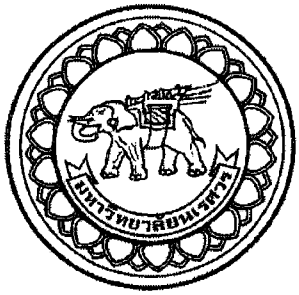


ระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

Automatic Turn On and Turn Off Computer System

นางสาวบุษบา แสงสว่าง รหัส 47380028

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2550

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	ระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ
ผู้ดำเนินโครงการ	นางสาวบุษบา แสงสว่าง รหัส 47380028
อาจารย์ที่ปรึกษา	ดร.อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ดร.อัครพันธ์ วงศ์กังแห)

.....กรรมการ
(ดร.ชัยรัตน์ พินทอง)

.....กรรมการ
(นางสาวศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	ระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ
ผู้ดำเนินโครงการ	นางสาวบุษบา แสงสว่าง รหัส 47380028
อาจารย์ที่ปรึกษา	ดร.อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

บทคัดย่อ

โครงงานนี้มีจุดมุ่งหมายเพื่อพัฒนาชุดคำสั่งที่ใช้ในการตั้งเวลาการเปิด-ปิดคอมพิวเตอร์ พร้อมกับสร้างอุปกรณ์ที่ใช้ในการรับส่งข้อมูลและเปิดคอมพิวเตอร์

จากผลการทดสอบการตั้งเวลาเปิดและปิดคอมพิวเตอร์ในระยะเวลาที่แตกต่างกันจำนวน 10 ครั้งด้วยชุดคำสั่งและอุปกรณ์ที่พัฒนาขึ้น พบว่า สามารถเปิดและปิดคอมพิวเตอร์ได้อย่างอัตโนมัติและตรงตามระยะเวลาที่กำหนดได้อย่างถูกต้อง ทำให้สามารถนำมาใช้ในการควบคุมการเปิด-ปิดเครื่องคอมพิวเตอร์ที่ใช้เป็นเครื่องเซิร์ฟเวอร์ที่มีการใช้งานเป็นช่วงเวลาได้อย่างอัตโนมัติ

Project Title Automatic turn on and turn off computer System

Name Miss Budsaba Sangswang ID. 47380028

Project Advisor Akaraphunt Vongkunghae, Ph.D.

Major Computer Engineering

Department Electrical and Computer Engineering

Academic 2006

.....

ABSTRACT

The purpose of this project is to build interface hardware and utility software for controlling the time to turn on and off a personal computer (PC).

This automation turn on and turn off system (ATTS) aims to save energy wrong for computer clear rooms which are mostly occupied on day time but were left unattentionally at nighttime. The computer disable to shutdown at off work time and readied to use in the morning workday. Which the time can be set by user

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากปัจจุบันเทคโนโลยีต่างๆได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์เรามากขึ้น เช่น ใช้ในการเรียนการสอน ใช้ในการทำงานธุรกิจต่างๆทั้งในภาครัฐและเอกชน เป็นต้น ซึ่งในการทำงานขององค์กรต่างๆนั้นมักจะมีเซิร์ฟเวอร์เป็นเครื่องคอมพิวเตอร์ที่บันทึกข้อมูลและโปรแกรมเอาไว้เพื่อให้ผู้ที่ใช้เครื่องคอมพิวเตอร์อื่นๆ ในเครือข่ายนั้นๆเข้ามาใช้ข้อมูลได้ แต่ในบางครั้งผู้ที่ดูแลเซิร์ฟเวอร์อาจจะมาสายหรือไม่มาทำงาน และไม่มีผู้ที่คอยดูแลดังนั้นจึงทำให้ผู้ใช้งานคอมพิวเตอร์คนอื่นๆไม่สามารถที่จะทำงานที่เกี่ยวข้องกับข้อมูลที่เก็บไว้ในฐานข้อมูล ได้ และส่งผลกระทบกับงานนั้นๆ

นอกจากนี้โปรแกรมที่ใช้ในการตั้งเวลาเปิด-ปิดคอมพิวเตอร์นั้นยังไม่มี ส่วนใหญ่ที่มีใช้นั้นจะเป็นเฉพาะการปิดคอมพิวเตอร์เท่านั้น และโปรแกรมการเปิดคอมพิวเตอร์ที่มีใช้ปัจจุบันนั้นจะเป็นการตั้งเวลาสำหรับช่วงเวลาที่สามารถเข้าไปใช้งานคอมพิวเตอร์ หากยังไม่ถึงเวลาที่ตั้งไว้คอมพิวเตอร์ก็จะทำการ Shutdown ตัวเองทันที

ผู้จัดทำโครงงานได้เล็งเห็นความสำคัญในเรื่องนี้ ดังนั้นจึงได้เสนอโครงงานการสร้างระบบเปิด-ปิดเครื่องคอมพิวเตอร์อัตโนมัติขึ้น เพื่อแก้ไขปัญหาที่เกิดขึ้นเหล่านี้และเพิ่มประสิทธิภาพในการทำงานให้ดีขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อจัดทำระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ

1.2.2 เพื่อพัฒนาโปรแกรมในการติดต่อสื่อสารกับไมโครคอนโทรลเลอร์ที่ใช้ในการเปิด-ปิดเครื่องคอมพิวเตอร์

1.2.3 เพื่อจัดทำอุปกรณ์รับข้อมูลเข้าไปประมวลผลในการเปิด-ปิดเครื่องคอมพิวเตอร์

1.3 ขอบข่ายของโครงงาน

1.3.1 ออกแบบสร้างและทดสอบระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ

1.3.2 ออกแบบและสร้างไมโครคอนโทรลเลอร์ที่ใช้ในการรับส่งข้อมูลในการเปิดปิดคอมพิวเตอร์

1.3.3 ออกแบบและพัฒนาซอร์ฟแวร์ในการเปิดปิดเครื่องคอมพิวเตอร์

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาการสร้างอุปกรณ์เชื่อมต่อข้อมูลกับคอมพิวเตอร์ทั้งฮาร์ดแวร์และซอฟท์แวร์

1.4.2 ออกแบบตัวคอนโทรลเลอร์ที่จะใช้ในการรับส่งข้อมูลและการนับเวลาเพื่อทำการเปิดและปิดคอมพิวเตอร์

1.4.3 สร้างอุปกรณ์ที่จะใช้ในการรับส่งข้อมูลและการนับเวลาเพื่อทำการเปิดปิดคอมพิวเตอร์

1.4.4 ออกแบบโปรแกรมระบบการเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ

1.4.5 พัฒนาโปรแกรมระบบการเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ

1.4.6 ทดสอบการใช้งานระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติและตรวจสอบแก้ไขข้อผิดพลาด

1.4.7 จัดทำรูปเล่มรายงาน

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงานของการทำงานระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ

กิจกรรม	ระยะเวลาดำเนินการ	
	มี.ค	เม.ย
1.5.1 ศึกษาการสร้างอุปกรณ์เชื่อมต่อข้อมูลกับคอมพิวเตอร์ทั้งฮาร์ดแวร์และซอฟท์แวร์	■	
1.5.2 ออกแบบตัวคอนโทรลเลอร์ที่จะใช้ในการรับส่งข้อมูลและการนับเวลาเพื่อทำการเปิดและปิดคอมพิวเตอร์	■	
1.5.3 สร้างอุปกรณ์ที่จะใช้ในการรับส่งข้อมูลและการนับเวลาเพื่อทำการเปิดปิดคอมพิวเตอร์	■	
1.5.4 ออกแบบโปรแกรมระบบการเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ	■	
1.5.5 พัฒนาโปรแกรมระบบการเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ		■
1.5.6 ทดสอบการใช้งานระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติและตรวจสอบแก้ไขข้อผิดพลาด		■
1.5.7 จัดทำรูปเล่มรายงาน		■

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 ได้เครื่องมือและโปรแกรมที่ทำให้เครื่องคอมพิวเตอร์สามารถเปิด-ปิดตามเวลาที่กำหนดไว้

1.6.2 ผู้ใช้คอมพิวเตอร์ตั้งโต๊ะ (Personal Computer: PC) ทั่วไปสามารถนำไปใช้กับคอมพิวเตอร์ของตัวเองได้

1.6.3 สามารถในไปใช้กับคอมพิวเตอร์ที่ใช้เป็นเครื่องเซิร์ฟเวอร์ได้

1.6.4 ผู้อื่นสามารถนำไปศึกษาและพัฒนาต่อไปได้

1.7 งบประมาณที่ใช้

- ค่าหนังสือ	500	บาท
- ค่าถ่ายเอกสารและจัดทำรูปเล่ม	500	บาท
- ค่าวัสดุสร้างชิ้นงาน	500	บาท
รวม	1,500	บาท

บทที่ 2
หลักการและทฤษฎี

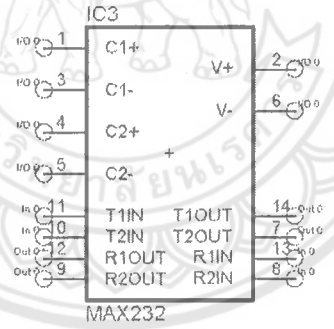
ในการทำโครงงานระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติคอมพิวเตอร์ จะต้องศึกษาการการติดต่อกับคอมพิวเตอร์ทั้งในส่วนของฮาร์ดแวร์และซอฟต์แวร์ก่อน เพื่อนำไปประยุกต์ใช้ในการสร้างระบบเปิดและปิดเครื่องคอมพิวเตอร์อัตโนมัติ จึงศึกษาทฤษฎีต่างๆดังนี้

1. การสื่อสารข้อมูลผ่านพอร์ตอนุกรม
2. การเขียนโปรแกรมติดต่อกับพอร์ตอนุกรมโดย Visual Basic 6.0
3. พื้นฐานการสื่อสารแบบอนุกรมโดย Visual Basic 6.0
4. การใช้งานฟังก์ชัน API ด้วย Visual Basic 6.0

2.1 การสื่อสารข้อมูลผ่านพอร์ตอนุกรม

2.1.1 Max232

MAX232 เป็นไอซีตัวหนึ่งที่ทำหน้าที่ติดต่อกับพอร์ต RS232 โดยจะรับข้อมูลเข้ามาทางอินพุต แล้วส่งให้พอร์ตRS232 ทางเอาต์พุตของตัวไอซีเองและจะรับข้อมูลจากRS232 แล้วส่งออกไปใช้งานอีกด้วย ลักษณะของMAX232แสดงดัง รูปที่ 2.1



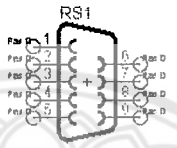
รูปที่ 2.1 แสดงขาต่างๆ ของ MAX232

ในการวัดข้อมูลที่ได้นี้เราจะได้ผลลัพธ์ออกมาเป็นแรงดันและแรงดันนั้นจะมีค่าความต่างศักย์ที่สูง เราจึงจำเป็นที่จะต้องทำการลดทอนสัญญาณเพื่อให้นำมาใช้กับไมโครคอนโทรลเลอร์ ในการแปลงสัญญาณอะนาลอกเป็นดิจิตอลเนื่องจากวงจรอิเล็กทรอนิกส์นั้นสามารถรับแรงดันได้เพียง 5 โวลต์ เท่านั้นในโครงงานนี้เราจะใช้วงจรแบ่งแรงดัน (Voltage Divider) เป็นตัวที่จะลดทอนแรงดันที่ได้ให้เป็นไปตามที่เราต้องการ

2.1.2 RS232

ในการที่จะติดต่อกับคอมพิวเตอร์นั้นไม่เพียงแต่ทำการแปลงอะนาลอกเป็นดิจิตอลเท่านั้นแต่เราจำเป็นต้องเลือกพอร์ตที่จะทำการเชื่อมต่อด้วยซึ่งมีอยู่หลายแบบ เช่น พอร์ตขนาน, พอร์ตอนุกรม แล้วพอร์ตอนุกรมก็สามารถแบ่งได้หลายแบบ เช่น DB9, DB15, DB25 ในที่นี้เราเลือกพอร์ต DB9 ในการติดต่อกับคอมพิวเตอร์ พอร์ตRS232 แบบ DB9 แสดงดังรูปที่ 2.2

ในการสื่อสารข้องมูลผ่านพอร์ตอนุกรมนี้จะมีสายส่งข้อมูลอยู่ 2 เส้นคือขา Txd (Transmitted Data) และ Rxd (Received Data) ขา Txd คือขาที่จะใช้ในการส่งข้อมูลส่วนขา Rxd ทำหน้าที่ในการรับข้อมูล หน้าที่การทำงานที่ของแต่ละขาของพอร์ต DB9 แสดงในตารางที่2.1



รูปที่ 2.2 ช่องพอร์ต DB9

ตารางที่ 21 แสดงคุณสมบัติขาต่างๆ ของพอร์ต

Pin	Signal
1	Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

สำหรับรายละเอียดของสายสัญญาณนั้นประกอบไปด้วย

- Carrier Detect (CD) ขานี้จะทำงานก็ต่อเมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม
- Received Data (RD) ใช้สำหรับรับข้อมูลอนุกรมเข้ามายังคอมพิวเตอร์
- Transmitted Data (TD) ใช้สำหรับส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์

- Data Terminal Ready (DTR) ใช้สำหรับบอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง
- Signal Ground (SG) เป็นกราวน์ของระบบ
- Data Set Ready (DSR) ใช้สำหรับตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง จะใช้คู่กับขา DTR
- Request to Send (RTS) ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทาง เพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา
- Clear to Send (CTS) ใช้สำหรับตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อด้วยพร้อมที่จะรับข้อมูลหรือไม่ โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะทำการส่งข้อมูลออกทางขา TD
- Ring Indicator (RI) ขานี้จะทำงานเมื่อโมเด็มได้รับสัญญาณเรียกเข้าจากสายโทรศัพท์

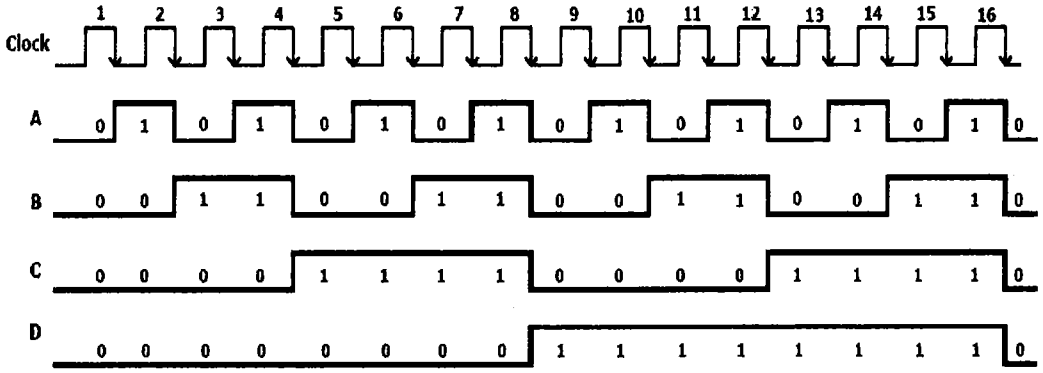
2.2 พื้นฐานการสื่อสารแบบอนุกรมโดย Visual Basic 6.0

การสื่อสารแบบอนุกรมในเครื่องคอมพิวเตอร์จะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน เพราะว่าการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลครั้งละ 1 บิต แต่พอร์ตขนานสามารถส่งข้อมูลได้ครั้งละหลายๆ บิตพร้อมกัน แต่การส่งข้อมูลแบบอนุกรมนั้นมีข้อดีกว่าการส่งข้อมูลแบบขนาน คือ การสามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังน้อยกว่าการส่งข้อมูลแบบขนาน การสื่อสารแบบอนุกรมสามารถแบ่งออกเป็น 3 รูปแบบดังนี้

1. Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว
2. Half-Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลในเวลาเดียวกันได้
3. Full-Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้สามารถแบ่งประเภทของการสื่อสารแบบอนุกรมตามลักษณะสัญญาณในการส่งได้อีก 2 แบบคือ

- การสื่อสารแบบซิงโครนัส(Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณ ซึ่งการสื่อสารแบบซิงโครนัสเหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนัก



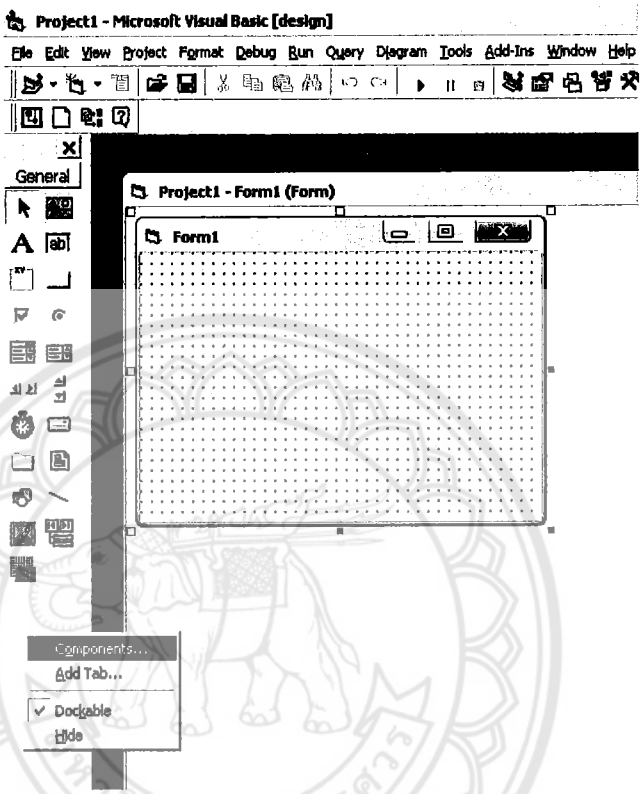
รูปที่ 2.3 ลักษณะสัญญาณการสื่อสารแบบซิงโครนัส

- **การสื่อสารแบบอะซิงโครนัส(Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัส** นั้นจะใช้สายข้อมูลเพียงตัวเดียว แต่จะใช้รูปแบบการส่งข้อมูล **หรือ** Bit Pattern **เป็น** ตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล, ส่วนไหนเป็น**ข้อมูล, ส่วนไหนจะเป็น** ส่วนตรวจสอบความถูกต้องข้อมูล และส่วนไหนเป็นส่วน**ปิดท้ายของข้อมูล** โดย ต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่ง และภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ ชื่อว่า UART หรือ Universal Asynchronous Receiver/Transmitter คอยควบคุมการ รับและส่งข้อมูล

รูปที่ 2.4 UART อุปกรณ์ควบคุมการส่งสัญญาณแบบอะซิงโคนัส

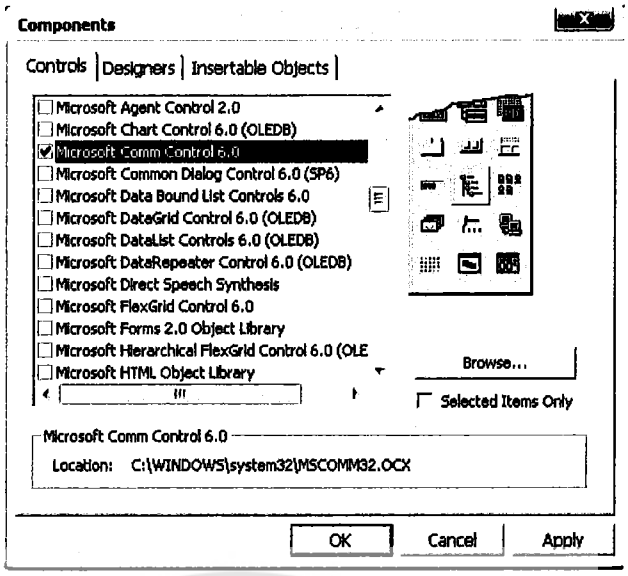
2.3 การเขียนโปรแกรมติดต่อกับพอร์ตอนุกรมโดย Visual Basic 6.0

คอนโทรลที่สำคัญในการทำให้ Visual Basic สามารถสื่อสารผ่านพอร์ตอนุกรมได้นั่นก็คือ คอนโทรล MSComm ซึ่งไม่ใช่คอนโทรลมาตรฐาน ดังนั้นถ้าเราต้องการใช้งาน MSComm เราจะต้องทำการเพิ่มคอนโทรลนี้เข้าไปใน ToolBox ซึ่งสามารถกระทำได้โดยคลิกขวาที่ ToolBox แล้วเลือกเมนู Components ดังรูปที่ 2.5



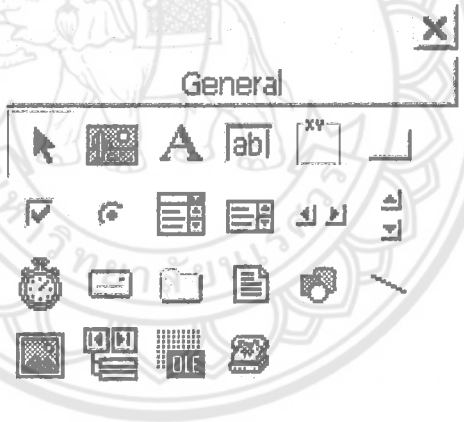
รูปที่ 2.5 การเรียกคำสั่ง Components

จากนั้นจะปรากฏไดอะล็อก Components ขึ้นมา จากนั้นให้คลิกเลือกที่ Microsoft Comm Control 6.0 แล้วคลิกปุ่ม OK ดังรูปที่ 2.6



รูปที่ 2.6 การเลือก Microsoft Comm Control 6.0

จากนั้นจะปรากฏภายใน Toolbox จะมีไอคอนรูปโทรศัพท์ ซึ่งเป็นไอคอนของคอนโทรล MSComm ปรากฏขึ้นมาให้เราเลือกใช้งานดังรูปที่ 2.7



รูปที่ 2.7 การแสดงไอคอน Microsoft Comm Control 6.0

2.3.1 พร็อพเพอร์ตี้(Property) ที่สำคัญในการใช้งาน MSComm

- CommPort ใช้ในการกำหนดหมายเลขของพอร์ตอนุกรมที่เราต้องการจะติดต่อ โดยมีรูปแบบของการใช้งานดังนี้

  object.CommPort [= value]

- Setting ใช้ในการกำหนดอัตราบอด (Baud Rate) หรือความเร็วในการส่งข้อมูล มีหน่วยเป็นบิตต่อวินาที, พาริตี้, จำนวนของบิตข้อมูล, จำนวนของบิตปิดท้าย โดยมีรูปแบบของการใช้งานดังนี้

  object.Setting [= value]

- PortOpen ใช้สำหรับเปิดและปิดการใช้งานพอร์ตอนุกรม โดยมีรูปแบบการทำงานดังนี้

  object.PortOpen [= value]

- InBuffersize เป็นการกำหนดขนาดของ Buffer ในการรับข้อมูลเข้ามา โดยมีรูปแบบการทำงานดังนี้

  object.InBuffersize [= value]

- OutBuffersize เป็นการกำหนดขนาดของ Buffer ในการส่งข้อมูลออกไป โดยมีรูปแบบการกำหนดค่าดังนี้

  object.OutBuffersize [= value]

- Inputlen เป็นการกำหนดค่าของข้อมูลที่อ่านจาก Buffer ภาครับ โดยมีรูปแบบการกำหนดค่าดังนี้

  object.Inputlen [= value]

- InputMode เป็นการกำหนดค่าชนิดของข้อมูลที่รับเข้ามา โดยมีรูปแบบการกำหนดค่าดังนี้

  object.InputMode [= value]

  โดยที่เราสามารถเลือกชนิดของข้อมูลได้ 2 ประเภท คือ

  - ComInputModeText ข้อมูลที่รับเข้ามาเป็นข้อความปกติเราสามารถตั้งค่าให้อยู่ในโหมดโดยการกำหนด value ให้เป็น “0”
  - ComInputModeBinary ข้อมูลที่รับเข้ามาเป็นข้อมูลไบนารี่ เราสามารถตั้งค่าให้อยู่ในโหมดโดยการกำหนด value ให้เป็น “1”

- Input ใช้ในการอ่านค่าข้อมูลจากพอร์ตอนุกรม โดยมีรูปแบบการอ่านค่าดังนี้

  object.input

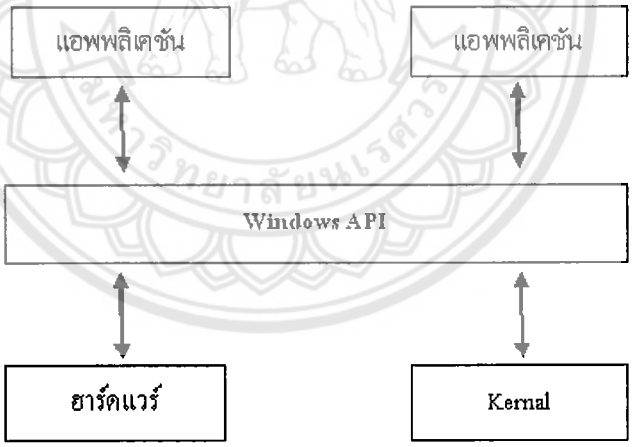
2.4 การใช้งานฟังก์ชัน API ด้วย Visual Basic 6.0

เนื่องจาก Visual Basic เองอาจจะมีความสามารถไม่ครอบคลุมการทำงานทั้งหมดทำให้ในงานบางอย่างต้องพึ่งพาความสามารถของวินโดว์ โดยวินโดว์ก็ได้เตรียมฟังก์ชันชนิดต่างๆไว้รองรับการทำงานมากมายนับพันฟังก์ชัน ซึ่งเราเรียกฟังก์ชันเหล่านั้นว่า Windows API

Windows API ย่อมาจาก Windows Application Programming Interface เป็นชุดฟังก์ชันที่พร้อมให้ใช้งาน ทำให้เราไม่ต้องเขียนโค๊ดยาวๆเพื่อทำงานที่ซับซ้อน ซึ่งวินโดว์ทำได้ดีกว่าและเร็วกว่า เช่นถ้าต้องการทราบว่า เนื้อที่ว่างในฮาร์ดดิสก์เหลือเท่าไหร่ ก็สามารถใช้ฟังก์ชัน GetDiskFreeSpace เพียงฟังก์เดียวก็ได้คำตอบ เป็นต้น

ฟังชันของ Windows API นั้นมีนับพันฟังก์ชัน แบ่งออกเป็นหมวดหมู่ตามประเภทของการใช้งาน เช่น API ด้านกราฟฟิก, API ด้านเน็ตเวิร์ก, API ด้านการพิมพ์ เป็นต้น

การใช้งาน Windows API นั้นทำให้เราได้วิธีการใช้งานวินโดว์ที่มีมาตรฐานเดียว ช่วยลดเวลาการพัฒนาฟังก์ชันขึ้นมาเอง แถมยังลดความผิดพลาดจากการเรียกใช้งานฮาร์ดดิสก์โดยตรงอีกด้วย โดย Windows API จะอยู่ตรงกลางระหว่างแอพพลิเคชันกับฮาร์ดแวร์ และระบบปฏิบัติการวินโดว์ ดังรูปที่ 2.8



รูปที่ 2.8 แนวคิดการทำงานของ Windows API

นอกจากนี้การใช้งานฟังก์ชันของ Windows API ยังมีข้อดีอีกคือ ไม่ต้องกังวลเรื่องของการเปลี่ยนแปลงเวอร์ชันวินโดว์มากนัก เพราะไม่ว่าจะใช้วินโดว์ต่างเวอร์ชันก็ยังใช้ชื่อเช่นเดิม แต่การทำงานภายในเปลี่ยนแปลงไปตามความสามารถที่เพิ่มขึ้น ซึ่งทำให้เราไม่จำเป็นต้องติดตามความเปลี่ยนแปลงที่เกิดขึ้นมากนัก ยกวันว่าจะมีการยกเลิกการใช้งานในฟังก์ชันใดฟังก์ชันหนึ่ง

2.4.1 กลุ่มของฟังก์ชัน Windows API

Windows API นั้นบางคนเรียนว่า Win32API เป็นชุดฟังก์ชันนับพันๆฟังก์ชันที่ระบบปฏิบัติการตระกูลวินโดว์ตั้งแต่ Window 95, 98, 2000, Me, XP ได้เตรียมไว้ให้เราได้ใช้งานความสามารถของตัวระบบปฏิบัติการ โดยแบ่งเป็นหมวดหมู่ และเก็บไว้ในไฟล์ .DLL ในโฟล์เดอร์ C:Windows\System หรือ C:WINNT\System32 ซึ่งไฟล์ที่สำคัญโดยแสดงไว้ในตาราง 2.2 ดังนี้

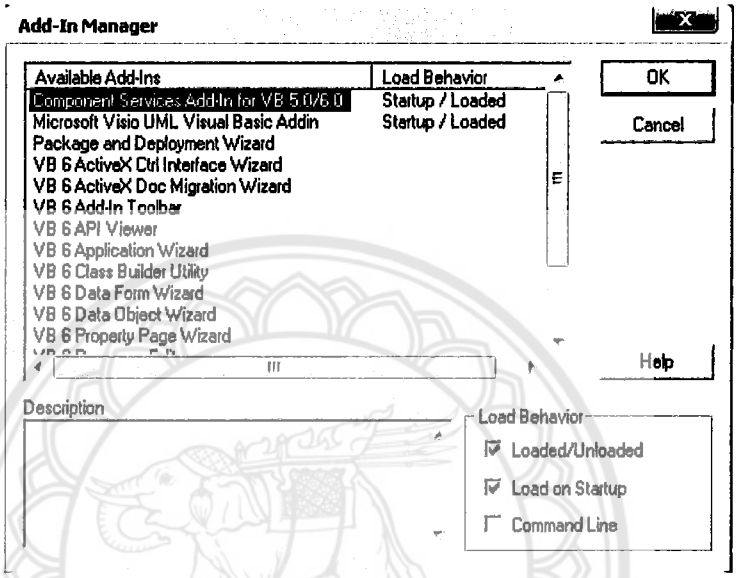
ตาราง 2.2 ไฟล์ .DLL ที่สำคัญในโฟล์เดอร์ C: Windows\System หรือ C: WINNT\System32

ชื่อไฟล์	คำอธิบาย
AVAPI32.DLL	มาจาก Advanced API ซึ่งจะเก็บฟังก์ชันเกี่ยวกับความปลอกภัย (Security) และ Registry
COMDLG.DLL	เก็บฟังก์ชันเกี่ยวกับ Common Dialog เช่น ไดอะล็อก Open, ไดอะล็อก Save เป็นต้น
GDI32.DLL	จะเก็บฟังก์ชันเกี่ยวกับส่วนติดต่อผู้ใช้แบบกราฟฟิก หรือ GUI (Graphic Device Interface) ทั้งที่เป็นฟังก์ชันเกี่ยวกับการแสดงผล และฟังก์ชันด้านกราฟฟิก
KERNEL32.DLL	เก็บฟังก์ชันในส่วนที่ใช้จัดการทรัพยากรของระบบปฏิบัติการ
LZ332.DLL	เก็บฟังก์ชันเกี่ยวกับการบีบอัดข้อมูล
MPR.DLL	เก็บฟังก์เกี่ยวกับ Multiple Provide Router
NETAPI32.DLL	เก็บฟังก์ชันเกี่ยวกับการจัดการเครือข่าย (Network)
SHELL32.DLL	เก็บฟังก์ชันเกี่ยวกับเชลล์ หรือคำสั่งทั่วไปของระบบปฏิบัติการ
USER32.DLL	เก็บฟังก์ชันเกี่ยวกับการจัดการผู้ใช้งาน
VERSION.DLL	เก็บฟังก์ชันที่ใช้จัดการรุ่น หรือเวอร์ชันของซอฟต์แวร์
WINMM.DLL	เก็บฟังก์ชันที่จัดการด้านมัลติมีเดีย เช่น เสียง วีดีโอ ฯลฯ

2.4.2 การเรียกใช้งาน Windows API

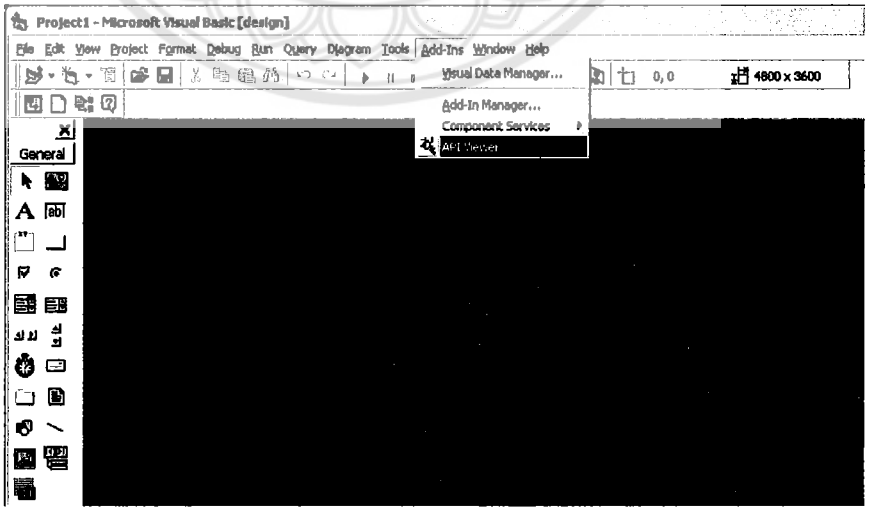
สำหรับการเรียกใช้งานฟังก์ชันของ Windows API ใน Visual Basic นั้นเราจะใช้เครื่องมือชื่อว่า API Viewer ในการเรียกใช้ซึ่งมีขั้นตอนดังนี้

1. เมื่อเข้าใช้งาน Visual Basic ให้เลือกเมนู Add-Ins >Add-In Manager...
2. ในไดอะล็อก Add-In Manager ให้คลิกเลือก VB6 API Viewer
3. คลิกเลือกเช็คบ็อกซ์ Loaded/Unloaded แล้วคลิกปุ่ม OK



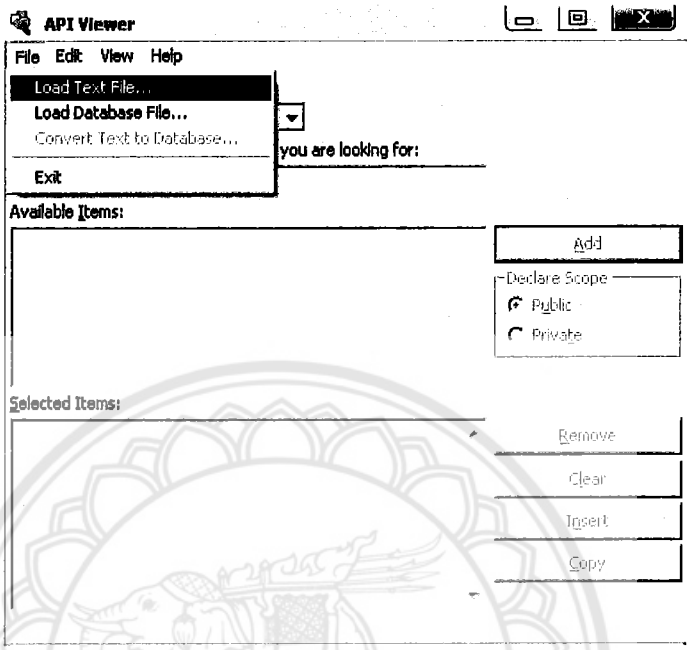
รูปที่ 2.9 การเลือก API Viewer

4. จากนั้น Visual Basic จะแทรกเมนู API Viewer เข้ามาให้ภายใต้เมนู Add-Ins



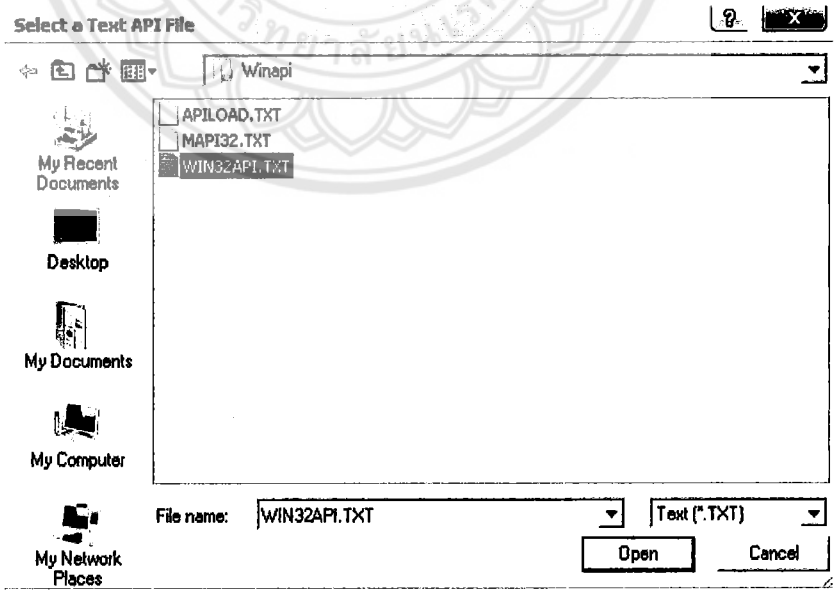
รูปที่ 2.10 การเพิ่มเติมเมนู API Viewer เข้ามา

5. เมื่อเลือกเมนู Add-Ins >API Viewer จะปรากฏหน้าต่าง API Viewer ขึ้นมา
6. ให้เลือกเมนู File > Load Text File เพื่อโหลดเท็กซ์ไฟล์ที่เก็บรายละเอียดเกี่ยวกับฟังก์ชันของ Windows API ขึ้นมาใช้งาน ดังรูปที่ 2.11



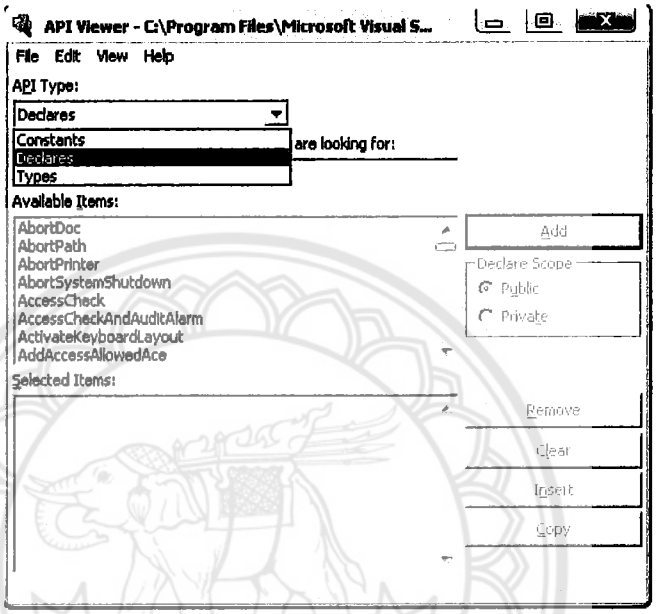
รูปที่ 2.11 การเรียกใช้ Text File

7. ให้เลือกไฟล์ Win32api.txt ขึ้นมา ดังรูปที่ 2.12



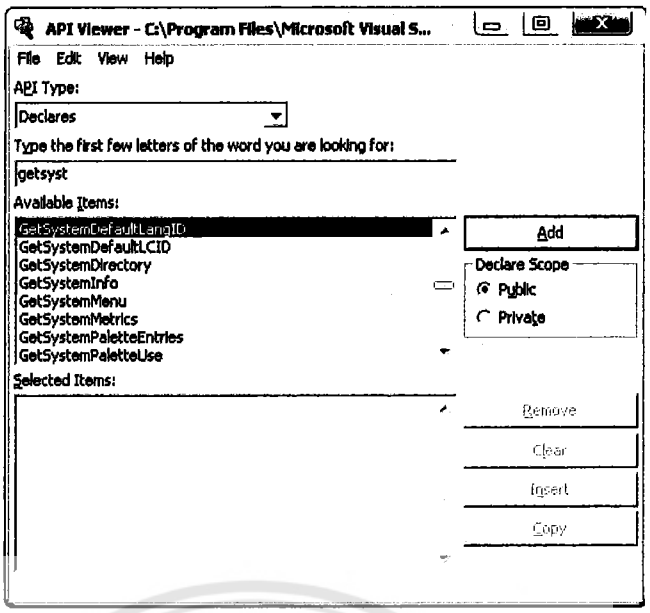
รูปที่ 2.12 การเลือก Text File ที่ต้องการ

8. ในช่อง API Type นั้นเราสามารถเลือกได้ว่าจะดูรายละเอียดเกี่ยวกับอะไร ซึ่งประกอบไปด้วย
   - การประกาศค่า (Declares)
   - ค่าคงที่ (Constant)
   - ชนิดข้อมูลที่ได้มีการประกาศไว้ (Types)



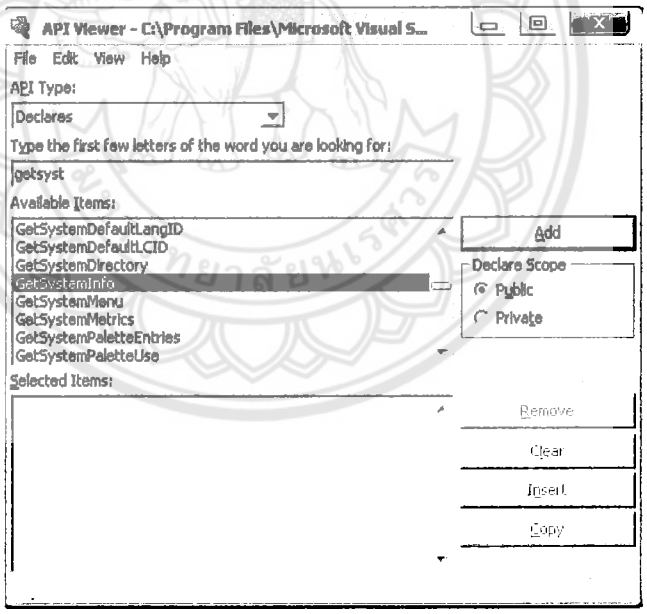
รูปที่ 2.13 การเลือกชนิดที่ต้องการดู

9. ส่วนใหญ่แล้วเรามักจะสนใจเกี่ยวกับการประกาศฟังก์ชัน ดังนั้นเราจึงเริ่มจากการพิมพ์ชื่อฟังก์ชันที่ต้องการลงไปในช่อง Type the first few letters of the word you are looking for : ซึ่งเราเก็บเพียงแค่ไม่กี่คำที่เป็นชื่อของฟังก์ชันที่ต้องการ ที่ช่อง Available Items: ก็จะเลือกชื่อฟังก์ชันที่ใกล้เคียงมาให้เราได้เลือกแล้ว
10. คลิกเลือกชื่อฟังก์ชันที่ต้องการ แล้วคลิกปุ่ม Add



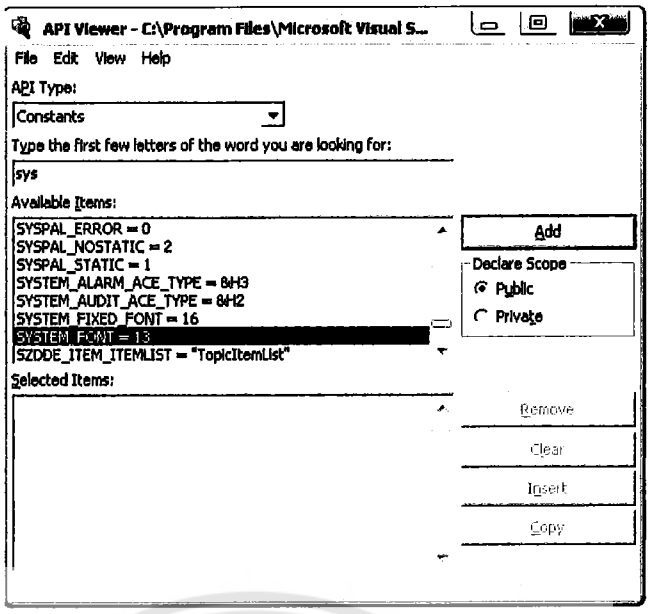
รูปที่ 2.14 การพิมพ์ชื่อฟังก์ชันที่สนใจเข้าไป

11. ฟังก์ชันที่เราได้เลือกจะเข้ามาอยู่ในช่อง Selected Items : จากนั้นให้เราคัดลอกไปยัง Code Window ของโมดูลหรือฟอร์มที่เราต้องการใช้งานทันที



รูปที่ 2.15 แสดงการประกาศฟังก์ชันที่ถูกเลือก

12. กรณีที่เราเลือกดูค่าคงที่ก็สามารถทำได้โดยเปลี่ยนแปลงที่ช่อง API Type : ให้เป็นรูปแบบที่ต้องการ แล้วใช้วิธีการค้นหาคล้ายๆกับการประกาศฟังก์ชัน



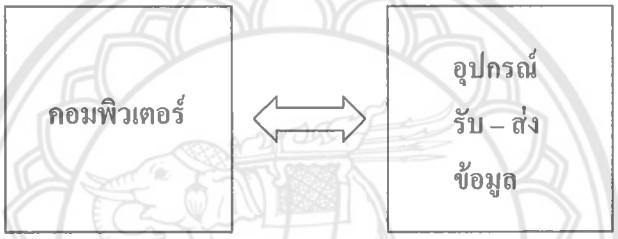
รูปที่ 2.16 ตัวอย่างการใช้งานค่าคงที่

บทที่ 3
การออกแบบระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

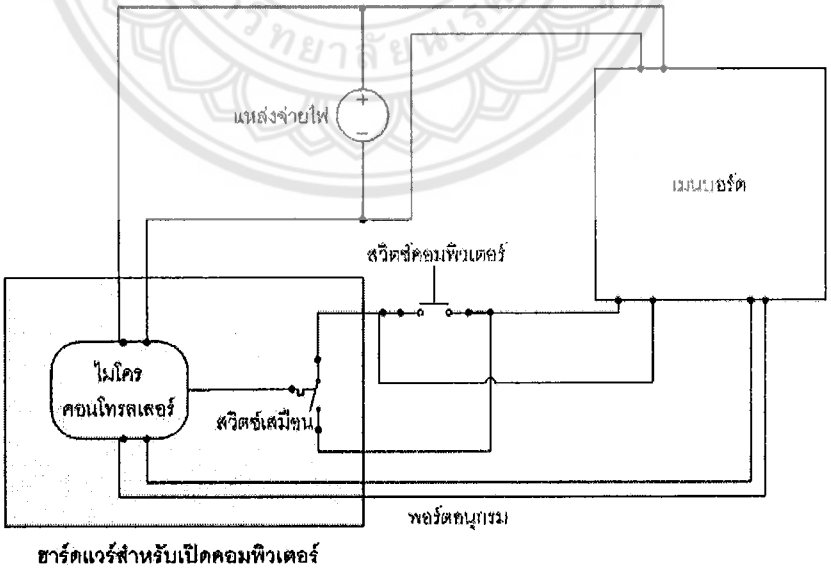
เมื่อได้ทำการศึกษาทฤษฎีที่เกี่ยวข้องที่จะใช้ในการออกแบบระบบเปิดปิดคอมพิวเตอร์อัตโนมัติแล้ว จากนี้จะเป็นการออกแบบระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ ซึ่งสามารถแบ่งได้สองส่วนด้วยกันคือส่วนที่เป็นอุปกรณ์รับ-ส่งข้อมูลหรือไมโครลคอนโทรลเลอร์และส่วนของโปรแกรมที่ใช้ติดต่อกับผู้ใช้งาน รายละเอียดของการออกแบบมีดังนี้

3.1 การออกแบบอุปกรณ์รับ-ส่งข้อมูล

อุปกรณ์รับ-ส่งข้อมูล ซึ่งมีหน้าที่ในการรับสัญญาณอินพุตต่างๆจากคอมพิวเตอร์เพื่อนำไปประมวลผล โดยมีขั้นตอนการทำงานซึ่งสามารถแสดงได้ดังรูปที่ 3.1 และรูปที่ 3.2

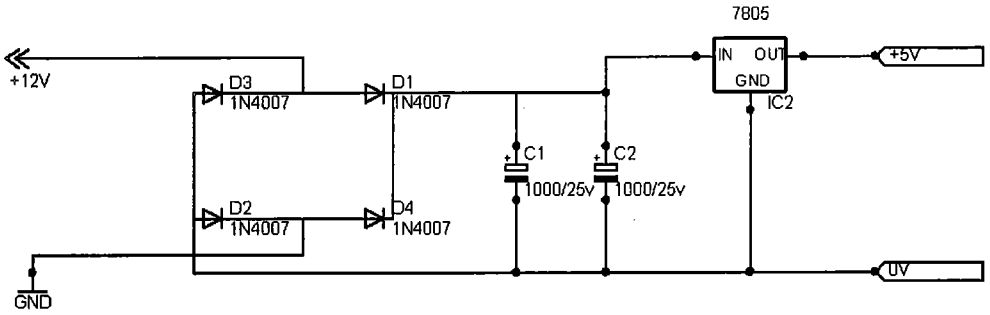


รูปที่ 3.1 โครงสร้างการทำงานโดยรวมของระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ



รูปที่ 3.2 โครงสร้างการทำงานโดยรวมของการเปิดคอมพิวเตอร์อัตโนมัติ

อุปกรณ์รับ – ส่งข้อมูลจะทำงานได้ก็ต้องมีแหล่งจ่ายไฟให้ โดยอุปกรณ์รับส่งที่สร้างขึ้นนี้จะใช้ไฟฟ้ากระแสตรงแรงดัน 5 โวลต์ซึ่งแสดงดังรูปที่ 3.3



รูปที่ 3.3 วงจรจ่ายไฟในอุปกรณ์รับ-ส่งข้อมูล

การทำงานของวงจรจ่ายไฟคือ ไดโอดจะผลัดกันนำกระแสครั้งละ 2 ตัว โดยเมื่อไซเคิลบวกของแรงดันไฟสลับ (Vin) ปรากฏที่ด้านบนของขดทุติยภูมิของหม้อแปลงและด้านล่างจะเป็นลบ จะทำให้ไดโอด D_1 และ D_2 ได้รับไบอัสตรงจะมีกระแสไหลผ่านไดโอด D_1 ผ่านโหลด R_L ผ่านไดโอด D_2 ครบวงจรที่หม้อแปลงด้านล่าง มีแรงดันตกคร่อมโหลด R_L ด้านบนเป็นบวก ด้านล่างเป็นลบ ได้แรงดันไฟช่วงบวกออกทางเอาท์พุท ในช่วงเวลาต่อมาไซเคิลลบของแรงดันไฟสลับ (Vin) ปรากฏที่ด้านบนของขดทุติยภูมิของหม้อแปลง และด้านล่าง เป็นบวก ในช่วงเวลานี้ไดโอด D_1 และ D_2 จะได้รับไบอัสกลับแต่ไดโอด D_3 และ D_4 จะได้รับไบอัสตรง ทำให้มีกระแสไหลผ่านไดโอด D_4 ผ่านโหลด R_L และผ่านไดโอด D_3 ครบวงจร มีแรงดันตกคร่อมโหลด ด้านบนเป็นบวกด้านล่างเป็นลบ ได้แรงดันไฟช่วงบวกออกทางเอาต์พุท และจะได้ไฟฟ้ากระแสตรงแรงดัน 5 โวลต์มาใช้ในการจ่ายไฟฟ้ากับอุปกรณ์รับ - ส่งข้อมูล

ในส่วนของการทำงานของอุปกรณ์รับ-ส่งข้อมูลนี้จะทำการรับสัญญาณอินพุตที่ได้จากการที่ผู้ใช้ได้กรอกข้อมูลตามที่กำหนดไว้ จากนั้นคอมพิวเตอร์ก็จะทำการประมวลผล เมื่อคอมพิวเตอร์ประมวลผลเสร็จแล้วก็จะได้จำนวนเอาต์พุตที่จะส่งไปอุปกรณ์รับ-ส่งข้อมูล จำนวน 5 เอาต์พุตด้วยกัน คือ

- Chk_every คือค่าที่ผู้ใช้ต้องการเปิดคอมพิวเตอร์ทุกวันหรือไม่
- จำนวนวัน คือ ผลต่างของวันที่ที่ผู้ใช้ต้องการจะเปิดกับวันที่ผู้ใช้นั้นได้ทำการตั้งเวลาโดยใช้เวลาของคอมพิวเตอร์เครื่องนั้นๆ ณ วันนั้น เช่น ต้องการเปิดคอมพิวเตอร์ในวันที่ 10 มีนาคม 2551 โดยได้ทำการตั้งเวลาในวันที่ 5 มีนาคม 2551 จำนวนวันที่ได้ก็จะเป็น 5 วัน
- จำนวนชั่วโมง คือ ผลต่างของเวลาที่ผู้ใช้ได้ทำการตั้งเวลาไว้ว่าจะเปิดคอมพิวเตอร์เมื่อไหร่กับเวลาของคอมพิวเตอร์เครื่องนั้นๆ ณ เวลานั้น

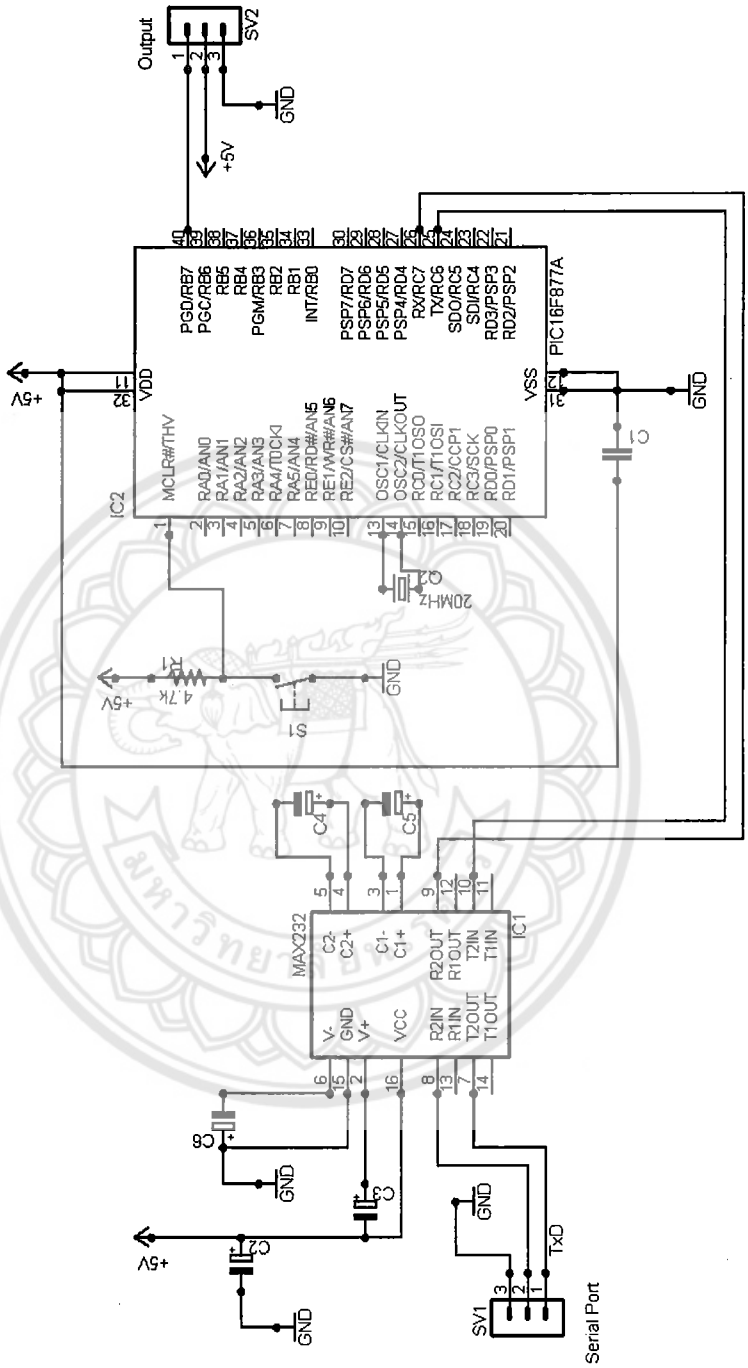
- จำนวนนาที คือ ผลต่างของเวลาที่ผู้ใช้ได้ทำการตั้งเวลาไว้ว่าจะเปิดคอมพิวเตอร์เมื่อไหร่กับเวลาของคอมพิวเตอร์เครื่องนั้นๆ ณ เวลานั้น
- จำนวนวินาที คือ ผลต่างของเวลาที่ผู้ใช้ได้ทำการตั้งเวลาไว้ว่าจะเปิดคอมพิวเตอร์เมื่อไหร่กับเวลาของคอมพิวเตอร์เครื่องนั้นๆ ณ เวลานั้น

เมื่อคอมพิวเตอร์ส่งข้อมูลให้กับอุปกรณ์รับ-ส่งข้อมูลเสร็จแล้ว อุปกรณ์รับส่งข้อมูลก็จะทำการส่งข้อมูลกลับเพื่อให้คอมพิวเตอร์รับรู้ว่าได้รับข้อมูลแล้ว จากนั้นอุปกรณ์รับ-ส่งข้อมูลก็เริ่มทำการประมวลผล ซึ่งในการประมวลผลของไมโครคอนโทรลเลอร์นั้นจะใช้ PIC Basic Pro คอมไพเลอร์ เขียนโปรแกรมในการควบคุมไมโครคอนโทรลเลอร์

ลำดับการทำงานของไมโครคอนโทรลเลอร์จะเริ่มจากเมื่อมีการส่งสัญญาณจากคอมพิวเตอร์มายังไมโครลคอนโทรเลอร์เพื่อตรวจสอบความพร้อมของไมโครคอนโทรลเลอร์ ในกรณีที่ไมโครคอนโทรลเลอร์ได้รับสัญญาณนั้นก็จะทำการส่งสัญญาณตอบกลับไปเพื่อยืนยันว่าพร้อมสำหรับการทำงานแล้ว แต่หากไมโครคอนโทรลเลอร์ไม่มีการตอบกลับไปแสดงว่าไมโครคอนโทรลเลอร์นั้นไม่ได้รับสัญญาณที่คอมพิวเตอร์ส่งมาให้ ดังนั้นให้คอมพิวเตอร์ทำการส่งสัญญาณใหม่ เมื่อไมโครคอนโทรลเลอร์พร้อมทำงานแล้ว คอมพิวเตอร์จะทำการส่งข้อมูลเอาต์พุต จำนวน 4 เอาต์พุตดังข้างต้นไปยังไมรโครคอนโทรลเลอร์ เมื่อไมรโครลคอนโทรลเลอร์ได้รับข้อมูลแล้วก็จะนำไปทำการประมวลผลต่อไป เมื่อไมโครลคอนโทรลเลร์ประมวลผลเสร็จ ต่อไปก็จะเช็คว่าคอมพิวเตอร์นั้นเปิดหรือปิด โดยจะส่งสัญญาณออกไปหากมีการตอบกลับแสดงว่ามีคอมพิวเตอร์นั้นเปิดอยู่ก็ให้ไมโครคอนโทรลเลอร์หยุดการทำงาน แต่หากไม่มีการตอบกลับแสดงว่าคอมพิวเตอร์นั้นปิด เมื่อคอมพิวเตอร์ปิดไมโครคอนโทรลเลอร์ก็จะทำการส่งสัญญาณออกไปเพื่อเปิดคอมพิวเตอร์ต่อไป โดยแสดงดังรูปที่ 3.4

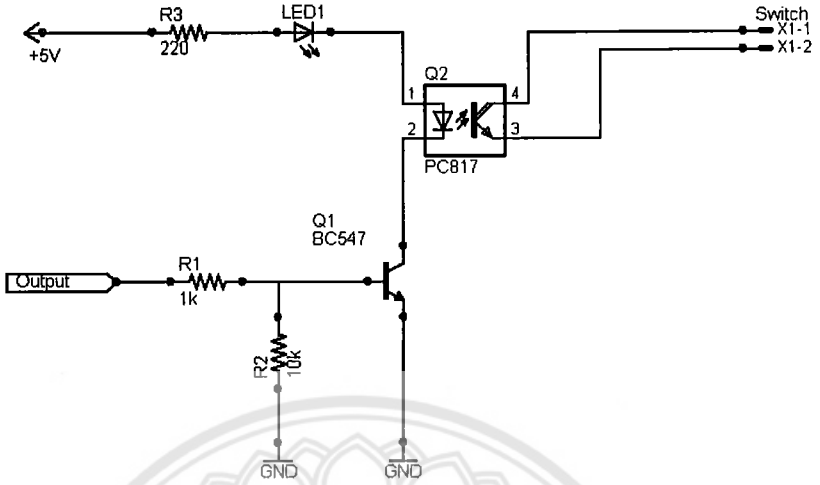
รูปที่ 3.4 การทำงานของอุปกรณ์รับ-ส่งข้อมูล

ในส่วนของไมโครคอนโทรลเลอร์นั้นจะมีวงจรการทำงานดังรูปที่ 3.5



รูปที่ 3.5 วงจรการทำงานของอุปกรณ์รับ-ส่งข้อมูล

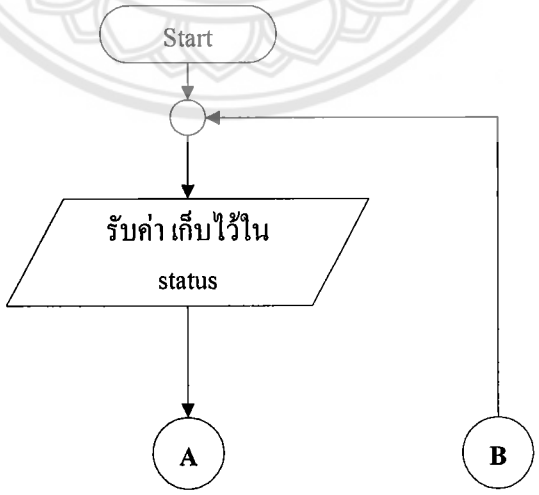
โดยวงจรในส่วนนี้จะทำหน้าที่ในการรับข้อมูลจากคอมพิวเตอร์ที่ผู้ใช้เป็นคนกรอกไว้ เพื่อนำมาประมวลผล และส่งสัญญาณลอจิก "1" ออกไปเพื่อใช้เป็นอินพุตในส่วนของวงจรเอาท์พุต ซึ่งแสดงดังรูปที่ 3.6



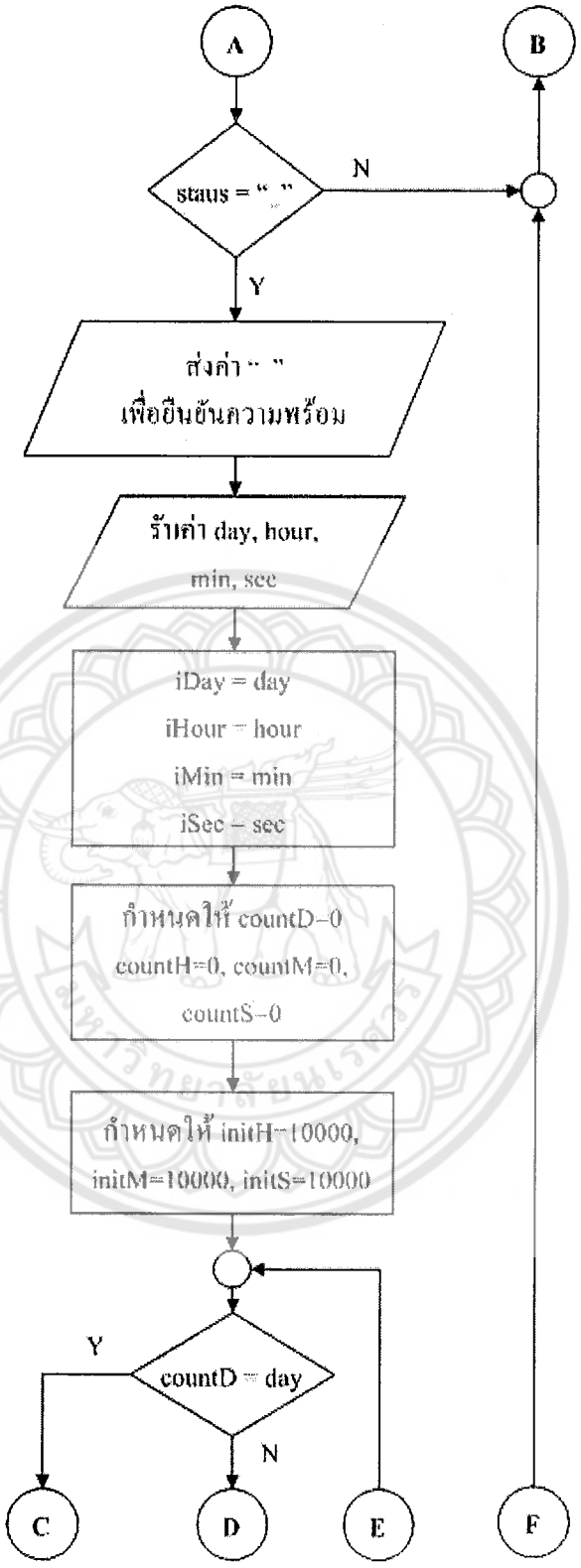
รูปที่ 3.6 วงจรการทำงานของวงจรเอาท์พุต

จากรูปที่ 3.6 จะมีการทำงานคือ เมื่อมีสัญญาณลอจิก "1" เข้ามาทางอินพุตแรงดัน 5 โวลต์นี้จะถูกแบ่งแรงดันด้วยตัวต้านทาน 1k และ 10k โดยแรงดันที่ถูกแบ่งนี้จะไปกระตุ้นทางขา B ของทรานซิสเตอร์ BC547 ส่งผลให้ LED ใน Photocoupler นั้นทำงานจึงทำให้ขา 3 และ ขา 4 ของ Photocoupler นำกระแสตามไปด้วย ส่งผลให้โหลดทางเอาพุตนั้นครบวงจร จึงทำให้คอมพิวเตอร์เปิดเองอัตโนมัติ

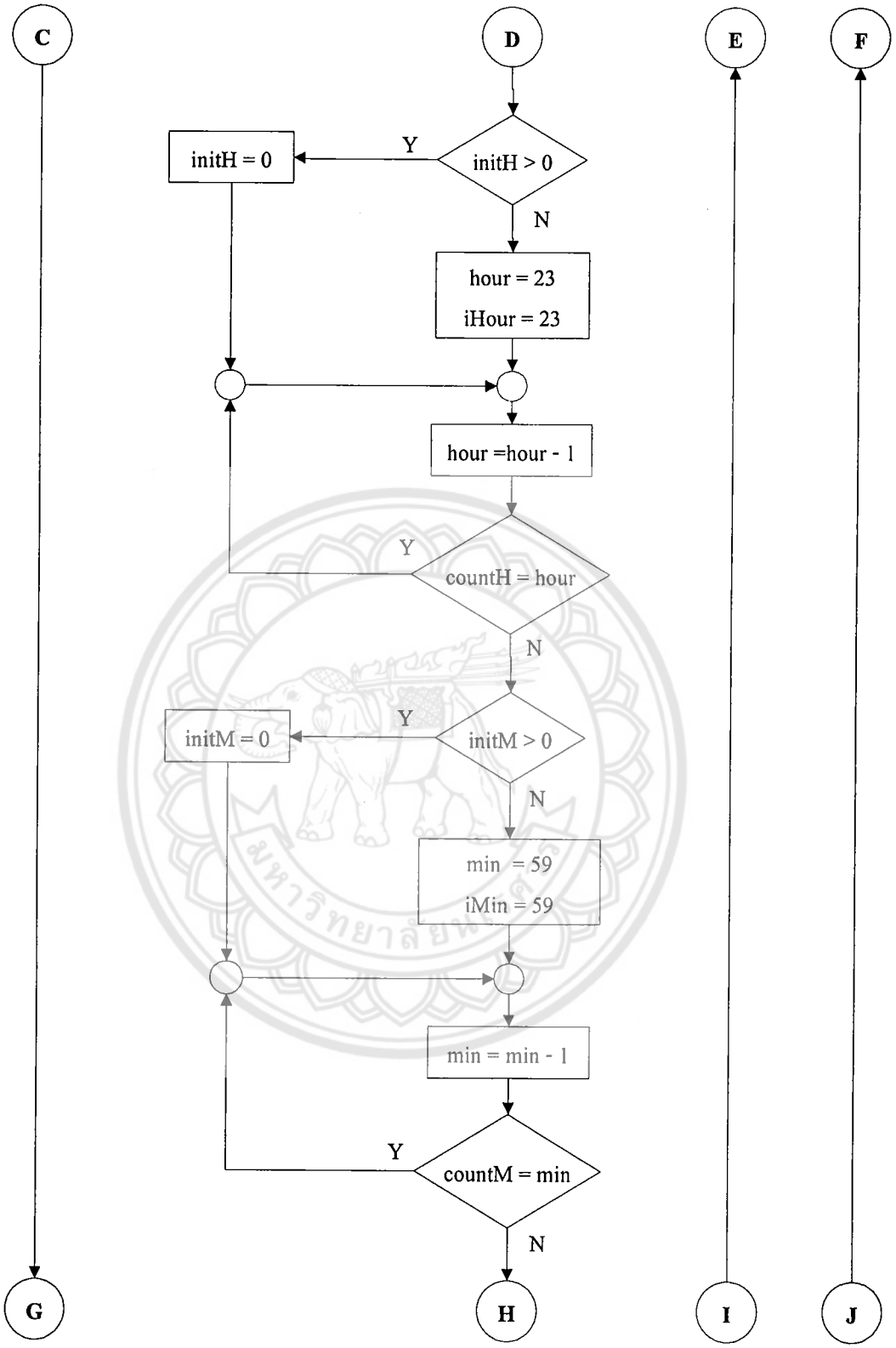
ในส่วนของการประมวลผลของไมโครคอนโทรลเลอร์นั้นมีขั้นตอนการทำงานดังรูปที่ 3.7



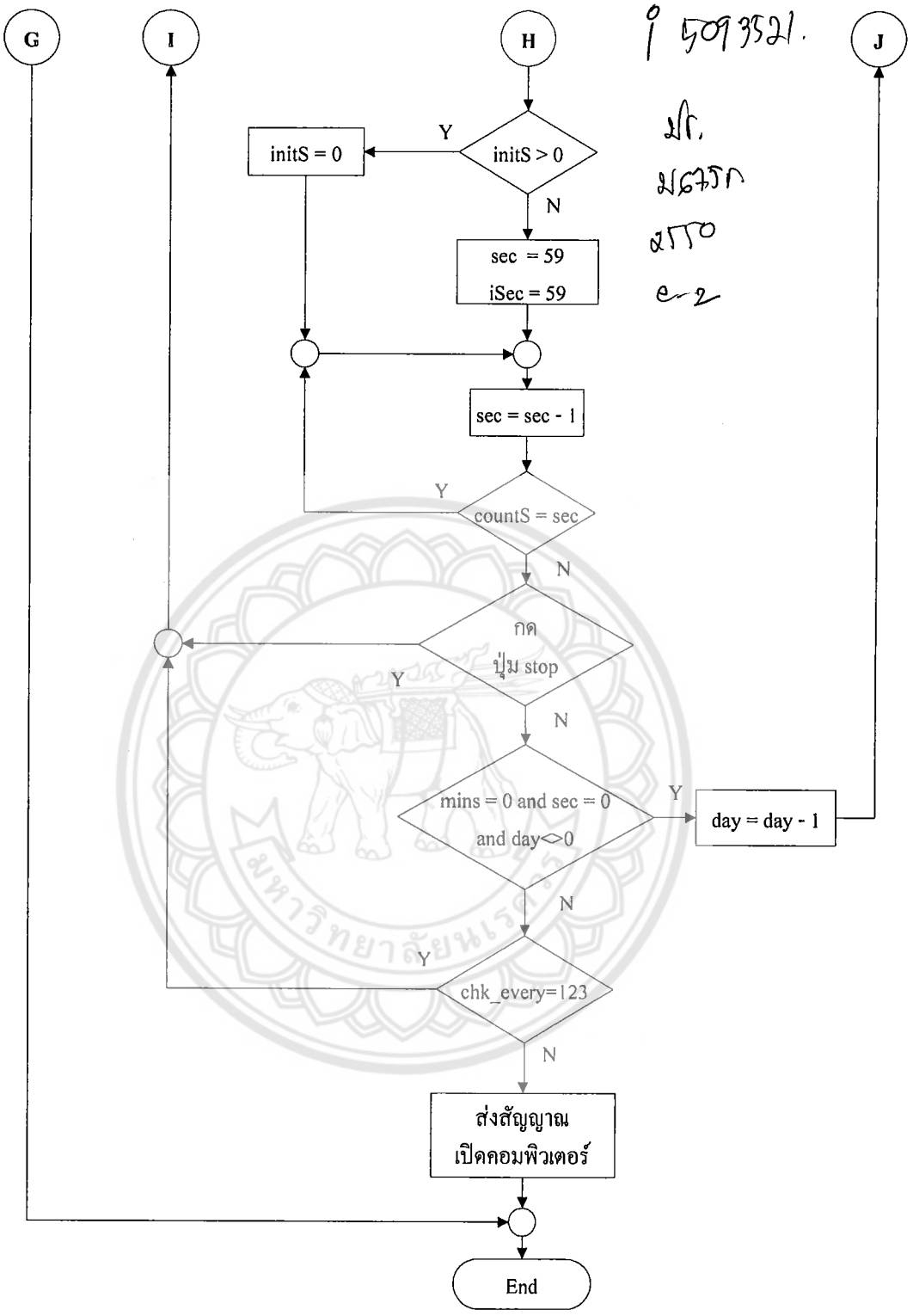
รูปที่ 3.7 แผนภาพแสดงการทำงานของโปรแกรมในไมโครคอนโทรลเลอร์



รูปที่ 3.7 (ต่อ) แผนภาพแสดงการทำงานของโปรแกรมในไมโครคอนโทรลเลอร์



รูปที่ 3.7 (ต่อ) แผนภาพแสดงการทำงานของโปรแกรมในไมโครคอนโทรลเลอร์

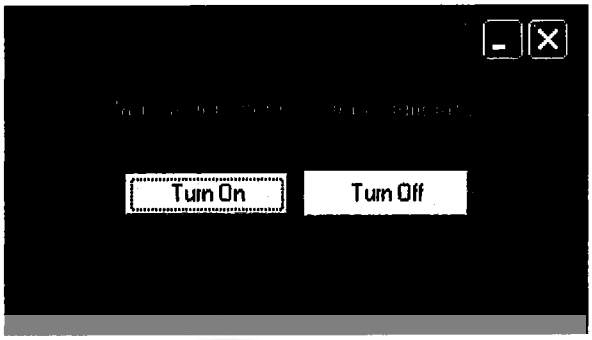


รูปที่ 3.7 (ต่อ) แผนภาพแสดงการทำงานของโปรแกรมในไมโครคอนโทรลเลอร์

3.2 การออกแบบโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

การออกแบบโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้นจะมีอยู่ 3 หน้าคือ

1. หน้าหลัก จะเป็นหน้าที่ใช้ในการเลือกว่าจะเปิดหรือปิดคอมพิวเตอร์ แต่ก็สามารถทำได้ทั้งสองอย่าง โดยได้มีการออกแบบไว้ดังรูปที่ 3.8

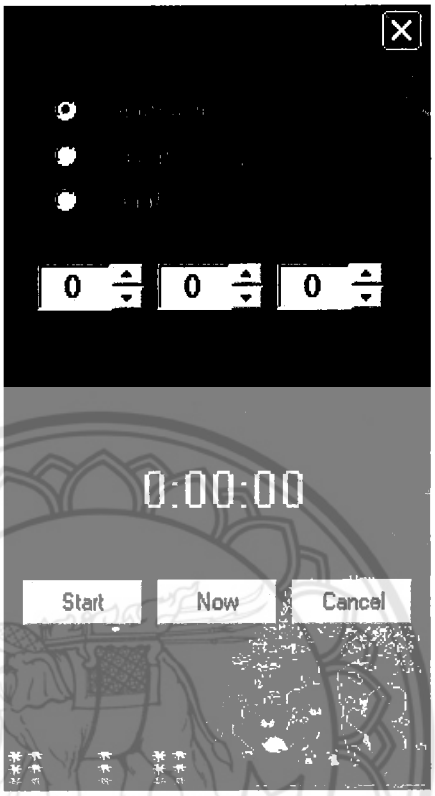


รูปที่ 3.8 แสดงหน้าหลักของโปรแกรม

2. หน้าในส่วนของการเปิดคอมพิวเตอร์ จะมีช่องให้กรอกวันและเวลาที่จะทำการเปิดคอมพิวเตอร์ ดังแสดงดังในรูปที่ 3.9

รูปที่ 3.9 หน้าต่างโปรแกรมในการเปิดคอมพิวเตอร์

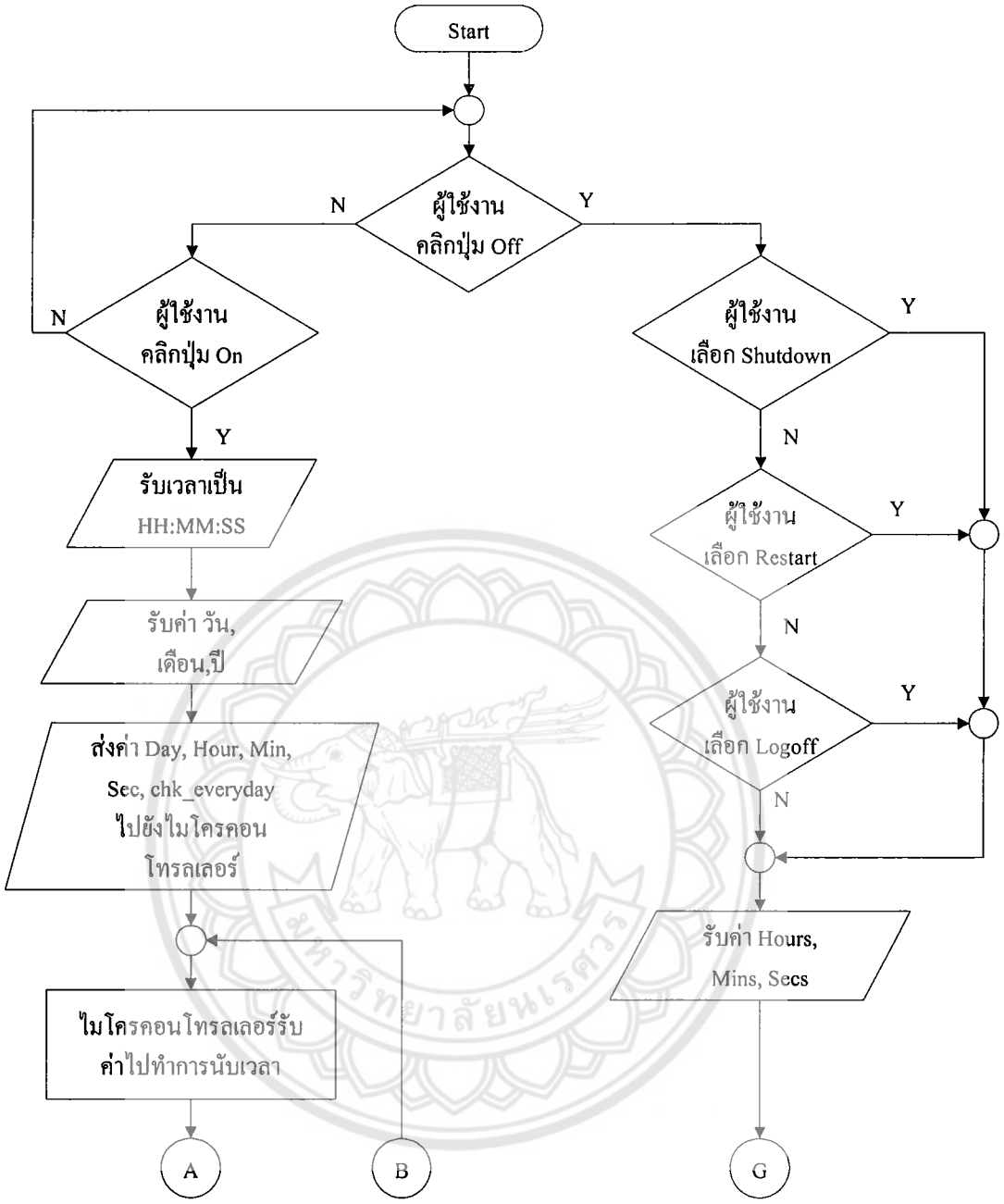
3. หน้าในส่วนของการปิดคอมพิวเตอร์ มีให้เลือกว่าจะทำการ Shutdown หรือ Restart หรือ Log Off จากนั้นก็ให้ตั้งเวลาว่าอีกนานเท่าไรถึงจะให้ปิดคอมพิวเตอร์ ดังแสดงดังในรูปที่ 3.10



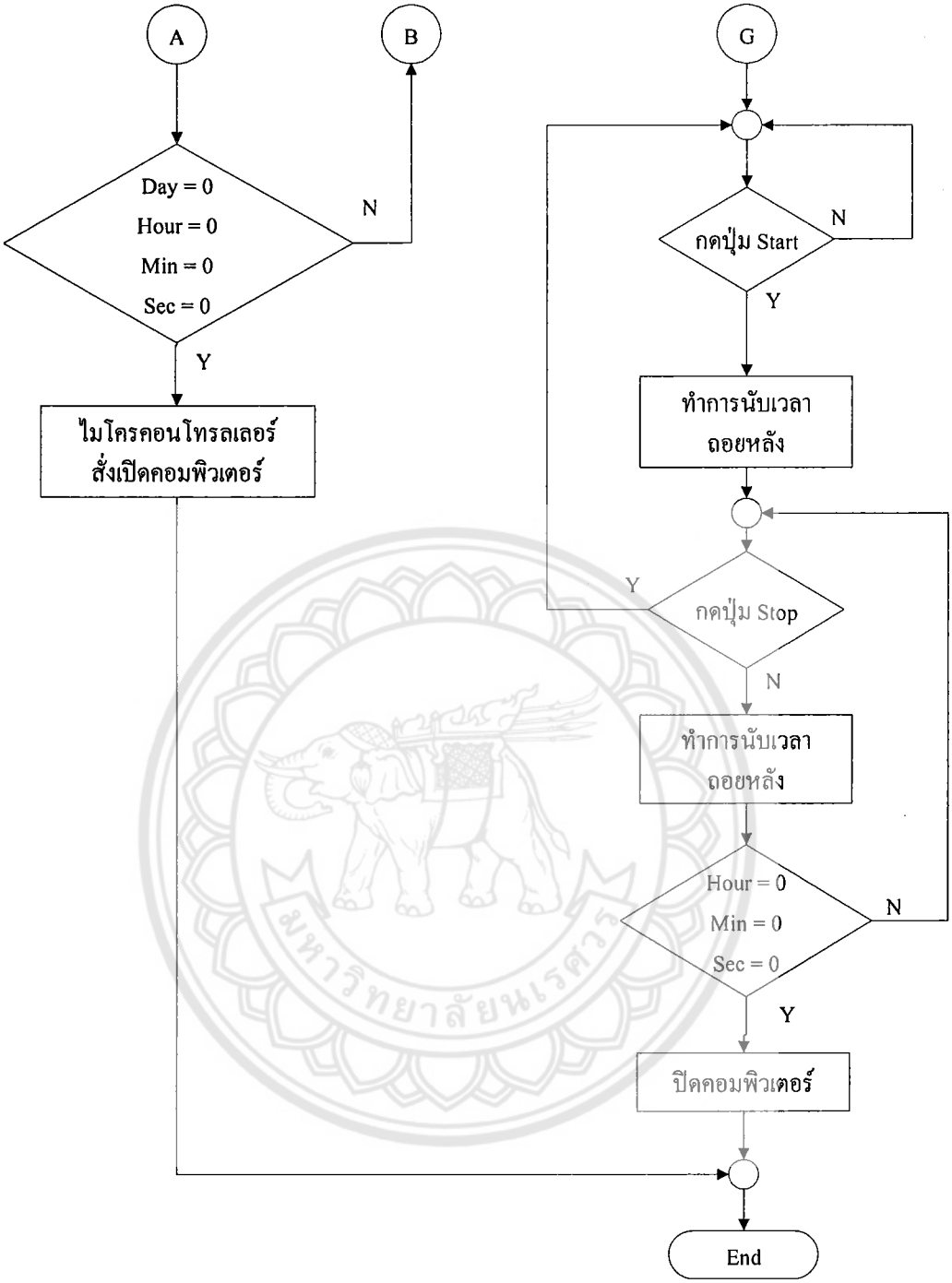
รูปที่ 3.10 หน้าต่างโปรแกรมในการปิดคอมพิวเตอร์

ในส่วนของการปิดคอมพิวเตอร์นั้นจะใช้ฟังก์ชันของระบบปฏิบัติการระดูลวินโดว์ที่ได้เตรียมไว้ให้ในไฟล์ User32.dll ซึ่งเป็นฟังก์ชันที่เกี่ยวข้องกับการจัดการผู้ใช้งาน โดยในไฟล์นั้นจะมีฟังก์ชันย่อยให้เราเลือกใช้งาน

การเขียนโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้นจะทำหน้าที่ในการติดต่อกับผู้ใช้งานนั้น เพื่อรับค่าต่างๆจากผู้ใช้ ซึ่งจะมีขั้นตอนการทำงานดังรูปที่ 3.11



รูปที่ 3.11 แผนภาพแสดงการทำงานของโปรแกรมในส่วนที่ติดต่อกับผู้ใช้งาน



รูปที่ 3.11 (ต่อ) แผนภาพแสดงการทำงานของโปรแกรมในส่วนที่ติดต่อกับผู้ใช้งาน

บทที่ 4
ผลการทดสอบและวิเคราะห์ผลการทดสอบการเปิดปิดคอมพิวเตอร์

ในการทดสอบประสิทธิภาพการเปิด – ปิดคอมพิวเตอร์ด้วยอุปกรณ์ที่สร้างขึ้นและชุดคำสั่งหรือโปรแกรมที่พัฒนาขึ้น จะทดสอบว่าสามารถเปิดและปิดคอมพิวเตอร์และตรงตามเวลาที่กำหนดได้หรือไม่

ในบทนี้จะแบ่งเป็น 3 ส่วน คือ ส่วนของผลการทดสอบการเปิดคอมพิวเตอร์, ส่วนของการเปิดคอมพิวเตอร์และส่วนของการวิเคราะห์ผลการทดสอบการเปิด – ปิดคอมพิวเตอร์ ซึ่งจะทำการทดสอบการเปิดและปิดคอมพิวเตอร์จำนวนอย่างละ 10 ครั้งโดยผลการทดสอบการเปิดคอมพิวเตอร์จะอยู่ในหัวข้อที่ 4.1 ส่วนของการเปิดคอมพิวเตอร์จะอยู่ในหัวข้อที่ 4.2 และการวิเคราะห์ผลการทดสอบอยู่ในหัวข้อที่ 4.3 ดังนี้

4.1 ผลการทดสอบการเปิดคอมพิวเตอร์

ผลการทดสอบการเปิดคอมพิวเตอร์จำนวน 10 ครั้ง แสดงดังตารางที่ 4.1

ตารางที่ 4.1 ผลการทดสอบการเปิดคอมพิวเตอร์

ครั้งที่	เวลาที่ตั้งให้เปิด	วันที่ตั้งให้เปิด	วันที่ตั้งให้ปิด	สถานะการทำงาน
1	8:00:00	25 April 2008	24 April 2008	คอมพิวเตอร์เปิด
2	9:00:00	26 April 2008	25 April 2008	คอมพิวเตอร์เปิด
3	10:00:00	27 April 2008	26 April 2008	คอมพิวเตอร์เปิด
4	11:00:00	28 April 2008	27 April 2008	คอมพิวเตอร์เปิด
5	12:00:00	29 April 2008	28 April 2008	คอมพิวเตอร์เปิด
6	8:00:00	30 April 2008	29 April 2008	คอมพิวเตอร์เปิด
7	14:00:00	1 May 2008	30 April 2008	คอมพิวเตอร์เปิด
8	18:00:00	2 May 2008	1 May 2008	คอมพิวเตอร์เปิด
9	20:00:00	3 May 2008	2 May 2008	คอมพิวเตอร์เปิด
10	10:45:00	4 May 2008	3 May 2008	คอมพิวเตอร์เปิด

จากตารางที่ 4.1 เราจะเห็นว่าคอมพิวเตอร์สามารถเปิดได้ตรงตามเวลาที่กำหนดโดยจะใช้เวลาได้ใกล้เคียงกับเวลาจริง คือจะมีการคลาดเคลื่อนกับเวลาจริง 3-4 วินาที

4.3 ผลการทดสอบการปิดคอมพิวเตอร์

ผลการทดสอบการปิดคอมพิวเตอร์ การ Restart และการ logoff จำนวน 10 ครั้ง แสดงดังตารางที่ 4.2

ตารางที่ 4.2 ผลการทดสอบการปิดคอมพิวเตอร์

ครั้งที่	กิจกรรม	เวลาที่ตั้ง (ชั่วโมง:นาที:วินาที)	ผลการทดสอบ	เวลาที่ใช้ทั้งหมด
1	shutdown	1:30:00	คอมพิวเตอร์ปิด	1:30:00
2	shutdown	3:00:00	คอมพิวเตอร์ปิด	3:00:00
3	shutdown	5:00:00	คอมพิวเตอร์ปิด	5:00:00
4	shutdown	10:05:00	คอมพิวเตอร์ปิด	10:05:00
5	shutdown	13:45:20	คอมพิวเตอร์ปิด	13:45:20
6	shutdown	15:10:00	คอมพิวเตอร์ปิด	15:10:00
7	restart	0:03:00	คอมพิวเตอร์ Restart	0:03:00
8	restart	1:00:00	คอมพิวเตอร์ Restart	1:00:00
9	logoff	2:35:00	คอมพิวเตอร์ Logoff	2:35:00
10	logoff	21:15:00	คอมพิวเตอร์ Logoff	21:15:00

จากตารางที่ 4.1 เราจะเห็นว่าคอมพิวเตอร์สามารถปิด หรือ Restart หรือ Logoff ได้ตรงตามที่เรากำหนด และใช้เวลาในการปิด หรือ Restart หรือ Logoff ได้ตรงกับที่กำหนดไว้

4.3 วิเคราะห์ผลการทดสอบ

จากผลการทำสอบระบบเปิดปิดคอมพิวเตอร์อัตโนมัติดังตารางที่ 4.1 และ 4.2 แล้วปรากฏว่าอุปกรณ์และชุดคำสั่งหรือโปรแกรมที่พัฒนาขึ้นนั้นสามารถเปิด - ปิดคอมพิวเตอร์ได้ตรงตามเวลาที่ต้องการเสมือนกับคนเปิดหรือปิดคอมพิวเตอร์จริง

บทที่ 5
บทสรุป

5.1 สรุปผล

จากการพัฒนาระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ ซึ่งได้มีการพัฒนาระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้น มีการสร้างอุปกรณ์รับส่งข้อมูลที่ใช้สำหรับรับข้อมูลวันและเวลาที่ผู้ใช้ได้ตั้งไว้สำหรับใช้ในการเปิดคอมพิวเตอร์อัตโนมัติและประมวลผล จากนั้นเมื่อถึงเวลาที่กำหนดไมโครคอนโทรลเลอร์ที่สร้างขึ้นนั้นก็จะเป็นเสมือนสวิตซ์ในการเปิดคอมพิวเตอร์อัตโนมัติ และพัฒนาโปรแกรมในการประมวลผลในส่วนของการปิดคอมพิวเตอร์ ได้ข้อสรุปว่าการทดลองใช้ระบบเปิดปิดคอมพิวเตอร์อัตโนมัติสามารถทำงานได้ตรงตามวัตถุประสงค์ที่ได้ตั้งไว้ สามารถเปิดและปิดคอมพิวเตอร์ได้ตามความต้องการของผู้ใช้งาน และสามารถนำไปใช้ในการควบคุมการเปิด-ปิดเครื่องคอมพิวเตอร์ที่ใช้เป็นเครื่องเซิร์ฟเวอร์ที่มีการใช้งานเป็นช่วงเวลาได้อย่างอัตโนมัติ

5.2 ข้อเสนอแนะ

- การติดต่อรับ-ส่งข้อมูลอาจจะใช้พอร์ตขนานเพื่อเพิ่มความเร็วในการส่งข้อมูล
- สามารถนำไปพัฒนาให้มีความสามารถในการทำงานได้มากขึ้น

เอกสารอ้างอิง

[1] กฤษดา ใจเย็น, ณัฏฐพล วงศ์สุนทรชัย, ชัยวัฒน์ ลิ้มพรจิตรวิไล. **เรียนรู้และใช้งาน PICBASIC PRO คอมไพเลอร์ เขียนโปรแกรมภาษาเบสิกควบคุมไมโครคอมโทรลเลอร์ PIC.** กรุงเทพมหานคร : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. 2542

[2] ฉันทวุฒิ พืชผล, พิชิต สันติกุลานนท์. **คู่มือเรียน Visual Basic 6.** กรุงเทพมหานคร : บริษัท โปรชั่น จำกัด. 2542

[3] ธีระศักดิ์ สุโชตินันท์, ประยุทธ อินแบน. **โปรแกรมเมอร์มือใหม่หัดเขียนโปรแกรม Microsoft Visual Basic 6.0.** กรุงเทพมหานคร : บริษัท ธรรมดาเพรส จำกัด. 2549

[4] สัจจะ จรัสรุ่งรวีวร. **คู่มือเขียนโปรแกรม Visual Basic 6 ฉบับผู้เริ่มต้น.** นนทบุรี : บริษัท ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์ จำกัด. 2548

[5] อภิชาติ ภู่พลับ. **เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic.** กรุงเทพมหานคร : บริษัท ด่านสถิทธาการพิมพ์ จำกัด. 2546

[6] อภิชาติ ภู่พลับ. **สนุกกับการประยุกต์ใช้ Visual Basic.** กรุงเทพมหานคร : บริษัท ด่านสุทธากา รพิมพ์ จำกัด. 2546

ภาคผนวก

ภาคผนวก ก

การติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

1. เตรียมความพร้อมก่อนติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

ก่อนจะเริ่มติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติให้กับคอมพิวเตอร์นั้น ควรเริ่มจากการสำรวจความพร้อมของคอมพิวเตอร์ และตรวจสอบสิ่งที่จำเป็นต้องใช้งานก่อน เพื่อให้การติดตั้งและการใช้งานเป็นไปอย่างราบรื่น

ตรวจความพร้อมของคอมพิวเตอร์และฮาร์ดแวร์

สำหรับฮาร์ดแวร์ที่ต้องใช้เพื่อการติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้นสรุปได้ตามตารางที่ 1

ตารางที่ 1 ส่วนประกอบฮาร์ดแวร์ที่ต้องใช้ในการติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

รายการ	คำอธิบาย
ซีพียู	ซีพียูตั้งแต่ Pentium 166 MHz ขึ้นไปหรือ AMD K6ขึ้นไป หรือซีพียู Cyrix รุ่นที่มีความเร็ว 120 MHz ขึ้นไป
ฮาร์ดดิสก์ (Hard Disk)	ควรมีเนื้อที่ว่างในฮาร์ดดิสก์สำหรับติดตั้งและองค์ประกอบอื่นๆ ดังนี้ - Standard Edition: 80 MB - Professional Edition: เต็มที่ 80 MB - Enterprise Edition: เต็มที่ 147 MB
แรม(RAM)	แรมที่สามารถใช้งานควรมีตั้งแต่ 32 MB ขึ้นไป
ซีดีรอม(CD-ROM)	ต้องมี โดยเป็นซีดีรอม 2X ขึ้นไป

ตรวจความพร้อมของซอฟท์แวร์

สำหรับซอฟท์แวร์ที่ใช้งานเพื่อการติดตั้งโปรแกรมระบบระบบเปิดปิดคอมพิวเตอร์อัตโนมัติได้ตามตารางที่ 2

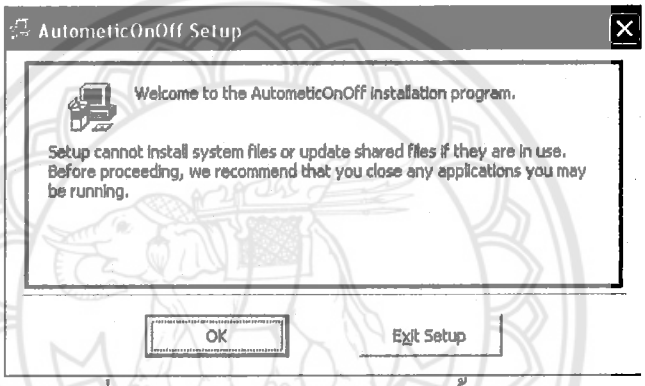
ตารางที่ 2 ส่วนประกอบซอฟท์แวร์ที่ต้องใช้ในการติดตั้งโปรแกรมระบบรายงานการตรวจสอบคุณภาพไดชาร์จด้วยคอมพิวเตอร์

รายการ	คำอธิบาย
ระบบปฏิบัติการ	ต้องเป็นระบบปฏิบัติการตระกูล Windows ของไมโครซอฟท์เท่านั้น ซึ่งรุ่นที่ใช้งาน ได้แก่Windows95/98/Me/2000(ทุกรุ่น)/XP(ทุกรุ่น)
โปรแกรมที่ผู้วิจัยทำขึ้น	เพื่อนำมาใช้ในการติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

2. วิธีการติดตั้งโปรแกรมระบบรายงานการตรวจสอบคุณภาพไดชาร์จด้วยคอมพิวเตอร์

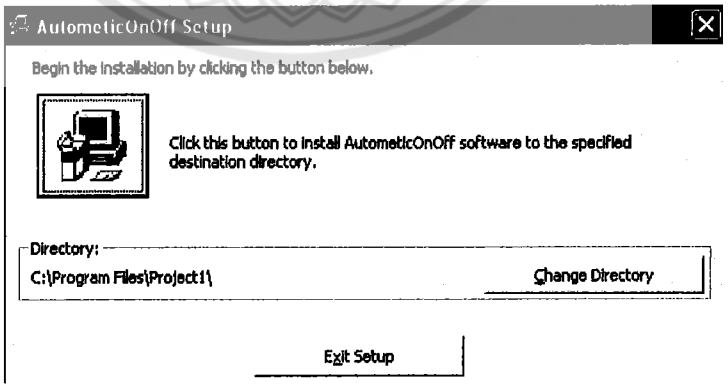
วิธีการติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้น สามารถทำได้ดังนี้

1. ให้ดับเบิ้ลคลิ๊กที่ setup.exe และจะขึ้นหน้าต่างดังรูปที่ 1



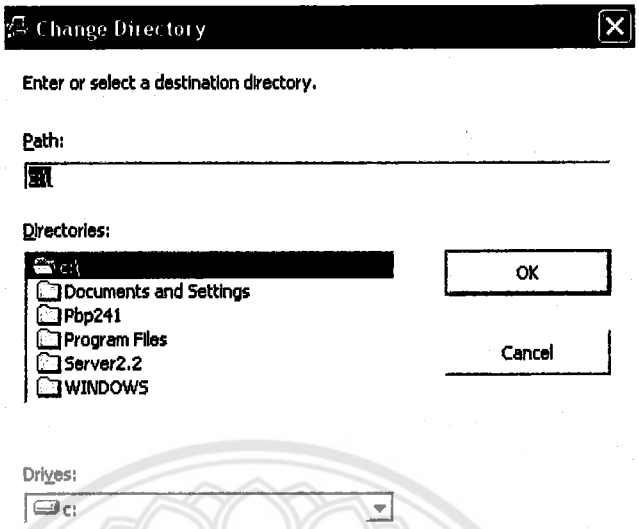
รูปที่ 1 แสดงหน้าต่างแรกของการติดตั้งโปรแกรม

2. ให้คลิ๊กปุ่ม OK จากนั้นจะขึ้นหน้าต่างดังรูปที่ 2



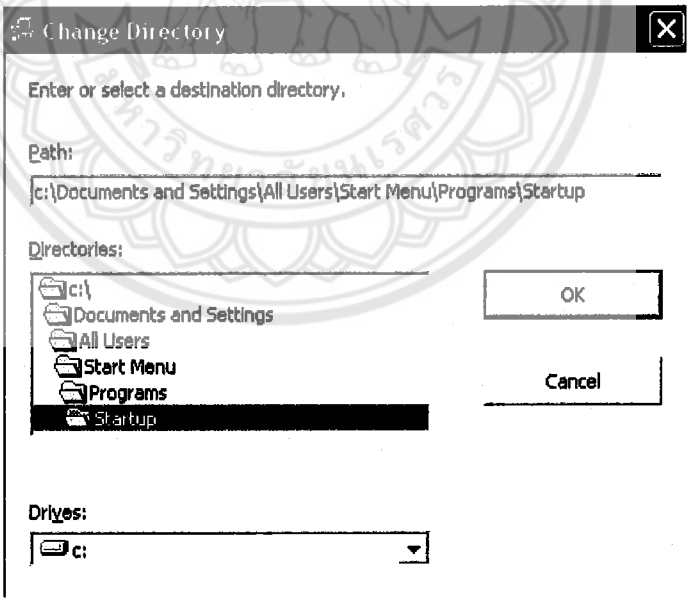
รูปที่ 1 แสดงหน้าต่างแรกของการติดตั้งโปรแกรม

3. คลิ๊ก Change Directory แล้วจะปรากฏหน้าต่างใหม่ขึ้นมาดังรูปที่ 3



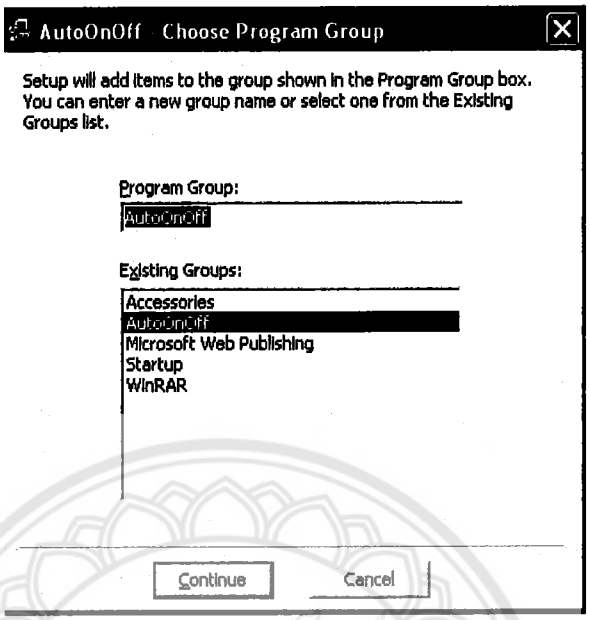
รูปที่ 3 แสดงหน้าต่างใหม่หลังจากคลิ๊กปุ่ม Change Directory

4. ให้เปลี่ยน Path เป็น C:\Documents and Settings\All Users\Start Menu\Programs\Startup ดังรูปที่ 4



รูปที่ 4 แสดงการเปลี่ยน Path ที่จะทำการติดตั้งโปรแกรม

5. ให้คลิ๊กปุ่ม OK และคลิ๊กที่  จะมีหน้าต่างใหม่ขึ้นมาดังรูปที่ 6 ให้กดปุ่ม Next



รูปที่ 6 แสดงหน้าต่างหลังจากกดปุ่มที่เป็นรูปคอมพิวเตอร์

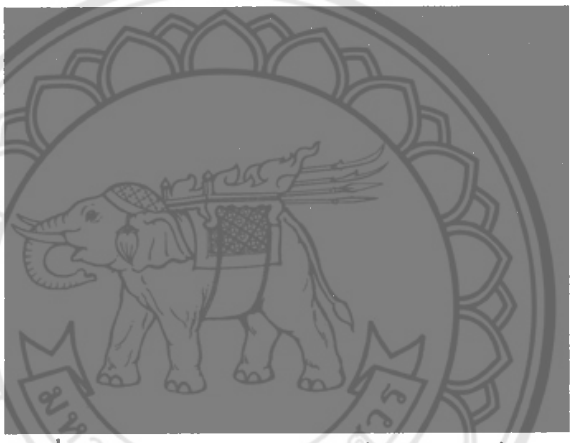
6. จากนั้นจะปรากฏหน้าต่างดังรูปที่ 6 ให้กด OK นั่นคือติดตั้งโปรแกรมเสร็จเรียบร้อยแล้ว และโปรแกรมก็จะอยู่ที่ Start >All Program > Startup > auto Shutdown

รูปที่ 6 แสดงหน้าต่างที่แสดงว่าติดตั้งโปรแกรมเสร็จแล้ว

3. วิธีการติดตั้งอุปกรณ์รับ - ส่งข้อมูล

หลังจากที่ทำการติดตั้งโปรแกรมระบบเปิดปิดคอมพิวเตอร์เสร็จแล้ว ต่อไปก็ให้ติดตั้งอุปกรณ์รับ – ส่งข้อมูล ซึ่งสามารถทำได้ดังนี้

1. ให้ตรวจสอบคู่มือเมนบอร์ดคอมพิวเตอร์ที่ใช้อยู่ว่าสายไฟเส้นไหนหรือสีอะไรที่เชื่อมต่อกับปุ่มเปิดคอมพิวเตอร์ หรือดูจากคู่มือว่าสวิตซ์เปิดคอมพิวเตอร์นั้นอยู่ตรงไหน
2. เมื่อหาสายไฟที่เป็นสวิตซ์ของเมนบอร์ดเจอแล้วให้ดึงออกมา แล้วไปเสียบไว้ในส่วนที่เป็นส่วนของการต่อสวิตซ์ โดยให้ส่วนที่เป็นไฟบวกต่อกับสายไฟสีขาว และส่วนที่เป็นไฟลบต่อกับกราวน์ที่เป็นสายไฟสีดำ
3. จากนั้นให้นำสายไฟ ดังรูปที่ 5 ไปต่อกับส่วนที่เป็นการต่อสวิตซ์ ซึ่งสายดีแดงเป็นไฟบวกและสานสีดำเป็นไฟลบ



รูปที่ 5 สายที่ใช้ในการต่อระหว่างเมนบอร์ดกับอุปกรณ์รับ - ส่งข้อมูล

4. ต่อไปให้นำสายอีกข้างหนึ่งต่อเข้าไปที่เมนบอร์ดตรงส่วนที่เราดึงออกมา
5. หลังจากนั้นให้ทำการต่อสาย serial ดังรูปที่ 6 เข้าที่พอร์ตอนุกรมของคอมพิวเตอร์ ส่วนข้างที่เป็น Terminal 4 pin นั้นให้มาเสียบไว้ที่อุปกรณ์รับส่ง ดังรูปที่ 7

รูปที่ 6 แสดงรูปสาย serial ที่ใช้ในการต่อระหว่างอุปกรณ์รับส่งข้อมูลและคอมพิวเตอร์

รูปที่ 7 การต่ออุปกรณ์รับส่งข้อมูล

6. เมื่อติดตั้งอุปกรณ์รับ - ส่งข้อมูลเสร็จแล้วก็สามารถใช้งานโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติได้ทันที

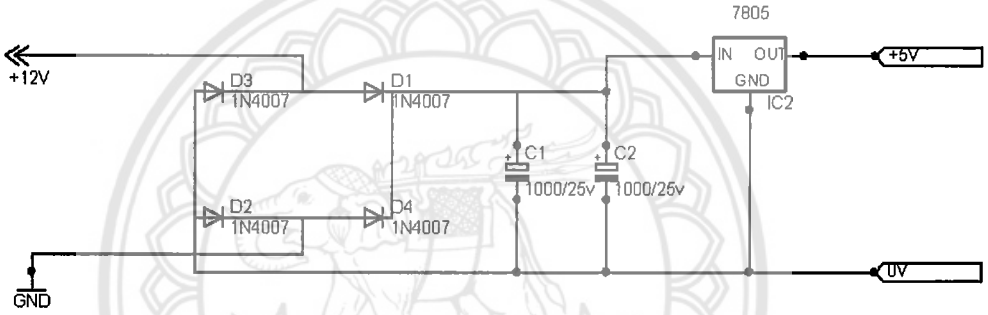
ภาคผนวก ข

การสร้างอุปกรณ์รับ-ส่งข้อมูล

อุปกรณ์อิเลคทรอนิคส์ต่างๆ ที่ใช้ในการสร้างอุปกรณ์รับ-ส่งข้อมูลมีดังนี้

ตารางที่ 1 อุปกรณ์สำหรับใช้สร้างวงจรจ่ายไฟในอุปกรณ์รับ-ส่งข้อมูล

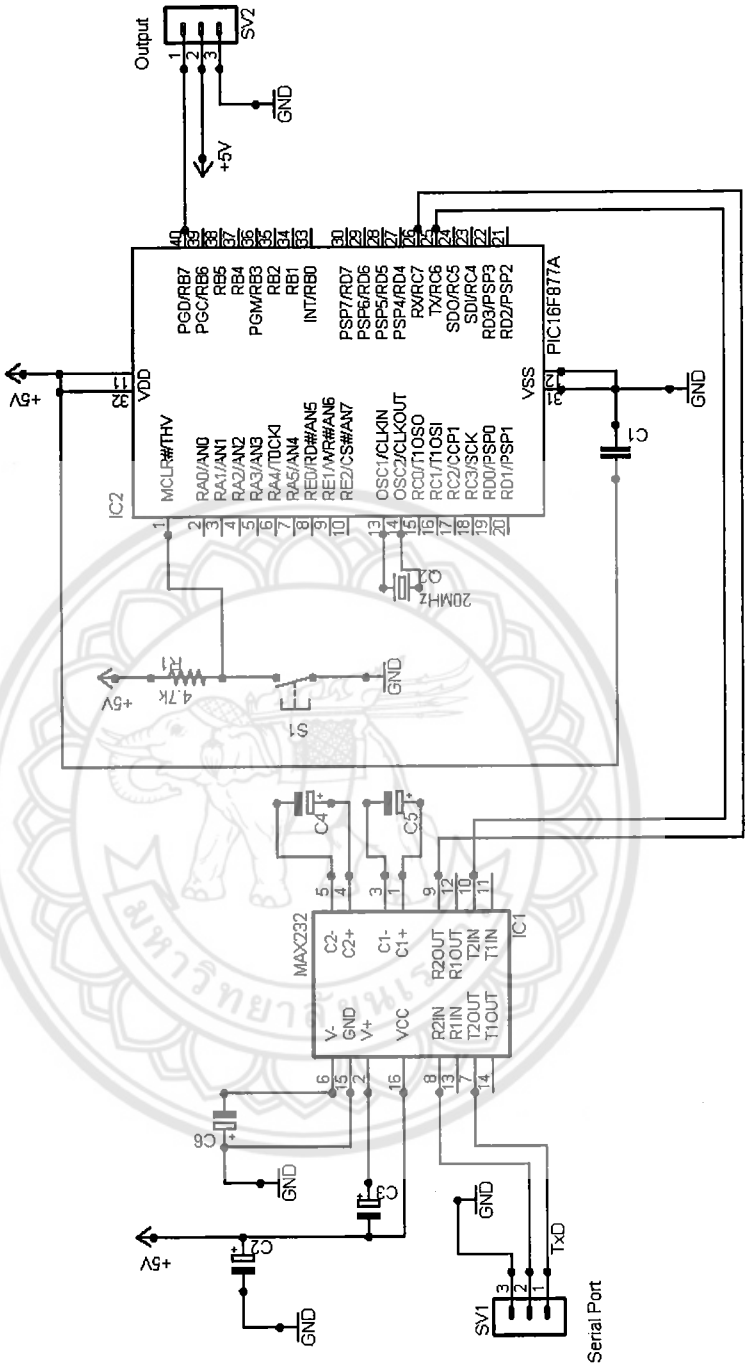
ลำดับที่	ชื่ออ้างอิงในวงจร	รายการ	จำนวนทั้งหมดที่ใช้(ตัว)
1	IC2	7805	1
2	D1-D4	1N4007	4
3	C1,C2	1000 uF/25V	2



รูปที่ 1 วงจรจ่ายไฟในอุปกรณ์รับ-ส่งข้อมูล

ตารางที่ 2 อุปกรณ์สำหรับใช้สร้างวงจรการทำงานในอุปกรณ์รับ-ส่งข้อมูล

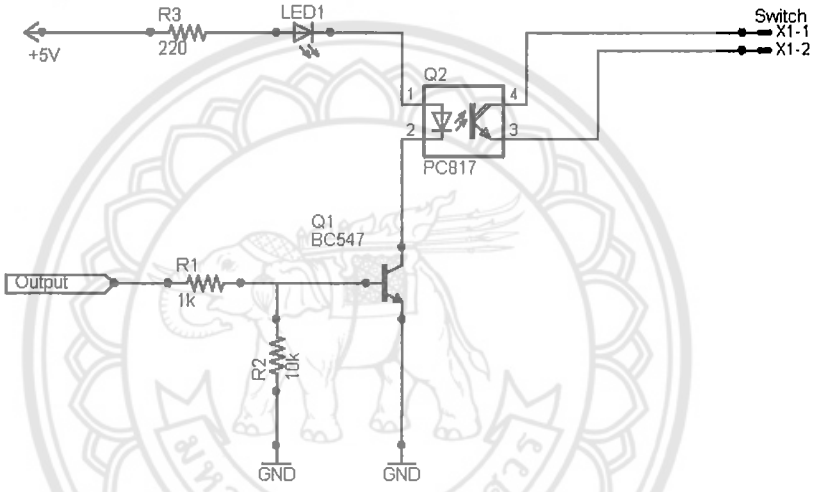
ลำดับที่	ชื่ออ้างอิงในวงจร	รายการ	จำนวนทั้งหมดที่ใช้(ตัว)
1	IC1	Max232	1
2	IC1	PIC 16F877A	1
3	C1	0.1uF/50V	1
4	C2-C6	10uF/50V	5
5	SV1	Terminal 3 pin	1
6	SV2	Terminal 3 pin	1
7	Q2	Crystal 20 MHz.	1
8	R1	4.7 K ohm	1
9	S1		1



รูปที่ 2 วงจรการทำงานของอุปกรณ์รับ-ส่งข้อมูล

ตารางที่ 2 อุปกรณ์สำหรับใช้สร้างวงจรการทำงานของวงจรเอาท์พุต

ลำดับที่	ชื่ออ้างอิงในวงจร	รายการ	จำนวนทั้งหมดที่ใช้(ตัว)
1	R1	10 K ohm	1
2	R2	1 K ohm	1
3	R3	220 ohm	1
4	Q1	BC547	1
5	Q2	PC817	1
6	Switch	Terminal 2 pin	1



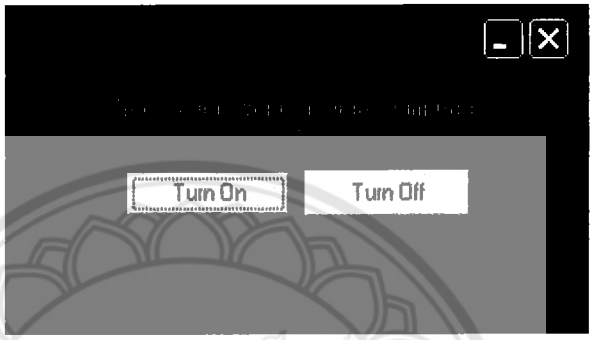
รูปที่ 3 วงจรการทำงานของวงจรเอาท์พุต

ภาคผนวก ค

วิธีการใช้โปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

วิธีการใช้โปรแกรมระบบเปิดขปิดคอมพิวเตอร์อัตโนมัตินั้นสามารถทำได้ดังนี้

1. ให้เปิดโปรแกรม โดยเข้าไปที่ Start > All Program > Startup > AutoOnOff จากนั้นจะปรากฏหน้าต่างดังรูปที่ 1



รูปที่ 1 หน้าต่างหลักของโปรแกรม

2. ให้เลือกว่าต้องการจะตั้งเวลาเปิดหรือปิด หากต้องการเปิดให้คลิ๊กที่ปุ่ม "Turn On" หรือถ้าต้องการปิดให้คลิ๊กที่ปุ่ม "Turn Off" และทำตามขั้นตอนดังนี้

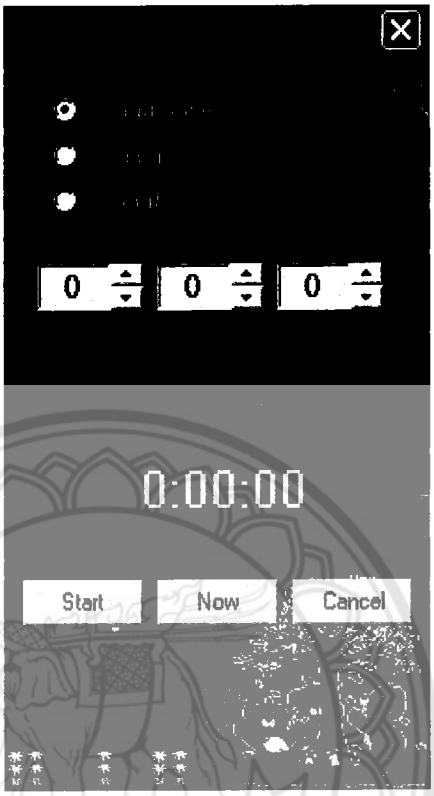
กรณีที่เลือกเปิดคอมพิวเตอร์

รูปที่ 2 หน้าต่างโปรแกรมในการเปิดคอมพิวเตอร์

1. ให้กรอกเวลาและวันที่เราต้องการเปิด หากต้องการเปิดคอมพิวเตอร์ทุกวันและเวลาเดียวกับที่ตั้งไว้ให้เช็คที่ Everyday แล้วกดปุ่ม "Start" จากนั้นก็สามารถปิดคอมพิวเตอร์ได้เลย และเมื่อถึงเวลาที่ตั้งไว้คอมพิวเตอร์ก็จะเปิดเองโดยอัตโนมัติ

2. หากต้องการหยุดหรือเปลี่ยนเวลาในการเปิดคอมพิวเตอร์ให้เปิดโปรแกรมแล้วกดปุ่ม "Stop" จากนั้นให้ปิดหน้าต่างนั้นแล้วเปิดใหม่และทำการตั้งเวลาใหม่ได้เลย

กรณีที่เลือกเปิดคอมพิวเตอร์



รูปที่ 3 หน้าต่างโปรแกรมในการปิดคอมพิวเตอร์

1. ให้เลือกว่าต้องการ Shutdown หรือ Restart หรือ Log Off

2. กรอกเวลาที่ต้องการจะปิด เช่น ต้องการจะปิดในอีก 1 ชั่วโมง 30 นาที ก็ให้กรอกตรงช่องของของ Hour เป็น 1, ช่องของ Min เป็น 30 และช่องของ Sec เป็น 0

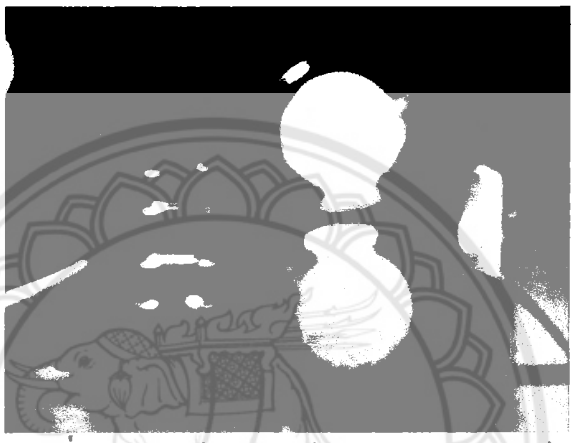
3. จากนั้นให้คลิ๊กที่ปุ่ม "Start" ปล่อยให้โปรแกรมรันโดยไม่ต้องปิด และเมื่อถึงเวลาที่ตั้งไว้คอมพิวเตอร์ก็จะเปิดเองโดยอัตโนมัติ

ภาคผนวก จ

อุปกรณ์รับ- ส่งข้อมูลสำหรับระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

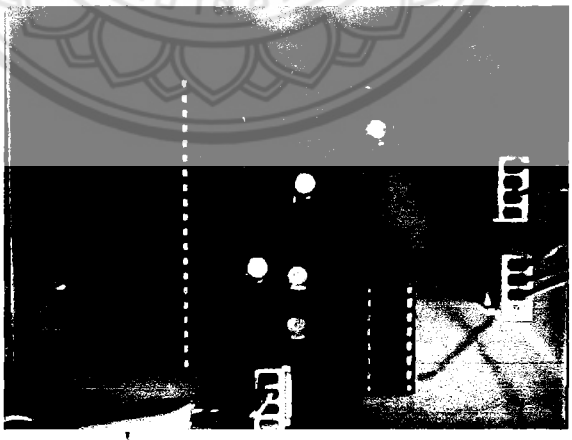
จากที่ได้มีการพัฒนาโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัตินั้น ได้มีการสร้างไมโครคอนโทรลเลอร์ที่ใช้ในการนับเวลาและส่งสัญญาณเปิดคอมพิวเตอร์ โดยจะแบ่งออกเป็น 4 ส่วนด้วยกัน คือ

1. ส่วนที่เป็นแหล่งจ่ายไฟ ซึ่งแสดงได้ดังรูปที่ 1



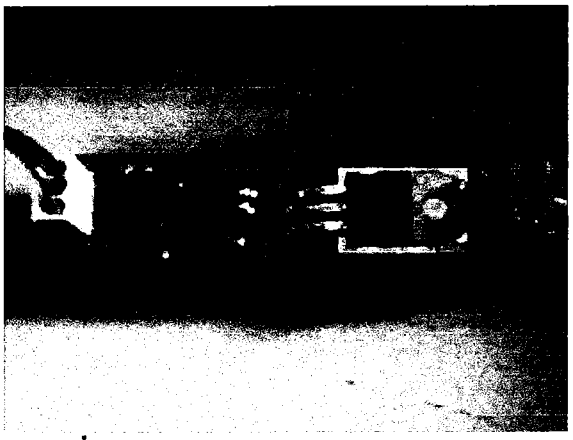
รูปที่ 1 แหล่งจ่ายไฟสำหรับไมโครคอนโทรลเลอร์

2. ส่วนของวงจรการทำงานของ PIC16F877 ซึ่งแสดงได้ดังรูปที่ 2



รูปที่ 2วงจรการทำงานของ PIC16F877

3. ส่วนของวงจรเอาต์พุต ซึ่งแสดงได้ดังรูปที่ 3

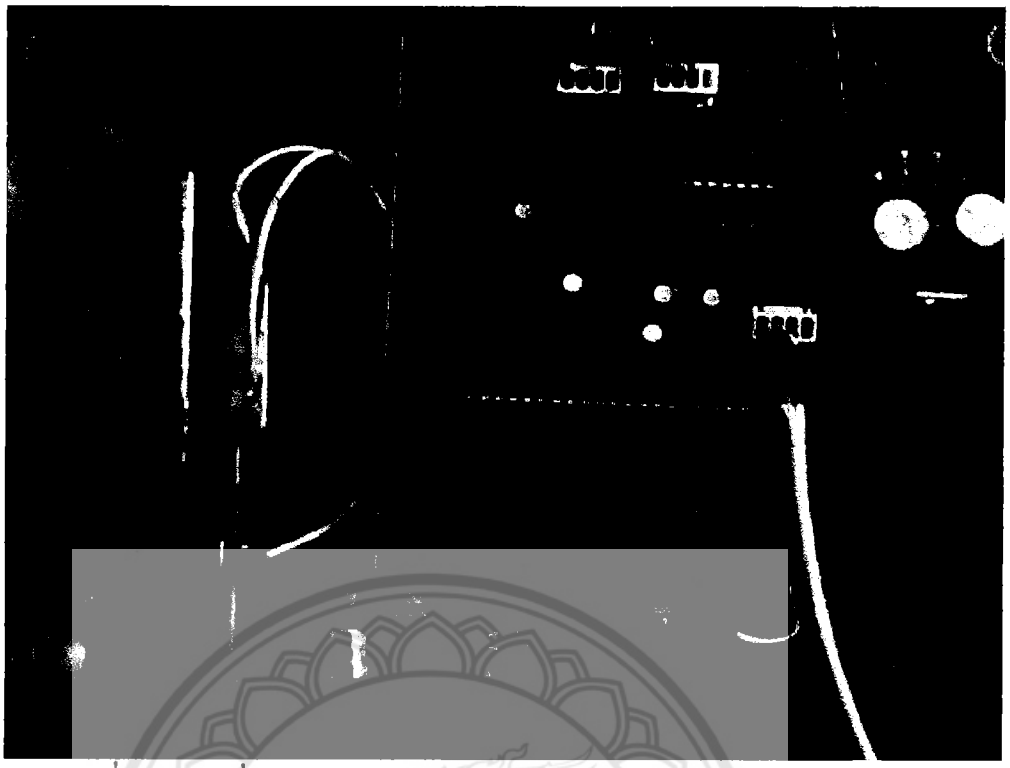


รูปที่ 3 วงจรเอาต์พุตของไมโครคอนโทรลเลอร์

4. ส่วนที่ใช้ในการต่อสวิตซ์ ซึ่งแสดงได้ดังรูปที่ 4

รูปที่ 4 ส่วนที่ใช้ในการต่อสวิตซ์

จากส่วนประกอบ 4 ส่วนดังข้างต้นนั้นเมื่อนำมาต่อรวมกันก็จะได้อุปกรณ์ที่ใช้ในการรับ- ส่งข้อมูลสำหรับระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ ซึ่งสามารถแสดงได้ดังรูปที่ 5



รูปที่ 5 อุปกรณ์ที่ใช้ในการรับ- ส่งข้อมูลสำหรับระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

ภาคผนวก ฉ

Source Code

1. Source Code ในส่วนของโปรแกรมระบบเปิดปิดคอมพิวเตอร์อัตโนมัติ

1.1 Code Form1 (Main Form)

```
Dim chkOnOff As Integer
Private Sub cmdOff_Click()
   chkOnOff = 1
   Form3.Show
   'Form1.Hide
End Sub

Private Sub cmdOn_Click()
   chkOnOff = 2
   Form2.Show
   'Form1.Hide
End Sub

Private Sub Form_Load()
   'Form1.Hide
   chkOnOff = 0
   'We set up Picture1 to accept callback data
   'in it's MouseMove procedure.
   NotifyIcon.cbSize = Len(NotifyIcon)
   NotifyIcon.hWnd = Picture1.hWnd
   NotifyIcon.uID = 1&
   NotifyIcon.uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
   NotifyIcon.uCallbackMessage = WM_MOUSEMOVE
   'Now, we set up the icon and tool tip message
   NotifyIcon.hIcon = Picture1.Picture
   NotifyIcon.szTip = "Autometic Shutdown" & Chr$(0)
```

```
    'Lastly, we add the icon
    Shell_NotifyIcon NIM_ADD, NotifyIcon
End Sub

Private Sub picExitMain_Click()
    'Remove the icon from the taskbar
    Shell_NotifyIcon NIM_DELETE, NotifyIcon

    'End the program
    Unload Me
    Unload Form2
    Unload Form3
End Sub

Private Sub picMiniMain_Click()
    Form1.Hide
    Form2.Hide
    Form3.Hide
    'We set up Picture1 to accept callback data
    'in it's MouseMove procedure.
    NotifyIcon.cbSize = Len(NotifyIcon)
    NotifyIcon.hWnd = Picture1.hWnd
    NotifyIcon.uID = 1&
    NotifyIcon.uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
    NotifyIcon.uCallbackMessage = WM_MOUSEMOVE

    'Now, we set up the icon and tool tip message
    NotifyIcon.hIcon = Picture1.Picture
    NotifyIcon.szTip = "Autometic Shutdown" & Chr$(0)
```

```
    'Lastly, we add the icon
    Shell_NotifyIcon NIM_ADD, NotifyIcon
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Hex(X) = "1E0F" Then
        Shell_NotifyIcon NIM_DELETE, NotifyIcon
        If chkOnOff = 0 Then
            Form1.Show
        ElseIf chkOnOff = 1 Then
            Form1.Show
            Form3.Show
        ElseIf chkOnOff = 2 Then
            Form1.Show
            Form2.Show
        Else
            Form1.Show
            Form2.Show
            Form3.Show
        End If
    End If
End Sub
```

1.2 Code Form2 (Turn On Form)

```
Option Explicit

Private Declare Function ExitWindowsEx Lib "user32" _
  (ByVal dwOptions As Long, _
  ByVal dwReserved As Long) As Long
```

```
Private Const EWX_LOGOFF As Long = &H0
Private Const EWX_SHUTDOWN As Long = &H1
Private Const EWX_REBOOT As Long = &H2
Private Const EWX_POWEROFF As Long = &H8

Dim timeNow As Variant
Dim timeDif As Integer
Dim hourDif As Integer
Dim minDif As Integer
Dim secDif As Integer

Dim counterDate As Integer        ' date of cboDate
Dim counterMonth As Integer       ' month of cboMonth
Dim counterYear As Integer        ' year of cboYear
Dim dayOut As Integer
Dim indexYear  As Long
Dim stopInput As Integer
Dim lastDay As Integer
Dim indexMonth As Integer
Dim indexDay As Integer
Dim dayNow As Integer

Dim replyStatus As Integer
Dim c_reply As Integer
Dim i As Integer

Private Sub cmdReset_Click()
    txtTime.Text = "HH:MM:SS"
    cboDate.Text = "Date"
    cboMonth.Text = "Month"
    cboYear.Text = "Year"
```

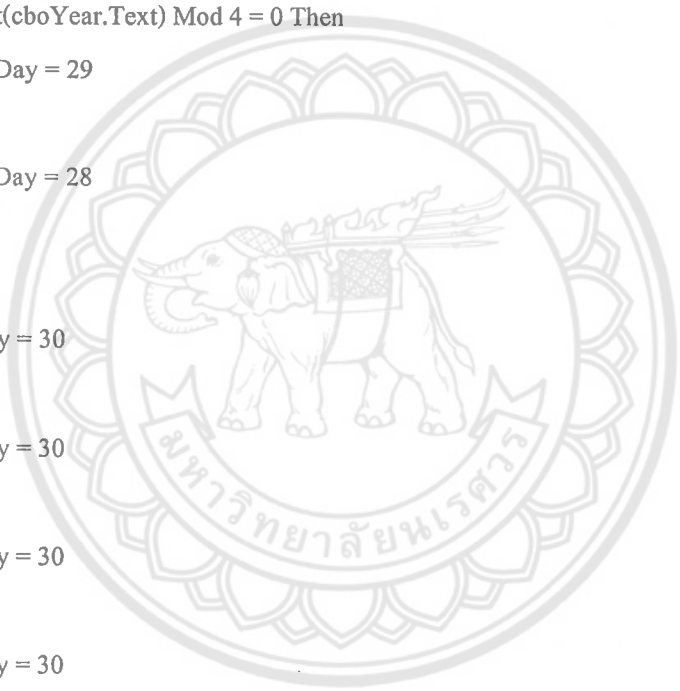
```
    'lblShowTime.Caption = ""
    lblDifTime.Caption = ""
    Label5.Caption = ""
End Sub

Private Sub cmdStop_Click()
  mscSender.Output = Chr(&H5C)   'send "\"
  Label4.Caption = ""
End Sub

Private Sub Form_Load()
  chkEveryday.Value = 0
  mscSender.CommPort = 1              ' receive data in port com1
  mscSender.Settings = "9600,n,8,1"
  mscSender.PortOpen = True
  tmrTurnOn.Interval = 100
  tmrTurnOn.Enabled = True
  mscSender.Output = Chr(&H5F)   ' send "_"

  ' Add day to cboDate 1-31
  For counterDate = 1 To 31
    cboDate.AddItem (counterDate)
  Next counterDate

  'Add name of month to cboMonth
  For counterMonth = 1 To 12
    Select Case counterMonth
      Case 1
        cboMonth.AddItem ("January")
      Case 2
        cboMonth.AddItem ("February")
```

```
        Case 3
            cboMonth.AddItem ("March")
        Case 4
            cboMonth.AddItem ("April")
        Case 5
            cboMonth.AddItem ("May")
        Case 6
            cboMonth.AddItem ("June")
        Case 7
            cboMonth.AddItem ("July")
        Case 8
            cboMonth.AddItem ("August")
        Case 9
            cboMonth.AddItem ("September")
        Case 10
            cboMonth.AddItem ("October")
        Case 11
            cboMonth.AddItem ("November")
        Case 12
            cboMonth.AddItem ("December")
        End Select
Next counterMonth

' Add year cboYear 2006-2030
For counterYear = 2006 To 2030
    cboYear.AddItem (counterYear)
Next counterYear

If cboYear.Text = "Year" Then
    indexYear = 100
Else
```

```
    indexYear = CLng(Right(cboYear.Text, 4)) + 543
  End If

End Sub

Private Sub cmdStartStop_Click()

  If replyStatus <> 0 Then
    If txtTime.Text = "" Or cboDate.Text = "Date" Or cboMonth.Text = "Month" Or cboYear.Text =
"Year" Then
      MsgBox ("You must enter Time, Date, Month and Year"), vbOKOnly + vbExclamation,
"Error"
      Exit Sub
    Else
      If Len(txtTime.Text) <> 8 Then
        MsgBox ("Error, Please enter time length only 8 character"), vbOKOnly + vbExclamation,
"Error"
        txtTime.Text = ""
        txtTime.Text = "hh:mm:ss"
        Exit Sub
      Else

        If txtTime.Text = "24:00:00" Or CInt(Left(txtTime.Text, 2)) > 23 Or
CInt(Mid(txtTime.Text, 4, 2)) > 59 Or CInt(Right(txtTime.Text, 2)) > 59 Then
          MsgBox ("Error, Please enter time is correct agian"), vbOKOnly + vbExclamation,
"Error"
          txtTime.Text = ""
          txtTime.Text = "hh:mm:ss"
          Exit Sub
        Else
          Call calDay
```

```
            End If
          End If
        End If
      Else
        MsgBox ("Micro controller is not ready."), vbOKOnly + vbExclamation, "Error"
        Exit Sub
      End If
    End Sub

    Sub calDay()
      indexYear = CLng(Right(cboYear.Text, 4))
      indexDay = CInt(cboDate.Text)
      Select Case cboMonth.Text   ' find index of month in each year
        Case "January"
          indexMonth = 1
        Case "February"
          indexMonth = 2
        Case "March"
          indexMonth = 3
        Case "April"
          indexMonth = 4
        Case "May"
          indexMonth = 5
        Case "June"
          indexMonth = 6
        Case "July"
          indexMonth = 7
        Case "August"
          indexMonth = 8
        Case "September"
          indexMonth = 9
```

```
    Case "October"
        indexMonth = 10
    Case "November"
        indexMonth = 11
    Case "December"
        indexMonth = 12
End Select

Select Case (indexMonth - 1)        ' find the end of day in last month
    Case 2
        If CInt(cboYear.Text) Mod 4 = 0 Then
            lastDay = 29
        Else
            lastDay = 28
        End If
    Case 4
        lastDay = 30
    Case 6
        lastDay = 30
    Case 9
        lastDay = 30
    Case 11
        lastDay = 30
    Case Else
        lastDay = 31
End Select
```



```
    ' check year
    If Int(cboYear.Text) = Int(Year(Date)) Or (Int(cboYear.Text) - Int(Year(Date))) < 1 Then ' Are
year present?
        Call chk_Month
    Else
        MsgBox ("Please enter year again"), vbOKOnly + vbExclamation, "Error"
        cboYear.Text = "Year"
        Exit Sub
    End If
End Sub

Sub chk_Month()
    ' case month input equal or more than month now
    If indexMonth = Int(Month(Date)) And Int(cboYear.Text) = Int(Year(Date)) Then
        Call chk_Day
    Else
        ' case month input more than month now for 2 month
        If indexMonth - Int(Month(Date)) = 1 Then
            Call chk_Day
        Else
            ' case month input less than month now and year input more than year now is 1 year
            If indexMonth < Month(Date) And Int((cboYear.Text) - Int(Year(Date)) = 1) Then
                Call chk_Day
            Else    ' any case above is false
                MsgBox ("Please enter month again"), vbOKOnly + vbExclamation, "Error"
                cboMonth.Text = "Month"
                Exit Sub
            End If
        End If
    End If
End Sub
```

```
Sub chk_Day()
  If Int(cboDate.Text) >= Int(Day(Date)) And indexMonth = Int(Month(Date)) Then
    dayOut = indexDay - CInt(Day(Date))
    'lblDifTime.Caption = TimeSerial(hourDif, minDif, secDif) ' show time to remain
    Call chk_DayMax
  Else
    If Int(cboDate.Text) < Int(Day(Date)) And (indexMonth - Int(Month(Date)) = 1) Then
      dayOut = lastDay - CInt(Day(Date)) + indexDay
      'lblDifTime.Caption = TimeSerial(hourDif, minDif, secDif)   ' show time to remain
      Call chk_DayMax
    Else
      If Int(cboDate.Text) > Int(Day(Date)) And (indexMonth - Int(Month(Date)) = 1) Then
        dayOut = lastDay - CInt(Day(Date)) + indexDay
        'lblDifTime.Caption = TimeSerial(hourDif, minDif, secDif)   ' show time to remain
        Call chk_DayMax
      Else
        MsgBox ("Please enter date again"), vbOKOnly + vbExclamation, "Error"
        cboDate.Text = "Date"
        Exit Sub
      End If
    End If
  End If
End Sub

Sub chk_DayMax()
  If dayOut > 7 Then
    MsgBox ("Please enter date start again, because maximun is 7 day"), vbOKOnly +
vbExclamation, "Error"
    cboDate.Text = "Date"
    cboMonth.Text = "Month"
    Exit Sub
```

```
    Else
      Call chk_time
    End If
End Sub

Sub chk_time()
    txtTime.Text = Format(txtTime, "hh:mm:ss")
    'lblShowTime.Caption = txtTime
    timeNow = Format(Time, "Long Time")
    If dayOut = 0 Then
      i = 1
    Else
      i = 2
    End If

    Select Case i
      Case 1
       ' ****** check time Input more than time now *********
        If CInt(Hour(timeNow)) > CInt(Hour(txtTime.Text)) Then
          MsgBox ("Please enter time again"), vbOKOnly + vbExclamation, "Error"
          txtTime.Text = "hh:mm:ss"
          Exit Sub
        Else
          If CInt(Hour(timeNow)) = CInt(Hour(txtTime.Text)) Then
            If (CInt(Minute(timeNow)) = CInt(Minute(txtTime.Text))) And
CInt(Second(timeNow)) >= CInt(Second(txtTime.Text)) Then
              MsgBox ("Please enter time again"), vbOKOnly + vbExclamation, "Error"
              txtTime.Text = "hh:mm:ss"
              Exit Sub
            Else
              hourDif = Abs(Hour(txtTime.Text) - Hour(timeNow))
```

```
            minDif = Abs(Minute(txtTime.Text) - Minute(timeNow))
            secDif = Abs(Second(txtTime.Text) - Second(timeNow))
          End If
        Else
          hourDif = Abs(Hour(txtTime.Text) - Hour(timeNow)) - 1
          minDif = Abs(Minute(txtTime.Text) - Minute(timeNow) + 59)
          secDif = Abs(Second(txtTime.Text) - Second(timeNow) + 59)
        End If
      End If
    Case 2
      If CInt(Hour(timeNow)) > CInt(Hour(txtTime.Text)) Then
        hourDif = Abs(23 - Hour(timeNow) + Hour(txtTime.Text))
        minDif = Abs(Minute(txtTime.Text) + 59 - Minute(timeNow))
        secDif = Abs(Second(txtTime.Text) + 59 - Second(timeNow))
      Else
        If CInt(Hour(timeNow)) < CInt(Hour(txtTime.Text)) Then
          hourDif = Abs(23 - Hour(timeNow) + (Hour(txtTime.Text) - Hour(timeNow)))
          minDif = Abs(59 - Minute(timeNow) + (Minute(txtTime.Text) - Minute(timeNow)))
          secDif = Abs(59 - Second(timeNow) + (Second(txtTime.Text) - Second(timeNow)))
        Else
          hourDif = 23
          minDif = 59
          secDif = 59
        End If
      End If
End Select

lblDifTime.Caption = TimeSerial(hourDif, minDif, secDif) ' show time to remain
If dayOut = 0 Or dayOut = 1 Then
  Label5.Caption = 0
Else
```

```
        Label5.Caption = dayOut - 1 ' output day
    End If
    Label7.Caption = hourDif
    Label8.Caption = minDif
    Label9.Caption = secDif
    Call chk_everyday
    Call sendDays
    Call sendHour
    Call sendMin
    Call sendSec
End Sub

Sub chk_everyday()
    Select Case chkEveryday.Value
        Case 1
            mscSender.Output = Chr$(&H7B)     ' send "{"
        Case Else
            mscSender.Output = Chr$(&H7D)     ' send "}"
    End Select
End Sub

Sub sendDays()
    Select Case dayOut
        Case "0"
            mscSender.Output = Chr$(&H0)    ' 0
        Case "1"
            mscSender.Output = Chr$(&H0)    ' 1
        Case "2"
            mscSender.Output = Chr$(&H1)    ' 2
        Case "3"
            mscSender.Output = Chr$(&H2)    ' 3
```

```
        Case "4"
            mscSender.Output = Chr$(&H3)   ' 4
        Case "5"
            mscSender.Output = Chr$(&H4)   ' 5
        Case "6"
            mscSender.Output = Chr$(&H5)   ' 6
        Case "7"
            mscSender.Output = Chr$(&H6)   ' 7
    End Select
End Sub

Sub sendHour()
    Select Case hourDif
        Case "0"
            mscSender.Output = Chr$(&H0)   ' 0
        Case "1"
            mscSender.Output = Chr$(&H1)   ' 1
        Case "2"
            mscSender.Output = Chr$(&H2)   ' 2
        Case "3"
            mscSender.Output = Chr$(&H3)   ' 3
        Case "4"
            mscSender.Output = Chr$(&H4)   ' 4
        Case "5"
            mscSender.Output = Chr$(&H5)   ' 5
        Case "6"
            mscSender.Output = Chr$(&H6)   ' 6
        Case "7"
            mscSender.Output = Chr$(&H7)   ' 7
        Case "8"
            mscSender.Output = Chr$(&H8)   ' 8
```

```
Case "9"
  mscSender.Output = Chr$(&H9)   ' 9
Case "10"
   mscSender.Output = Chr$(&HA)   ' 10
Case "11"
   mscSender.Output = Chr$(&HB)   ' 11
Case "12"
   mscSender.Output = Chr$(&HC)   ' 12
Case "13"
   mscSender.Output = Chr$(&HD)   ' 13
Case "14"
   mscSender.Output = Chr$(&HE)   ' 14
Case "15"
  mscSender.Output = Chr$(&HF)    ' 15
Case "16"
  mscSender.Output = Chr$(&H10)   ' 16
Case "17"
  mscSender.Output = Chr$(&H11)   ' 17
Case "18"
  mscSender.Output = Chr$(&H12)   ' 18
Case "19"
   mscSender.Output = Chr$(&H13)   ' 19
Case "20"
  mscSender.Output = Chr$(&H14)   ' 20
Case "21"
   mscSender.Output = Chr$(&H15)   ' 21
Case "22"
   mscSender.Output = Chr$(&H16)   ' 22
Case "23"
   mscSender.Output = Chr$(&H17)   ' 23
End Select
```

```
End Sub

Sub sendMin()
  Select Case minDif
    Case "0"
      mscSender.Output = Chr$(&H0)   ' 0
    Case "1"
      mscSender.Output = Chr$(&H1)   ' 1
    Case "2"
      mscSender.Output = Chr$(&H2)   ' 2
    Case "3"
      mscSender.Output = Chr$(&H3)   ' 3
    Case "4"
      mscSender.Output = Chr$(&H4)   ' 4
    Case "5"
      mscSender.Output = Chr$(&H5)   ' 5
    Case "6"
      mscSender.Output = Chr$(&H6)   ' 6
    Case "7"
      mscSender.Output = Chr$(&H7)   ' 7
    Case "8"
      mscSender.Output = Chr$(&H8)   ' 8
    Case "9"
      mscSender.Output = Chr$(&H9)   ' 9
    Case "10"
      mscSender.Output = Chr$(&HA)   ' 10
    Case "11"
      mscSender.Output = Chr$(&HB)   ' 11
    Case "12"
      mscSender.Output = Chr$(&HC)   ' 12
```

```
Case "13"
   mscSender.Output = Chr$(&HD)   ' 13
Case "14"
   mscSender.Output = Chr$(&HE)   ' 14
Case "15"
  mscSender.Output = Chr$(&HF)   ' 15
Case "16"
  mscSender.Output = Chr$(&H10)   ' 16
Case "17"
  mscSender.Output = Chr$(&H11)   ' 17
Case "18"
  mscSender.Output = Chr$(&H12)   ' 18
Case "19"
   mscSender.Output = Chr$(&H13)   ' 19
Case "20"
  mscSender.Output = Chr$(&H14)   ' 20
Case "21"
   mscSender.Output = Chr$(&H15)   ' 21
Case "22"
   mscSender.Output = Chr$(&H16)   ' 22
Case "23"
   mscSender.Output = Chr$(&H17)   ' 23
Case "24"
   mscSender.Output = Chr$(&H18)   ' 24
Case "25"
   mscSender.Output = Chr$(&H19)   ' 25
Case "26"
   mscSender.Output = Chr$(&H1A)   ' 26
Case "27"
   mscSender.Output = Chr$(&H1B)   ' 27
```

```
Case "28"
   mscSender.Output = Chr$(&H1C)   ' 28
Case "29"
   mscSender.Output = Chr$(&H1D)   ' 29
Case "30"
   mscSender.Output = Chr$(&H1E)   ' 30
Case "31"
   mscSender.Output = Chr$(&H1F)   ' 31
Case "32"
   mscSender.Output = Chr$(&H20)   ' 32
Case "33"
   mscSender.Output = Chr$(&H21)   ' 33
Case "34"
   mscSender.Output = Chr$(&H22)   ' 34
Case "35"
   mscSender.Output = Chr$(&H23)   ' 35
Case "36"
   mscSender.Output = Chr$(&H24)   ' 36
Case "37"
   mscSender.Output = Chr$(&H25)   ' 37
Case "38"
   mscSender.Output = Chr$(&H26)   ' 38
Case "39"
   mscSender.Output = Chr$(&H27)   ' 39
Case "40"
   mscSender.Output = Chr$(&H28)   ' 40
Case "41"
   mscSender.Output = Chr$(&H29)   ' 41
Case "42"
   mscSender.Output = Chr$(&H2A)   ' 42
```

```
Case "43"
   mscSender.Output = Chr$(&H2B)   ' 43
Case "44"
   mscSender.Output = Chr$(&H2C)   ' 44
Case "45"
   mscSender.Output = Chr$(&H2D)   ' 45
Case "46"
   mscSender.Output = Chr$(&H2E)   ' 46
Case "47"
   mscSender.Output = Chr$(&H2F)   '47
Case "48"
   mscSender.Output = Chr$(&H30)   ' 48
Case "49"
   mscSender.Output = Chr$(&H31)   ' 49
Case "50"
   mscSender.Output = Chr$(&H32)   ' 50
Case "51"
   mscSender.Output = Chr$(&H33)   ' 51
Case "52"
   mscSender.Output = Chr$(&H34)   ' 52
Case "53"
   mscSender.Output = Chr$(&H35)   ' 53
Case "54"
   mscSender.Output = Chr$(&H36)   ' 54
Case "55"
   mscSender.Output = Chr$(&H37)   ' 55
Case "56"
   mscSender.Output = Chr$(&H38)   ' 56
Case "57"
   mscSender.Output = Chr$(&H39)   ' 57
```

```
        Case "58"
            mscSender.Output = Chr$(&H3A)   ' 58
        Case "59"
            mscSender.Output = Chr$(&H3B)   ' 59
        End Select
End Sub

Sub sendSec()
    Select Case secDif
        Case "0"
            mscSender.Output = Chr$(&H0)   ' 0
        Case "1"
            mscSender.Output = Chr$(&H1)   ' 1
        Case "2"
            mscSender.Output = Chr$(&H2)   ' 2
        Case "3"
            mscSender.Output = Chr$(&H3)   ' 3
        Case "4"
            mscSender.Output = Chr$(&H4)   ' 4
        Case "5"
            mscSender.Output = Chr$(&H5)   ' 5
        Case "6"
            mscSender.Output = Chr$(&H6)   ' 6
        Case "7"
            mscSender.Output = Chr$(&H7)   ' 7
        Case "8"
            mscSender.Output = Chr$(&H8)   ' 8
        Case "9"
            mscSender.Output = Chr$(&H9)   ' 9
        Case "10"
            mscSender.Output = Chr$(&HA)   ' 10
```

```
Case "11"
   mscSender.Output = Chr$(&HB)   ' 11
Case "12"
   mscSender.Output = Chr$(&HC)   ' 12
Case "13"
   mscSender.Output = Chr$(&HD)   ' 13
Case "14"
   mscSender.Output = Chr$(&HE)   ' 14
Case "15"
  mscSender.Output = Chr$(&HF)   ' 15
Case "16"
  mscSender.Output = Chr$(&H10)   ' 16
Case "17"
  mscSender.Output = Chr$(&H11)   ' 17
Case "18"
  mscSender.Output = Chr$(&H12)   ' 18
Case "19"
   mscSender.Output = Chr$(&H13)   ' 19
Case "20"
  mscSender.Output = Chr$(&H14)   ' 20
Case "21"
   mscSender.Output = Chr$(&H15)   ' 21
Case "22"
   mscSender.Output = Chr$(&H16)   ' 22
Case "23"
   mscSender.Output = Chr$(&H17)   ' 23
Case "24"
   mscSender.Output = Chr$(&H18)   ' 24
Case "25"
   mscSender.Output = Chr$(&H19)   ' 25
```

```
Case "26"
   mscSender.Output = Chr$(&H1A)   ' 26
Case "27"
   mscSender.Output = Chr$(&H1B)   ' 27
Case "28"
   mscSender.Output = Chr$(&H1C)   ' 28
Case "29"
   mscSender.Output = Chr$(&H1D)   ' 29
Case "30"
   mscSender.Output = Chr$(&H1E)   ' 30
Case "31"
   mscSender.Output = Chr$(&H1F)   ' 31
Case "32"
   mscSender.Output = Chr$(&H20)   ' 32
Case "33"
   mscSender.Output = Chr$(&H21)   ' 33
Case "34"
   mscSender.Output = Chr$(&H22)   ' 34
Case "35"
   mscSender.Output = Chr$(&H23)   ' 35
Case "36"
   mscSender.Output = Chr$(&H24)   ' 36
Case "37"
   mscSender.Output = Chr$(&H25)   ' 37
Case "38"
   mscSender.Output = Chr$(&H26)   ' 38
Case "39"
   mscSender.Output = Chr$(&H27)   ' 39
Case "40"
   mscSender.Output = Chr$(&H28)   ' 40
```

```
Case "41"
   mscSender.Output = Chr$(&H29)   ' 41
Case "42"
   mscSender.Output = Chr$(&H2A)   ' 42
Case "43"
   mscSender.Output = Chr$(&H2B)   ' 43
Case "44"
   mscSender.Output = Chr$(&H2C)   ' 44
Case "45"
   mscSender.Output = Chr$(&H2D)   ' 45
Case "46"
   mscSender.Output = Chr$(&H2E)   ' 46
Case "47"
   mscSender.Output = Chr$(&H2F)   '47
Case "48"
   mscSender.Output = Chr$(&H30)   ' 48
Case "49"
   mscSender.Output = Chr$(&H31)   ' 49
Case "50"
   mscSender.Output = Chr$(&H32)   ' 50
Case "51"
   mscSender.Output = Chr$(&H33)   ' 51
Case "52"
   mscSender.Output = Chr$(&H34)   ' 52
Case "53"
   mscSender.Output = Chr$(&H35)   ' 53
Case "54"
   mscSender.Output = Chr$(&H36)   ' 54
Case "55"
   mscSender.Output = Chr$(&H37)   ' 55
```

```
        Case "56"
            mscSender.Output = Chr$(&H38)   ' 56
        Case "57"
            mscSender.Output = Chr$(&H39)   ' 57
        Case "58"
            mscSender.Output = Chr$(&H3A)   ' 58
        Case "59"
            mscSender.Output = Chr$(&H3B)   ' 59
        End Select
    End Sub

    Private Sub picExitOn_Click()
        'Remove the icon from the taskbar
        Shell_NotifyIcon NIM_DELETE, NotifyIcon

        'End the program
        Unload Me
    End Sub

    Private Sub tmrTurnOn_Timer()
    On Error Resume Next
        tmrTurnOn.Enabled = True
        mscSender.DTREnable = False
        mscSender.DTREnable = True
        mscSender.InputLen = 0
        replyStatus = mscSender.Input
        c_reply = Asc(replyStatus)
        Label4.Caption = replyStatus
    End Sub
```

1.3 Code Form3 (Turn Off Form)

```
Option Explicit

Private Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal
hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As
Long, ByVal wFlags As Long) As Long
Private Const SWP_NOMOVE = &H2
Private Const SWP_NOSIZE = &H1
Private Const HWND_TOPMOST = -1
Private Const HWND_NOTOPMOST = -2

Private Const TOKEN_ADJUST_PRIVILEGES As Long = &H20
Private Const TOKEN_QUERY As Long = &H8
Private Const SE_PRIVILEGE_ENABLED As Long = &H2

Private Const EWX_LOGOFF As Long = &H0
Private Const EWX_SHUTDOWN As Long = &H1
Private Const EWX_REBOOT As Long = &H2
Private Const EWX_FORCE As Long = &H4
Private Const EWX_POWEROFF As Long = &H8
Private Const EWX_FORCEIFHUNG As Long = &H10 '2000/XP only

Private Const VER_PLATFORM_WIN32_NT As Long = 2

Private CountDown As Date

Private Type OSVERSIONINFO
  OSVSize         As Long
  dwVerMajor      As Long
  dwVerMinor      As Long
  dwBuildNumber   As Long
```

```
 PlatformID      As Long
 szCSDVersion    As String * 128
End Type

Private Type LUID
  dwLowPart As Long
  dwHighPart As Long
End Type

Private Type LUID_AND_ATTRIBUTES
  udtLUID As LUID
  dwAttributes As Long
End Type

Private Type TOKEN_PRIVILEGES
  PrivilegeCount As Long
  laa As LUID_AND_ATTRIBUTES
End Type

Private Declare Function ExitWindowsEx Lib "user32" _
  (ByVal dwOptions As Long, _
  ByVal dwReserved As Long) As Long

Private Declare Function GetCurrentProcess Lib "kernel32" () As Long

Private Declare Function OpenProcessToken Lib "advapi32" _
 (ByVal ProcessHandle As Long, _
  ByVal DesiredAccess As Long, _
  TokenHandle As Long) As Long
```

```
Private Declare Function LookupPrivilegeValue Lib "advapi32" _
  Alias "LookupPrivilegeValueA" _
 (ByVal lpSystemName As String, _
  ByVal lpName As String, _
  lpLuid As LUID) As Long

Private Declare Function AdjustTokenPrivileges Lib "advapi32" _
 (ByVal TokenHandle As Long, _
  ByVal DisableAllPrivileges As Long, _
  NewState As TOKEN_PRIVILEGES, _
  ByVal BufferLength As Long, _
  PreviousState As Any, _
  ReturnLength As Long) As Long

Private Declare Function GetVersionEx Lib "kernel32" _
  Alias "GetVersionExA" _
 (lpVersionInformation As OSVERSIONINFO) As Long

Dim uFlags As Long
Dim success As Long

Dim StartStop As Integer
Dim State As Integer
Dim State2 As Integer

Dim DoItNow As Boolean

Dim Secs As Long 'Has to be a long number for the hours calculation
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' The code below is to move the form with the mouse, i didn't write it
' and i dont know who did. I found it on a website many months ago.
' If you wrote it thank you and i appologise for using it without permission
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' The following are used when moving the form using
' the GetCursorPos and SetWindowPlacement functions.

Dim lFormTopMouseDown  As Long
Dim lFormLeftMouseDown As Long
Dim CursorLoc          As POINTAPI
Dim lpwndpl            As WINDOWPLACEMENT

Const SW_SHOWNORMAL = 1

Private Type POINTAPI
   X As Long
   Y As Long
End Type

Private Type RECT
   Left   As Long
   Top    As Long
   Right  As Long
   Bottom As Long
End Type

Private Type WINDOWPLACEMENT
   Length        As Long
   flags         As Long
   ShowCmd       As Long
```

```
  ptMinPosition    As POINTAPI
  ptMaxPosition    As POINTAPI
  rcNormalPosition As RECT
End Type

Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Private Declare Function SetWindowPlacement Lib "user32" (ByVal hWnd As Long, lpwndpl As
WINDOWPLACEMENT) As Long
'
' Thes following are used when using the
' ReleaseCapture method of moving the form.
'
Const HTCAPTION = 2
Const WM_NCLBUTTONDOWN = &HA1
'dss
Const HTLEFT = 10
Const HTRight = 11

Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long,
ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call ReleaseCapture
Call SendMessage(hWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0&)
End Sub
''''''''''''''''''''''''''''''''''' END OF MOVE FORM CODE''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
Private Sub cmdCancleOff_Click()
  'End the form turn Off
  Unload Me
End Sub

Private Sub cmdDoItNowOff_Click()
  DoItNow = True

  If OptShut.Value = True Then
    If MsgBox("Windows Will Now Shut Down", vbOKCancel Or vbExclamation, "Shut Down") =
vbCancel Then
      DoItNow = False
      Exit Sub
    Else
      uFlags = EWX_POWEROFF
    End If

  ElseIf OptLogOff.Value = True Then
    If MsgBox("Windows Will Now Log Off", vbOKCancel Or vbExclamation, "Log Off") =
vbCancel Then
      DoItNow = False
      Exit Sub
    Else
      uFlags = EWX_LOGOFF
    End If

  ElseIf OptRestart.Value = True Then
    If MsgBox("Windows Will Now Reboot", vbOKCancel Or vbExclamation, "Reboot") =
vbCancel Then
      DoItNow = False
      Exit Sub
```

```
    Else
      uFlags = EWX_REBOOT
    End If
  End If

  'if running under NT or better,
  'the shutdown privileges need to
  'be adjusted to allow the ExitWindowsEx
  'call. If the adjust call fails on a NT+
  'system, success holds False, preventing shutdown.
  If IsWinNTPlus() Then
    success = EnableShutdownPrivledges()
  End If

  If success Then
    Call ExitWindowsEx(uFlags, 0&)
  Else
    '9x system, so just do it
    Call ExitWindowsEx(uFlags, 0&)
  End If
End Sub

'*********************************************************************************
'* This is the code for the start stop button.
'*********************************************************************************
Private Sub cmdStartStopOff_Click()
   ' check lblHours,lblMins and blSecs must have value
  Secs = (lblHoursOff * 60 * 60) + (lblMinsOff.Caption * 60) + lblSecsOff

  If Secs = 0 Then
    MsgBox ("Enter Duration"), vbOKOnly + vbExclamation, "Error"
```

```
    Exit Sub
End If
' cmdStartStopOff is a start or atop
If StartStop = 2 Then
    StartStop = 1
Else
    StartStop = StartStop + 1
End If
CountDown = DateAdd("s", Secs, Now)
Select Case StartStop
    ' cmdStartStopOff is a start
    Case 1
        cmdStartStopOff.Caption = "&Start"
        tmrTime.Enabled = False
        lblCounter.Caption = "0:00:00"
        UpDownHoursOff.Enabled = True
        UpDownMinsOff.Enabled = True
        updownSecsOff.Enabled = True
        lblHoursOff.Enabled = True
        lblMinsOff.Enabled = True
        lblSecsOff.Enabled = True
        lblHoursOff.Caption = 0
        lblMinsOff.Caption = 0
        lblSecsOff.Caption = 0
        UpDownHoursOff.Value = 0
        UpDownMinsOff.Value = 0
        updownSecsOff.Value = 0
        OptShut.Enabled = True
        OptLogOff.Enabled = True
        OptRestart.Enabled = True
    ' cmdStartStopOff is a stop
```

```
        Case 2
            cmdStartStopOff.Caption = "&Stop"
            tmrTime.Enabled = True
            UpDownHoursOff.Enabled = False
            UpDownMinsOff.Enabled = False
            updownSecsOff.Enabled = False
            lblHoursOff.Enabled = False
            lblMinsOff.Enabled = False
            lblSecsOff.Enabled = False
            OptShut.Enabled = False
            OptLogOff.Enabled = False
            OptRestart.Enabled = False
    End Select
End Sub

Private Sub Form_Load()
    State2 = 1
    StartStop = 1
    DoItNow = False

    OptShut.Value = True
    tmrTime.Enabled = False

    If App.PrevInstance = True Then
        MsgBox "The program is already running!", vbCritical Or vbSystemModal, "Shut It! - Error"
        End
    End If
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
   If lblCounter.Caption <> "0:00:00" And DoItNow = False Then
      Unload Me
   End If
End Sub

Private Sub OptLogOff_Click()
   If OptLogOff.Value = True Then
      lblAction.Caption = "~~ Log Off In ~~"
   End If
End Sub

Private Sub OptRestart_Click()
   If OptRestart.Value = True Then
      lblAction.Caption = "~~ Restart In ~~"
  End If
End Sub

Private Sub OptShut_Click()
   If OptShut.Value = True Then
      lblAction.Caption = "~~ Shutdown In ~~"
   End If
End Sub

Private Sub picExit_Click()
   'Remove the icon from the taskbar
   Shell_NotifyIcon NIM_DELETE, NotifyIcon
   'End the program
   Unload Me
End Sub
```

```
Private Sub tmrTime_Timer()
   Dim txt As String
   txt = Format$(CountDown - Now, "h:mm:ss")
   lblCounter.Caption = txt

   If OptShut.Value = True Then
      Me.Caption = "Shutdown in " & txt
   ElseIf OptLogOff.Value = True Then
      Me.Caption = "Logoff in: " & txt
   ElseIf OptRestart.Value = True Then
      Me.Caption = "Restart in: " & txt
   End If

   If lblCounter.Caption = "0:00:00" Then
      tmrTime.Enabled = False
      cmdStartStopOff.Caption = "&Finish"
      If OptShut.Value = True Then
         uFlags = EWX_POWEROFF
      ElseIf OptLogOff.Value = True Then
         uFlags = EWX_LOGOFF
      ElseIf OptRestart.Value = True Then
         uFlags = EWX_REBOOT
      End If
      'if running under NT or better,
      'the shutdown privileges need to
      'be adjusted to allow the ExitWindowsEx
      'call. If the adjust call fails on a NT+
      'system, success holds False, preventing shutdown.
      If IsWinNTPlus() Then
         success = EnableShutdownPrivledges()
      End If
```

```
        If success Then
            Call ExitWindowsEx(uFlags, 0&)
        Else
            '9x system, so just do it
            Call ExitWindowsEx(uFlags, 0&)
        End If
    End If
End Sub
Private Function IsWinNTPlus() As Boolean
  'returns True if running Windows NT,
  'Windows 2000, Windows XP, or .net server
  #If Win32 Then
    Dim OSV As OSVERSIONINFO
    OSV.OSVSize = Len(OSV)
    If GetVersionEx(OSV) = 1 Then
      IsWinNTPlus = (OSV.PlatformID = VER_PLATFORM_WIN32_NT) And _
                (OSV.dwVerMajor >= 4)
    End If
  #End If
'End Function

Private Function EnableShutdownPrivledges() As Boolean

  Dim hProcessHandle As Long
  Dim hTokenHandle As Long
  Dim lpv_la As LUID
  Dim token As TOKEN_PRIVILEGES
  hProcessHandle = GetCurrentProcess()
  If hProcessHandle <> 0 Then
    'open the access token associated
    'with the current process. hTokenHandle
```

```
'returns a handle identifying the
'newly-opened access token
 If OpenProcessToken(hProcessHandle, _
            (TOKEN_ADJUST_PRIVILEGES Or TOKEN_QUERY), _
             hTokenHandle) <> 0 Then

  'obtain the locally unique identifier
  '(LUID) used on the specified system
  'to locally represent the specified
  'privilege name. Passing vbNullString
  'causes the api to attempt to find
  'the privilege name on the local system.
  If LookupPrivilegeValue(vbNullString, _
                "SeShutdownPrivilege", _
                lpv_la) <> 0 Then

   'TOKEN_PRIVILEGES contains info about
   'a set of privileges for an access token.
   'Prepare the TOKEN_PRIVILEGES structure
   'by enabling one privilege.
    With token
      .PrivilegeCount = 1
      .laa.udtLUID = lpv_la
      .laa.dwAttributes = SE_PRIVILEGE_ENABLED
    End With

   'Enable the shutdown privilege in the access token of
   'this process.

   'hTokenHandle: access token containing the privileges to be modified.
```

```
        'DisableAllPrivileges: if True the function disables all privileges
        'and ignores the

        'NewState parameter. If FALSE, the function modifies privileges based
        'on the information pointed to by NewState.

        'token: TOKEN_PRIVILEGES structure specifying an array of privileges
        'and their attributes.

        'Since were just adjusting to shut down, BufferLength, PreviousState
        'and ReturnLength can be passed as null.

        If AdjustTokenPrivileges(hTokenHandle, _
                                 False, _
                                 token, _
                                 ByVal 0&, _
                                 ByVal 0&, _
                                 ByVal 0&) <> 0 Then

          'success, so return True
          EnableShutdownPrivledges = True
        End If  'AdjustTokenPrivileges
      End If  'LookupPrivilegeValue
    End If  'OpenProcessToken
  End If  'hProcessHandle
End Function
```

1.4 Code Shellnotify

```
'For more info you may wish to check out the
'original article by the Cobb Group that this
'project was derived from at http://www.cobb.com/
'(Now part of Ziff-Davis I believe.) That is
'where I got these declarations.

'Declare the API function Shell_NotifyIcon
Public Declare Function Shell_NotifyIcon Lib _
"shell32.dll" Alias "Shell_NotifyIconA" _
(ByVal dwMessage As Long, lpData As NOTIFYICONDATA) As Long

'Make a data type that will store icon properties
  Public Type NOTIFYICONDATA
     cbSize As Long
     hWnd As Long
     uID As Long
     uFlags As Long
     uCallbackMessage As Long
     hIcon As Long
     szTip As String * 64
  End Type

'Create a variable of that data type
  Global NotifyIcon As NOTIFYICONDATA

'Some constants needed for Shell_NotifyIcon
  Global Const NIM_ADD = &H0
  Global Const NIM_MODIFY = &H1
  Global Const NIM_DELETE = &H2
  Global Const NIF_MESSAGE = &H1
```

```
Global Const NIF_ICON = &H2
Global Const NIF_TIP = &H4
Global Const WM_MOUSEMOVE = &H200
```

2. Source Code ในส่วนของไมโครคอนโทรลเลอร์

```
'========== DEFINED CHIP'S CONFIGURATION ==========
@  DEVICE PIC16F877A,HS_OSC,LVP_OFF,BOD_OFF,WDT_OFF

'========== DEFINED PARAMETER ==========
DEFINE osc    20     ' 20 MHz.
DEFINE mode   0      ' non-inverter mode

'========== DERAIL CONSTANT ==========
LED VAR PORTA.5
OUTOPEN VAR PORTB.7 ' SIGNAL TO TURN ON COMPUTER

HOURS   VAR BYTE    ' DECARED HOURS IS BYTE SIZE
MINS    VAR BYTE    ' DECARED MINS IS BYTE SIZE
SECS    VAR BYTE    ' DECARED SECS IS BYTE SIZE
DAY     VAR BYTE    ' DECARED DAY IS BYTE SIZE

COUNTH  VAR BYTE
COUNTM  VAR BYTE
COUNTS  VAR BYTE
COUNTD  VAR BYTE
IDAY    VAR BYTE
IHOURS  VAR BYTE
IMINS   VAR BYTE
ISECS   VAR BYTE
INITH   VAR BYTE
INITM   VAR BYTE
INITS   VAR BYTE
CHK_EVERY  VAR BYTE
CHKSTOP VAR BYTE
CHAR  var byte
```

```
INITH = 10
INITM = 10
INITS = 10

'=============================================================
'=                        MAIN PROGRAM                       =
'=============================================================
MAIN:
    SERIN2  PORTC.7,84,[CHAR]
    IF CHAR = 95 THEN
      serout2 PORTC.6,84,[#CHAR,13,10,13,10]
      GOTO   INPUTS
    ELSE
      GOTO   MAIN
    ENDIF

INPUTS:
    SERIN2  PORTC.7,84,[CHK_EVERY]
    SERIN2  PORTC.7,84,[IDAY]       ' recieve day
    SERIN2  PORTC.7,84,[IHOURS]     ' recieve hour
    SERIN2  PORTC.7,84,[IMINS]      ' receive minute
    SERIN2  PORTC.7,84,[ISECS]      ' receive second
    HOURS = IHOURS
    MINS = IMINS
    SECS = ISECS
    INITH = 10
    INITM = 10
    INITS = 10

DAYS:
    IF IDAY = 0 THEN
      DAY = IDAY
    ELSE
      DAY = IDAY - 1
    ENDIF
```

```
    FOR COUNTD = 0 TO IDAY STEP 1    ' Count Day
      IF INITH > 0 THEN
        INITH = 0
      ELSE
        HOURS = 23
        IHOURS = 23
      ENDIF
      SEROUT2 PORTC.6,84,[#DAY,"  DAYS",13,10]    ' send hour
      GOSUB  HOUR
      DAY = DAY - 1
    NEXT COUNTD

HOUR:
    FORCOUNTH = 0 TO IHOURS STEP 1    ' Count Hour
      IF INITM > 0 THEN
        INITM = 0
      ELSE
        MINS = 59
        IMINS = 59
      ENDIF
      SEROUT2  PORTC.6,84,[#HOURS,"  HOURS",13,10]  ' send hour
      GOSUB  MINUTE
      HOURS = HOURS - 1
    NEXT COUNTH

MINUTE:
    FOR COUNTM = 0 TO IMINS STEP 1     ' Count Minute
      IF INITS > 0 THEN
        INITS = 0
      ELSE
        SECS = 59
        ISECS = 59
      ENDIF
      SEROUT2  PORTC.6,84,[#MINS,"  MINS",13,10]  ' send minute
      GOSUB  SECOND
      MINS = MINS - 1
```

```
    NEXT COUNTM
    RETURN

SECOND:
    FOR COUNTS = 0 TO ISECS STEP 1      ' Count Second
      SEROUT2  PORTC.6,84,[#SECS,13,10]' send second
      PAUSE  890
      GOSUB  CHK_STOP
      IF HOURS = 0 AND MINS = 0 AND SECS = 0 AND DAY<>0 THEN
        DAY = DAY - 1
        GOTO DAYS
      ELSE
        IF HOURS = 0 AND MINS=0 AND SECS = 0 AND DAY = 0 THEN
          PAUSE  1000
          HIGH   OUTOPEN
          PAUSE  300
          LOW    OUTOPEN
          IF CHK_EVERY = 123 THEN
            'SEROUT2 PORTC.6,84,[#CHK_EVERY,13,10]
            DAY = 1
            GOTO   DAYS
          ELSE
            GOTO   MAIN
          ENDIF
        ENDIF
      ENDIF
      SECS = SECS - 1
    next COUNTS
    RETURN
```

```
CHK_STOP:
        SERIN2   PORTC.7, 84, 100, RETURNS, [CHKSTOP]
        IF CHKSTOP = 92 THEN
          GOTO   MAIN
        ELSE
          RETURN
        ENDIF

RETURNS:   RETURN
```