fields("SUB").Value = "EPML"
```

```
            ADO1.Recordset.Fields("EQUIP").Value = "VDU"
            ADO1.Recordset.Fields("ITEM").Value = "WORK"
            ADO1.Recordset.Fields("STATUS").Value = "STOP"
            ADO1.Recordset.Update
            ADO1.Refresh
            ADO1.Recordset.MoveLast
End Sub
Private Sub Help_Click(Index As Integer)
frmAbout.Show
End Sub
Private Sub Logging_Utility_Click()
    frmLoggingUtility.Show
End Sub
Private Sub Login_Click()
    frmLogin.Show
End Sub
Private Sub Logout_Click()
Dim Reply As Integer
Beep
Reply = MsgBox("Do you want to logout from EPML ?", 36, "EPML")
If Reply = 6 Then
    frmMain.Logging_Utility.Enabled = False
    frmMain.System_Configuration.Enabled = False
    frmMain.Logout.Enabled = False
    frmMain.Change_Password = False
    frmMain.Login.Enabled = True
    frmMain.Toolbar1.Buttons(2).Enabled = False
    frmMain.Toolbar1.Buttons(3).Enabled = False          ' Logout
    frmMain.Toolbar1.Buttons(5).Enabled = False          ' Logging Utility
    frmMain.Toolbar1.Buttons(7).Enabled = False          ' Configuration
    frmMain.Toolbar1.Buttons(9).Enabled = False          ' Exit
```

```
        frmMain.Toolbar1.Buttons(1).Enabled = True          ' Login
        frmMain.Exit.Enabled = False
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Logout xxxxxxxxxxxxxxxxxxxxxxxxxxxx
        frmMain.ADO1.Recordset.AddNew
        frmMain.ADO1.Recordset.Fields("DATE").Value = Date
        frmMain.ADO1.Recordset.Fields("TIME").Value = Time
        frmMain.ADO1.Recordset.Fields("SUB").Value = "EPML"
        frmMain.ADO1.Recordset.Fields("EQUIP").Value = "VDU"
        frmMain.ADO1.Recordset.Fields("ITEM").Value = UserN
        frmMain.ADO1.Recordset.Fields("STATUS").Value = "LOGOUT"
        frmMain.ADO1.Recordset.Update
        frmMain.ADO1.Refresh
       frmMain.ADO1.Recordset.MoveLast
End If
End Sub
Private Sub System_Configuration_Click()
     frmConfiguration.Show
End Sub
Private Sub Timer1_Timer()
Dim recData As Integer
Dim Rest As String
Dim Data As String
On Error Resume Next
     N = N + 1
     mscReceiver.InputLen = 0
     Data = mscReceiver.Input
     Test = Trim(Data)
     If InStr(Test, "T") Then                              ' Check Link
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 01 Under Voltage xxxxxxxxxxxxxxxxxxxxxxxxx
          If InStr(Test, "01") Then
               S01.BackColor = &HFF00&
```

```
            S01PEA.BackColor = &HFF00&
            lbl01.BackColor = &HFF00&
            lbl01PEA.BackColor = &HFF00&
            lbl011PEA.BackColor = &HFF00&
            lbl011PEA.BackColor = &HFF00&
            Call S01Good
        Else
            S01.BackColor = &HFF&
            S01PEA.BackColor = &HFF&
            lbl01.BackColor = &HFF&
            lbl01PEA.BackColor = &HFF&
            lbl011PEA.BackColor = &HFF&
            Call S01Fault
        End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 02 Over Voltage xxxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "02") Then
            S02.BackColor = &HFF00&
            lbl02.BackColor = &HFF00&
            Call S02Good
        Else
            S02.BackColor = &HFF&
            lbl02.BackColor = &HFF&
            Call S02Fault
        End If
```

```
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 03 YU MDB xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "03") Then
            S03.BackColor = &HFF00&
            lbl03.BackColor = &HFF00&
            Call S03Good
        Else
            S03.BackColor = &HE0E0E0    '&HFF&
            lbl03.BackColor = &HE0E0E0    '&HFF&
            Call S03FAULT
        End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 04 MDB xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "04") Then
            S04.BackColor = &HFF00&
            S041.BackColor = &HFF00&
            lbl04.BackColor = &HFF00&
            lbl041.BackColor = &HFF00&
            lblMDB.BackColor = &HFF00&
            lMDBA.Y1 = Aon
            lMDBA.Y2 = Aon
            lMDBB.Y1 = Bon
            lMDBB.Y2 = Bon
            lMDBC.Y1 = Con
            lMDBC.Y2 = Con
            lMDBN.Y1 = Non
            lMDBN.Y2 = Non
            BlockMDB.BackColor = &HFF00&
            Call S04Good
        Else
            S04.BackColor = &HFF&
            S041.BackColor = &HFF&
            lbl04.BackColor = &HFF&
```

```
            lbl041.BackColor = &HFF&
            lblMDB.BackColor = &HFF&
            Call S04Fult
            lMDBA.Y1 = Aoff
            lMDBA.Y2 = Aoff
            lMDBB.Y1 = Boff
            lMDBB.Y2 = Boff
            lMDBC.Y1 = Coff
            lMDBC.Y2 = Coff
            lMDBN.Y1 = Noff
            lMDBN.Y2 = Noff
            BlockMDB.BackColor = &HFF&
        End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxx 05 YU ACB:MAIN xxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "05") Then
            S05.BackColor = &HFF00&
            lbl05.BackColor = &HFF00&
            Call S05Good
        Else
            S05.BackColor = &HE0E0E0    '&HFF&
            lbl05.BackColor = &HE0E0E0    '&HFF&
            BlockBkM.BackColor = &HFF&
            lblBkM.BackColor = &HFF&
            Call S05Fult
        End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 06  ACB:MAIN xxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "06") Then
            LMA.Y1 = Aon
            LMA.Y2 = Aon
            LMB.Y1 = Bon
            LMB.Y2 = Bon
```

```
                LMC.Y1 = Con
                LMC.Y2 = Con
                LMN.Y1 = Non
                LMN.Y2 = Non
                BlockBkM.BackColor = &HFF00&
                lblBkM.BackColor = &HFF00&
                Call S06Good
            Else
                LMA.Y1 = Aoff
                LMA.Y2 = Aoff
                LMB.Y1 = Boff
                LMB.Y2 = Boff
                LMC.Y1 = Coff
                LMC.Y2 = Coff
                LMN.Y1 = Noff
                LMN.Y2 = Noff
                BlockBkM.BackColor = &HFF&
                lblBkM.BackColor = &HFF&
                Call S06Fault
            End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxx 07 YU ACB-GEN xxxxxxxxxxxxxxxxxxxxx
            If InStr(Test, "07") Then
                S07.BackColor = &HFF00&
                lbl07.BackColor = &HFF00&
                Call S07Good
            Else
                S07.BackColor = &HE0E0E0
                lbl07.BackColor = &HE0E0E0
                Call S07Fault
            End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxx 08 ACB-GEN xxxxxxxxxxxxxxxxxxxxxxx
```

```
If InStr(Test, "08") Then
    LGA.Y1 = Aon
    LGA.Y2 = Aon
    LGB.Y1 = Bon
    LGB.Y2 = Bon
    LGC.Y1 = Con
    LGC.Y2 = Con
    LGN.Y1 = Non
    LGN.Y2 = Non
    BlockBkG.BackColor = &HFF00&
    lblBkG.BackColor = &HFF00&
    Call S08Good
Else
    LGA.Y1 = Aoff
    LGA.Y2 = Aoff
    LGB.Y1 = Boff
    LGB.Y2 = Boff
    LGC.Y1 = Coff
    LGC.Y2 = Coff
    LGN.Y1 = Noff
    LGN.Y2 = Noff
    BlockBkG.BackColor = &HE0E0E0
    lblBkG.BackColor = &HE0E0E0
    Call S08Fault
End If
```

```
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 09 Sing Start xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "09") Then
            Call S09Good
            lblStatus.ForeColor = &HFF&
            lblStatus.Caption = "Starting !"
        Else
            lblStatus.Caption = ""
            Call S09Fault
        End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxx 10 GENERATOR xxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "T") Then
            If InStr(Test, "10") Then
                S10.BackColor = 65280 ' &HE0E0E0
                lbl10.BackColor = 65280 ' &HE0E0E0
                lblKVA.BackColor = 65280
                If InStr(Test, "09") Then
                lblStatus.Caption = "Starting !"
                lblStatus.BackColor = &HFF00&
            Else
                lblStatus.Caption = "Working !"
                lblStatus.BackColor = &HFF00&
            End If
            Call S10Good
        Else
            S10.BackColor = &HE0E0E0 '65280
            lbl10.BackColor = &HE0E0E0              '65280
            lblKVA.BackColor = &HE0E0E0
            If InStr(Test, "09") Then
                lblStatus.Caption = "Starting !"
                lblStatus.BackColor = &HE0E0E0
            Else
```

```
                lblStatus.Caption = "Stand-By"
                lblStatus.ForeColor = &HFF&
                lblStatus.BackColor = &HE0E0E0
            End If
            Call S10Fault
        End If
    End If
' xxxxxxxxxxxxxxxxxxxxxxxxxxxx 11 EMDB xxxxxxxxxxxxxxxxxxxxxxxxxx
    If InStr(Test, "T") Then
        If InStr(Test, "11") Then
            S11.BackColor = &HFF00&
            S111.BackColor = &HFF00&
            S112.BackColor = &HFF00&
            S113.BackColor = &HFF00&
            S114.BackColor = &HFF00&
            lbl11.BackColor = &HFF00&
            lbl111.BackColor = &HFF00&
            lbl112.BackColor = &HFF00&
            lbl113.BackColor = &HFF00&
            lbl114.BackColor = &HFF00&
            Call S11Good
        Else
            S11.BackColor = &HFF&
            S111.BackColor = &HFF&
            S112.BackColor = &HFF&
            S113.BackColor = &HFF&
            S114.BackColor = &HFF&
            lbl11.BackColor = &HFF&
            lbl111.BackColor = &HFF&
            lbl112.BackColor = &HFF&
            lbl113.BackColor = &HFF&
```

```
                lbl114.BackColor = &HFF&
                Call S11Fault
            End If
        End If
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 12 TWE  xxxxxxxxxxxxxxxxxxxxxxxxxxxx
        If InStr(Test, "T") Then
            If InStr(Test, "12") Then
                lblTWE.Visible = False
                Call S12Good
            Else
                lblTWE.Visible = True
                Call S12Fault
            End If
        End If
        Call SxTGood
    Else
    ' xxxxxxxxxxxxxxxxxxxxxx Check link  (T) xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        Call SxTFault
        S11.BackColor = &HFF00&
        S11.BackColor = &HE0E0E0
        S111.BackColor = &HE0E0E0
        S112.BackColor = &HE0E0E0
        S113.BackColor = &HE0E0E0
        S114.BackColor = &HE0E0E0
        lbl11.BackColor = &HE0E0E0
        lbl111.BackColor = &HE0E0E0
        lbl112.BackColor = &HE0E0E0
        lbl113.BackColor = &HE0E0E0
        lbl114.BackColor = &HE0E0E0
        lblStatus.BackColor = &HE0E0E0
        lbl07.BackColor = &HE0E0E0
```

```
S07.BackColor = &HE0E0E0
lblMDB.BackColor = &HE0E0E0
S01.BackColor = &HE0E0E0
S01PEA.BackColor = &HE0E0E0
S02.BackColor = &HE0E0E0
S03.BackColor = &HE0E0E0
BlockMDB.BackColor = &HE0E0E0
S04.BackColor = &HE0E0E0
S041.BackColor = &HE0E0E0
S05.BackColor = &HE0E0E0
S091.BackColor = &HE0E0E0
S10.BackColor = &HE0E0E0
lblStatus.Caption = ""
lbl01.BackColor = &HE0E0E0
lbl011PEA.BackColor = &HE0E0E0
lbl011PEA.BackColor = &HFF00&
lbl02.BackColor = &HE0E0E0
lbl03.BackColor = &HE0E0E0
lbl04.BackColor = &HE0E0E0
lbl041.BackColor = &HE0E0E0
lbl05.BackColor = &HE0E0E0
lbl091.BackColor = &HE0E0E0
lbl10.BackColor = &HE0E0E0
lblKVA.BackColor = &HE0E0E0
BlockBkM.BackColor = &HE0E0E0
BlockBkG.BackColor = &HE0E0E0
lblBkG.BackColor = &HE0E0E0
lblBkM.BackColor = &HE0E0E0
lbl011PEA.BackColor = &HE0E0E0
lbl01PEA.BackColor = &HE0E0E0
End If
```

```
If Tck = True Then
    ADO1.Recordset.AddNew
    ADO1.Recordset.Fields("DATE").Value = Date
    ADO1.Recordset.Fields("TIME").Value = Time
    ADO1.Recordset.Fields("SUB").Value = "EPML"
    ADO1.Recordset.Fields("EQUIP").Value = "VDU"
    ADO1.Recordset.Fields("ITEM").Value = "WORK"
    ADO1.Recordset.Fields("STATUS").Value = "WORK"
    ADO1.Recordset.Update
    ADO1.Refresh
    ADO1.Recordset.MoveLast
    Tck = False
End If
    If Date <> lblDate.Caption Then lblDate.Caption = Format(Date, "mm-dd-yy  ")
    If Time <> lblDate.Caption Then lblTime.Caption = Format(Time, "hh:mm:ss")
    If mscReceiver.PortOpen Then
        lblStatus2.ForeColor = &H808080
        lblStatus2.Caption = "Connected"
    Else
        lblStatus2.ForeColor = &HFF&
        lblStatus2.Caption = "Disconnected"
    End If
    lblStatus2.Caption = lblStatus2.Caption & " : 9600, 8-N-1," & " COM" &
mscReceiver.CommPort
End Sub
```

```
' xxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 01 xxxxxxxxxxxxxxxxxxxxxxxxxxx
Public Sub S01Good()
Dim SQL, Test As String
Dim rs As Recordset
    If S01R = True Then              ' Sensor 01 Recovered
        S01O = True
        S01R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=1"
            AddLogging
        End If
    End If
End Sub
Public Sub S01Fault()
Dim a As String
    If (S01O = True) Then
        S01R = True
        S01O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=2"
            AddLogging
            cmdACK.Visible = True
        End If
    End If
End Sub
' xxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 02 xxxxxxxxxxxxxxxxxxxxxxxx
Public Sub S02Good()
    If S02R = True Then
        S02O = True
        S02R = False
        If N > 1 Then
```

```
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=3"
                AddLogging
            End If
        End If
    End Sub

    Public Sub S02Fault()
        If S02O = True Then
            S02R = True
            S02O = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=4"
                AddLogging
                cmdACK.Visible = True
            End If
        End If
    End Sub
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 03 xxxxxxxxxxxxxxxxxxxxxxx
    Public Sub S03Good()
        If S03R = True Then
            S03O = True
            S03R = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=5"
                AddLogging
            End If
        End If
        End Sub
    Public Sub S03FAULT()
        If S03O = True Then
            S03R = True
```

```
            S03O = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=6"
                Date_F = Date$
                Time_F = Time$
                DateTime_F = Now
                AddLogging
                cmdACK.Visible = True
            End If
        End If
    End Sub
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 04 xxxxxxxxxxxxxxxxxxxxxxxxxxxx
    Public Sub S04Good()
        If S04R = True Then
            S04O = True
            S04R = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=7"
                AddLogging
            End If
        End If
    End Sub
    Public Sub S04Fult()
        If S04O = True Then
            S04R = True
            S04O = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=8"
                Date_F = Date$
                Time_F = Time$
                DateTime_F = Now
```

```
                AddLogging
                cmdACK.Visible = True
            End If
        End If
    End Sub
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 05 xxxxxxxxxxxxxxxxxxxxxxxxxx
    Public Sub S05Good()
        If S05R = True Then
            S05O = True
            S05R = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=9"
                AddLogging
            End If
        End If
    End Sub
    Public Sub S05Fult()
        If S05O = True Then
            S05R = True
            S05O = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=10"
                AddLogging
                cmdACK.Visible = True
            End If
        End If
    End Sub
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 06 xxxxxxxxxxxxxxxxxxxxxxxxx
    Public Sub S06Good()
        If S06R = True Then
            S06O = True
```

```
            S06R = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=11"
                AddLogging
            End If
        End If
    End Sub

    Public Sub S06Fault()
        If S06O = True Then
            S06R = True
            S06O = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=12"
                AddLogging
                cmdACK.Visible = True
            End If
        End If
    End Sub
    ' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 07 xxxxxxxxxxxxxxxxxxxxxx
    Public Sub S07Good()
        If S07R = True Then
            S07O = True
            S07R = False
            If N > 1 Then
                ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=13"
                AddLogging
            End If
        End If
    End Sub
    Public Sub S07Fault()
```

```
    If S07O = True Then
        S07R = True
        S07O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=14"
            AddLogging
            cmdACK.Visible = True
        End If
    End If
        End Sub
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 08 xxxxxxxxxxxxxxxxxxxxxxxx
Public Sub S08Good()
    If S08R = True Then
        S08O = True
        S08R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=15"
            AddLogging
        End If
    End If
End Sub
Public Sub S08Fault()
    If S08O = True Then
        S08R = True
        S08O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=16"
            AddLogging
            cmdACK.Visible = True
        End If
    End If
```

```
End Sub
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 09 xxxxxxxxxxxxxxxxxxxxxxx
Public Sub S09Good()
    If S09R = True Then
        S09O = True
        S09R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=17"
            AddLogging
        End If
    End If
End Sub
Public Sub S09Fault()
    If S09O = True Then
        S09R = True
        S09O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=18"
            AddLogging
        End If
    End If
End Sub
```

```
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 10 xxxxxxxxxxxxxxxxxxxxxxx
Public Sub S10Good()
    If S10R = True Then
        S10O = True
        S10R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=19"
            AddLogging
        End If
    End If
End Sub
Public Sub S10Fault()
    If S10O = True Then
        S10R = True
        S10O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=20"
            AddLogging
        End If
    End If
End Sub
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 11 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Public Sub S11Good()
    If S11R = True Then
        S11O = True
        S11R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=21"
            AddLogging
        End If
    End If
```

```
End Sub
Public Sub S11Fault()
    If S11O = True Then
        S11R = True
        S11O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=22"
            AddLogging
            cmdACK.Visible = True
        End If
    End If
End Sub
' xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Logging Sensor 12 xxxxxxxxxxxxxxxxxxxxxxxx
Public Sub S12Good()
    If S12R = True Then
        S12O = True
        S12R = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=23"
            AddLogging
        End If
    End If
End Sub
Public Sub S12Fault()
    If S12O = True Then
        S12R = True
        S12O = False
        If N > 1 Then
            ADO2.RecordSource = "SELECT * FROM PortConfig WHERE PortID=24"
            AddLogging
            cmdACK.Visible = True
```

```
            End If
        End If
    End Sub
    Public Sub FlagON()
        SxTO = True
        SxTR = True
        S01O = True
        S01R = True
        S02O = True
        S02R = True
        S03O = True
        S03R = True
        S04O = True
        S04R = True
        S05O = True
        S05R = True
        S06O = True
        S06R = True
        S07O = True
        S07R = True
        S08O = True
        S08R = True
        S09O = True
        S09R = True
        S10O = True
        S10R = True
    End Sub
    Public Sub FlagOFF()
        SxTO = False
        SxTR = False
        S01O = False
```

```
        S01R = False
        S02O = False
        S02R = False
        S03O = False
        S03R = False
        S04O = False
        S04R = False
        S05O = False
        S05R = False
        S06O = False
        S06R = False
        S07O = False
        S07R = False
        S08O = False
        S08R = False
        S09O = False
        S09R = False
        S10O = False
        S10R = False
    End Sub
Public Sub SxTGood()
    If SxTR = True Then
        SxTO = True
        SxTR = False
        If N > 1 Then
            lblStatus1.Caption = ""
            ADO1.Recordset.AddNew
            ADO1.Recordset.Fields("DATE").Value = Date
            ADO1.Recordset.Fields("TIME").Value = Time
            ADO1.Recordset.Fields("SUB").Value = "EPML"
            ADO1.Recordset.Fields("EQUIP").Value = "LINK"
```

```
                ADO1.Recordset.Fields("ITEM").Value = "FAULT"
                ADO1.Recordset.Fields("STATUS").Value = "RECOVERED"
                ADO1.Recordset.Update
                ADO1.Refresh
                ADO1.Recordset.MoveLast
            End If
        End If
    End Sub
    Public Sub SxTFault()
        If SxTO = True Then
            SxTR = True
            SxTO = False
            If N > 1 Then
                lblStatus1.Caption = ": Link Fault"
                ADO1.Recordset.AddNew
                ADO1.Recordset.Fields("DATE").Value = Date
                ADO1.Recordset.Fields("TIME").Value = Time
                ADO1.Recordset.Fields("SUB").Value = "EPML"
                ADO1.Recordset.Fields("EQUIP").Value = "LINK"
                ADO1.Recordset.Fields("ITEM").Value = "FAULT"
                ADO1.Recordset.Fields("STATUS").Value = "OCCURRED"
                ADO1.Recordset.Update
                ADO1.Refresh
                ADO1.Recordset.MoveLast
                cmdACK.Visible = True
            End If
        End If
    End Sub
    Private Sub Timer2_Timer()
        If cmdACK.Visible = True Then
            MMControl1.Command = "close"
```

```
            MMControl1.Command = "open"
            MMControl1.Command = "play"
        End If
    End Sub
    Public Sub AddLogging()
                ADO2.Refresh
                ADO1.Recordset.AddNew
                ADO1.Recordset.Fields("DATE").Value = Date
                ADO1.Recordset.Fields("TIME").Value = Time
                ADO1.Recordset.Fields("SUB").Value =
                _ADO2.Recordset.Fields("Subsystem").Value
                ADO1.Recordset.Fields("EQUIP").Value =
                _ADO2.Recordset.Fields("Equipment").Value
                ADO1.Recordset.Fields("ITEM").Value =
ADO2.Recordset.Fields("Item").Value
                ADO1.Recordset.Fields("STATUS").Value =
ADO2.Recordset.Fields("Status").Value
                ADO1.Recordset.Fields("REMARK").Value =
ADO2.Recordset.Fields("Remark").Value
                ADO1.Recordset.Update
                ADO1.Refresh
                ADO1.Recordset.MoveLast
    End Sub
    Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
        Select Case Button.Key
            Case "LoggingUtility"
                Logging_Utility_Click
            Case "Exit"
                Unload Me
            Case "Configuration"
                System_Configuration_Click
```

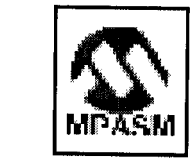
```
            Case "Login"
                Login_Click
                frmMain.Login.Enabled = False
            Case "Logout"
                Logout_Click
            Case "ChangePassword"
                Change_Password_Click
            Case Else
        End Select
    End Sub
```

ภาคผนวก ง
การใช้งานโปรแกรม MPASM

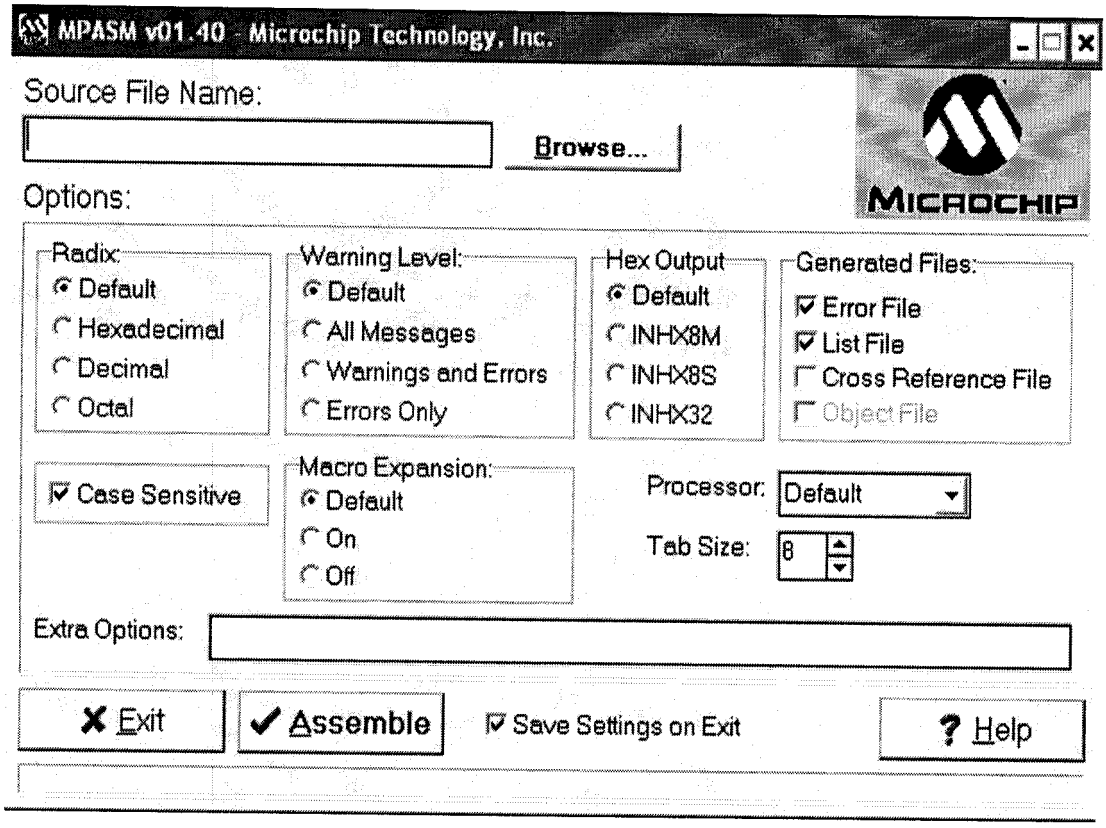
การใช้งานโปรแกรม MPASM

1. Run โปรแกรม MPASM

ให้ดับเบิลคลิกที่ไอคอน MPASM จากนั้นหน้าแรกก็จะถูกเปิดขึ้น ดังแสดงในภาพที่ 101 (ซึ่งต้องทำการลงโปรแกรมให้เรียบร้อยก่อน)



MPASM.EXE



ภาพที่ 101 การเลือกเงื่อนไขในการแอสเซมเบลอร์

2. การกำหนดค่า Options

ในการแอสเซมเบลอร์ เป็นการตั้งค่าเริ่มต้นในการทำงานของโปรแกรม MPASM เพื่อเลือกเงื่อนไขในการแอสเซมเบลอร์ซอร์สโค้ดภาษาแอสเซมบลี

ตั้งค่าต่าง ๆ ดังนี้

(1) Radix = Default

(2) Warnning Level = Default

(3) Hex Output = Default

(4) Generated File = Error File และ List File

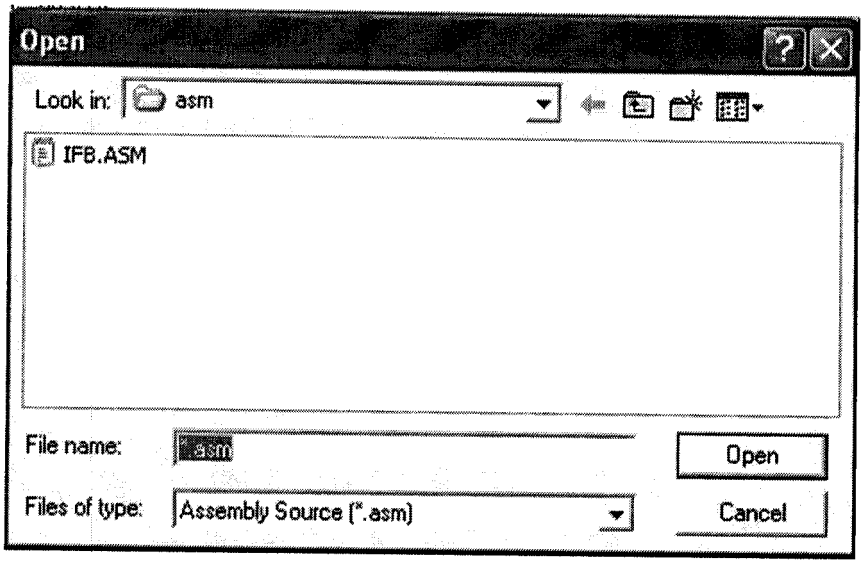
(5) เลือก Case Sensitive

(6) Macro Expansion = Default

(7) Processor = PIC16F84

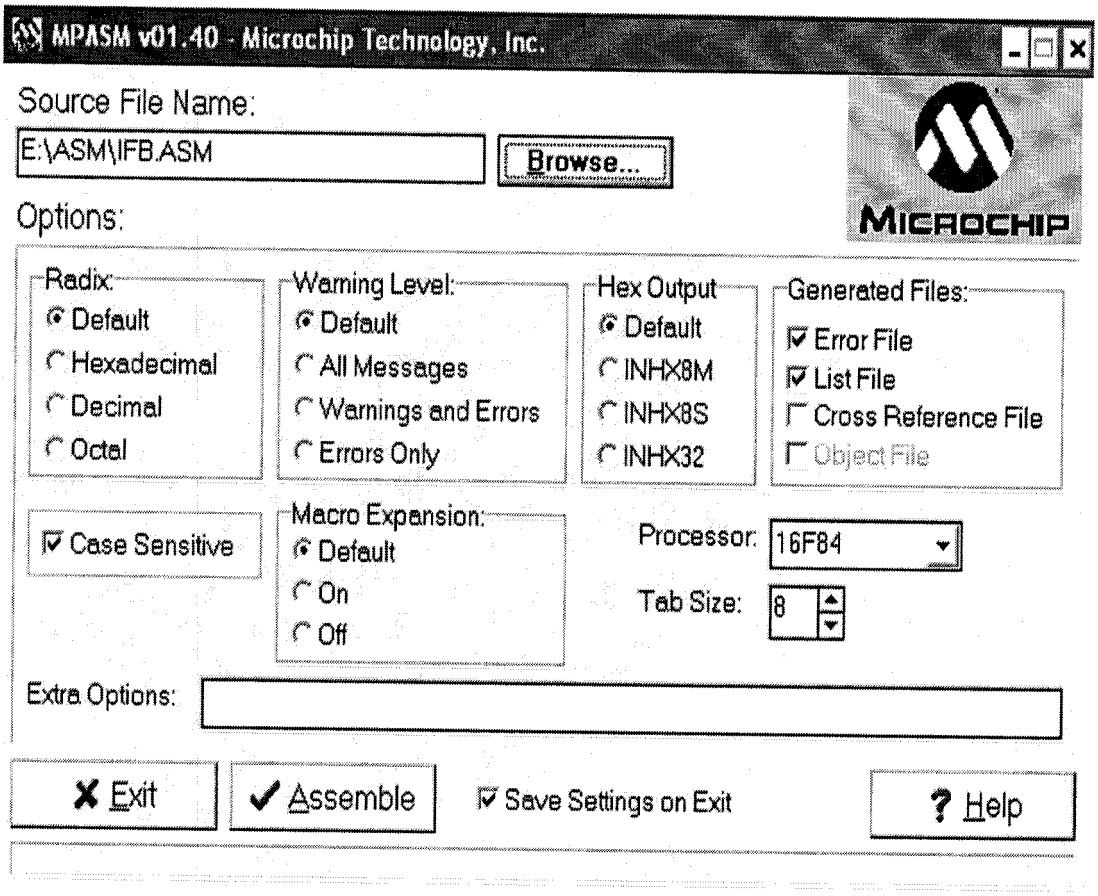
(8) Tab Size = 8

เมื่อกำหนด Options เสร็จเรียบร้อยแล้ว จากนั้นให้กดปุ่ม Browse เพื่อไฟล์ซอร์สโค้ดที่เขียนโดยภาษาแอสเซมบลีที่มีนามสกุลเป็น .ASM ในตัวอย่างด้านล่าง ดังแสดงในภาพที่ 102 ซึ่งมีชื่อไฟล์ว่า IFB.ASM



ภาพที่ 102 การค้นหาไฟล์ซอร์สโค้ด

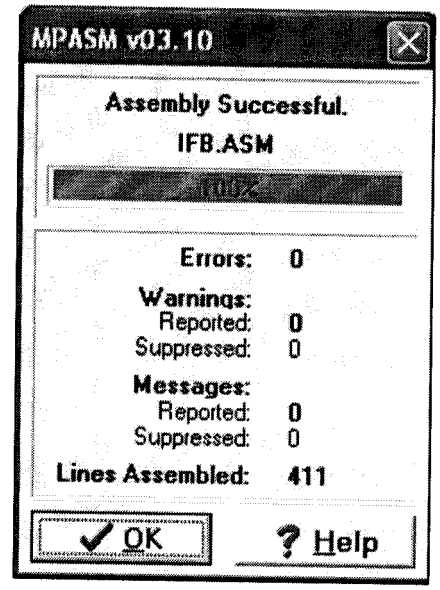
ต่อจากนั้นให้คลิกที่ไฟล์ IFB.ASM จากนั้นให้คลิกที่ปุ่ม Open ชื่อไฟล์ซอร์สโค้ดที่ต้องการจะแอสเซมเบลอร์ ก็จะไปปรากฏในช่อง Source File Name ดังแสดงในภาพที่ 103



ภาพที่ 103 ไฟล์ซอร์สโค้ดที่จะแอสเซมเบลอร์

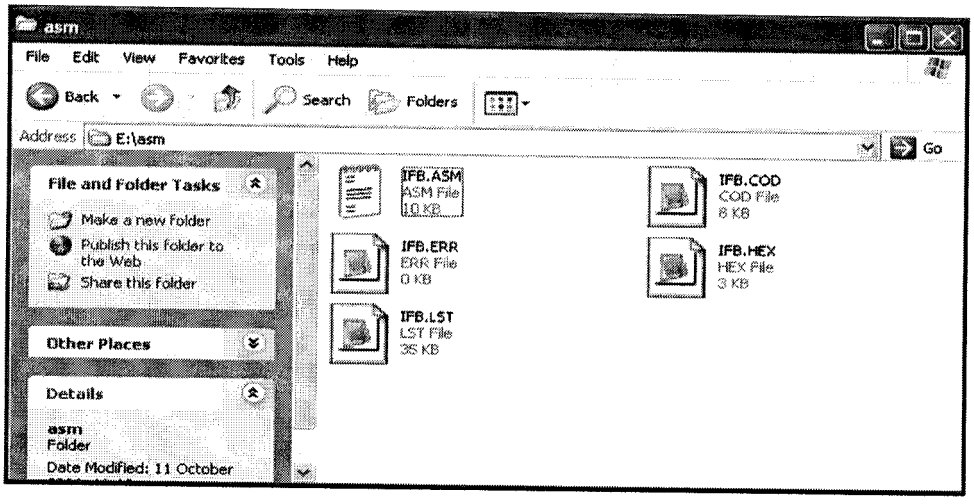
3. ทำการแอสเซมเบลอร์

เมื่อกำหนด Option ต่าง ๆ เรียบร้อยหมดแล้ว จากนั้นให้คลิกที่ปุ่ม Assemble เพื่อทำการแปลซอร์สโค้ดจากภาษาแอสเซมบลี (ไฟล์นามสกุล .ASM) ให้เป็นภาษาเครื่อง (ไฟล์นามสกุลเป็น .HEX) ซึ่งเรียกว่า "การแอสเซมเบลอร์" ถ้าหากซอร์สโค้ดเขียนถูกต้องตามหลักไวยากรณ์ของภาษาแอสเซมบลี ก็จะสามารถทำการแอสเซมเบลอร์ได้สำเร็จ และโปรแกรม MPASM ก็จะแสดงผลการแอสเซมเบลอร์ออกมา ดังแสดงในภาพที่ 104



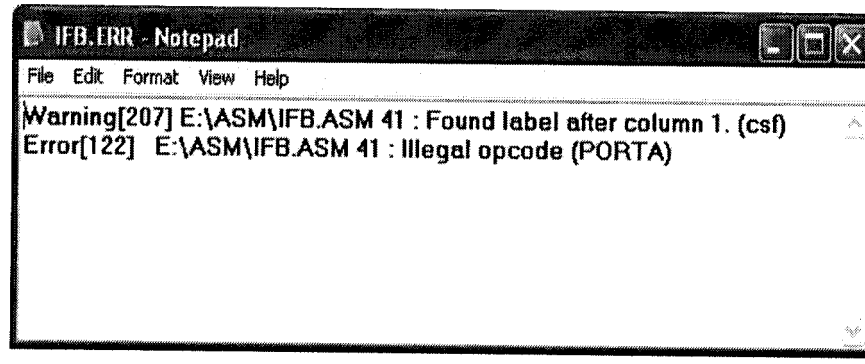
ภาพที่ 104 ตัวอย่างผลการแอสเซมเบลอร์สำเร็จ

หลังจากทำการแอสเซมเบลอร์แล้วจะมีไฟล์เพิ่มขึ้นมาอีก 4 ไฟล์คือ .COD, .EER, .LST, .HEX ดังแสดงในภาพที่ 105 ส่วนไฟล์ที่มีนามสกุล .ERR ดังแสดงในภาพที่ 106 นั้น จะบอกรายละเอียดเกี่ยวกับข้อผิดพลาดต่าง ๆ ในการแอสเซมเบลอร์ส่วนไฟล์ที่มีนามสกุล .LST ดังแสดงในภาพที่ 107 นั้นจะบอกรายละเอียดของโปรแกรม ทั้งนีโมนิกและออปโค้ด ซึ่งเป็นเลขฐานสิบหก ตลอดจนข้อความที่ใช้อธิบายการทำงานของโปรแกรม ส่วนไฟล์นามสกุล .HEX ดังแสดงในภาพที่ 108 จะเป็นข้อมูลเลขฐานสิบหกที่ใช้สำหรับเขียนลงในหน่วยความจำโปรแกรมบนไมโครคอนโทรลเลอร์

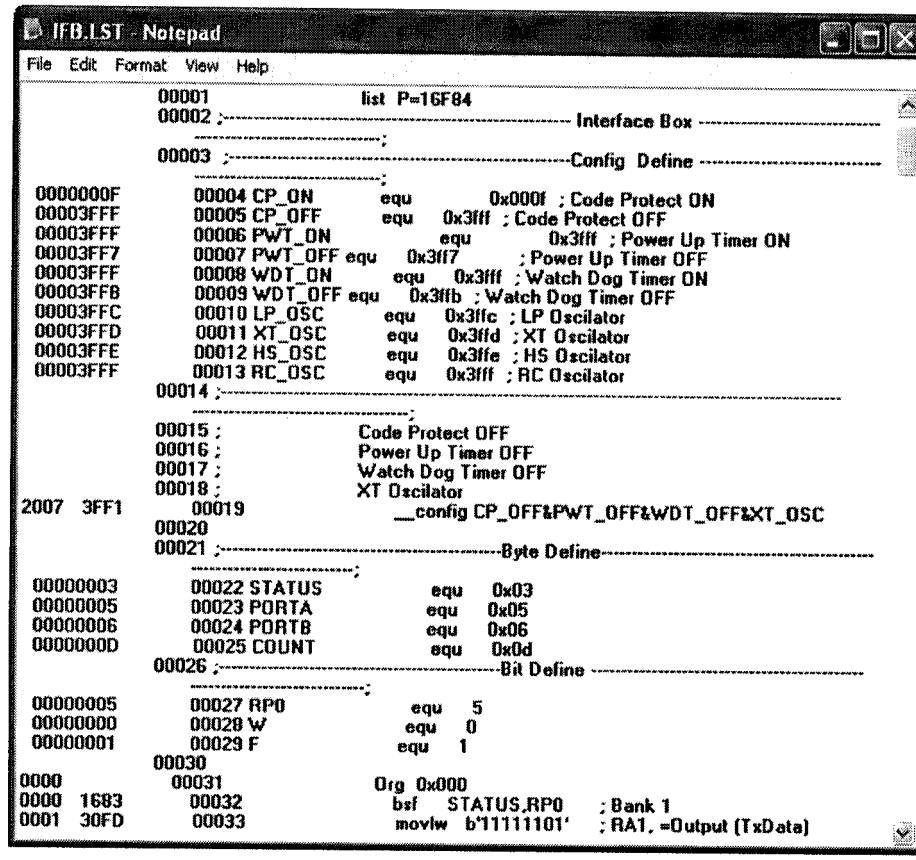


ภาพที่ 105 ให้เห็นไฟล์ที่เพิ่มขึ้นมาอีก 4 ไฟล์

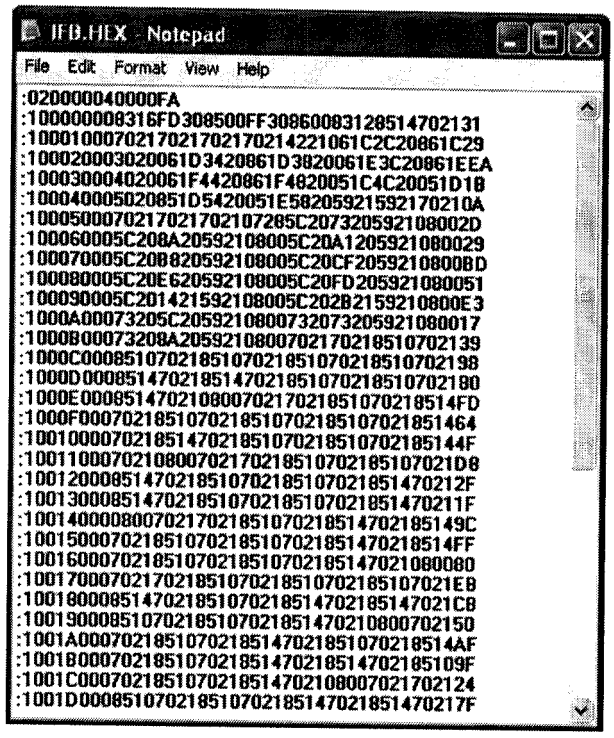
และถ้าหากว่าซอร์สโค้ดที่เขียนด้วยภาษาแอสเซมบลี เขียนไม่ถูกต้องตามหลังไวยากรณ์ เมื่อทำการแอสเซมเบลอร์จะมีปัญหาคือ มีข้อผิดพลาด (Error) เกิดขึ้น โปรแกรม MPASM จะสรุปรายละเอียดในข้อผิดพลาดต่าง ๆ ที่เกิดขึ้นไว้ในไฟล์นามสกุล .ERR ดังแสดงในภาพที่ 106



ภาพที่ 106 รายละเอียดในไฟล์นามสกุล .ERR

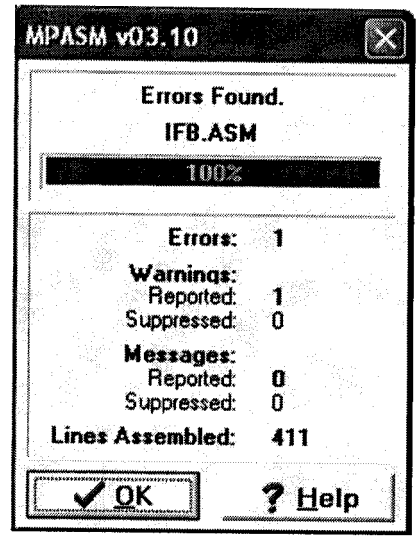


ภาพที่ 107 รายละเอียดในไฟล์นามสกุล .LST



ภาพที่ 108 รายละเอียดในไฟล์นามสกุล .HEX

จากภาพที่ 109 เป็นตัวอย่างของการแอสเซมเบลอร์ ที่มีข้อผิดพลาดเนื่องมาจากเขียนโค้ดโปรแกรมภาษาแอสเซมบลี ผิดหลักไวยากรณ์ ซึ่งผลจากการรายงานดังแสดงในภาพที่ 109 ซึ่งจะบอกความหมายได้ดังนี้ ไฟล์ภาษาแอสเซมบลี ชื่อ IFB.ASM มี Error 1 ครั้ง เพราะเขียนโค้ดผิดหลักไวยากรณ์ โดยโปรแกรมจะมีรายงานการเตือน 1 ครั้ง และบอกว่า มีการแอสเซมเบลอร์ทั้งหมด 411 บรรทัด



ภาพที่ 109 ตัวอย่างผลการแอสเซมเบลอร์ที่มีข้อผิดพลาด

หากต้องการตรวจสอบรายละเอียดของข้อผิดพลาดต่าง ๆ ว่าเขียนโค้ดภาษาแอสเซมบลีผิดพลาดจุดใดก็สามารถตรวจสอบได้โดยการเปิดดูที่ แฟ้มที่มีนามสกุล .ERR ใน Folder เดียวกันกับ ไฟล์ซอร์สโค้ด ดังแสดงในภาพที่ 106 ซึ่งเป็นตัวอย่างรายละเอียดของข้อผิดพลาดพอสรุปได้ว่า มีการเขียนภาษาแอสเซมบลีผิดหลักไวยากรณ์ในที่บรรทัดที่ 41 ในไฟล์ซอร์สโค้ด (คำสั่งไมโครคอนโทรลเลอร์ในตระกูล PIC ไม่มีคำสั่ง csf จะมีแต่คำสั่ง bsf)

ภาคผนวก จ

การใช้งานโปรแกรม ICPROG

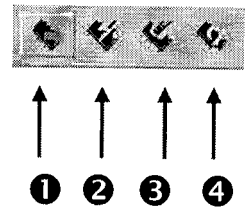
การใช้งานโปรแกรม ICPROG

1. ต่อ PIC programmer เข้ากับ serial port

2. Run โปรแกรม icprog (ต้องทำการลงโปรแกรมให้เรียบร้อยก่อน)

ภาพที่ 110 การ Run โปรแกรม icprog โดยการดับเบิลคลิกไอคอน icprog

3. **กำหนดค่าต่างๆ ในโปรแกรม icprog**



❶ Read All สำหรับอ่านข้อมูลจาก chip เข้ามาสู่โปรแกรม

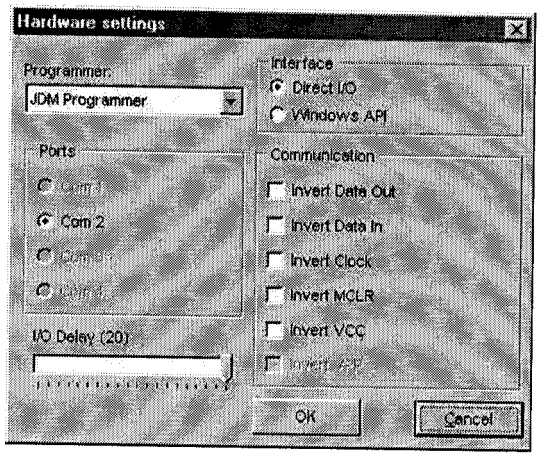
❷ Program All สำหรับเขียนข้อมูลจากโปรแกรมลงไปใน chip

❸ Erase All สำหรับทำการลบข้อมูลภายใน chip

❹ Verify สำหรับทำการเปรียบเทียบข้อมูลภายใน chip กับ ข้อมูลที่อยู่บนโปรแกรม

ภาพที่ 111 Toolbar บน โปรแกรม icprog เรียงจากซ้ายไปขวา

3.1 เลือกไปที่ Setting > Hardware ดังแสดงในภาพที่ 112



ภาพที่ 112 การตั้งค่าอุปกรณ์ (Hardware settings)

ตั้งค่าต่าง ๆ ดังนี้

(1) programmer = JDM Programmer

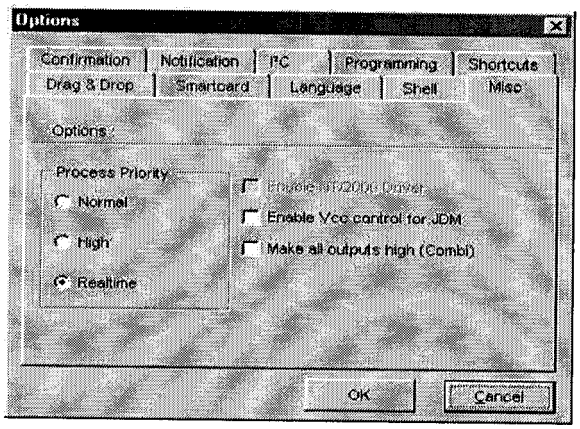
(2) Ports = Com2 (ในที่นี้ใช้การต่อ JDM Programmer ไว้ที่ Com2 ของ computer)

(3) I/O Delay = 20

(4) Interface = Direct I/O

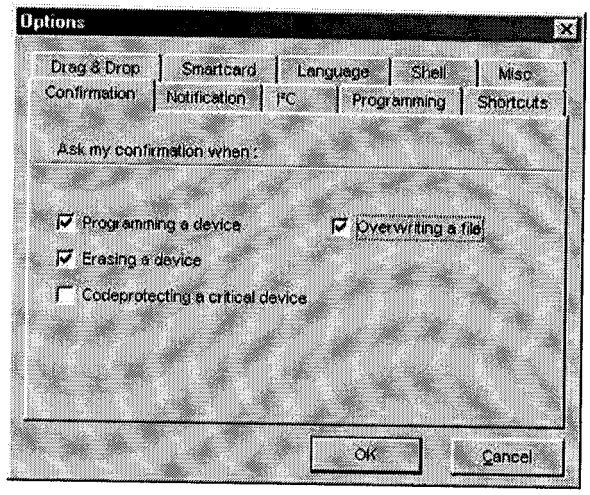
(5) Communication = (ไม่ต้องกำหนด) เมื่อตั้งค่าอุปกรณ์ต่าง ๆ เสร็จแล้ว ให้เลือกปุ่ม OK

(6) เลือกไปที่ Setting > Options แล้วเลือก Tab sheet ไปที่ Misc ดังแสดงในภาพที่ 113



ภาพที่ 113 การเลือกรายละเอียดอื่น ๆ

(7) เลือก Process Priority = Realtime เพื่อให้มี Priority ที่สูงสุดขณะทำการโปรแกรม ให้คลิกไปที่ Confirmation Tabsheet ดังแสดงในภาพที่ 114



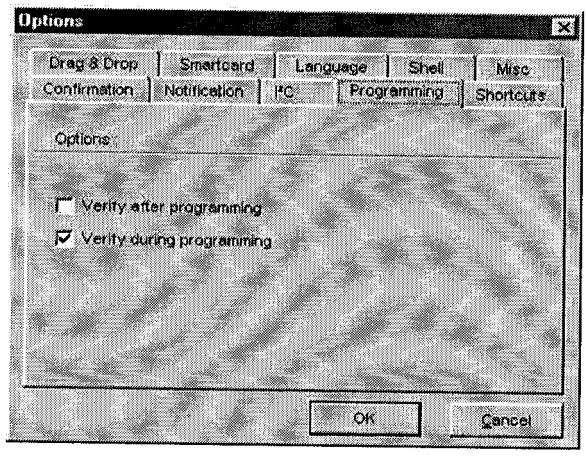
ภาพที่ 114 การเลือกการตรวจสอบการทำงาน

(8) ให้ทำการตรวจสอบโปรแกรม ดังนี้

Check ที่ Programming a device เพื่อให้ program ถามก่อนทำการ program MCU

Check ที่ Erasing a device เพื่อให้ program ถามก่อนทำการลบ program ใน MCU

Check ที่ Overwriting a file เพื่อให้ program ถามก่อนทำการ Save file ชื่อซ้ำกัน คลิก ไปที่ Programming Tabsheet ดังแสดงในภาพที่ 115



ภาพที่ 115 การเลือกวิธีการตรวจสอบขณะทำการโปรแกรม

(9) ให้ทำการเลือกตามรายละเอียด ดังต่อไปนี้

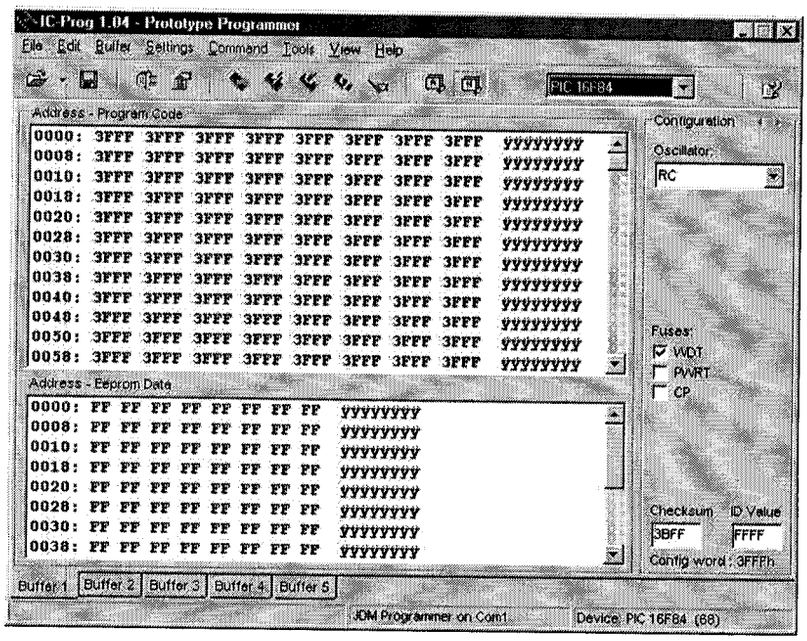
Check ที่ Verify after programming เพื่อให้ทำการ Verify program ขณะทำการ program MCU

Check ที่ Verify during programming เพื่อให้ทำการ Verify program หลังจากทำการprogram MCU เสร็จแล้ว

<u>หมายเหตุ</u> ในการโปรแกรมแบบ CODE PROTECT ถ้าต้องการ verify ต้องใช้วิธีการ Verify during programming เพราะถ้าเลือก Verify after programming จะเกิด Error เพราะ CODE ถูกปิดไม่ให้อ่านได้

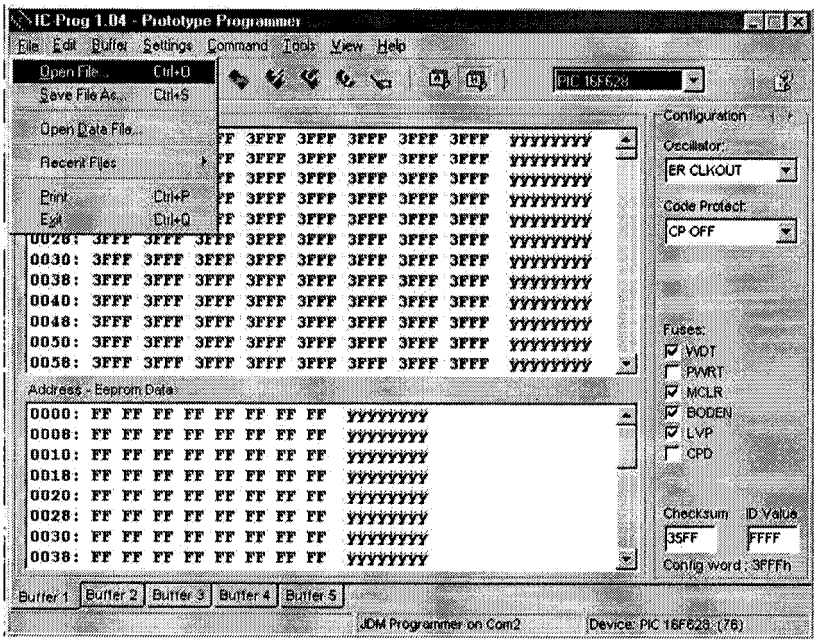
4. โปรแกรมไมโครคอนโทรลเลอร์

4.1 เลือก MCU หรือ EEPROM ให้ตรง ดังแสดงในภาพที่ 116



ภาพที่ 116 หน้าต่างสำหรับการโปรแกรมไมโครคอนโทรลเลอร์

4.2 เลือกที่ File > Open file เพื่อเปิดไฟล์ที่จะโหลดใส่ program memory (ส่วนใหญ่เป็น Hex file) หรือเลือกที่ File > Open Data file เพื่อเปิดไฟล์ที่จะโหลดใส่ไมโครคอนโทรลเลอร์ ดังภาพที่ 117



ภาพที่ 117 วิธีการเพื่อเปิดไฟล์ที่จะโหลดใส่ไมโครคอนโทรลเลอร์ กำหนด Oscillator และ Fuses bit ตามความต้องการ

4.3 ใส่ Chip ให้ตรงกับ Socket ที่ Microcontroller Programmer กำหนด

4.4 กดคลิกที่ Program all Toolbar เพื่อทำการโปรแกรมตามความต้องการ

ภาคผนวก ฉ
ตารางรหัสแอสกี้

ภาคผนวก ฉ
ตารางรหัสแอสกี้

ASCII ย่อมาจาก American Standard Code for Information Interchange โดยที่รหัสแอสกี้เป็นรหัสที่คอมพิวเตอร์สามารถเข้าใจโดยเป็นรหัสที่ใช้แทนตัวเลข ตัวอักษร เครื่องหมายและอักขระพิเศษต่าง ๆ ในการสื่อสาร ในช่วงแรกจะมีการใช้งานกับเครื่องมือสื่อสารประเภทเครื่องโทรพิมพ์ จากเดิมนั้นรหัสแอสกี้มีใช้กัน 7 บิต ทำให้มีรหัสที่ใช้เพียง 128 ตัว ต่อมาได้ขยายเป็น 8 บิต สำหรับ 32 ตัวแรก (0 ถึง 31) จะเป็นอักขระพิเศษที่ใช้ควบคุม ที่เรียกว่า Non-Printing Character [7]

ตารางที่ 16 ตารางรหัสแอสกี้

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

ประวัติผู้วิจัย

ชื่อ นายธวัชชัย สิงหัษฐิต

ประวัติการศึกษา

โรงเรียนเบ็ญจะมะมหาราช จังหวัดอุบลราชธานี
มัธยมศึกษาตอนต้น พ.ศ. 2522 – 2524
วิทยาลัยเทคนิคจังหวัดอุบลราชธานี
ระดับ ปวช.(อิเล็กทรอนิกส์) พ.ศ. 2524 – 2527
สถาบันเทคโนโลยีปทุมวัน จังหวัดกรุงเทพฯ
ระดับ ปวส.(อิเล็กทรอนิกส์อุตสาหกรรม)
พ.ศ. 2527 – 2529
มหาวิทยาลัยภาคตะวันออกเฉียงเหนือ จังหวัดขอนแก่น
ระดับปริญญาตรี อุตสาหกรรมศาสตรบัณฑิต
(วิศวกรรมอิเล็กทรอนิกส์) พ.ศ. 2535 - 2537

ประวัติการวิจัย -

ประวัติการทำงาน

พ.ศ. 2529 – 2538 ช่างโทรคมนาคม ระดับ 2 - 4
ศูนย์โทรคมนาคมขอนแก่น (กสท.) จ.ขอนแก่น
พ.ศ. 2538 – 2542 วิศวกร ระดับ 4 - 6
ชุมสาย Telex (กสท.) บางรัก จ.กรุงเทพมหานคร
พ.ศ. 2542 – 2549 วิศวกร ระดับ 6 - 8
สถานีดาวเทียมสิรินธร(กสท.) อ.สิรินธร จ.อุบลราชธานี

ตำแหน่งและสถานที่ทำงานปัจจุบัน

วิศวกร ระดับ 8
หน่วยระบบและเครือข่าย สำนักงานบริการลูกค้า
ขอนแก่น (กสท.) ถ.ศูนย์ราชการ อ.เมือง จ.ขอนแก่น
โทรศัพท์ (043) 241560 ต่อ 4107-8
E-mail tawatsing@cattelecom.com