ชุดควบคุมจานรับสัญญาณดาวเทียมอัตโนมัติแบบเคลื่อนย้ายได้

AUTOMATIC MOBILE SATELLITE RECEIVER CONTROLLER

โดย

นาย ฑิฆัมพร บุญชัก

นาย วินัย แจ่มกระจ่าง

นาย สุรศักดิ์ สุขสมพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ชุดควบคุมจานรับสัญญาณดาวเทียมอัตโนมัติแบบเคลื่อนย้ายได้

AUTOMATIC MOBILE SATELLITE RECEIVER CONTROLLER

โดย

นาย ฑิมัมพร บุญชัก 38013013

นาย วินัย แจ่มกระจ่าง 38013027

นาย สุรศักดิ์ สุขสมพงษ์ 38013037

อาจารย์ที่ปรึกษา

รศ. ณรงค์ เหมกรณ์

ผศ. นิภา ลีลารุจิ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดควบคุมจานรับสัญญาณดาวเทียมอัตโนมัติแบบเคลื่อนย้ายได้

AUTOMATIC MOBILE SATELLITE RECEIVER CONTROLLER

ผู้จัดทำ

1. นาย ฑิฆัมพร บุญชัก 38013013
2. นาย วินัย แจ่มกระจ่าง 38013027
3. นาย สุรศักดิ์ สุขสมพงษ์ 38013037

.....อาจารย์ที่ปรึกษา

(รศ. ณรงค์ เหมกรณ์)

.....อาจารย์ที่ปรึกษา

(ผศ. นิภา ลีลารุจิ)

ชุดควบคุมจานรับสัญญาณดาวเทียมอัตโนมัติแบบเคลื่อนย้ายได้

AUTOMATIC MOBILE SATELLITE RECEIVER CONTROLLER

โดย นาย ทิฆัมพร บุญชัก 38013013

นาย วินัย แจ่มกระจ่าง 38013027

นาย สุรศักดิ์ สุขสมพงษ์ 38013037

อาจารย์ที่ปรึกษา รศ. ณรงค์ เหมกรณ์

ผศ. นิภา ลีลารุจิ

บทคัดย่อ

สำหรับเครื่องควบคุมจานรับสัญญาณดาวเทียมในโครงการนี้ ใช้ MCS-51 เป็นตัวควบคุมการเคลื่อนที่ของจานรับสัญญาณดาวเทียม มีหลักการที่สำคัญ คือ ใช้คอมพิวเตอร์ คำนวณมุม ตามสถานที่และดาวเทียมที่แตกต่างกัน ส่งผลที่ได้ให้กับ MCS-51 เพื่อควบคุมสเตปปิงมอเตอร์

หลังจากจานรับสัญญาณดาวเทียมเคลื่อนที่ไปอยู่ที่ตำแหน่ง ซึ่งรับสัญญาณจากดาวเทียมได้แล้ว เครื่องรับก็จะทำหน้าที่ ในการแปลงสัญญาณต่าง ๆ ที่รับได้ เพื่อเปลี่ยนเป็นสัญญาณภาพและเสียงไปใช้งานต่อไป

ABSTRACT

This project proposes a automatic mobile satellite receiver controller . It is a satellite antenna position controller. That can control and process antenna disk moving by MCS-51 microcontroller , microcomputer. The concept is using microcomputer calculate angle from location of station and satellite. Then send the result to control stepping motor by MCS-51 .

After the antenna went to suitable position for receiving signal from satellite. The receiver will convert each signal for change to be video and audio signals.

สารบัญ

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
คำนำ

1.1 บทนำ

ระบบการสื่อสารดาวเทียมเป็นที่นิยมใช้กันมากทั่วโลก มีการส่งดาวเทียมขึ้นไปโคจรอยู่ในวงโคจรเป็นจำนวนนับร้อยดวง เนื่องจากระบบการสื่อสารผ่านดาวเทียม มีข้อดีหลายอย่าง การทำงานมีพื้นที่คลอบคลุมส่วนต่างๆ ของพื้นผิวโลกได้กว้างขึ้นอยู่กับการออกแบบจานสายอากาศบนดาวเทียมให้เหมาะสมกับการใช้งาน ระบบดาวเทียมเหมาะที่จะใช้สำหรับการติดต่อสื่อสารระหว่างประเทศ, การติดต่อสื่อสารภายในประเทศ หรือใช้สำหรับการแพร่สัญญาณโทรทัศน์ให้ครอบคลุมพื้นที่กว้างๆ

ในการใช้ดาวเทียมในการติดต่อสื่อสารนั้น จำเป็นต้องมีส่วนประกอบหลายส่วน เช่น ดาวเทียม สถานีรับสัญญาณดาวเทียม เนื่องจาก สัญญาณที่ถูกส่งมาจากดาวเทียมมีระดับสัญญาณต่ำ เพราะการติดตั้ง จานรับสัญญาณให้ถูกต้องและถูกตำแหน่ง จึงมีความสำคัญ เนื่องจาก การติดตั้งจานรับสัญญาณดาวเทียม ต้องทราบมุมกวาดและมุมเงย ของตำแหน่งที่ตั้งซึ่งแตกต่างกัน จึงมีความยุ่งยากสำหรับผู้ใช้งานบางคน ถ้าเราสามารถแก้ปัญหา นี้ได้ ก็จะเป็นการสะดวกสบายต่อการใช้งาน และใช้งานได้อย่างมีประสิทธิภาพอีกด้วย

1.1.1 ความเป็นมาของดาวเทียมบนฟากฟ้า

นวนิยายและเรื่องราวเกี่ยวกับวิทยาศาสตร์ เขียนขึ้นโดย "นายอาเธอร์ ซี. คลาค" (Arthur C. Clarke) เมื่อปี พ.ศ. 2488 ได้เสนอแนวความคิดในการที่จะใช้สถานีดาวเทียมซึ่งลอยอยู่กับที่ในอวกาศเพื่อส่งสัญญาณโทรทัศน์ และสัญญาณที่ใช้ติดต่อสื่อสารต่าง ๆ ลงมาบนพื้นโลก โดยให้ดาวเทียมโคจรอยู่ในวงโคจรรูปวงกลมที่เรียกว่า Geosynchronous Orbit หรือ Geostationary ซึ่งความหมายก็คือ ดาวเทียมจะต้องลอยอยู่ในอวกาศเหนือตำแหน่งเส้นศูนย์สูตร (Equator) ณ ความสูงระดับหนึ่งซึ่งจะทำให้การโคจรของดาวเทียมมีความเร็วพอดีกับการหมุนของโลก เมื่อโลกของเราใช้เวลาในการหมุนรอบตัวเองหนึ่งรอบ 24 ชั่วโมง จะเท่ากับดาวเทียมโคจรรอบโลกหนึ่งรอบพอดี และหากเราควบคุมให้ดาวเทียมเดินทางไปในทิศทางเดียวกันกับการหมุนรอบตัวเองของโลก ก็จะมีผลทำให้เหมือนกับว่า ดาวเทียมนั้นลอยอยู่ที่ตำแหน่งเดิมตลอดเวลาเมื่อเทียบกับจุดสังเกตการณ์บนพื้นโลก และดาวเทียมจะต้องอยู่ที่ระดับ 35,786 กิโลเมตรเหนือพื้นโลก

นอกจากนี้นายอาเธอร์ ซี. คลาค ยังให้แนวความคิดไว้ว่า หากใช้ดาวเทียมเพียงดวงเดียวลอยอยู่เหนืออเมริกา จะสามารถทำให้สะดวกต่อการส่งสัญญาณรายการโทรทัศน์และ วิทยุ เป็นอันมาก อีกทั้งมีประสิทธิภาพสูง และการลงทุนค่อนข้างต่ำ เพราะไม่ต้องมีการสร้างสถานีทวนสัญญาณ (Repeater) ภาคพื้นดินให้มากมาย เพียงแต่ส่งสัญญาณรายการดังกล่าวจากดาวเทียมลงมาที่บ้านของประชาชนโดยตรง

ในครั้งแรกนั้นแนวความคิดของนายอาเธอร์ ซี. คลาค ไม่ได้รับความสนใจจากวงการนักวิทยาศาสตร์เลย เพราะคิดว่าเป็นไปไม่ได้ เนื่องจากเป็นแนวความคิดของนักเขียนนวนิยายทางวิทยาศาสตร์ที่มีแต่ความเพ้อฝัน แต่ต่อมาก็กลายเป็นบทเรียนที่ว่า บ่อยครั้งทีเดียวที่จุดกำเนิดของความคิดและทฤษฎีใหม่ ๆ ที่เกิดจากการเพ้อฝันเหล่านี้ บางกรณีได้กลายมาเป็นความจริงทางวิทยาศาตร์ ทฤษฎีของการวางตำแหน่งวงโคจรของดาวเทียมให้ลอยอยู่เหนือ เส้นศูนย์สูตรในระดับความสูงที่เหมาะสม ซึ่งเกิดจากแนวความคิดของคลาคนี้ ต่อมามีผู้ตั้งสมญานามให้ใหม่ว่า Clarke Orbit หรือ Clarke Belt ในประเทศไทยเรียกดาวเทียมแบบที่มีตำแหน่งอยู่กับที่เมื่อเปรียบเทียบกับจุดสังเกตการณ์บนโลกนี้ว่า "ดาวเทียมค้างฟ้า"

ต่อมาในปี พ.ศ. 2500 ประเทศสหภาพโซเวียต (รัสเซีย) เป็นชาติแรกที่ส่งดาวเทียมชื่อว่า "สปุตนิก 1" (Sputnik 1) ไปโคจรในอวกาศในระดับต่ำแล้วส่งข้อมูลเกี่ยวกับความหนาแน่นและอุณหภูมิของบรรยากาศชั้นสูงกลับมาสู่โลก ด้วยความถี่ 20.005 MHz และ 40.005 Mhz ซึ่งถือว่าเป็นก้าวแรกแห่งการพัฒนาเทคโนโลยีทางด้านอวกาศและดาวเทียมของโลกทีเดียว

ต่อจากนั้น ได้มีการพัฒนาสิ่งอื่น ๆ เพิ่มขึ้นมาอีกมากมาย เพื่อทำให้การสื่อสารทางดาวเทียมเป็นไปได้ตามที่นายอาเธอร์ ซี. คลาค ได้จินตนาการเอาไว้ ไม่ว่าจะเป็นแผงโซล่าร์เซลล์ ซึ่งประกอบไปด้วยโฟโต้วอลตาอิกเซลล์ (Photovoltaic) จำนวนหลายพันเซลล์เพื่อใช้ในการผลิตกระแสไฟฟ้าจากแสงอาทิตย์ ไปเลี้ยงตัวเอง อีกทั้งมีการพัฒนาอุปกรณ์ทางอิเล็กทรอนิกส์ให้อยู่ในรูปของวงจรรวมที่มีขนาดเล็กมาก ๆ น้ำหนักจึงน้อยทำให้สามารถส่งไปในวงโคจรที่สูง ๆ ได้ง่าย

มาถึงในปี พ.ศ. 2505 ประเทศสหรัฐอเมริกาได้ส่งดาวเทียมชื่อว่า "เทลสตาร์ 1" (Telstar 1) ขึ้นไปโคจรรอบโลกเป็นรูปวงรี แต่ยังไม่อยู่ในวงโคจรที่เรียกว่า Geostationary โดยใช้การควบคุมการโคจรจากสถานีภาคพื้นดินที่อยู่บนโลก ดาวเทียมดวงนี้ถือว่าเป็นดาวเทียมดวงแรกของโลกที่ใช้ในการสื่อสารอย่างแท้จริง และใช้ส่งรายการโทรทัศน์รวมลงมาด้วยต่อมาอีกสามปี คือ ปี พ.ศ. 2508 ได้มีการส่งดาวเทียมที่ชื่อว่า เออร์ลี่เบิร์ด (Early Bird) ขึ้นไปโคจร Geostationary ซึ่งถือว่าเป็นดาวเทียมที่อยู่ในวงโคจรแบบค้างฟ้าและใช้ในเชิงพาณิชย์เป็นดวงแรกอย่างแท้จริง โดยมีช่องสัญญาณการถ่ายทอดสัญญาณเกี่ยวกับโทรศัพท์เทเล็กซ์ ข่าวสารต่าง ๆ รวมทั้งรายการโทรทัศน์ ที่รับมาจากด้านหนึ่งของมหาสมุทรแอตแลนติกผ่านดาวเทียมเพื่อส่งไปส่วนอื่น ๆ ของประเทศ หลังจากที่ดาวเทียมเออร์ลี่เบิร์ดประสบความสำเร็จแล้วองค์การดาวเทียมเพื่อการสื่อสารโทรคมนาคมระหว่างประเทศ (INTELSAT : International Telecommunications satellite Organization) ก็ได้ส่งดาวเทียมขึ้นสู่วงโคจรอีกหลายดวง ภายใต้ชื่ออินเทลแซท (INTELSAT) โดยมีหมายเลขเรียงลำดับก่อนหลังกันไป เมื่อรวมกับดาวเทียมของบริษัทอื่น ๆ ที่ส่งขึ้นไปโคจรรอบโลกในปัจจุบัน จะมีจำนวนประมาณ 100 ดวง เศษ โดยโคจรอยู่ในวงโคจร Geostationary มีใช้ทั้งงานการให้บริการภายในประเทศและระหว่างประเทศรวมกันไป

1.1.2 ความถี่ขาขึ้นและขาลง

ดาวเทียมแต่ละดวงนั้นเป็นเสมือนกับสถานีทวนสัญญาณ หรือที่เรียกว่า รีพีทเตอร์ (Repeater) ซึ่งติดตั้งอยู่สูงมากถึง 35,786 กิโลเมตร จึงต้องทำหน้าที่เป็นทั้งเครื่องรับและเครื่องส่งเพื่อติดต่อกับสถานีภาคพื้นดิน โดยสถานีภาคพื้นดินจะส่งสัญญาณในช่วง "ขาขึ้น" ที่ความถี่หนึ่งซึ่งเรียกว่า Uplink ไปให้ดาวเทียม เมื่อดาวเทียมได้รับก็จะทำการเปลี่ยนความถี่ที่รับ ได้ให้เป็นอีกความถี่หนึ่ง และส่งกลับมาให้สถานีภาคพื้นดินอื่น ๆ ซึ่งสัญญาณที่ส่งลงมาจากดาวเทียมจะเรียกว่า Downlink หรือความถี่ "ขาลง" โดยสัญญาณที่ส่งลงมานี้สามารถที่จะครอบคลุมพื้นผิวของโลกได้ถึง 40% ของจำนวนพื้นที่โลกทั้งหมด

ในการออกแบบดาวเทียมแต่ละดวงนั้น จะออกแบบให้มีอุปกรณ์หรือวงจรส่วนต่าง ๆ ที่เหมือนกันสำรองเอาไว้เป็นอะไหล่ เพื่อจะสามารถทำงานได้ทันทีในกรณีที่อุปกรณ์ หลักเกิดขัดข้องโดยการควบคุมจากสถานีภาคพื้นดิน ที่ต้องออกแบบให้เป็นเช่นนี้เพราะค่าใช้จ่ายในการสร้าง และการนำขึ้นไปไว้ในวงโคจรต้องเสียค่าใช้จ่ายไม่ต่ำกว่า 100 ล้านเหรียญสหรัฐ หรือเท่ากับเงินไทยประมาณ 2,500 ล้านบาท หากไม่มีการสำรอง

อุปกรณ์เอาไว้ให้มากพอก็คงจะไม่คุ้ม กับการลงทุนและความเสียหายที่จะเกิดขึ้น อีทั้งคงจะไม่เป็นการง่ายนักที่จะส่งช่างขึ้นไปซ่อมดาวเทียมในวงโคจร Geostationary กันได้บ่อย ๆ

1.1.3 ฟุตปริ้นท์

ส่วนที่เป็นสายอากาศของดาวเทียม จะทำหน้าที่ส่งสัญญาณโทรทัศน์ลงมายังพื้นโลกให้มีรูปร่างเฉพาะตัวได้ ตัวอย่างเช่น หากต้องการส่งสัญญาณโทรทัศน์มายังประเทศไทยโดยเฉพาะ ก็ออกแบบสายอากาศของดาวเทียมให้มีลำคลื่น (Beam) ครอบคลุมเฉพาะประเทศไทย โดยลำคลื่นจะออกมาครอบคลุมพื้นที่ของประเทศไทยให้มากที่สุด ลักษณะของลำคลื่นที่ออกแบบไว้ให้ครอบคลุมเฉพาะพื้นที่ที่ต้องการเรียกว่า ฟุตปริ้นท์ (Footprint) ดาวเทียมแต่ละดวงจะมีฟุตปริ้นท์เป็นลักษณะเฉพาะของตัวเอง ซึ่งพื้นที่ที่จะได้รับสัญญาณจากดาวเทียมได้ดีหรือแรงที่สุดจะอยู่ในส่วนที่เรียกว่า ศูนย์กลาง (Center) ของฟุตปริ้นท์ หากหลุดออกไปจากศูนย์กลางนี้ความแรงของสัญญาณก็จะลดลง ผู้ที่อาศัยอยู่ในพื้นที่ที่อยู่ในจุด ศูนย์กลางของฟุตปริ้นท์ เช่น กรุงเทพฯ และจังหวัดใกล้เคียงสามารถรับสัญญาณจากดาวเทียมด้วยจานรับสัญญาณที่มีขนาดเล็กกว่าผู้ที่อาศัยอยู่ภายนอกของศูนย์กลางออกไป เช่น ผู้ที่อาศัยอยู่ในประเทศกลุ่มอินโดจีนอาจต้องใช้จานรับสัญญาณดาวเทียมขนาด 10 ฟุต เพื่อรับสัญญาณที่มีคุณภาพของภาพชัดเจนเท่ากับผู้ที่อาศัยอยู่ในประเทศไทยซึ่งใช้จานรับสัญญาณขนาดเพียง 4 ฟุต แต่โดยทั่วไปแล้วจานที่ใช้รับสัญญาณดาวสำหรับที่พักอาศัยมักจะมีขนาดเส้นผ่านศูนย์กลางตั้งแต่ 6-12 ฟุต

ดาวเทียมที่ส่งสัญญาณในความถี่ย่าน C-band นั้นมักจะมีกำลังส่งค่อนข้างต่ำประมาณ 8-16 วัตต์เท่านั้น ดังนั้นเมื่อสัญญาณเดินทางมาถึงเราจึงมีขนาดของสัญญาณที่อ่อนมาก เราจึงจำเป็นจะต้องใช้จานรับสัญญาณที่มีขนาดเส้นผ่านศูนย์กลางค่อนข้างใหญ่ แต่ก็มีข้อดี คือ ครอบคลุมพื้นที่ได้กว้างขวางมากรวมทั้งสามารถตั้งมุมยิง ของสายอากาศให้มีจุดศูนย์ กลางของสัญญาณเน้นความเข้มไว้สองจุดก็ได้

ทุกวันนี้ดาวเทียมที่ส่งสัญญาณในความถี่ย่าน Ku-band จะส่งสัญญาณด้วยกำลังส่งปานกลาง ประมาณ 20-50 วัตต์ แต่ในประเทศญี่ปุ่นใช้กำลังส่งสูงมากประมาณ 80-100 วัตต์ ในการส่งสัญญาณจากดาวเทียม Japanese DBS (Direct Broadcasting Satellite) ตรงไปยังจานรับสัญญาณ ซึ่งมีขนาดเล็กมาก ๆ เส้นผ่านศูนย์กลางเพียง 1.5 ฟุตเท่านั้นก็สามารถรับสัญญาณได้แล้ว ทั้งนี้รวมทั้งดาวเทียม "ไทยคม" ของไทยที่จะเริ่มส่งสัญญาณกลับมายังพื้นโลกในช่วงต้นปี พ.ศ. 2537 นี้ด้วย

1.1.4 การรับสัญญาณดาวเทียมโดยตรงในเอเชีย

วันที่ 7 เมษายน พ.ศ. 2533 ดาวเทียมทางด้านสื่อสารเชิงพาณิชย์ดวงแรกที่ถูกออกแบบให้ใช้สำหรับประเทศที่อยู่ในทวีปเอเชียได้ถูกส่งเข้าสู่วงโคจรโดย China Great Wall Industry Corporation ดาวเทียมดวงนี้ถูกออกแบบให้มีศูนย์กลางของลำคลื่นครอบคลุมพื้นที่สองส่วน โดยส่วนแรกครอบคลุมพื้นที่ส่วนใหญ่ของประเทศสาธารณรัฐประชาชนจีน และอีกส่วนหนึ่งครอบคลุมพื้นที่ของประเทศที่อยู่ในเอเชียตะวันออกเฉียงใต้ ดาวเทียมดวงนี้ได้ออกแบบให้มีทรานสปอนเดอร์ในการส่งสัญญาณรายการโทรทัศน์มากถึง 24 ช่อง โดยใช้ชื่อว่า เอชียแซท (Asiasat)

ต่อมาในวันที่ 19 เมษายน พ.ศ. 2534 บริษัท Hutchvision ของ Hong Kong ได้เริ่มใช้ดาวเทียมเอเชียแซทดวงนี้ส่งสัญญาณโทรทัศน์แบบมัลติแชนแนล เพื่อให้บริการกับโทรทัศน์ระบบเคเบิล และระบบรับจากดาวเทียมโดยตรงที่ติดตั้งในเอเชีย อินเดีย และตะวันออกกลาง ซึ่งเรารู้จักกันในชื่อของ STAR TV โดยทุกวันนี้ได้

ส่งรายการข่าว กีฬา มิวสิควีดีโอ ภาพยนตร์ และรายการบันเทิงทั่วไป ๆ อีกทั้งยังส่งในระบบเสียงหลายภาษา (Multi-Language Audio) ในรายการโทรทัศน์เดียวกัน สำหรับการบริการของ BBC World Service อีกด้วย

ยังมีประเทศอื่นๆ อีกหลายประเทศเลือกใช้ดาวเทียมเอเชียแซท ส่งสัญญาณรายการโทรทัศน์ เพื่อใช้ภายในประเทศของตนเอง เช่น ปากีสถาน สาธารณรัฐประชาชนจีน พม่า และมองโกเลีย

พอมาถึงเดือนเมษายน พ.ศ. 2535 CNN International และ ESPN ได้เริ่มขยายการให้บริการเข้ามาในทวีปเอเชีย โดยเริ่มใช้ดาวเทียมปาลาปา (Palapa) B2P ของประเทศอินโดนีเซียทดลองส่งสัญญาณรายการโทรทัศน์ระบบเคเบิลจากอเมริกามายังทวีปเอเชียซึ่งดาวเทียมปาลาปา B2P นี้ ยังได้ให้บริการส่งสัญญาณโทรทัศน์ไปยังสถานีลูกข่ายของประเทศในทวีปเอเชียตะวันออกเฉียงใต้ เพื่อเป็นการให้บริการภายในของประเทศนั้น ๆ เช่น TV1 และ TV3 ของประเทศมาเลเซีย RCTI, TVRI และ CPTI ของประเทศอินโดนีเซียเอง รวมทั้งสถานีโทรทัศน์ช่อง 5, ช่อง 7 และช่อง 11 ของไทย

ในต้นปี พ.ศ. 2536 สถานีโทรทัศน์ช่อง 5 และช่อง 7 ของไทยได้เปลี่ยนมาใช้บริการดาวเทียมปาลาปา B4 ซึ่งเป็นดาวเทียมดวงใหม่ของประเทศอินโดนีเซีย และถูกส่งเข้าสู่วงโคจรโดยจรวดของบริษัท McDonnell Douglas ประเทศสหรัฐอเมริกา ตั้งแต่กลางปี พ.ศ. 2535

1.2 ดาวเทียมไทยคม1 และไทยคม 2

เมื่อวันที่ 8 ตุลาคม พ.ศ. 2534 บริษัท ชินวัตร คอมพิวเตอร์ แอนด์ คอมมิวนิเคชั่น จำกัด ได้ลงนามในสัญญาว่าจ้าง บริษัท ฮิวส์ คอมมินิเคชั่น อินเตอร์เนชั่นแนล (Hughes Communication International) และบริษัท แอดวานซ์ อิเล็กทรอนิกส์ ซิสเต็มส์ อินเตอร์เนชั่นแนล (Advanced Electronic Systems International) ซึ่งทั้งสองบริษัทเป็นส่วนหนึ่งของบริษัท ฮิวส์ แอร์คราฟท์ (Hughes Aircraft Company) ประเทศสหรัฐอเมริกา โดยให้ทำการออกแบบสร้างระบบดาวเทียมสื่อสารแห่งชาติทั้งสองดวงให้แก่ประเทศไทย

ศัพท์ดาวเทียม

คำศัพท์	ความหมาย
ทรานส์พอนเดอร์	ช่องสัญญาณบนตัวดาวเทียมที่ใช้เป็นตัวกลางในการส่งผ่านคลื่นความถี่ในการติดต่อสื่อสารผ่านดาวเทียม ทรานส์พอนเดอร์หรือช่องสัญญาณดาวเทียมของไทยคมประกอบด้วยทรานส์พอนเดอร์ที่ใช้งานในย่านความถี่ C-band และ Ku-band
C-band	ความถี่หรือชื่อความถี่ที่ใช้ในการสื่อสารผ่านดาวเทียม ซึ่งย่านความถี่ C-band จะมีความในการส่งคลื่นสัญญาณขาขึ้นเท่ากับ 6 GHz (กิกะเฮิร์ต) และคลื่นสัญญาณขาลงเท่ากับ 4 GHz
Ku-band	ความถี่หรือชื่อความถี่ที่ใช้ในการสื่อสารผ่านดาวเทียม ซึ่งย่านความถี่ Ku-band จะมีความในการส่งคลื่นสัญญาณขาขึ้นเท่ากับ 14 GHz (กิกะเฮิร์ต) และคลื่นสัญญาณขาลงเท่ากับ 12 GHz

ตารางที่ 1.1 ความหมายของศัพท์ที่ใช้บ่อยในการสื่อสารดาวเทียม

บทที่2
ทฤษฎีที่เกี่ยวข้อง

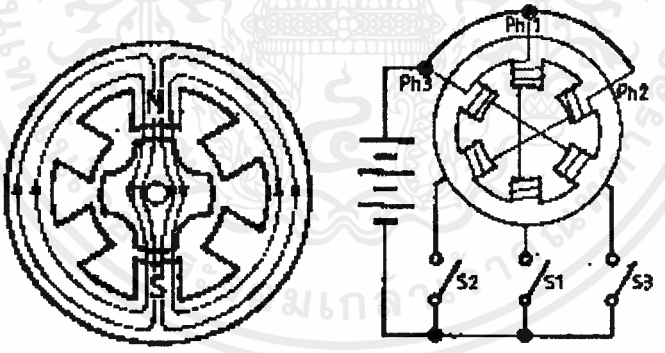
2.1 ความรู้พื้นฐานสเต็ปปิ้งมอเตอร์

ชนิดของสเต็ปปิ้งมอเตอร์ สามารถแบ่งออกได้ 3 ชนิด ตามโครงสร้างของโรเตอร์คือ

- Variable Reluctance
- Permanent Magnet
- Hybrid

Variable Reluctance

Variable Magnet Step Motor **มีโรเตอร์** (rotor) **เป็นเหล็กอ่อน มีฟันแบบแกติ ดังรูปที่ 2.1 (a) จะหมุนเมื่อมีแรงแม่เหล็กไฟฟ้าจากฟันของสเตเตอร์** (stator) **มากระทำ ซึ่งคล้ายกับ** solenoid **โรเตอร์แบบนี้จะมีแรงเฉื่อยต่ำกว่าชนิดอื่นๆ ทำให้การตอบสนองดีขึ้น อย่างไรก็ตามเมื่อโรเตอร์ไม่มีสภาพแม่เหล็ก ทำให้ไม่มีแรงบิดตกค้าง เมื่อมอเตอร์คลายพลังงานมอเตอร์จึงหมุนฟรีได้ โดยปกติ** step angle **ของ** variable reluctance step motor **คือ 7.5 หรือ 1.5 ลักษณะของมอเตอร์ชนิดนี้แสดงดังรูปที่ 2.1 (a)**

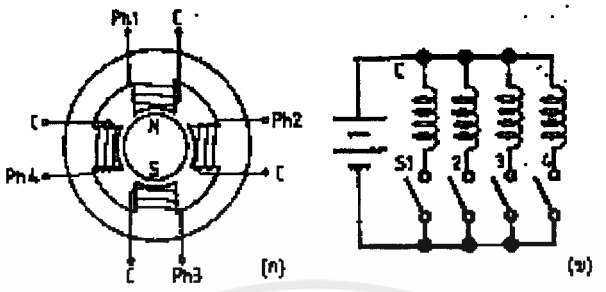


รูปที่ 2.1 ภาพหน้าตัดและการพันขดลวดของ สเต็ปปิงมอเตอร์ แบบ Variable Magnet

Permanent Magnet

Permanent Magnet Step Motor **ประกอบด้วย** rotor **ที่เป็นแม่เหล็กถาวร จะเปลี่ยนขั้วสลับไปมาระหว่างขั้วเหนือและขั้วใต้ มอเตอร์บางตัวจะมีแม่เหล็กสอดอยู่ใน** stator **เพื่อเพิ่มสนามแม่เหล็กและเพิ่มแรงบิด แม่เหล็กมักทำมาจาก** alnico **(อะลูมิเนียม,นิเกิล,โลหะผสมโคบอลต์) หรือสารใหม่กว่า** (samarium cobalt) **ที่ให้พลังแม่เหล็กมากกว่า** Permanent Magnet Step Motor **ต้องการพลังงานน้อยกว่าชนิดอื่น และยังมีผลตอบสนอง** damping **ดีกว่า มอเตอร์ชนิดนี้จะมีฟันของดรเตอร์และสเตเตอร์ถี่ๆ เป็นจำนวนมาก ทำให้** step angle

เล็กลง มาตรฐานของ step angle ที่ใช้ เช่น 1.8,7.5,15,30,45 และ 90 ลักษณะของมอเตอร์ชนิดนี้แสดงดังรูปที่ 2.2



รูปที่ 2.2 ภาพหน้าตัดของ PM สเต็ปปิงมอเตอร์แบบ 4 เฟส

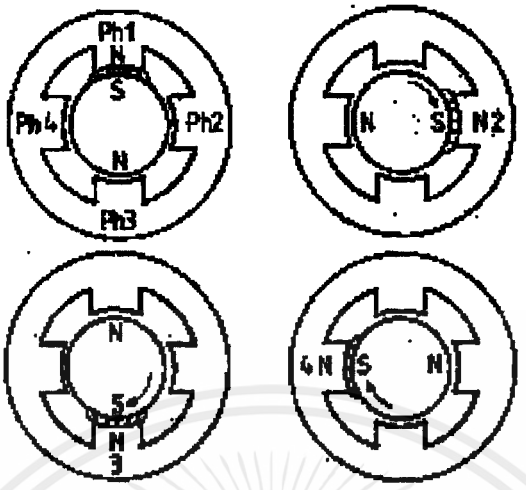
Hybrid

Hybrid Step Motor มี rotor ทั้งแบบ variable reluctance และแบบ permanent magnet แม่เหล็กถาวรเล็กๆ จะอยู่รอบๆ แกน ซึ่งแตกต่างจาก permanent magnet ทั่วไป โดยมีปลายข้างหนึ่งของ rotor เป็นขั้วเหนือ และปลายอีกข้างหนึ่งเป็นขั้วใต้ ฟันของ rotor ถูกแบ่งเป็น 2 แกน ซึ่งปลายทั้ง 2 จะติดแน่น Hybrid step motor ใช้โครงสร้างแบบที่ 2 เท่านั้น Hybrid step motor มีฟันของ rotor มากกว่าและให้แรงบิดสูงกว่า step angle คือ 0.9 และ 1.8

หลักการทำงานของสเต็ปปิ้งมอเตอร์

จากหลักการของ rotor ที่เป็นแม่เหล็กถาวรมีขั้วเดี่ยวเหนือและใต้ stator มี 4 ฟัน มีขดลวด A - A และ B - B พันอยู่เมื่อ A - A (และ B - B) มีไฟ DC ป้อนและกราวด์ตามลำดับ ฟันด้านบน(และฟันด้านขวา)จะเป็นขั้วเหนือ และขั้วตรงข้าม จะเป็นขั้วใต้ซึ่งจะขับให้ rotor หมุนไปเป็นมุม +45 ขั้วของ rotor จะตรงข้ามกับขั้วของ stator เมื่อขั้วของ A - A ถูกเปลี่ยนโดยการสลับขั้วจ่ายไฟ rotor จะหมุนกับไป 90 อยู่ที่มุมใหม่ที่ -45 ซึ่งหมายถึง step angle จะมีค่า 90 การทำงานในลักษณะนี้เรียกว่า full step mode โดยที่แต่ละ step ขึ้นอยู่กับจำนวนเฉพาะของ step angle ในลักษณะ full step mode นี้จะมีการทำงานที่ต่อเนื่องกัน 4 step ดังตารางที่ 2.1

Step Motor สามารถทำให้หมุนแบบ half step ได้โดยทำงานใน half step mode นี้ ขดลวดเพียงขดเดียวเท่านั้นที่จะได้รับพลังงาน ทำให้เกิดการทำงานที่ต่อเนื่องถึง 8 step ดังตารางที่ 2 .2



รูปที่ 2.3 การหมุนของสเต็ปมอเตอร์

ตารางที่ 2.1 full-step mode

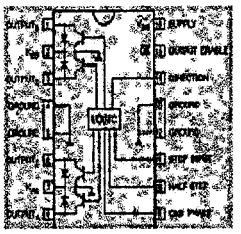
สเต็ป	Ph1	Ph2	Ph3	Ph4
1	H	L	H	L
2	L	H	H	L
3	L	H	L	H
4	H	L	L	H

ตารางที่ 2.2 half-step mode

สเต็ป	Ph1	Ph2	Ph3	Ph4
1	H	L	H	L
2	L	L	L	H
3	L	H	H	L
4	H	L	L	L
5	L	H	L	H
6	L	L	H	L
7	H	L	L	H
8	L	H	L	L

2..1.2 ไอซีขับเต็ปเปอร์มอเตอร์ยูนิโพลาร์ UNC 5804B

UNC 5804B คืออุปกรณ์ขับสเต็ปเปอร์มอเตอร์ที่ภายในรวมเอาวงจรควบคุมลอจิกแบบซีมอสที่มีอัตราการสิ้นเปลืองกำลังงานต่ำเข้าไว้ด้วยโดยที่มุ่งเน้นคุณสมบัติของการขับกระแสและแรงดันออกทางเอาต์พุตที่สูงหรือกำลังงานทางเอาต์พุตสูงด้วยดาร์ลิงตัน เพาเวอร์ทรานซิสเตอร์ภายใน และให้เอาต์พุตออกมาแบบไบโพลาร์ไอซีตัวนี้ถูกเรียกว่าเป็นไอซีแบบไบมอส (BiMOS II) ที่ทำหน้าที่แปลงสัญญาณการควบคุมและการขับออกเอาท์พุทที่สมบูรณ์ เพื่อขับสเต็ปปิงมอเตอร์ยูนิโพลาร์ 4 เฟส ด้วยอัตรากระแสขับทางเอาต์พุทต่อเนื่องสูงถึง 1.25 แอมป์ ต่อเฟสที่แรงดัน 35 โวลท์ (กระแสขับขณะเริ่มต้นสูงถึง 1.5 แอมป์) ในรูปที่ 2..4 แสดงโครงสร้างภายในและการจัดขาใช้งานของ UNC 5804B



รูปที่ 2.4โครงสร้างภายในและการจัดขาของ UNC 5804B

คุณสมบัติโดยทั่วไปของไอซีนอกจากจะเป็นการขับสเต็ปปิงมอเตอร์ในแบบขับโดยตรงแล้วยังสามารถที่จะควบคุมได้ด้วยลอจิกทางอินพุตอีก คุณสมบัติทางเทคนิคโดยทั่วๆ ไปของไอซีมีดังต่อไปนี้

- ระดับแรงดันเอาต์พุตสูงสุด (Vce) 50 โวล์ต
- ระดับแรงดันเอาต์พุตคงที่ (Vce sus) 35 โวล์ต
- กระแสซิงค์เอาต์พุต (Iout) 1.5 แอมป์
- รูปแบบการขับได้ทั้งแบบ 2 เฟส, ครึ่งสเต็ป, และต่อเนื่อง
- มีไดโอดแคลมป์ต่อยู่ภายใน
- เอาต์พุตจะทำงานตามสถานะการควบคุมทิศทางทางอินพุต
- มีระบบการทำงานแบบเพาเวอร์ออนรีเซ็ท
- มีวงจรตรวจจับและควบคุมการทำงานเมื่ออุณหภูมิสูงเกิน

รายละเอียดข้อมูลของคุณสมบัติทางไฟฟ้าของ UNC 5804B ดังตารางที่ 2.3

ตารางที่ 2.3 คุณสมบัติทางไฟฟ้าของ UNC 5804B

คุณสมบัติ	สัญญลักษณ์	ค่า	หน่วย
แรงดันเอาท์พุตสูงสุด	Vce	50	โวลต์
แรงดันเอาท์พุทต่อเนื่อง	Vce sus	35	โวลต์
กระแสซิงค์เอาท์พุต	Iout	1.5	แอมป์
แรงดันไฟเลี้ยงไอซีหรือระดับลอจิก	Vdd	7.0	โวลต์
แรงดันอินพุต	Vin	7.0	โวลต์
กระแสรั่วไหลทางเอาต์พุต	Icex	10-50	ไมโครแอมป์
แรงดันไบแอสตรงในไดโอดแคลมป์	Vf	1.5-3.0	ไมโครแอมป์
แรงดันเอาต์พุตจุดทำงาน	Vce sat	1.1-1.5	โวลต์
กระแสอินพุต	Iin	0.5-5.0	ไมโครแอมป์

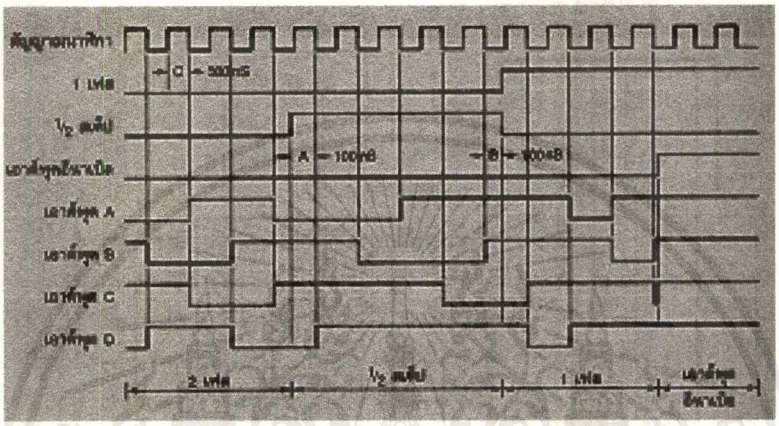
กระแสรั่วไหลในไดโอดแคลมป์	Ir	10-50	ไมโครแอมป์
กระแสสำหรับเลี้ยงตัวไอซี	Idd	20-30	มิลลิแอมป์
ช่วงเวลาหน่วงขณะหยุดทำงาน	Toff	10	ไมโครวินาที
ช่วงเวลาขณะเริ่มทำงาน	Ton	10	ไมโครวินาที
ค่าอุณหภูมิสูงสุดที่จะหยุดทำงานอัตโนมัติ	Tj	165	องศาเซลเซียส

ในส่วนของซีมอสลอจิกภายในตัวไอซีจะทำหน้าที่กำหนดลำดับลอจิกควบคุมที่รับเข้ามาทางขาควบคุมอินพุต ขาควบคุมทิศทาง (DIRECTION) และขาควบคุมการขับทางเอาต์พุต (OUTPUT ENABLE) รวมทั้งลอจิกการควบคุมการรีเซตตัวเองเมื่อเริ่มจ่ายแรงดันให้ (power on) การควบคุมทางเอาต์พุตนั้นจะมีทั้งหมดสามรูปแบบการขับสเต็ปปิงมอเตอร์ที่ต่ออยู่ทางเอาต์พุต คือ

- แบบ 1 เฟส (one phase) หรือแบบ wave-drive
- แบบ 2 เฟส
- แบบ ครึ่งสเต็ป

โดยที่การควบคุมการขับทางเอาต์พุตนี้จะถูกทำการเลือกรูปแบบการขับจากสัญญาณหรือสวิตช์เลือกภายนอก สัญญาการควบคุมทางอินพุตของไอซีนี้สามารถรับได้ทั้งสัญญาณควบคุมจากเอาต์พุตของอุปกรณ์ในวงจรประเภท CMOS, PMOS, NMOS แต่สำหรับสัญญาณควบคุมจากเอาต์พุตของอุปกรณ์ประเภท TTL หรือ LSTTL นั้นในบางครั้งหรือส่วนมากจำเป็นต้องต่อตัวต้านทานพูลอัพเพิ่มเข้ามาด้วยเพื่อให้เกิดความแน่ใจได้ว่าในระดับลอจิก 'high' ทางอินพุทของไอซีมีเสถียรภาพของระดับลอจิกที่แน่นอน

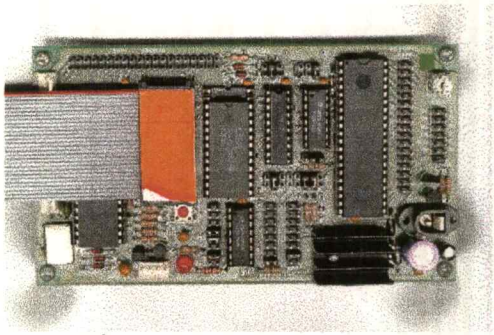
รูปแบบการขับแบบหนึ่งเฟสจะประกอบด้วยการจ่ายกำลังงานไปยังเฟสมอเตอร์หนึ่งเฟสที่เวลาใด ๆ ตามลำดับในลักษณะ A-B-C-D (หรือ D-C-B-A) ซึ่งในโหมดการกระตุ้นทำงานหรือโหมดการขับแบบเฟสเดียวนี้จะทำให้มีการสิ้นเปลืองกำลังงานต่ำมากๆ และแน่ใจได้ว่าจะมีความเที่ยงตรงในตำแหน่งของการหมุนโดยที่ไม่ต้องคำนึงถึงความสมดุลย์ของการหมุนในมอเตอร์ ส่วนรูปแบบการขับแบบ สองเฟส จะเป็นการขับกำลังงานไปยังเฟสของมอเตอร์ซึ่งเป็นเฟสที่ติดกันและในตำแหน่งที่เป็นลำดับไปคือ AB-BC-CD-DA ลำดับการขับในเฟสติดกันของคหมดนี้ จะเป็นการขับเพื่อรองรับและปรับปรุงค่าของแรงกับความเร็ว (torque-speed) ที่เกิดขึ้นและเป็นการหน่วงแรงได้ดีที่สุดและทำให้เกิดการสั่นสะเทือนน้อยที่สุดในมอเตอร์ นอกจากนั้นในรูปแบบการขับแบบครึ่งสเต็ปจะเป็นการขับแบบสลับกันระหว่างรูปแบบหนึ่งเฟสกับสองเฟส ดังนี้ A-AB-B-BC-C-CD-D-DA และก็จะแบ่งออกมาได้ทั้งหมด 8 สเต็ปเป็นลำดับกัน



รูปที่ 2.5 ไทมิ่งไดอะแกรมของไอซี

ทางด้านเอาต์พุตที่เป็นแบบไบโพลาร์นี้จะมีอัตราการขับกระแสซิงค์ออกมาสูงถึง 1.5 แอมป์ และที่แรงดันคงที่ 50 โวล์ตขณะที่อยู่ในสภาวะ ปิด (off) การต่อไดโอดแรงดันสูงแคลมป์ถึงกราวด์ภายในเพาเวอร์เอาต์พุตของไอซีนี้จะเป็นการช่วยป้องกันทรานเซียนต์เนื่องจากค่าความเหนี่ยวนำของขดลวดในสเต็ปปิงมอเตอร์ให้ลดลง นอกจากนั้นแล้วยังมีวงจรตรวจจับและป้องกันอุณหภูมิสูงเกินอันเนื่องมาจากการขับกระแสทางเอาต์พุตมีความผิดปกติจนทำให้เกิดความร้อนขึ้นสูงเกิน ซึ่งโดยคุณสมบัติแล้ว UNC5804B สามารถทำงานได้ในสภาวะของอุณหภูมิที่ตัวไอซีมีค่าระหว่าง -20 องศาเซลเซียส ถึง +85 องศาเซลเซียส

2.1.3 บอร์ดคอนโทรล CP-SB31



รูปที่ 2.6 บอร์ดคอนโทรล CP-SB31

CP-SB31 เป็นบอร์ดที่ใช้ในงานควบคุม ใช้ CPU ในตระกูล MCS 51 คือเป็นไมโครคอนโทรลเลอร์ โครงสร้างทางกายภาพของบอร์ด CP-SB31 มีดังนี้

ลักษณะของบอร์ด CP-SB31

CPU 8031 (ON BOARD OR 8032,8052,8751

MEMORY

มี socket ขนาด 28 pin 2 ตัว สามารถใส่หน่วยความจำได้สูงสุด 96 KB

I/O

8 * 3 บิทinput/output (8255)

8 * 1 บิท input/output (port1)

1 serial port (RS 232)

POWER

10 VDC power supply jack

5 VDC (regulate) 7805 on board

คุณลักษณะพิเศษของ CP-SB31

หน่วยความจำ สามารถเลือกได้ทั้งขนาด, ตำแหน่ง และลักษณะการทำงาน (data memory, code memory, code & data memory)

สามารถพัฒนาโปรแกรมได้ทั้งภาษาแอสเซมบลี้ (ร่วมกับ SB31-DEBUGGER) หรือ ภาษาเบสิค (เพื่อใช้ 8052 AH-BASIC) หรือ ET-8051 EM หรือใช้ ET EPROM EMULATOR ก็ได้

ต่อกับ LCD ได้ทันทีโดยไม่ต้องใช้ I/O พอร์ต

มี I/O พอร์ต ขนาด 8 บิท ถึง 4 พอร์ต

ต่อร่วมกับอุปกรณ์สนับสนุนของบริษัทอีทีที ได้ทันที เช่น SSRAC, RTC, 72IO, ET-AD ฯลฯ

การติดตั้งหน่วยความจำให้กับ CP-SB31

เนื่องจากCP-SB31 ถูกสร้างมาให้เป็นอิสระในการเลือกใช้หน่วยความจำได้หลายขนาดทั้ง EPROM และRAM รวมทั้งตำแหน่งแอดเดรสของหน่วยความจำผู้ใช้ก็ยังสามารถกำหนดได้ตามต้องการซึ่งทั้งหมดนี้ขึ้นอยู่กับการใส่ตำแหน่งของ JUMPER ต่างๆ ให้ถูกต้อง ซึ่ง U3 และ U4 จะถูกควบคุมด้วย JUMPER ซึ่งมีรายละเอียดดังนี้

U3 JP3 เลือกเบอร์ของชิปหน่วยความจำที่ใส่อยู่บน U3 (2764, 27256, 27512, 6264, 62556)

JP4 สำหรับเลือกว่าจะให้หน่วยความจำที่ U3 เป็น data memory หรือ code memory หรือเป็นทั้ง code & data memory

JP7 เลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำ U3

U4 JP5 เลือกเบอร์ของชิปหน่วยความจำที่ใส่อยู่บน U4 (27256, 6116, 6264, 62256)

JP6 เลือกลักษณะการทำงานของ U4 ว่าจะให้เป็น data memory หรือ code memory หรือเป็นทั้ง code & data memory

JP8 เลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำ U4

JP9 เลือกว่าจะอนุญาติให้มีการใช้ I/O port (8255) ภายนอกอีกหรือ ไม่ ถ้าไม่มีพอร์ตภายนอก U4 จะมีขนาดสูงสุดได้ถึง 32 KB

ข้อจำกัดในการติดตั้งหน่วยความจำแบบแยก DATA และ PROGRAM ออกจากกัน

เมื่อดูจากวงจรของ CP-SB31 จะเห็นว่า ส่วนของ JP7 และ JP8 ด้านหนึ่งจะถูกต่อถึงกันในตำแหน่งที่ตรงกัน (เช่น 0000 ของ JP7 จะต่อกับ 0000 ของ JP8) จึงทำให้เกิดการซ้อน PAGE ขึ้นในกรณีที่ขนาดหน่วยความจำ U3 และ U4 มีขนาดไม่เท่ากัน ถึงแม้ว่าผู้ใช้จะใส่หน่วยความจำแต่ละตัวเริ่มต้นที่ใด และควรสิ้นสุดที่ใด

ตัวอย่างเช่น ที่ U3 ใส่หน่วยความจำ 8 KB และที่ U4 ใส่หน่วยความจำขนาด 2 KB ที่ตำแหน่งของหน่วยความจำน้อย (U4) จะเกิดการซ้อนตัวเอง เช่น ในตำแหน่ง 2000 H และตำแหน่ง 2800 H จะเป็นตำแหน่งเดียวกัน (เกิดการซ้อน)

รายละเอียดเกี่ยวกับ CONNECTOR

CP-SB31 ได้ถูกออกแบบมาให้ใช้ได้กับบอร์ดอินเตอร์เฟสต่างๆ ของบริษัท อีทีที โดยเฉพาะอย่างยิ่งบอร์ดประเภท อินพุท/เอาท์พุท ดังมีรายละเอียดดังต่อไปนี้

EXP1 เป็น connector ขนาด 40 pin ซึ่งมีสัญญาณที่ขาต่างๆ คล้ายกับสัญญาณที่ต่อจาก CP-U Z80 ทั้งนี้เพื่อสนับสนุนบอร์ดต่างๆ เช่น 72IOZ80, ET-RTC เป็นต้น แต่เนื่องจาก CP-SB31 ใช้ CPU คนละตระกูลกับ Z80 จึงมีสัญญาณบางสัญญาณไม่ตรงกัน

EXP2 เป็น อินพุท/เอาท์พุท พอร์ตอิสระขนาด 8 บิท

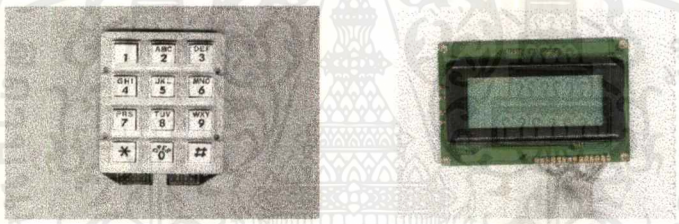
EXP3 เป็นขาของ Interrupt และ Timer/Counter ของตัว CPU

EXP4 เป็น connector ขนาด 34 ขา ซึ่งมีการวางตำแหน่งของขาตรงกับบอร์ด 72IO ซึ่งสามารถต่อกับอุปกรณ์ I/O ต่างๆ เช่น SSRAC, ET-AD เป็นต้น

EXP5 เป็น connector ขนาด 14 ขาที่ ใช้ต่อกับ LC ได้โดยตรง

2.1.4 จอแสดงผลและคีย์บอร์ด

จอแสดงผลแบบ LCD และคีย์บอร์ดจากรูปที่ 2.7 (ก) และ 2.7 (ข)

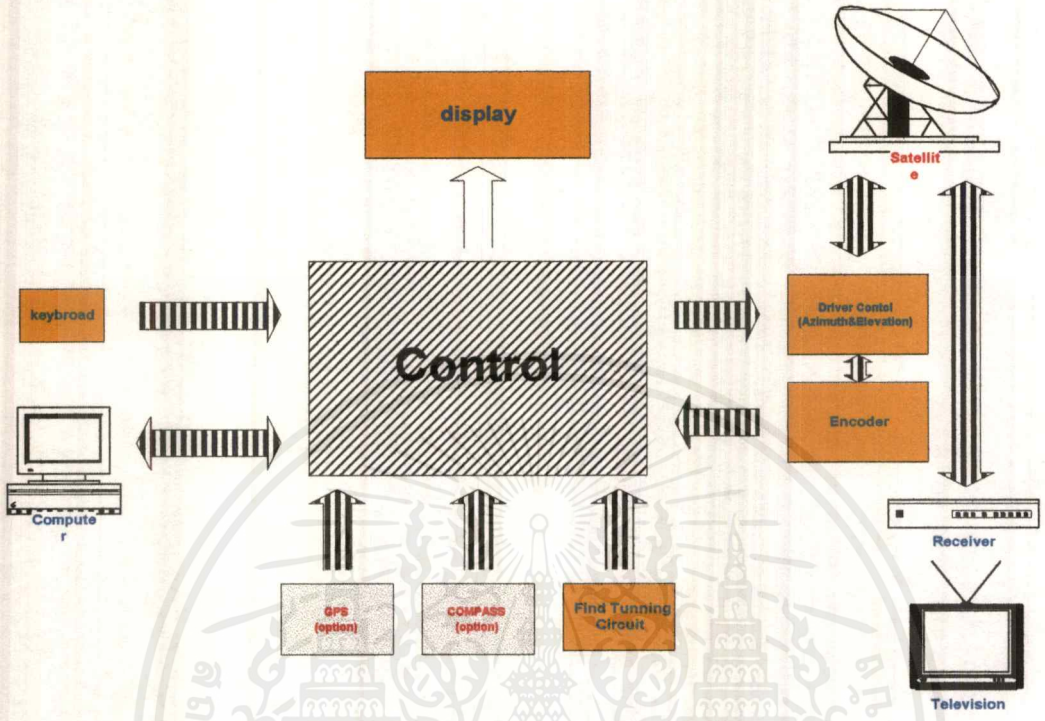


(ก) (ข)

รูปที่ 2.7 จอแสดงผลและคีย์บอร์ด

2.1.5 หลักการทำงานของโครงงาน

การสื่อสารส่วนใหญ่ ที่มีความเกี่ยวข้องกับดาวเทียม และอุปกรณ์ที่มีความสำคัญตัวหนึ่งก็คือ จานรับสัญญาณดาวเทียม แต่เนื่องจากการติดตั้งจานรับสัญญาณดาวเทียมให้มีประสิทธิภาพสูงสุดนั้น มีขั้นตอนและวิธีการค่อนข้างยุ่งยาก อีกทั้งยังต้องอาศัยผู้ที่มีความรู้ความเข้าใจเกี่ยวกับการติดตั้งเป็นผู้ดำเนินงาน จากที่กล่าวมาแล้วจะเห็นได้ว่าการติดตั้งจานรับสัญญาณดาวเทียมนั้นไม่ง่าย นักดังนั้นคณะผู้จัดทำจึงได้พยายามศึกษาและนำอุปกรณ์ควบคุมกว้างๆมาใช้เพื่อให้การติดจานรับสัญญาณดาวเทียมสะดวกง่ายดาย มีประสิทธิภาพและสามารถใช้งานได้กับบุคคลทั่วไปโดยผู้ใช้ไม่จำเป็นต้องมีความรู้ทางด้านการติดตั้งจานรับสัญญาณดาวเทียมมาก่อน หลักการทำงานของโครงงาน จะมี บล็อกไดอะแกรมดังนี้



รูปที่ 2.8 บล็อกไดอะแกรม ของโครงงาน

จากบล็อกไดอะแกรมแสดงส่วนประกอบของโครงงานทั้งหมดที่สมบูรณ์แล้ว จะเห็นได้ว่าการควบคุมให้จานรับสัญญาณดาวเทียมเคลื่อนที่นั้นสามารถควบคุมได้ 2 วิธีคือ ควบคุมจากไมโครคอนโทรลเลอร์ และควบคุมจากไมโครคอมพิวเตอร์ ส่วน ตัวบอกพิกัดตำแหน่ง (GPS) และ เข็มทิศ อิเล็กทรอนิกส์ (COMPASS) จะเป็นส่วนของ ส่วนเพิ่มเติม (OPTION) โดยถ้าสามารถจัดหาได้โครงงานนี้จะเป็นโครงงานที่มีการทำงานแบบอัตโนมัติได้

ขั้นตอนการทำงานของโครงงาน

สำหรับการทำงานของโครงงานนี้ในส่วนของโปรแกรมไมโครคอนโทรลเลอร์ และ ไมโครคอมพิวเตอร์ จะมีลักษณะ และหลักการทำงานคล้ายกันโดยจะมีขั้นตอนในการทำงานดังนี้

- ผู้ใช้เลือกดาวเทียมที่ต้องการจะติดต่อด้วย
- หากไม่มีโหมดการทำงานแบบอัตโนมัติจะต้อง ใส่พิกัดตำแหน่งของจานรับสัญญาณดาวเทียมด้วย
- เมื่อได้ชื่อดาวเทียมที่ต้องการจะรับสัญญาณแล้ว โปรแกรมจะทำการคำนวณ มุมกวาด และมุมเงยของจานรับสัญญาณดาวเทียม โดยใช้ตำแหน่งของจานสายอากาศที่ ได้จาก เครื่องบอกพิกัด แล้วนำค่าที่ได้ไปคำนวณออกมา ส่วนของ เข็มทิศอิเล็คทรอนิคส์ จะเป็นตัวบอกทิศเหนือ เพื่อ ใช้อ้างอิงกับมุม อะซีมูธ

- เมื่อส่วนคอนโทรลได้รับค่ามุมต่างๆ แล้วจะทำการส่งสัญญาณควบคุมไปยังชุดขับจาน เพื่อให้จานรับสัญญาณเคลื่อนที่ไปยังตำแหน่งที่คำนวณได้ ซึ่งชุดคอนโทรลจะรู้ตำแหน่งที่หมุนไปของจานสายอากาศ จาก ชุด เอ็นโค้ดเดอร์ ซึ่งจะให้ผลที่เที่ยงตรงกว่าการนับพัลส์ของ สเต็ปปิง มอเตอร์
- เมื่อจานสายอากาศ เคลื่อนที่ไปยังตำแหน่งที่คำนวณได้แล้ว สัญญาณที่รับได้ส่วนหนึ่งจะถูกส่งไปยังชุด ปรับละเอียด เพื่อทำการปรับตำแหน่งของจานสายอากาศไปยังตำแหน่งที่ให้สัญญาณแรงที่สุด

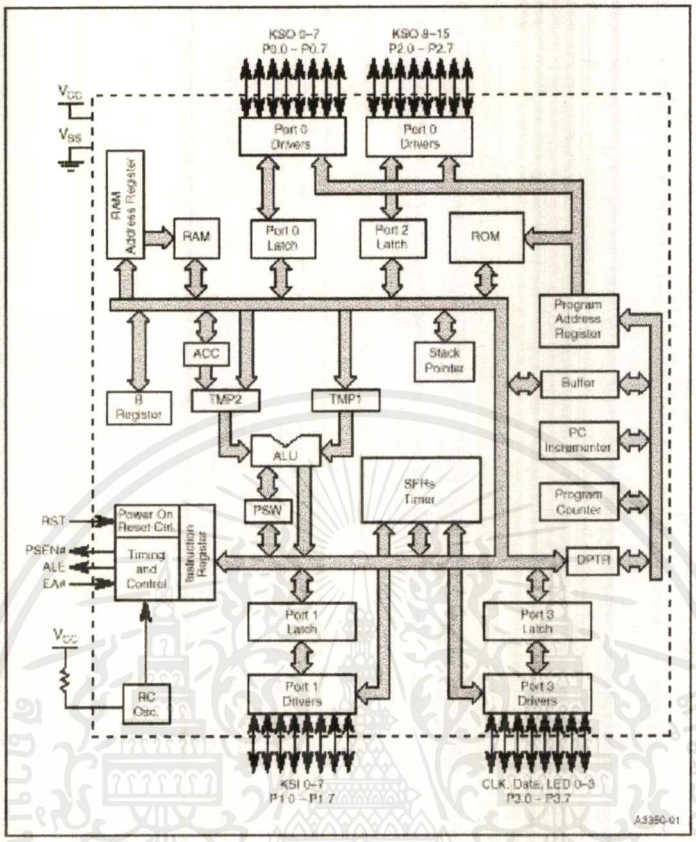
2.2 โครงสร้างของ MCS-51

2.2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

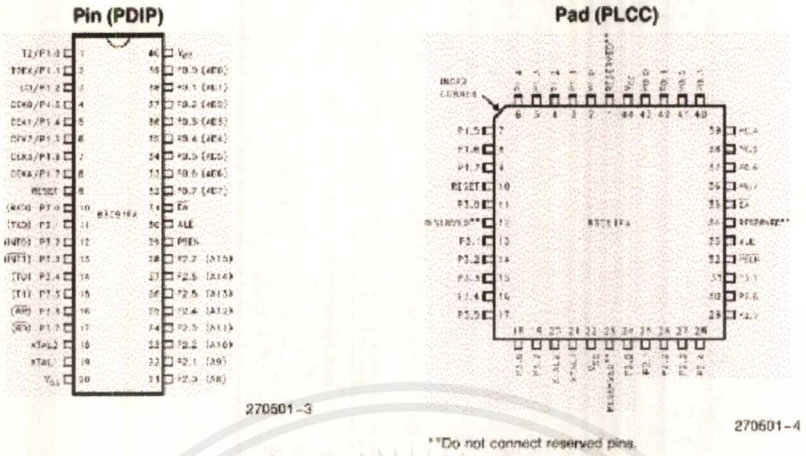
- ต้องการแหล่งจ่ายไฟ +5 โวลท์ ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ 8031,8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์ หน่วยความจำสำหรับโปรแกรมและดาต้า (Program Memory และ Data Memory) แยกออกจากกันอย่างละ 64 กิโลไบต์
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 μS เมื่อทำงานที่ความถี่ 12 MHz
- มี Timer/Counter ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด)ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ท ทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
- มีคำสั่งในการทำ AND,OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิตและ 1 บิต

2.2.2 โครงสร้างภายในของ 8051

MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวก สำหรับโปรแกรมเมอร์ที่จะเขียนโปรแกรมด้วยภาษาเบสิก บล็อกไดอะแกรม และตำแหน่งขา ของ MCS-51 แสดงจากรูป 2.9



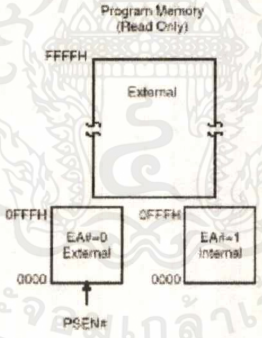
รูปที่ 2.9 บล็อกไดอะแกรมของ MCS-51



รูปที่ 2.10การจัดวางขาของ 8051

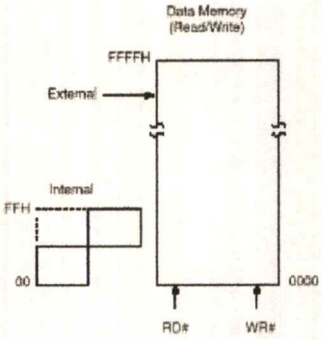
2.2.3 การแบ่งประเภทของหน่วยความจำ

หน่วยความจำที่ใช้กับ MCS-51 มีอยู่ด้วยกัน 2 ชนิด คือ Program Memory และ Data Memory Program Memory ซึ่งเป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุอยู่ในชิพ 8051 ส่วนที่ เป็น Program Memory ก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง ดังแสดงจากรูป 2.11



รูปที่ 2.11 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8051

Data Memory เป็นหน่วยความจำที่ใช้เก็บข้อมูล หน่วยความจำนี้ สามารถเขียนข้อมูลลงไป และอ่านข้อมูลออกมาได้ซึ่งเป็นหน่วยความจำภายในชิพมีเพียง 128 ไบต์ สำหรับเบอร์ 8051 และ 256 ไบต์ สำหรับเบอร์ 8052 ส่วนหน่วยความจำภายนอกชิพมี 64 กิโลไบต์ดังแสดงในรูปที่ 2.12



รูปที่ 2.12 ผังหน่วยความจำสำหรับ Data Memory เบอร์ 8051

บางครั้งอาจจะสงสัยว่าตำแหน่งของหน่วยความจำสำหรับโปรแกรมและดาต้า มีตำแหน่งที่ซ้อนกันซีพียูจะรู้ได้อย่างไรว่าติดต่อกับหน่วยความจำที่เป็นโปรแกรมและดาต้า บริษัทอินเทลได้ออกแบบแยกคำสั่งออกเป็น 3ส่วนคือ

MOV ใช้ติดต่อกับ RAM ภายใน

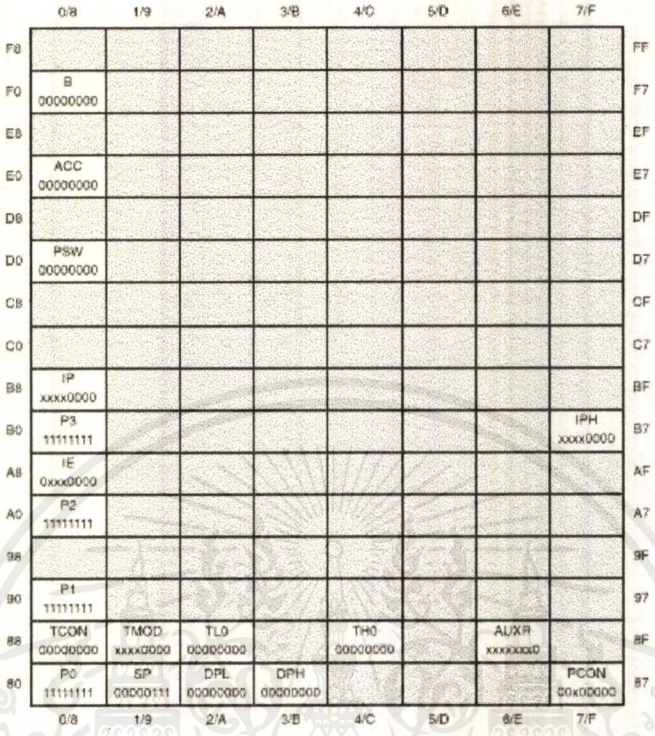
MOVC ใช้ติดต่อ Program Memory

MOVX ใช้ติดต่อ Data Memory ภายนอกชิพโดยระบุตำแหน่งผ่านรีจิสเตอร์ DPTR

ชิพเบอร์ 8052 จะมีพื้นที่บริเวณ 80h-FFh ซึ่งถ้าจะเขียนอ่านข้อมูล ณ บริเวณนี้จะเข้าถึงข้อมูลโดยทางอ้อมเท่านั้น

พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางอ้อมเท่านั้น (indirect Address Area)

พื้นที่หน่วยความจำบริเวณ (80h-FFh) เป็นพื้นที่ซ้อนกันอยู่อย่างละ 128 ไบต์โดยส่วนแรกจะเป็น SFR แอดเดรสและ indirect Address Area ดังนั้น ผู้เขียนโปรแกรมถ้าจะติดต่อกับ SFR จะต้องใช้คำสั่งแบบเข้าถึงข้อมูลโดยตรงเท่านั้น (Direct Address Area) ส่วนพื้นที่อีกส่วนหนึ่งจะเข้าถึงข้อมูลแบบทางอ้อมเท่านั้น (indirect Address Only)



รูปที่ 2.13 แสดงตำแหน่งบิตต่างๆ ในตำแหน่งแอดเดรส (20-2Fh) รีจิสเตอร์หน้าที่พิเศษ Special

2.2.4 ไทม์เมอร์/ เคาน์เตอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีรีจิสเตอร์พิเศษ ที่สามารถเลือกให้งานเป็นไทม์เมอร์ หรือเคาน์เตอร์อย่างใดอย่างหนึ่งมีอยู่ด้วยกัน 2 ตัว คือ ไทม์เมอร์ 0 และ ไทม์เมอร์ 1 แต่ละตัวเป็นแบบ 16 บิติ ไทม์เมอร์ 0 ประกอบด้วย TL0,TH0 ไทม์เมอร์ 1 ประกอบด้วย TL1,TH1 (ใน MCS-51 เบอร์ 8052 ,8032 จะมีไทม์เมอร์ 2 ด้วย ซึ่งจะไม่ขอกล่าวในที่นี้) การกำหนดการทำงานของไทม์เมอร์ 0 และ ไทม์เมอร์1 สามารถเลือกให้มีการทำงานเป็น ไทม์เมอร์ หรือ เคาน์เตอร์อย่างใดอย่างหนึ่ง โหมดไทม์เมอร์ ค่าในรีจิสเตอร์จะถูกเพิ่มค่าทุกๆ แมชชีนไซเคิล 1/12 คาบเวลาของ ออสซิลเลเตอร์ รีจิสเตอร์จะถูกเพิ่มค่าทีละหนึ่งเมื่อมีการเปลี่ยนสถานะ ซึ่งตรวจจับได้จากขา T0,T1 อยู่ที่ขา 14 และ 15 ตามลำดับ แล้วแต่จะเลือก สามารถเลือกได้โดยกำหนดค่าบิต C//T ในรีจิสเตอร์ใช้งานเฉพาะ TMOD ดังแสดงจากรูปที่ 2.14

TMOD Address: 89H
Reset State: XXXX 0000B

7							0
—	—	—	—	GATE	C/T#	M1	M0

Bit Number	Bit Mnemonic	Function
7-4	—	Reserved.
3	GATE	Timer 0 Gate: When GATE = 0, run control bit TR0 gates the input signal to the timer register. When GATE = 1 and TR0 = 1, external signal INT0# gates the timer input.
2	C/T#	Timer 0 Counter/Timer Select: C/T# = 0 selects timer operation: timer 0 counts the divided-down system clock. C/T# = 1 selects counter operation: timer 0 counts negative transitions on external pin T0.
1, 0	M1, M0	Timer 0 Mode Select: M1 M0 0 0 Mode 0: 8-bit timer/counter (TH0) with 5-bit prescaler (TL0) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow 1 1 Mode 3: TL0 is 8-bit timer/counter. TH0 is an 8-bit timer.

รูปที่ 2.14 Timer/Counter Mode Register (TMOD) อยู่ใน SFR ตำแหน่งที่ (89H)

TCON Address: 88H
Reset State: 0000 0000B

7							0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Function
7	TF1	TH0 Overflow Flag (Mode 3 only) Set by hardware when TH0 overflows. Cleared by hardware when the processor vectors to the interrupt routine.
6	TR1	TH0 Run Control Bit (Mode 3 only) Set and cleared by software to switch TH0 on or off.
5	TF0	Timer 0 Overflow Flag: Set by hardware when TH0 overflows. Cleared by hardware when the processor vectors to the interrupt routine.
4	TR0	Timer 0 Run Control Bit: Set and cleared by software to turn timer 0 on or off.
3	IE1	Interrupt 1 Flag for INT1#: Set by hardware when an INT1# event is detected. Edge- or level-triggered (see IT1). Cleared by interrupt vector if edge-triggered.
2	IT1	Interrupt 1 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 1. Clear this bit to select level-triggered (active-low).
1	IE0	Interrupt 1 Flag for INT0#: Set by hardware when an INT0# event is detected. Edge- or level-triggered (see IT0). Cleared by interrupt vector if edge-triggered.
0	IT0	Interrupt 0 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 0. Clear this bit to select level-triggered (active-low).

รูปที่ 2.15 TCON Timer Control Register

ข้อควรสังเกตุ

T หมายถึง Timer/Counter หรือ Type

R หมายถึง RUN

F หมายถึง Flag

E หมายถึง External

1,0 หมายถึง Channel หรือ Channel 1 บางทีใช้ x เช่น TRx

2.2.5 IE (Interrupt Enable Register)

ใช้ควบอินเตอร์รัปต์ได้ 6 แหล่ง คือ จาก (TI,RI), TF0, TF1, INT0, INT1 เราสามารถสั่งห้ามหรือไม่สั่งห้ามอินเตอร์รัปต์ได้จากรีจิสเตอร์ชุดนี้ดังมีรายละเอียดดังนี้บน

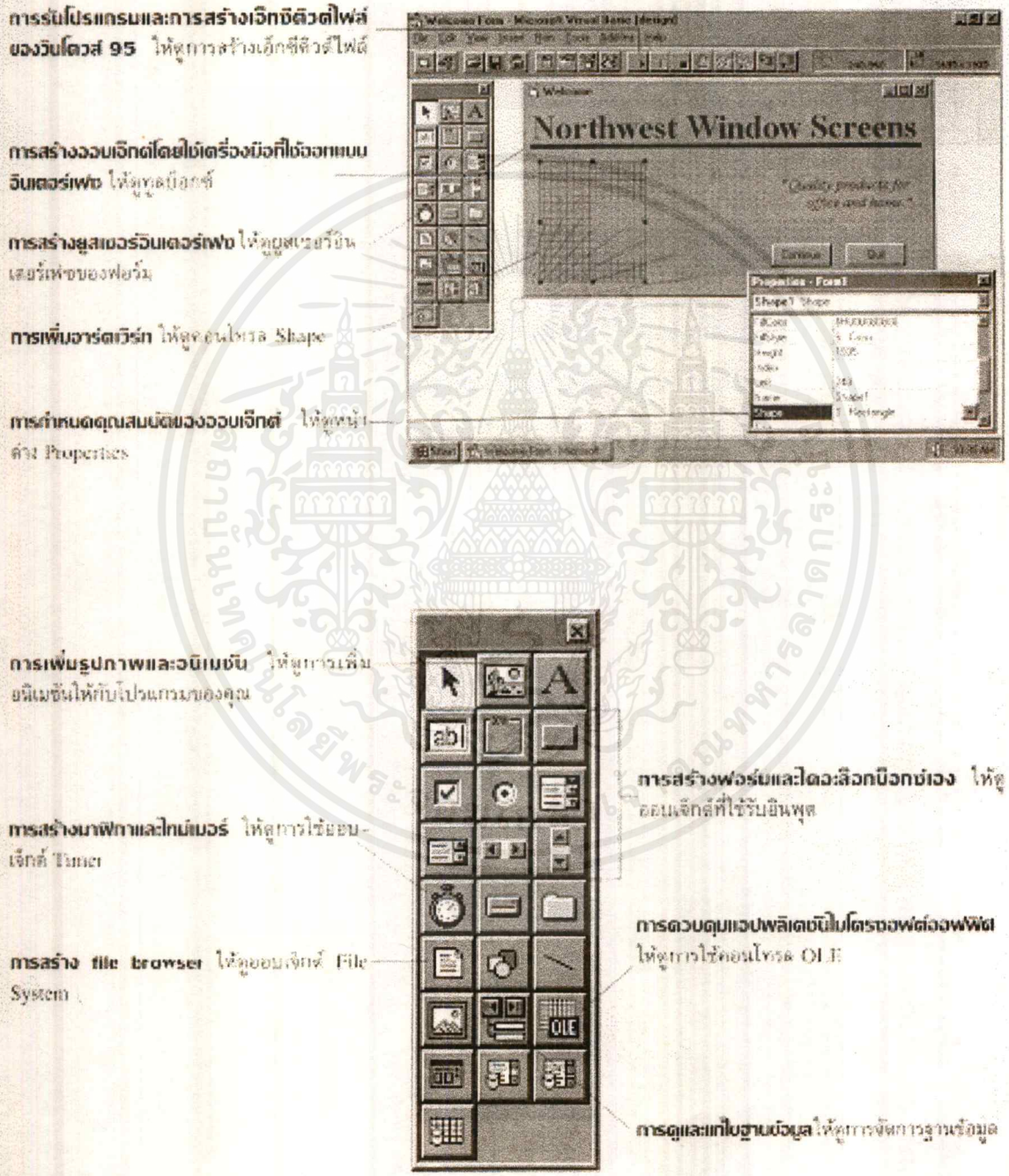
IE — Address: A8H, Reset State: 0XXX 0000B

7							0
EA	—	—	—	ET1	EX1	ET0	EX0

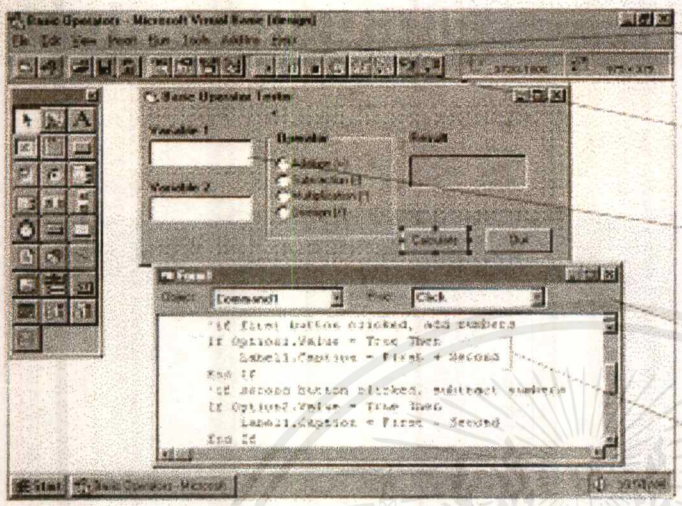
Bit Number	Bit Mnemonic	Function
7	EA	Global Interrupt Enable: Setting this bit enables all interrupts enabled by bits 0-2 in this register.
6:4	—	Reserved
3	ET1	TH0 Overflow Interrupt Enable (Mode 3 only).
2	EX1	External Interrupt 1 Enable: Setting this bit enables external interrupt 1.
1	ET0	Timer 0 Overflow Interrupt Enable: Setting this bit enables the timer 0 overflow interrupt.
0	EX0	External Interrupt 0 Enable: Setting this bit enables external interrupt 0.

รูปที่ 2.16 Interrupt Enable Register

2.3 การใช้โปรแกรม VISUAL BASIC ในการสร้างแอปพลิเคชันแบบ 32 บิต



รูปที่ 2.17 ส่วนต่างๆของโปรแกรม VISUAL BASIC



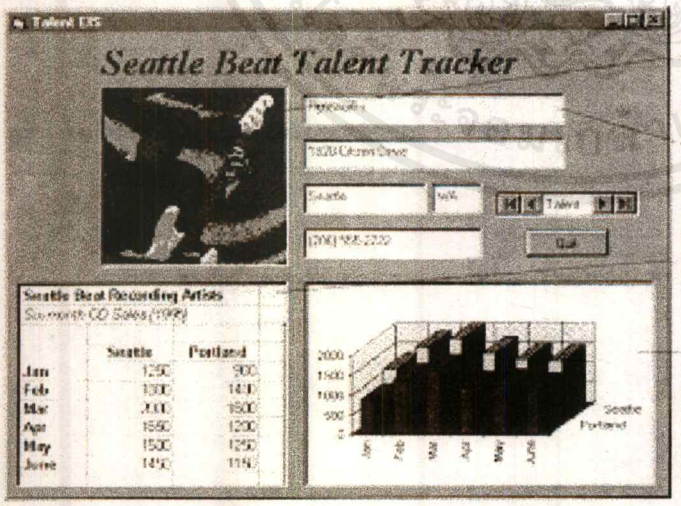
การตรวจดูออบเจ็กต์ในโปรแกรมของคุณ ให้ดู Visual Object Browser

การดีบักแอปพลิเคชัน ให้ดูการค้นหาและแก้ไขความผิดพลาด

การทำงานกับข้อมูลในโปรแกรม ให้ดูการใช้ตัวแปรเก็บข้อมูล

การป้อนและแก้ไขสเตตเมนต์ในวิชวลเบสิก ให้ดูการเขียนโค้ด

การเพิ่มโปรแกรมลอจิก ให้ดูโครงสร้าง If.. Then



การรวมข้อมูลมาใช้ร่วมกัน ให้ดูการสร้างระบบข้อมูลกิจการ (EIS)

การแสดงฟิลด์ข้อมูลของฐานข้อมูล ให้ดูการใช้ยูบแจ็กซ์ TextBox แสดงข้อมูล

การสร้างการเชื่อมต่อพิเศษกับเวิร์กชีตของเวิร์กบุ๊ค ให้ดูการใช้คอนโทรล OLE

การเพิ่มชาร์ตที่อัปเดตอัตโนมัติ ให้ดูการเพิ่มออบเจ็กต์แอปพลิเคชัน

รูปที่ 2.17ส่วนต่างๆของโปรแกรม VISUAL BASIC (ต่อ)

VISUAL BASIC ไดรับการออกแบบมาโดยเฉพาะเพื่อให้สามารถสร้างแอปพลิเคชันแบบ 32 บิต ที่ทำงานอยู่บนเร็วขึ้นปลอดภัยกว่าเดิมและใช้งานในสภาพแวดล้อมที่เป็นมัลติทาสก์ได้ง่ายกว่าแอปพลิเคชันแบบ16 ระบบปฏิบัติการแบบ 32 บิต ของไมโครซอฟต์อย่างเช่น WINDOWS 95 และ WINDOWS NT ได้ แอปพลิเคชันแบบ 32 บิตจะได้เปรียบตรงที่อ้างหน่วยความจำได้มากขึ้นและสามารถใช้ความสามารถในการจัดการหน่วยความจำของ WINDOWS 95 และ WINDOWS NT ได้อีกด้วย ดังนั้นแอปพลิเคชันเหล่านี้จึงทำงานบิตเดิม เมื่อคุณคอมไพล์โปรแกรม VISUAL BASIC บน WINDOWS 95 หรือ WINDOWS NT คุณจะได้รับลักษณะพิเศษต่างๆของแอปพลิเคชันแบบ 32 บิตโดยอัตโนมัติเพราะโปรแกรมของคุณถูกสร้างจากคอมไพเลอร์แบบ 32 บิตที่คุ้นเคยกับลักษณะของวินโดส์95 วินโดส์ NT และไมโครซอพต์ WIN 32 API (Application Program Interface) แอปพลิเคชันแบบ 32 บิต ของ VISUAL BASIC ที่ทำงานบน วินโดส์ 95 และวินโดส์ NT
จะได้รับประโยชน์ ต่างๆดังนี้

- เข้าถึงหน่วยความจำทั้งหมดที่อ้างตำแหน่งแบบ 32 บิต ของไมโคร โปรเซสเซอร์ 386 486 และเพรเทียมของอินเทล โดยใช้หน่วยความจำของระบบได้สูงสุด 2 กิโลไบต์
- เพิ่มประสิทธิภาพในการทำงานเพราะการคำนวณและการทำงานของหน่วยความจำเป็นแบบ 32 บิต
- เพิ่มความสามารถในการป้องกันระบบล้มที่เป็นผลมาจากการทำงานที่ไม่เหมาะสมของโปรแกรมอื่นๆ โดยที่แอฟปลิเคชันของ VISUAL BASIC จะทำงานอยู่ในบริเวณหน่วยความจำที่กันไว้ใช้เองทำให้การผิดพลาดเนื่องจากโปรแกรมอื่นได้
- สามารถใช้งานแบบมัลติทาสก์ได้อย่างเต็มที่กล่าวคือ สามารถที่จะสลับการใช้งานจากโปรแกรมหนึ่งไปอีกโปรแกรมหนึ่งได้ทันที ผู้ใช้สามารถสลับการใช้งานไปมาระหว่างแอปพลิเคชันของ VISUAL BASIC ได้ตามต้องการซึ่งจะทำให้ผู้ใช้ควบคุมวิธีการต่างๆ ได้ดีขึ้น

สามารถเข้าถึง WIN 32 APIซึ่งจะทำให้โปรแกรมเมอร์ที่มีประสพการณ์สามารถใช้ความสามารถขั้นสูงต่างๆ ได้ในโปรแกรม

2.3.1 การวางแผนการเขียนโปรแกรม

ขั้นตอนการเขียนโปรแกรมคือการกำหนดว่าคุณต้องการให้โปรแกรมแสดงผลลัพธ์อะไร ฟังดูเหมือนง่ายแต่ไม่ค่อยมีใครเห็นด้วยนักเพราะแม้แต่โปรแกรมเมอร์ที่ดีที่สุดยังมีปัญหาในการสร้างแอปพลิเคชันเลย การวางแผนการเขียนโปรแกรมเป็นเรื่องเล็กน้อยเหมือนกับการเตรียมบาร์บีคิว เพื่อให้การจัดบาร์บีคิวเป็นไปอย่างราบรื่น คุณจำเป็นต้องเตรียมการไว้ก่อนล่วงหน้า คุณต้องจัดเมนู เชิญเพื่อนๆ ซื้ออาหารและทำความสะอาดบ้านของคุณ แม้การจัดบาร์บีคิวสร้างความบันเทิงให้คุณแต่ถ้าเพื่อนๆ คุณทิ้งขยะไว้ให้ ทำนองเดียวกันโปรแกรมต่างๆ คงไม่ดีที่สุดแน่ถ้าโปรแกรมถูกสร้างขึ้นด้วยความคิดที่มีความผิดพลาดซ่อนอยู่

เมื่อคุณกำลังเรียนรู้โปรแกรม คุณควรจะใช้เวลาบางส่วนคิดถึงวิธีแก้ปัญหาด้วยการเขียนโปรแกรมก่อนจะไปนั่งหน้าเครื่องคอมพิวเตอร์ การวางแผนล่วงหน้าจะช่วยลดเวลาในการพัฒนาโปรแกรมลงและบางครั้งคุณอาจจะได้ผลลัพธ์ที่ถูกใจกว่าก็ได้ ส่วนหนึ่งของการวางแผนอาจจะเป็นการจัดลำดับขั้นตอนการเขียนโปรแกรมที่เรียกกันว่า อัลกอริทึม อัลกอลิทึมในการจัดบาร์บีคิวอาจจะมีลักษณะดังรูปต่อไปนี้

2.3.2 การออกแบบยูสเซอร์อินเตอร์เฟซ

หลังจากที่ตั้งเป้าหมายให้โปรแกรมของคุณแล้ว เรื่องสำคัญที่ต้องคิดต่อไปคือโปรแกรมควรมีหน้าตาเป็นอย่างไรและจะนำข้อมูลไปใช้ได้อย่างไร หน้าจอและรูปภาพทั้งหมดที่ใช้ในโปรแกรมรวมเรียกว่ายูสเซอร์อินเตอร์เฟซ ยูสเซอร์อินเตอร์เฟซจะครอบคลุมไปถึงเมนู ไดอะล็อกบ็อกซ์ ปุ่ม วัตถุ และรูปภาพที่ผู้ใช้มองเห็นขณะที่ใช้งานแอปพลิเคชันนั้น ระบบการเขียนโปรแกรมของวิชวลเบสิกให้คุณสร้างส่วนประกอบทั้งหมดของแอปพลิเคชันบนวินโดวส์ได้อย่างรวดเร็วและมีประสิทธิภาพ คุณจะได้เห็นส่วนประกอบหลายๆ อย่างในโปรแกรมวินโดวส์ 95 ที่สร้างจากวิชวลเบสิก

ถ้าคุณคุ้นเคยกับโปรแกรมบนวินโดวส์ คุณคงจำองค์ประกอบที่เป็นยูสเซอร์อินเตอร์เฟซได้เกือบทั้งหมด เมนูจะประกอบไปด้วยคำสั่งมากมายที่ใช้งานในแอปพลิเคชัน ไดอะล็อกบ็อกซ์ ปุ่ม และเมาส์พอยน์เตอร์จะช่วยผู้ใช้จัดหาข้อมูลให้กับโปรแกรม ส่วนหน้าต่างและสกรอลล์บาร์จะให้ผู้ใช้ดูและนำร่องไปยังข้อมูลที่แสดงบนหน้าจอได้ เพื่อให้ใช้ความสามารถของโปรแกรมวิชวลเบสิกได้อย่างเต็มที่จึงเป็นเรื่องสำคัญที่คุณควรใช้ส่วนประกอบยูสเซอร์อินเตอร์เฟซของวินโดวส์ไปในแนวทางมาตรฐานเดียวกัน ไมโครซอฟท์ได้จัดพิมพ์ชุดแนะนำการออกแบบแอปพลิเคชันบนวินโดวส์ คุณจะได้เรียนรู้เกี่ยวกับข้อแนะนำสำคัญๆ ที่เกี่ยวกับอินเตอร์เฟซหลังจากอ่านหนังสือเล่มนี้แล้ว และเนื่องจากแอปพลิเคชันบนวินโดวส์ส่วนใหญ่ถูกใช้โดยบุคคลทั่วไปมากกว่าที่จะเป็นโปรแกรมเมอร์จึงจำเป็นต้องออกแบบให้มีลักษณะคล้ายคลึงกันมากที่สุดเท่าที่เป็นไปได้

ตั้งคำถามให้ตัวคุณเอง

เมื่อคุณวางแผนสร้างโปรเจ็กต์วิชวลเบสิก คุณจะพบว่าการร่างยูสเซอร์อินเตอร์เฟซคร่าวๆ ที่คุณชอบเพื่อนำไปเสนอให้กับผู้ใช้เป็นประโยชน์อย่างยิ่ง การทำเช่นนี้นอกจากจะช่วยการออกแบบอินเตอร์เฟซ แล้วยัง

ช่วยให้คุณได้คิดด้วยว่าโปรแกรมควรทำงานอย่างไร ให้ลองถามคำถามต่อไปนี้กับตัวคุณเองบางทีอาจทำให้เกิดประโยชน์ขึ้นมาบ้าง

- วัตถุประสงค์หรือเป้าหมายของโปรแกรมที่คุณเขียนคืออะไร
- ผู้ใช้งานโปรแกรมนี้เป็นใคร
- โปรแกรมควรมีรูปร่างหน้าตาเป็นอย่างไรตอนเริ่มต้น
- ข้อมูลอะไรบ้างที่ให้ผู้ใช้ป้อนเข้าโปรแกรม
- โปรแกรมประมวลผลอินพุตอย่างไร
- โปรแกรมจะสร้างข้อมูลผลลัพธ์อะไร

ก่อนที่คุณจะรู้ว่าทำอย่างไรให้โปรแกรมทำงานตามที่ต้องการ คุณควรจะคิดก่อนว่าจะให้โปรแกรมทำอะไรบ้าง และควรมีรูปร่างหน้าตายังไง หลังจากทำงานเบื้องต้นเหล่านี้เสร็จคุณพร้อมแล้วที่จะสร้างโปรแกรมด้วยวิชวลเบสิก

2.3.3 การสร้างโปรแกรมด้วยวิชวลเบสิก

การสร้างแอปพลิเคชันบนวินโดวส์ด้วยวิชวลเบสิกจะเกี่ยวข้องกับขั้นตอน 3 ขั้นตอนดังนี้

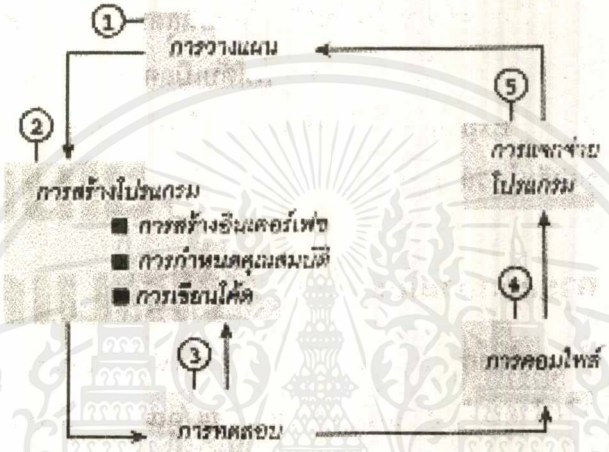
1. สร้างยูสเซอร์อินเตอร์เฟซโดยใช้คอนโทรลของวิชวลเบสิก
2. การกำหนดคุณลักษณะ หรือคุณสมบัติขององค์ประกอบที่เป็นยูสเซอร์อินเตอร์เฟซตามที่จำเป็น
3. การเขียนโปรแกรมให้กับองค์ประกอบที่ใช้ยูสเซอร์อินเตอร์เฟซตามที่จำเป็น

การสร้างโปรแกรมบนวินโดวส์โดยใช้วิชวลเบสิกนั้นง่ายมาก ถ้าคุณเคยใช้โปรแกรมวาดภาพ เช่น โปรแกรม Paint คุณจะมีทักษะหลายอย่างที่จำเป็นในการสร้างยูสเซอร์อินเตอร์เฟซที่มีประสิทธิภาพสูง การสร้างองค์ประกอบนั้นโดยคลิกแล้วลากเมาส์ ปกติแล้วการคลิกเป็นการกำหนดมุมจากนั้นลากเมาส์เป็นกรอบสี่เหลี่ยมขนาดตามที่คุณต้องการ หลังจากที่สร้างเสร็จแล้ว (สมมติว่าเป็น TextBox) คุณสามารถปรับแต่งองค์ประกอบนั้นได้โดยกำหนดคุณสมบัติขององค์ประกอบ สำหรับ TextBox คุณสามารถกำหนดคุณสมบัติเพื่อให้แสดงผลข้อความเป็นแบบตัวหนา ตัวเอียงหรือตัวอักษรขีดเส้นใต้ได้เป็นต้น

คุณทำให้โปรแกรมสมบูรณ์ได้โดยพิมพ์โค้ดให้กับองค์ประกอบต่างๆ ที่เป็นยูสเซอร์อินเตอร์เฟซในหน้าต่างพิเศษที่ชื่อหน้าต่าง Code การเขียนโค้ดโปรแกรมจะช่วยให้คุณควบคุมการทำงานของโปรแกรมได้ดีกว่าที่ได้รับจากการออกแบบคอนโทรลเพียงอย่างเดียว การใช้โค้ดโปรแกรมนี้จะทำให้คุณสามารถถ่ายทอดความคิดว่าโปรแกรมจะประมวลผลข้อมูลอินพุต เอาต์พุตได้อย่างไร ภาษาวิชวลเบสิก มีสเตตเมนต์ ฟังก์ชัน และตัวอักษรพิเศษอยู่นับร้อย แต่ส่วนมากแล้วโปรแกรมของคุณจะคีย์เวิร์ดที่จำง่ายราวๆ 20 หรือ 30 อัน ภาษาวิชวลเบสิกไม่ได้มีใช้อยู่ในวิชวลเบสิกเพียงอย่างเดียวแต่ใช้รวมอยู่ในไมโครซอฟต์เอ็กเซล ไมโครซอฟต์แอ็กเซส ไมโครซอฟต์โปรเจ็กต์ และแอปพลิเคชันบนวินโดวส์อื่นๆ ด้วย ถ้าคุณเคยใช้เวอร์ชันล่าสุดของโปรแกรมอันอันหนึ่งข้างต้น หรือเคยใช้ภาษาเบสิกมาก่อนคุณจะรู้สึกคุ้นเคยกับคีย์เวิร์ด สเตตเมนต์ และฟังก์ชันต่างๆ ในวิชวลเบสิก 4 นี้

2.3.4 การทดสอบการคอมไพล์และการแจกจ่ายโปรแกรม

หลังจากที่คุณสร้างโปรแกรมใช้งานจริงขึ้นมา คุณจำเป็นต้องทดสอบโปรแกรมนั้นอย่างระมัดระวัง คอมไพล์เป็นเอ็กซีคิวส์โปรแกรม (เป็นโปรแกรมบนวินโดวส์ที่ทำงานได้โดยลำพัง) จากนั้นจึงแจกจ่ายให้กับผู้ใช้ ถ้าต่อมาคุณตัดสินใจปรับปรุงโปแกรมใหม่ คุณก็จะทำขบวนการเดิมซ้ำอีกโดยเริ่มต้นจาการวางแผนและวิเคราะห์เป้าหมายใหม่ ขั้นตอนต่างๆ เหล่านี้เป็นวงจรในการพัฒนาซอฟต์แวร์ทั้งหมด ดังแสดงในไดอะแกรมต่อไปนี้



2.3.5 การทดสอบโปรแกรม

การทดสอบโปรแกรมเป็นตรวจสอบโปรแกรมในสภาวะต่างๆ ว่าทำงานได้ถูกต้องหรือไม่ ปัญหาที่ทำให้โปรแกรมหยุดชะงักขณะที่กำลังใช้งานหรือสร้างผลลัพธ์รวมเรียกว่าความผิดพลาดของซอฟต์แวร์หรือบัก ถ้าคุณใช้ซอฟต์แวร์มาระยะหนึ่งแล้วเกิดปัญหาที่มีสาเหตุมาจากความผิดพลาดของซอฟต์แวร์ ตอนนี้คุณเป็นโปรแกรมเมอร์แล้วจึงเป็นงานของคุณที่จะทำให้ปัญหาเหล่านี้หมดไปก่อนจะถึงมือผู้ใช้ และเป็นโชคดีสำหรับเราทั้งหลายที่วิชวลเบสิกมีเครื่องมือดีบักที่ใช้ตรวจหาบักไว้ให้ด้วย

2.3.6 การคอมไพล์โปรแกรม

เมื่อคุณสร้างและทดสอบโปรแกรมเสร็จแล้ว คุณสามารถคอมไพล์โปรแกรมเป็นเอ็กซีคิวส์ไฟล์ที่ทำงานบนวินโดวส์ 95 หรือวินโดวส์ NT ได้ การสร้างเอ็กซีคิวส์ไฟล์ในวิชวลเบสิกเป็นเรื่องง่ายคุณเพียงแต่คลิกคำสั่ง Make EXE File ในเมนู File ถ้าคุณวางแผนที่จะรันแอปพลิเคชันวิชวลเบสิกตัวใหม่นี้ไว้ใช้เฉพาะบนระบบของคุณ การสร้างเอ็กซีคิวต์ไฟล์จะเป็นขั้นตอนสุดท้ายในการสร้างโปรแกรม เมื่อคุณต้องรันโปรแกรมใหม่นี้ให้ดับเบิลคลิกไฟล์นามสกุล EXE ใน Windows Explorer หรือสร้างไอคอนให้กับเอ็กซีคิวต์ไฟล์ไว้บนเดสก์ทอปของวินโดวส์แทนก็ได้

2.3.7 การแจกจ่าย โปรแกรม

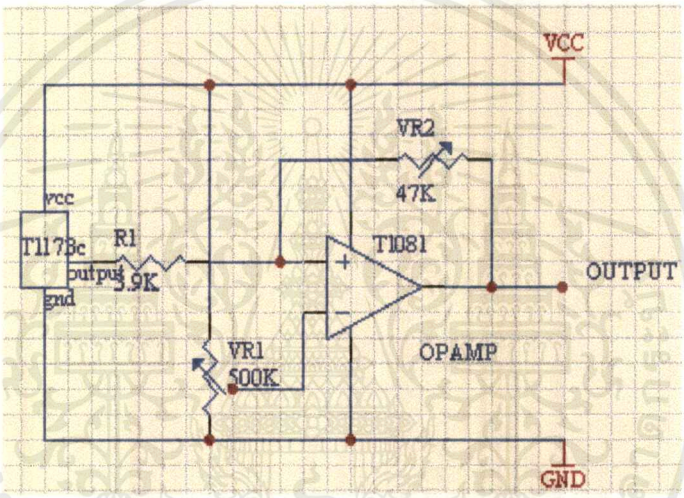
ถ้าต้องการนำโปรแกรมอันใหม่ไปให้กับเพื่อนๆ หรือวิทยาลัยต่างๆ หรือต้องขายโปรแกรม จำเป็นที่จะต้องใส่โปรแกรมไฟล์ที่จำเป็นบางไฟล์ลงในดิสก์ (หรือเน็ตเวิร์ก) ที่ผู้ใช้หรือลูกค้านำไปใช้ได้ การรันโปรแกรมที่สร้างจากวิชวลเบสิกจำเป็นต้องใช้ไฟล์ DLL ด้วย ดังนั้นคุณจึงต้องก๊อปปี้ไฟล์เหล่านี้จากฮาร์ดดิสก์มาในดิสก์ที่แจกจ่ายให้กับผู้ใช้ด้วย ไฟล์ DLL ที่จำเป็นต้องใช้จะขึ้นกับระบบ ปฏิบัติการที่ผู้ใช้ทำงานอยู่และลักษณะของวิชวลเบสิกที่ใช้สร้างโปรแกรม

บทที่3
การคำนวณและการสร้าง

3.1 ฮาร์ดแวร์

ส่วนของฮาร์ดแวร์จะประกอบด้วย สเต็ปปิงมอเตอร์ ชุดขับสเตปปิงมอเตอร์ ชุดวัดมุมเงย ชุดวัดมุมเอียง และชุดหาทิศเหนือ ซึ่ง ชุดขับสเต็ปปิงมอเตอร์ จะไม่นำมาอธิบายเพราะเป็นวงจรขับสเตปปิง มอเตอร์แบบทั่วๆ ไป จะอธิบายวงจรของชุดหาทิศเหนือและ ชุดวัดมุมเงยและมุมเอียง

3.1.1 ชุดหาทิศเหนือ

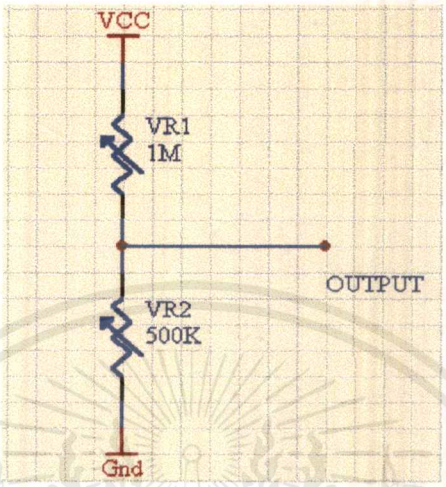


รูปที่ 3.1 รูปแสดงวงจรชุดหาทิศเหนือ

จากรูปที่ 3.1 จะเป็นวงจรที่ใช้ในการหาทิศเหนือ ใช้หลักการ ของ ลิเนียร ฮอลล์ เอฟเฟค (Linear Hall Effect) ที่มีแรงดันเอาท์พุทเปลี่ยนแปลงไปตามความหนาแน่นของสนามแม่เหล็กที่ตัดผ่าน

จากรูปวงจรประกอบด้วย ไอซี TL 173 CLP ลิเนียร ฮอลล์ เอฟเฟค สำหรับเปลี่ยนความหนาแน่นสนามแม่เหล็กเป็นสัญญาณไฟฟ้า ไอซี ออป แอมป์ TL 081 เป็นชุดปรับแรงดันให้อยู่ในช่วง 0 ถึง 5 โวลท์ โดยที่ 0 โวลท์จะเป็นทิศเหนือ และ 5 โวลท์จะเป็น ทิศใต้

3.1.2 ชุดวัดมุมเงยและชุดวัดมุมเอียง



รูปที่ 3.2 ชุดวัดมุมเงยและมุมเอียง

จากรูปที่ 3.2 เป็นวงจรที่ใช้วัด มุมเงยและ มุมเอียง ประกอบด้วย วีอาร์1 (VR1) และ วีอาร์ 2 (VR2) วีอาร์ 1 จะเป็นตัวปรับช่วงแรงดันเอาท์พุทในอยู่ในช่วง 0 ถึง 5 โวลท์ ซึ่งจะขึ้นอยู่กับวีอาร์ 2 ด้วย วีอาร์ 2 จะเป็นตัวที่ใช้วัดมุม สำหรับชุดวัดมุมเงยจะตั้งแรงดัน เอาท์พุท ที่ ศูนย์ องศา ให้เป็น ศูนย์ โวลท์ โดยการหมุน วีอาร์ 2 ให้มีค่า ศูนย์ โอห์ม และที่ 90 องศา ให้มีแรงดันเป็น 5 โวลท์ โดยการปรับที่ วีอาร์ 1 วีอาร์ที่ใช้นั้นเป็นแบบ บี (B TYPE) ซึ่งจะทำให้แรงดันเอาท์พุทที่ได้เป็นแบบ ลิเนียร

ในส่วนของชุดวัด มุมเอียงก็จะมีวงจรเหมือนกับชุดวัด มุมเงย แต่การติดตั้งจะแตกต่างกัน

3.2 โปรแกรมไมโครคอนโทรลเลอร์

การเขียนโปรแกรมเพื่อให้ไมโครคอนโทรลเลอร์ไปควบคุมการเคลื่อนที่ของจานรับสัญญาณดาวเทียมนั้น ต้องพิจารณาจุดมุ่งหมายของการควบคุม รูปแบบการควบคุม และวิธีการควบคุมโดยในส่วนของโครงงานนี้จะมีการใช้งาน 2 รูปแบบ คือ ควบคุมจาก ไมโครคอนโทรลเลอร์ และ ไมโครคอมพิวเตอร์ ดังนั้น โปรแกรมเริ่มต้นการทำงานของ ไมโครคอมพิวเตอร์ จะต้องแสดงผลที่ตัวแสดงผลว่า ตอนนั้นต้องการรับคำสั่งของผู้ใช้จากคีย์บอร์ด เมื่อผู้ใช้กดหมายเลขโปรแกรมใน ไมโครคอนโทรลเลอร์ ก็จะประมวลผลและไปทำงานตามขั้นตอนของโปรแกรม จากรูปที่ 3.3

โดยเริ่มจาก แสดงชื่อโครงงานในตอนเปิดเครื่อง ต่อจากนั้น จะถามว่าจะควบคุมด้วยไมโครคอมพิวเตอร์หรือไม่ ถ้าใช่จอแสดงผลก็จะปรากฏว่า ควบคุมด้วยคอมพิวเตอร์ และ โปรดรอการติดต่อกับคอมพิวเตอร์ ซึ่งจะรอการติดต่อกับไมโครคอมพิวเตอร์ จนกว่าจะมีสัญญาณตอบรับจากคอมพิวเตอร์ แล้วการควบคุมทั้งหมด จะถูกควบคุมโดยไมโครคอมพิวเตอร์ทั้งหมด และถ้าตอบว่าไม่ใช่ โปรแกรมก็จะเป็นการควบคุมด้วยไมโครคอนโทรลเลอร์แทน

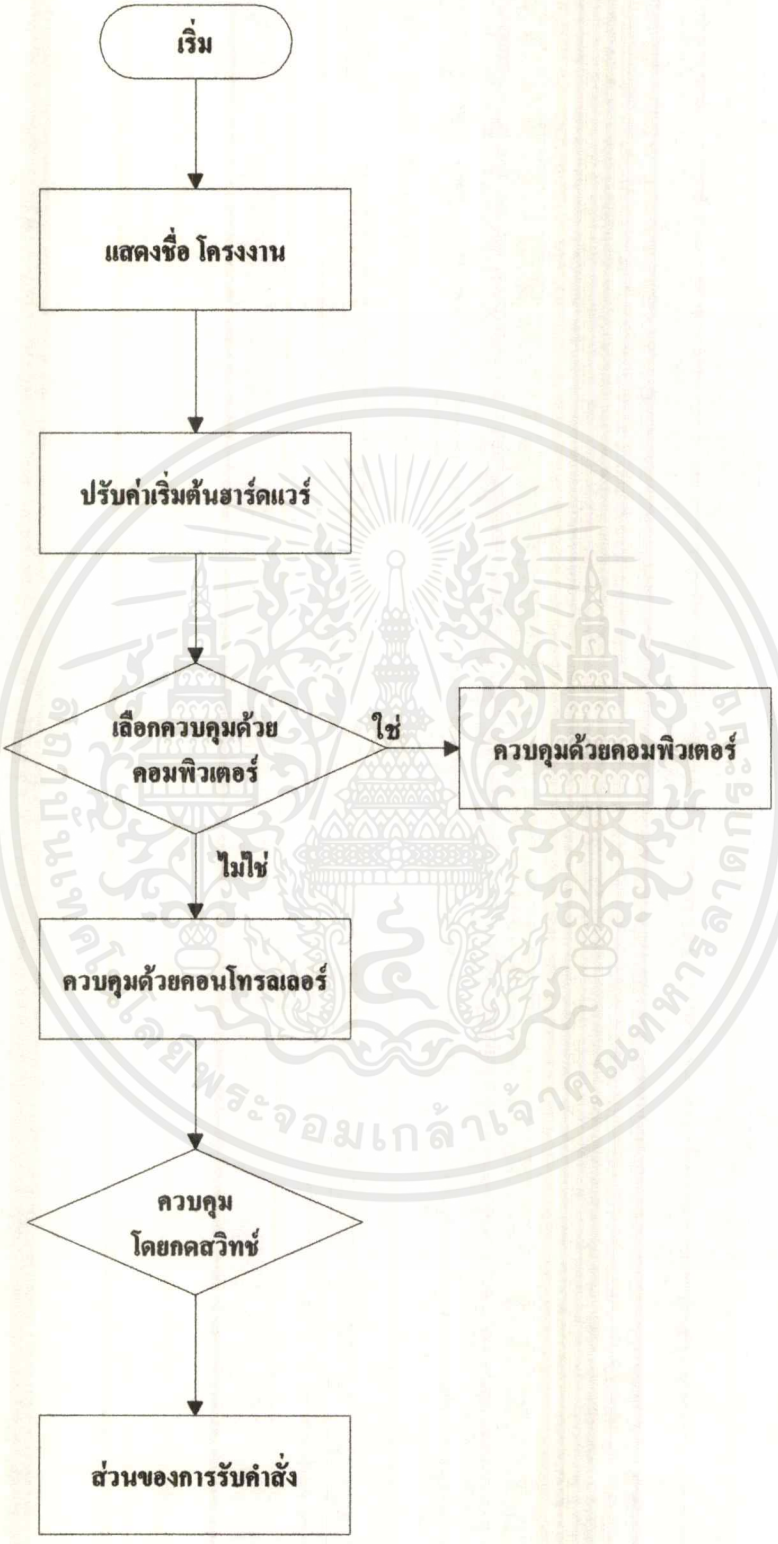
หลังจากที่เลือกการควบคุมด้วยไมโครคอนโทรลเลอร์แล้ว โปรแกรมจะถามอีกว่า เป็นแบบ อัตโนมัติ หรือไม่ ถ้าใช่ ก็จะเข้าสู่การทำงานแบบอัตโนมัติ ซึ่งยังไม่มีการใช้งาน เพราะยังไม่สามารถจัดหา เครื่องบอกพิกัดและ เข็มทิศอิเล็คทรอนิกส์ได้ และถ้าตอบว่าไม่ใช่ โปรแกรมก็จะถามต่อว่าเป็นการควบคุมแบบ กึ่งอัตโนมัติ ใช่หรือไม่ ถ้าใช่ก็จะเข้าสู่การทำงานแบบกึ่งอัตโนมัติ ก็จะให้ใส่ค่าของ เส้นรุ้งเส้นแวงของตำแหน่งที่อยู่ปัจจุบัน และชื่อของดาวเทียม ส่วนที่เหลือ โปรแกรมจะทำงานต่อ และถ้าตอบว่าไม่ใช่ ก็จะเป็นการควบคุมเอง ซึ่งผู้ใช้จะต้องทำการคำนวณค่ามุมเงย และ มุม อะซีมูธ เอง แล้วใส่ค่าที่ได้เพื่อไปปรับจานสายอากาศอีกทีหนึ่ง

3.2 ขั้นตอนการออกแบบการเขียนโปรแกรมไมโครคอนโทรลเลอร์

ในส่วนของการโปรแกรมไมโครคอนโทรลเลอร์สามารถแบ่งออกเป็น 3 ส่วนด้วยกันคือ ส่วนของเมนูหลัก ส่วนของการควบคุมแบบปรับเอง และส่วนของการควบคุมโดยใช้ คอมพิวเตอร์ หรือเรียกว่าการทำงานในรีโมตโหมด

3.2.1 การออกแบบเมนูหลัก

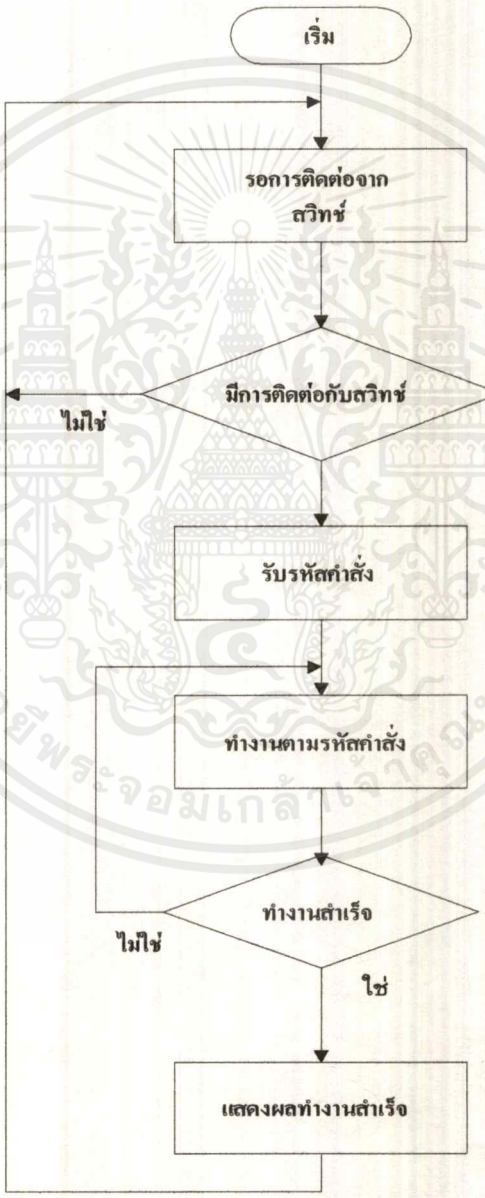
ในส่วนของเมนูหลักจะมีการออกแบบไว้โดยเมื่อเริ่มทำการเปิดเครื่อง โปรแกรมมอนิเตอร์ของไมโครคอนโทรลเลอร์จะทำการแสดงชื่อของโครงงานและจะทำการเข้าสู่โหมดปรับค่าเริ่มต้นของฮาร์ดแวร์ เมื่อทำการปรับค่าเริ่มต้นที่ถูกต้องแล้วจะเข้าไปยังส่วนของการเลือกโหมดการทำงาน โดยจะมีให้เลือก 2 โหมด ลำดับขั้นการทำงานของเมนูหลักแสดงไว้จากรูปที่ 3.4



รูปที่ 3.4 ลำดับขั้นการทำงานของเมนูหลัก

3.2.2 การออกแบบ การควบคุมแบบปรับเอง

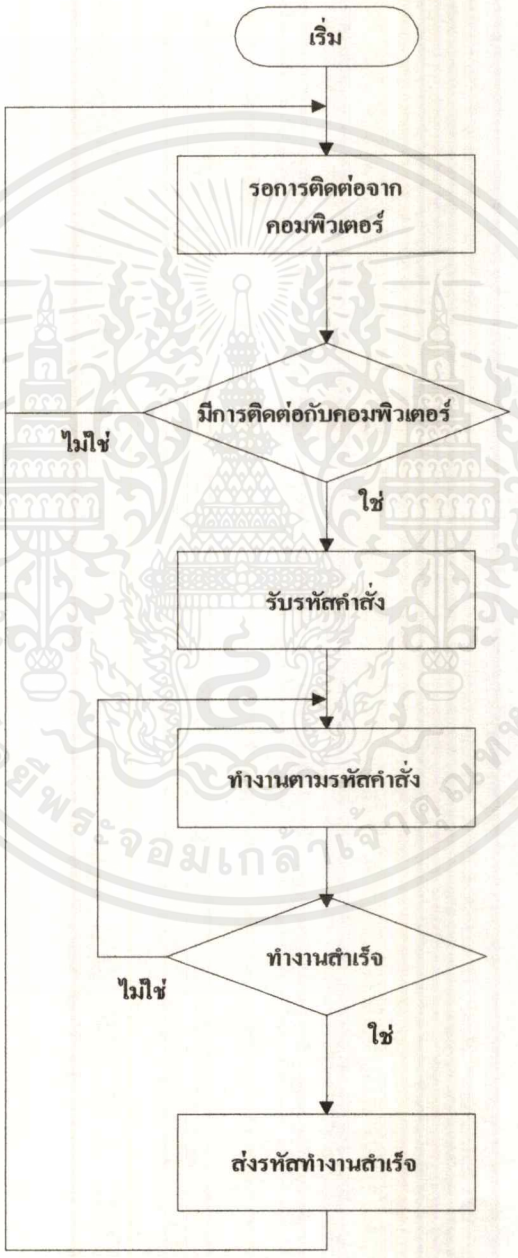
ในส่วนของการควบคุมแบบปรับเองนี้จะสามารถปรับทิศทางการเคลื่อนที่ของชุดควบคุมได้ 4 ทางคือ ปรับมุมเงยขึ้น ปรับมุมเงยลง ปรับมุมกวาดไปทางขวา และ ปรับมุมกวาดไปทางซ้าย เมื่อมีการกดสวิทช์ ไมโครคอนโทรลเลอร์จะทำการประมวลผล แสดงค่าตำแหน่งปัจจุบันของมอเตอร์ และควบคุมทิศทางการเคลื่อนที่ไปพร้อมกันลำดับขั้นการทำงานแสดงจากรูปที่ 3.5



รูปที่ 3.5 ลำดับขั้นการทำงานของ การควบคุมแบบปรับเอง

3.2.3 การออกแบบการควบคุมโดย คอมพิวเตอร์

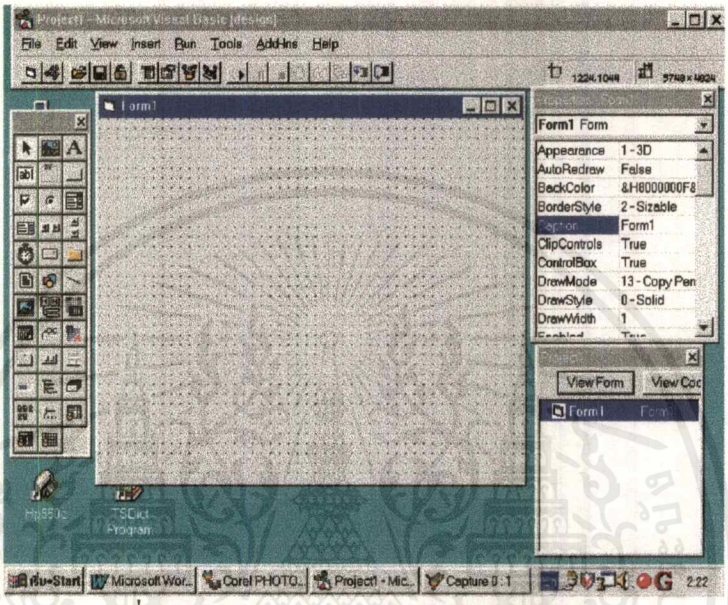
ในส่วนของการควบคุมโดย คอมพิวเตอร์ๆจะมีการควบคุมโดยการส่งรหัสคำสั่งมาที่ไมโครคอนโทรลเลอร์ และทำการประมวลผลตามนหัสคำสั่งที่ตั้งไว้ ลำดับขั้นการทำงานแสดงจากรูปที่ 3. 6



รูปที่ 3.6 ลำดับขั้นการทำงานของการควบคุมโดยใช้คอมพิวเตอร์

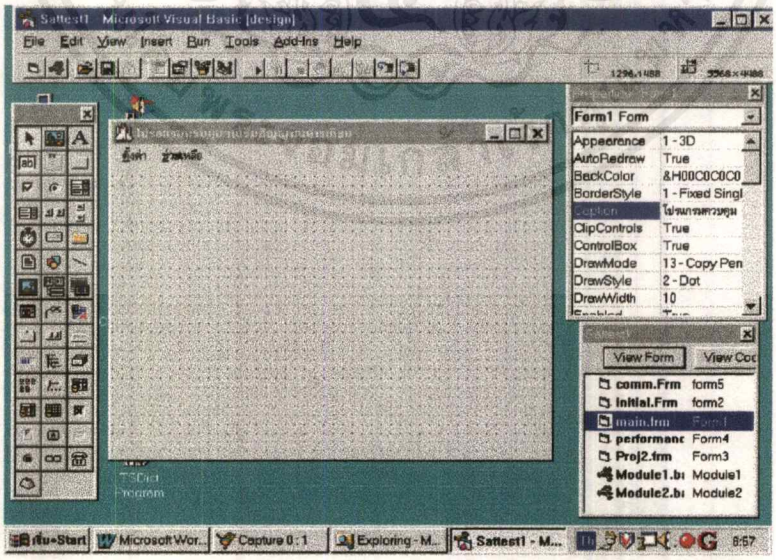
3.3 ขั้นตอนการเขียนโปรแกรมโดยใช้โปรแกรม VISUAL BASIC

คลิกไอคอนโปรแกรม VISUAL BASIC 4.0 สภาพแวดล้อมในการเขียนโปรแกรมของวิชวลเบสิกจะปรากฏขึ้นจากรูปที่ 3.7

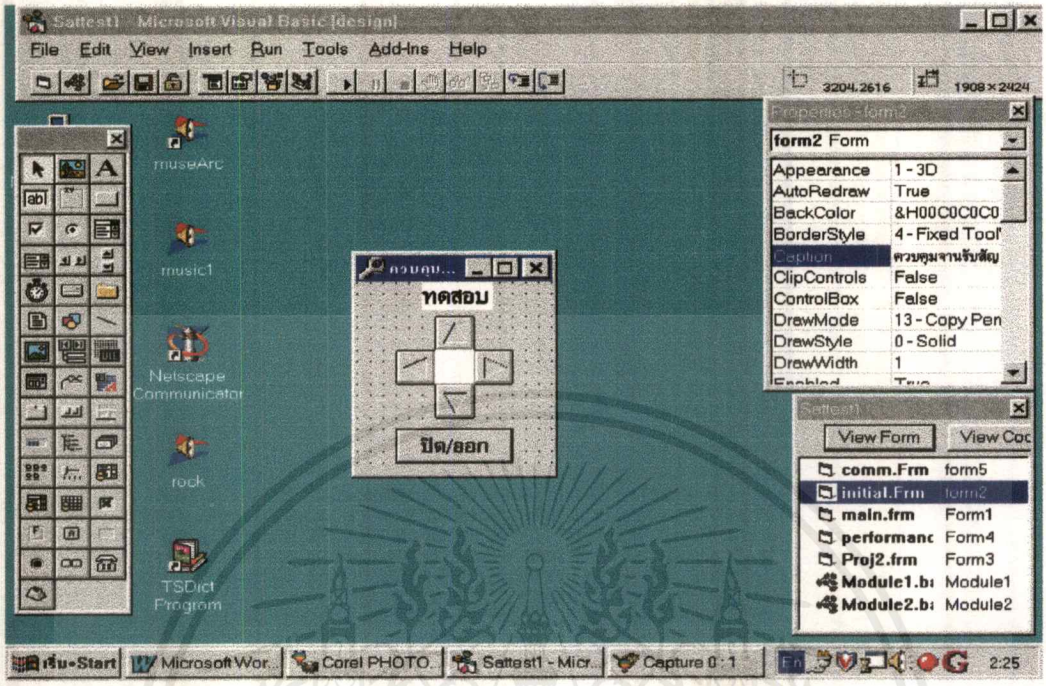


รูปที่ 3.7 โปรแกรม VLSUAL BASIC 4.0

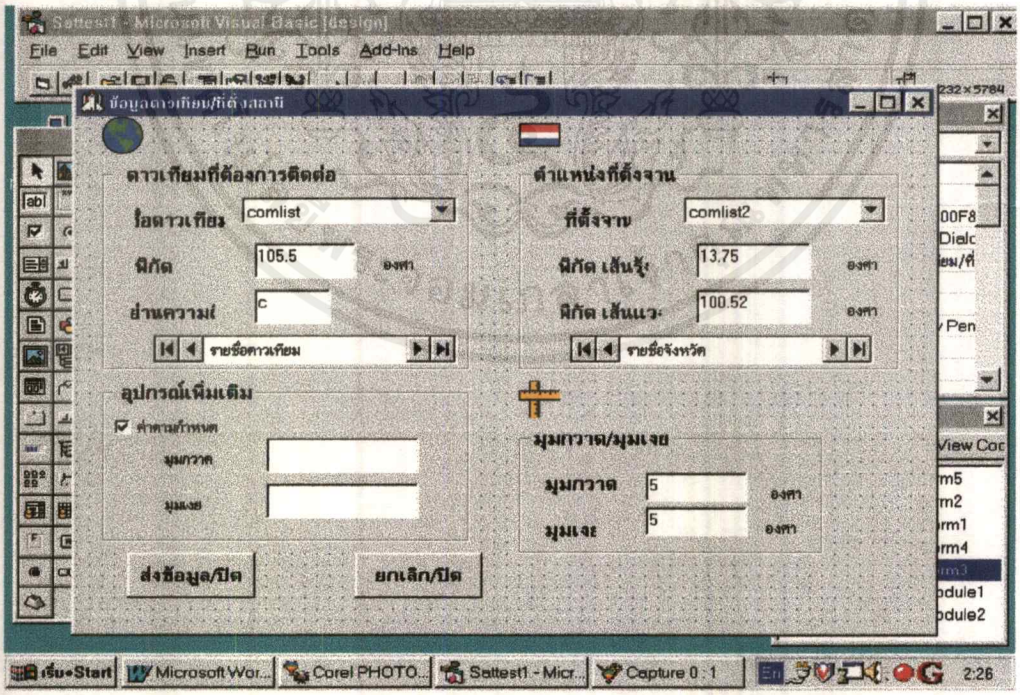
ออกแบบหน้าจอที่ต้องการ พร้อมกับเขียนโปรแกรมตามที่ต้องการ (โปรแกรมแสดงที่ภาคผนวก)



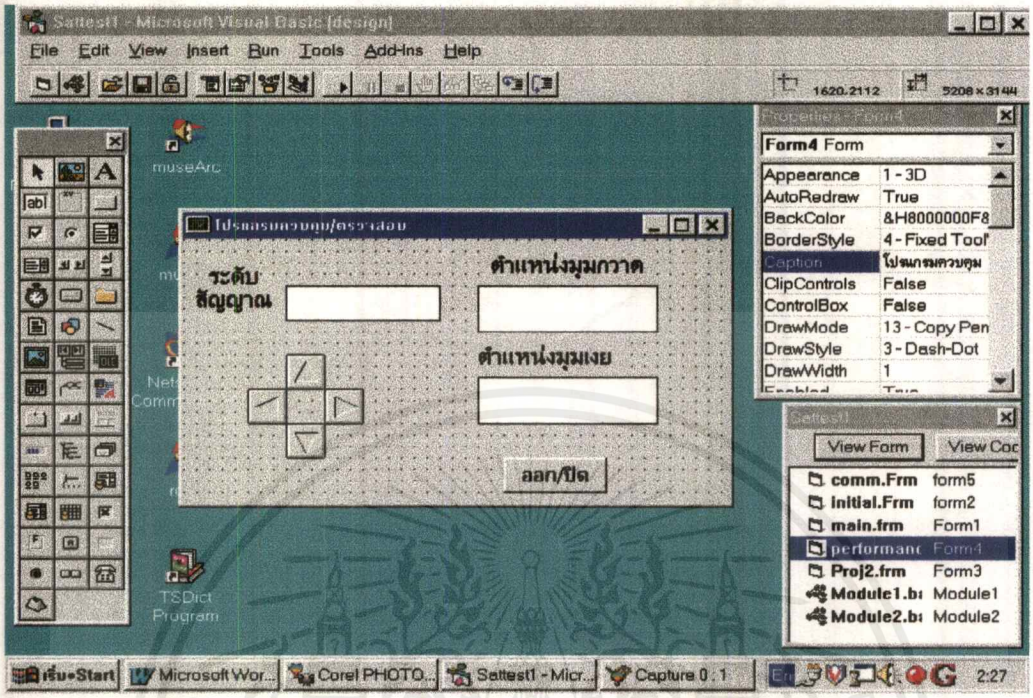
รูปที่ 3.8 การออกแบบหน้าจอหลักของโปรแกรม



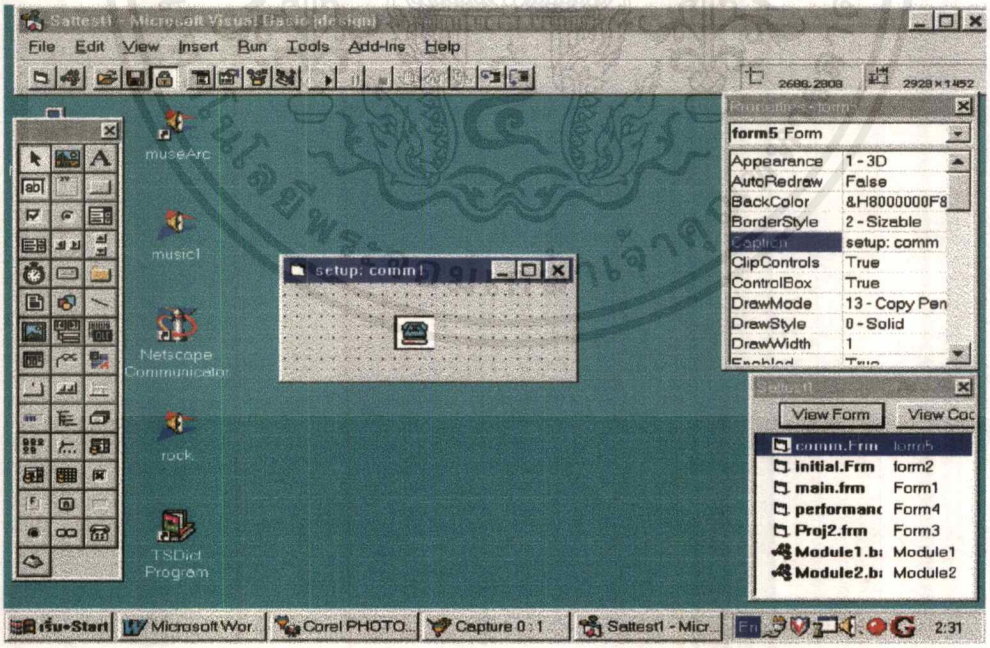
รูปที่ 3.9 ออกแบบหน้าจอสำหรับตรวจสอบอุปกรณ์ภายนอก



รูปที่ 3.10 ออกแบบหน้าจอสำหรับการเลือกสถานที่ตั้งจานและดาวเทียมที่ต้องการติดต่อ

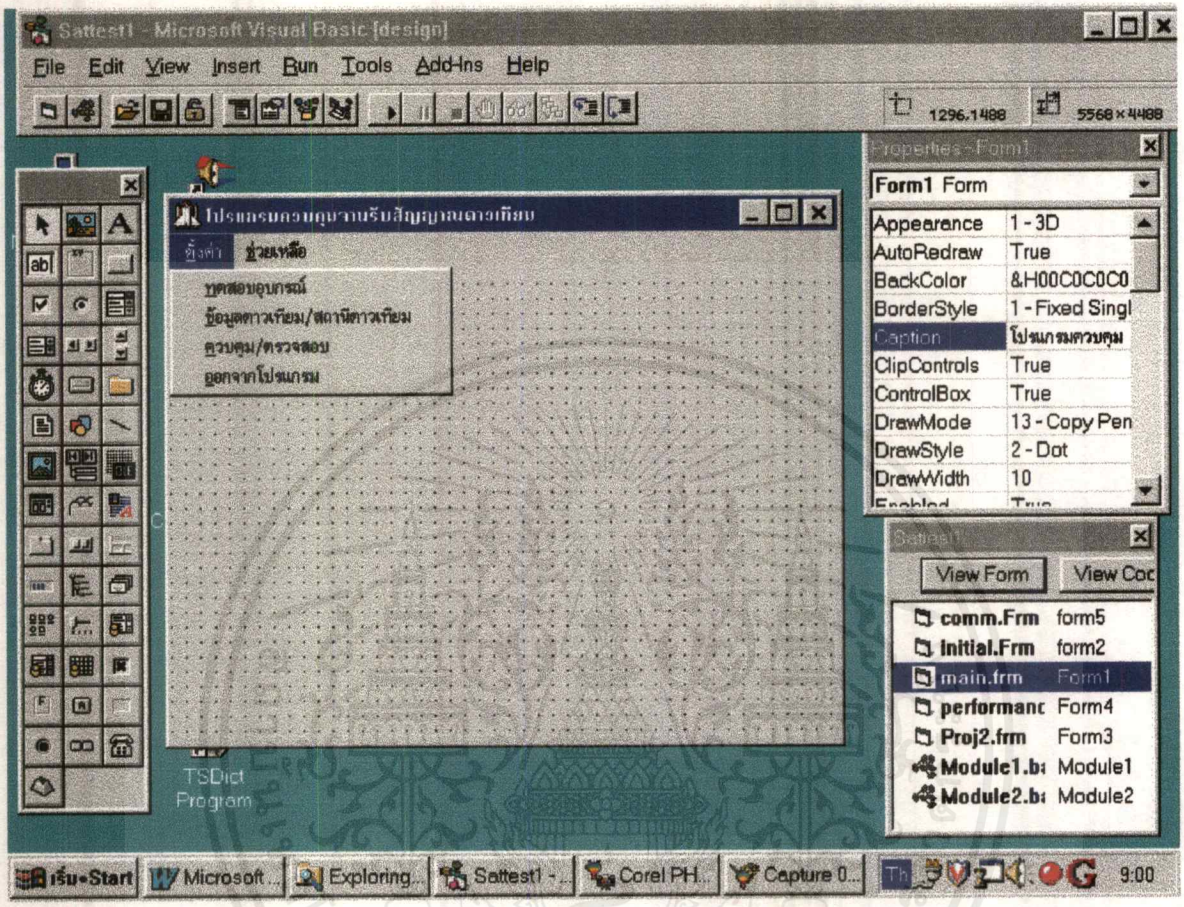


รูปที่ 3.11 ออกแบบหน้าจอขณะทำงาน



รูปที่ 3.12 โปรแกรมสำหรับการติดต่อกับอุปกรณ์ภายนอก

นำหน้าจอที่ออกแบบทั้งหมดมาไว้ในโปรแกรมหลัก ซึ่งเป็นวิธีที่ใช้ในการเขียนโปรแกรมที่ทำงานบน WINDOWS และ ทดสอบโปรแกรมที่สมบูรณ์แล้ว



รูปที่ 3.13 หน้าจอที่สมบูรณ์แล้ว

บทที่ 4
ผลการทดลอง

4.1 ผลจากการใช้ ไมโครคอนโทรลเลอร์ควบคุม

การทดลองและบันทึกผลการทดลองในส่วนของการควบคุมการเคลื่อนที่ ของจานรับสัญญาณดาวเทียมได้แบ่งการทดลองออกเป็นส่วนเพื่อง่ายแก่การเข้าใจดังนี้

4.1.1 การปรับค่าเริ่มต้นของฮาร์ดแวร์

เมื่อเริ่มการใช้งานเครื่องทุกครั้งจะต้องทำการปรับค่าเริ่มต้นของฮาร์ดแวร์ เพื่อนำไปอ้างอิงการคำนวนหาทิศทางดาวเทียม

4.1.1.1การปรับทิศ

ในการคำนวนมุมของจานรับสัญญาณดาวเทียมมีความจำเป็นต้องหาทิศ อ้างอิงซึ่งส่วนใหญ่นิยมใช้ทิศเหนือ จากรูปที่ 4. 1และ 4.2 แสดงการปรับทิศของโครงงาน โดยโครงงานจะทำการหมุนปรับให้จานดาวเทียมชี้ไปยังทิศเหนือ เพื่อใช้เป็น ทิศอ้างอิง โดยการอ้างอิงนี้จะใช้เข็มทิศอิเล็กทรอนิกส์มาเป็นตัวตรวจจับ

รูปที่ 4.1 หน้าจอการปรับให้จานรับสัญญาณอยู่ในทิศทางที่กำหนด

รูปที่ 4.2 แสดงการหมุนหาทิศเหนือของจานรับสัญญาณดาวเทียม

4.1.1.2ปรับมุมเงย

ในการคำนวนมีความจำเป็นต้องใช้การอ้างอิงมุมเงย เช่นเดียวกับการ ทิศทาง จากรูปที่ 4.3 แสดงการปรับมุมเงยของจานดาวเทียมโดยจะทำการปรับ มุมเงยเริ่มต้นของจารรับสัญญาณดาวเทียมให้อยู่ที่ 0 องศา

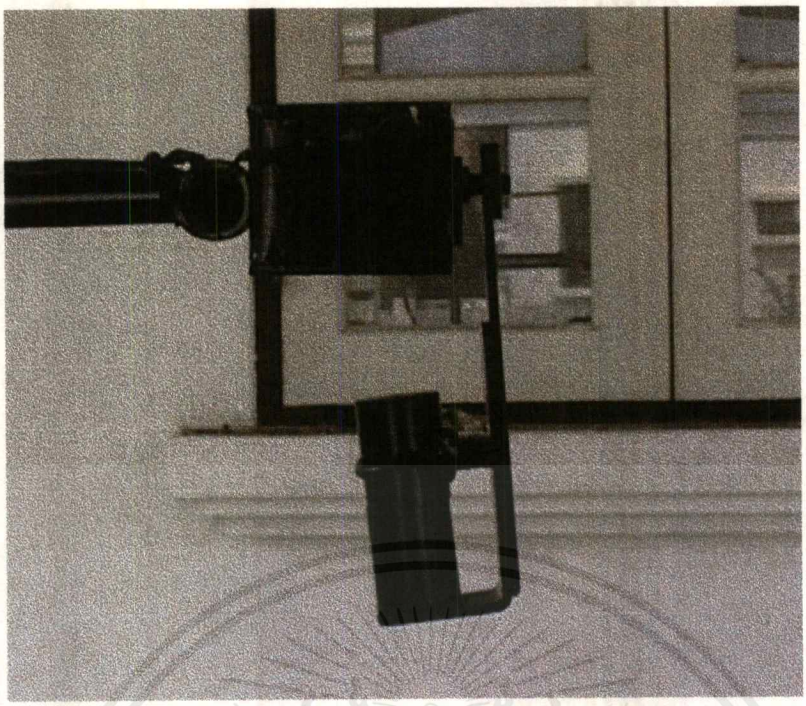
รูปที่ 4.3 หน้าจอการปรับมุมเงยอัตโนมัติ

รูปที่ 4.4 แสดงการปรับมุมเงยเป็น 0 องศา

4.1.1.3ปรับมุมเอียง

มุมเอียงก็มีผลต่อความคลาดเคลื่อนของมุมที่คำนวนออกมาได้ดังนั้นจึงต้องมีการปรับแต่ง มุมเอียงให้มีค่าเป็นศูนย์หรือให้ได้ระนาบมากที่สุดจากรูปที่ 4.6 แสดงอุปกรณ์ตรวจจับมุมเอียง

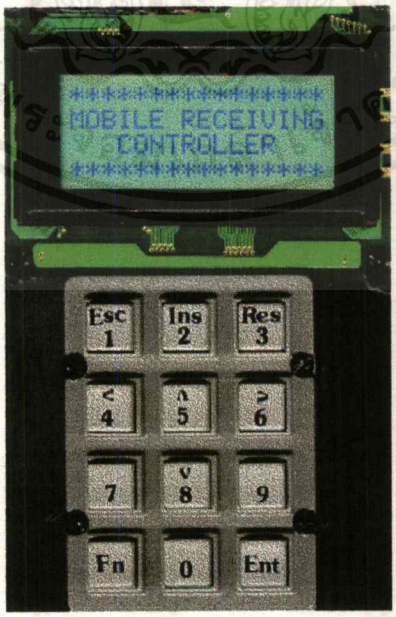
รูปที่ 4.5 หน้าจอการปรับ มุมเอียงในโหมดควบคุมเอง



จากรูปที่ 4.6 แสดงอุปกรณ์วัด มุมเอียงของจาน

4.1.2 โหมดโลคัล

หลังจากผ่านการปรับเครื่องแล้ว ก็จะเลือกการทำงานของโครงงาน ว่าจะเลือกการทำงานของโครงงานว่าจะเป็นแบบรีโมตซึ่งควบคุมโดยคอมพิวเตอร์ หรือแบบโลคัลซึ่งควบคุมโดยไมโครคอนโทรลเลอร์ ในโหมด โลคัล จะเป็นการควบคุมแบบ ควบคุมเอง โดยจะมีการปรับ หมุน ซ้าย-ขวา ขึ้นบน-ลงล่าง ของจานรับสัญญาณดาวเทียม ค่ามุมกวาดและ มุมเงย สามารถดูได้จากจอ แสดงผล



(ก)

(ข)

รูปที่ 4.7(ก) และ (ข) หน้าจอแสดงชื่อโครงงาน

การแสดงผลนี้เป็นการแสดงชื่อของโครงงาน โดยครั้งแรกเมื่อเริ่มทำการเปิดเครื่องโปรแกรมเริ่มการทำงาน (PEOGRAM MONITOR) ในหน่วยความจำ ก็จะทำงานตามลำดับทำให้ไมโครคอนโทรลเลอร์ส่งสัญญาณไปให้ตัวแสดงผลทำการแสดงผลออกมา จากรูปที่ 4.8 การแสดงให้มีการเลือกโหมดใช้งาน

รูปที่ 4.8 หน้าจอการเลือกโหมด โลคัล และ โหมดรีโมต

เมื่อทำการเลือก โลคัลโหมดแล้วจะมีการแสดงผลให้กดสวิทช์จากรูปที่ 4.8 แล้วลำดับต่อไปเรา สามารถควบคุมการปรับมุมของจานรับสัญญาณดาวเทียมได้ทันทีดังแสดงจากรูปที่ 4.9

รูปที่ 4.9 การแสดงผลพร้อมที่จะรับการกดสวทช์

4.1.3 โหมดรีโมต

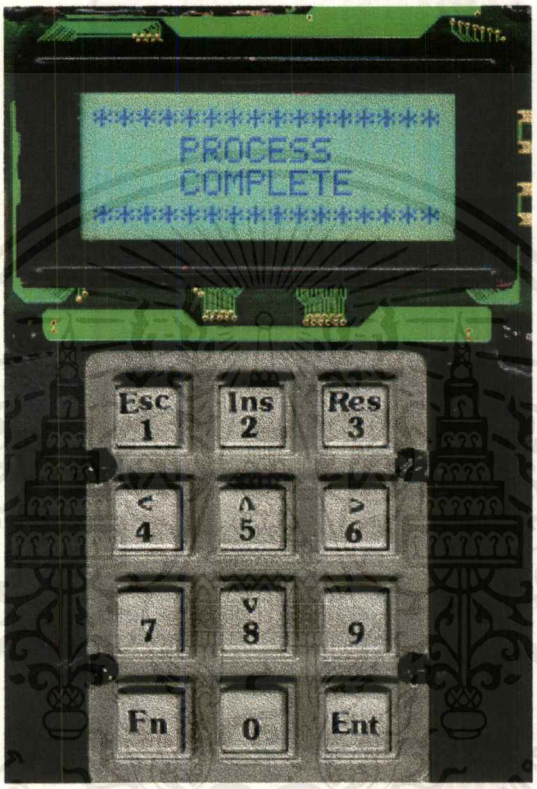
ในโหมดนี้จะทำการติดต่อกับคอมพิวเตอร์โดยการคำนวนมุมต่างๆจะคำนวนโดยคอมพิวเตอร์เมื่อคำนวนเสร็จ คอมพิวเตอร์จะทำการส่งรหัสสัญญาณควบคุมมาที่ ไมโครคอนโทรลเลอร์เพื่อให้ปรับเปลียนตำแหน่งของจานรับสัญญาณดาวเทียมไปยังทิศที่ต้องการการแสดงผลและการควบคุมจากรูปที่ 4.10 และ 4.11

รูปที่ 4.10 หน้าจอการติดต่อกับคอมพิวเตอร์ และแสดงค่า มุมกวาด มุมเงย

รูปที่ 4.11 หน้าจอการปรับมุมเงย

4.1.3 การตรวจสอบความผิดพลาด

ในส่วนของการรับส่งข้อมูลจะมีการเข้ารหัสข้อ เพื่อตรวจสอบความผิดพลาดของการส่งรหัสคำสั่งถ้าหากคอมพิวเตอร์ส่งรหัสที่ผิดหรือ รหัสที่ ไมโครคอนโทรลเลอร์ไม่สามารถถอดรหัสออกมาได้ก็จะทำการแจ้งไปยังคอมพิวเตอร์และหากไมโครคอนโทรลเลอร์ทำงานสำเร็จก็จะแจ้งไปยังคอมพิวเตอร์เหมือนกันจากรูปที่ 4.11จะแสดงขอ้ความเมื่อมีการรับส่งขอ้มูลมีปัญหา และจากรูปที่ 4.10เป็นการแสดงผลเมื่อการทำงานสำเร็จ และหากจานเคลื่อนที่ไปยังตำแหน่งที่กำหนดไว้ก็จะมีข้อความแสดงจากรูปที่ 4.12



รูปที่ 4.10 หน้าจอแสดงการทำงาน สมบูรณ์แล้ว

รูปที่ 4.12 หน้าจอการทำงานผิดพลาด

รูปที่ 4.13 หน้าจอการปรับตำแหน่งเกิน ที่กำหนด

4.1.4 โหมดติดตั้งอุปกรณ์และทดสอบ (DIAGNOSTIC and TEST)

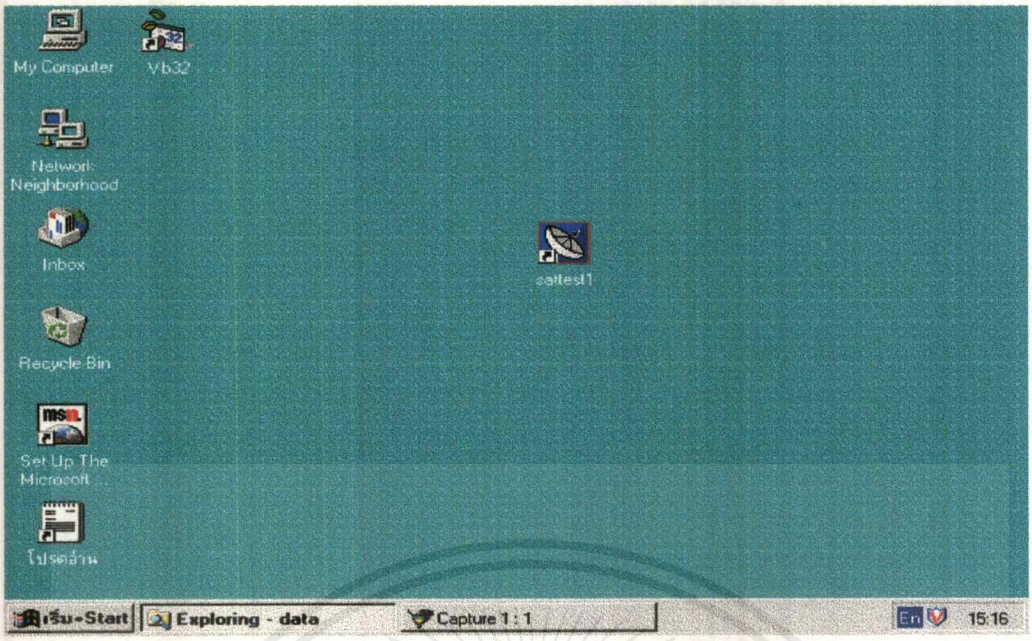
ในโหมดนี้จะเป็นการเซ็ตค่าให้กับโปรแกรมว่าจะทำการปรับเครื่อง แบบอัตโนมัติ หรือ แบบควบคุมเอง และเป็นโปรแกรมช่วยในการตรวจสอบการติดต่อกับคอมพิวเตอร์

รูปที่ 4.14 หน้าจอแสดงตัวเลือกในโหมดติดตั้งอุปกรณ์และทดสอบ

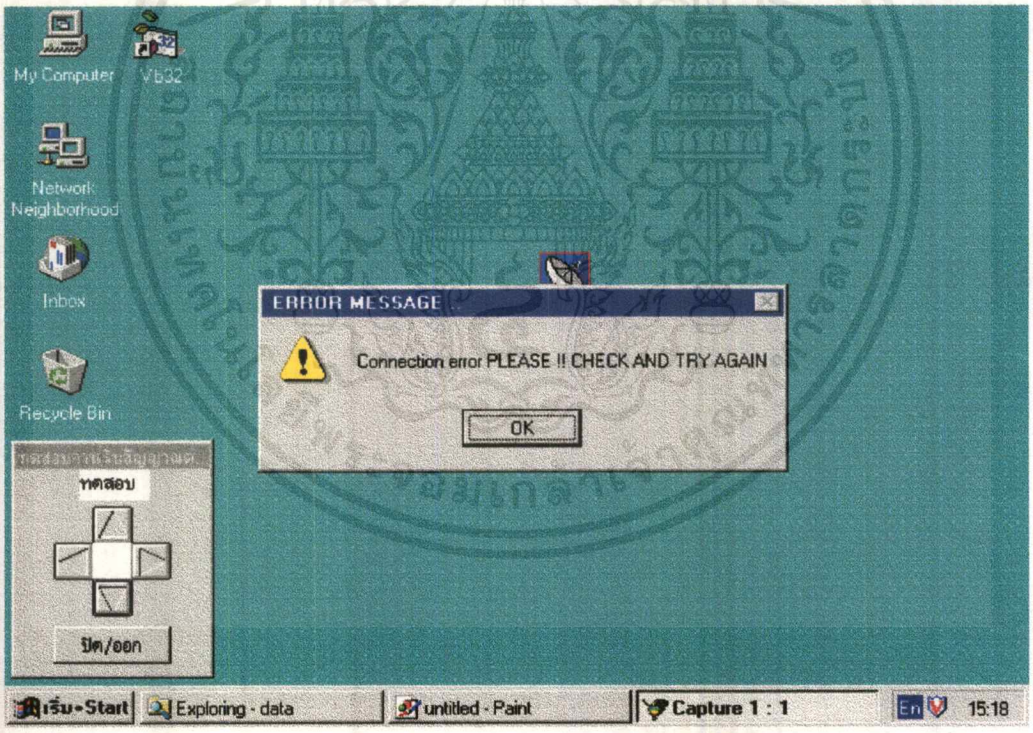
โหมดนี้ สำหรับการควบคุมด้วยคอมพิวเตอร์ ซึ่งคอมพิวเตอร์ จะส่งค่าต่างๆ มายังไมโครคอนโทรลเลอร์ เพื่อใช้ในการควบคุมจานรับสัญญาณดาวเทียม ต่อคอมพิวเตอร์เข้ากับ ชุดควบคุมและเปลี่ยนโหมดการทำงานของชุดควบคุมจาก โลคัล โหมด ไปเป็น ริโมท โหมด จากรูปที่ 4.15 แสดงให้เห็นถึงการติดตั้ง คอมพิวเตอร์กับชุดควบคุม

รูปที่ 4.15 แสดงการต่อสายต่าง ๆ

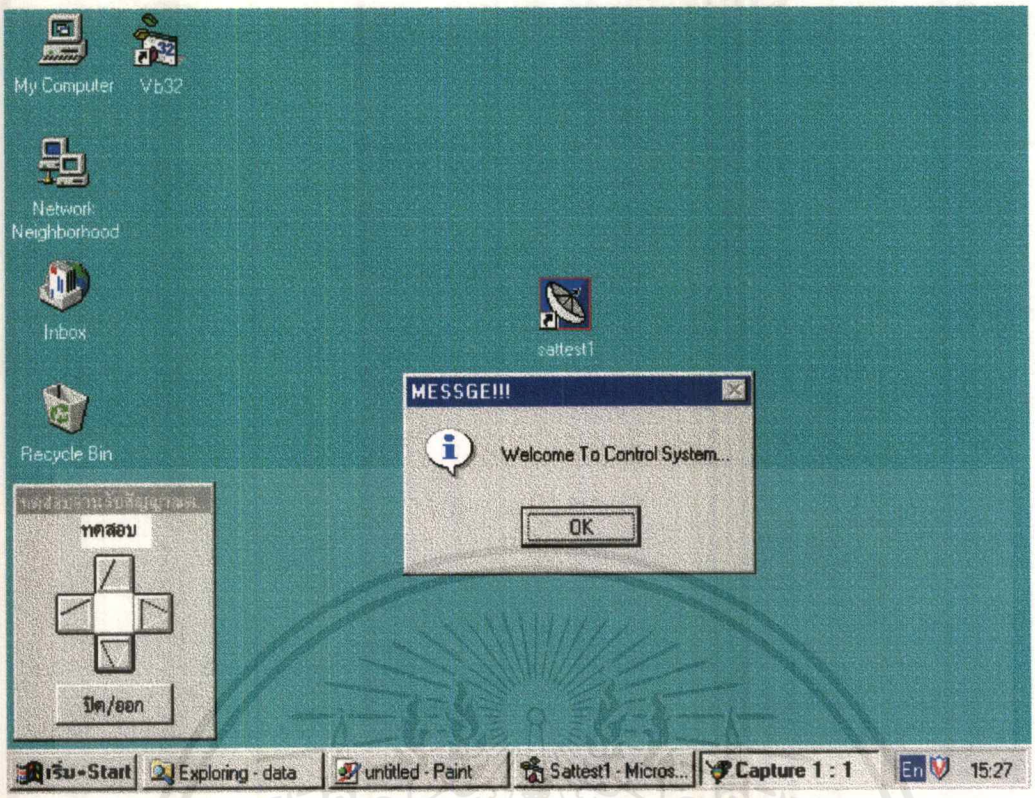
เมื่อทำการต่อสายเรียบร้อยแล้วก็เปิดเครื่องคอมพิวเตอร์ เข้าสู่โปรแกรม วินโดส์ จากรูปที่ 4.16 แสดงภาพของหน้าจอของโปรแกรมวินโดส์ เมื่อเข้าสู่ดปรแกรม วินโดสีเรียบร้อยแล้ว ท่านจะพบกับรูปของจานรับสัณฉาญดาวเทียมอยู่ที่กลางจอภาพ ใช้เมาส์กด 2 ครั้งติดๆกันเพื่อเข้าสู่โปรแกรมควบคุม หลังจากที่กดเมาส์2ครั้งติดๆกันจะปรากฏภาพดังรูปที่ 4.17 และรูปที่4.18 จะปรากฏถ้าการติดตั้งถูกต้อง



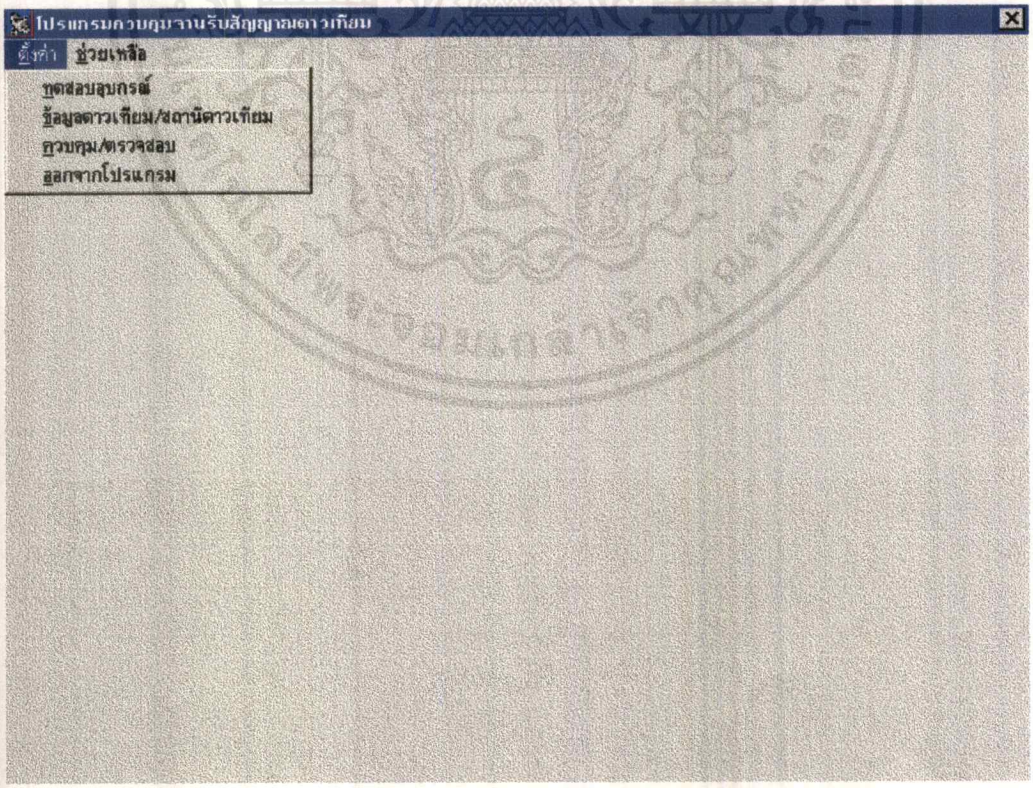
รูปที่ 4.16 หน้าจอของโปรแกรมวินโดว์



รูปที่ 4.17 แสดงข้อความขณะติดต่อ

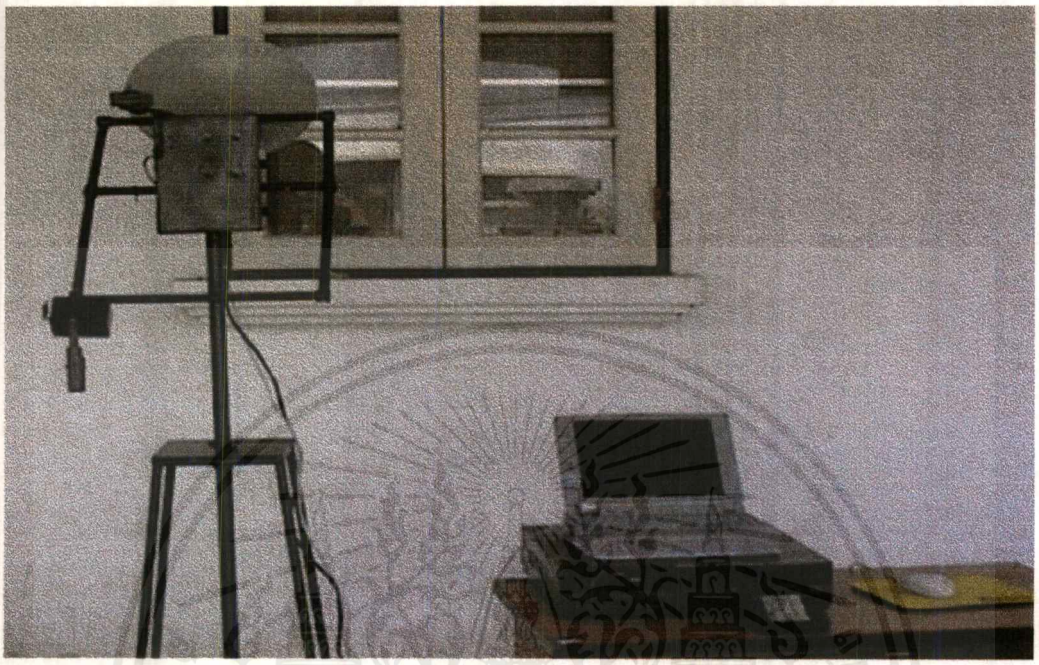


รูปที่ 4.18 แสดงข้อความเมื่อทำการติดต่อได้แล้ว

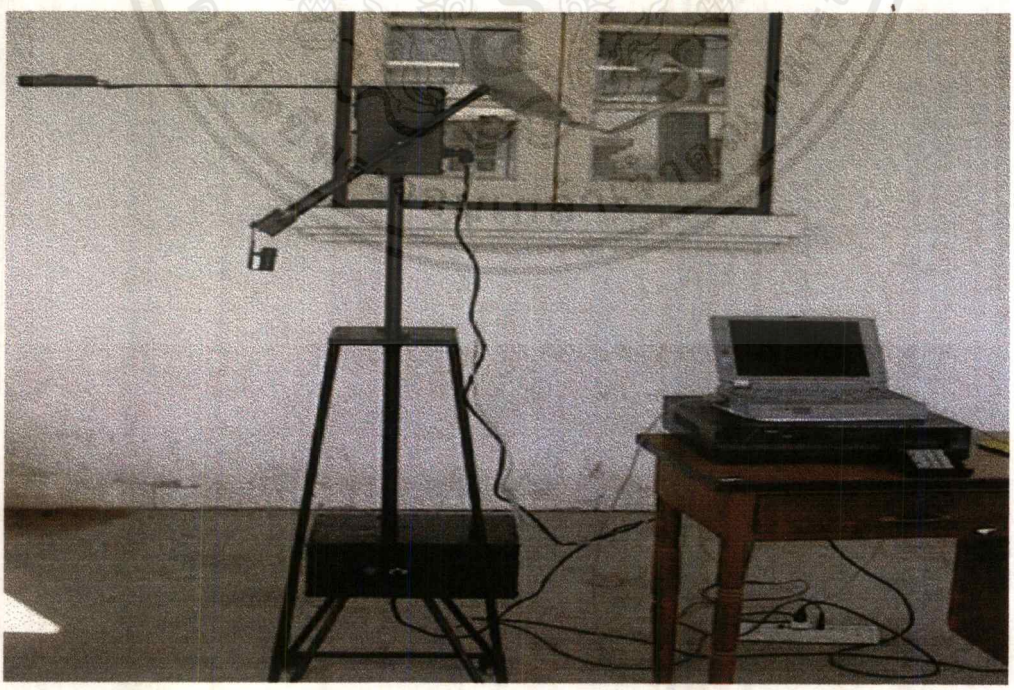


รูปที่ 4.19 แสดงการเลือกค่าในเมนู

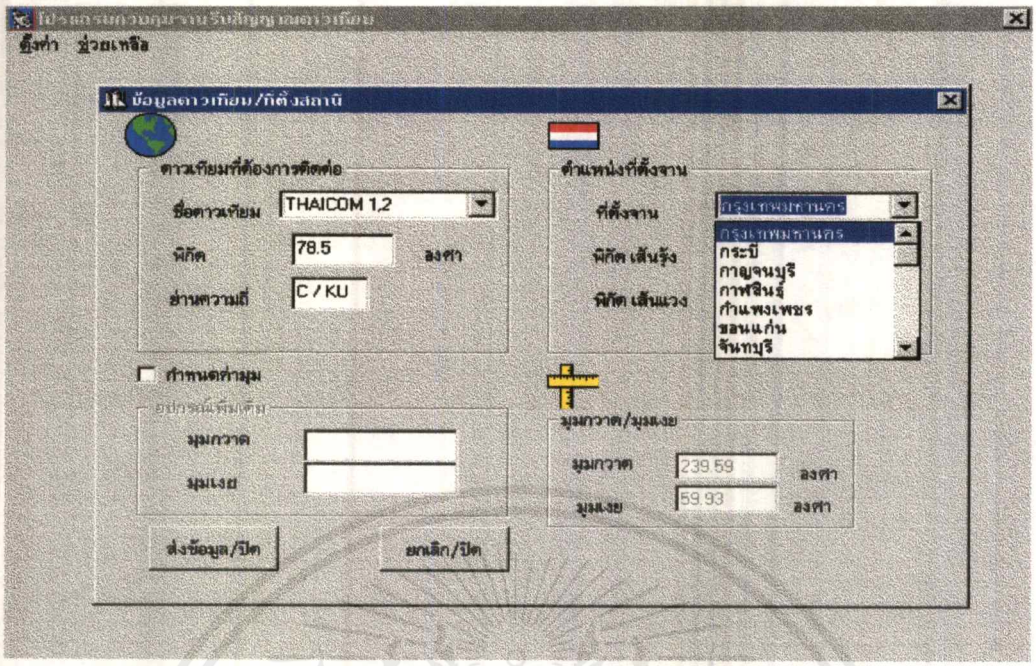
จากรูปที่ 4.19 เป็นรูปของโปรแกรมควบคุม หลังจากนั้นทำการเลือกค่าต่างตามขั้นตอน ในการทดลองนี้ จะทำการควบคุมจานรับสัญญาณดาวเทียมให้เปลี่ยนตำแหน่ง คือ ให้เคลื่อนที่ไปทางขวา และเงยขึ้น จากรูปที่4.20และรูปที่ 4.21 แสดงผลการทดลอง



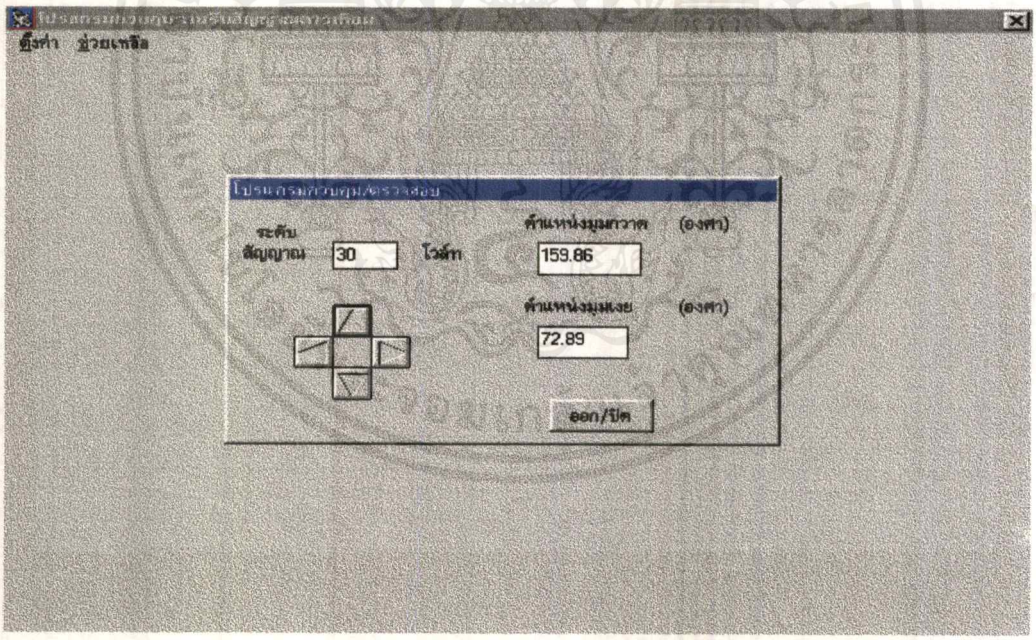
รูปที่ 4.20 แสดงให้เห็นการทำงาน



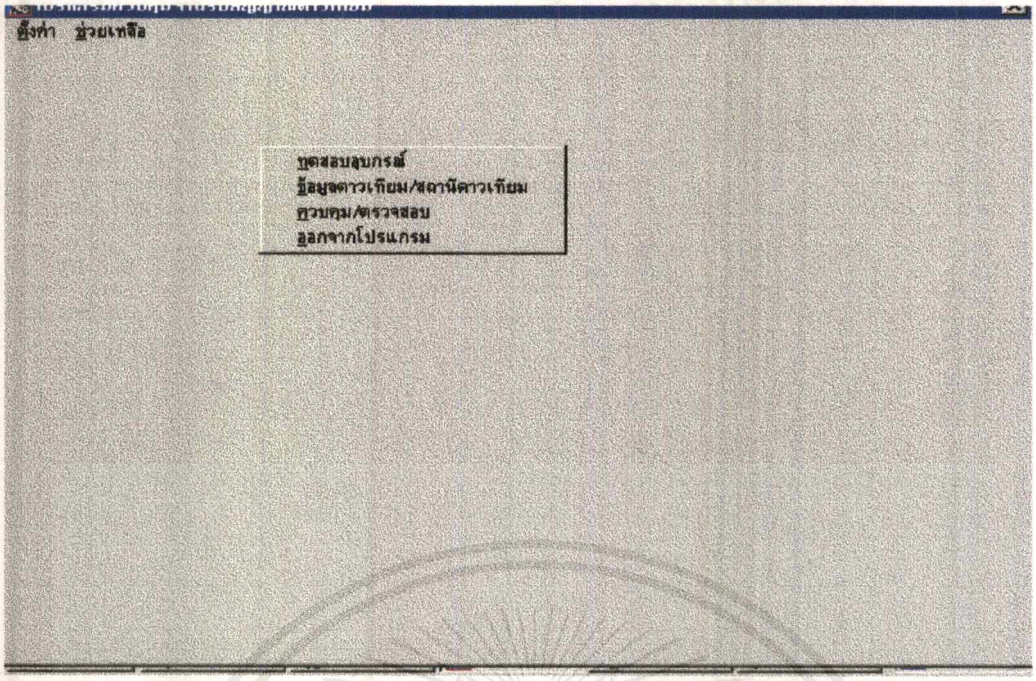
รูปที่4.21 แสดงการเปลี่ยนตำแหน่ง



รูปที่ 4.22 แสดงการเลือกดาวเทียม/สถานที่ตั้งสถานี



รูปที่ 4.23 แสดงตำแหน่งปัจจุบันของจานรับสัญญาณดาวเทียมและระดับของสัญญาณ



รูปที่ 4.24 แสดงการออกจากโปรแกรม

บทที่ 5
บทวิจารณ์และบทสรุป

จากเป้าหมายของโครงงานนี้คือเราสามารถที่จะติดตั้งจานรับสัญญาณดาวเทียมได้โดยง่าย สะดวกและรวดเร็ว เพียงแค่ผู้ที่ทำการติดตั้งชุดจานรับสัญญาณดาวเทียมนี้ไว้ในตำแหน่งที่เหมาะสมแล้ว ทำการเลือกชื่อดาวเทียมดวงที่ต้องการจะรับสัญญาณ เมื่อเลือกแล้วชุดควบคุมจานรับสัญญาณดาวเทียมก็จะทำการคำนวณและควบคุมให้จานรับสัญญาณดาวเทียมเคลื่อนที่ไปยังตำแหน่งที่ถูกต้องได้โดยอัตโนมัติ และเป้าหมายที่สำคัญและน่าสนใจในการใช้เป็นแนวทางการพัฒนาโครงงานต่อไปคือ การติดตั้งโครงงานนี้ไว้บนยานพาหนะเช่น รถยนต์ รถไฟ และ เรือเป็นต้น โดยชุดควบคุมจะทำการคำนวณและควบคุมให้จานรับสัญญาณดาวเทียมให้เคลื่อนที่ไปในทิศทางที่ถูกต้องตลอดเวลา

โครงงานนี้ได้แบ่งการควบคุมเป็น 2 แบบคือ แบบควบคุมโดยไมโครคอนโทรลเลอร์ และแบบควบคุมโดยไมโครคอมพิวเตอร์ จุดประสงค์ที่มีการควบคุมโดยคอมพิวเตอร์คือ ผู้ใช้สามารถใช้งานได้ง่ายเนื่องจากโปรแกรมบนไมโครคอมพิวเตอร์มีการติดต่อกับผู้ใช้แบบกราฟฟิก และการแก้ไข เพิ่มเติมข้อมูลใหม่จะทำได้ง่ายกว่า การแก้ไขโปรแกรมบนไมโครคอนโทรลเลอร์ แต่ข้อดีของการควบคุมโดยใช้ไมโครคอนโทรลเลอร์คือ ประหยัด มีความกระทัดรัด และเหมาะสมกับงานที่มีการเคลื่อนย้ายบ่อย

ในส่วนของโปรแกรมทั้งบนไมโครคอมพิวเตอร์ มีโหมดการทำงาน 2 โหมดคือ แบบอัตโนมัติ และแบบควบคุมเอง และส่วนไมโครคอโทรลเลอร์จะมีเฉพาะการทำงานแบบควบคุมเอง โดยแบบอัตโนมัติต้องมีอุปกรณ์ประกอบคือ เครื่องบอกทิศทาง โดยเราต้องทำการกำหนดชื่อดาวเทียม และพิกัดตำแหน่งของจานรับสัญญาณดาวเทียมโดยพิกัดนี้จะใส่เป็นชื่อจังหวัดหรือเส้นละติจูด และเส้นลองจิจูดก็ได้ การควบคุมแบบสุดท้ายจะเป็นการควบคุมแบบ ควบคุมเอง โดยเราจะทำการกดสวิทช์เพื่อทำการปรับจานรับสัญญาณดาวเทียมทั้งมุมกวาดและมุมเงย เมื่อได้ตำแหน่งที่ต้องการจึงหยุด

ในทางปฏิบัติโครงงานนี้ยังไม่สามารถใช้งานได้เต็มรูปแบบ ยังไม่สามารถจัดหาเครื่องบอกตำแหน่ง และเครื่องบอกทิศทางได้ โครงงานนี้จึงยังไม่สามารถใช้งานในโหมดอัตโนมัติได้ ส่วนการใช้งานบนยานพาหนะที่กำลังเคลื่อนที่ก็ยังไม่สามารถทำได้ เนื่องจากยังไม่มีข้อมูล และเทคนิคในการควบคุมตำแหน่งจากกรณีนี้

แนวทางในการพัฒนาโครงงาน

ในส่วนของชุดฮาร์ดแวร์ซึ่งเป็นตัวขับจานรับสัญญาณดาวเทียมยังไม่มีความละเอียดเพียงพอ ในอนาคต ถ้ามีการพัฒนาโครงงานนี้ ควรเปลี่ยนไปใช้ระบบขับจานแบบเซอร์โวมอเตอร์ หรือไฮดรอลิก ซึ่งจะให้ความละเอียดและแม่นยำมากกว่าชุดฮาร์ดแวร์ที่ใช้อยู่ปัจจุบัน

เพื่อให้ผลจากการคำนวณค่ามุมโดยโปรแกรมของมุมกวาดและมุมเงยจากโปรแกรมควบคุมมีความแม่นยำมากขึ้น ควรนำอุปกรณ์ชี้ตำแหน่งจากดาวเทียม (GPS) และอุปกรณ์ชี้ทิศทางที่ได้มาตรฐานมาใช้ร่วมกับโครงงานนี้เพื่อความแม่นยำที่ดีขึ้นและใช้งานได้ง่ายขึ้น

ในส่วนของไมโครคอนโทรลเลอร์ สามารถที่จะพัฒนาให้มี ฐานข้อมูลและสามารถคำนวณมุมกวาดและมุมเงยได้ด้วยตัวเอง

ภาคผนวก (ก)

```
;/*********MOBILE RECEIVING CONTROLLER***********/
;/**************Program Main Menu***************/

                ORG     0000H
COMMAND         EQU     0E0C0H
READBUSY        EQU     0E0C1H
WRITEDAT        EQU     0E0C2H
READDATA        EQU     0E0C3H

ADCPORT         EQU     0E020H
FPORT           EQU     0E020H
DPORT           EQU     0E021H
APORT           EQU     0E022H
PPORT           EQU     0E023H

FADDR           EQU     00H
DADDR           EQU     01H
AADDR           EQU     02H
PADDR           EQU     03H

PORTA           EQU     0E0E0H
PORTB           EQU     0E0E1H
PORTC           EQU     0E0E2H
ELMOTOR         EQU     0E0E0H
AZMOTOR         EQU     0E0E1H
A_TO_D          EQU     0E0E2H
CTRLPORT        EQU     0E0E3H


STACK           EQU     30H

WORD_BUF        EQU     70H
RX_BUF          EQU     60H
ADC_BUF         EQU     50H                     ;keep for display

AZDRV_BUF       EQU     10H
ELDRV_BUF       EQU     12H
AZLOC_BUF       EQU     14H
ELLOC_BUF       EQU     16H

KEY_BUF         EQU     20H                     ;keep key for check bit
B00             EQU     00H                     ;bit buf start 20H
B01             EQU     01H
B02             EQU     02H

SAME            EQU     8                       ;address 21H
COMPASS         EQU     9                       ;set when have compass
ANGLE           EQU     10                      ;set when have angle
finder
PLANE           EQU     11
REMOT_BIT       EQU     12
LIMIT_BIT       EQU     13
RV_BIT          EQU     14

EL_BUF          EQU     22H                     ;22H keep is step value
AZ_BUF          EQU     23H                     ;23h-24h is az buf
METER_BUF       EQU     25H
AUX             EQU     26H
AUX7            EQU     37H
DIRECT_BUF      EQU     29H                     ;buffer of direction
motor

LDIRECT         EQU     3                       ;for hardware
MANGLE          EQU     127
MPLANE          EQU     127
```

```
HEX8        EQU   08H
HEX9        EQU   09H
HEXA        EQU   0AH
HEXB        EQU   0BH
HEXC        EQU   0CH
HEXD        EQU   0DH
HEXE        EQU   0EH
HEXF        EQU   0FH
HEXERR      EQU   0FFH
;********************
VAL0         EQU   00H
VAL1         EQU   01H                    ;for display
VAL2         EQU   02H
VAL3         EQU   03H
VAL4         EQU   04H
VAL5         EQU   05H
VAL6         EQU   06H
VAL7         EQU   07H
VAL8         EQU   08H
VAL9         EQU   09H
VALA         EQU   0AH
VALB         EQU   0BH
VALC         EQU   0CH
VALD         EQU   0DH
VALE         EQU   0EH
VALF         EQU   0FH

CVAL0         EQU   30H
CVAL1         EQU   31H
CVAL2         EQU   32H
CVAL3         EQU   33H
CVAL4         EQU   34H
CVAL5         EQU   35H
CVAL6         EQU   36H
CVAL7         EQU   37H
CVAL8         EQU   38H
CVAL9         EQU   39H
CVALA         EQU   41H
CVALB         EQU   42H
CVALC         EQU   43H
CVALD         EQU   44H
CVALE         EQU   45H
CVALF         EQU   46H
;********************
D1PHASE_F   EQU   10H                    ;code drive 1 phase
foword
D1PHASE_R   EQU   14H
D2PHASE_F   EQU   12H
D2PHASE_R   EQU   16H
DHPHASE_F   EQU   11H
DHPHASE_R   EQU   15H
BREAK       EQU   13H                    ;break
STOP        EQU   1BH

PULSE0      EQU   0EFH                   ;use by and
PULSE1      EQU   10H                    ;use by or

;**********speed of drive motor*******************
FAST_EL     EQU   20                     ;fast drive 40 ms/2
(pules)
MIDD_EL     EQU   40
SLOW_EL     EQU   60

FAST_AZ     EQU   14
MIDD_AZ     EQU   28
SLOW_AZ     EQU   42
```

```
PORTA_BUF EQU   27H
PORTB_BUF EQU   28H


KEY1      EQU   1
KEY2      EQU   2
KEY3      EQU   3
KEY4      EQU   4
KEY5      EQU   5
KEY6      EQU   6
KEY7      EQU   7
KEY8      EQU   8
KEY9      EQU   9
KEYA      EQU   16
KEYB      EQU   17
KEYC      EQU   18
KEYD      EQU   19
KEYE      EQU   20
KEYF      EQU   21

SHIFT     EQU   10
KEY0      EQU   11
ENTER     EQU   12
ESC       EQU   13                    ;press shift key
INSERT    EQU   14
RES       EQU   15
KEYUP     EQU   17
KEYLEFT   EQU   19
KEYRIGHT  EQU   21
KEYDOWN   EQU   20
CLEAR     EQU   24

CKEY1      EQU   31H                  ;use for converse to
asci
CKEY2      EQU   32H
CKEY3      EQU   33H

CKEY4      EQU   34H
CKEY5      EQU   35H
CKEY6      EQU   36H
CKEY7      EQU   37H
CKEY8      EQU   38H
CKEY9      EQU   39H
CSHIFT     EQU   40H
CKEY0      EQU   30H
CENTER     EQU   0FFH
CESC       EQU   65H                   ;press shift key
CINSERT    EQU   69H
CRES       EQU   21H
CKEYA      EQU   'A'
CKEYB      EQU   'B'
CKEYC      EQU   'C'
CKEYD      EQU   'D'
CKEYE      EQU   'E'
CKEYF      EQU   'F'
CCLEAR     EQU   4FH
CNONE      EQU   25H
;*********************
HEX0       EQU   00H
HEX1       EQU   01H
HEX2       EQU   02H
HEX3       EQU   03H
HEX4       EQU   04H
HEX5       EQU   05H
HEX6       EQU   06H
HEX7       EQU   07H
```

```
        ;************************************************
        END_CHAR    EQU    '#'                    ;character of end word
        END_BYTE    EQU    3                      ;number  of end byte

        BAUD        EQU    0FDH
        ;************************************************
      ; GPS           EQU 'GPS'                                  ;XXX = is
value of gps
      ; COMPASS       EQU 'CPS'
        EL_WORD       EQU 'E'
        AZ_WORD       EQU 'A'
        UP_WORD       EQU 'U'
        DOWN_WORD     EQU 'D'
        L_WORD        EQU 'L'
        R_WORD        EQU 'R'
        ELD_WORD      EQU 'e'
        AZD_WORD      EQU 'a'
        SELD_WORD     EQU 'c'
        SAZD_WORD     EQU 'd'

        COMPLEATE     EQU 'C'
        ERROR         EQU 'O'
        BUSY          EQU 'B'
        READY         EQU 'Y'
        F_METER       EQU 'F'
        ST_WORD       EQU 'S'
        END_WORD      EQU '#'
        LIMIT         EQU '1'

    ;*************START PROGRAM*************
    ;
        MAIN:       MOV     SP,#STACK
                    CLR     REMOT_BIT
                    MOV     R1,#2                  ;delay for ready device
                    LCALL   DEY_1ms
                    MOV     DPTR,#CTRLPORT
                    MOV     A,#80H                 ;set intial port o/o/o
                    MOVX    @DPTR,A
                    MOV     A,#BREAK               ;break mortor
                    MOV     PORTA_BUF,A
                    MOV     PORTB_BUF,A
                    MOV     DPTR,#ELMOTOR
                    MOVX    @DPTR,A
                    MOV     DPTR,#AZMOTOR
                    MOVX    @DPTR,A
                    LCALL   INIT
                    CLR     REMOT_BIT
        START:      MOV   DPTR,#TITLE
                    LCALL DISPLAY
                    MOV     R1,#2                  ;delay 2 second
                    LCALL   DEY_1s
                    MOV     DPTR,#ABOUT            ;display about team
                    LCALL   DISPLAY
                    MOV     R1,#2                  ;delay 2 second
                    LCALL   DEY_1s
                    lcall   rx_keyc
                    cjne    r0,#insert,start1
                    ljmp    sethard
        start1:     LCALL   ADJ_HW                 ;adjust hardware

        START2:     MOV     DPTR,#LC_OR_RM
                    LCALL   DISPLAY
        REC_KEY:    LCALL RX_KEY                   ;check key

                    CJNE  R0,#INSERT,REC1
                    LJMP  TESTCOMM
```

```
REC1:     CJNE  R0,#KEY1,REC2
          SJMP  LOCAL
REC2:     CJNE  R0,#KEY2,START2
          SJMP  REMOTE
          ;*********SELECT LOCAL***************
          ;
LOCAL:    MOV   DPTR,#SELECT_LC          ;display your select
local
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1s
          LJMP  MANUAL
         ;LJMP  SL_USE                   ;jump to select mode
auto or manual

 ;******************SELECT REMOTE*****************

 REMOTE:  SETB  REMOT_BIT
          MOV   DPTR,#SELECT_RM          ;display your select
remote
          LCALL DISPLAY
          MOV    R1,#2                   ;delay 2 second
          CALL  DEY_1s

          MOV   DPTR,#WAIT_CON           ;display wait for
connecting
          LCALL DISPLAY

 REMOTE_1:LCALL RX_WORD
          MOV   R0,#RX_BUF
          CJNE  @R0,#ST_WORD,REMOTE_1    ;check control word

          MOV   R1,#WORD_BUF
          MOV   @R1,#ST_WORD
          MOV   R2,#1
          LCALL TXR_WORD

COMPARE:  MOV   DPTR,#CONNECT
          lcall disp_ascim
          LCALL RX_WORD
          MOV   R0,#RX_BUF

          CJNE  @R0,#UP_WORD,C_DOWN
          MOV   DPTR,#TRACK_U             ;up
          lcall disp_ascim
          LCALL ASCI_VAL                 ;R3,R4 is drv motor
          MOV   DPTR,#ELMOTOR            ;out el motor
          MOV   R1,#MIDD_EL              ;speed middle
          MOV   R2,#DHPHASE_F            ;hafe phase reverse
          LCALL DRV_MOTOR
          LCALL TX_COMPTE
          LJMP  COMPARE

C_DOWN:   CJNE  @R0,#DOWN_WORD,C_LEFT     ;down
          MOV   DPTR,#TRACK_D
          lcall disp_ascim
          LCALL ASCI_VAL                  ;keep pulse for drv
routeen
          MOV   DPTR,#ELMOTOR            ;out el motor
          MOV   R1,#MIDD_EL              ;speed middle
          MOV   R2,#DHPHASE_R            ;hafe phase reverse
          LCALL DRV_MOTOR
          LCALL TX_COMPTE
          LJMP  COMPARE
```

```
REC1:     CJNE  R0,#KEY1,REC2
          SJMP  LOCAL
REC2:     CJNE  R0,#KEY2,START2
          SJMP  REMOTE
;*********SELECT LOCAL***************
;
LOCAL:    MOV   DPTR,#SELECT_LC          ;display your select
local
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1s
          LJMP  MANUAL
         ;LJMP  SL_USE                   ;jump to select mode
auto or manual

 ;******************SELECT REMOTE*****************

 REMOTE:  SETB  REMOT_BIT
          MOV   DPTR,#SELECT_RM          ;display your select
remote
          LCALL DISPLAY
          MOV    R1,#2                   ;delay 2 second
          CALL  DEY_1s

          MOV   DPTR,#WAIT_CON           ;display wait for
connecting
          LCALL DISPLAY

 REMOTE_1:LCALL RX_WORD
          MOV   R0,#RX_BUF
          CJNE  @R0,#ST_WORD,REMOTE_1    ;check control word

          MOV   R1,#WORD_BUF
          MOV   @R1,#ST_WORD
          MOV   R2,#1
          LCALL TXR_WORD

COMPARE:  MOV   DPTR,#CONNECT
          lcall disp_ascim
          LCALL RX_WORD
          MOV   R0,#RX_BUF

          CJNE  @R0,#UP_WORD,C_DOWN
          MOV   DPTR,#TRACK_U             ;up
          lcall disp_ascim
          LCALL ASCI_VAL                 ;R3,R4 is drv motor
          MOV   DPTR,#ELMOTOR            ;out el motor
          MOV   R1,#MIDD_EL              ;speed middle
          MOV   R2,#DHPHASE_F            ;hafe phase reverse
          LCALL DRV_MOTOR
          LCALL TX_COMPTE
          LJMP  COMPARE

C_DOWN:   CJNE  @R0,#DOWN_WORD,C_LEFT     ;down
          MOV   DPTR,#TRACK_D
          lcall disp_ascim
          LCALL ASCI_VAL                  ;keep pulse for drv
routeen
          MOV   DPTR,#ELMOTOR            ;out el motor
          MOV   R1,#MIDD_EL              ;speed middle
          MOV   R2,#DHPHASE_R            ;hafe phase reverse
          LCALL DRV_MOTOR
          LCALL TX_COMPTE
          LJMP  COMPARE
```

```
                DJNZ  R1,TX12B1
                LCALL TXR_WORD
                MOV   R1,#1
                LCALL DEY_1s
                RET

        TX12B1: INC   R0
                INC   R2
                MOV   A,#'X'
                SJMP  TX12B

        ;*****************************************************
        TX_COMPTE:MOV   R5,#COMPLEATE
                lcall status_f
                RET

        ;*****************************************************
        ;converse val to pulse

        AZVAL_P:  mov   r5,#0
                mov   r6,#0b4h
                lcall sub16bit
                mov   r5,az_buf+1
                mov   r6,az_buf
                lcall sub16bit
                ret

        ELVAL_P:  MOV   R5,EL_BUF+1
                MOV   R6,EL_BUF
                LCALL SUB16BIT                  ;pulse-now
location=pulse for drv
                RET
        ;**********2 complem ment***************
        ; i/p is r3,r4
        ;
        t2cpl:    mov   a,r4
                cpl   a
                mov   r4,a
                mov   a,r3
                cpl   a
                mov   r3,a

                clr   c
                mov   a,#1
                add   a,r4
                mov   r4,a
                mov   a,#0
                addc  a,r3
                mov   r3,a
                ret
;*************************************************

        SL_USE:   MOV   DPTR,#SLUSE             ;display
                LCALL DISPLAY
                MOV   R1,#5
                LCALL DEY_100mS
        SL_USE1:  LCALL RX_KEY
                CJNE  R0,#KEY1,SLUSEL1
                SJMP  AUTO
        SLUSEL1:  CJNE  R0,#KEY2,SLUSEL2
                SJMP  SEMI
        SLUSEL2:  CJNE  R0,#KEY3,SLUSEL3
                SJMP  MANUAL
        SLUSEL3:  MOV   DPTR,#START2
                LCALL RET_MENU
                SJMP  SL_USE
```

```
;*********************************************************
        ;
MANUAL:   MOV   DPTR,#MANMODE
          LCALL DISPLAY
          MOV   R1,#3
          CALL  DEY_1s
          SJMP  SUBMANUAL
       ;**********************
AUTO:
         ;lcall drvstep
          MOV   DPTR,#NONE_OP
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1s
        LJMP  SL_USE

         ;*******************
SEMI:     MOV   DPTR,#SOON
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1s
       LJMP  SL_USE
     ;*************************

SUBMANUAL:mov    a,#aaddr
          lcall  read_adc
          mov    el_buf,a
          mov    dptr,#mankey
          lcall  disp_ascim

MANL1:    LCALL RX_KEYC                    ;check key
          MOV   DPTR,#SL_USE
          LCALL RET_MENU

          CJNE  R0,#KEYUP,MANL2
          MOV   DPTR,#MANUP
          lcall disp_ascim

          MOV   DPTR,#ELMOTOR              ;out el motor
          MOV   R1,#MIDD_EL                ;speed middle
          MOV   R2,#DHPHASE_F              ;hafe phase reverse
          MOV   R3,#0                      ;pulse = 5
          MOV   R4,#1
          LCALL DRV_MOTOR
          SJMP  SUBMANUAL

MANL2:    CJNE  R0,#KEYDOWN,MANL3
          MOV   DPTR,#MANDOWN
          lcall disp_ascim

          MOV   DPTR,#ELMOTOR              ;out el motor
          MOV   R1,#MIDD_EL                ;speed middle
          MOV   R2,#DHPHASE_R              ;hafe phase reverse
          MOV   R3,#0                      ;pulse = 5
          MOV   R4,#1
          LCALL DRV_MOTOR
          SJMP  SUBMANUAL

          ;
MANL3:    CJNE  R0,#KEYLEFT,MANL4
          MOV   DPTR,#MANLEFT
          lcall disp_ascim

          MOV   DPTR,#AZMOTOR              ;out az motor
          MOV   R1,#MIDD_AZ                ;speed middle
          MOV   R2,#DHPHASE_R              ;hafe phase forword
```

```
                MOV   R3,#0                   ;pulse = 5
                MOV   R4,#1
                LCALL DRV_MOTOR
                SJMP  SUBMANUAL

                ;
MANL4:          CJNE  R0,#KEYRIGHT,MANL5
                MOV   DPTR,#MANRIGHT
                lcall disp_ascim

                MOV   DPTR,#AZMOTOR           ;out az motor
                MOV   R1,#MIDD_AZ             ;speed middle
                MOV   R2,#DHPHASE_F           ;hafe phase reverse
                MOV   R3,#0                   ;pulse = 5
                MOV   R4,#1
                LCALL DRV_MOTOR

MANL5:          LJMP  SUBMANUAL
;**********************DISPLAY*********************
;input is dptr,r1 is buf
dp3m:       ;define dptr
                lcall  display
                mov    a,#11                   ;start  locate
                lcall  line3a
               ;mov    r1,#word_buf           ;display number from buf
                mov    r0,#4
                lcall  swarpr
                ret

dp4m:       ;define dptr
                lcall  display
                mov    a,#11                   ;start  locate
                lcall  line4a
               ;mov    r1,#word_buf           ;display number from buf
                mov    r0,#4
                lcall  swarpr
                ret

;**************display asci**************
;i/p  is dptr(address table display)
;buf(1 byte) is value

disp_ascim:push  00h
                push  01h
                push  02h
                mov   r3,#0
                mov   r4,el_buf
                lcall val_asci                ;converse for display
                mov   r0,#word_buf
                mov   r1,#adc_buf
                mov   r2,#4                   ;transfer  buf
dpasml0:        mov   a,@r0
                mov   @r1,a
                djnz  r2,dpasml1
                sjmp  dpasml2
dpasml1:        inc   r0
                inc   r1
                sjmp  dpasml0
dpasml2:        mov   r1,#adc_buf
                lcall dp3m

                ;*****************************
                mov   r3,az_buf+1
                mov   r4,az_buf
                lcall val_asci                ;converse for display
                mov   r0,#word_buf
```

```
                mov   r1,#adc_buf
                mov   r2,#4                     ;transfer  buf
dpasm210:       mov   a,@r0
                mov   @r1,a
                djnz  r2,dpasm211
                sjmp  dpasm212
dpasm211:       inc   r0
                inc   r1
                sjmp  dpasm210
dpasm212:       mov   r1,#adc_buf

                mov    a,#0                              ;start  locate
                lcall  line4a
                mov    r0,#10
                lcall  swarp

                mov    a,#11                 ;start  locate
                lcall  line4a
               ;mov    r1,#word_buf          ;display number from buf
                mov    r0,#4
                lcall  swarpr
                pop    02h
                pop    01h
                pop    00h
                ret

;******************START SUBROUTEAN************************

        ;**********ABOUT DISPLAY************
         WRITE:    PUSH  DPL
                   PUSH  DPH
                   MOV   DPTR,#WRITEDAT
                   MOVX  @DPTR,A
                 LCALL WAITBF
                   POP   DPH
                   POP   DPL
                   RET

         WAITBF:   PUSH  DPL
                   PUSH  DPH
                   MOV   DPTR,#READBUSY
       RDY1:     MOVX  A,@DPTR
                   JB    ACC.7,RDY1
                   POP   DPH
                   POP   DPL
                   RET

         INIT:     PUSH  DPL
                   PUSH  DPH
                   MOV   DPTR,#COMMAND
                   MOV   A,#38H
                   MOVX  @DPTR,A
                   LCALL WAITBF
                   MOV   A,#0FH
                   MOVX  @DPTR,A
                   LCALL WAITBF
                   MOV   A,#6
                   MOVX  @DPTR,A
                   LCALL WAITBF
                   POP   DPH
                   POP   DPL
                   RET

        ;***********DISPLAY NEW LINE **************
        ;
        ;R0 Keep number of charecter
```

```
;Input =

LINE1:    ; PUSH  00H              ;if push error
           PUSH  DPL
           PUSH  DPH
           MOV   A,#80H            ;NEW LINE
           MOV   DPTR,#COMMAND
           MOVX  @DPTR,A
           LCALL WAITBF             ;WAIT BUSY FLAG
           MOV   R0,#16             ;16 char
           POP   DPH
           POP   DPL
          ; POP   00H
           RET

LINE2:    ; PUSH  00h
           PUSH  DPL
           PUSH  DPH
           MOV   A,#0C0H            ;NEW LINE
           MOV   DPTR,#COMMAND
           MOVX  @DPTR,A
           LCALL WAITBF             ;WAIT BUSY FLAG
           MOV   R0,#16
           POP   DPH
           POP   DPL
          ; POP   00H
           RET

LINE3:     ;PUSH  00H
           PUSH  DPL
           PUSH  DPH
           MOV   A,#90H            ;NEW LINE
           MOV   DPTR,#COMMAND
           MOVX  @DPTR,A
           LCALL WAITBF             ;WAIT BUSY FLAG
           MOV   R0,#16
           POP   DPH
           POP   DPL
           ;POP   00H
           RET

LINE4:     ;PUSH  00H
           PUSH  DPL
           PUSH  DPH
           MOV   A,#0D0H           ;NEW LINE
           MOV   DPTR,#COMMAND
           MOVX  @DPTR,A
           LCALL WAITBF             ;WAIT BUSY FLAG
           MOV   R0,#16
           POP   DPH
           POP   DPL
           ;POP   00H
           RET
;**********************************
;bigin address char is a
line3a:    PUSH  DPL
           PUSH  DPH
           ADD   A,#90H            ;NEW LINE
           MOV   DPTR,#COMMAND
           MOVX  @DPTR,A
           LCALL WAITBF             ;WAIT BUSY FLAG
           POP   DPH
           POP   DPL
           ret

line4a:    PUSH  DPL
```

```
            PUSH  DPH
            ADD   A,#0D0H              ;NEW LINE
            MOV   DPTR,#COMMAND
            MOVX  @DPTR,A
            LCALL WAITBF               ;WAIT BUSY FLAG
            POP   DPH
            POP   DPL
            ret

;/******************FORMAT DISPLAY*********************
;
;******************DISPLAY MENU*******************
;input = DPTR by put address of information

DISPLAY:
            PUSH  00H
            PUSH  01H
            PUSH  02H

            LCALL EMPTY                ;display line1
            LCALL LINE1
            LCALL SWARP

            LCALL LINE2
            LCALL INC16                    ;increate char in lcd
            LCALL SWARP

            LCALL LINE3
            LCALL INC16
            LCALL SWARP

            LCALL LINE4
            LCALL INC16
            LCALL SWARP

            POP   02H
            POP   01H
            POP   00H
            RET
 INC16:     PUSH  01H
            MOV   R1,#16                    ;increate DPTR
 INC16_2:   INC   DPTR
            DJNZ  R1,INC16_2
            POP   01H
            RET


;
;******************CLEAR DISPLAY*********************
EMPTY:      PUSH  DPL
            PUSH  DPH
            LCALL WAITBF
            MOV   DPTR,#COMMAND
            MOV   A,#1
            MOVX  @DPTR,A
            LCALL WAITBF
            POP   DPH
            POP   DPL
            RET
            ;
;/******************DISPLAY SWARP*****************
SWARP:      PUSH  00H
            PUSH  DPL                       ;has problem if push
            PUSH  DPH
SWARP1:     MOVX  A,@DPTR
            LCALL WRITE
```

```
                INC   DPTR
                DJNZ  R0,SWARP1
                POP   DPH
                POP   DPL
                POP   00H
                RET

        ;same swarp but use R1
        SWARPR:   PUSH  dph
                  push  dpl
        SWARPR1:  MOV   A,@R1
                  LCALL WRITE
                  INC   R1
                  DJNZ  R0,SWARPR1
                  POP   dpl
                  POP   dph
                  RET


    ;*****************************************************
        ;*****************enter word**********************
        ;o/p is word_buf and R2 keep num byte

        ENT_WORD:  MOV   R1,#WORD_BUF
                   MOV   R2,#0
        ENTW1:
                   push  01h
                   push  02h                    ;byte=0
                   MOV   DPTR,#KEYDISP
                   LCALL DISPLAY

                   CALL  RX_KEY
                   MOV   DPTR,#TCOMM0
                   LCALL RET_MENU
                   LCALL KEY_ASCI                          ;converse'code

                   POP   02H
                   POP   01H

                   MOV   A,R0                              ;keep char
                   MOV   @R1,A

                   PUSH  01H
                   PUSH  02H
                   push  01h
                   mov   dptr,#keydisp+16
                   lcall line2
                   mov   r1,#16
                   lcall swarp

                   lcall line2
                   pop   01h
                   mov   r0,#1
                   lcall swarpr
                   mov   r1,#1
                   lcall dey_1s


                   POP   02H
                   POP   01H
        ENTW2:     CJNE  @R1,#CENTER,ENTW3                 ;check
enter
        ENTEND:

                   RET
```

```
ENTW3:      CJNE  R2,#15,ENTW5
            PUSH  01
            PUSH  02
            MOV   DPTR,#LIMITKEY
            LCALL DISPLAY
            MOV   R1,#2
            LCALL DEY_1S
            POP   02
            POP   01
            CJNE  @R1,#CENTER,ENTW1
            SJMP  ENTEND

ENTW5:      INC   R1
            INC   R2
            LJMP  ENTW1
;**********************return menu******************
;i/p DPTR of address jump
;
RET_MENU:
            CJNE  R0,#ESC,RET_RES
            DEC   SP                                ;dec of lcall
            DEC   SP
            PUSH  DPL
            PUSH  DPH
            RET                                     ;return to
address to dptr
RET_RES:    CJNE  R0,#RES,RET_END
            MOV   SP,#STACK
            LJMP  START2
RET_END:    RET

;************************************************
;
;*****************Receive Key***********
;**************************************
;RO keep Key value is Output Value
;R1 keep Key Max row
;R2 Keep Pattern
;R3 Keep Input Receive Key
;R4 Keep Number of Row
;KEY_BUF is
 RX_KEY:    MOV   R4,#0
            MOV   R0,#0                     ;key valule = 0
            MOV   R1,#4                     ;set max row
            MOV   R2,#01111111B             ;for check shift key
            MOV   P1,R2
            MOV   R3,P1
            MOV   KEY_BUF,R3
            JB    B00,SENDPAT               ;if shift key key
value+12
            LCALL DEBOUNCE                  ;protect noise
            JB    B00,RX_KEY
            MOV   A,#12
            ADD   A,R4
            MOV   R4,A
 SENDPAT:   MOV   R2,#11101111B             ;set pattern
 SENDPAT1:  MOV   P1,R2                     ;send pattern
            MOV   R3,P1                     ;receive key
            MOV   KEY_BUF,R3                ;keep value of key
            JB    B00,CHKC2

            LCALL DEBOUNCE                  ;protect noise
            JB    B00,RX_KEY

            INC   R0                        ;increate key
```

```
                    MOV   A,R0
                    ADD   A,R4
                    MOV   R0,A
                    SJMP  CHKPRESS

          CHKC2:    JB    B01,CHKC3                ;jump if bit=1

                    LCALL DEBOUNCE                 ;protect noise
                    JB    B01,RX_KEY

                    INC   R0
                    INC   R0
                    MOV   A,R0
                    ADD   A,R4                     ;add r0,row
                    MOV   R0,A
                    SJMP  CHKPRESS
          CHKC3:    JB    B02,RECHK
                    LCALL DEBOUNCE                 ;protect noise
                    JB    B02,RX_KEY

                    INC   R0
                    INC   R0
                    INC   R0
                    MOV   A,R0
                    ADD   A,R4
                    MOV   R0,A
                    SJMP  CHKPRESS

          RECHK:    DEC   R1
                    CJNE  R1,#0,RECHK2
                    LJMP  RX_KEY                   ;back to begin check

          RECHK2:   MOV   A,#3
                    ADD
A,R4
                    MOV   R4,A                     ;keep key value

                    MOV   A,R2                     ;rotate pattern
                    RL    A
                    ORL   A,0FH
                    MOV   R2,A
                    SJMP  SENDPAT1

          CHKPRESS: CJNE  R0,#SHIFT,CHKPRESS2      ;if press only shift
recheck
                    LJMP  RX_KEY
          CHKPRESS2: CJNE R0,#SHIFT+12,END_RXKEY     ;if press only shift
recheck
                    LJMP  RX_KEY

          END_RXKEY: RET
;************************************************
                    ;same rx_key but need press key
           RX_KEYC: MOV   R4,#0
                    MOV   R0,#0                    ;key valule = 0
                    MOV   R1,#4                    ;set max row
                    MOV   R2,#01111111B            ;for check shift key
                    MOV   P1,R2
                    MOV   R3,P1
                    MOV   KEY_BUF,R3
                    JB    B00,SENDPATC             ;if shift key key
value+12
                    LCALL DEBOUNCE                 ;protect noise
                    JB    B00,RX_KEYC
                    MOV   A,#12
                    ADD   A,R4
```

```
                MOV   R4,A
SENDPATC:       MOV   R2,#11101111B          ;set pattern
SENDPAT1C:      MOV   P1,R2                  ;send pattern
                MOV   R3,P1                  ;receive key
                MOV   KEY_BUF,R3             ;keep value of key
                JB    B00,CHKC2C

                LCALL DEBOUNCE               ;protect noise
                JB    B00,RX_KEYC

                INC   R0                     ;increate key
                MOV   A,R0
                ADD   A,R4
                MOV   R0,A
                SJMP  CHKPRESSC

CHKC2C:     JB    B01,CHKC3C                 ;jump if bit=1

                LCALL DEBOUNCE               ;protect noise
                JB    B01,RX_KEYC

                INC   R0
                INC   R0
                MOV   A,R0
                ADD   A,R4                   ;add r0,row
                MOV   R0,A
                SJMP  CHKPRESSC
CHKC3C:         JB    B02,RECHKC
                LCALL DEBOUNCE               ;protect noise
                JB    B02,RX_KEYC

                INC   R0
                INC   R0
                INC   R0
                MOV   A,R0
                ADD   A,R4
                MOV   R0,A
                SJMP  CHKPRESSC

RECHKC:         DEC   R1
                CJNE  R1,#0,RECHK2C
                RET

RECHK2C:         MOV   A,#3
                ADD
A,R4
                MOV   R4,A                   ;keep key value

                MOV   A,R2                   ;rotate pattern
                RL    A
                ORL   A,0FH
                MOV   R2,A
                SJMP  SENDPAT1C

CHKPRESSC:   CJNE  R0,#SHIFT,CHKPRESSC2      ;if press only shift
recheck
                LJMP  RX_KEYC
CHKPRESSC2:CJNE  R0,#SHIFT+12,END_RXKEY      ;if press only shift
recheck
                LJMP  RX_KEYC

END_RXKEYC:RET


;*****************DEBOUNCE***************
;repeat send pattern for protect noise
```

```
DEBOUNCE:
            push  01h
            MOV   R1,#1
            LCALL DEY_10ms
            MOV   P1,R2                    ;send pattern

            MOV   R3,P1                    ;receive key
            MOV   KEY_BUF,R3               ;keep value of key
            pop   01h
            RET
;*************************DELAY*******************
        ;
        ;input = R1
        DEY_1ms:   PUSH  01
                   PUSH  02
                   PUSH  03
                   MOV   TMOD,#01H         ;set timer mode 1 16 bit
count
         L1ms_1:   MOV   TH0,#0FFH
                   MOV   TL0,#0DH
                   CLR   TF0
                   SETB  TR0               ;start run timer
         L1ms_2:   JNB   TF0,L1ms_2
                   CLR   TR0
                   CLR   TF0
                   DJNZ  R1,L1ms_1
                   POP   03
                   POP   02
                   POP   01
                   RET


        DEY_10ms:  MOV   TMOD,#01H         ;set timer mode 1 16 bit
count
         L10ms_1:  MOV   TH0,#0ECH
                   MOV   TL0,#79H
                   CLR   TF0
                   SETB  TR0               ;start run timer
         L10ms_2:  JNB   TF0,L10ms_2
                   CLR   TR0
                   CLR   TF0
                   DJNZ  R1,L10ms_1
                   RET


        ;
        ;input = R1
        DEY_100ms: MOV   TMOD,#01H         ;set timer mode 1 16 bit
count
        L100ms_1:  MOV   TH0,#3CH
                   MOV   TL0,#0B1H
                   CLR   TF0
                   SETB  TR0               ;start run timer
         L100ms_2: JNB   TF0,L100ms_2
                   CLR   TR0
                   CLR   TF0
                   DJNZ  R1,L100ms_1
                   RET
        ;

        ;input = R1
        DEY_1s:    MOV   A,R1
                   MOV   B,#10
                   MUL   AB
                   MOV   R1,A
```

```
                MOV   TMOD,#01H              ;set timer mode 1 16 bit
count
        L1s_1:  MOV   TH0,#3CH
                MOV   TL0,#0B1H
                CLR   TF0
                SETB  TR0                    ;start run timer
        L1s_2:  JNB   TF0,L1s_2
                CLR   TR0
                CLR   TF0
                DJNZ  R1,L1s_1
                RET

;************************TX CONTROL WORD*****************
        ;input is R1
        ; "     "  R2 keep number byte of data
        ;when finish data then send endword
        ;
        TX_WORD:  MOVX  A,@R1
                  SETB  TB8            ;***
                  MOV   SCON,#40H                    ;set serial
mode 1
                  MOV   TMOD,#20H                    ;Timer mode 2
                  MOV   TH1,#0FDH                    ;Baud Rate
19.2k
                  MOV   PCON,#80H
                  SETB  TR1                          ;Start timer 1
        TX_WORD1: MOV   SBUF,A
        WAIT_TX:  JNB   TI,WAIT_TX
                  CLR   TI
                  INC   R1
                  MOVX  A,@DPTR
                  DJNZ  R2,TX_WORD1
                  RET
        ;************************************************
        ;same TX_WORD but use I/P is data word_buf
        ;                     R2 is num byte

        TXR_WORD: PUSH  02H
                  MOV  R1,#WORD_BUF
                  MOV  A,@R1
                  SETB  TB8            ;***
                  MOV   SCON,#40H                    ;set serial
mode 2
                  MOV   TMOD,#20H                    ;Timer mode 2
                  MOV   TH1,#0FDH                    ;Baud Rate 9600
                  MOV   PCON,#80H
                  SETB  TR1                          ;Start timer 1
        TXR_WORD1: MOV  SBUF,A
        WAIT_TXR: JNB   TI,WAIT_TXR
                  CLR   TI
                  INC   R1
                  MOV   A,@R1
                  DJNZ  R2,TXR_WORD1
                  POP   02H
                  RET

        ;************************************************
        END_FRAME: PUSH  01
                   PUSH  02
                   MOV   R1,#WORD_BUF
                   MOV   @R1,#END_CHAR
                   INC   R1
                   MOV   @R1,#0
                   INC   R1
                   MOV   A,R2                        ;number of
byte
```

```
                    ADD   A,#END_BYTE                    ;combine end
fram
                    MOV   @R1,A
                    MOV   R2,#END_BYTE                   ;number byte
of end frame
                    LCALL TXR_WORD
                    POP   02
                    POP   01
                    RET


        ;*****************RX CONTROL WORD*******************
        ;outputis  RX_BUF ,R2=num byte
        ;between receive data then check end word
        ;
        RX_WORD:    MOV   R2,#0
                    MOV   R1;#RX_BUF                  ;defind buf
control word
                    MOV   R0,#0FFH                      ;receive  word
limit FF
                    MOV   SCON,#50H                     ;set serial
mode 2
                    MOV   TMOD,#20H                     ;Timer mode 2
                    MOV   TH1,#0FDH                     ;Baud Rate
19.2K
                    MOV   PCON,#80H
                    SETB  TR1                           ;Start timer 1
        WAIT_RX:   ; push  00h
                   ; lcall rx_keyc
                   ; mov   dptr,#start2
                   ; lcall ret_menu
                   ; pop   00h

                    JNB   RI,WAIT_RX
                    CLR   RI
                    MOV   A,SBUF

                    MOV   @R1,A
                    INC   R1
                    INC   R2
                    CJNE  A,#END_CHAR,RX_L1
                    MOV   R0,#END_BYTE

        RX_L1:      DJNZ  R0,WAIT_RX
                    RET
                    ;
 ;********************ADDJUST HARDWARE*********************
        ;
        ADJ_HW:     JNB    COMPASS,MN_SOUTH
                    LCALL  AT_SOUTH
                    SJMP   ADJ_ANGLE
       MN_SOUTH:    mov    a,#daddr
                    lcall  read_adc
                    mov    dptr,#direct_dp
                    lcall  disp_asci
                    lcall  rx_keyc

                    CJNE   R0,#KEYLEFT,SOUTH1
                    MOV   DPTR,#AZMOTOR             ;out az motor
                    MOV   R1,#MIDD_AZ               ;speed middle
                    MOV   R2,#DHPHASE_F             ;hafe phase forword
                    MOV   R3,#5                     ;pulse = 5
                    MOV   R4,#5
                    LCALL DRV_MOTOR
                    SJMP  MN_SOUTH
                    ;
```

```
SOUTH1:   CJNE  R0,#KEYRIGHT,SOUTH2

          MOV   DPTR,#AZMOTOR          ;out az motor
          MOV   R1,#MIDD_AZ            ;speed middle
          MOV   R2,#DHPHASE_R          ;hafe phase reverse
          MOV   R3,#5                  ;pulse = 5
          MOV   R4,#5
          LCALL DRV_MOTOR
          SJMP  MN_SOUTH

SOUTH2:   CJNE  R0,#INSERT,SOUTH3
          LJMP  BURN

SOUTH3:   CJNE  R0,#ENTER,MN_SOUTH
          mov   r1,#5
          lcall dey_100ms

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

ADJ_ANGLE:JNB    ANGLE,MN_ANGLE
          LCALL  AT_ANGLE              ;call auto angle
          SJMP   ADJ_PLANE

MN_ANGLE: mov    a,#aaddr
          lcall  read_adc
          mov    dptr,#angle_dp
          lcall  disp_asci
          LCALL  RX_KEYC

          CJNE   R0,#KEYUP,ANDOWN
          MOV    DPTR,#ELMOTOR         ;out az motor
          MOV    R1,#MIDD_EL           ;speed middle
          MOV    R2,#DHPHASE_F         ;hafe phase forword
          MOV    R3,#5                 ;pulse = 5
          MOV    R4,#5
          LCALL  DRV_MOTOR
          SJMP   MN_ANGLE
          ;
ANDOWN:   CJNE   R0,#KEYDOWN,ANENTER
          MOV    DPTR,#ELMOTOR         ;out az motor
          MOV    R1,#MIDD_EL           ;speed middle
          MOV    R2,#DHPHASE_R         ;hafe phase reverse
          MOV    R3,#5                 ;pulse = 5
          MOV    R4,#5
          LCALL  DRV_MOTOR
          SJMP   MN_ANGLE
ANENTER:  CJNE   R0,#ENTER,MN_ANGLE
          mov    r1,#5
          lcall  dey_100ms
;************************************************

ADJ_PLANE:JNB    plane,MN_PLANE
          LCALL  AT_PLANE
          LJMP   ADJ_END
MN_PLANE: mov    a,#paddr
          lcall  read_adc
          mov    dptr,#plane_dp
          lcall  disp_asci
MNAXIS:   LCALL  RX_KEYC
          CJNE   R0,#ENTER,MN_PLANE
          LJMP   ADJ_END

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

AT_SOUTH: ;mov    dptr,#dport
          mov    a,#daddr
```

```
                lcall read_adc
                mov   aux,a                ;keep value
                mov   dptr,#adirect_dp
                lcall disp_asci

                ;*****set fw or rv************
                mov   dptr,#azmotor            ;check locate
                mov   r2,#dhphase_f            ;forward motor
                mov   r3,#0                    ;pulse = 5
                mov   r4,#5
                lcall drv_motor

                mov   a,#daddr
                lcall read_adc
                push  acc
                mov   dptr,#adirect_dp
                lcall disp_asci
                pop   acc

                clr   c
                subb  a,aux
                jnc   atsl1
                mov   direct_buf,#dhphase_f    ;drive forward
                sjmp  atsl2
atsl1:          mov   direct_buf,#dhphase_r

                ;*********start compare ******
atsl2:          mov   dptr,#azmotor            ;check locate
                mov   r2,#direct_buf           ;forward motor
                mov   r3,#0                    ;pulse = 5
                mov   r4,#5
                lcall drv_motor

                mov   a,#daddr
                lcall read_adc
                mov   aux,a
                push  acc
                mov   dptr,#adirect_dp
                lcall disp_asci
                pop   acc
                clr   c
                subb  a,#ldirect
                jc    endats
                lcall rx_keyc
                cjne  r0,#enter,atsl2          ;if enter pass

endats:         mov   dptr,#ds_disp
                lcall display
                mov   r1,#2
                lcall dey_1s
                ret
                ;*****************************
AT_ANGLE:       mov   a,#aaddr
                lcall setdirect

atangle0:       mov   a,#aaddr
                lcall read_adc
                push  acc
                mov   dptr,#aangle_dp
                lcall disp_asci
                pop   acc

                cjne  a,#mangle-2,atal1
                sjmp  endaangle
atal1:          cjne  a,#mangle-1,atal2
                sjmp  endaangle
```

```
atal2:    cjne  a,#mangle,atal3
          sjmp  endaangle
atal3:    cjne  a,#mangle+1,atal4
          sjmp  endaangle
atal4:    cjne  a,#mangle+1,atal5
          sjmp  endaangle

atal5:    lcall rx_keyc
          cjne  r0,#enter,atangle1
          sjmp  endaangle
atangle1: mov   dptr,#elmotor          ;check locate
          mov   r2,#direct_buf         ;forward motor
          mov   r3,#0                  ;pulse = 5
          mov   r4,#5
          lcall drv_motor
          sjmp  atangle0

endaangle:mov   dptr,#as_disp
          lcall display
          mov   r1,#2
          lcall dey_1s
          RET
          ;*****************************
at_plane: mov   a,#pport
          lcall read_adc
          push  acc
          mov   dptr,#plane_dp
          lcall disp_asci
          mov   r1,#3
          lcall dey_100ms
          pop   acc
          cjne  a,#mplane-2,atpl1
          sjmp  end_ap
atpl1:    cjne  a,#mplane-1,atpl2
          sjmp  end_ap
atpl2:    cjne  a,#mplane,atpl3
          sjmp  end_ap
atpl3:    cjne  a,#mplane+1,atpl4
          sjmp  end_ap
atpl4:    cjne  a,#mplane+1,atpl5
          sjmp  end_ap
atpl5:    lcall rx_keyc
          cjne  r0,#enter,at_plane
end_ap:   mov   dptr,#ps_disp
          lcall display
          mov   r1,#2
          lcall dey_1s
          RET
          ;
          ;*******************
ADJ_END:  mov   a,#0
          mov   az_buf,a               ;start az is 0 degree
          mov   az_buf+1,a
          mov   a,#aaddr               ;keep el locate
          lcall read_adc
          mov   el_buf,a

          MOV   DPTR,#ADJFINISH
          LCALL DISPLAY
          MOV   R1,#2
          LCALL DEY_1s
          RET
          ;*******************
          ;input is dptr,r1 is buf
dpline4:
          ;define dptr
```

```
                lcall  display

                mov    a,#0                              ;start  locate
                lcall  line4a
                mov    r0,#6
                lcall  swarp

                mov    a,#6                       ;start  locate
                lcall  line4a
               ;mov    r1,#word_buf              ;display number from buf
                mov    r0,#4
                lcall  swarpr

                clr    c                          ;incress locate is 11
                mov    a,#11
                add    a,dpl
                mov    dpl,a
                mov    a,#0
                addc   a,dph
                mov    dph,a

                mov    a,#11
                lcall  line4a
                mov    r0,#6
                lcall  swarp
                ret
;*****************************
;i/p=r2 is compare  <-- referrence
;   =a is address of port
;o/p=direct_buf
;
setdirect: lcall read_adc
           clr   c
           subb a,r2
           jc    setdl1
           mov   direct_buf,#dhphase_r   ;set forward
           sjmp  setend
 setdl1:   mov   direct_buf,#dhphase_f   ;set reverse
 setend:   ret
           ;
 ;****compare between input and r2*******
 ;i/p = r2 is compare
 ;      a is value
 ;o/p =same(bit)
 cpip:     mov   r1,direct_buf
           cjne  @r1,#dhphase_f,cpl1
           clr   c
           subb  a,r2                          ;copare middle angle
           jc    cpl3                          ;if more than set same
           sjmp  cpl2
 cpl1:     clr   c
           subb  a,r2
           jc    cpl3
 cpl2:     clr   same
           ret
 cpl3:     setb  same
 cplend:   ret
 ;*************READ ADC*****************
 ;i/p = a is address of input
 ;output is A
 read_adc: mov   dptr,#portc
           movx  @dptr,a
           mov   r1,#1
           lcall dey_10ms
           mov   dptr,#adcport
           movx  a,@dptr
```

```
                movx  a,@dptr

                ret
                ;
        ;**************display asci**************
        ;i/p  is dptr(address table display)
        ;a is value

        disp_asci: push  00h
                   push  01h
                   push  02h
                   mov   r3,#0
                   mov   r4,a
                   lcall val_asci                ;converse for display
                   mov   r0,#word_buf
                   mov   r1,#adc_buf
                   mov   r2,#4                   ;transfer  buf
         dpasl0:   mov   a,@r0
                   mov   @r1,a
                   djnz  r2,dpasl1
                   sjmp  dpasl2
         dpasl1:   inc   r0
                   inc   r1
                   sjmp  dpasl0
         dpasl2:
                   mov   r1,#adc_buf
                   lcall dpline4
                   mov  r1,#5
                   lcall dey_1ms
                   pop   02h
                   pop   01h
                   pop   00h
                   ret

        ;*****************UTILITY****************
        ;IN PUT IS R0
        testport:
                   mov    dptr,#porta
                   mov    a,#0ffh
                  movx   @dptr,a

                  ; mov    dptr,#face
                  ; lcall  display
                   mov    r1,#1
                   lcall  dey_1s

                   mov    a,#0
                  mov    dptr,#porta
                  movx   @dptr,a
                  ;mov    dptr,#face1
                  ;lcall  display

                   mov    r1,#1
                   lcall  dey_1s
                   sjmp   testport


        OUTPORTA:  PUSH   DPL
                   PUSH   DPH
                   MOV    A,R0
                   MOV    DPTR,#PORTA
                   MOVX   @DPTR,A
                   POP    DPH
                   POP    DPL
                   RET
```

```
FACEDISP: MOV   DPTR,#TESTDISP          ;display test
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1s
          MOV   DPTR,#FACE
          LCALL DISPLAY
          MOV   R1,#2
          CALL  DEY_1S
          SJMP  FACEDISP

;*********************************************************
          ;
TESTCOMM: MOV    DPTR,#TMODEDISP        ;display about team
          LCALL  DISPLAY
          MOV    R1,#2                  ;delay 2 second
          CALL   DEY_1s
TCOMM0:   MOV    DPTR,#SEL_COMM
          LCALL  DISPLAY
          MOV    R1,#1                  ;delay 2 second
          CALL   DEY_1s
TCOMM1:   LCALL  RX_KEY                 ;check key
          CJNE   R0,#KEY2,TCL1
          SJMP   TCOMM_TX
TCL1:     CJNE   R0,#KEY3,TCL2
          SJMP   TCOMM_RX
TCL2:     CJNE   R0,#KEY1,TCL3
          LJMP   SETHARD
TCL3:     MOV    DPTR,#START2           ;return menu
          LCALL  RET_MENU
          SJMP   TCOMM1
         ;*****************************
TCOMM_TX: MOV    DPTR,#PRESSKEY
          LCALL  DISPLAY
          MOV    R1,#2                  ;delay 2 second
          CALL   DEY_1s

  TCTX1:  LCALL  ENT_WORD                ;REC KEY
          LCALL  TXR_WORD
          LCALL  WBDISP
          SJMP   TCTX1

  WBDISP: MOV    DPTR,#SENDDISP
          LCALL  DISPLAY
          lcall  line3
          mov    r1,#word_buf
          mov    A,r2
          mov    r0,a                   ;same mov r0,r2
          lcall  swarpr

         ;lcall  end_frame
          MOV    R1,#2
          LCALL  DEY_1S                 ;for last display(*)
          RET

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

TCOMM_RX: MOV    DPTR,#WAIT_REC
          LCALL  DISPLAY
          MOV    R1,#2                  ;delay 2 second
          CALL   DEY_1s

         ; TX=0
         ; CHK RX

TCRX1:    LCALL  RX_WORD
          MOV    DPTR,#REC_DISP
```

```
                LCALL  DISPLAY

                lcall  line3
                mov    r1,#RX_buf
                mov    a,r2
                mov    r0,a                          ;same mov r0,r2
                lcall  swarpr
                mov    r1,#1
                lcall  dey_1s
                mov    dptr,#empdisp
                lcall  display
              SJMP   TCRX1

SETHARD:  mov   dptr,#sethard_dp
          lcall display
          mov   r1,#3
          lcall dey_1s
          mov   dptr,#set_cdp
          lcall display
          mov   r1,#5
          lcall dey_100ms

          lcall rx_key
          cjne  r0,#key1,sh12
          setb  compass
          sjmp  setangle
sh12:     cjne  r0,#key2,sh11
          clr   compass
          sjmp  setangle
sh11:     mov   dptr,#tcomm0
          lcall ret_menu
          ljmp  sethard

setangle: mov   dptr,#set_adp
          lcall display
          mov   r1,#5
          lcall dey_100ms
          lcall rx_key
          cjne  r0,#key1,sal2
          setb  angle
          sjmp  setplane
sal2:     cjne  r0,#key2,sal1
          clr   angle
          sjmp  setplane
sal1:     mov   dptr,#tcomm0
          lcall ret_menu
          sjmp  setangle

setplane: mov   dptr,#set_pdp
          lcall display
          mov   r1,#5
          lcall dey_100ms

          lcall rx_key
          cjne  r0,#key1,spl2
          setb  plane
          sjmp  spl3
spl2:     cjne  r0,#key2,spl1
          clr   plane
          sjmp  spl3
spl1:     mov   dptr,#tcomm0
          lcall ret_menu
          sjmp  setplane
spl3:     mov   dptr,#setcpt
          lcall display
          mov   r1,#2
```

```
                lcall dey_1s
                ret

        ;************************************************
DISP_RX:    MOV   DPTR,#REC_DISP
            LCALL DISPLAY
            lcall line3
            mov   r1,#RX_buf
            mov   a,r2
            mov   r0,a                      ;same mov r0,r2
            lcall swarpr
           mov r1,#5
           lcall dey_100ms

           mov   dptr,#empdisp
           lcall display
           RET


        ;******************drive motor*******************
        ;portmotor=dptr
        ;pattern =R2
        ;number of pulse =R3,R4
        ;speed   =R1
        ;
        DRV_MOTOR:
        DRV0:     MOV   A,R2
                  CJNE  A,#BREAK,DRV1                   ;if break out
motor break
                  SJMP  DRV3
        DRV1:     CJNE  A,#STOP,DRV2
                  SJMP  DRV4

        DRV2:     lcall chklimit                ;check limit motor
                  JNB   LIMIT_BIT,STDRV

                  ;limit motor
                  MOV   DPTR,#LIMIT_DP
                  LCALL DISPLAY
                  MOV   R1,#2
                  LCALL DEY_1S
                  jnb   remot_bit,endlimit

        TX_LIMIT: MOV   R1,#12                  ;send 12 byte
                  MOV   R0,#WORD_BUF            ;send error
                  MOV   R2,#12
                  MOV   A,#LIMIT
        TX12BLM:  MOV   @R0,A

                  DJNZ  R1,TX12BLM
                  LCALL TXR_WORD

        ENDLIMIT: RET

                  ;******************************
        STDRV:    mov   a,r2
                  ANL   A,#PULSE0                 ;out pulse 0
                  MOVX  @DPTR,A
                  mov r1,#1
                  lcall dey_100ms               ;dey for pulse width

                  MOV   A,R2                    ;out pulse 1
                  ORL   A,#PULSE1               ;
                  MOVX  @DPTR,A
                  mov   r1,#1
                  lcall dey_100ms
```

```
                lcall keepaz                    ;keep az locate

DRV5:           djnz  r4,drv2                   ;loop chk limit
                cjne  r3,#0,drv6

DRV3:           mov   a,#faddr                  ;keep angle and signal
                lcall read_adc
                mov   meter_buf,a
                mov   a,#aaddr
                lcall read_adc
                mov   el_buf,a                  ;az in az_buf&az_buf+1

                MOV   A,#BREAK
                MOVX  @DPTR,A
                jb    remot_bit,totxframe

enddrv:         RET
totxframe:ljmp  txframe

DRV4:           MOV   A,#STOP
                MOVX  @DPTR,A
                RET
drv6:           djnz  r3,drv2
                sjmp  drv3

;***************************************
chklimit: mov   a,dpl
                cjne  a,#0e0h,limitaz
                lcall chk_rv                    ;using el mortor
                jb    rv_bit,limel_rv

                ;*****forward*****
                mov   a,el_buf
                clr   c
                subb  a,#0ffh
                jc    limel_l1
                setb  limit_bit
                ret
                ;****don't limit***
limel_l1:       clr   limit_bit
                ret

                ;*****reverse*****
limel_rv:
                mov   a,el_buf
                clr   c
                subb  a,#0
                jnc   limel_l2
                setb  limit_bit
                ret
                ;****don't limit***
limel_l2:       clr   limit_bit
                ret

                ;*****************************
 limitaz:                                       ;using az motor
                lcall chk_rv
                jb    rv_bit,limaz_rv

                ;*****forward*****
                mov   r4,az_buf
                mov   r3,az_buf+1
                mov   r6,#0d0h
                mov   r5,#02h
```

```
                clr   c
                lcall sub16bit
                jc    limaz_l1
                setb  limit_bit
                ret
                ;****don't limit***
limaz_l1:       clr   limit_bit
                ret

                ;*****reverse*****
limaz_rv:
                mov   r4,az_buf
                mov   r3,az_buf+1
                mov   r6,#03h
                mov   r5,#0fdh

                clr   c
                lcall sub16bit

                jnc   limel_l2
                setb  limit_bit
                ret
                ;****don't limit***
limaz_l2:       clr   limit_bit
                ret

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

 chk_rv:        cjne   r2,#d1phase_f,chkl1
                sjmp   chkf_l1
 chkl1:         cjne   r2,#d2phase_f,chkl2
                sjmp   chkf_l1
 chkl2:         cjne   r2,#dhphase_f,chkl3
                sjmp   chkf_l1

 chkl3:         cjne   r2,#d1phase_r,chkl4
                sjmp   chkr_l1
 chkl4:         cjne   r2,#d2phase_r,chkl5
                sjmp   chkr_l1
 chkl5:         cjne   r2,#dhphase_r,chkl6
                sjmp   chkr_l1

 chkf_l1:       clr    rv_bit
                ret
 chkr_l1:       setb   rv_bit
                ret
 chkl6:         clr    rv_bit
                ret

 ;**************************************
 ;increate az_buf
 ;
 keepaz:    mov    a,#0E1H
            cjne   a,dpl,endkeep          ;check motor using

            cjne   r2,#d1phase_f,kazl1
            sjmp   az_fw
kazl1:      cjne   r2,#d2phase_f,kazl2
            sjmp   az_fw
kazl2:      cjne   r2,#dhphase_f,kazl3
            sjmp   az_fw

kazl3:      cjne   r2,#d1phase_r,kazl4
            sjmp   az_rv
kazl4:      cjne   r2,#d2phase_r,kazl5
            sjmp   az_rv
```

```
kaz15:      cjne    r2,#dhphase_r,endkeep
            sjmp    az_rv

az_fw:      mov     a,az_buf
            add     a,#1
            mov     az_buf,a
            jc      azfwl1
            ret
azfwl1:     inc     az_buf+1
            ret

az_rv:      mov     a,az_buf
            clr     c
            subb    a,#1
            mov     az_buf,a
            jc      azrvl1
            ret
azrvl1:     dec     az_buf+1
endkeep:    ret

;*************************************
txframe:    PUSH    00H
            PUSH    02H
            PUSH    04H
            PUSH    05H
            PUSH    DPH
            PUSH    DPL
            MOV     R5,#'X'
            LCALL   STATUS_F
            POP     DPL
            POP     DPH
            POP     05H
            POP     04H
            POP     02H
            POP     00H
            ljmp    enddrv
;*************************************
STATUS_F:   mov     r3,#0
            mov     r4,meter_buf
            lcall   val_asci
            mov     r0,#word_buf
            mov     a,r5
            mov     @r0,a
            inc     r0
            mov     @r0,#f_meter
            MOV     R2,#4
            LCALL   TXR_WORD
            ;*****************************
            MOV     R3,#0
            MOV     R4,EL_BUF
            LCALL   VAL_ASCI
            MOV     R0,#WORD_BUF
            MOV     @R0,#EL_WORD            ;send elevation word
            MOV     R2,#4
            LCALL   TXR_WORD
            ;*****************************
            MOV     R3,AZ_BUF+1
            MOV     R4,AZ_BUF
            LCALL   VAL_ASCI
            MOV     R0,#WORD_BUF
            MOV     @R0,#AZ_WORD            ;send elevation word
            MOV     R2,#4
            LCALL   TXR_WORD
            ret

;*************************************************
```

```
DRVSTEP:  MOV   R4,#8                    ;8 step
          MOV   DPTR,#STEP_PAT
 DRVSTEP1:MOV   A,PORTA_BUF
          ANL   A,#00111111B
          MOV   PORTA_BUF,A
          MOVX   A,@DPTR
          ANL   A,#11000000B
          ORL   A,PORTA_BUF              ;or old data
          MOV   PORTA_BUF,A

          MOV   A,PORTB_BUF
          ANL   A,#00111111B
          MOV   PORTB_BUF,A
          MOVX  A,@DPTR
          RL    A
          RL    A
          ANL   A,11000000B
          ORL   A,PORTB_BUF
          MOV   PORTB_BUF,A

          PUSH  DPH
          PUSH  DPL
          MOV   DPTR,#PORTB
          MOVX  @DPTR,A                  ;out
          MOV   A,PORTA_BUF
          MOV   DPTR,#PORTA
          MOVX  @DPTR,A
          POP   DPL
          POP   DPH

          MOV   R1,#1
          LCALL DEY_10ms
          INC   DPTR
          DJNZ  R4,DRVSTEP1
          SJMP  DRVSTEP

;*************************************************************
          ;use converse num key to asci code
          ;i/p ,o/p =R0
          ;if no code then display "none"

          KEY_ASCI: CJNE   R0,#KEY1,ASCI1
                    MOV    R0,#CKEY1
                    RET
          ASCI1:    CJNE   R0,#KEY2,ASCI2
                    MOV    R0,#CKEY2
                    RET
          ASCI2:    CJNE   R0,#KEY3,ASCI3
                    MOV    R0,#CKEY3
                    RET
          ASCI3:    CJNE   R0,#KEY4,ASCI4
                    MOV    R0,#CKEY4
                    RET
          ASCI4:    CJNE   R0,#KEY5,ASCI5
                    MOV    R0,#CKEY5
                    RET
          ASCI5:    CJNE   R0,#KEY6,ASCI6
                    MOV    R0,#CKEY6
                    RET
          ASCI6:    CJNE   R0,#KEY7,ASCI7
                    MOV    R0,#CKEY7
                    RET
          ASCI7:    CJNE   R0,#KEY8,ASCI8
                    MOV    R0,#CKEY8
                    RET
          ASCI8:    CJNE   R0,#KEY9,ASCI9
```

```
            MOV     R0,#CKEY9
            RET
ASCI9:      CJNE    R0,#KEY0,ASCI10
            MOV     R0,#CKEY0
            RET
ASCI10:     CJNE    R0,#SHIFT,ASCI11
            MOV     R0,#CSHIFT
            RET
ASCI11:     CJNE    R0,#ENTER,ASCI12
            MOV     R0,#CENTER
            RET
ASCI12:     CJNE    R0,#ESC,ASCI13
            MOV     R0,#CESC
            RET
ASCI13:     CJNE    R0,#INSERT,ASCI14
            MOV     R0,#CINSERT
            RET
ASCI14:     CJNE    R0,#RES,ASCI15
            MOV     R0,#CRES
            RET
ASCI15:     CJNE    R0,#KEYA,ASCI16
            MOV     R0,#CKEYA
            RET
ASCI16:     CJNE    R0,#KEYB,ASCI17
            MOV     R0,#CKEYB
            RET
ASCI17:     CJNE    R0,#KEYC,ASCI18
            MOV     R0,#CKEYC
            RET
ASCI18:     CJNE    R0,#KEYD,ASCI19
            MOV     R0,#CKEYD
            RET
ASCI19:     CJNE    R0,#KEYE,ASCI20
            MOV     R0,#CKEYE
            RET
ASCI20:     CJNE    R0,#KEYF,ASCI21
            MOV     R0,#CKEYF
            RET

ASCI21:     MOV     R0,#CNONE
            RET

;***********************************************
;use converse num key to code
;i/p ,o/p =R0
;if no code then display "none"

KEY_HEX:    CJNE    R0,#KEY1,KVAL1
            MOV     R0,#HEX1
            RET
KVAL1:      CJNE    R0,#KEY2,KVAL2
            MOV     R0,#HEX2
            RET
KVAL2:      CJNE    R0,#KEY3,KVAL3
            MOV     R0,#HEX3
            RET
KVAL3:      CJNE    R0,#KEY4,KVAL4
            MOV     R0,#HEX4
            RET
KVAL4:      CJNE    R0,#KEY5,KVAL5
            MOV     R0,#HEX5
            RET
KVAL5:      CJNE    R0,#KEY6,KVAL6
            MOV     R0,#HEX6
            RET
KVAL6:      CJNE    R0,#KEY7,KVAL7
```

```
                MOV     R0,#HEX7
                RET
KVAL7:          CJNE    R0,#KEY8,KVAL8
                MOV     R0,#HEX8
                RET
KVAL8:          CJNE    R0,#KEY9,KVAL9
                MOV     R0,#HEX9
                RET
KVAL9:          CJNE    R0,#KEY0,KVAL10
                MOV     R0,#HEX0
                RET
KVAL10:         CJNE    R0,#KEYA,KVAL11
                MOV     R0,#HEXA
                RET
KVAL11:         CJNE    R0,#KEYB,KVAL12
                MOV     R0,#HEXB
                RET
KVAL12:         CJNE    R0,#KEYC,KVAL13
                MOV     R0,#HEXC
                RET
KVAL13:         CJNE    R0,#KEYD,KVAL14
                MOV     R0,#HEXD
                RET
KVAL14:         CJNE    R0,#KEYE,KVAL15
                MOV     R0,#HEXE
                RET
KVAL15:         CJNE    R0,#KEYF,KVAL16
                MOV     R0,#HEXF
                RET
KVAL16:         CJNE    R0,#ENTER,ENDKVAL
                RET
ENDKVAL:        MOV     R0,#HEXERR
                RET
;*************converse value to assci  code********
;i/p=R3,R4 ,o/p=word_buf
;
VAL_ASCI:       MOV     R1,#WORD_BUF
                MOV     A,R3
                SWAP    A                               ;select hi
byte
                ANL     A,#0FH                          ;mask
                CALL    VAL_AS0
                MOV     @R1,A
                INC     R1
                MOV     A,R3
                ANL     A,#0FH                           ;mask
                CALL    VAL_AS0
                MOV     @R1,A
                INC     R1

                MOV     A,R4
                SWAP    A                               ;select hi
byte
                ANL     A,#0FH                          ;mask
                CALL    VAL_AS0
                MOV     @R1,A
                INC     R1
                MOV     A,R4
                ANL     A,#0FH                           ;mask
                CALL    VAL_AS0
                MOV     @R1,A
                RET
VAL_AS0:        CJNE    A,#VAL0,VAL_AS1
                MOV     A,#CVAL0
                RET
VAL_AS1:        CJNE    A,#VAL1,VAL_AS2
```

```
                    MOV     A,#CVAL1
                    RET
          VAL_AS2:  CJNE    A,#VAL2,VAL_AS3
                    MOV     A,#CVAL2
                    RET
          VAL_AS3:  CJNE    A,#VAL3,VAL_AS4
                    MOV     A,#CVAL3
                    RET
          VAL_AS4:  CJNE    A,#VAL4,VAL_AS5
                    MOV     A,#CVAL4
                    RET
          VAL_AS5:  CJNE    A,#VAL5,VAL_AS6
                    MOV     A,#CVAL5
                    RET
          VAL_AS6:  CJNE    A,#VAL6,VAL_AS7
                    MOV     A,#CVAL6
                    RET
          VAL_AS7:  CJNE    A,#VAL7,VAL_AS8
                    MOV     A,#CVAL7
                    RET
          VAL_AS8:  CJNE    A,#VAL8,VAL_AS9
                    MOV     A,#CVAL8
                    RET
          VAL_AS9:  CJNE    A,#VAL9,VAL_ASA
                    MOV     A,#CVAL9
                    RET
          VAL_ASA:  CJNE    A,#VALA,VAL_ASB
                    MOV     A,#CVALA
                    RET
          VAL_ASB:  CJNE    A,#VALB,VAL_ASC
                    MOV     A,#CVALB
                    RET
          VAL_ASC:  CJNE    A,#VALC,VAL_ASD
                    MOV     A,#CVALC
                    RET
          VAL_ASD:  CJNE    A,#VALD,VAL_ASE
                    MOV     A,#CVALD
                    RET
          VAL_ASE:  CJNE    A,#VALE,VAL_ASF
                    MOV     A,#CVALE
                    RET
          VAL_ASF:  CJNE    A,#VALF,END_VAL
                    MOV     A,#CVALF
          END_VAL:  RET
                    ;
;*******************converse asci to value*****************
;i/p is rx_buf
;o/p is r3,r4
;
          asci_val:push     00h
                   mov      r0,#rx_buf
                   inc      r0                              ;skip word
                   mov      a,@r0
                   lcall    as_val0
                   mov      r3,a                            ;a is val

                   inc      r0
                   mov      a,@r0
                   lcall    as_val0
                   swap     a
                   mov      r4,a

                   inc      r0
                   mov      a,@r0
                   lcall    as_val0
                   orl      a,r4
```

```
                    mov     r4,a                                    ;r4 is value
xx

                    pop     00h
                    ret

          as_val0: cjne     a,#cval0,as_val1
                    mov     a,#val0
                    ret
          as_val1: cjne     a,#cval1,as_val2
                    mov     a,#val1
                    ret
          as_val2: cjne     a,#cval2,as_val3
                    mov     a,#val2
                    ret
          as_val3: cjne     a,#cval3,as_val4
                    mov     a,#val3
                    ret
          as_val4: cjne     a,#cval4,as_val5
                    mov     a,#val4
                    ret
          as_val5: cjne     a,#cval5,as_val6
                    mov     a,#val5
                    ret
          as_val6: cjne     a,#cval6,as_val7
                    mov     a,#val7
                    ret
          as_val7: cjne     a,#cval7,as_val8
                    mov     a,#val7
                    ret
          as_val8: cjne     a,#cval8,as_val9
                    mov     a,#val8
                    ret
          as_val9: cjne     a,#cval9,as_vala
                    mov     a,#val9
                    ret
          as_vala: cjne     a,#cvala,as_valb
                    mov     a,#vala
                    ret
          as_valb: cjne     a,#cvalb,as_valc
                    mov     a,#valb
                    ret
          as_valc: cjne     a,#cvalc,as_vale
                    mov     a,#valc
                    ret
          as_vald: cjne     a,#cvald,as_vale
                    mov     a,#vald
                    ret
          as_vale: cjne     a,#cvale,as_valf
                    mov     a,#vale
                    ret
          as_valf: cjne     a,#cvalf,asvalend
                    mov     a,#valf
                    ret
         asvalend: mov      a,#00h
                    ret


;*************************************************************
        ;*******************calculate**********************
        ; i/p r3,r4 and r5,r6
        ; o/p r3,r4
        ;
        ADD16BIT: CLR   C
                  MOV   A,R4
                  ADD   A,R6
```

```
                MOV   R4,A
                ;
                MOV   A,R3
                ADDC  A,R5
                MOV   R3,A
                RET
                ;
        SUB16BIT: CLR   C
                MOV   A,R4
                SUBB  A,R6
                MOV   R4,A
                ;
                MOV   A,R3
                SUBB  A,R5
                MOV   R3,A
                RET
        ;*****************************
        ;i/p r3,r4
        RESULT:   MOV   DPTR,#ADD_DISP
                LCALL DISPLAY
                LCALL LINE3
                MOV   R1,#WORD_BUF
                MOV   R0,#4                     ;4 char
                LCALL SWARPR
                RET

;*******************PATTERN STERPING**********************

        STEP_PAT: DB 00010000B
                DB 00110000B
                DB 00100000B
                DB 01100000B
                DB 01000000B
                DB 11000000B
                DB 10000000B
                DB 10010000B


;******************BUFFER OF CHARECTER **********************
        ;
        TITLE:    DB "****************"
                DB "MOBILE RECEIVING"
                DB "   CONTROLLER   "
                DB "****************"

        ABOUT:    DB "****************"
                DB "   PROJECT  BY  "
                DB "TELECOM ENGINEER"
                DB "****************"

        DIRECT_DP:DB " press <- or -> "
                DB "adjust direction"
                DB "& ent for accept"
                DB "----->    <-----"

        ADIRECT_DP:DB "  pleate  wait  "
                 DB " for  automatic "
                 DB "adjust direction"
                 DB "----->    <-----"

        DS_DISP:  DB "****************"
                DB "ADJUST DIRECTION"
                DB "     SUCCESS    "
                DB "****************"

        ANGLE_DP: DB "press up or down"
```

```
                DB "  adjust angle  "
                DB " &ent for accept"
                DB "----->    <-----"

AANGLE_DP:DB "  pleate  wait  "
                DB " for  automatic "
                DB "  adjust angle  "
                 DB "----->    <-----"

 AS_DISP: DB "****************"
                DB "  ADJUST ANGLE  "
                DB "     SUCCESS    "
                DB "****************"

 PLANE_DP: DB "  press  adjust "
                DB "     plane      "
                DB "&ent for accept "
                DB "----->    <-----"

 PS_DISP:  DB "****************"
                DB "  ADJUST PLANE  "
                DB "    SUCCESS     "
                DB "****************"

 ADJFINISH:DB "****************"
                DB " ADJUST FINISH  "
                DB "    --------    "
                DB "****************"


 LC_OR_RM: DB "  SELECT MODE   "
                DB "  1. LOCAL      "
                DB "  2. REMOTE     "
                DB "    -------     "

 SELECT_LC:DB "****************"
                DB "  LOCAL  MODE   "
                DB "  IS SELECTED   "
                DB "****************"

SELECT_RM: DB "****************"
                DB "  REMOTE  MODE  "
                DB "  IS SELECTED   "
                DB "****************"

 WAIT_CON: DB "  PLEASE WAIT   "
                DB "      FOR       "
                DB "  CONNECTING    "
                DB "     ------     "

 CONNECT:  DB "   connecting   "
                DB "    computer    "
                DB " EL angle=      "
                DB " AZ angle=      "

 TRACK_EL: DB "    tracking    "
                DB "   elevaltion   "
                DB " EL angle=      "
                DB " AZ angle=      "

 TRACK_AZ: DB "    tracking    "
                DB "    azimuth     "
                DB " EL angle=      "
                DB " AZ angle=      "

 TRACK_U:  DB "    tracking    "
```

```
            DB "       up        "
            DB " EL angle=       "
            DB " AZ angle=       "

TRACK_D:    DB "    tracking     "
            DB "      down       "
            DB " EL angle=       "
            DB " AZ angle=       "



TRACK_L:    DB "    tracking     "
            DB "      left       "
            DB " EL angle=       "
            DB " AZ angle=       "

TRACK_R:    DB "    tracking     "
            DB "     right       "
            DB " EL angle=       "
            DB " AZ angle=       "

P_COMPT:    DB "*****************"
            DB "    PROCESS      "
            DB "  COMPLEATE      "
            DB "*****************"

P_ERROR:    DB "*****************"
            DB "    PROCESS      "
            DB "     ERROR       "
            DB "*****************"

LIMIT_DP:   DB "*****************"
            DB "  CAN NOT MOVE   "
            DB " MOTOR IS LIMIT  "
            DB "*****************"

SLUSE:      DB "  SELECT MODE    "
            DB "  1. AUTO        "
            DB "  2. SEMI AUTO   "
            DB "  3. MANUAL      "
            DB "  ------------   "

SATT:       DB "*****************"
            DB " PLEASE SELECT   "
            DB "   SATTELLITE    "
            DB "*****************"

SATTIS:     DB "  SATTELLITE IS  "
            DB "                 "
            DB "                 "
            DB "                 "

LOCATE:     DB "*****************"
            DB " PLEASE SELECT   "
            DB " YOUR LOCATION   "
            DB "*****************"

MANMODE:    DB "*****************"
            DB " manual  mode    "
            DB "     ready       "
            DB "*****************"

MANKEY:     DB " pleate  select  "
            DB "   arrow  key    "
            DB " EL angle=       "
```

```
                DB " AZ angl =      "

MANLEFT:    DB "    tracking    "
            DB "     left       "
            DB " EL angle=      "
            DB " AZ angle=      "



MANRIGHT:   DB "    tracking    "
            DB "     right      "
            DB " EL angle=      "
            DB " AZ angle=      "

MANUP:      DB "    tracking    "
            DB "       up       "
            DB " EL angle=      "
            DB " AZ angle=      "

MANDOWN:    DB "    tracking    "
            DB "      down      "
            DB " EL angle=      "
            DB " AZ angle=      "

SELECT_LO:DB " Press Arrow Key"
            DB " Or INS if know "
            DB "   Lititude &   "
            DB "   Longitude    "

LOCATEIS:   DB " YOUR LOCATION  "
            DB "       IS       "
            DB "                "
            DB "                "

LATITUDE:   DB "****************"
            DB "LATITUDE IS     "
            DB "LONGITUDE IS    "
            DB "****************"

NONE_OP:    DB "****************"
            DB "      SORY      "
            DB "   NO  OPTION   "
            DB "****************"


SOON:       DB "****************"
            DB "  This  option  "
            DB "   is up next   "
            DB "****************"


DING:       DB "****************"
            DB "  DINGONOSTIC   "
            DB "      MODE      "
            DB "****************"

TESTDISP:   DB "****************"
            DB "  THIS IS TEST  "
            DB "    DISPLAY     "
            DB "****************"

FACE:       DB "   ***###***    "
            DB "  @ -O---O- @   "
            DB "    |   o   |   "
```

```
                DB "    | ----- |     "

        FACE1:  DB "    *******     "
                DB "  @  -   -  @    "
                DB "     |   o   |    "
                DB "     |   *   |    "


        EMPDISP: DB "                 "
                DB "  waiting  key   "
                DB "                 "
                DB "                 "



ADD_DISP: DB "*****************"
                DB "   RESULT IS     "
                DB "                 "
                DB "*****************"

        TMODEDISP:DB "  DINGONOSTIC    "
                DB "       AND       "
                DB "  COFIG  MODE    "
                DB "    --------     "

        SEL_COMM: DB " pleast select  "
                DB "1.set hardware  "
                DB "2.send test     "
                DB "3.receive test  "

        SETHARD_DP:DB  "*****************"
                 DB  "  SET HARDWARE   "
                 DB  "      MODE       "
                 DB  "*****************"

        SET_CDP:  DB "   do you have   "
                DB "compass sensor ?"
                DB "  1. YES  2. NO  "
                DB "    ----------   "

        SET_ADP:  DB "  do you have    "
                DB "auto adj angle ?"
                DB " 1. YES   2. NO  "
                DB "    ----------   "

        SET_PDP:  DB "  do you have    "
                DB "auto adj plane ?"
                DB " 1. YES   2. NO  "
                DB "    ----------   "

        SETCPT:   DB "*****************"
                DB "  SET  HARDWARE  "
                DB "    COMPLEATE    "
                DB "*****************"

        PRESSKEY: DB "*****************"
                DB " please press    "
                DB " key 0-9 & ENT   "
                DB "*****************"

        KEYDISP:  DB "You press key-->"
                DB "              -->"
                DB "              -->"
                DB "              -->"
```

```
LIMITKEY: DB "*****************"
          DB "  limit 16 key  "
          DB "  please enter  "
          DB "*****************"


SENDDISP: DB "*****************"
          DB "  send key  is  "
          DB "                "
          DB "*****************"

WAIT_REC: DB "*****************"
          DB "   plest wait   "
          DB " for  receiving "
          DB "*****************"

REC_DISP: DB "*****************"
          DB "   RECEIVE IS   "
          DB "                "
          DB "*****************"

          END
```

ภาคผนวก (ข)

```
Private Sub Form_Load()
Dim i
Form1.WindowState = 2
Load form2
Load Form3
Load Form4
Load form5
port
test
form5.comm1.PortOpen = False
End Sub
############################
Private Sub Hardware_Click()
form2.Show
Form1.Enabled = False
End Sub
############################
Private Sub DatItm_Click()
Form1.Enabled = False
Form3.Show
End Sub
#########################
Private Sub perItm_Click()
Form4.Show
Form1.Enabled = False
End Sub
#########################
Private Sub Exititm_Click()
Unload Form1
Unload form2
Unload Form3
Unload Form4
Unload form5
MsgBox ("GOOD    LUCK !!"), , "MESSAGE!!!..."
Exit Sub
End Sub
#########################
Private Sub btnup_Click()
form2.Comm1.CommPort
port
i = "U" + "005" + "#" + "0" + "5"
sent (i)
form5.comm1.PortOpen = False
End Sub
#########################
Private Sub btnright_Click()
port
i = "R" + "005" + "#" + "0" + "5"
sent (i)
form5.comm1.PortOpen = False
End Sub
#########################
Private Sub btndown_Click()
port
i = "D" + "005" + "#" + "0" + "5"
sent (i)
form5.comm1.PortOpen = False
End Sub
#########################
Private Sub btnleft_Click()
port
i = "L" + "005" + "#" + "0" + "5"
sent (i)
```

```
form5.comm1.PortOpen = False
End Sub
########################
Private Sub btnclose_Click()
form2.Hide
Form1.Enabled = True
End Sub
########################
Private Sub comlist_Click(Area As Integer)
Dim db As Database
Dim myset As Recordset
Dim tloc As String
Set db = OpenDatabase("C:\data\data.mdb")
Set myset = db.OpenRecordset("satellite")
tloc = comlist
myset.Index = "satellite"
myset.Seek "=", tloc
If Not myset.NoMatch Then
txtloc = myset!location
txtband = myset!band
End If
End Sub
########################
Private Sub txtloc_Change()
Dim dif As Single
Dim azd1, azd2, azd3, azd4  As Single
Dim azr1, azr2, azr3 As Single
Dim total, t1, t2, t3 As Single
Dim eld1, eld2, eld3, eld4 As Single
Dim elr1, elr2, elr4 As Single

        'FINE AZIMUTH DEGREE
dif = Val(txtlong) - Val(txtloc)
azr1 = dif * (3.14 / 180)
azd1 = Tan(azr1)
azr2 = txtlat * (3.14 / 180)
azd2 = Sin(azr2)
azd3 = azd1 / azd2
total = Atn(azd3)
Deg_Az1 = total * (180 / 3.14)
Deg_Az2 = Deg_Az1 + 180
Deg_Az = Format(Deg_Az2, "##,##00.00")
textaz = Deg_Az     'DISPLAY AZIMUTH

'VALUE TO SENT
Val_DegAz_Step = Deg_Az2 / 0.25
Val_DegAz_Sent = Format(Val_DegAz_Step, "#,#0")
        'FIND ELEVATION DEGREE
el1 = Cos(azr1)
el2 = Cos(azr2)
el3 = el1 ^ 2
el4 = el2 ^ 2
eld1 = (el1 * el2) - 0.1508
eld2 = Sqr(1 - (el3 * el4))
eld4 = eld1 / eld2
eld5 = Atn(eld4)
eld6 = eld5 * (180 / 3.14)
eld7 = Format(eld6, "##,##00.00")
textel = eld7      'DISPLAY ELEVATION DEGREE

        'VALUE TO SENT
Val_DegEl_Step = eld6 / 0.36
Val_DegEl_Sent = Format(Val_DegEl_Step, "#,#0")
End Sub
```

```
Private Sub comlist2_Click(Area As Integer)
Dim db As Database
Dim myset As Recordset
Dim tlist As String
Set db = OpenDatabase("C:\data\data.mdb")
Set myset = db.OpenRecordset("thailand")
tlist = comlist2
myset.Index = "city"
myset.Seek "=", tlist
If Not myset.NoMatch Then
txtlat = myset!lati
txtlong = myset!Long
End If
End Sub
########################
Private Sub Check1_click()
If Form3.Check1.Value = 1 Then
Frame1.Enabled = False
Frame2.Enabled = False
Frame3.Enabled = False
Else
Frame1.Enabled = True
Frame2.Enabled = True
Frame3.Enabled = True
End If
End Sub
########################
Private Sub BtnOk_Click()
If Check1.Value = 1 Then
port

        ' VALUE STEP MOTOR FOR DEFINE AZIMUTH
Stg_DefAz = Textda.Text
Val_DefAz = Val(Stg_DefAz)
Val_DefAz_Step = Val_DefAz / 0.25
Val_DefAz_Sent = Format(Val_DefAz_Step, "###")
Stg_DefAz_Sent = Str(Val_DefAz_Sent)


        'VALUE STEP MOTOR FOR DEFINE ELEVATION
Stg_DefEl = Textde.Text
Val_DefEl = Val(Stg_DefEl)
Val_DefEl_Step = Val_DefEl / 0.36
Val_DefEl_Sent = Format(Val_DefEl_Step, "#")
Num_El = Len(Val_DefEl_Sent)
Stg_DefEl_Sent = Str(Val_DefEl_Sent)


        'SENT DEFINE AZIMUTH DEGREE
i = "a" + "0" + Stg_DefAz_Sent + "#07"
sent (i)
Do
dummy = DoEvents()
Loop Until form5.comm1.InBufferCount >= 12
Turn_Word = form5.comm1.Input
compare (Turn_Word)


        'SENT DEFINE ELEVATION DEGREE
i = "e" + "0" + Stg_DefEl_Sent + "#07"
sent (i)
Do
dummy = DoEvents()
Loop Until form5.comm1.InBufferCount >= 12
```

```
Turn_Word = form5.comm1.Input
compare (Turn_Word)
Else

        'SENT AZIMUTH DEGREE
Val_DegAz_Step = textaz.Text
Val_DegAz_Step = Val(Val_DegAz_Step)
Val_DegAz_Sent = Format(Val_DegAz_Step, "#0")
Az = Right(Val_DegAz_Sent, 3)
i = "A00" & Az & "#" & "0" & "5"
sent (i)
Do
dummy = DoEvents()
Loop Until form5.comm1.InBufferCount >= 12
Turn_Word = form5.comm1.Input
compare (Turn_Word)


        'SENT ELEVATION DEGREE
Val_DegEl_Step = textel.Text
Val_DegEl_Step = Val(Val_DegEl_Step)
Val_DegEl_Sent = Format(Val_DegEl_Step, "#0")
El = Right(Val_DegEl_Sent, 3)
i = "E00" & El & "#05"
sent (i)
Do
dummy = DoEvents()
Loop Until form5.comm1.InBufferCount >= 12
Turn_Word = form5.comm1.Input
compare (Turn_Word)
form5.comm1.PortOpen = False
End If
Form3.Hide
Form1.Enabled = True
End Sub
#########################
Private Sub BtnCancle_Click()
If form5.comm1.PortOpen = True Then
form5.comm1.PortOpen = False
Else
End If
Form3.Hide
Form1.Show
Form1.Enabled = True
End Sub
#########################
Private Sub comm1_OnComm()
    Select Case comm1.CommEvent
       ' Events
        Case comEvCTS    ' Change in the CTS line.
Case comEvDSR    ' Change in the DSR line.
    If comm1.CommEvent = 4 Then
       form2.sspanel1.BackColor = &HFF00&
       Else
       form2.sspanel1.BackColor = &HC0C0C
       Do
          MsgBox ("Can'n Connect Hardware"), , "Error"
          Loop While form2.sspanel1.BackColor = &HFF00&
    End If
        Case comEvReceive    ' Received RThreshold # of chars.
        Case comEvSend  ' There are SThreshold number of
                        ' characters in the transmit buffer.
    End Select
End Sub
```

```
########################
Global U As String
Global L As String
Global R As String
Global E As String
Global t3 As Integer
Global Stg_DefAz As String
Global Val_DefAz As Integer
Global Val_DefAz_Step As Integer
Global Val_DefAz_Sent As Integer
Global Stg_DefEl As String
Global Val_DefEl As Integer
Global Val_DefEl_Step As Integer
Global Stg_DefEl_Sent As String
Global Val_DegAz_Step As Integer
Global Val_DegEl_Step As Integer
Sub compare(Turn_Word)
First_Chr = Left(Turn_Word, 1)
Select Case First_Chr
'Case Is = "O"
'sent (i)
Case Is = "X"
My_Chr1 = Right(Turn_Word, 11)
Field_Chr = Left(My_Chr1, 3)
Field_Num = Right(Field_Chr, 2)
    My_Chr2 = Right(My_Chr1, 8)
El_Chr = Left(My_Chr2, 4)
El_Num = Right(El_Chr, 3)
Az_Chr = Right(My_Chr2, 4)
Az_Num = Right(Az_Chr, 3)

'display Az
Form4.Enabled = True
Form4.textaz.Text = Az_Num

'display El
Form4.Enabled = True
Form4.textel.Text = El_Num

'display s/n
Form4.Enabled = True
Form4.Textsn.Text = Field_Num
Do
Turn_Word = InputBox("") 'test
'Turn_Word = form5.comm1.Input
Loop Until Turn_Word = "C"
Exit Sub
End Select
End Sub
########################
Sub port()
form5.comm1.CommPort = 1
form5.comm1.Settings = "19200,N,8,1"
form5.comm1.InputLen = 12
form5.comm1.PortOpen = True
End Sub
########################
Sub sent(i)
form5.comm1.Output = i
End Sub
########################
Sub test()
form2.Show
i = "S" + "#" + "01"
```

```
sent (i)
Do
Income = form5.comm1.Input
Tst = Tst + 1
Loop Until Income = "S" Or Tst = 900
If Tst = 900 Then
MsgBox "Connection error PLEASE !! CHECK AND TRY AGAIN", 48, " ERROR
MESSAGE .."
test
Else
MsgBox "Welcome To Control System... ", 64, "MESSGE!!!"
End If
End Sub
########################
```

ภาคผนวก (ค)

1.0 INTRODUCTION TO THE 83C51KB

The 83C51KB is an application specific read-only-memory (ROM) microcontroller. The 83C51KB is designed as a keyboard scanner and is architecturally similar but not identical to the 80C51BH. Certain features not on the 80C51 have been added to support keyboard scanning. In order to streamline 83C51KB cost/performance factors, 80C51 peripherals not applicable to keyboard scanning are no longer present.

Refer to the *MCS® 51 Microcontroller Family User's Manual* (order number 272383) for a complete description of the 80C51. Figure 1 contains a basic memory map of the architecture. 83C51KB electrical information is contained in the datasheet (order number 272800).

1.1 Comparing the 80C51BH and 83C51KB

The differences between the 83C51KB and the 80C51 are briefly described here.

- The 83C51KB contains circuitry to support use of an RC clock oscillator. The 80C51 has a crystal clock oscillator.
- The 83C51KB has one 16-bit timer/counter (TH0, TL0). This timer is identical with Timer 0 on the 80C51. Other 80C51 timers are not present on the 83C51KB.
- The 83C51KB is designed to communicate with the keyboard controller on a personal computer (PC) through the CLK and DATA signals. The 80C51 has a serial port that is not present in the 83C51KB.
- 80C51 ROM security features are not present on the 83C51KB.
- The 80C51 reset operation is retained but the 83C51KB device also provides an automatic internal reset during power-up.
- Two new bits are implemented in the 83C51KB PCON register. One bit selects an interrupt for the KSI port, and the other controls the enhanced pullups for CLK and DATA signals.
- The 83C51KB has the same idle and power down modes as the 80C51. However, the 83C51KB differs in one respect. In the 83C51KB, either a hardware reset or an external interrupt can terminate the power down operation. The 80C51 is limited to a hardware reset termination of this mode.
- With the exception of port 2, some 80C51 port signals on the 83C51KB have been modified to facilitate keyboard scanning operation.
  - — All 83C51KB ports are now quasi-bidirectional outputs and require no external pullups. The 80C51 had an open-drain design on the port 0 outputs and needed external pullups.
  - — The 83C51KB port 1 circuitry contains new logic to generate an interrupt when enabled and any port 1 pin is pulled to V_{IL}. This feature supports the key-scan input (KSI) operation.
  - — Some 83C51KB port 3 signals are new. Enhanced pullups are available on two Port 3 signals to support the new CLK and DATA functions. Four port 3 outputs now allow direct LED connection.

This document presents a thorough description of the on-chip hardware features of the 83C51KB. It begins with a discussion of the individual signals and on-chip memory and then discusses each peripheral.

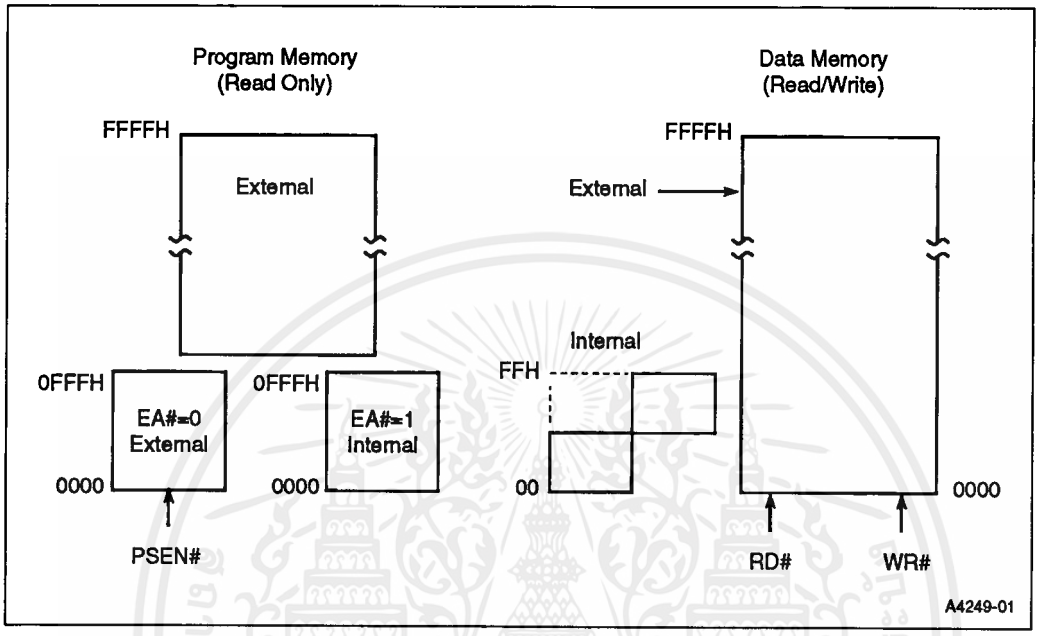


Figure 1. 83C51KB Memory Map

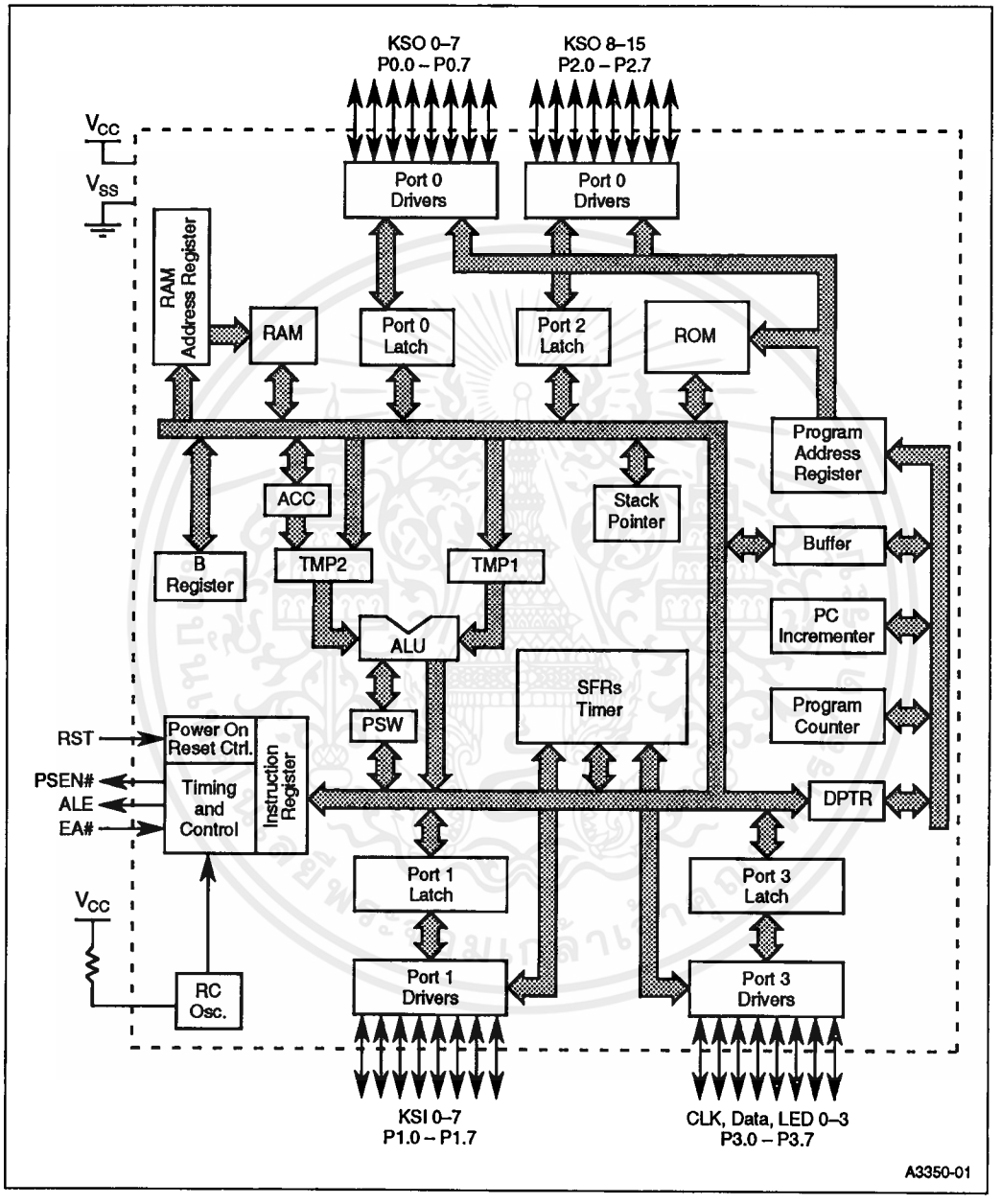


Figure 2. 83C51KB Block Diagram

2.0 SIGNAL SUMMARY

There are forty primary pin functions on the 83C51KB with some alternate functions.

The following tables provide a summary of these signals.

Table 1. 40-pin DIP Signals Arranged by Name

Keyboard	
Name	**Pin**
P0.7/KSO7/AD7	32
P0.6/KSO6/AD6	33
P0.5/KSO5/AD5	34
P0.4/KSO4/AD4	35
P0.3/KSO3/AD3	36
P0.2/KSO2/AD2	37
P0.1/KSO1/AD1	38
P0.0/KSO0/AD0	39
P2.7/KSO15/A15	28
P2.6/KSO14/A14	27
P2.5/KSO13/A13	26
P2.4/KSO12/A12	25
P2.3/KSO11/A11	24
P2.2KSO10/A10	23
P2.1/KSO9/A9	22
P2.0/KSO8/A8	21

Keyboard	
Name	**Pin**
P 1.0/KSI0	1
P1.1/KSI1	2
P1.2/KSI2	3
P1.3/KSI3	4
P1.4/KSI4	5
P1.5/KSI5	6
P1.6/KSI6	7
P1.7/KSI7	8
P3.0/DATA	10
P3.1	11
P3.2/CLK0/INT0#	12
P3.3/CLK1/INT1#	13
P3.4/LED0/T0	14
P3.5/LED1	15
P3.6/LED2/WR#	16
P3.7/LED3/RD#	17

Chip Control	
Name	**Pin**
RCIN	19
RST	9
ALE	30
PSEN#	29
EA#	31

Power & Ground	
Name	**Pin**
V_{CC}	40
V_{SS}	20

Table 2. 40-pin DIP Signals Arranged by Pin Number

Pin	Name	Pin	Name
1	P 1.0/KSI0	21	P2.0/KSO8/A8
2	P1.1/KSI1	22	P2.1/KSO9/A9
3	P1.2/KSI2	23	P2.2KSO10/A10
4	P1.3/KSI3	24	P2.3/KSO11/A11
5	P1.4/KSI4	25	P2.4/KSO12/A12
6	P1.5/KSI5	26	P2.5/KSO13/A13
7	P1.6/KSI6	27	P2.6/KSO14/A14
8	P1.7/KSI7	28	P2.7/KSO15/A15
9	RST	29	PSEN#
10	P3.0/DATA	30	ALE
11	P3.1	31	EA#
12	P3.2/CLK0/INT0#	32	P0.7/KSO7/AD7
13	P3.3/CLK1/INT1#	33	P0.6/KSO6/AD6
14	P3.4/LED0/T0	34	P0.5/KSO5/AD5
15	P3.5/LED1	35	P0.4/KSO4/AD4
16	P3.6/LED2/WR#	36	P0.3/KSO3/AD3
17	P3.7/LED3/RD#	37	P0.2/KSO2/AD2
18	NC	38	P0.1/KSO1/AD1
19	RCIN	39	P0.0/KSO0/AD0
20	V_{SS}	40	V_{CC}

Table 3. Signal Descriptions

Signal Name	Type	Description	Alternate Function
A15:8†	O	**Address Signals**. Upper address lines for the external bus. These signals are normally used for the KSO15:8 scan function and are not available for external memory access in a keyboard application. (See KSO signals).	KSO.15:8 P2.15:8
AD7:0†	I/O	**Address/Data Signals**. Multiplexed lower address and data signals for external memory. These signals are normally used for the KSO7:0 scan function and are not available for external memory access in a keyboard application. (See KSO)	KSO.7:0 P0.7:0
ALE†	O	**Address Latch Enable**. ALE signals the start of an external bus cycle and indicates that valid address information is available on lines A15:8 and AD7:0. Since these external address signals are normally used for the KSO scan function, the ALE should not be used for external memory access in a keyboard application. ALE can be disabled when not used for external memory access by setting bit 0 of SFR AUXR at address 8EH.	
CLK1:0 P3.3:2	I/O	**Clock signal**. Either P3.2 or P3.3 is configurable with a 1.8KΩ pullup and with external interrupt INT0# or INT1# and used as keyboard CLK signal.	INT1:0#
DATA P3.0	I/O	**DATA signal**. P3.0 is configurable with a 1.8KΩ pullup and used as a keyboard Data signal..	
EA#	I	**External Access**. Directs program memory accesses to on-chip or off-chip code memory. For EA# = 0, all program memory accesses are off-chip. EA# should always be strapped to V_{CC} for keyboard applications using the 83C51KB.	
INT1:0#†	I	**External Interrupts 0 and 1**. These inputs set bits IE1:0 in the TCON register. If bits IT1:0 in the TCON register are set, bits IE1:0 are set by a falling edge on INT1#/INT0#. If bits INT1:0 are clear, bits IE1:0 are set by a low level on INT1:0#. For keyboard applications, these signals are normally used for the CLK signals. (See KSIINT and CDPU bits in the PCON register)	CLK1:0 P3.3:2
KSI7:0 P1.7:0	I/O	**Keyboard Scan Inputs**. Application specific keyboard signals.	
KSO15:0 P2.15:8 P0.7:0	I/O	**Keyboard Scan Outputs**. The KSO signals are application specific to keyboard scan functions.	
LED3:0 P3.7:4	I/O	**Light Emitting Diode Drivers**. The LED signals are specifically designed to drive LEDs connected to Vcc directly (see D.C. Characteristics). The alternate functions are not available for keyboard applications.	RD#, WR#, T0
N/C	—	**No Connection Signal**. This signal is to be unconnected.	
P0.7:0†	I/O	**Port 0**. This is an 8-bit quasi-bidirectional I/O port (see KSO signals, see also AD7:0).	AD7:0
P1.7:0	I/O	**Port 1**. This is an 8-bit quasi-bidirectional I/O port (see KSI signals).	
P2.7:0	I/O	**Port 2**. This is an 8-bit quasi-bidirectional I/O port (see also A15:8).	A15:8
P3.7:0	I/O	**Port 3**. This is an 8-bit quasi-bidirectional I/O port (see CLK1:0, DATA, LED3:0).	

† The descriptions of RD#, WR#, ALE, PSEN#, A15:8/P2.7:0 and AD7:0/P0.7:0 are documented for the standard MCS 51 microcontrollers. They are not used for these functions in keyboard applications.

Table 3. Signal Descriptions (Continued)

Signal Name	Type	Description	Alternate Function
PSEN#†	O	**Program Store Enable.** This output is asserted for external program memory fetch operations. It is not available for keyboard applications.	—
RCIN	I	**Resonant Clock Input.** The internal RC signal is generated by connecting a resistor to V_{CC} or provide an external clock input from an external clock device (see Figure).	
RD#†	O	**Read.** Read signal output for external data memory fetch operations. It is not available for keyboard applications.	LED3
RST	I	**Reset.** Asserting RST when the chip is in idle mode or powerdown mode returns the chip to normal operation. This signal is input only.	—
V_{CC}	PWR	**Supply Voltage.** Connect this pin to the +5V supply voltage.	—
V_{SS}	GND	**Circuit Ground.** Connect this pin to ground.	—
WR#†	O	**Write.** Write signal output for external memory write operations. It is not available for keyboard applications.	LED2

† The descriptions of RD#, WR#, ALE, PSEN#, A15:8/P2.7:0 and AD7:0/P0.7:0 are documented for the standard MCS 51 microcontrollers. They are not used for these functions in keyboard applications.

3.0 MEMORY ORGANIZATION

All MCS-51 devices have two primary memory types; program (instruction code) and data memory. Up to 64 Kbytes each of program and data memory may be addressed. Each memory type is a combination of on-chip and off-chip memory. The 83C51KB contains 4 Kbytes of on-chip program memory. In addition to external data memory, the 83C51KB also consists of 128 bytes of on-chip data RAM and 128 bytes of special function registers (SFRs).

3.1 Program Memory

If the EA# pin is connected to VSS, all program (instruction code) fetches are directed to an external 64K-byte memory system (0000H to FFFFH). During external instruction fetch cycles the PSEN# signal is activated to enable program code output from the external memory system. If the EA# pin is connected to VCC, then program fetches to addresses 0000H through 0FFFH are directed to internal ROM and fetches to addresses 1000H through FFFFH are to the external memory system. After the reset operation completes, the 83C51KB execution unit fetches and executes the instruction code located at the reset vector address (0000H).

3.2 Data Memory

Up to 64K bytes of data RAM may be designed into the external memory system. The external data memory is accessed with indirect address instructions (i.e. MOVX). Using these indirect instructions activates the RD# or WR# signal for external data transfer operations.

3.3 Using External Memory

When the 83C51KB execution unit needs external program or data memory, all port 0 pins are used for low-order address signals. The bus interface unit contains an address/data multiplexer and also uses port 0 for data transfer. All port 2 pins are used for high-order address output (except when MOVX @Ri is used). These signals may not be used for general purpose input/output under these conditions.

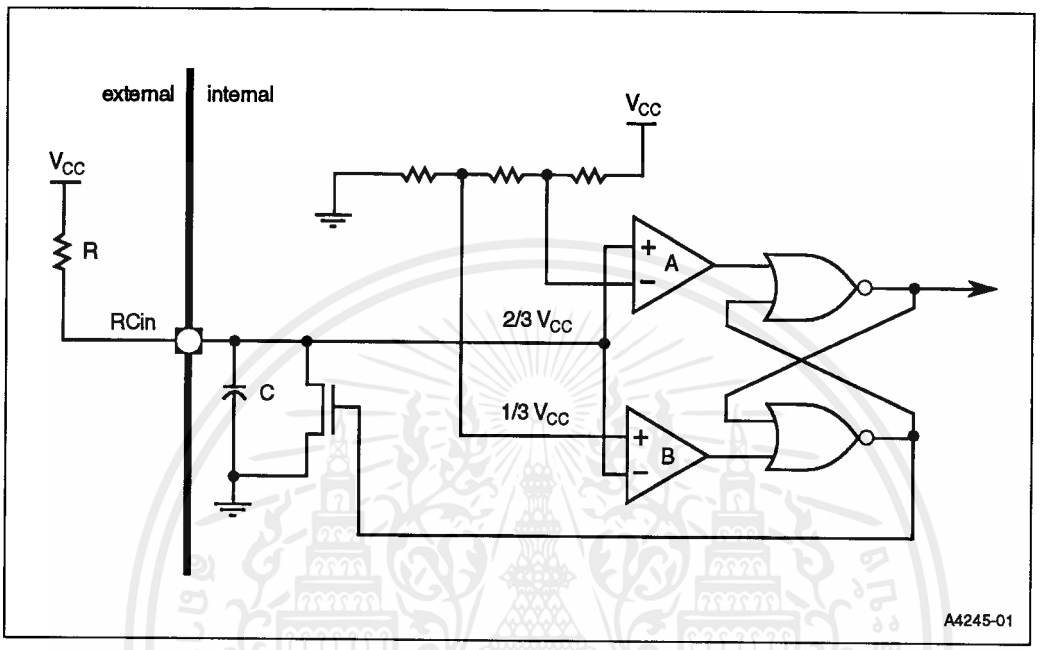
3.4 On-Chip Special Function Registers (SFRs) and RAM

The 83C51KB implements 128 bytes of on-chip RAM (00H to 7FH). This is accessible using direct or indirect addressing. There are four 8-byte general purpose register banks (address 00H to 1FH) and 16 bytes of bit addressable data memory

(address 20H to 2FH). Special function registers are located from 80H to FFH. SFRs can only be accessed with direct address instructions. The shaded SFR addresses in Table 4 are reserved.

Table 4. Special Function Register Map

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8									FF
F0	B 00000000								F7
E8									EF
E0	ACC 00000000								E7
D8									DF
D0	PSW 00000000								D7
C8									CF
C0									C7
B8	IP xxxx0000								BF
B0	P3 11111111							IPH xxxx0000	B7
A8	IE 0xxx0000								AF
A0	P2 11111111								A7
98									9F
90	P1 11111111								97
88	TCON 00000000	TMOD xxxx0000	TL0 00000000		TH0 00000000		AUXR xxxxxxx0		8F
80	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 00x00000	87
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	



RC Oscillator

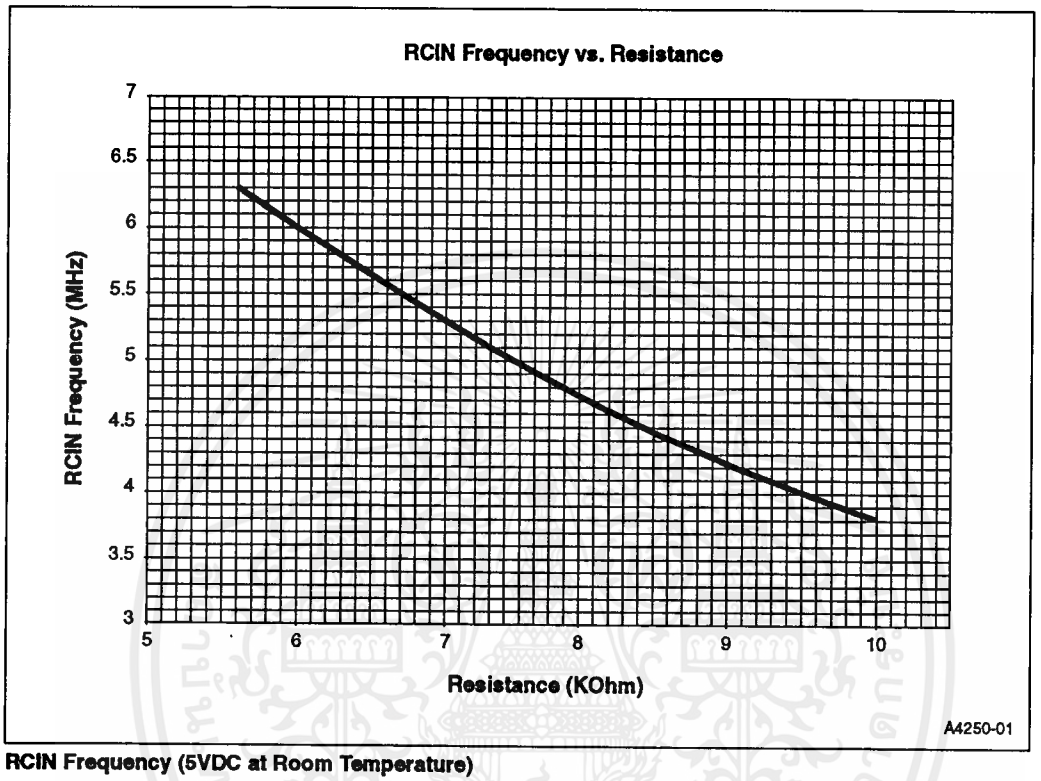
4.0 RC OSCILLATOR

The 83C51KB contains an RC oscillator. The timing is set by a single external resistor connected between the RCIN pin and V_{CC}. The charge time of the circuit is illustrated in formula. The oscillator circuit is shown in Figure .

The circuit operates as follows. Capacitor C charges through the external resistor R, until its voltage reaches 2/3 V_{CC}. At that point comparator A switches the NOR-gate latch, turning on the pulldown transistor.

Then C discharges through this transistor until its voltage falls to 1/3 V_{CC}. At that point comparator B switches the NOR-gate latch, turning off the pulldown transistor, and the cycle repeats. The time it takes the capacitor to charge from 1/3 V_{CC} to 2/3 V_{CC} is:

$$T\ (Charge) = RC \times \ln 2$$



RCIN Frequency (5VDC at Room Temperature)

The circuit is designed to discharge the capacitor very quickly and therefore the corresponding charge time is the primary parameter in the oscillation period. Charge time is controlled with an external resistor. A ±5% variation of the RC resonator frequency is achievable with a 1% precision external resistor (assuming stable VCC and temperature). Reduced resistor precision results in a wider variation of frequency. Detailed specifications of oscillation frequency as a function of external resistance values are presented in the datasheet. A generalized chart is presented in Figure .

The RCIN pin can also be driven with an external oscillator. The external circuit will have to meet the I_{IH} spec for this pin. The I_{IH} spec for this pin is higher than for others, because when the RCIN pin is at a logic high the pulldown FET is on.

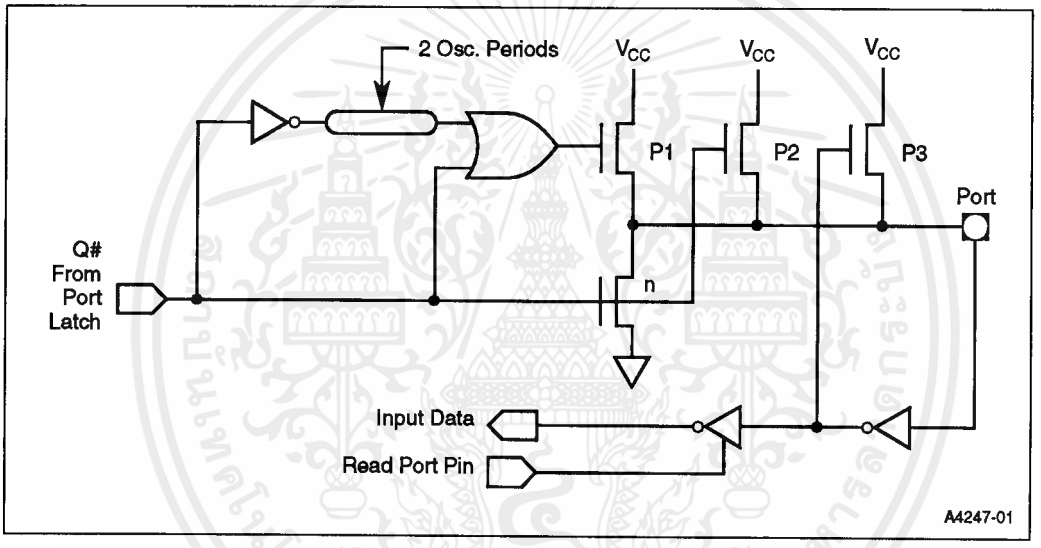
5.0 83C51KB PORT DESIGN

All four input/output ports are quasi-bidirectional structures. If the data change on a port requires a 0-to-1 transition, an additional pullup is turned on. This is done to increase the transition speed. The extra pullup can source about 100 times normal current levels. The internal pullups are field-effect transistors, not linear resistors. The pull-up arrangement is shown in Figure 5.1 and described in the following paragraphs.

The pullup consists of three p-channel (pFET) transistor devices and the pulldown consists of one n-channel (nFET) device. Note that the nFET is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. Conversely the pFET is on when the gate sees a 0, and off when the gate is at 1. The device labeled P1 in Figure 5.1 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. A 1 at the port pin turns on P3 (a weak pull-

up), through the inverter. This inverter and pFET form a latch which hold the 1.

If the pin is emitting a 1, a negative glitch on the pin from some external source can turn off P3, causing the pin to go into a float state. P2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about 1/10 the strength of P3. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch. P1 is turned on for 2 oscillator periods after Q makes a 0-to-1 transition. During this time, P1 also turns on P3 through the inverter to form a latch which holds the 1. P2 is also on. Port 2 is similar except that it holds the strong pullup on while emitting 1s that are address bits.The output buffers may each sink 3.2 mA at 0.45 VDC. These port pins can be driven by open-collector and open-drain outputs although 0-to-1 transitions will not be fast since there is little current pulling the pin up. An input 0 turns off pullup P3, leaving only the very weak pullup P2 to drive the transition. With the exception of port 2, some 80C51 port signals on the 83C51KB have been modified to facilitate keyboard scanning operation.



5.1 Port Pullup Configuration Port 1

Port 1 is the same on the 83C51KB as on the 80C51, except that logic has been added to generate an interrupt when any of the Port 1 pins is pulled low, as shown in Figure 6. The interrupt can be directed to either INT0# or INT1#, depending on the state of the KSIINT bit, bit 6 in the PCON register. KSIINT = 0 directs the interrupt to INT0#. KSIINT = 1 directs it to INT1#.

NOTE: KSIINT = 0 disconnects INT0# from the P3.2 pin, and KSIINT = 1 disconnects INT1# from the P3.3 pin.

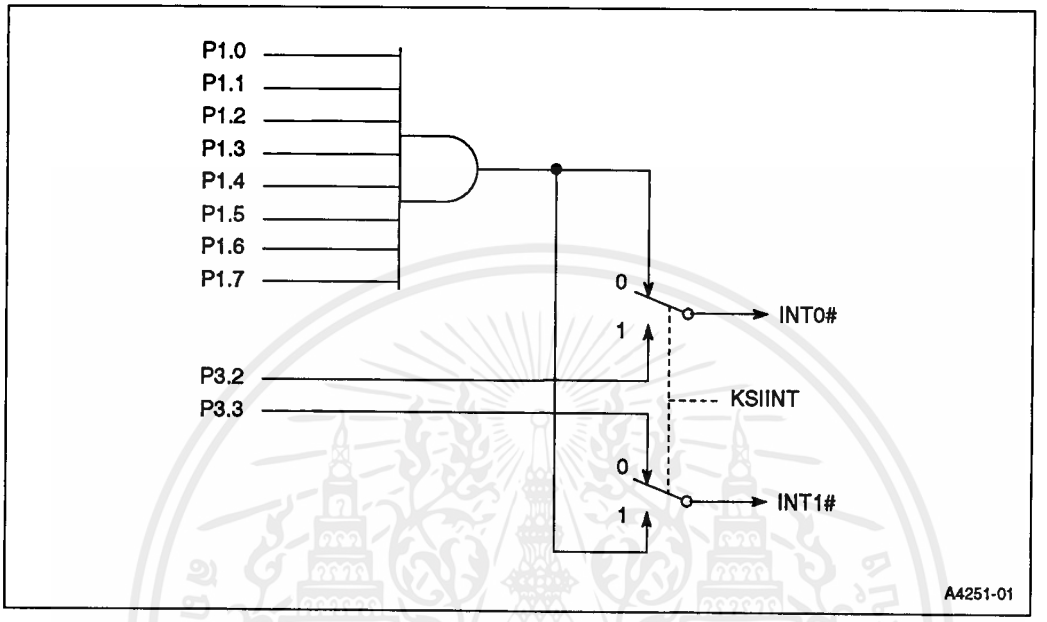


Figure 3. Port 1 and Port 3 KSIINT Structure

5.2 Port 3

As the 83C51KB has no UART or Timer 1, the port 3 alternate functions associated with these peripherals are not applicable.

Communication with the keyboard controller on the PC motherboard is accomplished with the DATA and CLK signals. The port 3.0 signal and either port 3.2 or port 3.3 are used as the DATA and CLK signals. In keyboard applications the enhanced strength pullup device is enabled on the port by setting the CDPU bit (PCON.7).

When the KSI port uses INT0# (e.g. KSIINT = 0), then the enhanced pullups need to be enabled on port 3.0 and port 3.3 (INT1#). The port 3.0 pin is then used for the DATA signal and the port 3.3 (INT1#) pin is used for the CLK signal.

When the KSI port uses INT1# (e.g. KSIINT = 1), then the enhanced pullups are enabled on port 3.0 and port 3.2 (INT0#). The port 3.0 pin is used for DATA and the port 3.2 (INT0#) pin is used for CLK. The combined effects of the CDPU and KSIINT bits are shown in Table 5.

Table 5. CDPU and KSIINT Affects on Port 3

PCON.7	PCON.6	P3.0	P3.2	P3.3	Keyboard Functions	
CDPU	KSIINT	pullup	pullup	pullup	DATA	CLK
0	X	normal	normal	normal		
1	0	enhanced	normal	enhanced	P3.0	P3.3
1	1	enhanced	enhanced	normal	P3.0	P3.2

This arrangement allows either external interrupt to be used for the KSI port, and leaves the other external interrupt, with an enhanced pullup, available for CLK. The reset values of CDPU and KSIINT are both zero (see Figure 5).

Four outputs of port 3 (see Figure 4) have pulldown devices sized to limit their drain currents to a safe maximum on each pin when used to drive LEDs. The protected pins are port 3.7:4. Each output can drive an LED directly, as shown in Figure 4. Writing a one to the port pin turns the LED off, and writing a zero turns the LED on. When on, the LED voltage drop is about 2 VDC and raises V_{OL} to approximately 3 VDC. The drain current of the pulldown transistor is limited by the I_D / V_{DS} device characteristic.

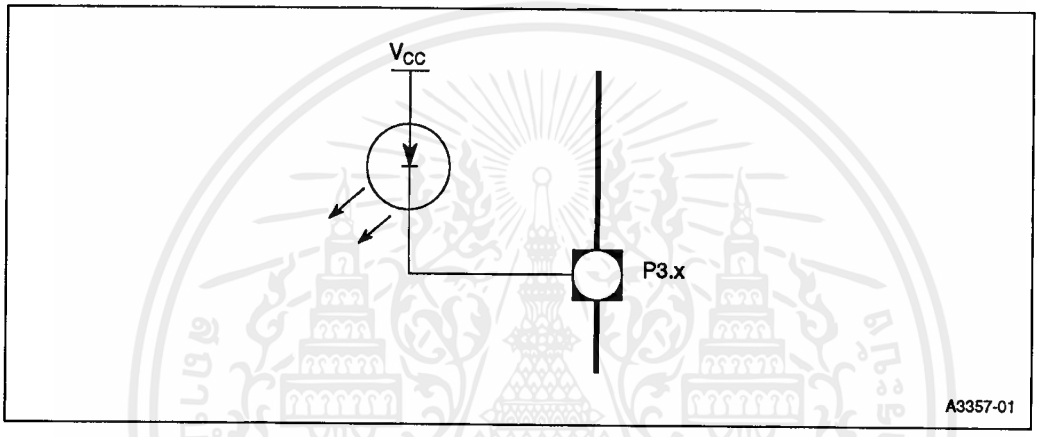


Figure 4. Driving an LED with P3.7:4

6.0 POWER-ON RESET

A special feature of the 83C51KB makes it possible to effect a power-on reset with no external components connected to the reset (RST) pin. When power is applied to the chip, the internal reset signal remains high for approximately 80ms to 260ms (see the datasheet for current specifications). During operation, should VCC fall below 3VDC and then return to normal operating levels, a seperate internal reset will be generated. The internal reset signal does not appear as an output on the RST pin, and the RST pin retains its full functionality. Additional reset time can be accomplished with an external capacitor connected between RST and VCC.

7.0 IDLE AND POWER DOWN

The Idle and power down modes are identical to the 80C51 with one exception. The 83C51KB power down can be terminated by either of the external interrupt request pins (INT0# or INT1#). Both the global interrupt bit and the individual interrupt must be enabled in the interrupt enable register (see Figure 8). The power control bits in the PCON register are located in the special function register area (address 87H, see Figure 5).

An instruction that sets the power down (PD) bit at PCON.1 is the last instruction executed prior to the power down sequence. In this mode the RC oscillator is stopped. With no internal clock signal, all chip functions are halted. The on-chip RAM and special function registers continue to retain their data. The port pins also continue to drive the values held by their respective SFRs. The ALE and PSEN# signals are driven low.

The VCC power supply can be reduced as low as 2VDC. However, lowering VCC below 3VDC and then raising it again generates an automatic power-on reset (see section 6.0 Power-On Reset).

The 83C51KB can exit power down mode with either a reset or an external interrupt request. The external interrupt (INT0# or INT1#) must be enabled and configured in the level-sensitive mode. Driving

the interrupt signal low restarts the oscillator. Driving the interrupt signal high completes the exit from power down. After the RETI instruction is executed in the interrupt service routine (ISR), the next instruction to be executed follows the instruction that set the PCON.1 (PD) bit.

PCON Address: 87H
Reset State: 00X0 0000B

Power Control Register. Contains the power off flag (POF) and bits for enabling the idle and powerdown modes. Also contains two general-purpose flags and two bits that control serial I/O functions—the double baud rate bit and a bit that selects whether accesses to SCON.7 are to the FE bit or the SM0 bit.

7							0
CDPU	KSIINT	—	POF	GF1	GF0	PD	IDL

Bit Number	Bit Mnemonic	Function
7	CDPU	Setting this bit enhances the pullup strength on port 3 keyboard signals. This bit works in concert with KSIINT (PCON.6). Refer to Table 5.
6	KSIINT	Setting this bit enhances the pullup strength on port 3 keyboard signals. This bit works in concert with CDPU (PCON.7). Refer to Table 5.
5	—	Reserved.
4	POF	Power Off Flag. Set by hardware on rising edge of V_{CC}. Set or cleared by software. Detects resets caused by power failures. V_{CC} must remain above 3 VDC to retain the flag value.
3	GF1	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
2	GF0	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
1	PD	Powerdown Mode Bit: When set, activates powerdown mode. Cleared by hardware when an interrupt or reset occurs.
0	IDL	Idle Mode Bit: When set, activates idle mode. Cleared by hardware when an interrupt or reset occurs. If IDL and PD are both set, PD takes precedence.

Figure 5. Power Control Register

8.0 TIMER 0

The 83C51KB has one 16-bit timer/counter (Timer 0). The timer consists of two 8-bit registers, TH0 and TL0. This peripheral circuit can be configured to operate either as a timer or event counter.

In the timer function, the peripheral register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the counter function, the register is incremented in response to a negative transition on a corresponding external input pin; in this case T0 configured as an alternate function to the port 3.4 pin. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

The Timer or Counter function is selected by control bits C/T in the Special Function Register TMOD (Figure 6). The timer/counter has four operating modes (Mode 0-3) selected by bit-pairs (M1, M0) in TMOD.

TMOD Address: 89H
Reset State: XXXX 0000B

7							0
—	—	—	—	GATE	C/T#	M1	M0

Bit Number	Bit Mnemonic	Function
7-4	—	Reserved.
3	GATE	Timer 0 Gate: When GATE = 0, run control bit TR0 gates the input signal to the timer register. When GATE = 1 and TR0 = 1, external signal INT0# gates the timer input.
2	C/T#	Timer 0 Counter/Timer Select: C/T# = 0 selects timer operation: timer 0 counts the divided-down system clock. C/T# = 1 selects counter operation: timer 0 counts negative transitions on external pin T0.
1, 0	M1, M0	Timer 0 Mode Select: **M1 M0** 0 0 Mode 0: 8-bit timer/counter (TH0) with 5-bit prescaler (TL0) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow 1 1 Mode 3: TL0 is 8-bit timer/counter. TH0 is an 8-bit timer.

Figure 6. TMOD: Timer/Counter Mode Register

8.1 Mode 0

Timer 0 in Mode 0 is an 8-bit counter with a divide-by-32 prescaler. In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TF0. The counted input is enabled to the timer when TR0 = 1 and either GATE = 0 or INT0# = 1. (Setting GATE = 1 allows the timer to be controlled by external input port 3.2 (see section 9.3 Timer Interrupts).

TR0 and TF0 are control bits in the TCON register (see Figure 7). The GATE bit is in the TMOD register (TMOD.3). The 13-bit register consists of all 8 bits of TH0 and the lower 5 bits of TL0. The upper 3 bits of TL0 are indeterminate and should be ignored. Setting the run flag (TR0) does not clear these registers.

8.2 Mode 1

Mode 1 is the same as Mode 0, except that the timer register uses all 16 bits. In this mode, TH0 and TL0 are cascaded without prescaler functionality.

8.3 Mode 2

Mode 2 configures the timer register as an 8-bit Counter (TL0) with automatic reload. Overflow from TL0 not only sets TF0, but also reloads TL0 with the contents of TH0, preset by software. The reload leaves TH0 unchanged.

8.4 Mode 3

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles). Mode 3 is provided for applications requiring an extra 8-bit timer or counter. When enabled in mode 3 the application software sets the TR1 bit to operate the TH0 register. The TF1 flag is set by hardware when the TH0 register overflows. The TF1 flag is cleared by hardware during the vector to the service routine. TH0 can be halted at any time by using application software to clear the TR1 bit.

TCON Address: 88H
Reset State: 0000 0000B

7							0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Function
7	TF1	TH0 Overflow Flag (Mode 3 only) Set by hardware when TH0 overflows. Cleared by hardware when the processor vectors to the interrupt routine.
6	TR1	TH0 Run Control Bit (Mode 3 only) Set and cleared by software to switch TH0 on or off.
5	TF0	Timer 0 Overflow Flag: Set by hardware when TH0 overflows. Cleared by hardware when the processor vectors to the interrupt routine.
4	TR0	Timer 0 Run Control Bit: Set and cleared by software to turn timer 0 on or off.
3	IE1	Interrupt 1 Flag for INT1#: Set by hardware when an INT1# event is detected. Edge- or level-triggered (see IT1). Cleared by interrupt vector if edge-triggered.
2	IT1	Interrupt 1 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 1. Clear this bit to select level-triggered (active-low).
1	IE0	Interrupt 1 Flag for INT0#: Set by hardware when an INT0# event is detected. Edge- or level-triggered (see IT0). Cleared by interrupt vector if edge-triggered.
0	IT0	Interrupt 0 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 0. Clear this bit to select level-triggered (active-low).

Figure 7. TCON: Timer/Counter Control Register

9.0 INTERRUPT SYSTEM

The 83C51KB, like other control-oriented computer architectures, employs a program interrupt method. This operation branches to a subroutine and performs some service in response to the interrupt. When the subroutine completes, execution resumes at the point where the interrupt occurred. 83C51KB interrupts may occur as a result of internal 83C51KB activity (e.g., a timer overflow) or at the initiation of electrical signals external to the microcontroller (e.g. an external interrupt). In all cases, interrupt operation is programmed by the system designer, who determines the priority of interrupt service relative to normal code execution and other interrupt service routines. The interrupts are enabled or disabled by the system designer and may be manipulated dynamically.

Table 6. Interrupt Description of INT1:0

Signal Name	Type	Description	Alternate Function
INT1:0#	I	**External Interrupts 0 and 1**. These inputs set bits IE1:0 in the TCON register. If bits IT1:0 in the TCON register are set, bits IE1:0 are set by a falling edge on INT1#/INT0#. If bits INT1:0 are clear, bits IE1:0 are set by a low level on INT1:0#. For keyboard applications, these signals are normally used for the CLK signals. (See KSIINT and CDPU bits in the PCON register)	CLK1:0 P3.3:2

9.1 Interrupt Sources

The 83C51KB has four maskable interrupt sources; the maskable sources include two external interrupts (INT0# and INT1#) and two timer interrupts associated with timer 0 operation. Each interrupt has an interrupt request flag, set by software as well as by hardware. For some interrupts, hardware clears the request flag when the CPU grants service to an interrupt. Software can clear any request flag to cancel a pending interrupt.

9.2 External Interrupts

External interrupts INT0# and INT1# (INTx#) pins may each be programmed to be level-triggered or edge-triggered, dependent upon bits IT0 and IT1 in the TCON register (see Figure 7). If ITx = 0, INTx# is triggered by a detected low at the pin. If ITx = 1, INTx# is negative-edge triggered.

External interrupts are enabled with bits EX0 and EX1 in the IE register (see Figure 8). Events on the external interrupt pins set the interrupt request flags IEx in TCON. These request bits are cleared by hardware vectors to service routines only if the interrupt is negative-edge triggered. If the interrupt is level-triggered, the interrupt source controls the interrrupt flag. The source must hold the request active until acknowledged in some manner by the ISR. After acknowledgment, the source must deassert INTx# before the service routine completes, or an additional interrupt is requested. External interrupt pins are sampled once every machine cycle. A level-triggered interrupt pin held low or high for at least a six-state time period guarantees detection. Edge-triggered external interrupts must hold the request pin high for at least 6 state times and low for at least six state times. This ensures edge recognition and sets interrupt request flag.

9.3 Timer Interrupts

Timer interrupts are enabled by ET0 in the IE register (see Figure 8). The timer-interrupt request flag TF0 is set in the TCON register by the timer 0 overflow signal (see Figure 7). The exception is Mode 3 where the interrupt is instead enabled by ET1 and the associated overflow sets the TF1 flag in TCON. When a timer interrupt is generated, the flag is cleared by an on-chip-hardware vector to an interrupt service routine.

IE Address: A8H
Reset State: 0XXX 0000B

7							0
EA	—	—	—	ET1	EX1	ET0	EXO

Bit Number	Bit Mnemonic	Function
7	EA	Global Interrupt Enable: Setting this bit enables all interrupts enabled by bits 0-2 in this register.
6:4	—	Reserved
3	ET1	TH0 Overflow Interrupt Enable (Mode 3 only).
2	EX1	External Interrupt 1 Enable: Setting this bit enables external interrupt 1.
1	ET0	Timer 0 Overflow Interrupt Enable: Setting this bit enables the timer 0 overflow interrupt.
0	EX0	External Interrupt 0 Enable: Setting this bit enables external interrupt 0.

Figure 8. Interrupt Enable Register

IP Address: B8H
Reset State: XXXX 0000B

7							0
—	—	—	—	PT1	PX1	PT0	PX0

Bit Number	Bit Mnemonic	Function
7:4	—	Reserved. The value read from this bit is indeterminate. Do not write a "1" to this bit.
3	PT1	TH0 Overflow Interrupt Priority Bit (Mode 3 only)
2	PX1	External Interrupt 1 Priority Bit
1	PT0	Timer 0 Overflow Interrupt Priority Bit
0	PX0	External Interrupt 0 Priority Bit Low

Figure 9. Interrupt Priority Register

9.4 Interrupt Enable

Each interrupt source is individually enabled or disabled with the appropriate interrupt enable bit in the IE register at A8H (see Figure 8). Note IE also contains a global disable bit (EA). If EA is set, interrupts are individually enabled or disabled by other bits in IE. If EA is clear, all interrupts are disabled.

9.5 Interrupt Priorities

Each of the 83C51KB interrupt sources may be individually programmed to one of four priority levels. This is accomplished with the IP.x and IPH.x bits in the interrupt priority (IP and IPH) registers (Figure 9).

Specify the priority level as shown in Table 8.A low-priority interrupt is always interrupted by a higher priority interrupt but not by another interrupt of equal or lower priority. The highest priority interrupt is not interrupted by any other interrupt source.

Higher priority interrupts are serviced before lower priority interrupts. The response to simultaneous occurrence of equal priority interrupts (i.e., sampled within the same interrupt request cycle) is determined by a hardware priority-within-level resolver (see Table 7).

Table 7. Interrupt Priority Within Level

Priority Number	Interrupt Name
1(Highest Priority)	INT0#
2	Timer 0
3	INT1#
4	TH0 (Mode 3 only)

Table 8. Level of Priority

IP Value	IPH Value	Priority Level
0	0	Level 0 (Low Priority)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (High Priority)

IPH

Address: B7H
Reset State: XXXX 0000B

7							0
—	—	—	—	PT1H	PX1H	PT0H	PX0H

Bit Number	Bit Mnemonic	Function
7:4	—	Reserved
3	PT1H	TH0 Interrupt Priority high bit (Mode 3 only)
2	PX1H	INT1# Priority high bit
1	PT0H	Timer 0 Interrupt Priority high bit
0	PX0H	INT0# Priority high bit

Table 9. Interrupt Priority High Register

9.6 Interrupt Processing

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

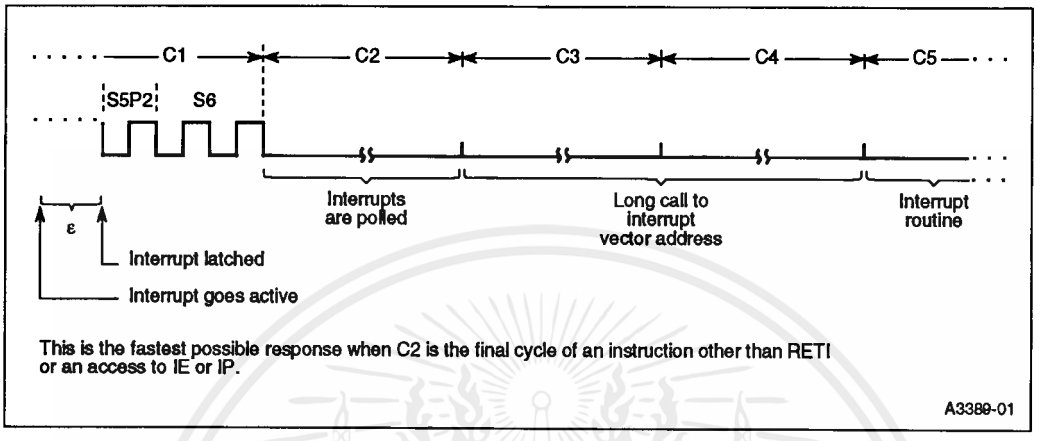
1. An interrupt of equal or higher priority level is already in progress.

2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.

3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to. The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. If the interrupt flag for a level-sensitive external interrupt is active but not being responded to for one of the above conditions and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The interrupt vector cycle/LCALL sequence is illustrated in Figure .



Interrupt Vector to LCALL

NOTE

Note that if an interrupt of a higher priority level goes active prior to S5P2 of the first machine cycle labeled C3 in Figure , then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

9.7 Interrupt Service Routine Cycle

When interrupt service is granted, the CPU breaks the instruction stream sequence and pushes program counter (PC) information onto the stack. The CPU then reloads the PC with a start address for the appropriate ISR.

The starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine.

The ISR executes until the RETI instruction is encountered. RETI informs the processor that the ISR is no longer in progress. Two bytes are popped from the stack and reloaded into the PC. Execution of the interrupted program continues from where it left off.

Table 10. Interrupt Vector Addresses

Interrupt Source	TCON Request Flag	Hardware Clear	Vector Address
INT0#	IE0	No (level) Yes (edge)	0003H
Timer 0	TF0	Yes	000BH
INT1#	IE1	No (level) Yes (edge)	0013H
TH0 Overflow (Mode 3 only)	TF1	Yes	001BH

9.8 Response Time

The INT0 and INT1 levels are inverted and latched into the Interrupt Flags IE0 and IE1 at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapses between activation of an external interrupt request and the beginning of execution of the service routine's first instruction. Figure illustrates interrupt response timing.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or write to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one or more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

10.0 ONCE MODE

The ONCE (ON-Circuit Emulation) mode facilitates testing and debugging of systems using the 83C51KB without having to remove the device from the circuit.

The ONCE mode is invoked by:

1. 1. Pulling ALE low while the device is in reset and PSEN is high;
2. 2. Holding ALE low as RST is deactivated.

While the device is in ONCE mode, the port pins, ALE, and PSEN are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a valid reset is applied.

11.0 ADDITIONAL REFERENCES

The following application notes provide supplemental information to this document and can be found in the two volumes of the Embedded Applications handbook (order number 270648).

1. AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments"
2. AP-155 "Oscillators for Microcontrollers"
3. AP-252 "Designing with the 80C51BH"

บรรณานุกรม

Mark Long And Jeffrey Keating **The World Of Satellite TV** Bangkok : CQ International Co.,Ltd., 1993

นิตยสารวิทยาศาสตร์ ปีที่ 2 ฉบับที่ 13 พฤษภาคม 2540 กรุงเทพมหานคร : มติชน, 2540

การใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51 : สมยศ จุณณะปิยะ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ARCHITECTURAL OVERVIEW OF THE MHS C51 FAMILY : MHS