



ระบบบันทึกสัญญาณ Video และ Audio อัตโนมัติ โดยส่งงานผ่านระบบ Network

โดย
นายณัฐพล พิรินทรีย์วง รหัส B4602620
นายอรรถพงษ์ ดีสวัสดิ์ รหัส B4611462
นายวรพลแก้ว จันท์ รหัส B4711100

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427494 และ 427499 โครงการวิศวกรรมโทรคมนาคม
หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2552

โครงการ ระบบบันทึกสัญญาณ
Network

Video และ Audio อัตโนมัติ โดยส่งงานผ่านระบบ

จัดทำโดย นายณัฐพล พิรินทร์ยวง

นายอรรถพงษ์ ดีสวัสดิ์

นายวรพล แก้วจันทร์

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ วงศ์สรรค์

สาขาวิชา วิศวกรรมโทรคมนาคม

ภาคการศึกษาที่ 3/2553

บทคัดย่อ
(Abstract)

เครื่องบันทึกสัญญาณ Video และ Audio เป็นอุปกรณ์ช่วยบันทึกสัญญาณภาพและเสียง จากนั้นทำการแปลงสัญญาณที่ได้รับเป็นรูปแบบของไฟล์ต่างๆ เช่น MPEG WMV MOV หรือในรูปแบบอื่นๆ ไปยัง Harddisk ซึ่งเป็นหน่วยความจำที่สามารถบันทึกและลบข้อมูลได้ไม่จำกัดจำนวนครั้ง โดยส่งงานผ่านระบบ Network

เนื่องจากในปัจจุบันต้องใช้คอมพิวเตอร์ ผสมกับ Card Capture ในการบันทึกและแปลงสัญญาณ ปัญหาที่เกิดขึ้น คือ ต้องใช้บุคลากรตลอดระยะเวลาในการบันทึกสัญญาณและแปลงไฟล์สัญญาณ

โครงการจึงมีประโยชน์ในด้าน ดังต่อไปนี้

- สามารถตั้งเวลาในการบันทึกล่วงหน้าได้
- ได้ไฟล์ในรูปแบบที่เราต้องการ โดยไม่ต้องเสียเวลาในการแปลงไฟล์และไม่ต้องใช้บุคลากรในการทำงานตลอดเวลา
- สามารถแจ้งเตือนสถานะ การทำงานผ่านระบบเครือข่ายได้

เครื่องบันทึกสัญญาณ Video และ Audio จะสามารถควบคุมการทำงานผ่าน Microcontroller ตามที่เราได้กำหนดค่าในการทำงานไว้

กิตติกรรมประกาศ (Acknowledgement)

จากการที่คณะจัดทำรายงานได้รับมอบหมายให้ทำโครงการเรื่อง ระบบบันทึกสัญญาณ Video และ Audio อัตโนมัติ โดยส่งงานผ่านระบบ Network ส่งผลให้คณะจัดทำรายงานได้รับความรู้และประสบการณ์ต่างๆ เกี่ยวกับการเขียนโปรแกรมด้วยโปรแกรมวิซวลเบสิกเวอร์ชันหกจุดศูนย์ (Visual Basic 6.0) และภาษาแอสเซมบลี (Assembly) เป็นอย่างมาก บัดนี้โครงการดังกล่าวพร้อมทั้งรายงานได้สำเร็จลงแล้ว ทั้งนี้ด้วยความร่วมมือและสนับสนุนจากบุคคลต่างๆ ดังนี้

1. ผศ.ดร.รังสรรค์ วงศ์สรรค์ (อาจารย์ที่ปรึกษาโครงการ)
2. นายอานวย ทีจันทิก (วิศวกร)
3. นางสาวณานิ นະพุทธะ (นักศึกษาปริญญาโท สาขาวิชาวิศวกรรมโทรคมนาคม)

ข้าพเจ้าคณะผู้จัดทำโครงการนี้ใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูลและเป็นที่ปรึกษาในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับพื้นฐานการใช้งาน โปรแกรม ซึ่งข้าพเจ้าขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

นายณัฐพล พิรินทร์ยวง

นายอรรถพงษ์ ดีสวัสดิ์

นายวรพล แก้วจันทร์

คณะผู้จัดทำ

สารบัญ

เรื่อง	หน้า	
บทคัดย่อ	ก	
กิตติกรรมประกาศ		ข
สารบัญ		ค
สารบัญภาพ		จ
สารบัญตาราง	ฉ	
บทที่ 1	บทนำ	1
1.1	หลักการและเหตุผล	1
1.2	วัตถุประสงค์	1
1.3	ขอบเขตงาน	2
1.4	ระยะเวลาการดำเนินงาน	2
1.5	งบประมาณ	3
1.6	ผลที่คาดว่าจะได้รับ	3
1.7	การประเมินผลงาน	3
บทที่ 2	ส่วนประกอบของโครงงานและทฤษฎีที่เกี่ยวข้อง	4
2.1	ไมโครคอนโทรลเลอร์	5
2.2	หน่วยแสดงผล LCD	7
2.3	ไอซีฐานเวลา RTC	8
2.4	ไอซีหน่วยความจำ EEPROM	9
2.5	ตัวจัดการข้อมูลผ่านเครือข่าย (Ethernet Modules)	10
2.6	ทฤษฎีของ TCP/IP	14
2.6.1	ชั้นโฮสเครือข่าย	15
2.6.2	ชั้นสื่อสารอินเทอร์เน็ต	15
2.6.3	ชั้นสื่อสารการนำส่งข้อมูล	15
2.6.4	ชั้นสื่อสารการประยุกต์	17
2.7	ความรู้เบื้องต้นเกี่ยวกับ VISUAL BASIC	18
	ทำความรู้จักกับ Winsock	19

สารบัญ (ต่อ)

เรื่อง	หน้า
บทที่ 3 การออกแบบโครงงาน	20
3.1 การออกแบบทางฮาร์ดแวร์	20
3.1.1 EZL-50L	21
3.1.2 Microprocessor	22
3.2 การออกแบบการทำงานผ่าน Visual Basic	24
3.2.1 การสร้างแอปพลิเคชันจาก Winsock Control	24
3.2.2 การออกแบบ Visual Basic	25
3.2.3 คำสั่งในการทำงานของ Visual Basic	28
3.2.4 วิธีการสร้างโปรแกรมเพื่อใช้งาน	30
3.3 การออกแบบระบบการทำงานโดยรวม	31
บทที่ 4 การทดสอบและการใช้งาน	33
4.1 การทดสอบการใช้งาน โดยการป้อนข้อมูลผ่านคีย์แปด	34
4.2 การทดสอบการใช้งาน โดยการป้อนข้อมูลผ่านระบบเครือข่าย การใช้งานผ่าน Visual Basic	39 44
บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ	47
5.1 สรุปผล	47
5.2 สิ่งที่ได้รับจากการทำโครงงาน	48
5.3 ปัญหาและอุปสรรค	48
5.4 ข้อเสนอแนะและแนวทางการพัฒนา	49
เอกสารอ้างอิง	50
ภาคผนวก	52

สารบัญภาพ

รายการ	หน้า	
รูปที่ 2.1 ภาพรวมการทำงานของระบบทั้งหมด		5
รูปที่ 2.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์		6
รูปที่ 2.3 LCD Module		7
รูปที่ 2.4 ภาพการจัดสรรหน่วยความจำใน DS1307		9
รูปที่ 2.5 ไคอะแกรมโครงสร้างของหน่วยความจำ EEPROM 24xxx		10
รูปที่ 2.6 แสดงลักษณะโดยรวมของ EZL-50L		11
รูปที่ 2.7 Ethernet Interface		12
รูปที่ 2.8 โครงสร้าง TCP/IP		14
รูปที่ 2.9 ไคอะแกรมการสื่อสารของ TCP		16
รูปที่ 3.1 ฟังก์ชันของโปรแกรมของโครงการ		20
รูปที่ 3.2 แสดงลักษณะของโปรแกรม ezConfig v.4.4c		21
รูปที่ 3.3 แผนภาพแสดงส่วนประกอบของ Microprocessor		22
รูปที่ 3.4 ภาพแสดงบล็อกข้อมูล	22	
รูปที่ 3.5 การต่อวงจรการทำงานของ IC817 กับรีโมท	23	
รูปที่ 3.6 การเลือก new project		25
รูปที่ 3.7 การเลือก Tab Components		25
รูปที่ 3.8 การเรียกใช้ Winsock		26
รูปที่ 3.9 Form Design		26
รูปที่ 3.10 Code คำสั่งการเชื่อมต่อ	27	
รูปที่ 3.11 แถบแสดงสถานะ	28	
รูปที่ 3.12 TimeSetup		28
รูปที่ 3.13 DaySetup		29
รูปที่ 3.14 Run/Stop, Rec, StopRec		29
รูปที่ 3.15 โปรแกรม Setup		30
รูปที่ 3.16 ตรวจสอบข้อมูล		30
รูปที่ 3.17 โปรแกรมบันทึก Video และ Audio อัตโนมัติผ่านระบบ Network.exe		31
รูปที่ 3.18 Flowchart โปรแกรมโดยรวม		32

สารบัญภาพ (ต่อ)

รายการ	หน้า
รูปที่ 4.1 ชุดอุปกรณ์ที่ได้รับการประกอบแล้ว	33
ภาพการใช้งานโดยการป้อนข้อมูลผ่านคีย์แพด	
รูปที่ 4.2 และ รูปที่ 4.3	34
รูปที่ 4.4 และ รูปที่ 4.5	35
รูปที่ 4.6	36
รูปที่ 4.7	37
รูปที่ 4.8	38
รูปที่ 4.9	39
ภาพการใช้งานโดยการป้อนข้อมูลผ่านระบบเครือข่าย	
รูปที่ 4.10	39
รูปที่ 4.11 และ รูปที่ 4.12	40
รูปที่ 4.13 และ รูปที่ 4.14	41
รูปที่ 4.15	42
รูปที่ 4.16	43
รูปที่ 4.17 ทดสอบการเล่นไฟล์ที่ได้จากการบันทึก	44
รูปที่ 4.18 รายละเอียดไฟล์	45
รูปที่ 4.19 รายละเอียดของไฟล์ที่ได้จากการบันทึก	46

สารบัญตาราง

รายการ	หน้า
ตารางที่ 2.1 รายละเอียดข้อมูลจำเพาะของ EZL-50L	11
ตารางที่ 2.2 I/O Interface	13

บทที่ 1

บทนำ

1.1 หลักการ และเหตุผล

เนื่องจากในปัจจุบันการทำการบันทึกข้อมูลภาพและเสียงไปยังหน่วยความจำต่างๆ ไม่ว่าจะเป็นฮาร์ดดิสก์ เมโมรี่การ์ด หรือตัวหม่พิมพ์ไคร์ฟ จะมีปัญหาอยู่ตรงที่เราต้องทำการนำอุปกรณ์ทั้งหลายที่กล่าวมาทั้งหมดไปเชื่อมต่อกับตัวอุปกรณ์ที่ทำการบันทึกข้อมูลภาพหรือเสียงโดยตรง ซึ่งอุปกรณ์จำพวกนี้ก็ได้แก่ กล้องถ่ายรูป กล้องถ่ายวิดีโอ โทรทัศน์ หรือคอมพิวเตอร์ ซึ่งบางครั้งอาจจะทำให้เกิดความล่าช้าในการนำข้อมูลภาพและเสียงที่ต้องการนำมาใช้ อย่างในกรณีที่สถานที่ที่ทำการบันทึกข้อมูลภาพและเสียงอยู่ห่างไกลจากสถานที่จากสถานที่ที่ต้องการจะใช้ข้อมูล จะทำให้เกิดการเสียเวลา เปลืองทรัพยากรบุคคลที่จะต้องทำหน้าที่ไปรับไปส่งข้อมูลทั้งหลายเหล่านี้ ซึ่งถ้าหากเกิดเหตุการณ์ที่บุคลากรเกิดการผิดพลาดในการนำข้อมูลภาพและเสียงมาไม่ว่าจะเป็นการนำมาผิดหรือเกิดเหตุการณ์ที่ทำให้ข้อมูลเสียหายก็จะเกิดความล่าช้าขึ้นในสายงานอีก และในส่วนของสถานที่ที่ทำการบันทึกภาพและเสียงถ้าเกิดมีการบันทึกที่จะต้องใช้เวลาในการบันทึกเป็นเวลานานก็จะต้องใช้บุคลากรในการควบคุมการบันทึกอยู่ตลอดเวลา ทำให้ต้องเสียบุคลากรไปทั้งที่ในปัจจุบันเทคโนโลยีด้านการสื่อสารผ่านระบบเครือข่าย (Network) ไม่ว่าจะเป็นในรูปแบบ TCP/IP หรือ HTTP มีความก้าวหน้า รวมทั้งตัวอุปกรณ์และตัวโปรแกรมต่างๆ ก็สามารถรองรับการสื่อสารผ่านระบบเหล่านี้ได้เป็นจำนวนมาก

ด้วยเหตุนี้ทางผู้จัดทำได้ตระหนักถึงปัญหาทั้งหลายที่ได้กล่าวมา จึงได้คิดค้นและประดิษฐ์อุปกรณ์เพื่อที่จะทำให้การทำงานในส่วนนี้มีความสะดวกสบายรวดเร็วมากขึ้น รวมทั้งช่วยประหยัดเวลาและบุคลากร โดยใช้หลักการในการสื่อสารและส่งผ่านข้อมูลในระบบเครือข่าย ซึ่งอุปกรณ์นี้มีชื่อว่า ระบบบันทึกสัญญาณ Video และ Audio อัตโนมัติผ่านระบบเครือข่าย

1.2 วัตถุประสงค์

1. เพื่อศึกษาโปรแกรมควบคุมและประยุกต์ใช้งานไมโครคอนโทรลเลอร์ (Microcontroller)
2. เพื่อศึกษาวิธีการสื่อสารข้อมูลและการประมวลผลข้อมูล
3. เพื่อนำความรู้ที่ได้จากการศึกษาภาคทฤษฎีของวิชาต่างๆ ที่ได้ศึกษามาปฏิบัติและประยุกต์ใช้ เพื่อสร้างชิ้นงานขึ้นมาและสามารถนำไปใช้งานได้จริง

1.3 ขอบเขตงาน

1. อุปกรณ์สามารถตั้ง วันและระยะเวลาในการบันทึกข้อมูลได้
2. สามารถส่งสถานะในการทำงานไปยังคอมพิวเตอร์ของผู้เชื่อมต่อระบบได้
3. การควบคุมตารางการบันทึกสามารถตั้งค่าผ่านระบบ Network ได้

1.4 ระยะเวลาการดำเนินงาน

กิจกรรม	พ.ศ.2553				
	1-15 ม.ค	16-31 ม.ค	1-14 ก.พ	15-28 ก.พ	1-30 มี.ค
1. ศึกษาขอบเขตของโครงการและอุปกรณ์ ที่ต้องใช้ในการทำโครงการ	←→				
2. สั่งซื้ออุปกรณ์และศึกษาการใช้งาน โปรแกรมไมโครคอนโทรลเลอร์		←→			
3. ประกอบวงจรอิเล็กทรอนิกส์			←→		
4. เขียนโปรแกรมให้อุปกรณ์ทำงานตรงตาม ขอบเขตของโครงการ				←→	
5. ทดลองใช้งานอุปกรณ์ พร้อมทั้งปรับปรุง แก้ไขให้อุปกรณ์ใช้งานได้ตามต้องการ					←→

1.5 งบประมาณ

1. Harddisk Player Recorder	3000 บาท
2. Microcontroller Board	2000 บาท
3. สายไฟและอุปกรณ์เชื่อมต่อ	1000 บาท
4. EZL-50L	2000 บาท
รวม	8000 บาท

1.6 ผลที่คาดว่าจะได้รับ

1. ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ (Microcontroller)
2. ได้เรียนรู้การทำงานของวงจรรีเลย์ทรอนิกส์
3. ได้เรียนรู้การสื่อสารข้อมูลและการประมวลผล
4. ได้เรียนรู้การทำงานเป็นทีมและได้นำความรู้ที่ได้จากการศึกษามาปฏิบัติและประยุกต์ใช้งานจริง

1.7 การประเมินผลงาน

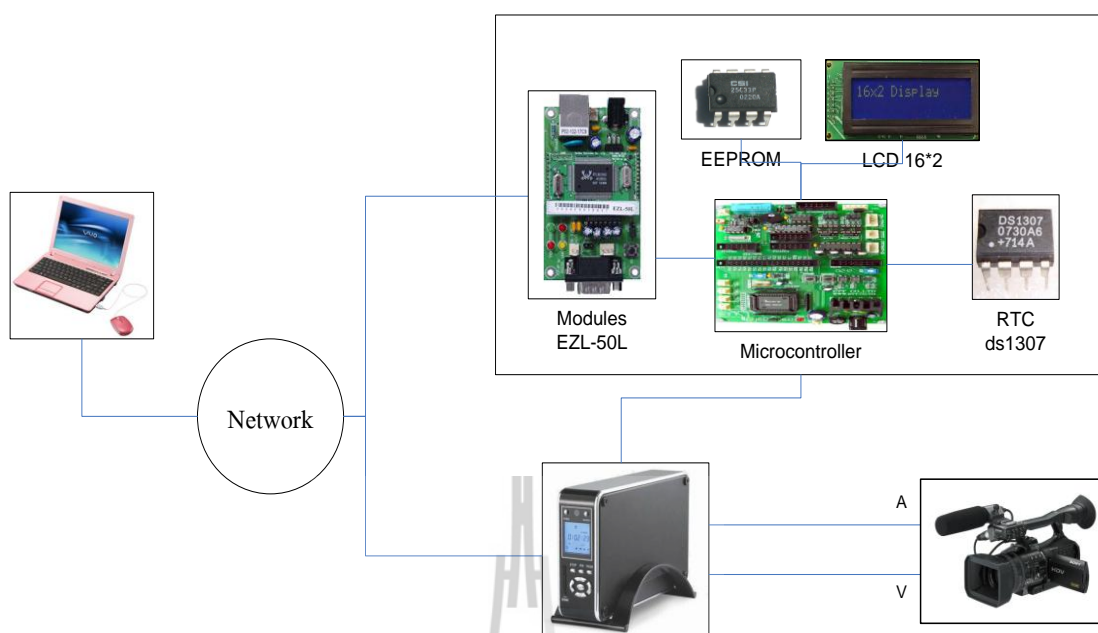
1. รายงานผลการทำโครงการให้อาจารย์ที่ปรึกษาโครงการตามขั้นตอนที่กำหนดในแผนปฏิบัติงาน
2. การสอบสัมภาษณ์จากอาจารย์ประจำสาขาวิชา

บทที่ 2

ส่วนประกอบของโครงงานและทฤษฎีที่เกี่ยวข้อง

ก่อนที่เราจะมากล่าวถึงส่วนประกอบของโครงงานว่าประกอบไปด้วยอุปกรณ์ชนิดใดบ้าง เราต้องมาทราบถึงภาพรวมของหลักการทำงานทั้งหมดของระบบเสียก่อน ซึ่งในระบบบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายนี้มีลักษณะการทำงานคือ เมื่อเราได้ทำการบันทึกสัญญาณภาพและเสียงไม่ว่าจะอยู่ที่ใดก็ตาม และสถานที่ที่ทำการบันทึกไม่จำเป็นจะต้องมีผู้ควบคุมการบันทึกอยู่ โดยที่ข้อมูลทั้งภาพและเสียงจะถูกนำไปเก็บไว้ในตัวฮาร์ดดิสก์เป็นการชั่วคราวก่อน แล้วผู้ใช้สามารถที่จะนำข้อมูลทั้งภาพและเสียงมาใช้งานโดยไม่จำเป็นต้องไปดึงข้อมูลโดยตรงที่ตัวฮาร์ดดิสก์ เพียงแต่ใช้คอมพิวเตอร์ในการดึงข้อมูลทั้งภาพและเสียงทั้งหมดผ่านระบบเครือข่ายในรูปแบบ TCP/IP และในขณะเดียวกันทางผู้ใช้ยังสามารถที่จะโปรแกรมสั่งการให้บันทึกหรือยกเลิกการบันทึกสัญญาณตอนไหนหรือแม้กระทั่งการตั้งเวลาให้กับการบันทึก โดยการสั่งการทั้งหมดนี้จะกระทำโดยผ่านชุดอุปกรณ์คอนโทรลเลอร์เป็นตัวกลางในการสั่งการได้โดยอัตโนมัติ โดยที่เราทำการโปรแกรมตั้งค่าลงในชุดอุปกรณ์คอนโทรลเลอร์ เมื่อเราสั่งการให้โปรแกรมทำงานตัวคอนโทรลเลอร์ก็จะทำการทำงานตามที่เราได้โปรแกรมไว้ ดังรูป 2.1 คือภาพรวมของระบบบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายอัตโนมัติ

จากการทำงานที่กล่าวมาทั้งหมด ในชุดคอนโทรลเลอร์จำเป็นที่จะต้องมียูนิทอื่นๆ ที่จำเป็นมาประกอบเข้าในชุดคอนโทรลเลอร์ ซึ่งเราจะกล่าวถึงในต่อไป



รูปที่ 2.1 ภาพรวมการทำงาน of ระบบทั้งหมด

โดยในส่วน of ระบบคอนโทรลเลอร์ที่จะใช้งานในการทำระบบการบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายนี้จะประกอบไปด้วยตัวอุปกรณ์ไอซีต่างๆ ที่สำคัญ คือ ไมโครคอนโทรลเลอร์, จอLCD, ไอซีฐานเวลา (Real Times Clock), ไอซีหน่วยความจำ (EEProm) และตัว Ethernet Modules

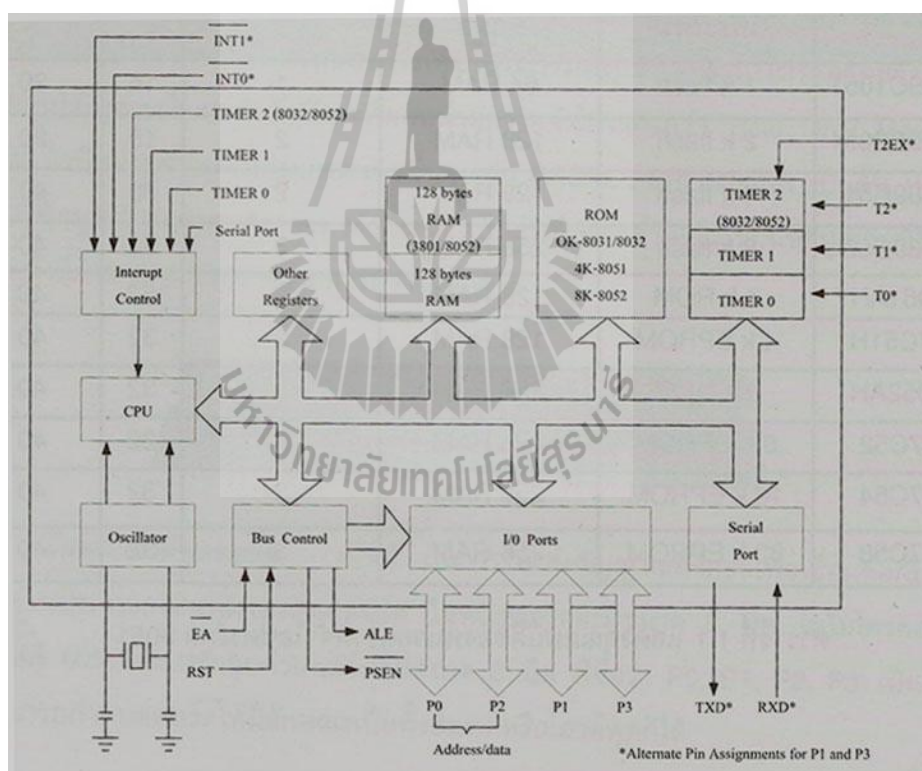
2.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่ง ที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน และเนื่องจากการทำงาน of อุปกรณ์ต่างๆ ในโครงงานนี้เราจะใช้ไมโครคอนโทรลเลอร์ในตระกูล 8051 มาเป็นตัวสั่งการในการทำงาน of ระบบ โดยในที่นี้เราจะใช้ไมโครคอนโทรลเลอร์ ในเบอร์ AT89C51 ซึ่งจะมีคุณสมบัติดังต่อไปนี้

- หน่วยประมวลผลกลาง (CPU) มีขนาด 8 บิต
- มีหน่วยความจำโปรแกรม (ROM) แบบ flash ขนาด 4 Kbyte
- มีหน่วยความจำข้อมูล (RAM) ขนาด 128 byte

- มีพอร์ตในการอินพุต/เอาต์พุตจำนวน 4 พอร์ต (Port0-3)
- มีตัวฟังก์ชัน Timer/counter จำนวน 2 ตัว Timer0, Timer1
- สามารถอินเทอร์รัปต์ได้ 5-6 แห่ง
- มีวงจรกำเนิดสัญญาณนาฬิกาภายในตัวไอซี
- มีพอร์ตอนุกรมที่สามารถสื่อสารรับส่งข้อมูลแบบ Full Duplex
- สามารถขยายหน่วยความจำโปรแกรมภายนอกได้ 64Kbyte
- สามารถขยายหน่วยความจำข้อมูลภายนอกได้ 64 Kbyte

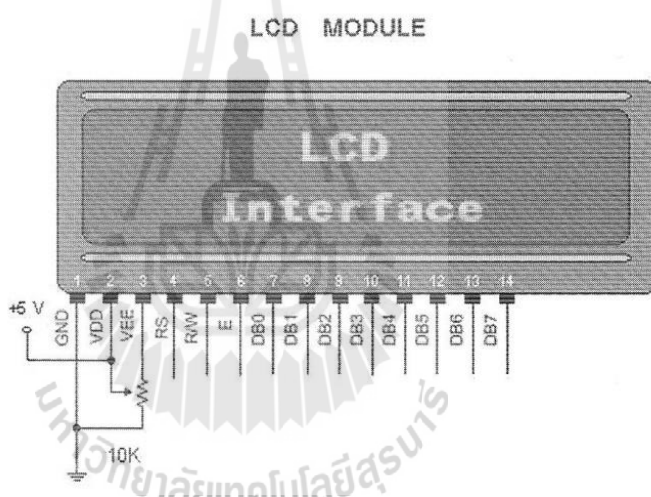
ซึ่งจะดูได้จากรูป 2.2 ที่แสดงภาพโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล 8051 เบอร์ AT89C51



รูปที่ 2.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์

2.2 หน่วยแสดงผล LCD (LCD Modules)

ตัวอุปกรณ์ที่เรียกว่า LCD Module (Liquid Crystal Display) เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ที่ปัจจุบันจะนิยมนำมาใช้งานด้านการแสดงผลของโครงการหรือสิ่งประดิษฐ์ต่างๆ ทั้งนี้ก็เพราะว่าตัวอุปกรณ์ LCD จะมีการแสดงผลที่ดูสวยงาม แสดงผลได้ทั้งตัวเลข ตัวอักษร หรือรูปภาพกราฟิก เช่น ในอุปกรณ์เครื่องคิดเลข, วิทยุเทปดีครดยนต์, เครื่องมือวัดต่างๆ เป็นต้น ตัวอุปกรณ์ LCD Module ที่นำมาใช้งานการเชื่อมต่อกับไมโครคอนโทรลเลอร์ MCS51 ของระบบการบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายนี้ จะมีขนาด 16 x2 (2 บรรทัด บรรทัดละ 16 ตัวอักษร) ข้อมูลรับ-ส่งขนาด 8 บิต สังเกตได้จากรูป 2.3



รูปที่ 2.3 LCD Module

การที่เรานำ LCD Module มาใช้ในการเชื่อมต่อกับบอร์ดเพื่อที่จะให้ตัวคอนโทรลเลอร์สามารถแสดงผลให้เราดูเมื่อเราอยู่ที่ชุดอุปกรณ์คอนโทรลเลอร์โดยไม่ต้องเชื่อมต่อกับคอมพิวเตอร์ และจากรูปที่ 2.3 เราจะอธิบายการทำงานของขาต่างๆของ LCD Module ได้ดังต่อไปนี้

- ขาที่ 1, GND เป็นขาที่ต่อ GND
- ขาที่ 2, VDD เป็นขาที่ต่อไฟเลี้ยง +5 V
- ขาที่ 3, เป็นขารับค่าแรงดันไฟฟ้าเพื่อปรับความเข้มสว่างหน้าจอ LCD

- ขาที่ 4, RS (Register Select) เป็นขาอินพุตรับค่าข้อมูล
RS=1 เป็นการรับค่าข้อมูลเพื่อส่งออกไปแสดงผลหน้าจอ LCD
RS=0 เป็นการรับค่าข้อมูลเพื่อเป็นคำสั่งควบคุมการทำงานของตัว LCD Module
- ขาที่ 5, R/W (Read/Write) เป็นขาอินพุตรับค่าข้อมูล
R/W = 0 เป็นการเลือกเขียนข้อมูลเข้าไปที่ตัว LCD Module
R/W = 1 เป็นการเลือกอ่านข้อมูลจากตัว LCD Module
- ขาที่ 6, E (Enable) เป็นขาอินพุตรับสัญญาณพัลส์อินาเปิดความกว้างของพัลส์ไม่น้อยกว่า 2 ms
- ขาที่ 7-14, D0-D7 (Data Bus) เป็นขารับ-ส่งข้อมูลขนาด 8 บิต

2.3 ไอซีฐานเวลา RTC (Real Time Clock)

ในที่นี้อุปกรณ์ที่เราใช้เป็นตัวไอซีฐานเวลา คือ DS 1307 ซึ่งเป็นอุปกรณ์ที่ใช้สร้างฐานเวลาจริงให้กับไมโครคอนโทรลเลอร์ โดยการทำงานผ่านระบบบัส I²C ซึ่งเป็นการสื่อสารข้อมูลในลักษณะอนุกรมที่ใช้สายสัญญาณในการรับส่งข้อมูลเพียงสองเส้น คือ สายสัญญาณนาฬิกาใช้ในการกำหนดจังหวะการสื่อสารข้อมูลและสายสัญญาณข้อมูลใช้ในการรับส่งข้อมูล

DS1307 จัดการเชื่อมต่อในแบบบัส I²C โดยทำงานเป็นอุปกรณ์สเลฟเสมอ ส่วนประกอบหลักที่สำคัญคือ วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง มีการเก็บค่าของเวลาไว้ในหน่วยความจำอนโวลไทม์แรม ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์ (ดูได้จากรูปที่ 2.4) วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสถานะของไอซี หากไฟเลี้ยงต่ำกว่า $1.25 \times V_{BAT}$ ก็จำควบคุมให้ DS1307 หยุดทำงาน ทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งานต้องระมัดระวังอย่าให้ไฟเลี้ยงต่ำกว่า $1.25 \times V_{BAT}$ หรือประมาณ 3.75 V ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที แต่วงจรสร้างเวลายังคงทำงานเพื่อให้ค่าเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป

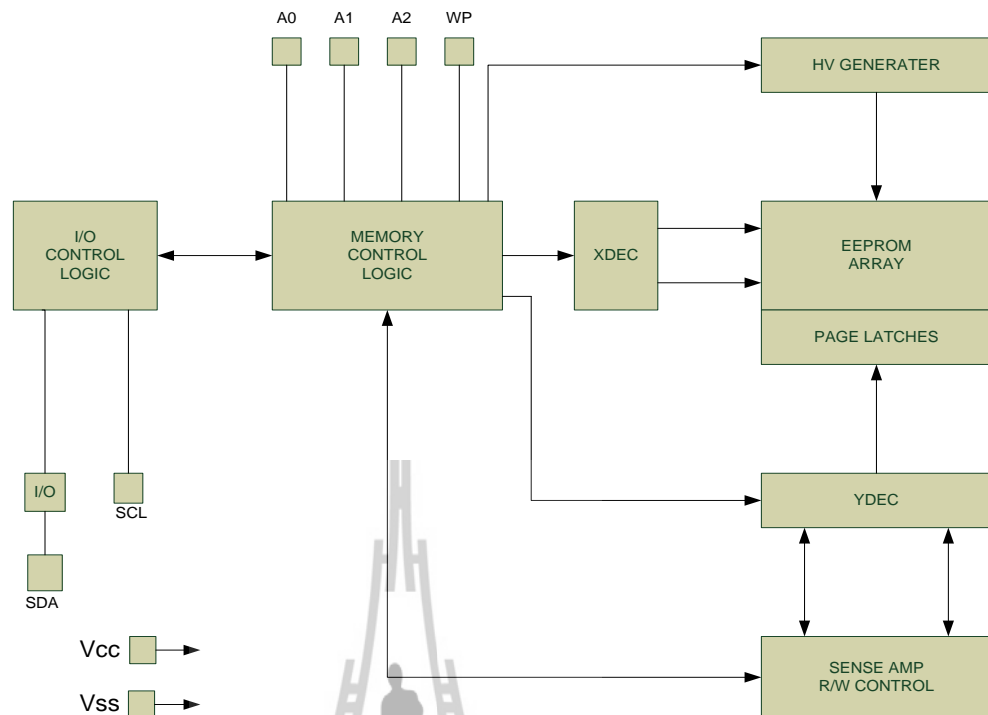
\$00	วินาที	นาฬิกา	ชั่วโมง	วัน	วันที่	เดือน	ปี	รีจิสเตอร์ควบคุม	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0	ค่าของข้อมูล		
									CH	ข้อมูลนาฬิกา (หลักสิบ)			ข้อมูลนาฬิกา (หลักหน่วย)				00-59		
									X	ข้อมูลนาฬิกา (หลักสิบ)			ข้อมูลนาฬิกา (หลักหน่วย)				00-59		
									X	12 ชั่วโมง	ชั่วโมง (หลักสิบ)	ข้อมูลชั่วโมง (หลักสิบ)	ข้อมูลชั่วโมง (หลักหน่วย)				01-12 00-23		
\$07										24 ชั่วโมง	AM/PM								
\$08									X	X	X	X	X					ข้อมูลวันในสัปดาห์	1-7
									X	X	ข้อมูลวันที่ (หลักสิบ)		ข้อมูลวันที่ (หลักหน่วย)				01-28/29 01-30 01-31		
									X	X	X	ข้อมูลเดือน (หลักสิบ)	ข้อมูลเดือน (หลักหน่วย)				01-12		
									ข้อมูลปี (หลักสิบ)			ข้อมูลปี (หลักหน่วย)				00-99			
\$3F									OUT	X	X	SQWE	X	X	RS1	RS0			

รูป 2.4 แสดงภาพการจัดสรรหน่วยความจำใน DS 1307

สำหรับโหมดการทำงานของ DS 1307 จะมีด้วยกัน 2 โหมดคือ โหมดเขียนข้อมูล และโหมดอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูล

2.4 ไอซีหน่วยความจำ EEPROM (Electrically Erasable Programmable Read-Only Memory)

หน่วยความจำแบบ EEPROM ก็คือหน่วยความจำรวม จัดเป็นหน่วยความจำถาวร ที่ผู้ใช้สามารถลบหรือแก้ไขหรือเขียนซ้ำข้อมูลที่บรรจุอยู่ภายในได้ และสามารถกระทำซ้ำได้หลายครั้งด้วยสัญญาณไฟฟ้า และยังสามารถเก็บรักษาข้อมูลภายในตัวไว้ได้ถึงแม้ว่าจะไม่มีการจ่ายไฟเลี้ยงให้กับตัวหน่วยความจำก็ตาม โดยในการเชื่อมต่อตัวหน่วยความจำนี้กับตัวไมโครคอนโทรลเลอร์จะใช้การเชื่อมต่อแบบอนุกรม I²C และจะใช้สัญญาณการเชื่อมต่ออย่างน้อยเส้นซึ่งภาพโครงสร้างของ EEPROM แสดงไว้ในภาพที่ 2.5



รูปที่ 2.5 แสดงไดอะแกรม โครงสร้างของหน่วยความจำ EEProm 24xxx

2.5 ตัวจัดการข้อมูลผ่านเครือข่าย (Ethernet Modules)

ชุดอุปกรณ์การจัดการข้อมูลผ่านเครือข่าย หรือ Ethernet Modules ในที่นี้จะใช้ชุดอุปกรณ์ที่มีชื่อว่า EZL-50L ซึ่งเป็นชุดเป็นผลิตภัณฑ์ ในกลุ่มผลิตภัณฑ์ ezTCP จะทำหน้าที่ในการจัดหา TCP/IP สำหรับการสื่อสารผ่าน Ethernet ซึ่ง EZL-50L จะส่งข้อมูลจาก Serial port ไปยัง LAN หลังจากนั้นจะเป็น กระบวนการของ TCP/IP ซึ่งทำให้เราไม่จำเป็นต้องจัดหาพอร์ตของ TCP/IP เอง เพราะอุปกรณ์ชิ้นนี้จะทำการประมวลผลจัดการในรูปแบบโปรโตคอล TCP/IP ซึ่งก็คือรับค่าข้อมูลเข้ามาไม่ว่าจากทางเครือข่ายที่ส่งให้กับตัวคอนโทรลเลอร์หรือจะเป็นตัวคอนโทรลเลอร์ส่งให้กับทางเครือข่าย โดยที่อุปกรณ์ตัวนี้จะทำการประกอบเลเยอร์ต่างๆ ให้กับตัวข้อมูลที่ทำกรรับส่งคือถ้าจะให้เข้าใจง่ายๆ ก็คืออุปกรณ์ชิ้นจะเป็นตัวที่สร้าง Header ในชั้นเลเยอร์ต่างๆ ให้กับข้อมูลนั่นเอง

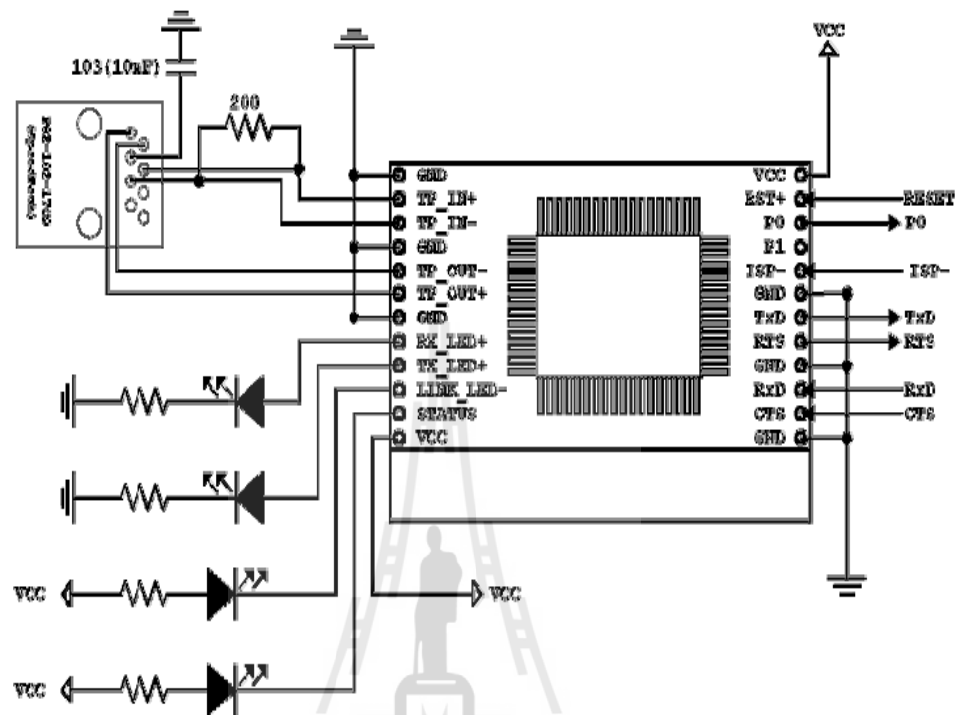
โครงสร้างโดยรวมของชุดอุปกรณ์ EZL-50L จะมีลักษณะดังรูป 2.6 ซึ่งรายละเอียดของข้อมูลจำเพาะของตัว EZL-50L จะดูได้จากตารางที่ 2.1



รูป 2.6 แสดงลักษณะโดยรวมของ EZL-50L

ตารางที่ 2.1 แสดงรายละเอียดข้อมูลจำเพาะของ EZL-50L

Power	Input Voltage	5V ($\pm 10\%$)
	Current	72mA typical
Dimension	50mm x 32mm x 11mm	
Weight	about 10g	
Interface	Serial	2mm pitch 1x12 connector
	Network	2mm pitch 1x12 connector
Serial Port	UART (1200bps ~ 115200bps)	
Network	10Base-T	
Protocols	TCP, UDP, IP, ICMP, ARP, DHCP, PPPoE	
Communication Mode	T2S	TCP Server Mode
	COD	TCP Client Mode
	ATC	TCP Server/Client Mode (AT command emulation)
Utilities	U2S	UDP
	ezConfig	Configuration utility via LAN
	ezterm	Socket test utility
	hotflash	Firmware download utility via TFTP



รูปที่ 2.7 Ethernet Interface

เนื่องจาก EZL-50L จะมี 10Base-T Ethernet Interface ซึ่งจำเป็นที่จะต้องต่อวงจรเพิ่มขึ้นมาเพื่อให้เป็น Ethernet Interface เพื่อให้สามารถส่งข้อมูลผ่านสาย RJ-45 ที่ต่อเข้ากับเครื่องแปลงพัลส์ ส่วนของสถานะการอินเทอร์เฟสอินพุท/เอาต์พุทเมื่อทำการเชื่อมต่อ จะดูได้จากตารางที่ 2.2

ตารางที่ 2.2 I/O Interface

Mode	Name	Status	Description
Normal mode	PWR	ON	Power is supplied
	STS	Blinks in every second	IP address is assigned Repetition of HIGH/LOW for 500ms
		Blinks once after 4 times short blinking	IP is not allocated. Repetition of [after repetition 4 times for 150ms, HIGH during 850ms]
		ON	During TCP connection – LOW
	LINK	ON	When connected to LAN – LOW
	RXD	Blinks	Data are being received from LAN – HIGH
	TXD	Blinks	Data are being transmitted to LAN – HIGH
	P0	ON	During TCP connected– LOW
		OFF	During TCP disconnected – HIGH
	P1	OFF	During the EZL-50L receive data from serial – HIGH For interfacing RS485 chip(TXDE)
ISP mode	PWR	ON	Power is supplied
	STS	Blinks rapidly	ISP Mode – Repetition HIGH/LOW for 50ms
	LINK	ON	When connected to LAN – LOW

ในส่วนของรายละเอียดปลีกย่อยของอุปกรณ์ต่างๆ ที่กล่าวมาทั้งหมดรวมถึงการทดสอบการใช้งานของอุปกรณ์จะขอออกไปอธิบายไว้ในภาคผนวก

2.6 ทฤษฎีของ TCP/IP

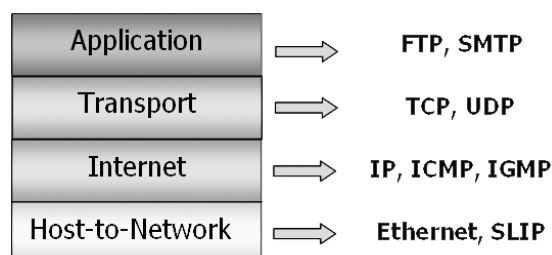
TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

ชุดโปรโตคอลนี้ได้รับการพัฒนามาตั้งแต่ปี 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเทอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน สามประการคือ

1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน
2. ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่นในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ
3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูล และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (Voice) และข้อมูล (data)

ในระบบชุดโปรโตคอล TCP/IP จะมีโครงสร้างประกอบไปด้วยข้อมูลเป็นเลขอร์ ดังรูป 2.8 ซึ่งจะสามารถอธิบายได้ดังนี้



รูปที่ 2.8 โครงสร้าง TCP/IP

2.6.1. ชั้นโฮสต์-เครือข่าย (Host-to-Network Layer)

โพรโตคอลสำหรับการควบคุมการสื่อสารในชั้นนี้เป็นสิ่งที่ไม่มีการกำหนดรายละเอียดอย่างเป็นทางการ

หน้าที่หลักคือการรับข้อมูลจากชั้นสื่อสาร IP มาแล้วส่งไปยังโหนดที่ระบุไว้ในเส้นทางเดินข้อมูลทางด้านผู้รับก็จะทำงานในทางกลับกัน คือรับข้อมูลจากสายสื่อสารแล้วนำส่งให้กับโปรแกรมในชั้นสื่อสาร

2.6.2. ชั้นสื่อสารอินเทอร์เน็ต (The Internet Layer)

ใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพ็กเก็ต (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (Connectionless) หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่า แพ็กเก็ต (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ หากว่ามี การส่งแพ็กเก็ตออกมาเป็นชุด โดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่าย แพ็กเก็ตแต่ละตัวในชุดนี้ก็จะ เป็นอิสระแก่กันและกัน ดังนั้น แพ็กเก็ตที่ส่งไปถึงปลายทางอาจจะไม่เป็นไปตามลำดับก็ได้ ซึ่งในเลเยอร์นี้จะมีข้อมูลสำคัญอยู่ก็คือ IP

IP เป็นโพรโตคอลในระดับเน็ตเวิร์กเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเก็ต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบดาต้าแกรม (datagram) เพื่อรองรับการส่งข้อมูลระดับ data link ที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำ IP ไปใช้บนโพรโตคอลอื่นได้หลากหลาย เช่น Ethernet ,Token Ring หรือ Apple Talk

การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบ connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆครั้งของการส่งข้อมูล 1 ดาต้าแกรม โดยจะไม่ทราบถึงข้อมูลดาต้าแกรมที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลใน 1 ดาต้าแกรม อาจจะมีการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นดาต้าแกรมเดิมเมื่อถึงปลายทาง

2.6.3. ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

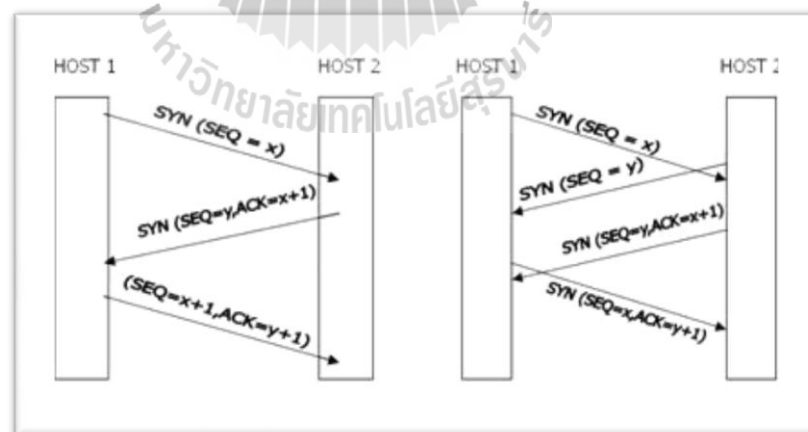
แบ่งเป็นโพรโตคอล 2 ชนิดตามลักษณะ ลักษณะแรกเรียกว่า Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบ Byte stream ที่ไว้วางใจได้โดยไม่มีข้อผิดพลาด ข้อมูลที่

มีปริมาณมากจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า message ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต ทางฝ่ายผู้รับจะนำ message มาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิม TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่ง ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย โพรโตคอลการนำส่งข้อมูลแบบที่สองเรียกว่า UDP (User Datagram Protocol) เป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) มีการตรวจสอบความถูกต้องของข้อมูล แต่จะไม่มีการแจ้งกลับไปยังผู้ส่ง จึงถือได้ว่าไม่มีการตรวจสอบความถูกต้องของข้อมูล ใดๆก็ตาม วิธีการนี้มีข้อดีในด้านความรวดเร็วในการส่งข้อมูล จึงนิยมใช้ในระบบผู้ให้และผู้ให้บริการ (client/server system) ซึ่งมีการสื่อสารแบบ ถาม/ตอบ (request/reply) นอกจากนั้นยังใช้ในการส่งข้อมูลประเภทภาพเคลื่อนไหวหรือการส่งเสียง (Voice) ทางอินเทอร์เน็ต

โดยที่ในที่นี้จะขอก้าวแค่รายละเอียดของ การสื่อสารแบบ TCP เนื่องจากการทำอุปกรณ์นี้จะใช้การรับส่งข้อมูลแบบ TCP/IP

- TCP : (Transmission Control Protocol)

จะอยู่ใน ชั้น Transport Layer ทำหน้าที่ในจัดการและควบคุมการรับส่งข้อมูล ซึ่งมีความสามารถและรายละเอียดมากกว่า UDP โดยค่าต่ำแกรมของ TCP จะมีความสัมพันธ์ต่อกัน และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (Reliable) และมีการสื่อสารอย่างเป็นทางการ (Connection-Oriented) โดยจะมีไดอะแกรมการสื่อสารดังรูป 2.9



รูป 2.9 ไดอะแกรมการสื่อสารของ TCP

โดยจากรูปที่ 2.9 จะสามารถอธิบายได้ดังนี้

- เมื่อเซกเมนต์ CONNECT (SYN = "1" และ ACK = "0") เดินทางมาถึง Entity TCP ที่โฮสต์ปลายทางจะค้นหาโปรเซสตามหมายเลขพอร์ตที่กำหนดในเขตข้อมูล Destination port ซึ่งถ้าหากไม่พบก็จะตอบปฏิเสธด้วยเซกเมนต์ที่มี RST = "1" กลับไปยังผู้ส่งเซกเมนต์ CONNECT ของผู้ส่งจะถูกส่งต่อไปยังโปรเซส ตามพอร์ตที่ระบุซึ่งอาจจะตอบรับหรือตอบปฏิเสธก็ได้ ถ้าโปรเซสนั้นต้องการสื่อสารด้วยก็จะส่งเซกเมนต์ตอบรับกลับไป (ดังรูป 2.9 ทางซ้าย) แสดงลำดับขั้นตอนการส่ง TCP เซกเมนต์ในการสร้างการเชื่อมต่อในสถานะปกติระหว่างผู้ส่งและผู้รับ

ในกรณีที่โฮสต์สองแห่งพยายามสร้างการเชื่อมต่อระหว่างซ็อกเก็ตคู่เดียวกันจะเกิดเป็นลำดับขั้นตอนแสดงในรูป ที่ 2.9(ขวา) ผลสุดท้ายจะมีการเชื่อมต่อเกิดขึ้นเพียงหนึ่งช่องทางเท่านั้นเนื่องจากการเชื่อมต่อในแต่ละช่องทางจะถูกกำหนดขึ้นโดยใช้หมายเลขซ็อกเก็ตผู้ส่งและผู้รับ ถ้าการเชื่อมต่อลำดับแรกสำเร็จก็จะถูกบันทึกไว้ในตารางการสื่อสาร เช่น (x, y) ถ้าการเชื่อมต่อลำดับที่สองสำเร็จในเวลาต่อมา ข้อมูลนี้ก็จะถูกบันทึกไว้ที่เดียวกันคือ (x, y)

2.6.4. ชั้นสื่อสารการประยุกต์ (Application Layer)

มีโพรโตคอลสำหรับสร้างจอร์มินัลเสมือน เรียกว่า TELNET โพรโตคอลสำหรับการจัดการเพิ่มข้อมูล เรียกว่า FTP และโพรโตคอลสำหรับการให้บริการจดหมายอิเล็กทรอนิกส์ เรียกว่า SMTP โดยโพรโตคอลสำหรับสร้างจอร์มินัลเสมือนช่วยให้ผู้ใช้สามารถติดต่อกับเครื่องโฮสต์ที่อยู่ไกลออกไปโดยผ่านอินเทอร์เน็ต และสามารถทำงานได้เสมือนกับว่ากำลังนั่งทำงานอยู่ที่เครื่องโฮสต์นั้น โพรโตคอลสำหรับการจัดการเพิ่มข้อมูลช่วยในการคัดลอกเพิ่มข้อมูลมาจากเครื่องอื่นที่อยู่ในระบบเครือข่ายหรือส่งสำเนาเพิ่มข้อมูลไปยังเครื่องใดๆก็ได้ โพรโตคอลสำหรับการให้บริการจดหมายอิเล็กทรอนิกส์ช่วยในการจัดส่งข้อความไปยังผู้ใช้ในระบบ หรือรับข้อความที่มีผู้ส่งเข้ามา

2.7 ความรู้เบื้องต้นเกี่ยวกับ Visual Basic (VB)

โปรแกรม Visual Basic (VB) เป็นโปรแกรมสำหรับพัฒนาโปรแกรมประยุกต์ที่กำลังเป็นที่ นิยมใช้อยู่ในปัจจุบัน โปรแกรม Visual Basic เป็นโปรแกรมที่ได้เปลี่ยนรูปแบบการเขียนโปรแกรมใหม่ โดยมีชุดคำสั่งมาสนับสนุนการทำงาน มีเครื่องมือต่าง ๆ ที่เรียกกันว่า คอนโทรล(Controls) ไว้สำหรับช่วยในการออกแบบโปรแกรม โดยเน้นการออกแบบหน้าจอแบบกราฟิก หรือที่เรียกว่า Graphic User Interface (GUI) ทำให้การจัดรูปแบบหน้าจอเป็นไปได้ง่าย และในการเขียนโปรแกรมนั้นจะเขียนแบบ Event - Driven Programming คือ โปรแกรมจะทำงานก็ต่อเมื่อเหตุการณ์ (Event) เกิดขึ้น ตัวอย่างของเหตุการณ์ได้แก่ ผู้ใช้เลื่อนเมาส์ ผู้ใช้กดปุ่มบนคีย์บอร์ด ผู้ใช้กดปุ่มเมาส์ เป็นต้น

เครื่องมือ หรือ คอนโทรล ต่าง ๆ ที่ Visual Basic ได้เตรียมไว้ให้ ไม่ว่าจะเป็น Form Textbox Label ฯลฯ ถือเป็นวัตถุ (Object ในที่นี้ขอใช้คำว่า ออบเจกต์) นั้นหมายความว่า ไม่ว่าจะเป็นเครื่องมือใด ๆ ใน Visual Basic จะเป็นออบเจกต์ทั้งสิ้น สามารถที่จะควบคุมการทำงาน แก้ไขคุณสมบัติของออบเจกต์นั้นได้โดยตรง ในทุกๆ ออบเจกต์จะมีคุณสมบัติ (properties) และเมธอด (Methods) ประจำตัว ซึ่งในแต่ละออบเจกต์ อาจจะมีคุณสมบัติและเมธอดที่เหมือน หรือต่างกันก็ได้ขึ้นอยู่กับชนิดของออบเจกต์

ในการพัฒนาโปรแกรมประยุกต์ด้วย Visual Basic การเขียนโค้ดจะถูกแบ่งออกเป็นส่วนๆ เรียกว่า โพรซีเจอร์ (procedure) แต่ละโพรซีเจอร์จะประกอบไปด้วย ชุดคำสั่งที่พิมพ์เข้าไปแล้ว ทำให้คอนโทรลหรือออบเจกต์นั้น ๆ ตอบสนองการกระทำของผู้ใช้ ซึ่งเรียกว่าการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming-OOP) แต่ตัวภาษา Visual Basic ยังไม่ถือว่าเป็นการเขียนโปรแกรมแบบ OOP อย่างแท้จริง เนื่องจากข้อจำกัดหลายๆ อย่างที่ Visual Basic ไม่สามารถกระทำได้นั่นเอง

ทำความรู้จักกับ winsock

การทำงานเกี่ยวกับเครือข่ายคอมพิวเตอร์นั้นเป็นเรื่องค่อนข้างซับซ้อน จนทำให้เราสร้างข้อกำหนดต่างๆ ขึ้นมาเพื่อให้วิธีการสื่อสารระหว่างกันเป็นไปได้โดยราบรื่นแล้วก็ทำให้เกิดเป็นโพรโตคอลมากมายหลายชนิดมาใช้งานร่วมกับเครือข่ายซึ่งเรามีการแบ่งโพรโตคอลต่างๆ ออกเป็นระดับชั้น เพื่อให้ง่ายต่อการใช้งาน และง่ายต่อการทำความเข้าใจ

สำหรับ Visual Basic แล้ว การสื่อสารที่ใช้งานอินเทอร์เน็ตเป็นเรื่องที่น่าสนใจ เพราะหมายถึงเราสามารถแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์ต่างๆ ที่เชื่อมต่อกันในอินเทอร์เน็ตได้อย่างไม่ยากเย็น โดยการใช้ ActiveX Control ที่ทรงประสิทธิภาพที่เตรียมไว้ให้ คือ Winsock Control

ระบบปฏิบัติการยูนิกซ์จะมีความสามารถอย่างหนึ่งที่เรียกว่า Socket ซึ่งก็คือ การที่โปรแกรมต่างๆ สามารถสื่อสารข้อมูลระหว่างกันได้ โดยไม่จำเป็นต้องรันอยู่บนคอมพิวเตอร์เครื่องเดียวกัน

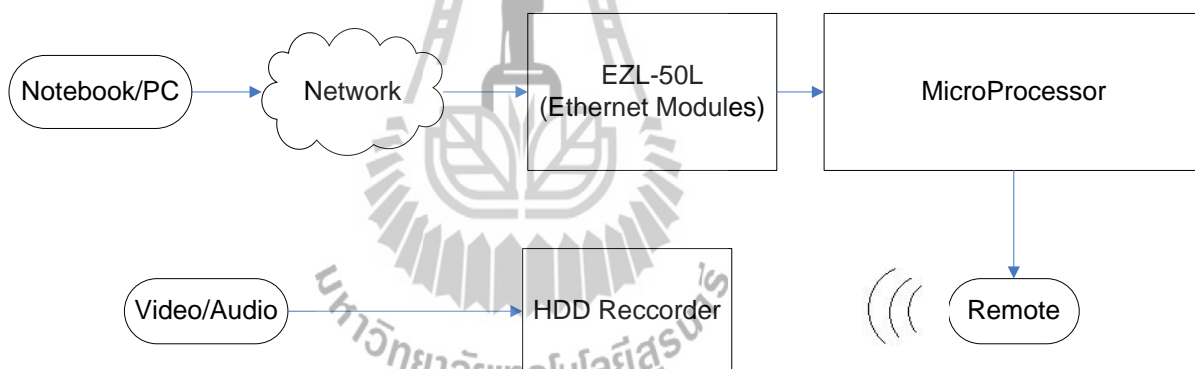
Socket เองก็มาจากแนวความคิด โคลเอ็น/เซิร์ฟเวอร์อันแสนจะคลาสสิก โดยส่วนที่ทำหน้าที่เป็นเซิร์ฟเวอร์ จะสร้าง Socket (ซึ่งถ้าแปลตามตัวก็จะหมายถึงช่องเสียบ หรือรูเสียบ) จะมีจำนวนเท่าใดก็ได้แล้วแต่ จากนั้นเมื่อมีโคลเอ็นต์ที่ต้องการสื่อสารด้วยก็จะติดต่อมาที่ Socket ที่เซิร์ฟเวอร์ได้เตรียมไว้ให้ เมื่อเชื่อมต่อกันเสร็จก็พร้อมจะแลกเปลี่ยนข้อมูลระหว่างกันได้โดยเป็นการสื่อสารแบบ 2 ทางชนิดเต็มรูปแบบ (2 Way Full-Duplex)

บทที่ 3

การออกแบบโครงงาน

ในบทที่ผ่านมาเราได้ทำความรู้จักอุปกรณ์และหลักการสำคัญที่ใช้ในระบบคอนโทรลเลอร์ รวมไปถึงทฤษฎีที่เกี่ยวข้องกับการทำงานในระบบนี้ โดยในการที่จะออกแบบและพัฒนาาระบบควบคุมการบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายอัตโนมัติ เราจะต้องเข้าใจถึงการทำงานโดยรวมของระบบนี้ ซึ่งในแต่ละส่วนก็ประกอบไปด้วยอุปกรณ์ต่างๆที่ต้องนำมาประกอบกันเพื่อให้สามารถทำงานได้ตามต้องการ ในขณะที่เดียวกันก็ต้องมีซอฟต์แวร์ที่เป็นตัวควบคุมการทำงานได้ตามต้องการ โดยจะสามารถอธิบายการออกแบบของโครงงานได้ดังต่อไปนี้

3.1 การออกแบบทางฮาร์ดแวร์



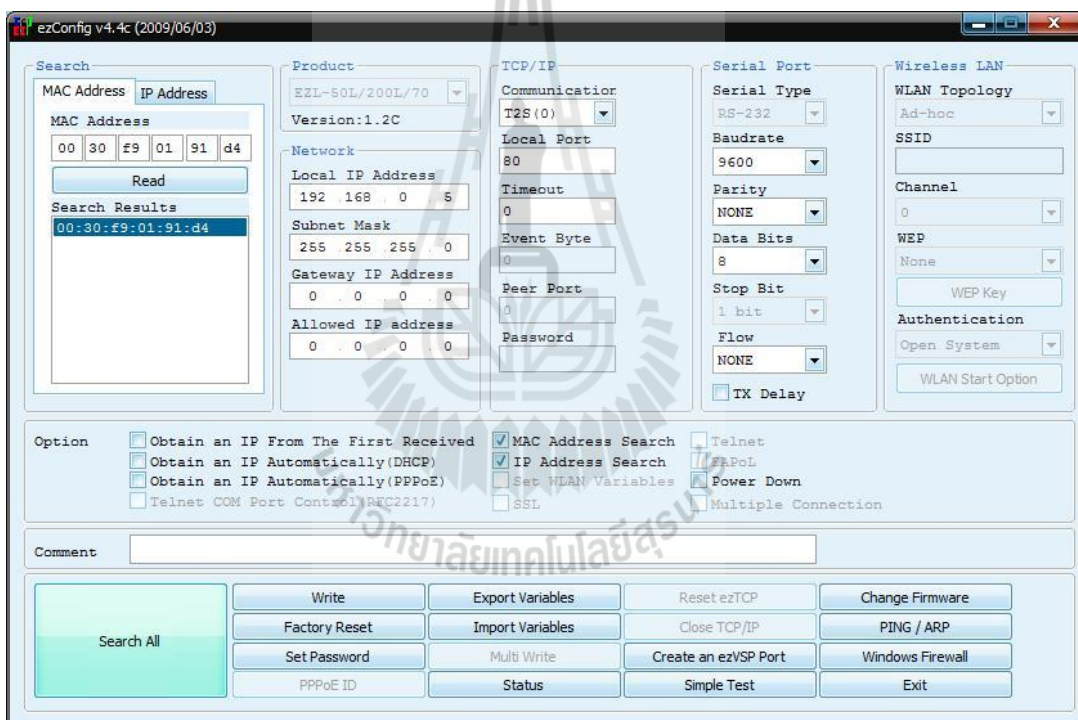
รูปที่ 3.1 ฟังก์ชันของโครงงาน

จากไดอะแกรมของโครงงานจะเห็นได้ว่า จะมีคำสั่งที่ส่งมาจากคอมพิวเตอร์โดยผ่านเครือข่ายไปยังตัวอุปกรณ์ EZL-50L จากนั้นข้อมูลจะถูกส่งไปยังไมโครคอนโทรลเลอร์เพื่อทำการประมวลผล โดยตัวไมโครคอนโทรลเลอร์นี้จะมีการเชื่อมต่อกับรีโมทซึ่งสามารถควบคุมให้รีโมททำงานสั่งตัวอุปกรณ์ Hard disk Recorder ได้ โดยการออกแบบฮาร์ดแวร์จะแบ่งออกเป็นสองส่วนหลักๆคือ

3.1.1 EZL-50L

เป็นอุปกรณ์ที่ทำหน้าที่แปลง Serial ไปเป็น TCP/IP Protocol และจัดหา TCP/IP สำหรับการสื่อสารผ่าน Ethernet ซึ่งอุปกรณ์นี้สามารถส่งข้อมูลจาก serial port ไปยัง Local Area Network ได้ และมีขั้นตอนในการกำหนด IP Address และ Port เชื่อมต่อให้กับตัวอุปกรณ์สามารถทำได้โดยใช้โปรแกรม ezConfig

ขั้นแรกให้ทำการ run โปรแกรม ezConfig จากนั้น คลิกที่ปุ่ม Search All ที่อยู่ในหน้าต่างของ ezConfig ซึ่ง โปรแกรม ezConfig ก็จะทำการค้นหา ezTCPs ที่มีอยู่ทั้งหมดใน local network เมื่อทำการค้นหา ezTCP ตัว MAC address ของ ezTCP จะแสดงขึ้นมาในหน้าต่างของ [Search Results] (MAC address จะระบุไว้ข้างใต้ของกล่องของผลิตภัณฑ์)

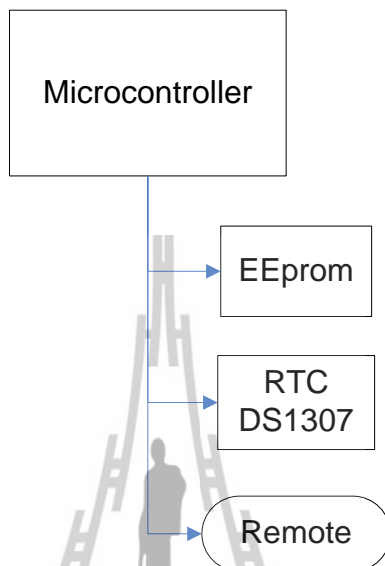


รูปที่ 3.2 แสดงลักษณะของโปรแกรม ezConfig v.4.4c

จากรูปที่ 3.2 เราจะสามารถเปลี่ยน IP Address และ Port ติดต่อได้โดยการใส่ค่าลงในโปรแกรมได้โดยตรง ในโครงการนี้ผู้จัดทำได้กำหนดให้ IP Address ของอุปกรณ์เป็น 192.168.0.5 และ Port ติดต่อคือ 80 จากนั้นคลิกที่ปุ่ม Write เป็นอันเสร็จสิ้นการกำหนดค่าของอุปกรณ์

3.1.2 MicroProcessor

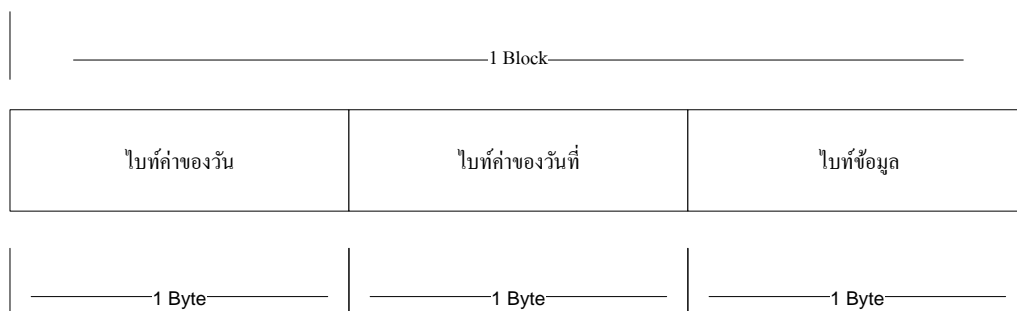
ประกอบไปด้วย หน่วยความจำ EEPROM , อุปกรณ์ฐานเวลา(Realtime Clock) และ วงจรการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับรีโมต



รูปที่ 3.3 แผนภาพแสดง ส่วนประกอบของ MicroProcessor

1) การออกแบบจัดการกับหน่วยความจำ EEPROM

สำหรับหน่วยความจำ EEPROM จะเป็นตัวที่จัดเก็บโปรแกรมคำสั่งต่างๆ ที่เราได้ทำการป้อนไว้ให้ ซึ่งก็คือจะเป็นส่วนที่ทำการบันทึกการตั้งวันเวลาที่เราจะให้บันทึกหรือหยุดบันทึกสัญญาณ Video และ Audio โดยการทำงานของ EEPROM ในโครงการนี้ เราจะทำการเขียนข้อมูลเป็นบล็อก บล็อกละ 3 ไบต์ ยกตัวอย่างเช่นการเขียนข้อมูลการตั้งวันการบันทึกในส่วนของไบต์แรกเราจะทำการเขียนค่าของวัน (อาทิตย์ -เสาร์) สำหรับไบต์ที่สองจะเป็นค่าของวันที่ และไบต์ที่สามจะเป็นไบต์ของข้อมูล ดังแสดงดังรูป 3.4



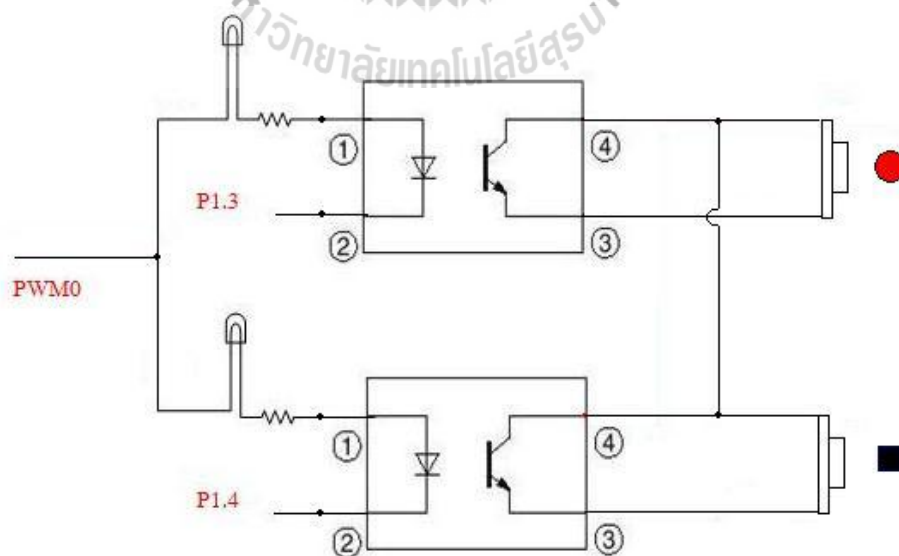
รูปที่ 3.4 ภาพแสดงบล็อกข้อมูล

2) ไอซีฐานเวลา

ไอซีฐานเวลาหรือ RTC (Real Time Clock) ในโครงงานชิ้นนี้เราจะใช้ไอซีฐานเวลาเบอร์ DS1307 โดยหน้าที่การทำงานก็คือจะเป็นตัวนับและแสดงวันเวลาให้กับระบบคอนโทรลเลอร์ เพื่อให้ระบบรับรู้ว่า ณ ขณะนั้นถึงเวลาที่ชุดคอนโทรลเลอร์จะต้องทำงานตามโปรแกรมที่ตั้งไว้หรือยัง ซึ่งเราจะทำการต่อตัว DS1307 เข้ากับไมโครคอนโทรลเลอร์ นำค่าวันและเวลาจาก DS1307 ออกแสดงผลที่หน้าจอ LCD

3) รีโมทคอนโทรล

ในการสั่งบันทึกหรือหยุดบันทึกในโครงงานนี้เราจะทำการสั่งงานผ่านรีโมทคอนโทรล สำหรับตัวรีโมทคอนโทรลนี้จะเป็นตัวรีโมทที่ให้มาในตัวฮาร์ดดิสก์ ซึ่งจะเป็นตัวควบคุมการทำงานของฮาร์ดดิสก์เรคคอร์ด โดยเราจะทำการต่อรีโมทเข้ากับพอร์ต P1.3 และ P1.4 ของไมโครคอนโทรลเลอร์เพื่อที่จะให้ไมโครคอนโทรลเลอร์ประมวลผลสั่งการในการบันทึกและหยุดบันทึก โดยในการต่อรีโมทเข้ากับไมโครคอนโทรลเลอร์เราจะกระทำผ่าน IC 817 จำนวนสองตัว สาเหตุที่เราจะต้องใช้ไอซีสองตัวเนื่องจากเราจะต้องใช้คำสั่งที่สำคัญในตัวรีโมทคอนโทรลสองคำสั่งก็คือ บันทึกและหยุดบันทึก ซึ่งไอซีตัวนี้จะทำหน้าที่เหมือนเป็นสวิสต์สั่งการ ทำให้เราไม่จำเป็นต้องไปกดที่ตัวรีโมทเอง โดยมีวงจรการต่อดังรูป 3.5



รูปที่ 3.5 การต่อวงจรของ IC817 กับตัวรีโมท

จากวงจรสายที่ต่อจากขาที่ 4 ของ IC817 ตัวบน จะต่อเข้ากับปุ่ม Rec ของตัวรีโมทซึ่งทำหน้าที่ในการสั่งบันทึก ส่วนสายที่ต่อออกจากขาที่ 4 ของ IC817 ตัวล่าง จะต่อเข้ากับปุ่ม Stop ทำหน้าที่ในการสั่งหยุดบันทึก ส่วนขาหนึ่งและขาสองของไอซีต่อเข้าที่พอร์ตในบอร์ดคอนโทรลเลอร์ตามรูปที่ 3.5

3.2 การออกแบบการทำงานผ่าน Visual Basic

3.2.1 การสร้างแอปพลิเคชันจาก Winsock Control

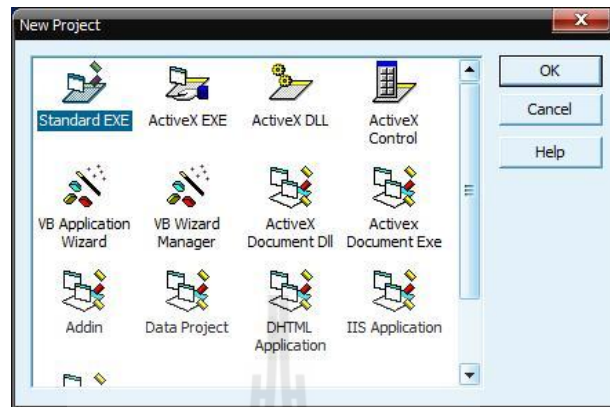
การทำงานเกี่ยวกับเครือข่ายคอมพิวเตอร์นั้นเป็นเรื่องที่ค่อนข้างซับซ้อน จนทำให้เราสร้างข้อกำหนดต่างๆ ขึ้นมาเพื่อให้การสื่อสารระหว่างกันเป็นไปได้โดยราบรื่น และยังทำให้เกิดเป็นโพรโตคอลมากมายหลายชนิดมาใช้งานร่วมกับเครือข่ายซึ่งเรามีการแบ่งโพรโตคอลต่างๆ ออกเป็นระดับชั้น เพื่อให้ง่ายต่อการใช้งาน และง่ายต่อการทำความเข้าใจ

สำหรับ Visual Basic แล้ว การสื่อสารที่ใช้งานอินเทอร์เน็ตเป็นเรื่องที่น่าสนใจ เพราะหมายถึงเราสามารถแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์ต่างๆ ที่เชื่อมต่อกันในอินเทอร์เน็ตได้อย่างไม่ยากเย็น โดยการใช ActiveX Control ที่ทรงประสิทธิภาพที่เตรียมไว้ให้ คือ Winsock Control

ในระบบควบคุมการบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายนี้ เมื่อเราทำการติดต่อกับชุดอุปกรณ์โดยผ่านระบบเครือข่าย เราจะใช้โปรแกรม Visual Basic ช่วยให้การใช้งานทางหน้าจอคอมพิวเตอร์ของผู้เรียกใช้เพื่อให้ระบบควบคุมนี้ง่ายขึ้น โดยเราจะทำการเขียน Code ให้ Visual Basic ติดต่อกับไมโครคอนโทรลเลอร์ผ่านทาง winsock โดยใช้คำสั่ง .RemoteHost แล้วตามด้วย ไอพีแอดเดรสของบอร์ดที่เราจะติดต่อ ซึ่งในที่นี้ใช้ 192.168.0.5 จากนั้นต้องกำหนดพอร์ตในการเชื่อมต่อโดยใช้คำสั่ง .RemotePort ซึ่งใช้พอร์ต 80 และคำสั่งที่จะทำให้สามารถรับข้อมูลที่ส่งมาได้ ก็คือ .Connect นั่นเอง เราจะสามารถทราบข้อมูลที่ส่งมาคืออะไรจะต้องมีคำสั่ง .SendData (KeyAscii) ซึ่งผู้จัดทำจะอธิบายการออกแบบโดยรวม ดังต่อไปนี้

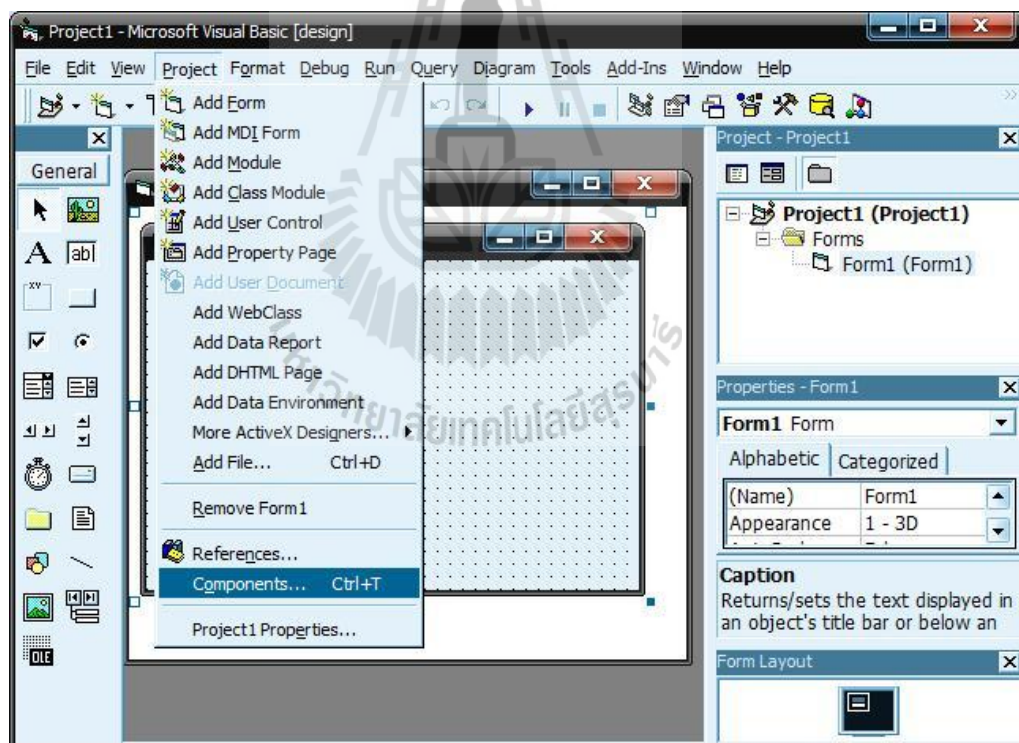
3.2.2 การออกแบบ

- 1). เข้าโปรแกรม Visual Basic 6 เลือก new project เป็น Standard EXE แล้ว คลิก OK



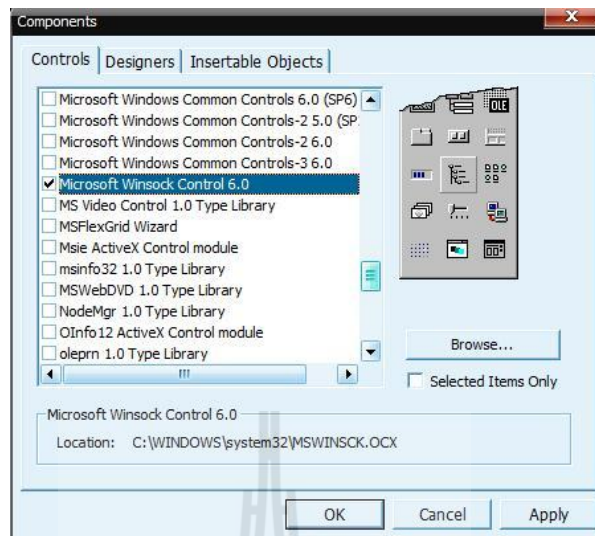
รูปที่ 3.6

- 2). เลือก Tab Project ไปที่ Components



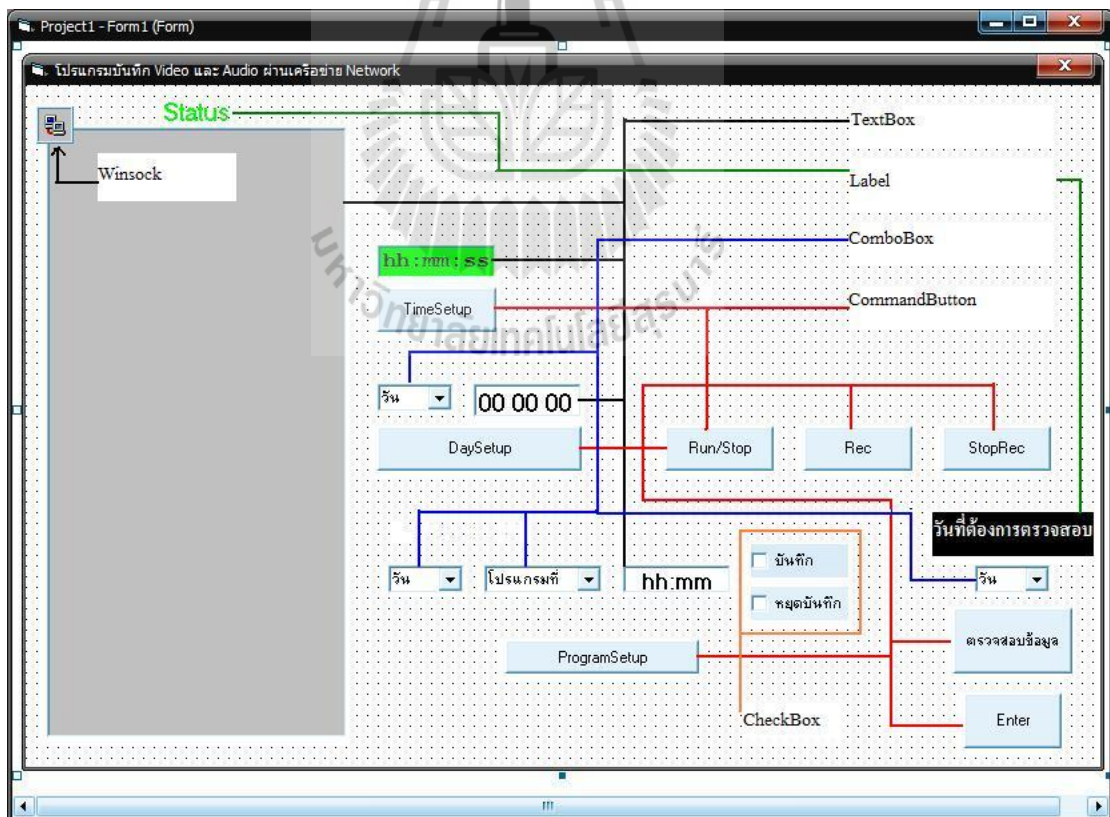
รูปที่ 3.7

3). ที่ Tab Controls เลือก Microsoft Winsock Control 6.0 คลิก OK



รูปที่ 3.8 เรียกใช้ Winsock

4). ออกแบบฟอร์มสำหรับ ระบบบันทึกสัญญาณ Video และ Audio อัตโนมัติ โดยทำงานผ่านระบบ Network ในฟอร์มดีไซน์ดังนี้

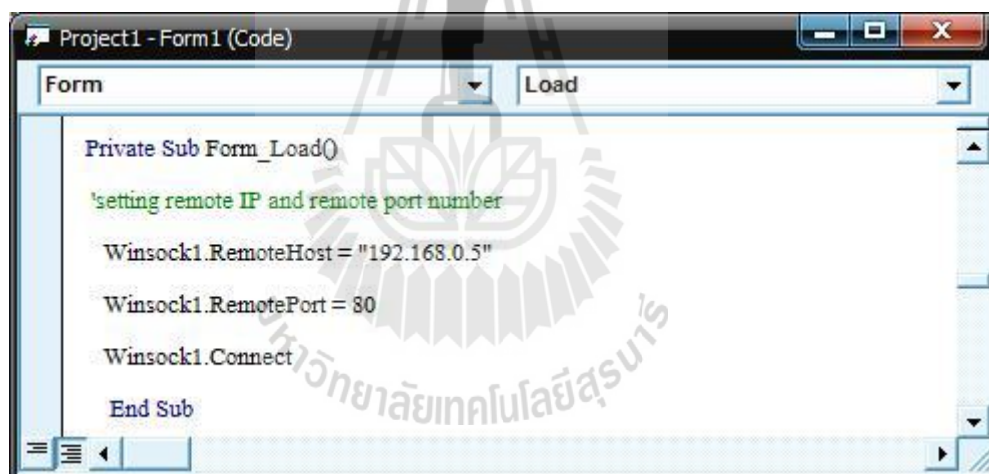


รูปที่ 3.9 Form Design

จากรูปจะเห็นได้ว่า ฟอรัมดีไซน์ข้างต้นมีการเลือกใช้ชุดคำสั่ง 6 แบบด้วยกัน ได้แก่

- TextBox (เส้นสีดำ) ซึ่งจะทำหน้าที่รับค่าที่ต้องการป้อน และเป็นตัวแสดงสถานะการทำงาน
- Label (เส้นสีเขียว) เป็นแถบอักษรหรือป้ายตัวอักษรที่เรากำหนดข้อความลงไปได้
- ComboBox (เส้นสีน้ำเงิน) จะทำหน้าที่เป็นเหมือนกล่องเลือกข้อมูลที่ใช้ต้องการ
- CommandButton (เส้นสีแดง) หรือ ปุ่มกด ซึ่งเราจะใช้ Command Button แทนคำสั่ง 1 คำสั่ง
- CheckBox ทำหน้าที่เป็นตัวเลือกให้ผู้ใช้งานโดยไม่จำเป็นต้องป้อนค่า
- Winsock Control มีหน้าที่ในการเชื่อมต่อผ่านเครือข่าย

5). เขียน Code เพื่อเรียกใช้งาน Winsock

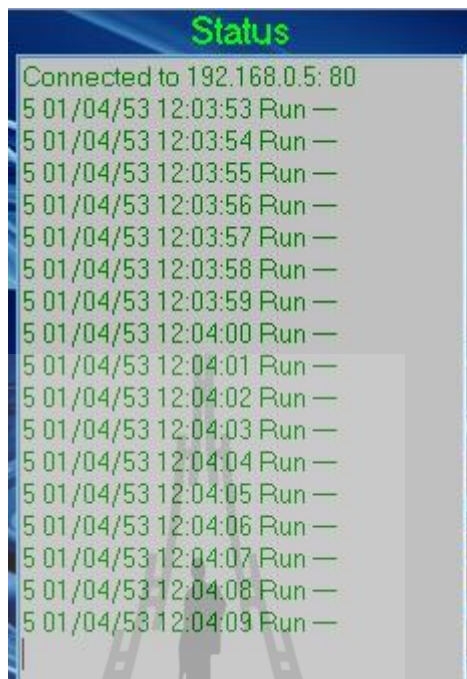


```

Project1 - Form1 (Code)
Form Load
Private Sub Form_Load()
    'setting remote IP and remote port number
    Winsock1.RemoteHost = "192.168.0.5"
    Winsock1.RemotePort = 80
    Winsock1.Connect
End Sub
  
```

รูปที่ 3.10 Code กำหนดไอพีแอดเดรสที่จะเชื่อมต่อ, กำหนดพอร์ตติดต่อ และสั่งให้เชื่อมต่อ ซึ่งโค้ดโปรแกรมอื่นๆสามารถศึกษาได้ที่ ภาคผนวก

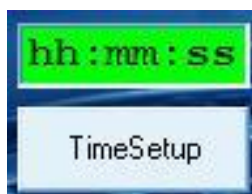
6). หลังจากเขียนโค้ดเพื่อทำการเชื่อมต่อแล้ว เราจะทำการเช็คข้อมูล ได้ด้วยการดูที่แถบ Status ที่ได้ออกแบบไว้ ดังรูป



รูปที่ 3.11 แถบแสดงสถานะ

3.2.3 คำสั่งในการทำงานของ Visual Basic

1). ตั้งเวลาให้กับบอร์ด โดยการใส่ค่าเวลาได้โดยตรง เมื่อทำการป้อนข้อมูล ชั่วโมง : นาที : วินาที แล้วคลิกที่ปุ่ม TimeSetup ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์ จะอยู่ในรูป [T hh:mm:ss] ซึ่งจะทำให้เวลาถูกตั้งค่าใหม่



รูปที่ 3.12 TimeSetup

2). ตั้งวัน/เดือน/ปี โดยเลือกวันจาก ComboBox ซึ่งจะมีตั้งแต่วันอาทิตย์ถึงวันเสาร์ และทำการป้อนค่า วัน/เดือน/ปี จากนั้นกดปุ่ม DaySetup โดยใน ComboBox นั้น หากเป็นวันอาทิตย์ ค่าของข้อมูลที่ส่งจะกำหนดให้เป็น 1 วันจันทร์ = 2, วันอังคาร = 3 ไปจนครบ 7 วัน ตัวอย่างเช่น ถ้าต้องการตั้งค่าให้เป็นวันศุกร์ ที่ 9 เมษายน 2553 ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะได้เป็น [D 6 09 04 53]



รูปที่ 3.13 DaySetup

3). สามารถสั่งให้ รั้น,บันทึก และหยุด ได้ด้วย 3 ปุ่ม ดังนี้

- Run/Stop หากทำการกดที่ปุ่มนี้ ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะเป็น [G]
- บันทึกจากปุ่ม Rec หากทำการกดที่ปุ่มนี้ ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะเป็น [R]
- หยุดบันทึกด้วยปุ่ม StopRec หากทำการกดที่ปุ่มนี้ ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะเป็น [S]



รูปที่ 3.14 Run/Stop, Rec, StopRec

4). การกำหนดวันเวลาในการบันทึก ทำได้โดย

- เลือกวันที่จะทำการบันทึก (อาทิตย์ - เสาร์)
- เลือกโปรแกรมที่จะทำการบันทึก ในหนึ่งวันเราสามารถโปรแกรมการทำงานได้ 21 โปรแกรมด้วยกัน คือตั้งแต่ 00 - 20 ซึ่งไม่ควรใช้โปรแกรมที่ซ้ำกัน
- ตั้งเวลาที่อยากบันทึกหรือหยุดการบันทึก
- คลิกที่ CheckBox ในการบันทึกหรือหยุดการบันทึก
- กด ProgramSetup

เมื่อทำตามขั้นตอนข้างต้นแล้ว ข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะเป็น [P วัน โปรแกรมที่ hh:mm บันทึกลงหรือหยุดบันทึก] เช่น

[P 1 00 13:50 rrrrrrr] เป็นคำสั่งการสั่งงานให้ทำการบันทึก (rrrrrrr) ในวันอาทิตย์ (1) เวลา 13.50 โดยใช้โปรแกรมลำดับที่ 00 หรือ

[P 3 15 19:31 -----] เป็นคำสั่งการสั่งงานให้หยุดบันทึก (-----) ในวันอังคาร (3) เวลา 19.31 โดยใช้โปรแกรมลำดับที่ 15



รูปที่ 3.15 ProgramSetup

5). หากเราต้องการที่จะตรวจสอบดูว่า วันที่เราทำการโปรแกรมไว้ นั้น ถูกต้องหรือไม่ สามารถทำได้โดย ไปที่วันที่ต้องการตรวจสอบ เลือกวันที่ต้องการ แล้วคลิก ตรวจสอบ ข้อมูล จะมีข้อมูลของโปรแกรมในวันที่เราเลือกแสดงขึ้นที่แถบสถานะ โดยข้อมูลที่ถูกส่งไปยังไมโครคอนโทรลเลอร์จะเป็น [E วัน] เช่น E1, E2,...,E7



รูปที่ 3.16 ตรวจสอบข้อมูล

3.2.4 วิธีการสร้างโปรแกรมเพื่อใช้งาน

หลังจากสามารถเขียนโปรแกรมติดต่อได้ตามความต้องการแล้ว เราจำเป็นต้องสร้างตัวโปรแกรมที่เขียนขึ้นมาให้สามารถใช้ได้ในคอมพิวเตอร์เครื่องอื่นๆ เพื่อความสะดวกและรวดเร็วในการใช้งาน โดยที่เครื่องนั้นไม่จำเป็นต้องมีโปรแกรม Visual Basic 6 โดยไปที่ File แล้วคลิกที่ Make... จะได้ไฟล์ที่เป็น .exe ซึ่งมีขนาด 804 กิโลไบต์เท่านั้น



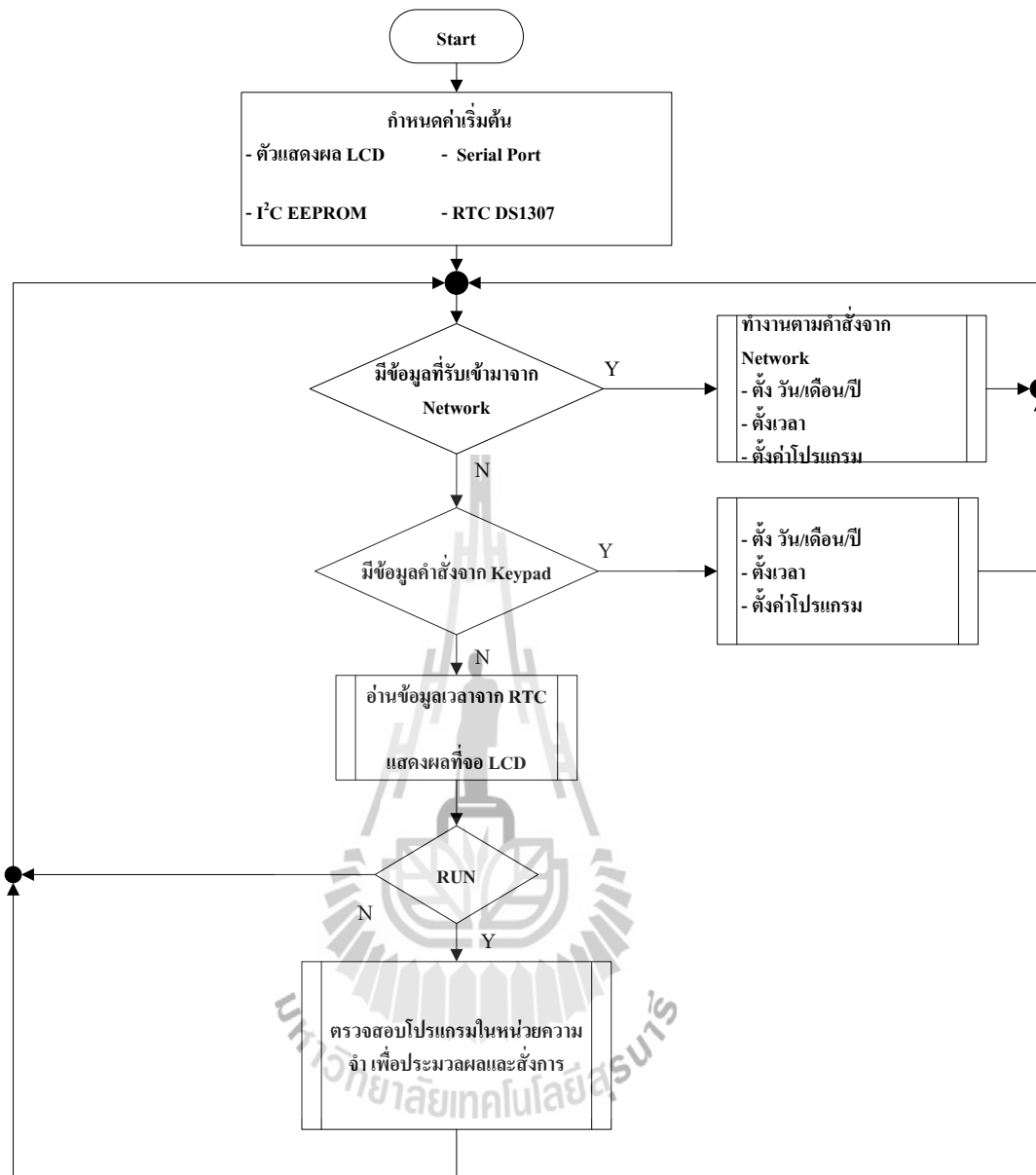
โปรแกรมบันทึก Video และ Audio ผ่าน Network.exe

รูปที่ 3.17 โปรแกรมบันทึก Video และ Audio อัตโนมัติผ่านระบบ Network

หากกล่าวโดยสรุปแล้วการใช้งานโปรแกรม Visual Basic ในโครงการนี้ มีคำสั่งที่ถูกส่งไป ยังไมโครคอนโทรลเลอร์หลายคำสั่งด้วยกัน ทั้งนี้คำสั่งต่างๆ จะแก้ไขได้โดยแก้ไขโปรแกรมที่เราได้ เขียนให้กับไมโครคอนโทรลเลอร์เท่านั้น แสดงให้เห็นว่าการออกแบบการทำงานผ่าน Visual Basic เป็นเพียงโปรแกรมที่เราเขียนขึ้นเพื่อให้สะดวกในการใช้งานมากยิ่งขึ้น

3.3 การออกแบบระบบการทำงานโดยรวม

แรกเริ่มเราจะทำการกำหนดค่าเริ่มต้น ให้กับอุปกรณ์ต่างๆ ที่เกี่ยวข้องทั้งหมดทั้งตัว แสดงผล LCD, หน่วยความจำ EEPROM , ตัวไอซีฐานเวลา DS1307 หรือแม้กระทั่งค่าของซีเรียลพอร์ตต่างๆ จากนั้นหน่วยประมวลผลก็จะตรวจสอบว่ามีคำสั่งเข้ามาจากระบบเครือข่ายหรือไม่ ถ้ามีการรับค่าเข้ามาจากระบบเครือข่าย หน่วยประมวลผลก็จะส่งค่าที่รับมาไปยังอุปกรณ์ต่างๆ ที่ เครือข่ายต้องการจะป้อนให้ เช่น ถ้าเป็นการตั้งวันเวลา ค่าที่ส่งมาก็จะถูกส่งไปที่ตัวอุปกรณ์ที่ชื่อว่า DS1307 ให้ทำการรับและเก็บค่านั้นไว้รอการเรียกแสดงผล เป็นต้น แต่ถ้าไม่มีการรับค่าเข้ามาจากระบบเครือข่าย หน่วยประมวลผลก็จะมารับข้อมูลจากทางคีย์บอร์ดที่ติดอยู่กับตัวอุปกรณ์ ถ้ามีการรับค่าจากทางคีย์บอร์ด ตัวประมวลผลก็จะเช็คค่าที่ถูกป้อนเข้ามาเกี่ยวข้องกับอุปกรณ์ตัวใด แล้วจึงทำการส่งค่าไปให้กับอุปกรณ์ตัวนั้นๆ เช่นเดียวกันกับค่าที่รับจากทางระบบเครือข่าย แต่ถ้าไม่มีค่าที่เข้ามา หน่วยประมวลผลก็จะไปอ่านข้อมูลจาก ไอซีฐานเวลา แล้วนำมาแสดงผลที่หน้าจอ LCD แต่ถ้ามีการป้อนข้อมูลเข้ามาไม่ว่าจะเป็นทางเครือข่าย หรือทางคีย์บอร์ด หน่วยประมวลผลก็จะนำข้อมูลที่ถูกรับมาแสดงที่ตัวแสดงผล LCD และถ้ามีการสั่งให้โปรแกรมที่ได้ป้อนไว้ทำงาน หน่วยประมวลผลก็จะสั่งการให้ตรวจสอบและดำเนินการตามข้อมูลในหน่วยความจำที่เราบันทึกไว้



รูปที่ 3.18 Flow Chart โปรแกรมโดยรวม

บทที่ 4

การทดสอบและการใช้งาน

หลังจากที่ทำการประกอบอุปกรณ์ทั้งหมดเข้าด้วยกันจะมีลักษณะดังรูป

4.1



รูปที่ 4.1 ชุดอุปกรณ์ที่ได้รับการประกอบแล้ว

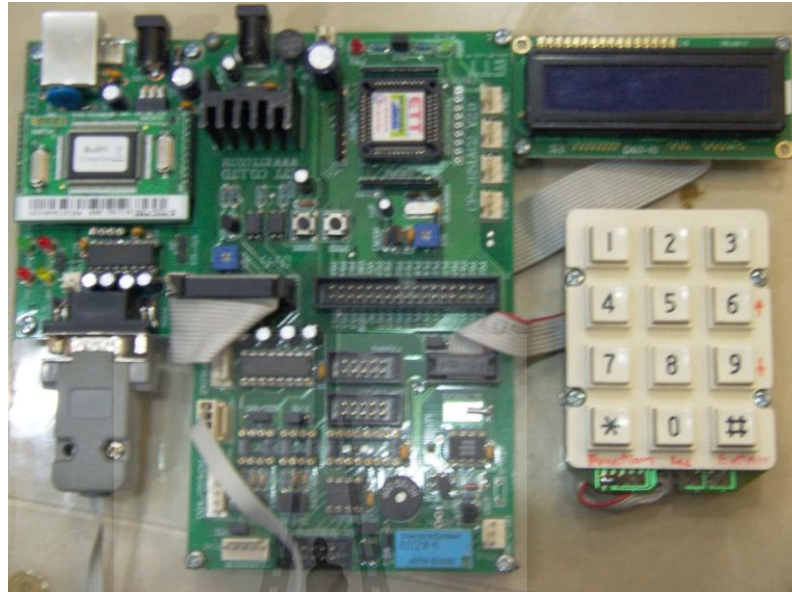
ขั้นตอนการทดลองการใช้บอร์ดมีอยู่สองวิธี คือ

4.1 การใช้งานโดยการป้อนข้อมูลผ่านคีย์บอร์ดโดยตรง (Manual)

4.2 การใช้งานโดยการรับค่าข้อมูลผ่านทางระบบเครือข่าย (Network)

4.1 การทดสอบวิธีที่ 1 การใช้งานโดยการป้อนข้อมูลผ่านคีย์บอร์ด

- 1) จากรูป 4.2 ลักษณะของบอร์ดทดลองก่อนที่จะจ่ายไฟเข้าบอร์ด



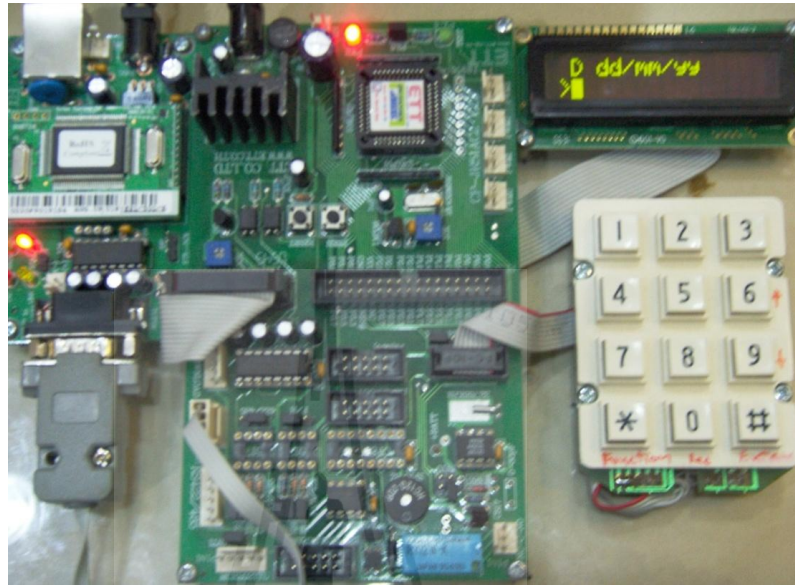
รูปที่ 4.2

- 2) เมื่อจ่ายไฟเข้าบอร์ดทดลอง หน้าจอ LCD จะแสดงวันที่และเวลา (แสดงดังรูป 4.3)

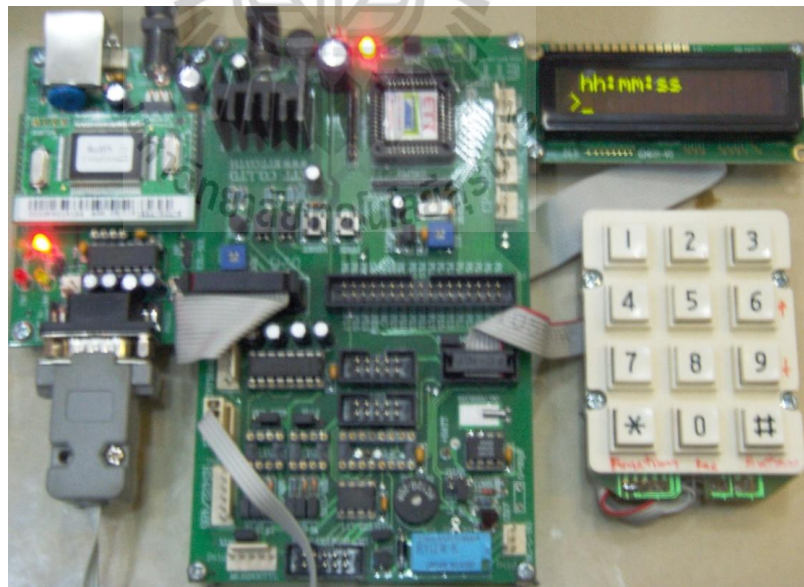


รูปที่ 4.3

- 3) เมื่อเราทำการกดปุ่ม function (*) จอ LCD จะแสดงหน้าจอการตั้งวันเดือนปีออกมา (รูป 4.4) เมื่อทำการตั้งวันเดือนปีเสร็จเรียบร้อย จึงกดปุ่ม Enter (#) หน้าจอจะแสดงการตั้งเวลา (รูป 4.5)



รูปที่ 4.4



รูปที่ 4.5

- 4) เมื่อเราทำการตั้งเวลาเสร็จเรียบร้อยแล้วกดปุ่ม Enter (#) หน้าจอ จะแสดงให้ตั้งค่าวันที่จะทำการเซตโปรแกรมให้ทำงานในวันไหน (รูป 4.6) โดยตามหน้าจอวันต่างๆ จะแทนด้วยเลขต่างๆ ดังนี้

เมื่อเรากดเลข 1 จะแทนด้วยวันอาทิตย์

เมื่อเรากดเลข 2 จะแทนด้วยวันจันทร์

เมื่อเรากดเลข 3 จะแทนด้วยวันอังคาร

เมื่อเรากดเลข 4 จะแทนด้วยวันพุธ

เมื่อเรากดเลข 5 จะแทนด้วยวันพฤหัสบดี

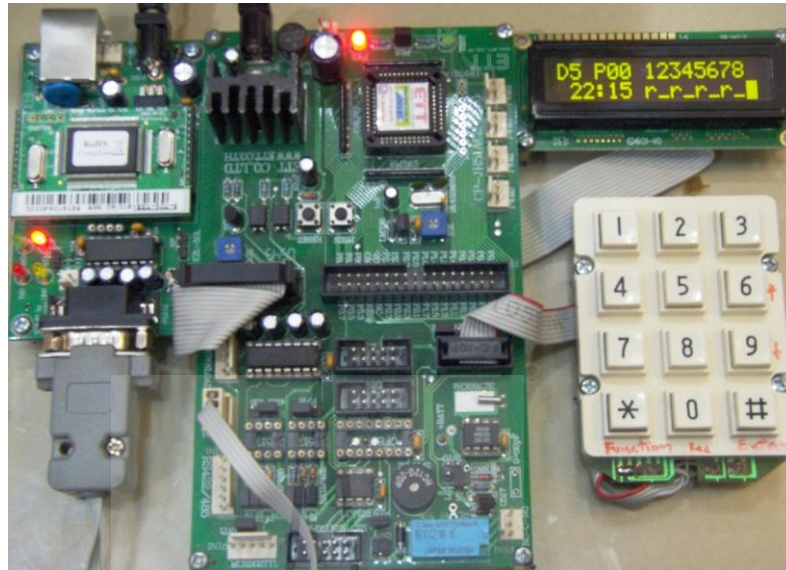
เมื่อเรากดเลข 6 จะแทนด้วยวันศุกร์

เมื่อเรากดเลข 7 จะแทนด้วยวันเสาร์



รูปที่ 4.6

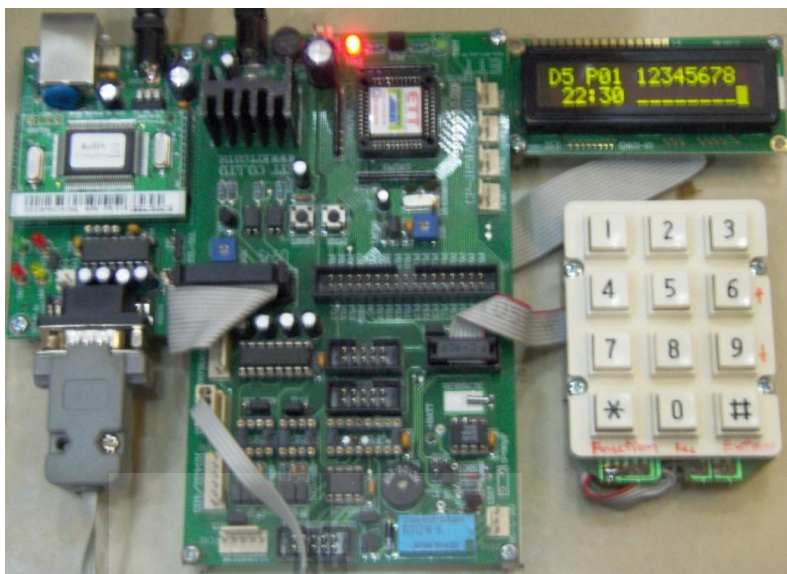
- 5) เมื่อเราทำการตั้งค่าวันที่ต้องการจะตั้งค่าให้โปรแกรมทำงานเสร็จแล้ว จึงกดปุ่ม Enter (#) จากนั้นหน้าจอจะแสดง ข้อความให้ตั้งค่าเวลาที่จะทำการบันทึกให้เราทำการกด Enter(#) อีกที เพื่อทำการตั้งเวลาที่บันทึกในวันที่เราตั้งไว้ในขั้นตอนที่ 4 เมื่อทำการตั้งเวลาเรียบร้อยแล้ว จึงกด Enter (#) เคอร์เซอร์จะเลื่อนมาเทียบได้ตัวเลข 1-8 ให้เราทำการเลือกช่องที่จะทำการบันทึก (ทำได้สูงสุด 8 ช่อง) โดยการกด record (0) ตรงตำแหน่งช่องที่เราต้องการ โดยถ้าไม่ต้องการจะบันทึกช่องไหน ให้ทำการกด Enter(#) เพื่อเว้นว่างช่องนั้นไว้ ดังรูป 4.7 จะแสดงการบันทึกสัญญาณในช่อง 1,3,5 และ 7 เริ่มต้นเวลา 22.15 น.



รูปที่ 4.7

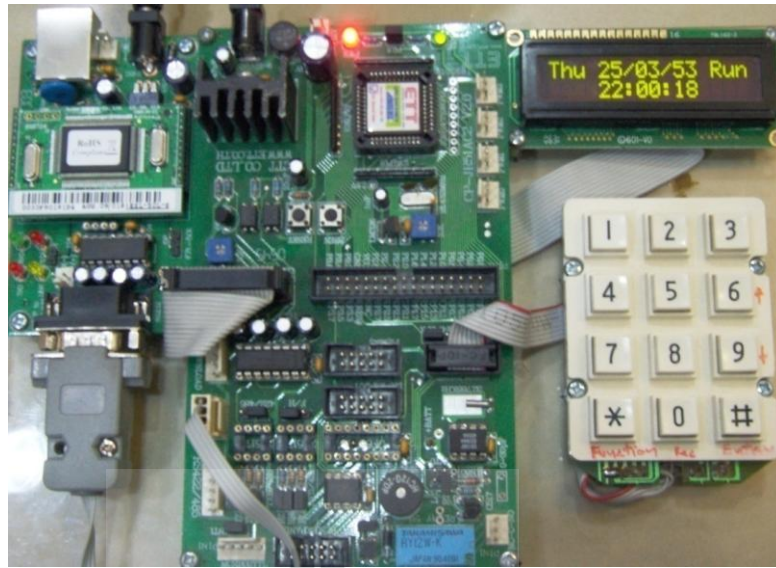
- 6) จากขั้นตอนที่ 5 เมื่อเราทำการตั้งค่าเสร็จเรียบร้อยแล้วให้เราทำการกด Enter (#) อีกครั้ง หน้าจอจะแสดงให้เราทำการตั้งเวลาเหมือนขั้นตอนที่ 5 แต่ในขั้นตอนนี้เราจะทำการตั้งโปรแกรมให้หยุดบันทึกสัญญาณที่เราได้ทำการตั้งไว้ในขั้นตอนที่แล้ว โดยจากรูปที่ 4.8 เราจะทำการยกเลิกการบันทึกในเวลา 22.30 น. เมื่อเราทำการตั้งเวลาที่ต้องการแล้ว Cursor จะข้ามมากระพริบได้ช่องสัญญาณ (1-8) ให้เราทำการกดเว้นว่างไว้ทั้งหมด โดยการกด Enter (#) เพื่อทำการหยุดบันทึกสัญญาณ

หมายเหตุ - ถ้าเราไม่ทำการตั้งค่าตามขั้นตอนที่ 6 โปรแกรมจะทำการบันทึกต่อเนื่องไปเรื่อยๆ



รูปที่ 4.8

- 7) เมื่อเสร็จขั้นตอนที่ 6 แล้ว ถ้าเราจะทำการตั้งโปรแกรมให้บันทึกสัญญาณอีก ให้กดปุ่ม Enter (#) แล้วทำการตั้งค่าตามขั้นตอนที่ 5 และ 6 (โดยจะทำการโปรแกรมได้สูงสุด 21 โปรแกรม) เมื่อเราต้องการจะจบการตั้งค่า (ตั้งค่าเสร็จเรียบร้อย) ของโปรแกรม ให้เรากดปุ่ม Function (*) เมื่อกดเรียบร้อยแล้วจะกลับมาแสดงวันและเวลาที่เรที่ตั้งไว้ ให้เราทำการกดปุ่ม Enter (#) อีกครั้ง เพื่อให้บอร์ดทำการรันโปรแกรมที่เราตั้งไว้ จากรูป 4.9 จะเห็นว่าเมื่อเรากด Enter (#) หน้าจอจะมีการแสดงคำว่า RUN ปรากฏขึ้น ซึ่งแสดงว่าบอร์ดได้ทำการบันทึกสัญญาณตามที่เราได้โปรแกรมไว้



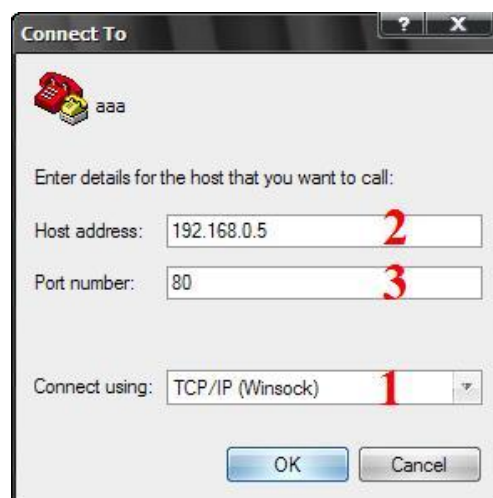
รูปที่ 4.9

8) จบการทดลองด้วยวิธีป้อนข้อมูลผ่านคีย์บอร์ด

4.2 การทดลองวิธีที่ 2 การใช้งานโดยการป้อนข้อมูลผ่านระบบเครือข่าย

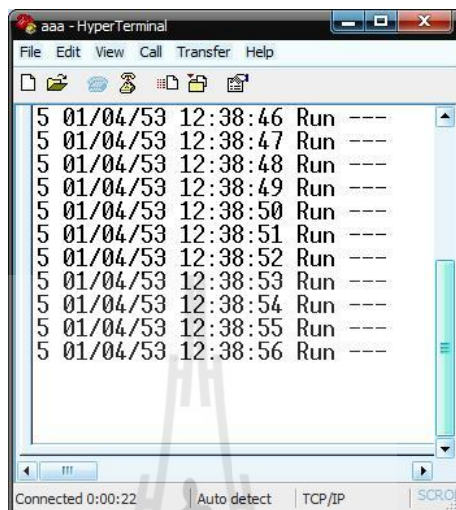
ซึ่งการทดสอบนี้เราจะทำการทดสอบโดยกระทำผ่านโปรแกรม Hyper Terminal ซึ่งมีมาให้ใน Window

- 1) ทำการเปิดโปรแกรม Hyper Terminal ขึ้นมาแล้วทำการตั้งค่าเชื่อมต่อตามที่แสดงในรูป 4.10 (ในที่นี้เราทำการกำหนดไอพีแอดเดรสของโฮสเป็น 192.168.0.5)



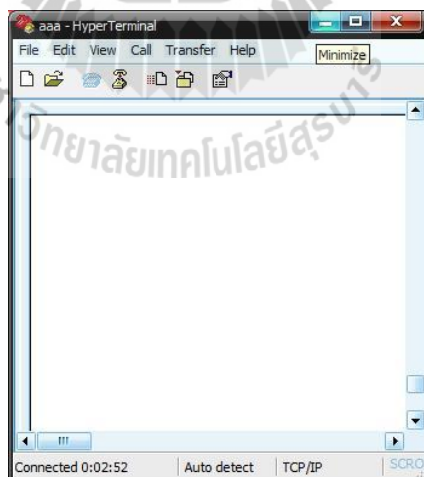
รูปที่ 4.10

- 2) เมื่อเราทำการตั้งค่าขั้นตอนแรกเรียบร้อยแล้ว เมื่อเรากดปุ่ม OK หน้าต่างใน hyper terminal จะปรากฏดังรูปที่ 4.11 ซึ่งแสดงว่าโปรแกรมได้ทำการเชื่อมต่อเข้ากับอุปกรณ์เรียบร้อยแล้ว (ซึ่ง Hyper Terminal จะทำการแสดงการรันของอุปกรณ์)



รูปที่ 4.11

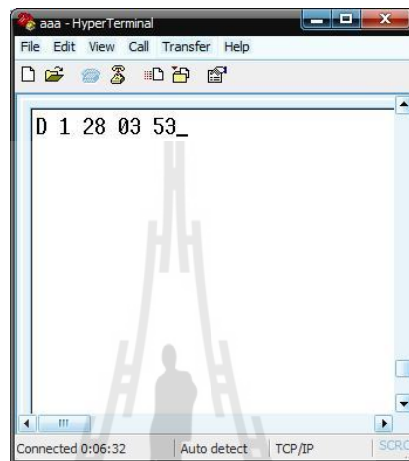
- 3) ในขั้นตอนนี้เมื่อเราทำการพิมพ์ H แล้วกดปุ่ม Enter หน้าต่างใน Hyper Terminal จะเป็นดังรูปที่ 4.12 ซึ่งหมายความว่าตัวอุปกรณ์รอรับการตั้งค่า



รูปที่ 4.12

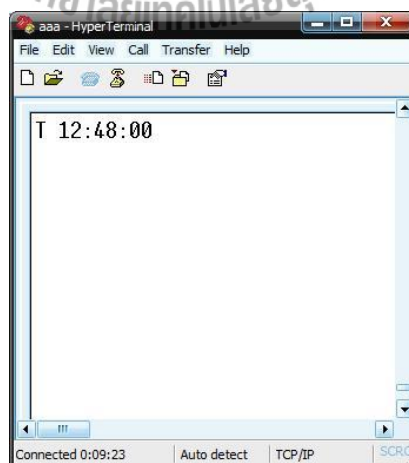
- 4) มาถึงขั้นตอนนี้เราจะทำการตั้งวันเดือนปีให้กับอุปกรณ์โดยผ่านโปรแกรมนี้ ซึ่งในการตั้งค่าวันจะกระทำได้โดยการพิมพ์ D แล้วเว้นวรรคหนึ่งครั้งแล้วใส่ตัวเลข 1-7 ตัวเลขนี้จะแทนวัน ซึ่งหมายเลข 1 ก็คือวันอาทิตย์ ไล่ไปจนถึงวันเสาร์ หมายเลข 7

เมื่อเราทำการใส่วันเสร็จเรียบร้อยแล้วจึงทำการเว้นวรรคหนึ่งครั้งเพื่อใส่วันที่ เดือน และปี (ซึ่งการใส่วันที่ เดือนและปี จะใช้เลขสองหลัก และเว้นวรรคหนึ่งครั้ง ระหว่างตัวเลขสองหลัก ดังรูปที่ 4.13) จากนั้นทำการกด Enter เป็นการจบการตั้ง ค่าวัน เดือน ปี



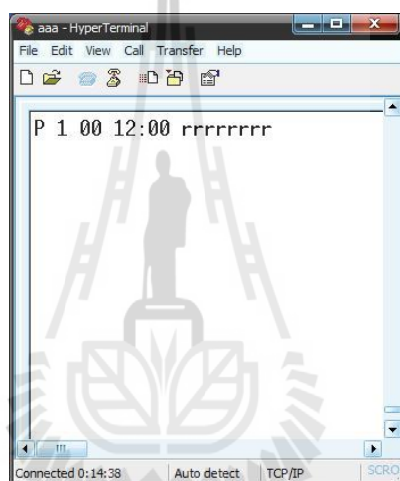
รูปที่ 4.13

- 5) ขั้นตอนนี้เป็นวิธีการตั้งเวลาซึ่งเราจะกระทำได้โดยการพิมพ์ T เว้นวรรคแล้ว ตามด้วยการใส่เวลา โดยจะใส่เป็น ชั่วโมง นาที และวินาที ดังรูปที่ 4.14 จากนั้นกด Enter



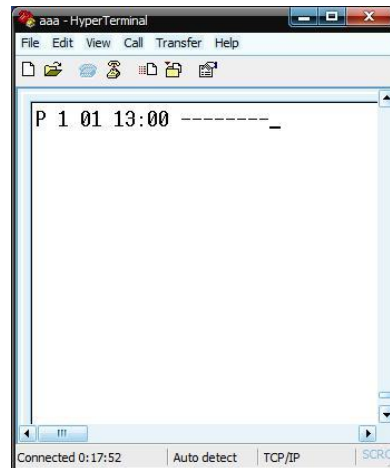
รูปที่ 4.14

- 6) ต่อมาเราจะทำการโปรแกรมสั่งการตั้งเวลาให้บอร์ดทำการบันทึกสัญญาณ โดยกระทำได้โดยพิมพ์ P ตามด้วยวันที่ต้องการบันทึก (หมายเลข 1-7) จากนั้นตามด้วยหมายเลขโปรแกรมที่จะทำการบันทึก (ทำได้สูงสุด 21 โปรแกรม) และตามด้วยเวลาที่จะทำการบันทึก ซึ่งจะใส่แค่เวลาเป็นชั่วโมงและนาที พอเรากระทำการตั้งค่าวันและโปรแกรมเรียบร้อยแล้ว ให้เราทำการกดเว้นวรรคหนึ่งครั้ง เพื่อที่จะสั่งการให้ช่องสัญญาณไหนทำการบันทึกบ้าง โดยการพิมพ์ r ซึ่งในที่นี้จะทำการพิมพ์ได้สูงสุด 8 ช่องสัญญาณ โดยถ้าเราไม่ต้องการให้ช่องไหนทำการบันทึกให้ใส่เครื่องหมาย - ไว้ (ดังรูปที่ 4.15 จะทำการบันทึกทุกช่องสัญญาณ) จากนั้นกด Enter



รูปที่ 4.15

- 7) เมื่อเราสั่งให้บอร์ดบันทึกสัญญาณแล้วมาถึงขั้นตอนที่เราจะสั่งให้บอร์ดหยุดการบันทึกสัญญาณซึ่งกระทำคล้ายกับขั้นตอนที่ 6 แต่จะแตกต่างกันตรงที่ในส่วนของหมายเลขโปรแกรมที่เราจะสั่งยกเลิกการบันทึก โดยในขั้นตอนที่แล้วเราบันทึกในโปรแกรมหมายเลข 00 เมื่อเราทำการยกเลิกเราจะต้องใส่หมายเลข 01 (เพิ่มมาหนึ่งโปรแกรม) และทำการตั้งเวลาที่จะทำการยกเลิก จากนั้นให้ใส่เครื่องหมาย _ ไว้ในช่องที่จะทำการยกเลิกการบันทึกดังรูปที่ 4.16 จึงกด Enter
- หมายเหตุ – ถ้าเราใส่อักษร r ไว้ได้ช่องสัญญาณใด ช่องสัญญาณนั้นจะทำการบันทึกต่อไป

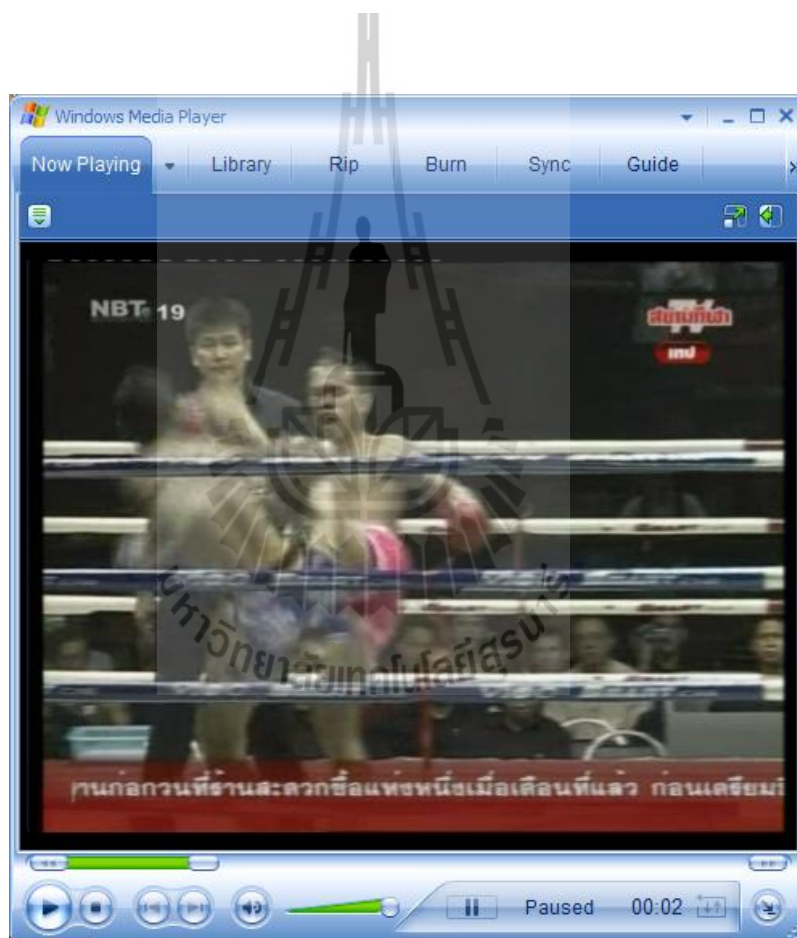


รูปที่ 4.16

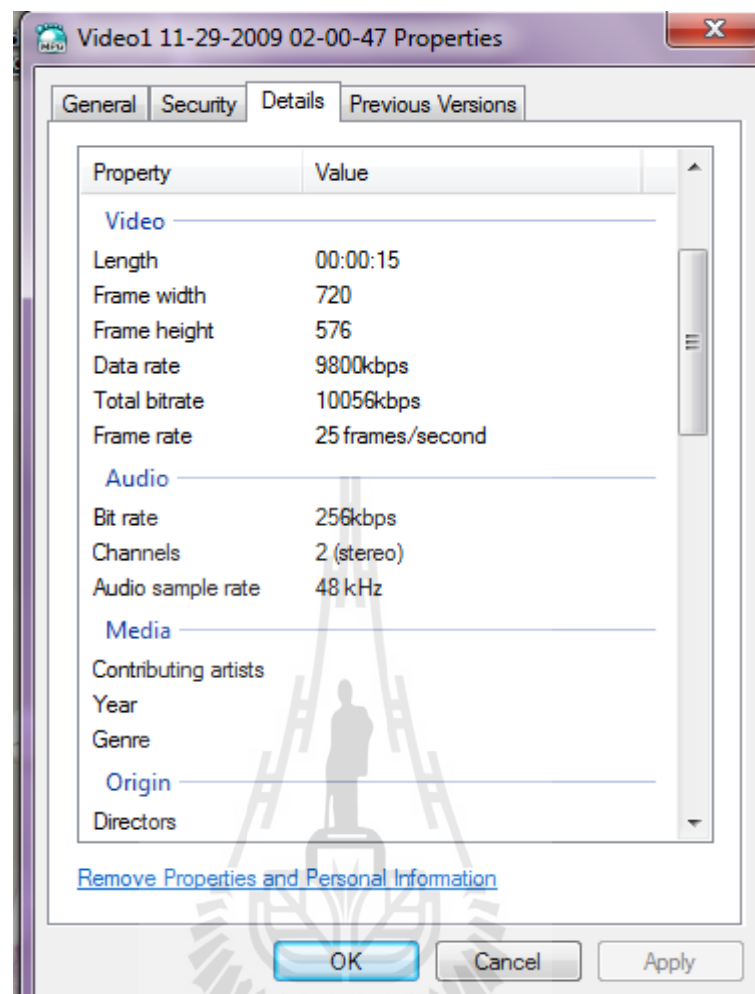
- 8) เมื่อกระทำครบทุกขั้นตอนจากนั้นกด G แล้วกด Enter เพื่อสั่งให้โปรแกรมทำงาน จากสิ่งที่เราได้กระทำตามขั้นตอนต่างๆ ข้อมูลทั้งหมดจะถูกนำไปเก็บไว้ในตัวบอร์ด ซึ่งเมื่อเราตั้งค่าอะไรใน Hyper Terminal ค่าต่างๆ เมื่อเราไปกดตรวจสอบดูในตัวอุปกรณ์ค่าต่างๆ จะแสดงตามที่เราได้ตั้งไว้ ซึ่งแสดงว่าตัวบอร์ดสามารถรับและส่งค่าผ่านระบบเครือข่ายเรียบร้อยแล้ว

การใช้งานผ่าน Visual Basic

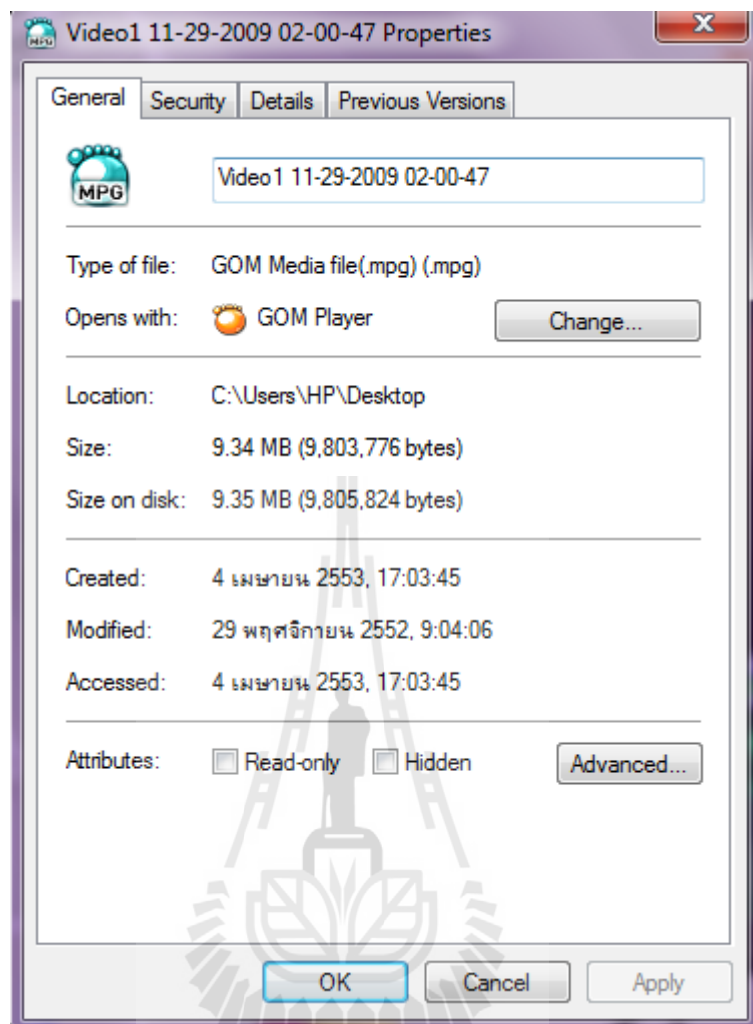
เปิดโปรแกรมที่ได้จากขั้นตอนที่ 3.2.4 (บทที่ 3) ขึ้นมา จากนั้นทำการเริ่มบันทึกสัญญาณ Video และ Audio โดยการตั้งเวลาให้บันทึก และหยุดบันทึก เมื่อถึงเวลาที่ตั้งไว้ตัวชุดอุปกรณ์ก็จะสั่งการให้ทำการบันทึกสัญญาณ ที่เราได้ทำการตั้งค่าไว้ หลังจากที่ตัวชุดอุปกรณ์ได้บันทึกสัญญาณเสร็จเรียบร้อยแล้ว เราก็จะทำการแชร์ตัว Hard disk ให้กับเครือข่าย เพื่อที่จะสามารถดึงข้อมูลที่เราบันทึกออกมาได้ โดยไฟล์ที่เราได้มาจะอยู่ในรูปแบบไฟล์ในตระกูล MPEG ซึ่งจะมีรายละเอียดดังนี้



รูปที่ 4.17 ทดสอบการเล่นไฟล์ที่ได้จากการบันทึก



รูปที่ 4.18 รายละเอียดไฟล์
มหาวิทยาลัยเทคโนโลยีสุรนารี



รูปที่ 4.19 รายละเอียดของไฟล์ที่ได้จากการบันทึก

ซึ่งจากการทดสอบการใช้งานจริงเราจะเห็นได้ว่าไฟล์ที่เราได้มาจะมีคุณภาพตามที่เราได้ตั้งค่าไว้กับตัวฮาร์ดดิสก์ ซึ่งในที่นี้เราได้ทำการตั้งค่าความละเอียดไว้ต่ำสุด เพื่อเป็นการประหยัดพื้นที่ความจุในตัวฮาร์ดดิสก์ เนื่องจากถ้าเราตั้งค่าของการบันทึกไว้ในระดับสูงสุดขนาดของไฟล์จะมีขนาดใหญ่มาก โดยขนาดของไฟล์จะมีขนาดตามระยะเวลาของการบันทึก

จากการทดสอบการใช้งานเราจะเห็นได้ว่าชุดอุปกรณ์สามารถทำงานได้ตามที่เราได้ตั้งค่าไว้ทุกประการ ทั้งการตั้งวันเวลาที่บันทึกสัญญาณ หรือแม้กระทั่งการตั้งให้ยกเลิกการบันทึกสัญญาณ และนอกจากนั้นเมื่อเราบันทึกสัญญาณเรียบร้อยแล้ว เรายังสามารถที่จะดึงไฟล์ที่ได้รับการบันทึกแล้วมาใช้งานโดยใช้วิธีดาวน์โหลดมาจากระบบเครือข่ายได้

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผล

จากการทดลองการใช้งานระบบควบคุมการบันทึกสัญญาณ Video และ Audio ผ่านระบบเครือข่ายโดยอัตโนมัติ เราจะเห็นได้ว่าการทำงานของชุดอุปกรณ์สามารถทำการควบคุมสั่งการการบันทึกสัญญาณ ได้เป็นไปตามค่าที่เราได้ตั้งไว้ และสามารถทำงานได้ตามขอบเขตที่กำหนดไว้ตั้งแต่แรกเริ่มของการทำโครงการชิ้นนี้

ซึ่งก็คือชุดอุปกรณ์สามารถทำการตั้งวันเวลา และระยะเวลาในการบันทึกสัญญาณได้ และในขณะที่ระบบได้ทำการบันทึกสัญญาณอยู่นั้นก็จะแสดงสถานการณ์ทำงานผ่านทางตัวจอ LCD ของตัวชุดอุปกรณ์ ในขณะที่เดียวกันทางหน้าเว็บเพจก็จะแสดงสถานการณ์ทำงานไว้ให้ดูด้วยเช่นกัน ส่วนในการทำงานผ่านระบบเครือข่ายก็จะสามารถทำงานได้ดีโดยผ่านโปรแกรมที่ผู้จัดทำได้ทำการสร้างขึ้นมาเพื่อให้ง่ายต่อการใช้งานในการควบคุมระบบบันทึกสัญญาณ

เราจะเห็นได้ว่าเมื่อเราทำการบันทึกสัญญาณ ไฟล์ของสัญญาณที่บันทึกจะอยู่ในรูปแบบไฟล์ในตระกูล MPEG ซึ่งเป็นไฟล์ที่รองรับระบบข้อมูลภาพและเสียงได้เป็นอย่างดี โดยมีความละเอียดของไฟล์ video อยู่ที่ 10056kbps และ audio ที่ 256kbps โดยไฟล์รวมทั้งหมดจะมีขนาด 9.34MB ในระยะเวลาการบันทึก 15 วินาที ซึ่งไฟล์ชนิดนี้จะสามารถเปิดใช้งานได้กับโปรแกรมหลายๆโปรแกรม อาทิ เช่น Window media player, Winamp ,Real Player และอื่นๆ อีกมากมาย ซึ่งโดยแรกเริ่มไฟล์จะถูกเก็บไว้ในตัวฮาร์ดดิสก์เป็นการชั่วคราวก่อน แล้วเมื่อเราจะนำไฟล์ใดมาใช้งานเราก็จะทำการดึงไฟล์ข้อมูลนั้นมาจากตัวฮาร์ดดิสก์โดยวิธีการดาวน์โหลดผ่านระบบเครือข่าย โดยความเร็วของการดาวน์โหลดไฟล์ข้อมูลนั้นจะขึ้นอยู่กับความแรงสัญญาณของระบบเครือข่าย ซึ่งในการทดลองนี้อัตราการความเร็วของเครือข่ายที่ใช้คือ 4MB

5.2 สิ่งที่ได้รับจากการทำโครงการ

- 1) ได้รับความรู้เกี่ยวกับการประยุกต์การใช้งานของอุปกรณ์ไมโครคอนโทรลเลอร์ชนิด MCS-51
- 2) ได้รับความรู้เกี่ยวกับการเขียนโปรแกรมในการสั่งงานระบบผ่านทางไมโครคอนโทรลเลอร์
- 3) ได้รับความรู้เกี่ยวกับการควบคุมการทำงานของชุดอุปกรณ์กับระบบเครือข่าย
- 4) ได้เรียนรู้การเขียนและพัฒนาโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์อย่างถูกต้องด้วยตัวเอง
- 5) ได้ทำงานร่วมกับผู้อื่น
- 6) สามารถนำความรู้ที่ได้จากการศึกษาทฤษฎีมาปฏิบัติจริง
- 7) สามารถนำความรู้ที่ได้จากโครงการไปประยุกต์ใช้งานในชีวิตได้

5.3 ปัญหาและอุปสรรค

- 1) ไม่มีความรู้มากพอเกี่ยวกับอุปกรณ์ต่างๆ จึงต้องใช้เวลาในการศึกษาข้อมูลก่อนทำการสั่งซื้อ
- 2) ไม่มีความรู้เกี่ยวกับลักษณะทางกายภาพของตัวไอซี รวมทั้งอุปกรณ์ต่างๆ ดังนั้นจึงใช้เวลาในการศึกษาคู่มือการใช้งานเป็นเวลานาน
- 3) ไม่มีความรู้มากพอเกี่ยวกับการเขียนโปรแกรมสั่งการผ่านระบบเครือข่าย จึงต้องทำการศึกษาการเขียนโปรแกรมเป็นเวลานาน ทำให้เสียเวลาเป็นอย่างมาก
- 4) ไฟล์ที่ได้จากการบันทึกสัญญาณจะออกมาในรูปแบบไฟล์ในตระกูล MPEG อย่างเดียว ดังนั้นถ้าต้องการจะใช้ไฟล์ในรูปแบบอื่นจะต้องใช้ตัวแปลงไฟล์ไปใช้งาน
- 5) ไม่มีความรู้ในการใช้งานโปรแกรม Visual Basic จึงต้องใช้เวลาการศึกษาเป็นเวลาค่อนข้างนาน
- 6) เมื่อจะทำการดาวน์โหลดไฟล์ความเร็วของการดาวน์โหลดจะขึ้นอยู่กับอัตราความแรงของสัญญาณเครือข่ายของ ณ สถานที่นั้น
- 7) เนื่องจากข้อจำกัดการใช้งานของตัวฮาร์ดดิสก์ที่เก็บข้อมูล ทำให้เวลาเซิร์ฟไฟล์ให้กับเครือข่ายจะไม่สามารถสั่งให้ทำการบันทึกสัญญาณได้

5.4 ข้อเสนอแนะและแนวทางการพัฒนา

ในโครงการระบบบันทึกสัญญาณ Video และ Audio โดยส่งงานผ่านระบบเครือข่ายโดยอัตโนมัติ จะสามารถนำไปใช้ในอาคารเรียนหรือสถานที่ต่างๆ ที่ต้องการจะทำการบันทึกสัญญาณได้ภาพหรือเสียงได้ อย่างเช่นในห้องเรียน เมื่อเราได้บันทึกการสอนของรายวิชาใดไว้ นักเรียนนักศึกษาผู้ที่ไม่ได้เข้าเรียนในรายวิชานั้น ก็จะสามารถเรียนย้อนหลังได้จากไฟล์ข้อมูลที่ได้ทำการบันทึกไว้ ซึ่งทั้งนี้ ก็ต้องขึ้นอยู่กับผู้ดูแลระบบด้วยว่าจะจัดรูปแบบนำเสนอในรูปแบบใด และชุดอุปกรณ์ยังสามารถรองรับการทำงานของการทำงานของการบันทึกสัญญาณภาพและเสียง ได้จากกล้องหลายๆ ตัว โดยการเพิ่มตัว HUB หรือส่วนต่อพ่วงลงไป และถ้าต้องการให้มีการจัดเก็บข้อมูลในปริมาณมากๆ เราก็แค่เพิ่มความจุของตัวฮาร์ดดิสก์เท่านั้น



เอกสารอ้างอิง

เอกชัย มະการ **ปฏิบัติการทดลองและใช้งานไมโครคอนโทรลเลอร์ MCS51 ด้วยภาษาเบสิก**
บริษัท อีทีที จำกัด กรุงเทพฯ (2547)

วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตร **วีไล เรียนรู้และปฏิบัติการ**
ไมโครคอนโทรลเลอร์ MCS51 บริษัท อิน โนเวตีฟ เอ็กเพอริเมนต์ จำกัด กรุงเทพฯ

เอกชัย มະการ **รู้จักและเข้าใจ CHIPS SUPPORT แบบ I2C BUS** บริษัท อีทีที จำกัด
กรุงเทพฯ(2545)

สัจจะ จรัสรุ่งรวีวร **Internet & Network Programming กับ VB 6.0 และ ASP** สำนักพิมพ์
อินโฟเพรส นนทบุรี (2542)

ธีระศักดิ์ สุโขตินันท์, ประยุทธ์ อินแบน **โปรแกรมเมอร์มือใหม่ หัดเขียนโปรแกรม**
Microsoft Visual Basic 6 สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย กรุงเทพฯ (2549)

อำนวยการที่จันทิกร **ระบบรายงานพิักัดตำแหน่งบนถนนด้วยเสียงพูด** วิศวกรรมศาสตร์ สาขา
วิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีสุรนารี นครราชสีมา (2552)

เอกชัย มະการ **คู่มือการใช้งานบอร์ดไมโครคอนโทรลเลอร์ CP-JR51AC2 V1&V2** บริษัท
อีทีที จำกัด กรุงเทพฯ (2545)

EZL-50L User's Manual Sollae Systems.Co.Ltd Republic Of Korea จาก

<http://www.sollae.com> (2551)



ภาคผนวก
Code ของโปรแกรม

Code โปรแกรมหลัก (Assembly)

P4	EQU	0C0H
Port_SCK	EQU	P4.0
Port_DIO	EQU	P4.1
Rx_Buffer	EQU	20H
CNT_Rx_Buffer	EQU	21H
end_Rx_Buffer	EQU	22H
YEAR	EQU	23H
MONTH	EQU	24H
DATE	EQU	25H
DAY	EQU	26H
HOUR	EQU	27H
MIN	EQU	28H
SEC	EQU	29H
Last_Sec	EQU	2AH
YearInput	EQU	2BH
MonthInput	EQU	2CH
DateInput	EQU	2DH
DayInput	EQU	2EH
HourInput	EQU	2FH
MinInput	EQU	30H
SecInput	EQU	31H

```

ADD_HIGH      EQU  32H
ADD_LOW       EQU  33H
DATA          EQU  34H
DATA_KEY      EQU  35H
Last_MIN      EQU  36H
DataSetRunPro EQU  37H
SkipShowSerail EQU  38H
DayPro        EQU  39H
HourPro       EQU  3AH
MinPro        EQU  3BH
DataPro       EQU  3CH
StepPro       EQU  3EH
AddLo         EQU  3FH
AddHI         EQU  40H
BUF           EQU  41H ; 41H-48H
StatusRec     EQU  49H
StatusStop    EQU  3DH
Rx_00         EQU  4AH ;4AH --> 5FH

ORG 0000H

LJMP START ;reset vectar

ORG 0023H

JMP Intr_Serial

START:

CALL DELAY100MS

```

```

MOV    SP,#63h                ;define stack = 40 byte

CALL   Initial_Serial

SETB   CS_LCD                 ; initial LCD

CALL   DELAY100MS            ; initial delay

CALL   INIT_LCD              ; initial LCD

SETB   Port_SCK

SETB   Port_DIO

CALL   READ_DATE_TIME

MOV    Last_MIN,MIN

CALL   ShowTimeLCD

CALL   DELAY_500M

MOV    CNT_Rx_Buffer,#49H     ;รับข้อมูล adress 4AH - 5fH

MOV    end_Rx_Buffer,#00H

MOV    DPTR,#02FEH

CALL   EEPROM_Read

MOV    DataSetRunPro,A

MOV    SkipShowSerail,#0FFh

SETB   EA

SETB   ES

```

```

;+++++

```

```

; กำหนดค่าเริ่มต้นให้ Run Program

```

```

CALL   SCAN_KEY

CJNE   A,#00H,MAIN1

MOV    A,#'R'

```

```

MOV DPTR,#02FEH

CALL EEPROM_Write

;#####

MOV DPTR,#02FFH

CALL EEPROM_Read

;#####

MAIN1:      MOV  A,end_Rx_Buffer

            CJNE A,#0DH,MAIN11

            CALL ChkSerial

            MOV  CNT_Rx_Buffer,#49H

            MOV  end_Rx_Buffer,#00H

MAIN11:     CALL SCAN_KEY

            XRL  A,#0FFH

            JZ   _RD_TIME

            CALL CHK_COMMAND

_RD_TIME:  MOV  DPL,#00H

            CALL READ_TIME

            XRL  A,Last_Sec

            JZ   MAIN1

;=====

MOV  A,DataSetRunPro

CJNE A,#'R',MAIN21

MOV  C,P2.6

```

```

CPL    C

MOV    P2.6,C

;=====

MAIN21:    CALL  READ_DATE_TIME

          CALL  ShowTimeLCD

          MOV   A,SkipShowSerail

          JZ    MAIN22

          CALL  ShowTimeSerial

MAIN22:    MOV   A,MIN

          XRL  A,Last_MIN

          JZ    MAIN1

          MOV  Last_MIN,MIN

          MOV  A,DataSetRunPro

          CJNE A,#'R',MAIN1

;#####

          CALL  ChkProGram

;#####

          JMP  MAIN1

Ready:    DB   'Ready',0AH,0DH,'>',00H

          $INCLUDE "00System.sub"

          $INCLUDE "Serial.sub"

          $INCLUDE "DS1307.sub"

          $INCLUDE "24256.sub"

          $INCLUDE "Scankey.sub"

```

```
$INCLUDE "LCD.sub"
```

```
$Include      "Display.Sub"
```

```
$Include      "Command.Sub"
```

```
$Include      "SetPro.Sub"
```

```
$Include      "ChkPro.Sub"
```

Code โปรแกรมย่อย

Serial sub

```
=====
;
;   Serial Interrupt
;
=====

Intr_Serial:      PUSH      IE           ; Store Interrupt
                  CLR       ES           ; Disable Serial Interrupt
                  JNB       TI,Rx_Data_In
                  JMP       _X_Int_Serial ; If Tx Interrupt goto Exit

_Rx_Data_In:     JNB       RI,$
                  PUSH     B
                  PUSH     Acc
                  PUSH     00H

;=====

                  MOV      A,SBUF
                  MOV      Rx_Buffer,A
                  INC      CNT_Rx_Buffer
                  MOV      R0,CNT_Rx_Buffer
                  MOV      @R0,Rx_Buffer ;เก็บค่าที่ CNT_Rx_Buffer
```



```

MOV      end_Rx_Buffer,A
MOV      A,CNT_Rx_Buffer
CJNE    A,#62H,_Rx_Data_RET      ; เก็บข้อมูล xx byte
MOV      CNT_Rx_Buffer,#49H

;=====
_Rx_Data_RET:  POP      00H
               POP      Acc
               POP      B
_X_Int_Serial: CLR      TI
               CLR      RI
               POP      IE      ; Read Interrupt
               RETI

;=====
ChkSerial:  MOV      A,Rx_00
               CJNE    A,#'P',SerialSetTime
               CALL    SerialWriteProgram
               JMP     EXIT_ChkSerial

SerialSetTime: CJNE    A,#'T',SerialSetDate
               CALL    SerialWriteTime
               JMP     EXIT_ChkSerial

SerialSetDate: CJNE    A,#'D',SerialRunStop
               CALL    SerialWriteDate
               JMP     EXIT_ChkSerial

SerialRunStop: CJNE    A,#'G',SerialManulRec

```

	CALL	SerialWriteRunStop
	JMP	EXIT_ChkSerial
SerialManulRec:	CJNE	A,#'R',SerialManulStop
	CLR	P1.3
	CALL	DELAY_500M
	SETB	P1.3
	MOV	StatusRec,#00h
	JMP	EXIT_ChkSerial
SerialManulStop:	CJNE	A,#'S',EXIT_ChkSerial
	CLR	P1.4
	CALL	DELAY_500M
	SETB	P1.4
	MOV	StatusRec,#0FFh
EXIT_ChkSerial:	RET	
	;=====	
SerialWriteProgram:	MOV	A,Rx_00+2
	ANL	A,#0FH
	MOV	DayPro,A
	MOV	A,Rx_00+4
	ANL	A,#0FH
	SWAP	A
	MOV	B,A
	MOV	A,Rx_00+5
	ANL	A,#0FH

ORL A,B
MOV StepPro,A
MOV A,Rx_00+7
ANL A,#0FH
SWAP A
MOV B,A
MOV A,Rx_00+8
ANL A,#0FH
ORL A,B
MOV HourPro,A
MOV A,Rx_00+9
ANL A,#0FH
SWAP A
MOV B,A
MOV A,Rx_00+10
ANL A,#0FH
ORL A,B
MOV MinPro,A
MOV A,Rx_00+12
MOV BUF+0,A
MOV A,Rx_00+13
MOV BUF+1,A
MOV A,Rx_00+14
MOV BUF+2,A

```
MOV      A,Rx_00+15
MOV      BUF+3,A
MOV      A,Rx_00+16
MOV      BUF+4,A
MOV      A,Rx_00+17
MOV      BUF+5,A
MOV      A,Rx_00+18
MOV      BUF+6,A
MOV      A,Rx_00+19
MOV      BUF+7,A
CALL     WriteDataEEP
RET
SerialWriteTime: MOV      A,Rx_00+2
ANL     A,#0FH
SWAP    A
MOV     B,A
MOV     A,Rx_00+3
ANL     A,#0FH
ORL     A,B
MOV     HourInput,A
MOV     A,Rx_00+5
ANL     A,#0FH
SWAP    A
MOV     B,A
```

MOV A,Rx_00+6

ANL A,#0FH

ORL A,B

MOV MinInput,A

MOV A,Rx_00+8

ANL A,#0FH

SWAP A

MOV B,A

MOV A,Rx_00+9

ANL A,#0FH

ORL A,B

MOV SecInput,A

CALL WR_Time

RET

SerialWriteDate:

MOV A,Rx_00+2

ANL A,#0FH

MOV DayInput,A

MOV A,Rx_00+4

ANL A,#0FH

SWAP A

MOV B,A

MOV A,Rx_00+5

ANL A,#0FH

ORL A,B

```

MOV      DateInput,A
MOV      A,Rx_00+7
ANL      A,#0FH
SWAP     A
MOV      B,A
MOV      A,Rx_00+8
ANL      A,#0FH
ORL      A,B
MOV      MonthInput,A
MOV      A,Rx_00+10
ANL      A,#0FH
SWAP     A
MOV      B,A
MOV      A,Rx_00+11
ANL      A,#0FH
ORL      A,B
MOV      YearInput,A
CALL     WR_DATE
RET
SerialWriteRunStop: CALL     RunStopPro
RET
;*****

```

LCD Sub

```

CS_LCD      EQU      P2.4    ; E LCD
RS_LCD      EQU      P2.5    ; RS LCD
D4_LCD      EQU      P2.0
D5_LCD      EQU      P2.1
D6_LCD      EQU      P2.2
D7_LCD      EQU      P2.3

```

```

;*****

```

```

;      Input :      ACC (ASCII)

```

```

;      Output :     Data bus LCD

```

```

;*****

```

```

WR_LCD:      SETB      RS_LCD
              MOV       B,A
              ANL       A,#0F0H
              CALL      SEND_DATA4BIT
              LCALL     EN_LCD
              MOV       A,B                      ;Low byte
              SWAP      A
              ANL       A,#0F0H
              CALL      SEND_DATA4BIT
              LCALL     EN_LCD
              RET

```

```

;*****

```

```

;      Write Instruction LCD

```

```

;      Input :      ACC (Command)

```

; Output : Data bus LCD

```
WR_INS:           CLR        RS_LCD  
  
                  MOV        B,A  
  
                  ANL        A,#0F0H  
  
                  CALL       SEND_DATA4BIT  
  
                  LCALL     EN_LCD  
  
                  MOV        A,B                   ; Low byte  
  
                  SWAP     A  
  
                  ANL        A,#0F0H  
  
                  CALL       SEND_DATA4BIT  
  
                  LCALL     EN_LCD  
  
                  RET
```

```
SEND_DATA4BIT:    MOV        C,ACC.4  
  
                  MOV        D4_LCD,C  
  
                  MOV        C,ACC.5  
  
                  MOV        D5_LCD,C  
  
                  MOV        C,ACC.6  
  
                  MOV        D6_LCD,C  
  
                  MOV        C,ACC.7  
  
                  MOV        D7_LCD,C  
  
                  RET
```

; Goto position of LCD

; Input : ACC (addr.)

```
GOTO_LCD:      SETB      ACC.7
               LCALL     WR_INS
               RET
```

; Initial LCD

; 4-Bit Interface

```
INIT_LCD:      CLR       RS_LCD
               MOV       A,#33H      ; Set DL = 1 3-time
               LCALL    WR_INS
               MOV       A,#32H      ; Clear DL = 0 1-time
               LCALL    WR_INS
               MOV       A,#28H      ; Function set
               LCALL    WR_INS      ; DL=0 4Bit,N=1 2Line,F=0 5X7
               MOV       A,#0fH
               LCALL    WR_INS      ; Entry display,cursor off,cursor not
blink
               MOV       A,#06H      ; Entry mode set
               LCALL    WR_INS      ; I/D=1 Increment,S=0 cursor shift
               MOV       A,#01H      ; Clear display
               LCALL    WR_INS      ; Clear display,set DD RAM address=0
               RET
```

```

;*****
;
;      Enable Pin E LCD
;
;      Active Chip select
;*****

EN_LCD:      SETB      CS_LCD      ; Enable LCD
             LCALL     BUSY        ; Busy delay time
             CLR       CS_LCD      ; Disable LCD
             RET

;*****
;      Delay time for Busy
;
;      Wait LCD Ready
;*****

BUSY:        PUSH      07H
             PUSH      06H
             MOV       R6,#07H

BUSY1:       MOV       R7,#0FFH
             DJNZ     R7,$
             DJNZ     R6,BUSY1
             POP       06H
             POP       07H
             RET

```

DS1307 Sub

```

;#####
CONT_BYTE_RTC_W EQU 0D0H

```

CONT_BYTE_RTC_R EQU 0D1H

;* WRITE TIME TO RTC 1 BYTE

;* INPUT : DPL of Address , A of Data

WRITE_TIME: MOV B,A

WRITE_TIME0: CLR Port_DIO ;Start bit

CLR Port_SCK

_WR_TYPE_RTC: MOV A,#CONT_BYTE_RTC_W ;Send Control Byte

CALL BYTE_RTC_WR

SETB Port_DIO

SETB Port_SCK

JB Port_DIO,WRITE_TIME

CLR Port_SCK

_WR_ADD_RTC: MOV A,DPL ;Send address low

CALL BYTE_RTC_WR

SETB Port_DIO

SETB Port_SCK

JB Port_DIO,WRITE_TIME

CLR Port_SCK

_WR_DATA_RTC: MOV A,B ;Send data

CALL BYTE_RTC_WR

SETB Port_DIO

```

SETB      Port_SCK

JB        Port_DIO,WRITE_TIME

CLR       Port_DIO

CLR       Port_SCK

SETB      Port_SCK          ;STOP BIT

SETB      Port_DIO

RET

;*****
;*****

;*      Byte RTC WRITE

;*      INPUT : Acc

;*****
;*****

BYTE_RTC_WR:  PUSH    02H

MOV        R2,#08H          ;Data 8 Bits

_BYTE_RTC_WR:  RLC     A

MOV        Port_DIO,C

SETB      Port_SCK

CLR       Port_SCK

DJNZ      R2,_BYTE_RTC_WR

POP        02H

RET

;*****
;*****

;*      READ TIME FROM RTC 1 BYTE

```

;* INPUT : DPL of Address

;* OUT PUT : A of Data

;

READ_TIME: CLR Port_DIO ;Start bit

CLR Port_SCK

_RD_TYPE_RTC: MOV A,#CONT_BYTE_RTC_W

CALL BYTE_RTC_WR

SETB Port_DIO

SETB Port_SCK

JB Port_DIO,READ_TIME

CLR Port_SCK

_RD_ADD_RTC: MOV A,DPL

CALL BYTE_RTC_WR

SETB Port_DIO

SETB Port_SCK

JB Port_DIO,READ_TIME

CLR Port_SCK

SETB Port_SCK

CLR Port_DIO ;Start bit

CLR Port_SCK

_RD_ECHO_RTC: MOV A,#CONT_BYTE_RTC_R

CALL BYTE_RTC_WR

SETB Port_DIO

```

SETB      Port_SCK

JB        Port_DIO,READ_TIME

_RD_DATA_RTC: CLR      Port_SCK

CALL     BYTE_RTC_RD

SETB     Port_DIO

SETB     Port_SCK

CLR      Port_SCK

SETB     Port_SCK           ;Stop bit

SETB     Port_DIO

RET

```

```

;*****
;***

```

```

;*      Byte RTC READ
;*      OUT PUT : Acc

```

```

;*****
;***

```

```

BYTE_RTC_RD:  PUSH    02H

MOV         R2,#08H

_BYTE_RTC_RD: SETB    Port_SCK

MOV         C,Port_DIO

CLR         Port_SCK

RLC        A

DJNZ       R2,_BYTE_RTC_RD

POP        02H

RET

```

;

READ_DATE_TIME:

```
MOV        DPL,#06H
CALL       READ_TIME
MOV        YEAR,A
MOV        DPL,#05H
CALL       READ_TIME
MOV        MONTH,A
MOV        DPL,#04H
CALL       READ_TIME
MOV        DATE,A
MOV        DPL,#03H
CALL       READ_TIME
MOV        DAY,A
MOV        DPL,#02h      ;_Send_Hour:
CALL       READ_TIME
MOV        HOUR,A
MOV        DPL,#01h      ;_Send_Min
CALL       READ_TIME
MOV        MIN,A
MOV        DPL,#00h      ;_Send_Sec
CALL       READ_TIME
MOV        SEC,A
MOV        Last_Sec,A
RET
```

```
WR_DATE:  MOV  A,YearInput
          MOV  DPL,#06H
          CALL WRITE_TIME
          MOV  A,MonthInput
          MOV  DPL,#05H
          CALL WRITE_TIME
          MOV  A,DateInput
          MOV  DPL,#04H
          CALL WRITE_TIME
          MOV  A,DayInput
          MOV  DPL,#03H
          CALL WRITE_TIME
          RET
WR_Time:  MOV  A,HourInput
          MOV  DPL,#02H
          CALL WRITE_TIME
          MOV  A,MinInput
          MOV  DPL,#01H
          CALL WRITE_TIME
          MOV  A,SecInput
          MOV  DPL,#00H
          CALL WRITE_TIME
          RET
```


System.Sub

===== Initial Serial Port Using Timer1 =====

```
Initial_Serial:    MOV        TMOD,#00100000B        ;timer1 mode2
                  MOV        SCON,#01010000B    ;mode1 serial port
                  MOV        TH1,#0FBH
                  MOV        A,#00H
                  MOV        PCON,A            ;SMOD = 0
                  SETB       TR1                ;start timer1
                  RET
```

===== Send Data to Serial Port =====

```
SEND_2HEX:  PUSH   Acc
            SWAP   A
            CALL  SEND_1HEX
            POP   Acc
SEND_1HEX:  PUSH   DPH
            PUSH  DPL
            MOV   DPTR,#HEXASC_TAB
            ANL  A,#0FH
            MOVC A,@A+DPTR
            POP  DPL
            POP  DPH
SEND_ASCII: PUSH   IE
            CLR  EA
            CLR  ES
```

```

        CLR        TI

        MOV        SBUF,A

        JNB        TI,$

        CLR        TI

        SETB       EA

        SETB       ES

        POP        IE

        RET

HEXASC_TAB:  DB      '0123456789ABCDEF'

;===== Send Line Feed =====
Send_LineFeed:  MOV      A,#0DH      ; Carriage Return to left margin

                CALL     Send_ASCII

                MOV      A,#0AH      ; Line Feed

                CALL     Send_ASCII

                RET

;===== Send Line Feed =====

Send_ClearScreen:  MOV      A,#0CH;New page

                  CALL     Send_ASCII

                  RET

;===== Send data Table to Serial Start at DPTR to '00H' =====

Send_Table:      CLR        A

                  MOVC     A,@A+DPTR

                  JZ        _X_Send_Table

                  CALL     SEND_ASCII

```

```

                INC            DPTR

                JMP            SEND_TABLE

_X_Send_Table:  RET

;*****
;*Delay Subroutine
;*****

DELAY_2SEC:    CALL            DELAY_1SEC ;Delay 1 Sec

                CALL            DELAY_1SEC ;Delay 1 Sec

                RET

DELAY_1SEC:    CALL            DELAY_500M ;Delay 500 mSec

                CALL            DELAY_500M ;Delay 500 mSec

                RET

DELAY_500M:    CALL            DELAY100MS ;Delay 100 mSec

                CALL            DELAY100MS ;Delay 100 mSec

DELAY300MS:    CALL            DELAY100MS ;Delay 100 mSec

                CALL            DELAY100MS ;Delay 100 mSec

                ;CALL            DELAY100MS ;Delay 100 mSec

                ;CALL            DELAY100MS ;Delay 100 mSec

                RET

DELAY100MS:    MOV             R6,#200            ; 200 * 500 = 100mS

DLY100MS:     CALL            DELAY500U    ;Delay 500 MicroSec

                DJNZ            R6,DELAY100MS

                RET

DELAY10MS:     MOV             R6,#20            ; 20 * 500 = 10mS

```

```

DELAY10MS:    CALL     DELAY500U    ;Delay 500 MicroSec
              DJNZ     R6,DELAY10MS
              RET

```

```

DELAY500U:

```

```

              MOV     R7,#00h      ;1 MC
              DJNZ     R7,$         ;(230*2)MC *12/18.432
                                      ; = 299mSec
              MOV     R7,#00h      ;1 MC
              DJNZ     R7,$         ;(230*2)MC *12/18.432
                                      ; = 299mSec
              RET

```

```

;=====
;===== BCD TO HEX 1BYTE
;INPUT      = A
;OUTPUT     = A
;REG        = R1,A,B
;=====

```

```

BCDTOHEX:    PUSH     01H
              PUSH     B
              MOV     R1,A
              SWAP    A
              ANL     A,#0FH      ;MAX
              MOV     B,#10
              MUL     AB

```

```

AB1:          XCH      A,R1      ;MIN

              ANL      A,#0FH

              ADD      A,R1

              POP      B

              POP      01H

              RET

```

```

;===== HEX TO BCD =====

```

```

;INPUT      = A

```

```

;OUTPUT     = A

```

```

;REG        = A,B

```

```

;=====

```

```

HEXTOBCD:    PUSH     B
              MOV     B,#10
              DIV     AB
              SWAP    A
              ORL    A,B
              POP     B
              RET

```

24256.Sub (EEPROM)

```

CONT_BYTE_W EQU      0A8H

```

```

CONT_BYTE_R EQU      0A9H

```

```

;===== EEPROM_Writes =====

```

```

;INPUT :DPTR (Address),Acc

```

;OUT :-

;Reg. :Acc,DPTR

;

```
EEPROM_Write:  MOV      ADD_HIGH,DPH
                MOV      ADD_LOW,DPL
                MOV      DATA,A
```

;

;*WRITE DATA TO EEPROM 1 BYTE *

;*INPUT : ADD_HIGH *

;*: ADD_LOW *

;*: DATA *

;

```
WRITE_BYTE: CLR      Port_DIO      ;start bit
              CLR      Port_SCK
              MOV      A,#CONT_BYTE_W ;send control byte
              LCALL    LOOP_BYTE
              SETB     Port_DIO
              SETB     Port_SCK
              JB       P4.1 ,WRITE_BYTE ;loop until busy1
              CLR      Port_SCK
              MOV      A,ADD_HIGH    ;send address low
              LCALL    LOOP_BYTE
              SETB     Port_DIO
              SETB     Port_SCK
```

```

JB      P4.1 ,WRITE_BYTE      ;loop until busy

CLR     Port_SCK

MOV     A,ADD_LOW             ;send address low

LCALL  LOOP_BYTE

SETB    Port_DIO

SETB    Port_SCK

JB      P4.1 ,WRITE_BYTE      ;loop until busy

CLR     Port_SCK

MOV     A,DATA                ;send data

LCALL  LOOP_BYTE

SETB    Port_DIO

SETB    Port_SCK

JB      P4.1 ,WRITE_BYTE      ;loop until busy

CLR     Port_DIO

CLR     Port_SCK

SETB    Port_SCK              ;stop bit

SETB    Port_DIO

LCALL  DELAY4M

RET

```

```

;===== EEPROM_Read =====

```

```

;INPUT : DPTR (Address)

```

```

;OUT   : Acc

```

```

;Reg.  :Acc,DPTR

```

```

;=====

```

EEPROM_Read:

MOV ADD_HIGH,DPH

MOV ADD_LOW,DPL

;*READ DATA FROM EEPROM 1 BYTE *

;* INPUT : ADD_HIGH *

;*ADD_LOW *

;* OUTPUT : DATA *

READ_BYTE: CLR Port_DIO ;start bit

CLR Port_SCK

MOV A,#CONT_BYTE_W ;send control byte

LCALL LOOP_BYTE

SETB Port_DIO

SETB Port_SCK

JB P4.1,READ_BYTE ;loop until busy

CLR Port_SCK

MOV A,ADD_HIGH ;send address low

LCALL LOOP_BYTE

SETB Port_DIO

SETB Port_SCK

JB P4.1,READ_BYTE ;loop until busy

CLR Port_SCK

MOV A,ADD_LOW ;send address low


```

LCALL    LOOP_BYTE

SETB     Port_DIO

SETB     Port_SCK

JB       P4.1 ,READ_BYTE    ;loop until busy

CLR      Port_SCK

SETB     Port_SCK

SETB     Port_DIO

CLR      Port_DIO          ;start bit

CLR      Port_SCK

MOV      A,#CONT_BYTE_R    ;send control byte

LCALL    LOOP_BYTE

SETB     Port_DIO

SETB     Port_SCK

JB       P4.1 ,READ_BYTE    ;loop until busy

CLR      Port_SCK

LCALL    LOOP_READ

SETB     Port_DIO

SETB     Port_SCK

CLR      Port_SCK

SETB     Port_SCK          ;stop bit

SETB     Port_DIO

MOV      A,DATA

RET

```

,*****

```

;*WRITE          *
;*INPUT: ACC     *

;*****
;
LOOP_BYTE:      PUSH      2H
                MOV       R2,#08H

LOOP_SEND:      RLC       A
                MOV       P4.1,C
                SETB      Port_SCK
                CLR       Port_SCK
                DJNZ      R2,LOOP_SEND
                POP       02H
                RET

;*****
;*READ          *
;* OUTPUT: ACC  *

;*****
;
LOOP_READ:      PUSH      02H
                MOV       R2,#08H

LOOP_READ1:     SETB      Port_SCK
                MOV       C,Port_DIO
                CLR       Port_SCK
                RLC       A
                DJNZ      R2,LOOP_READ1
                MOV       DATA,A

```

```

                POP        02H

                RET

DELAY4M:      PUSH        DPH

                PUSH        DPL

                MOV        DPTR,#0FFF0H

DEL4:         INC         DPTR

                MOV        A,DPL

                ORL        A,DPH

                JNZ        DEL4

                POP        DPL

                POP        DPH

                RET

```

CheckPro.Sub (ตรวจสอบโปรแกรม)

```

ChkProGram:   MOV        StepPro,#00H

_ChkPro0:     MOV        DayPro,DAY

                CALL        StartAddresIndex

                CALL        ReadDataEEP

                MOV        A,HOUR

                XRL        A,HourPro

                JNZ        NextStepPro

                MOV        A,MIN

                XRL        A,MinPro

                JNZ        NextStepPro

```

```

                CALL        On_OutPut
NextStepPro:   INC          StepPro
                MOV         A,StepPro
                CLR         C
                SUBB        A,#21H
                JC          _ChkPro0
                RET
On_OutPut:     MOV         A,DataPro
                JB          ACC.7,chkStopRec
                CALL        RemoteRec
                JMP         Exit_On_OutPut
chkStopRec:   CALL        StopRec
Exit_On_OutPut: RET
;
RemoteRec:    CLR         P1.3
                CALL        DELAY_500M
                SETB        P1.3
                MOV         StatusRec,#00h
exit_RemoteRec: RET
StopRec:      CLR         P1.4
                CALL        DELAY_500M
                SETB        P1.4
                MOV         StatusRec,#0FFh
exit_RemoteStop: RET

```

Command.Sub

```
=====
;
;           CHK_COMMAND
=====

CHK_COMMAND:  ;MOV      A,DATA_KEY

              ;XRL      A,#0AH

              ;JZ       _CHK_Run

              ;#####

              ;CALL     ManualON

              ;#####

              ;JMP      _EXIT_COMM

_CHK_Run:    MOV      A,DATA_KEY

            CJNE     A,#0BH,_CHK_Rec

            CALL     DELAY300MS

            CALL     RunStopPro

_CHK_Rec:    MOV      A,DATA_KEY

            CJNE     A,#00H,_CHK_DATE

            CALL     ManualRecStop

            CALL     DELAY300MS

_CHK_DATE:  MOV      A,DATA_KEY

            CJNE     A,#0AH,_CHK_TIME

            CALL     DELAY300MS

            CALL     SetDate
```

```

_CHK_TIME: MOV      A,DATA_KEY

           CJNE     A,#0AH,_CHK_PGM

           CALL    DELAY300MS

           CALL    SetTime

_CHK_PGM:  MOV      A,DATA_KEY

           CJNE     A,#0AH,_EXIT_COMM

           CALL    DELAY300MS

           CALL    SetProgram

_EXIT_COMM:MOV     A,#01H

           CALL    WR_INS

           CALL    READ_DATE_TIME

           CALL    ShowTimeLCD

           CALL    DELAY300MS

           RET

;=====
;           SetDate
;=====

SetDate:  CALL     ShowSetDate

           CALL    DELAY300MS

_SetDate00: CALL    SCAN_F

           MOV     B,A

           CJNE    A,#0AH,_SetDate001

           JMP     _exit_wr_date

_SetDate001: CJNE   A,#00H,_SetDate002

```

```

                JMP          SETDATE
_SetDate002:   CLR          C
                SUBB       A,#08H
                JNC        SETDATE
                MOV        A,B
                MOV        DayInput,A
                ORL        A,#30h
                CALL       WR_LCD
                MOV        A,#'/'
                CALL       WR_LCD
                CALL       DELAY300MS
_SetDate01:   CALL       SCAN_F
                SWAP       A
                MOV        DateInput,A
                SWAP       A
                ORL        A,#30h
                CALL       WR_LCD
                CALL       DELAY300MS
                CALL       SCAN_F
                ORL        A,DateInput
                MOV        DateInput,A
                ANL        A,#0FH
                ORL        A,#30H
                CALL       WR_LCD

```

```
MOV      A,#' '  
CALL    WR_LCD  
CALL    DELAY300MS  
CALL    SCAN_F  
SWAP    A  
MOV     MonthInput,A  
SWAP    A  
ORL     A,#30h  
CALL    WR_LCD  
CALL    DELAY300MS  
CALL    SCAN_F  
ORL     A,MonthInput  
MOV     MonthInput,A  
ANL    A,#0FH  
ORL     A,#30H  
CALL    WR_LCD  
MOV     A,#' '  
CALL    WR_LCD  
CALL    DELAY300MS  
CALL    SCAN_F  
SWAP    A  
MOV     YearInput,A  
SWAP    A  
ORL     A,#30h
```



```

CALL    WR_LCD

CALL    DELAY300MS

CALL    SCAN_F

ORL     A,YearInput

MOV     YearInput,A

ANL     A,#0FH

ORL     A,#30H

CALL    WR_LCD

CALL    DELAY300MS

CALL    SCAN_F

MOV     B,A

XRL     A,#0BH

JZ      _WR_DATE

MOV     A,#41H

CALL    GOTO_LCD

CALL    DELAY300MS

JMP     _SetDate00

_WR_DATE: CALL  WR_DATE

CALL    DELAY300MS

MOV     A,#01H           ; Clear display

LCALL   WR_INS

_exit_wr_date: RET

;=====

;           SetTime

```

;=====

```
SetTime:    CALL    ShowSetTime
            CALL    DELAY300MS
_SetTime01: CALL    SCAN_F
            CJNE   A,#0AH,_SetTime001
            JMP    _exit_wr_time
_SetTime001: SWAP   A
            MOV    HourInput,A
            SWAP   A
            ORL   A,#30h
            CALL   WR_LCD
            CALL   DELAY300MS
            CALL   SCAN_F
            ORL   A,HourInput
            MOV    HourInput,A
            ANL   A,#0FH
            ORL   A,#30H
            CALL   WR_LCD
            MOV    A,#':'
            CALL   WR_LCD
            CALL   DELAY300MS
            CALL   SCAN_F
            SWAP   A
            MOV    MinInput,A
```

```
SWAP    A
ORL     A,#30h
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
ORL     A,MinInput
MOV     MinInput,A
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,#'?'
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
SWAP    A
MOV     SecInput,A
SWAP    A
ORL     A,#30h
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
ORL     A,SecInput
MOV     SecInput,A
ANL     A,#0FH
```

```

        ORL        A,#30H

        CALL       WR_LCD

        CALL       DELAY300MS

        CALL       SCAN_F

        MOV        B,A

        XRL        A,#0BH

        JZ         _WR_SetTime

        MOV        A,#41H

        CALL       GOTO_LCD

        CALL       DELAY300MS

        JMP        _SetTime01

_WR_SetTime: CALL    WR_TIME

        CALL       DELAY300MS

        MOV        A,#01H ; Clear display

        LCALL     WR_INS

_exit_wr_time: RET

;=====

;          SetProgram

;=====

SetProgram: CALL    ShowSetProgram

           CALL    DELAY300MS

_SetPro00: CALL    SCAN_F

           CJNE   A,#0AH,_SetPro01

           JMP    _exit_SetProg

```

```

_SetPro01:  CJNE      A,#00H,_SetPro02
            JMP      SetProgram

_SetPro02:  CLR      C
            SUBB     A,#08H
            JNC     SetProgram
            MOV     A,DATA_KEY
            MOV     DayPro,A
            ORL     A,#30h
            CALL    WR_LCD
            CALL    DELAY300MS
            CALL    SCAN_F
            CJNE   A,#0BH,_SetPro03
            ;#####
            CALL    SetPro
            ;#####
            JMP     _exit_SetProg

_SetPro03:  MOV     A,#49H
            CALL    GOTO_LCD
            CALL    DELAY300MS
            JMP     _SetPro00

_exit_SetProg:  RET

RunStopPro: SETB     P2.6
            MOV     A,DataSetRunPro
            CPL     A

```

```

MOV          DataSetRunPro,A

MOV          DPTR,#02FEH

CALL        EEPROM_Write

RET

ManualRecStop: MOV      A,StatusRec

JZ          MnlStopRec

CALL        RemoteRec

JMP         ExitManualRecStop

MnlStopRec:  CALL        StopRec

ExitManualRecStop:  RET

Display.Sub

;*****
;
;          ShowTimeLCD
;*****

ShowTimeLCD: MOV      A,#00H

CALL        GOTO_LCD

MOV         A,DAY

CJNE       A,#01H,_Mon

MOV        DPTR,#DaySun

JMP        _LCD_Send_Table

_Mon:      CJNE       A,#02H,_Tue

MOV        DPTR,#DayMon

JMP        _LCD_Send_Table

_Tue:      CJNE       A,#03H,_Wed

```

```

MOV DPTR,#DayTue
JMP _LCD_Send_Table
_Wed: CJNE A,#04H,_Thu
MOV DPTR,#DayWed
JMP _LCD_Send_Table
_Thu: CJNE A,#05H,_Fri
MOV DPTR,#DayThu
JMP _LCD_Send_Table
_Fri: CJNE A,#06H,_Sat
MOV DPTR,#DayFri
JMP _LCD_Send_Table
_Sat: MOV DPTR,#DaySat
JMP _LCD_Send_Table
_LCD_Send_Table: CALL LCD_Send_Table
MOV A,#' '
CALL WR_LCD
MOV A,DATE
SWAP A
ANL A,#0FH
ORL A,#30H
CALL WR_LCD
MOV A,DATE
ANL A,#0FH
ORL A,#30H

```

```
CALL    WR_LCD
MOV     A,#'
CALL    WR_LCD
MOV     A,MONTH
SWAP    A
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,MONTH
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,#'
CALL    WR_LCD
MOV     A,YEAR
SWAP    A
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,YEAR
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,#'

```



```

CALL    WR_LCD
MOV     A,DataSetRunPro
CJNE   A,#'R',NotShowRun
MOV     A,#'R'
CALL    WR_LCD
MOV     A,#'u'
CALL    WR_LCD
MOV     A,#'n'
CALL    WR_LCD
JMP     ShowLowerLine
NotShowRun:
MOV     A,#'!'
CALL    WR_LCD
MOV     A,#'!'
CALL    WR_LCD
MOV     A,#'!'
CALL    WR_LCD
ShowLowerLine:
MOV     A,#44H
CALL    GOTO_LCD
MOV     A,HOUR
SWAP   A
ANL    A,#0FH
ORL    A,#30H
CALL    WR_LCD
MOV     A,HOUR

```

ANL A,#0FH
ORL A,#30H
CALL WR_LCD
MOV A,#'
CALL WR_LCD
MOV A,MIN
SWAP A
ANL A,#0FH
ORL A,#30H
CALL WR_LCD
MOV A,MIN
ANL A,#0FH
ORL A,#30H
CALL WR_LCD
MOV A,#'
CALL WR_LCD
MOV A,SEC
SWAP A
ANL A,#0FH
ORL A,#30H
CALL WR_LCD
MOV A,SEC
ANL A,#0FH
ORL A,#30H

```

CALL WR_LCD
MOV A,StatusRec
JZ ShowLcdRec
MOV A,#' '
CALL WR_LCD
MOV A,#' '
CALL WR_LCD
MOV A,#' '
CALL WR_LCD
MOV A,#' '
CALL WR_LCD
MOV A,#' '
CALL WR_LCD
MOV A,#' '
CALL WR_LCD
JMP ExitShowTimeLCD
ShowLcdRec: MOV A,#' '
CALL WR_LCD
MOV A,#'R'
CALL WR_LCD
MOV A,#'e'
CALL WR_LCD
MOV A,#'c'
CALL WR_LCD
ExitShowTimeLCD: RET

```

```

*****
,

```

; ShowTimeSerial

;

```
ShowTimeSerial:  CALL    Send_LineFeed
                  MOV     A,DAY
                  CALL    SEND_1HEX
                  MOV     A,#' '
                  CALL    SEND_ASCII
                  MOV     A,DATE
                  CALL    SEND_2HEX
                  MOV     A,#'/'
                  CALL    SEND_ASCII
                  MOV     A,MONTH
                  CALL    SEND_2HEX
                  MOV     A,#'/'
                  CALL    SEND_ASCII
                  MOV     A,YEAR
                  CALL    SEND_2HEX
                  MOV     A,#' '
                  CALL    SEND_ASCII
                  MOV     A,HOUR
                  CALL    SEND_2HEX
                  MOV     A,#':'
                  CALL    SEND_ASCII
                  MOV     A,MIN
```

```
CALL SEND_2HEX
MOV A,#'
CALL SEND_ASCII
MOV A,SEC
CALL SEND_2HEX
MOV A,#'
CALL SEND_ASCII
MOV A,DataSetRunPro
CJNE A,#'R',NotShowRunSerial
MOV A,#'R'
CALL SEND_ASCII
MOV A,#'n'
CALL SEND_ASCII
MOV A,#'n'
CALL SEND_ASCII
MOV A,#'
CALL SEND_ASCII
JMP ShowSerialRec
NotShowRunSerial: MOV A,#'-
CALL SEND_ASCII
MOV A,#'-
CALL SEND_ASCII
MOV A,#'-
CALL SEND_ASCII
```

```

MOV      A,#' '
CALL     SEND_ASCII

ShowSerialRec:
MOV      A,StatusRec
JZ       StatusSerialRec

MOV      A,#'- '
CALL     SEND_ASCII

MOV      A,#'- '
CALL     SEND_ASCII

MOV      A,#'- '
CALL     SEND_ASCII

JMP      ExitShowTimeSerial

StatusSerialRec:
MOV      A,#'R'
CALL     SEND_ASCII

MOV      A,#'e'
CALL     SEND_ASCII

MOV      A,#'c'
CALL     SEND_ASCII

ExitShowTimeSerial:
RET

;*****
;
;           ShowSetDate
;*****

ShowSetDate:
MOV      A,#01H      ; Clear display

LCALL   WR_INS

MOV      DPTR,#SetDateTable

```

```

CALL    LCD_Send_Table

MOV     A,#40H

CALL    GOTO_LCD

MOV     A,#>'

CALL    WR_LCD

RET

```

```

;*****
;

```

```

ShowSetTime

```

```

;*****
;

```

```

ShowSetTime:  MOV     A,#01H      ; Clear display
               LCALL    WR_INS
               MOV     DPTR,#SetTimeTable
               CALL    LCD_Send_Table
               MOV     A,#40H
               CALL    GOTO_LCD
               MOV     A,#>'
               CALL    WR_LCD

               RET

```

```

;*****
;

```

```

ShowSetProgram

```

```

;*****
;

```

```

ShowSetProgram :MOV     A,#01H      ; Clear display
                LCALL    WR_INS
                MOV     DPTR,#SetProTable

```

```

CALL    LCD_Send_Table

MOV     A,#40H

CALL    GOTO_LCD

MOV     DPTR,#SelectTable

CALL    LCD_Send_Table

MOV     A,#49H

CALL    GOTO_LCD

RET

```

```

;*****
;

```

```

;          ShowFormInput
;*****

```

```

ShowFormInput:  MOV     A,#01H      ; Clear display
                LCALL   WR_INS
                MOV     A,#'D'
                CALL    WR_LCD
                MOV     A,DayPro
                ORL     A,#30H
                CALL    WR_LCD
                MOV     A,#03H
                CALL    GOTO_LCD
                MOV     A,#'P'
                CALL    WR_LCD
                MOV     A,StepPro
                SWAP    A

```



```
ANL      A,#0FH
ORL      A,#30H
CALL     WR_LCD
MOV      A,StepPro
ANL      A,#0FH
ORL      A,#30H
CALL     WR_LCD
MOV      A,#07H
CALL     GOTO_LCD
MOV      DPTR,#NUMBER
CALL     LCD_Send_Table
MOV      A,#41H
CALL     GOTO_LCD
MOV      A,HourPro
SWAP     A
ANL      A,#0FH
ORL      A,#30H
CALL     WR_LCD
MOV      A,HourPro
ANL      A,#0FH
ORL      A,#30H
CALL     WR_LCD
MOV      A,#':'
CALL     WR_LCD
```

```

MOV      A,MinPro
SWAP    A
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,MinPro
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
CALL    Show_Ch_Control
MOV     A,#40H
CALL    GOTO_LCD
RET
LCD_Send_Table: CLR     A
MOVC    A,@A+DPTR
JZ      _X_LCD_Table
CALL    WR_LCD
INC     LCD_SEND_TABLE
_X_LCD_Table: RET
Show_Ch_Control: MOV     A,#47H
CALL    GOTO_LCD
MOV     A,DataPro
MOV     C,ACC.0
CALL    Ch_Status

```

MOV A,DataPro

MOV C,ACC.1

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.2

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.3

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.4

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.5

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.6

CALL Ch_Status

MOV A,DataPro

MOV C,ACC.7

CALL Ch_Status

RET

Ch_Status:

MOV A,#' _'

JC Ch_Show_LCD

	MOV	A,#'r'
Ch_Show_LCD:	CALL	WR_LCD
	RET	
DaySun:	DB	'Sun',00H
DayMon:	DB	'Mon',00H
DayTue:	DB	'Tue',00H
DayWed:	DB	'Wed',00H
DayThu:	DB	'Thu',00H
DayFri:	DB	'Fri',00H
DaySat:	DB	'Sat',00H
TimeTable:	DB	'Time->',00H
DateTable:	DB	'Date->',00H
SetDateTable:	DB	' D dd/mm/yy ',00H
SetTimeTable:	DB	' hh:mm:ss ',00H
SetProTable:	DB	'Set Programs ',00H
SelectTable:	DB	'Day(1-7)=',00H
NUMBER:	DB	'12345678',00H
TIME_IN_TAB:	DB	'hh:mm',00H
Profirst.Sub		
ProGramFistTime:	MOV	StepPro,#00H
_ProFistTime0:	MOV	DayPro,DAY
	CALL	StartAddresIndex
	CALL	ReadDataEEP

```
MOV      X1DPH,DPH
MOV      X1DPL,DPL
MOV      A,HourPro
SUBB     A,#24H      ; Hour Over ?=3FH
JNC      EX_ProFistTime
MOV      HourStepX,HourPro
MOV      MinStepX,MinPro
MOV      DataX,DataPro
MOV      DPH,X1DPH
MOV      DPL,X1DPL
INC      DPTR
CALL     ReadDataEEP
MOV      A,HourPro
SUBB     A,#24H      ; Hour Over ?3FH
JNC      EX_ProFistTime
MOV      HourStepX1,HourPro
MOV      MinStepX1,MinPro
MOV      DataX1,DataPro
CALL     FistTimeStepX
MOV      A,StepPro
ADD      A,#01H
DA       A
MOV      StepPro,A
JMP      _ProFistTime0
```

EX_ProFistTime:	RET	
FistTimeStepX:	MOV	A,HourStepX
	CJNE	A,Hour,FistTimeStep0
	CLR	C
	MOV	A,MinStepX
	SUBB	A,MIN
	JNC	FistTimeStep0
	CLR	C
	MOV	A,MinStepX
	SUBB	A,MIN
	JNC	FistTimeStep0
	MOV	DataPro,DataX
	CALL	On_OutPut
	JMP	EXIT_FistTimeStepX
FistTimeStep0:	MOV	A,HourStepX1
	CJNE	A,Hour,FistTimeStep1
	CLR	C
	MOV	A,MinStepX1
	SUBB	A,MIN
	JNC	FistTimeStep1
	CLR	C
	MOV	A,MinStepX1
	SUBB	A,MIN
	JNC	FistTimeStep1

```

MOV      DataPro,DataX1
CALL    On_OutPut
JMP     EXIT_FistTimeStepX
FistTimeStep1:
CLR     C
MOV     A,HourStepX
SUBB   A,HOUR
JNC    EXIT_FistTimeStepX
CLR     C
MOV     A,HourStepX1
SUBB   A,HOUR
JNC    EXIT_FistTimeStepX
MOV     DataPro,DataX
CALL    On_OutPut
EXIT_FistTimeStepX:
RET
Scankey.Sub
Row1    EQU   P0.0
Row2    EQU   P0.1
Row3    EQU   P0.2
Row4    EQU   P0.3
Col1    EQU   P0.4
Col2    EQU   P0.5
Col3    EQU   P0.6

```

; Input: -

; Output: Acc of Key Code, 0FFH is not Key Press

; Reg: DPTR,A

;*****

SCAN_KEY: MOV B,#00H ; Start of Value = 0

_SCAN_Key_C1: CLR Col1

SETB Col2

SETB Col3

JNB Row4,_X_SCAN_Key ; Is Row.4

INC B

JNB Row3,_X_SCAN_Key ; Is Row.3

INC B

JNB Row2,_X_SCAN_Key ; Is Row.2

INC B

JNB Row1,_X_SCAN_Key ; Is Row.1

INC B

_SCAN_Key_C2: SETB Col1

CLR Col2

SETB Col3

JNB Row4,_X_SCAN_Key ; Is Row.4

INC B

JNB Row3,_X_SCAN_Key ; Is Row.3

INC B

JNB Row2,_X_SCAN_Key ; Is Row.2

INC B


```

JNB      Row1,_X_SCAN_Key      ; Is Row.1
INC      B
_SCAN_Key_C3:
SETB     Col1
SETB     Col2
CLR      Col3
JNB      Row4,_X_SCAN_Key      ; Is Row.4
INC      B
JNB      Row3,_X_SCAN_Key      ; Is Row.3
INC      B
JNB      Row2,_X_SCAN_Key      ; Is Row.2
INC      B
JNB      Row1,_X_SCAN_Key      ; Is Row.1
INC      B
_X_SCAN_Key:
MOV      A,B
_Encode_Key:
MOV      DPTR,#Table_Key_Code
MOVC     A,@A+DPTR
MOV      DATA_KEY,A
RET
Table_Key_Code:
DB       0AH,07H,04H,01H,00H,08H,05H,02H
DB       0BH,09H,06H,03H,0FFH
SCAN_F:
CALL     SCAN_KEY      ;"ĐÍ;"Ò; scan_f àÁ×èÍÁÕ;ÒÃ;´ key
CPL     A      ;μÃÇ"ÊÍ°ÇèÒÁÕ;ÒÃ;´key ËÃ×ÍäÁè
JZ      SCAN_F
CPL     A

```

```

MOV          DATA_KEY,A

RET

SetPro.Sub

SetPro:      MOV          StepPro,#00H

_SetPr_0:    CALL          StartAdresIndex

              CALL          ReadDataEEP

              CALL          ShowFormInput

              CALL          DELAY300MS

_SetPr_1:    CALL          SCAN_F

              CJNE         A,#09H,_SetPr_2    ;-->

              MOV          A,StepPro

              add          a,#1

              add          a

              MOV          StepPro,A

              CJNE         A,#21H,_SetPr_0

              JMP          SetPro

_SetPr_2:    MOV          A,DATA_KEY

              CJNE         A,#06H,_SetPr_4

              MOV          A,StepPro

              CJNE         A,#00h,_SetPr_3

              JMP          SetPro

_SetPr_3:    MOV          A,StepPro

              CALL          BCDtoHEX

```

```

DEC      A
CALL    HEXtoBCD
MOV     StepPro,A
JMP     _SetPr_0
_SetPr_4: CJNE   A,#00H,_SetPr_5      ;CLR Memory
CALL    ResetEEPROM
JMP     _SetPr_0
_SetPr_5: CJNE   A,#0AH,_SetPr_6
JMP     _exitSetPr
_SetPr_6: CJNE   A,#0BH,_SetPr_1
CALL    SetTimeChControl
MOV     A,StepPro
add     add,#1
MOV     StepPro,A
CJNE   A,#21H,_SetPr_7
_SetPr_7: JMP     _SetPr_0
_exitSetPr: RET

```

```

SetTimeChControl: MOV     A,#41H
CALL    GOTO_LCD
CALL    DELAY300MS
CALL    SCAN_F
SWAP   A
MOV     HourPro,A

```

```
SWAP    A
ORL     A,#30H
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
ORL     A,HourPro
MOV     HourPro,A
ANL     A,#0FH
ORL     A,#30H
CALL    WR_LCD
MOV     A,#'!'
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
SWAP    A
MOV     MinPro,A
SWAP    A
ORL     A,#30H
CALL    WR_LCD
CALL    DELAY300MS
CALL    SCAN_F
ORL     A,MinPro
MOV     MinPro,A
ANL     A,#0FH
```

```

                                ORL            A,#30H

                                CALL           WR_LCD

                                MOV            A,#' '

                                CALL           WR_LCD

                                CALL           DELAY300MS

                                MOV            R0,#BUF

                                MOV            R1,#08H

_SetTCC0:                       CALL           SCAN_F

                                MOV            A,DATA_KEY

                                CJNE          A,#0BH,_CHK_00

                                MOV            A,#'_'

                                JMP            _SetTCC1

_Chk_00:                         CJNE          A,#00H,_SetTCC0

                                MOV            A,#'r'

_SetTCC1:                         MOV            @R0,A

                                INC            R0

                                CALL           WR_LCD

                                CALL           DELAY300MS

                                DJNZ          R1,_SetTCC0

                                CALL           SCAN_F

                                CJNE          A,#0BH,SetTimeChControl ;Enter

                                CALL           WriteDataEEP

                                RET

```

,*****

```

;          StartAddrIndex

;Input  DayPro,StepPro

;Out    DPTR

;Reg.   Acc,B,DPTR

;(DayPro-1)*2----->MOVC----->StartAddrHI

;{(DayPro-1)*2}+1-->MOVC-->StartAddrLO

;

;(StepPro)*3=BA

;DPL=StartAddrLO+A

;DPH=StartAddrHI+B+C

.*****

StartAddrIndex:  MOV     A,DayPro
                  DEC     A
                  MOV     B,#2
                  MUL     AB
                  MOV     DPTR,#StartAddTAB
                  MOVC    A,@A+DPTR
                  MOV     AddHI,A ;(DayPro-1)*2----->MOVC-----
>StartAddrHI

                  MOV     A,DayPro
                  DEC     A
                  MOV     B,#2
                  MUL     AB
                  INC     A
                  MOV     DPTR,#StartAddTAB

```

```

MOV      A,@A+DPTR

MOV      AddLo,A;{(DayPro-1)*2}+1-->MOVC-->StartAddrLO

MOV      A,StepPro

MOV      B,#3

MUL      AB

CLR      C

ADD      A,AddLo

MOV      DPL,A

MOV      A,B

ADDC     A,AddHI

MOV      DPH,A

RET

;Day 1 2 3 4 5 6 7

StartAddTAB:  DW  0000H,0100H,0200H,0300H,0400H,0500H,0600H

;*****

;          ReadDataEEP

;input:  DPTR,StepPro

;output: StepPro,HourPro,MinPro,DataPro

;Reg.   acc,dptr

;*****

ReadDataEEP:

CALL     EEPROM_Read      ;read EEPROM

MOV      HourPro,A

INC      DPTR

```

```

CALL    EEPROM_Read    ;read EEPROM

MOV     MinPro,A

INC     DPTR

CALL    EEPROM_Read    ;read EEPROM

MOV     DataPro,A

RET

;*****
;
;          WriteDataEEP
;input:  StepPro,DayPro,HourPro,MinPro
;output:
;Reg.
;*****

WriteDataEEP:  CALL    StartAddressIndex
               MOV     A,HourPro
               CALL    EEPROM_Write
               INC     DPTR
               MOV     A,MinPro
               CALL    EEPROM_Write

ConWriteData:  MOV     A,BUF
               CLR     B.0
               CJNE   A,#'_',_B1
               SETB   B.0

_B1:          MOV     A,BUF+1
               CLR     B.1

```



```

                                CJNE    A,#' ',_B2
                                SETB    B.1
_B2:                            MOV     A,BUF+2
                                CLR     B.2
                                CJNE    A,#' ',_B3
                                SETB    B.2
_B3:                            MOV     A,BUF+3
                                CLR     B.3
                                CJNE    A,#' ',_B4
                                SETB    B.3
_B4:                            MOV     A,BUF+4
                                CLR     B.4
                                CJNE    A,#' ',_B5
                                SETB    B.4
_B5:                            MOV     A,BUF+5
                                CLR     B.5
                                CJNE    A,#' ',_B6
                                SETB    B.5
_B6:                            MOV     A,BUF+6
                                CLR     B.6
                                CJNE    A,#' ',_B7
                                SETB    B.6
_B7:                            MOV     A,BUF+7
                                CLR     B.7
```

```

                                CJNE      A,#' ',_BE
                                SETB     B.7
_BE:                            MOV      A,B
                                INC      DPTR
                                CALL     EEPROM_Write
                                RET

```

```

ResetEEPROM:                   MOV      A,#' '
                                MOV      BUF+0,A
                                MOV      BUF+1,A
                                MOV      BUF+2,A
                                MOV      BUF+3,A
                                MOV      BUF+4,A
                                MOV      BUF+5,A
                                MOV      BUF+6,A
                                MOV      BUF+7,A
                                MOV      HourPro,#0FFH
                                MOV      MinPro,#0FFH
                                CALL     WriteDataEEP
                                RET

```

Wch_Sub

WMCON register	EQU	96H	;watchdog and memory control
EEMEN	EQU	00001000B	;EEPROM access enable bit
EEMWE	EQU	00010000B	;EEPROM write enable bit
WDTRST	EQU	00000010B	;EEPROM RDYor /BSY bit

```

WDTEN          EQU          00000001B      ;watchdog timer enable bit

PS0            EQU          00100000B      ;watchdog timer period select bits

PS1            EQU          01000000B      ;

PS2            EQU          10000000B      ;

Enable_WatchDog:  ORL          WMCON,#PS0

                 ORL          WMCON,#PS1

                 ORL          WMCON,#PS2

                 ORL          WMCON,#WDTEN      ;enable watchdog

                 RET

Clear_WatchDog:  ORL          WMCON,#WDTRST      ;keep watchdog at bay

                 RET

;===== EEPROM_Read =====
;INPUT : DPTR (Address)
;OUT   : Acc
;Reg.  :Acc,DPTR

;=====

EEPROM_Read:    ORL          WMCON,#EEMEN      ;enable EEPROM access

                 MOVX         A,@DPTR          ;read EEPROM

                 XRL          WMCON,#EEMEN      ;disable EEPROM access

                 RET

;===== EEPROM_Writes =====

;INPUT :DPTR (Address),Acc
;OUT   : -
;Reg.  :Acc,DPTR

```

```

=====
EEPROM_Write:    ORL        WMCON,#EEMEN    ;enable EEPROM access

                 ORL        WMCON,#EEMWE    ;enable EEPROM writes

                 MOVX       @DPTR,A

                 CALL       DELAY10MS    ;wait 10 ms

                 XRL        WMCON,#EEMWE    ;disable EEPROM writes

                 XRL        WMCON,#EEMEN    ;disable EEPROM access

                 RET

```

Visual Basic Code

Option Explicit ‘ใช้เขียนบรรทัดแรกของ form มีความหมายว่า ตัวแปรทุกตัวที่ใช้ใน form นั้น ต้องมีการประกาศตัวแปรก่อนเสมอ

```
Private Sub Command1_Click()
```

‘คำสั่งในการตั้งเวลา

```
Winsock1.SendData "T "
```

```
Winsock1.SendData InputTime.Text
```

```
Winsock1.SendData vbCrLf
```

```
End Sub
```

```
Private Sub Command2_Click()
```

‘คำสั่ง RUN และ STOP

```
Winsock1.SendData "G"
```

```
Winsock1.SendData vbCrLf
```

```
End Sub
```

```
Private Sub Command3_Click()
```

‘คำสั่งบันทึก

```
Winsock1.SendData "R"
```

```
Winsock1.SendData vbCrLf
```

```
End Sub
```

```
Private Sub Command4_Click()
```

‘คำสั่ง STOP

```
Winsock1.SendData "S"
```

```
Winsock1.SendData vbCrLf
```

```
End Sub
```

```
Private Sub Command5_Click()
```

‘คำสั่งโปรแกรมการบันทึก เช่น P 1 00 12:35 rrrrrrr

```
Winsock1.SendData "P "
```

```
If Combo1.Text = "อาทิตย์" Then
```

```
Winsock1.SendData "1"
```

```
ElseIf Combo1.Text = "จันทร์" Then
```

```
Winsock1.SendData "2"
```

```
ElseIf Combo1.Text = "อังคาร" Then
```

```
Winsock1.SendData "3"
```

```
ElseIf Combo1.Text = "พุธ" Then
```

```
Winsock1.SendData "4"
```

```
ElseIf Combo1.Text = "พฤหัสบดี" Then
```

```
Winsock1.SendData "5"
```

```
ElseIf Combo1.Text = "ศุกร์" Then
```

```
Winsock1.SendData "6"
```

ElseIf Combo1.Text = "เสาร์" Then

Winsock1.SendData "7"

End If

Winsock1.SendData " "

‘ส่งคำสั่ง space bar

Winsock1.SendData pro.Text

‘ส่งคำสั่งจาก การเลือกโปรแกรม

Winsock1.SendData " "

Winsock1.SendData time.Text

‘ส่งคำสั่งจากช่องตั้งเวลา

Winsock1.SendData " "

If Check1.Value = 1 Then

‘คำสั่งจาก checkbox

Winsock1.SendData "rrrrrrr"

ElseIf Check2.Value = 1 Then

Winsock1.SendData "-----"

End If

Winsock1.SendData vbCrLf

End Sub

Private Sub Command6_Click()

Winsock1.SendData "H"

Winsock1.SendData vbCrLf

End Sub

Private Sub Command7_Click()

‘คำสั่งในการตรวจสอบวันที่ต้องการ

Winsock1.SendData "E "

If Combo1.Text = "อาทิตย์" Then

Winsock1.SendData "1"

```
ElseIf Combo1.Text = "จันทร์" Then
Winsock1.SendData "2"
ElseIf Combo1.Text = "อังคาร" Then
Winsock1.SendData "3"
ElseIf Combo1.Text = "พุธ" Then
Winsock1.SendData "4"
ElseIf Combo1.Text = "พฤหัสบดี" Then
Winsock1.SendData "5"
ElseIf Combo1.Text = "ศุกร์" Then
Winsock1.SendData "6"
ElseIf Combo1.Text = "เสาร์" Then
Winsock1.SendData "7"
End If
Winsock1.SendData vbCrLf
End Sub
Private Sub Command8_Click()
```

คำสั่งตั้ง วัน เดือน ปี

```
Winsock1.SendData "D "
If Combo2.Text = "อาทิตย์" Then
Winsock1.SendData "1"
ElseIf Combo2.Text = "จันทร์" Then
Winsock1.SendData "2"
ElseIf Combo2.Text = "อังคาร" Then
Winsock1.SendData "3"
```

```

ElseIf Combo2.Text = "พุทธ" Then

Winsock1.SendData "4"

ElseIf Combo2.Text = "พญหัตถ์" Then

Winsock1.SendData "5"

ElseIf Combo2.Text = "สุกรี" Then

Winsock1.SendData "6"

ElseIf Combo2.Text = "เสารีย์" Then

Winsock1.SendData "7"

End If

Winsock1.SendData " "

Winsock1.SendData Text2.Text

Winsock1.SendData vbCrLf

End Sub

Private Sub Form_Load()

```

‘ตั้งค่า IP และ port ที่ต้องการติดต่อ

```
Winsock1.RemoteHost = "192.168.0.5"
```

```
Winsock1.RemotePort = 80
```

```
Winsock1.Connect
```

```
With Combo1
```

‘กำหนดค่าที่แสดงใน ComboBox

```
.AddItem "อาทิตย์"
```

```
.AddItem "จันทร์"
```

```
.AddItem "อังคาร"
```

```
.AddItem "พุธ"
```


.AddItem "พฤษหัต"

.AddItem "สุกรี"

.AddItem "เสาร"

End With

With day

.AddItem "อาทิตย"

.AddItem "จันทร"

.AddItem "อังคาร"

.AddItem "พุธ"

.AddItem "พฤษหัต"

.AddItem "สุกรี"

.AddItem "เสาร"

End With

With pro

.AddItem "00"

.AddItem "01"

.AddItem "02"

.AddItem "03"

.AddItem "04"

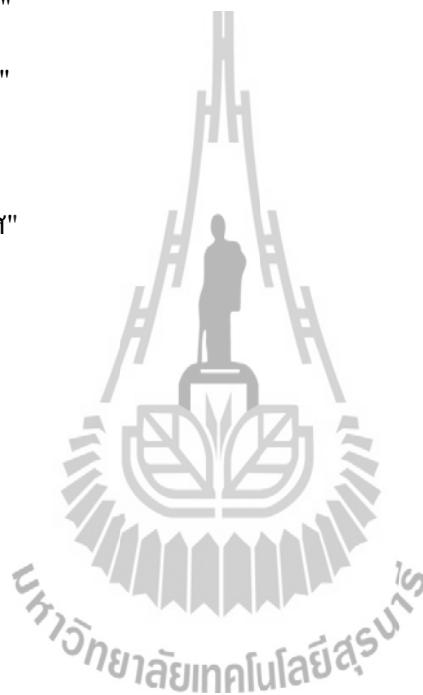
.AddItem "05"

.AddItem "06"

.AddItem "07"

.AddItem "08"

.AddItem "09"



.AddItem "10"

.AddItem "11"

.AddItem "12"

.AddItem "13"

.AddItem "14"

.AddItem "15"

.AddItem "16"

.AddItem "17"

.AddItem "18"

.AddItem "19"

.AddItem "20"

End With

With Combo2

.AddItem "อาทิตย์"

.AddItem "จันทร์"

.AddItem "อังคาร"

.AddItem "พุธ"

.AddItem "พฤหัสบดี"

.AddItem "ศุกร์"

.AddItem "เสาร์"

End With

End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)

Winsock1.SendData (KeyAscii)



End Sub

Private Sub Text3_Change()

End Sub

Private Sub Text4_Change()

End Sub

Private Sub Text5_Change()

End Sub

Private Sub Winsock1_Close()

Winsock1.Close

Text1.SetText = "connection closed" + vbCrLf

End Sub

Private Sub Winsock1_Connect()

‘แสดง IP และ Port เมื่อเชื่อมต่อ

Text1.SetText = "Connected to " + Winsock1.RemoteHost + ":" +
Str\$(Winsock1.RemotePort) + vbCrLf

End Sub

