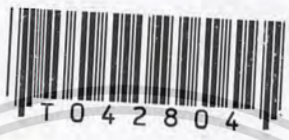
ชุดพัฒนาหุ่นยนต์ขนาดเล็ก

Micro Robot Developer Kit



นายวิทยา ศุภวรางกูล

นายเอนก กอธนสาร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ชุดพัฒนาหุ่นยนต์ขนาดเล็ก

Micro Robot Developer Kit

โดย

นายวิทยา ศุภวรางกูล

นายเอนก กอธนสาร

อาจารย์ที่ปรึกษา

รศ.สมศักดิ์ มิตะถา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดพัฒนาหุ่นยนต์ขนาดเล็ก

MICRO ROBOT DEVELOPER KIT

ผู้จัดทำ

1. นายวิทยา ศุภวรางกูล รหัสประจำตัว 40010728
2. นายเอนก กอธนสาร รหัสประจำตัว 40011051

_____ อาจารย์ที่ปรึกษา

(รศ.สมศักดิ์ มิตะถา)

ชุดพัฒนาหุ่นยนต์ขนาดเล็ก

นาย วิทยา ศุภวรางกูล 40010728
นาย เอนก กอธนสาร 40011051
รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอ ชุดพัฒนาหุ่นยนต์ขนาดเล็กซึ่งประกอบไปด้วยชิ้นส่วนสำเร็จรูปที่สามารถถอดประกอบกันได้อย่างอิสระ สำหรับการสร้างหุ่นยนต์ที่มีความซับซ้อนไม่มาก

โมดูลภายในชุดพัฒนาหุ่นยนต์ แบ่งเป็น 2 ประเภท คือ โมดูลควบคุมหลัก ทำหน้าที่ควบคุมการทำงานของหุ่นยนต์โดยการส่งคำสั่งให้โมดูลควบคุมอินพุต/เอาต์พุต แล้ว โมดูลอินพุต/เอาต์พุตจะควบคุมอุปกรณ์ตามที่โมดูลควบคุมหลักต้องการ ส่วน โมดูลอินพุต/เอาต์พุต ทำหน้าที่รับคำสั่งจากโมดูลควบคุมหลัก แล้ว ควบคุมการทำงานของอินพุต/เอาต์พุตตามคำสั่งที่รับมา ซึ่งในโครงงานนี้มีโมดูลอินพุต/เอาต์พุต ทั้งหมด 5 ประเภท ได้แก่ โมดูลขับเคลื่อนสำหรับควบคุมการเคลื่อนที่ของหุ่นยนต์ทั้งแบบที่ใช้สเตปปิ้งมอเตอร์ และ เซอร์โวมอเตอร์ , โมดูลตรวจจับด้วยอินฟราเรด สำหรับตรวจจับพื้นผิวที่มีระยะห่างไม่มาก , โมดูลตรวจจับด้วยอัลตราโซนิค สำหรับตรวจจับพื้นผิวในระยะห่าง และ โมดูลติดต่อกับผู้ใช้ ซึ่งรับข้อมูลจากผู้ใช้ผ่านทางคีย์แพด 12 คีย์ และ แสดงผลผ่านทางจอ LCD โดยโมดูลทั้งหมดสามารถเชื่อมต่อเข้ากับบัสของระบบซึ่งใช้การรับ/ส่ง ข้อมูลหรือ คำสั่งผ่านบัส I2C เพื่อประกอบเป็นหุ่นยนต์ที่ต้องการได้

การทดสอบการทำงานทำโดยการสร้างหุ่นยนต์จากชุดพัฒนา 2 แบบคือ หุ่นยนต์หลบสิ่งกีดขวางที่สามารถโปรแกรมได้ซึ่งเป็นหุ่นยนต์สามารถวิ่งตามรูปแบบที่ผู้ใช้โปรแกรมเข้าไป หากพบสิ่งกีดขวาง ก็สามารถหลบหลีกแล้วเดินตามรูปแบบที่โปรแกรมต่อไปได้ และ หุ่นยนต์เดินตามเส้นซึ่งเดินตามเส้นสีดำบนพื้นสีขาว ซึ่งหุ่นยนต์เหล่านี้อาจเอาไปประยุกต์ใช้โดยการสร้างโมดูลดูดฝุ่นมาประกอบเข้าไป เพื่อใช้เป็นหุ่นยนต์ดูดฝุ่นในบ้านได้

Micro Robot Developer Kit

Wittaya Supawarangkul
Anek Korthanasarn
Assoc.Prof.Somsak Mitatha Advisor

Abstract

This thesis is presented micro robot developer kit that consists of processed modules. These modules can join each other independently to make uncomplicated micro robot.

These modules are control module and input/output module. The control module control micro robot by sends the instructions to input/output module. In this thesis we have five kinds of input/output modules: moving modules (stepping motor and servo motor), infrared module (for short distance), ultrasonic module (for long distance), and user interface module (receives instruction from user via keypad and displays on LCD). These modules send and receive instructions from each other via I2C bus system.

We tested this developer kit to make up two kinds of micro robot. The first one, user can program the way it moves when it meets hindrances. The second one and move along black line on white floor, so it can apply to be vacuum cleaner robot.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและความร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ด้วยดีก็คือ อาจารย์สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก และพี่ต๋อง พี่เรศ พี่อ๊อด พี่วุธ พี่จุ๋ม พี่เด่อ และพี่ๆ ทุกๆ คน ที่อยู่ในห้องปฏิบัติการฮาร์ดแวร์ ที่ให้ความช่วยเหลือในทุกด้านตั้งแต่การทำงาน ตลอดจนความเป็นอยู่อย่างดีของน้องๆ ในห้องปฏิบัติการ เพื่อน ๆ 4D ทุกคนที่ช่วยทำโปรเจ็ค ช่วยให้กำลังใจ และช่วยก่อความไม่สงบในการทำโปรเจ็ค น้อง ๆ 3D และ 2D ที่มาช่วยงานของห้องฮาร์ดแวร์ ใครรู้ตัวว่ามาช่วยก็รับคำขอบคุณไปละกัน และที่ขาดไม่ได้ขอขอบคุณห้องปฏิบัติการฮาร์ดแวร์ที่ให้สถานที่ในการทำโปรเจ็คและเป็นที่อยู่อาศัยในบางช่วงที่จำเป็นต้องทำงานอย่างหนัก

และต้องขอขอบพระคุณบุคคลที่สำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสูดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ

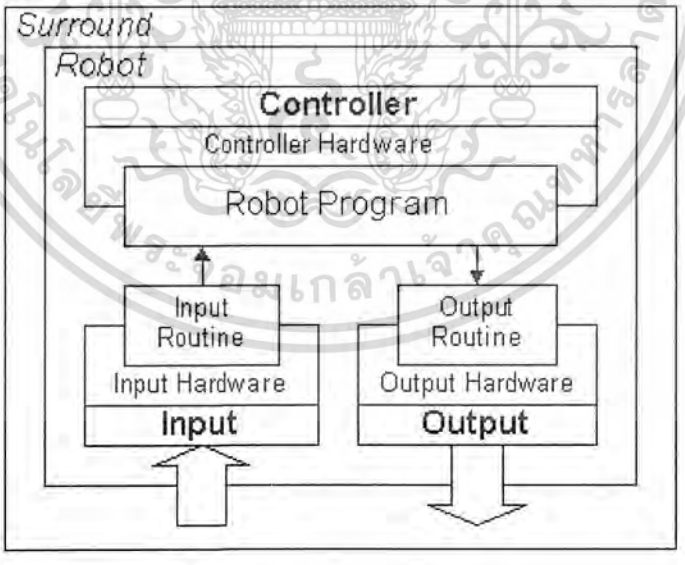
1.1 ความสำคัญ และ ที่มา

โครงงานนี้เกิดขึ้นมาจากความสนใจในเทคโนโลยี 2 สาขา ที่เกี่ยวข้องกัน ได้แก่ เทคโนโลยีด้านหุ่นยนต์ และ เทคโนโลยีการออกแบบวงจรดิจิตอลด้วยภาษาวีเอชดีแอล

1.1.1 หุ่นยนต์

ในปัจจุบัน หุ่นยนต์ได้เข้ามามีบทบาทในสังคมมนุษย์เป็นอย่างมาก ไม่ว่าจะเป็นหุ่นยนต์ที่ใช้ในสายการผลิตภายในโรงงานอุตสาหกรรม , หุ่นยนต์ที่ใช้เพื่อความบันเทิงอย่างหุ่นยนต์สุนัขไอโบ (Aibo) จากบริษัทโซนี่ หรือ หุ่นยนต์ที่ใช้ทำงานในสถานที่ที่มนุษย์ไม่สามารถเข้าถึง เช่น หุ่นยนต์พาธไฟน์เดอร์ (Pathfinder) ที่ใช้สำรวจดาวอังคาร จากสิ่งที่ได้กล่าวมาแล้ว เป็นที่ยืนยันได้ว่า หุ่นยนต์กำลังกลายเป็นเทคโนโลยีที่สำคัญสำหรับให้ปัจจุบันก้าวไปสู่อนาคต

ภายในหุ่นยนต์แต่ละตัวประกอบไปด้วยส่วนประกอบที่สำคัญ ได้แก่ ส่วนควบคุม (Controller) , ส่วนรับข้อมูลเข้า (Input) และ ส่วนนำข้อมูลออก (Output) ซึ่งภายในแต่ละส่วนก็จะประกอบไปด้วยส่วนที่เป็น ฮาร์ดแวร์ซึ่งเป็นวงจรอิเล็กทรอนิกส์ เพื่อทำให้หุ่นยนต์สามารถทำงานอยู่ได้ เปรียบเสมือนกับร่างกายของมนุษย์ และ ส่วนที่เป็น ซอฟต์แวร์ซึ่งทำหน้าที่ควบคุมการทำงานของส่วนต่างๆ ดังรูปที่ 1-1



รูปที่ 1-1 ส่วนประกอบภายในหุ่นยนต์

เมื่อพิจารณาจะพบว่า ในการสร้างหุ่นยนต์แต่ละตัวนั้น ส่วนของอินพุท และ เอาต์พุต ไม่ว่าจะเป็นฮาร์ดแวร์ หรือ ซอฟต์แวร์ ซึ่งเป็นรูทีนอยู่ภายในนั้น มักจะซ้ำกัน ตัวอย่างเช่น เซนเซอร์อินฟราเรด , เซนเซอร์อัลตราโซนิค หรือ มอเตอร์แบบต่างๆ เป็นต้น เมื่อมีการสร้างหุ่นยนต์ตัวใหม่ขึ้น ก็จะต้องสร้าง

ส่วนเหล่านี้ใหม่ทั้งหมด ทำให้สิ้นเปลืองงบประมาณ และ ทำให้เสียเวลาในการทดสอบ ถึงแม้ว่าจะใช้สิ่งที่ได้ออกแบบมาจากหุ่นยนต์ตัวก่อนหน้านั้นก็ตาม อีกประการหนึ่งคือ ในการสร้างหุ่นยนต์ให้ทำงานตามที่ต้องการนั้น (เทียบในกรณีที่ใช้อินพุต และ เอาต์พุต เหมือนกัน) ผู้สร้างสนใจเพียงการเขียนโปรแกรมควบคุมหุ่นยนต์ให้ทำงานในขั้นตอนที่ต้องการเท่านั้น การที่ต้องเข้ามายุ่งเกี่ยวกับการจัดการระดับล่างนั้นเป็นสิ่งไม่จำเป็น จากเหตุผลดังกล่าวทำให้เกิดแนวความคิดของการนำกลับมาใช้ใหม่ว่า "อินพุต และ เอาต์พุต ของหุ่นยนต์ควรจะเป็นชิ้นส่วนสำเร็จรูปที่ได้รับการทดสอบอย่างดีแล้ว สามารถนำกลับมาใช้ใหม่ได้ โดยที่ภายในควรจะมีรูทีนที่ใช้จัดการงานพื้นฐาน เมื่อผู้ใช้ต้องการสร้างหุ่นยนต์ ก็สามารถเลือกชิ้นส่วนของอินพุต , เอาต์พุต ที่จำเป็น และ เขียนโปรแกรมเฉพาะส่วนควบคุมหุ่นยนต์เท่านั้น"

1.1.2 การออกแบบวงจรดิจิตอลด้วยภาษาวีเอชดีแอล

เนื่องจากการพัฒนาเทคโนโลยีที่รุดหน้าอย่างไม่หยุดยั้ง และ สภาวะการแข่งขันในตลาดที่สูงขึ้น ทำให้ความต้องการวงจรอิเล็กทรอนิกส์ที่เป็นวงจรดิจิตอลมีมากขึ้น ดังนั้นขั้นตอนการออกแบบวงจรในสมัยก่อนที่ค่อนข้างยุ่งยาก และ ล่าช้า จำเป็นต้องได้รับการพัฒนาให้มีประสิทธิภาพมากขึ้น จากเดิมที่ต้องใช้มนุษย์ในการออกแบบทุกขั้นตอน จึงมีการนำคอมพิวเตอร์เข้ามาช่วยในขั้นตอนที่คอมพิวเตอร์สามารถทำได้ เช่น การลดรูปสมการ , การลดรูปสเตท เป็นต้น ซึ่งในปัจจุบัน ได้มีการพัฒนาภาษาบรรยายวงจรสำหรับการออกแบบวงจรดิจิตอลขึ้นมา

ในปัจจุบัน การออกแบบวงจรดิจิตอลจะใช้ภาษาบรรยายวงจรในการออกแบบ แล้ว ใช้ซอฟต์แวร์จำลองการทำงานของวงจรนั้น เมื่อผลการจำลองการทำงานเป็นที่ถูกต้อง ก็จะใช้ ซอฟต์แวร์เครื่องมือสังเคราะห์วงจรเพื่อแปลงให้เป็นวงจรดิจิตอลที่ทำงานได้ตามที่บรรยายไว้ วงจรที่ได้จากการสังเคราะห์สามารถนำไปสร้างเป็นไอซีเฉพาะงานที่เรียกว่า ASIC (Application Specific IC) หรือ จะนำไปโปรแกรมลงอุปกรณ์เพื่อให้ทำงานเป็นวงจรที่ต้องการได้

อุปกรณ์ที่สามารถโปรแกรมได้เพื่อให้จำลองตัวเป็นวงจรดิจิตอลที่ได้ออกแบบไว้ ยกตัวอย่าง เช่น ซีพีแอลดี (CPLD) , เอฟพีจีเอ (FPGA) เพื่อนำมาใช้ในการสร้างต้นแบบ , ทดสอบ หรือ นำไปใช้งานจริง อุปกรณ์เหล่านี้ทำให้การออกแบบวงจรดิจิตอล สามารถตรวจสอบความถูกต้องได้รวดเร็ว และ ง่ายขึ้น

สิ่งที่ได้กล่าวมานี้ ทำให้การออกแบบวงจรดิจิตอลด้วยภาษาบรรยายวงจร เป็นสิ่งที่น่าสนใจ โดยเฉพาะการออกแบบด้วยภาษาวีเอชดีแอล ซึ่งใช้กันอย่างแพร่หลายฯ

1.1.3 แนวคิดในการออกแบบ

เมื่อรวมเอาสิ่งที่ได้กล่าวมาในเทคโนโลยีทั้งสองเข้าด้วยกัน ทั้งความต้องการนำชิ้นส่วนในการสร้างหุ่นยนต์กลับมาใช้ใหม่ และ ความน่าสนใจในการออกแบบวงจรดิจิตอลด้วยภาษาวีเอชดีแอล ทำให้เกิดโครงงานชุดพัฒนาหุ่นยนต์ขนาดเล็กขึ้น โดยในโครงงานนี้จะเป็นการสร้างชิ้นส่วนสำเร็จรูปที่ใช้ในการสร้างหุ่นยนต์ขนาดเล็ก เพื่อทดสอบแนวความคิดในการสร้างชุดพัฒนาหุ่นยนต์ที่มีความสามารถมากขึ้นต่อไป โดยโครงงานนี้ จะแยกแต่ละชิ้นส่วนออกเป็นโมดูลที่เป็นอิสระจากกัน ติดต่อสื่อสารผ่านทาง

บัสที่ออกแบบขึ้นมาเฉพาะโดยมีใช้ระบบบัสแบบ I2C มาเป็นส่วนหนึ่งในการรับส่งข้อมูล และ ภายในโมดูลจะมีคำสั่งที่ใช้ในการทำงานระดับล่าง ดังนั้นส่วนควบคุมเพียงส่งคำสั่งผ่านทางบัสของระบบไปให้กับโมดูลเท่านั้น โดยไม่จำเป็นต้องยุ่งกับการควบคุมระดับล่าง ทำให้ผู้ใช้สนใจเพียงแต่การเขียนโปรแกรมควบคุมการทำงานของหุ่นยนต์เท่านั้น ข้อดีอีกประการหนึ่งคือ ผู้ใช้สามารถเปลี่ยนอุปกรณ์ที่ทำหน้าที่คล้ายกันได้โดยไม่จำเป็นต้องเขียนโปรแกรมใหม่ หากอุปกรณ์ตัวที่ต้องการเปลี่ยนนั้น มีคำสั่งที่เหมือนกัน ตัวอย่างเช่น เปลี่ยนจากโมดูลขับเคลื่อนด้วยเซอร์โวมอเตอร์ เป็น โมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์ เป็นต้น

โมดูลควบคุมหลัก ภายในโครงงานนี้ จะใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นตัวควบคุมการทำงาน ส่วนโมดูลอินพุต และ เอาต์พุต ในขั้นต้นได้ทดลองสร้างด้วยซีพีแอลดี แต่เนื่องจากเกิดอุปสรรคหลายประการ เช่น วงจรที่ออกแบบไม่สามารถจุลงในซีพีแอลดีตัวเดียวได้ ทำให้ต้องใช้ซีพีแอลดีหลายตัว จึงได้พิจารณาทางเลือกในการสร้างแบบอื่น และพบว่า การสร้างโมดูลอินพุต และ เอาต์พุต โดยใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ควบคุมการทำงานเช่นเดียวกับโมดูลควบคุมหลักนั้น เป็นทางเลือกที่เหมาะสมที่สุด อย่างไรก็ตามในโครงงานนี้ ได้รวมรายละเอียดโมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์ที่สร้างด้วยซีพีแอลดีเข้าไว้ด้วย เนื่องจากเป็นโมดูลแรกที่ทดลองสร้างด้วยซีพีแอลดี ก่อนที่จะเลือกเปลี่ยนไปสร้างด้วย MCS-51

เหตุผลอีกประการหนึ่งที่ทำให้เกิดโครงงานนี้ขึ้น คือ ต้องการนำไปเป็นส่วนหนึ่งของผลงานที่แสดงในงานวิศวะลาดกระบังนิทรรศน์ประยุกต์ 2000

1.2 วัตถุประสงค์

1. สร้างโมดูลของชุดพัฒนาหุ่นยนต์ขนาดเล็กที่มีการทำงานแต่ละส่วนเป็นอิสระจากกันโดยจะทำงานตามคำสั่งที่ได้รับจากโมดูลที่ทำหน้าที่เป็นตัวควบคุมหลักของตัวหุ่น
2. ออกแบบระบบบัสที่ใช้ติดต่อสื่อสารระหว่างแต่ละโมดูลที่สามารถรองรับการเพิ่มลดจำนวนโมดูล I/O ที่ติดตั้งภายในตัวหุ่นได้
3. ทำการสร้างหุ่นยนต์ทดสอบ ที่สามารถทำทดสอบการทำงานทั้งหมดของทุกโมดุลได้อย่างสมบูรณ์ และนำไปใช้งานต่อได้
4. ศึกษาการออกแบบวงจรดิจิตอลด้วยภาษาวีเอชดีแอล
5. แสดงในงานวิศวะลาดกระบังนิทรรศน์ประยุกต์ 2000

1.3 ขอบเขตของโครงงาน

ทำการสร้างชุดพัฒนาหุ่นยนต์ขนาดเล็กที่ประกอบด้วยโมดูลควบคุมหลักและโมดูล I/O อื่นๆ ที่ทำการติดต่อสื่อสารกันโดยผ่านทางบัส I2C ซึ่งแต่ละโมดูลจะใช้ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เป็นตัวควบคุมการทำงานของโมดูล โดยที่แต่ละโมดูล I/O แต่ละโมดูลทำงานเป็นอิสระจากกัน โดยรับคำสั่งจากโมดูลควบคุมหลักผ่านทางบัส I2C ไปทำงาน สามารถเพิ่มหรือลดโมดูล I/O ได้โดยที่หุ่นยนต์ยังคงสามารถทำงานได้ตามความสามารถของแต่ละโมดูล I/O ที่ยังคงติดตั้งอยู่ในตัวหุ่น

โครงงานนี้จะสร้างโมดูลของชุดพัฒนาหุ่นยนต์จำนวนทั้งหมด 7 โมดูล ได้แก่

- โมดูลขับเคลื่อนด้วยดีซีมอเตอร์
- โมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์
- โมดูลขับเคลื่อนด้วยเซอร์โวมอเตอร์
- โมดูลตรวจจับด้วยอินฟราเรด
- โมดูลตรวจจับด้วยอัลตราโซนิค
- โมดูลติดต่อกับผู้ใช้
- โมดูลควบคุมหลัก

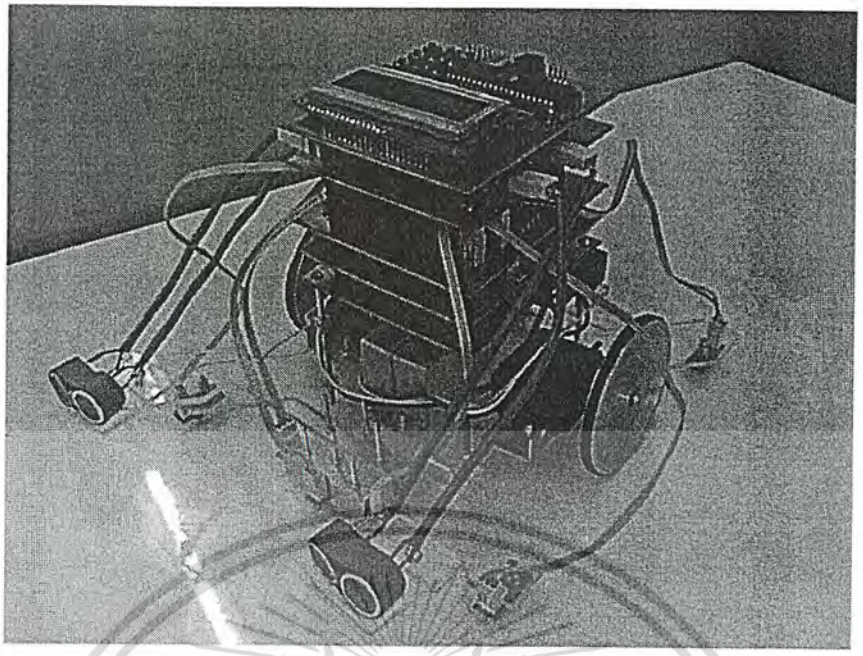
ซึ่งโมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์ จะมี 2 ชุด คือชุดที่ไมโครคอนโทรลเลอร์ ตระกูล MCS-51 และ อีกชุดหนึ่งทำงานด้วยซีพีแอลดี เมื่อสร้างโมดูลทั้งหมดเป็นที่เรียบร้อยแล้ว จะนำโมดูลทั้งหมดมาสร้างเป็นหุ่นยนต์เพื่อทดสอบการทำงาน โดยมีเกณฑ์ในการทดสอบดังนี้

1. หุ่นยนต์สามารถถอดประกอบโมดูล I/O ที่ไม่จำเป็นเพื่อโปรแกรมให้ทำงานต่างกันได้
2. หุ่นยนต์ต้องสามารถเปลี่ยนโมดูล I/O ที่ทำหน้าที่คล้ายกันได้ (ตัวอย่างเช่นโมดูลขับเคลื่อน) โดยที่ไม่ต้องแก้ไขโปรแกรมใหม่
3. หุ่นยนต์สามารถทำงานต่อไปได้ในกรณีที่ถอดโมดูล I/O บางโมดูลที่จำเป็นในการทำงานออกไป

ตัวอย่างหุ่นยนต์ที่ประกอบขึ้นจากโมดูลทั้งหมดที่สร้างขึ้นมาแสดงไว้ดังรูปที่ 1-1 เป็นหุ่นยนต์ที่สามารถโปรแกรมการเคลื่อนที่ได้โดยมีขอบเขตในทำงานคือจะเคลื่อนที่บนพื้นโต๊ะสีขาว หรือ พื้นที่มีความสามารถในการสะท้อนแสงได้และสามารถหลบหลีกสิ่งกีดขวางที่ขวางเส้นทางการเคลื่อนที่ของหุ่นได้ จากนั้นจึงกลับมาทำงานตามโปรแกรมที่ได้กำหนดไว้ต่อไป

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทำให้การสร้างหุ่นยนต์ขนาดเล็กที่ซับซ้อนไม่มาก ทำได้อย่างรวดเร็ว และ ลดค่าใช้จ่าย เนื่องจากนำชิ้นส่วนเดิมกลับมาใช้ใหม่ได้
2. ชุดพัฒนาหุ่นยนต์นี้ สามารถนำไปพัฒนาต่อเพื่อผลิตในเชิงพาณิชย์ได้
3. รู้จักขั้นตอนการทำงาน การวางแผนงาน การตรวจสอบการทำงาน และ แนวทางการแก้ปัญหาที่เกิดขึ้นได้
4. เข้าใจการทำงานของไมโครคอนโทรลเลอร์ MCS-51 , ซีพีแอลดี และ ภาษา VHDL มากขึ้น
5. เข้าใจการทำงานของระบบบัส I2C รวมถึงการดัดแปลงระบบบัสเพื่อให้เข้ากับระบบงานที่ต้องการออกแบบได้



รูปที่ 1-2 ตัวอย่างหุ่นยนต์ที่ประกอบขึ้นจากโมดูลต่างๆ ที่จำเป็นในการทำงาน

บทที่ 2
ไมโครคอนโทรลเลอร์ตระกูล MCS-51

2.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ หมายถึง อุปกรณ์อิเล็กทรอนิกส์ที่รวมเอาหน่วยประมวลผล หน่วยคำนวนทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาท์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกา เข้าไว้ด้วยกัน ทำให้มีความคล่องตัวในการนำไปประยุกต์ใช้งานด้านการควบคุมสูง

2.2 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์เบอร์ AT89S8252

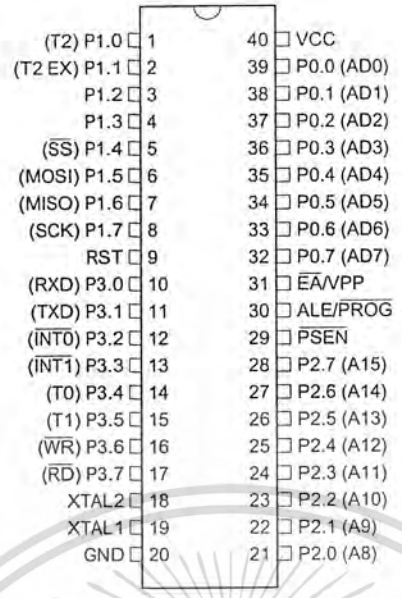
ไมโครคอนโทรลเลอร์ที่เลือกใช้ในโครงงานนี้ คือ ไมโครคอนโทรลเลอร์ตระกูล51 เบอร์ AT89S8252 ซึ่งมีคุณสมบัติทางเทคนิคดังนี้

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- มีหน่วยความจำแบบแฟลช ขนาด 8 กิโลไบต์ สำหรับเก็บโค้ด
- มีหน่วยความจำชั่วคราวขนาด 256 ไบต์
- พอร์ตทำงานได้สองทิศทาง เป็นทั้งอินพุตและเอาต์พุต
- มีแหล่งอินเตอร์รัปต์ได้ 6 ประเภท
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- มีไทม์เมอร์/เคาเตอร์ขนาด 16 บิต จำนวน 3 ชุด
- สามารถอ้างอิงหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- ประกอบด้วยพอร์ต จำนวน 4 พอร์ต พอร์ตละ 8 บิต สามารถทำการเซ็ตและเคลียร์แต่ละบิตได้อย่างอิสระ

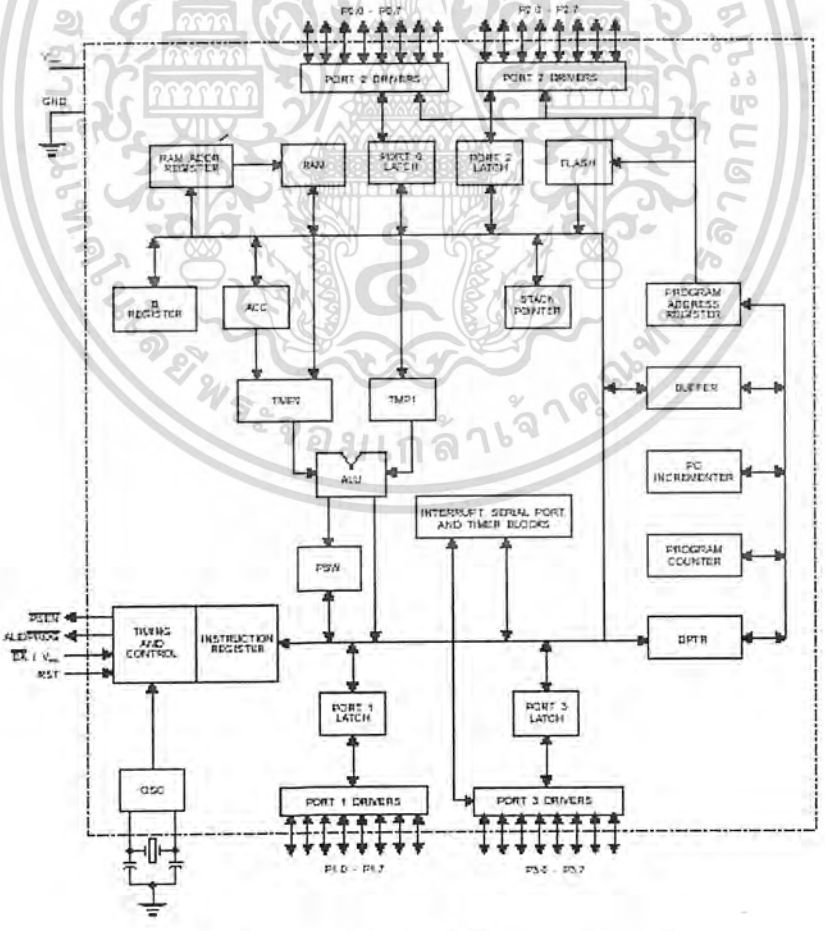
2.3 การจัดขาของไมโครคอนโทรลเลอร์เบอร์ AT89S8252

ไมโครคอนโทรลเลอร์เบอร์ AT89S8252 จัดอยู่ในตระกูล MCS-51 ซึ่งทำให้มีการจัดขาเหมือนกัน แต่จะมีบางขาที่เพิ่มเติมความสารถเพิ่มเติมเข้าไป รูปแบบการจัดขาแสดงไว้ดังรูปที่ 2-1 และโครงสร้างภายในดังรูปที่ 2-2 โดยมีรายละเอียดดังนี้

- ขา Vcc ใช้สำหรับต่อไปเลี้ยง +5 โวลต์ ให้กับตัวไมโครคอนโทรลเลอร์
- ขา Gnd ใช้สำหรับต่อกับกราวด์ของระบบ



รูปที่ 2-1 การจัดขาของไมโครคอนโทรลเลอร์เบอร์ AT89S8252



รูปที่ 2-2 โครงสร้างภายในไมโครคอนโทรลเลอร์

- ขาพอร์ต 0 (P0.0 – P0.7) มี 8 ขา โดยแต่ละขาสามารถทำหน้าที่เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ในกรณีที่ใช้ขาใดขาหนึ่งเป็นขาอินพุต จะต้องเขียนข้อมูล 1 ให้กับขานั้น ก่อนทำการอ่านข้อมูลจากขาพอร์ตนั้น นอกจากนั้นยังทำหน้าที่เป็นขาส่งแอดเดรสไบต์ต่ำ (A0-A7) และ เป็นขารับส่งข้อมูล (D0-D7) เมื่อทำการติดต่อกับหน่วยความจำภายนอก โดยจะมีการสลับการทำงาน
- ขาพอร์ต 1 (P1.0 – P1.7) มี 8 ขา โดยแต่ละขาสามารถทำหน้าที่เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ในกรณีที่ใช้ขาใดขาหนึ่งเป็นขาอินพุต จะต้องเขียนข้อมูล 1 ให้กับขานั้น ก่อนทำการอ่านข้อมูลจากขาพอร์ตนั้น นอกจากนั้นบิตที่ 0 และ 1 ยังใช้เป็นขาอินพุตทริกเกอร์สำหรับไทม์เมอร์ 2 เมื่อทำงานในโหมดเคาน์เตอร์ และ บิตที่ 4 ถึง 7 ยังใช้ในการโปรแกรมแบบ SPI ลงบนตัวชิปด้วย
- ขาพอร์ต 2 (P2.0 – P2.7) มี 8 ขา โดยแต่ละขาสามารถทำหน้าที่เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ในกรณีที่ใช้ขาใดขาหนึ่งเป็นขาอินพุต จะต้องเขียนข้อมูล 1 ให้กับขานั้น ก่อนทำการอ่านข้อมูลจากขาพอร์ตนั้น นอกจากนั้นยังทำหน้าที่เป็นขาส่งแอดเดรสไบต์สูง (A8-A15) เมื่อทำการติดต่อกับหน่วยความจำภายนอก
- ขาพอร์ต 3 (P3.0 – P3.7) มี 8 ขา โดยแต่ละขาสามารถทำหน้าที่เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ในกรณีที่ใช้ขาใดขาหนึ่งเป็นขาอินพุต จะต้องเขียนข้อมูล 1 ให้กับขานั้น ก่อนทำการอ่านข้อมูลจากพอร์ตนั้น นอกจากนั้น พอร์ต 3 ยังใช้ทำหน้าที่พิเศษดังนี้
  - P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือ ขา RxD
  - P3.1 ใช้เป็นขาเอาต์พุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือ ขา TxD
  - P3.2 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 0 หรือ ขา INT0
  - P3.3 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 1 หรือ ขา INT1
  - P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทม์เมอร์จากภายนอกช่องที่ 0 หรือ T0
  - P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทม์เมอร์จากภายนอกช่องที่ 1 หรือ T1
  - P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีที่มีการเชื่อมต่อกับหน่วยความจำภายนอก
  - P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีที่มีการเชื่อมต่อกับหน่วยความจำภายนอก
- ขารีเซต ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อทำการรีเซตนั้น ต้องให้ระดับสัญญาณอยู่ที่สถานะสูงเป็นเวลาอย่างน้อย 2 แมชชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานตามปกติ
- ขา ALE/PROG (Address Latch Enable/program pulse input) เป็นขาที่ใช้ควบคุมการแลตช์ค่าแอดเดรสของพอร์ต 0 เมื่อมีการเชื่อมต่อกับหน่วยความจำภายนอก นอกจากนั้นยังใช้เป็น

ขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในอีอีพรอมของไมโครคอนโทรลเลอร์

- ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ตัวไมโครคอนโทรลเลอร์ จะส่งสัญญาณออกมาจากขานี้ 2 ครั้งในแต่ละแมชชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอกขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา
- ขา EA/Vpp (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้มีสถานะต่ำจะเป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้มีสถานะสูงจะเป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงขนาด 12 โวลต์ สำหรับการโปรแกรมหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์
- ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.4 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ AT89S8252

ในไมโครคอนโทรลเลอร์ AT89S8252 จะประกอบด้วยหน่วยความจำภายในหลัก ๆ 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล โดยมีการจัดโครงสร้างดังนี้

หน่วยความจำโปรแกรม

ไมโครคอนโทรลเลอร์ AT89S8252 สามารถอ้างอิงหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับหน่วยความจำภายนอกหรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังรูปที่ 2-1 โดยในไมโครคอนโทรลเลอร์ AT89S8252 มีหน่วยความจำโปรแกรมอยู่ 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นอยู่ที่ 0000H เมื่อ ซีพียูได้รับการรีเซตให้เริ่มต้นการทำงานจะต้องมาเริ่มต้นอ่านโปรแกรมที่แอดเดรส 0000H เสมอ แต่ว่า พื้นที่บางส่วนของหน่วยความจำโปแกรมบางส่วนจะถูกสงวนไว้สำหรับการบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

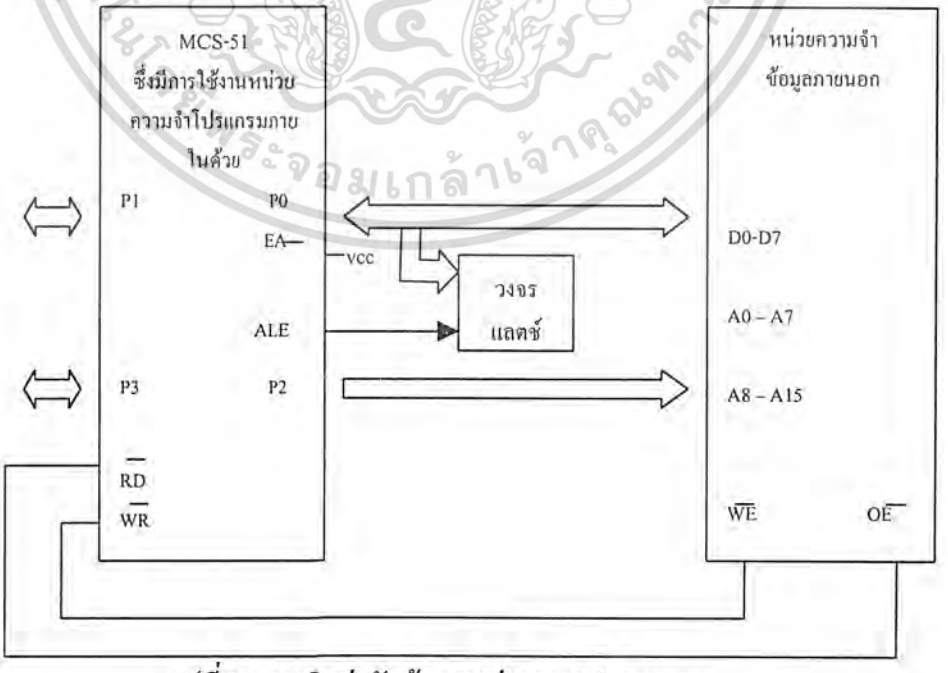
- อินเตอร์รัปต์จากภายนอกเบอร์ 0 กำหนดไว้ที่แอดเดรส 0003H
- อินเตอร์รัปต์จากไทม์เมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH
- อินเตอร์รัปต์จากภายนอกเบอร์ 1 กำหนดไว้ที่แอดเดรส 0013H

- อินเตอร์รัปต์จากไทม์เมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH
- อินเตอร์รัปต์จากการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H
- อินเตอร์รัปต์จากไทม์เมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

กรณีที่มีการใช้หน่วยความจำโปรแกรมทั้งภายในและภายนอก สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายใน ดังนี้ ไมโครคอนโทรลเลอร์ AT89S8252 มีหน่วยความจำโปรแกรมภายใน 8 กิโลไบต์ อ้างอิงจากแอดเดรสตั้งแต่ 0000H ถึง 1FFFH เมื่อต่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดแอดเดรสให้อยู่ในช่วง 2000H ถึง FFFFH ตัวอย่างการต่อหน่วยความจำภายนอกแสดงไว้ดังรูปที่ 2-2 จะเห็นได้ว่า ขาพอร์ต P0.0 – P0.7 ถูกใช้เป็นขาข้อมูล D0 – D7 และขาแอดเดรสไบต์ต่ำ โดยผ่านวงจรแลตช์ ซึ่งโดยปกติใช้ไอซีเบอร์ 74HC573 และใช้ขาสัญญาณ ALE และ PSEN ในการเลือกว่า ต้องการใช้งานขา P0.0 – P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเดรส ในขณะที่ขา P2.0 – P2.7 ใช้ในการเชื่อมต่อกับขาแอดเดรสไบต์สูง A8 – A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ไมโครคอนโทรลเลอร์ จะเหลือขาพอร์ตใช้งานเพียง 16 บิต คือ พอร์ต 1 และ พอร์ต 3 เท่านั้น

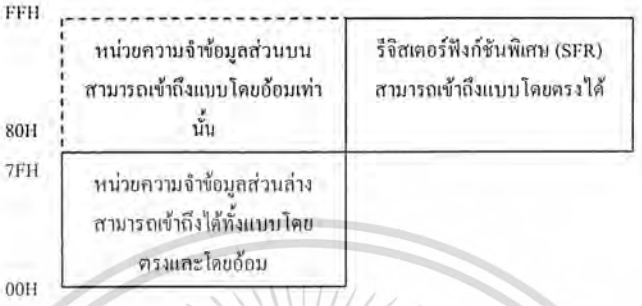
หน่วยความจำข้อมูล

แบ่งได้เป็น 2 แบบ คือ หน่วยความจำข้อมูลภายนอกและหน่วยความจำข้อมูลภายในการติดต่อกับหน่วยความจำข้อมูลภายนอกแสดงดังรูปที่ 2-3 จะเห็นได้ว่ามีลักษณะคล้ายกับการต่อหน่วยความจำโปรแกรมภายนอก แต่มีจะมีขาสัญญาณที่ใช้สำหรับการอ่านเขียนหน่วยความจำข้อมูลภายนอก เพิ่มขึ้นมาคือ ขา RD และ ขา WR



รูปที่ 2-3 การติดต่อกับข้อมูลหน่วยความจำภายนอก

หน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์เป็บแบบแรม (RAM : Random Access Memory) มีการจัดสรรหน่วยความจำภายในเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่าง หน่วยความจำข้อมูลส่วนบน และรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR : Sepcial Function Register) แต่ละส่วนมีขนาด 128 ไบต์ โดยแสดงไว้ดังรูปที่ 2-4



รูปที่ 2-4 การจัดสรรหน่วยความจำภายใน

จะเห็นได้ว่าหน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งทับซ้อนกัน แต่จะใช้วิธีการติดต่อที่แตกต่างกัน ทำให้ดูเสมือนว่าไมโครคอนโทรลเลอร์มีหน่วยความจำข้อมูลภายในสูงถึง 384 ไบต์ โดยหน่วยความจำข้อมูลส่วนล่างขนาด 128 ไบต์ มีแอดเดรสอยู่ที่ 00H – 70H สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128 ไบต์เช่นกัน อยู่ที่แอดเดรส 80H – FFH แต่สามารถเข้าถึงแบบโดยอ้อมได้เท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H – FFH เช่นเดียวกัน แต่จะเข้าถึงได้โดยตรง

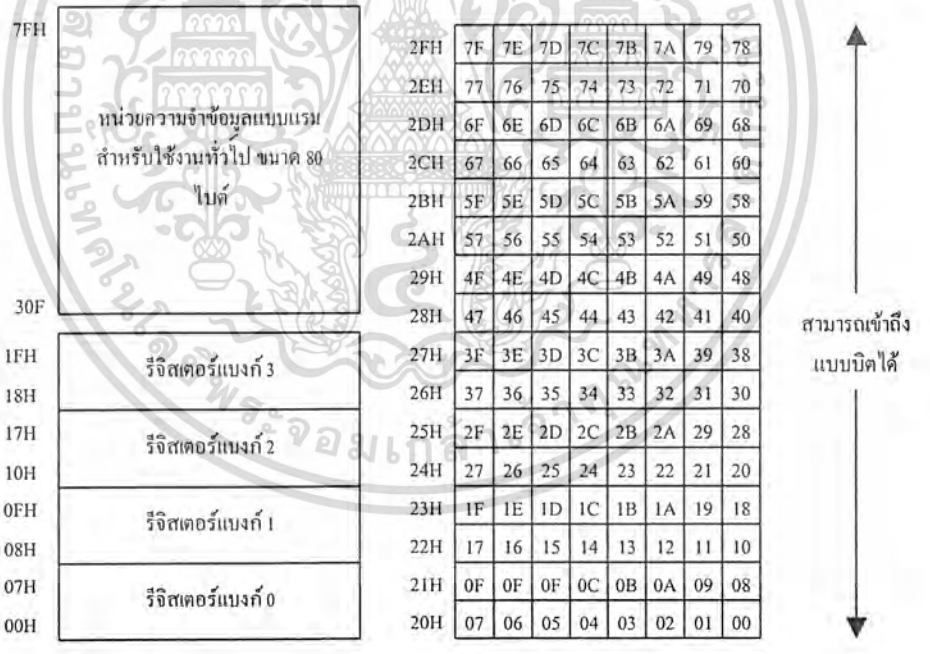
สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่อยู่ในอนุกรม x51 จะมีหน่วยความจำข้อมูลภายในเพียง128 ไบต์ โดยที่ไม่มีหน่วยความจำข้อมูลส่วนบน

ในการจัดสรรหน่วยความจำข้อมูลส่วนล่าง หน่วยความจำ 32 ไบต์ต่ำสุดที่แอดเดรส 00H – 1FH แบ่งได้เป็น 4 กลุ่ม เรียกว่า 4 แบงก์ แต่ละแบงก์มีรีจิสเตอร์ 8 ตัวคือ R0 – R7 การติดต่อรีจิสเตอร์ในแบงก์ใดสามารถกำหนดได้โดยรีจิสเตอร์ PSW (Program Status Word register) ที่อยู่ในรีจิสเตอร์ฟังก์ชั่นพิเศษ ซึ่งจะกล่าวถึงต่อไป

หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H – 2FH เป็นพื้นที่สำหรับใช้งานทั่วไป ที่สามารถเข้าถึงได้ในระดับบิต และหน่วยความจำข้อมูลที่เหลือ 80 ไบต์จะต้องแบ่งส่วนหนึ่งสำรองไว้เป็นพื้นที่ของสแต็ก ซึ่งต้องเข้าถึงในระดับไบต์เท่านั้น

ในรูป 2-5 แสดงการจัดโครงสร้างของหน่วยความจำข้อมูลส่วนล่าง และในรูปที่ 2-6 แสดงโครงสร้างของหน่วยความจำข้อมูลส่วนบนที่มีโครงสร้างคล้ายกับความจำส่วนล่างแต่ใน 80 ไบต์บนไม่จำเป็นต้องสำรองไว้สำหรับสแต็ก และต้องเข้าถึงโดยอ้อมเท่านั้น

รูปที่ 2-5 โครงสร้างหน่วยความจำข้อมูลส่วนล่าง



รูปที่ 2-6 โครงสร้างหน่วยความจำข้อมูลส่วนบน

รีจิสเตอร์ฟังก์ชั่นพิเศษ

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ มีทั้งหมด 28 ตัว มีแอดเดรสอยู่ระหว่าง 80H – FFH ซึ่งสามารถเข้าถึงได้โดยตรง ดังรูปที่ 2-7 รายละเอียดของรีจิสเตอร์ฟังก์ชั่นพิเศษมีดังนี้

แอดเดรส	บิต								
FFH									
F0H	B7	B6	B5	B4	B3	B2	B1	B0	รีจิสเตอร์ B
E0H	A7	A6	A5	A4	A3	A2	A1	A0	รีจิสเตอร์ ACC
D0H	D7	D6	D5	D4	D3	D2	-	D0	รีจิสเตอร์ PSW
B8H	-	-	-	D4	D3	D2	D1	D0	รีจิสเตอร์ IP
B0H	3.7	3.6	3.5	3.4	3.3	3.2	3.1	3.0	รีจิสเตอร์ P3
A8H	D7	-	-	D4	D3	D2	D1	D0	รีจิสเตอร์ IE
A0H	2.7	2.6	2.5	2.4	2.3	2.2	2.1	2.0	รีจิสเตอร์ P2
99H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ SBUF
98H	S7	S6	S5	S4	S3	S2	S1	S0	รีจิสเตอร์ SCON
90H	1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0	รีจิสเตอร์ P1
8DH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TH1
8CH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TH0
8BH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TL1
8AH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TL0
89H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TMOD
88H	T7	T6	T5	T4	T3	T2	T1	T0	รีจิสเตอร์ TCON
87H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ PCON
83H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ DPH
82H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ DPL
81H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ SP
80H	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	รีจิสเตอร์ P0

รูปที่ 2-7 รีจิสเตอร์ฟังก์ชั่นพิเศษ

รีจิสเตอร์แสดงสถานะของโปรแกรม (Pragram Status Word : PSW)

สามารถเข้าถึงได้ในระดับบิต มีแอดเดรสอยู่ที่ D0H เป็นรีจิสเตอร์ที่เก็บสถานะของการทำงานของโปรแกรมในขณะนั้น จะเรียกสถานะของโปรแกรมว่า "แฟลก" เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงจะถูกเก็บไว้ในรีจิสเตอร์นี้ รูปที่2-8 แสดงการเรียงบิตของรีจิสเตอร์สถานะซึ่งแต่ละบิตของรีจิสเตอร์ประกอบด้วย

- CY : เป็นแฟลกทด เป็น '1' เมื่อมีการกระทำทางคณิตศาสตร์และลอจิก แล้วค่าของแอกคิวมูลเลเตอร์เกิน 255 (ฐานสิบ) หรือ FFH
- AC : เป็นแฟลกทดเสริม เป็น '1' เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายังบิต 4
- F0 : เป็นแฟลกใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่ากระทำคำสั่งใดๆ ที่บิตนี้จะไม่มีการเปลี่ยนแปลง

- RS1 : เป็นบิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับบิต RS0 เพื่อเลือกแบงก์ของรีจิสเตอร์ R0 – R7
- RS0 : เป็นบิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับบิต RS1 เพื่อเลือกแบงก์ของรีจิสเตอร์ R0 – R7
- OV : บิตเกิน เป็น '1' เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ทำให้เกิดการทดข้ามจากบิตที่ 6 มายังบิตที่ 7 ของแอกคิวมูเลเตอร์ หรือ แอกคิวมูเลเตอร์มีค่าเกิน 127 (ฐานสิบ) นอกจากนั้นยังใช้เป็นการแสดงค่าลบอีกด้วย
- - : เป็นบิตว่าง ผู้ใช้งานสามารถกำหนดใช้งานได้อย่างอิสระ
- P : เป็นบิตพาริตี้ ใช้ในการตรวจสอบจำนวนค่า '1' ภายในแอกคิวมูเลเตอร์ ถ้าหากในแอกคิวมูเลเตอร์มีจำนวนบิตที่เป็น '1' รวมกันเป็นเลขคู่ บิตนี้จะเป็น '0' ถ้ารวมกันเป็นเลขคี่ บิตนี้จะเป็น '1'

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

รูปที่ 2-8 รีจิสเตอร์แสดงสถานะของโปรแกรม

แอกคิวมูเลเตอร์

มีขนาด 8 บิต สามารถเข้าถึงระดับบิตได้ มีแอดเดรสที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์ โดยเฉพาะอย่างยิ่งในการคำนวนทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้นั้นให้แก่ซีพียูเพื่อทำการประมวลผลต่อไป

รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ F0H มีหน้าที่พิเศษคือ การคูณหรือหารทางคณิตศาสตร์จะต้องนำข้อมูลที่ต้องการหารหรือคูณนั้น มาเก็บไว้ในรีจิสเตอร์ B นี้ แล้วจึงกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป ในกรณีที่ไม่ได้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์ B นี้ในการเก็บข้อมูลทั่วไปได้ เหมือนรีจิสเตอร์ปกติ แล้วสามารถเข้าถึงระดับบิตได้

โปรแกรมเคาน์เตอร์ (Program Counter : PC)

มีขนาด 16 บิต มีหน้าที่แจ้งแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งถัดไปที่ซีพียูจะต้องไปทำงาน แต่รีจิสเตอร์ PC เป็นรีจิสเตอร์เพียงตัวเดียวที่ไม่ได้ถูกจัดรวมไว้ รวมกับรีจิสเตอร์ SFR ตัวอื่นๆ

สแต็กพอยน์เตอร์ (Stack Pointer : SP)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81H ใช้ในการเก็บค่าตำแหน่งของตัวชี้สแต็ก ซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อย หรือโปรแกรมกระโดดกลับมายังโปรแกรมหลัก เมื่อมีการรีเซ็ตขึ้น ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H นั่นหมายความว่า ตัวชี้สแต็กมีค่า 07H แอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H

รีจิสเตอร์ขี้ข้อมูล (Data Pointer : DPTR)

มีขนาด 16 บิต แบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไบต์สูง (DPH) และรีจิสเตอร์ชี้ข้อมูลไบต์ต่ำ (DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H และ 83H สำหรับ DPH รีจิสเตอร์ DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

รีจิสเตอร์พอร์ต

มีขนาด 8 บิตที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ มีด้วยกัน 4 ตัว คือ รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H รีจิสเตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H รีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหรือเขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องทำผ่านรีจิสเตอร์นี้ทุกครั้ง

รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ทำการส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ในไมโครคอนโทรลเลอร์โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และ รีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้น จะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ ขา P3.1 เมื่อมีการรับข้อมูลจากขา RxD หรือ ขา P3.0 ข้อมูลจะถูกนำมาเก็บไว้ในบัฟเฟอร์สำหรับรับข้อมูล แล้วจึงมีการอ่านข้อมูลจากบัฟเฟอร์ต่อไป

รีจิสเตอร์ไทม์เมอร์/เคาน์เตอร์

มีขนาด 16 บิต แต่จะจัดแบ่งเป็นไบต์สูงและไบต์ต่ำเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทม์เมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ ภายในไมโครคอนโทรลเลอร์ เพื่อใช้ในการสร้างฐานเวลา จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน

รีจิสเตอร์ควบคุม (Control register)

เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานในส่วนต่าง ประกอบด้วย

- PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์
- SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์
- TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทม์เมอร์/เคาน์เตอร์ ภายในไมโครคอนโทรลเลอร์
- TMOD และ T2MOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะการทำงานของไทม์เมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์
- IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเตอร์รัปต์ โดยที่ IE เป็นรีจิสเตอร์สำหรับเอนาเบิลหรือใช้ในการกำหนดลักษณะการเกิดอินเตอร์รัปต์ ส่วน IP เป็นรีจิสเตอร์ที่ใช้สำหรับกำหนดลำดับความสำคัญในการตอบสนองต่ออินเตอร์รัปต์

2.5 ไทม์เมอร์ / เคาน์เตอร์ในไมโครคอนโทรลเลอร์ AT89S8252

เป็นส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์ เมื่อใช้ในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลา ทั้งนี้เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณพัลส์ เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าของการนับ รวมถึงการกำหนดอัตราเร็วในการสื่อสารข้อมูลอนุกรมด้วย รีจิสเตอร์ T0 และ T1 เป็นรีจิสเตอร์ขนาด 16 บิตที่ใช้สำหรับนับค่าของไทม์เมอร์/เคาเตอร์ ของ ไทม์เมอร์ 0 และ 1 ตามลำดับ

การทำงานเป็นไทม์เมอร์

เมื่อกำหนดให้เป็นตัวตั้งเวลาหรือไทม์เมอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นทุกๆ แมชชีนไซเกิล และเนื่องจากใน 1 แมชชีนไซเกิลประกอบด้วยคาบเวลาจากวงจรกำเนิดสัญญาณนาฬิกาจำนวน 12 คาบเวลา ดังนั้นการนับของรีจิสเตอร์จึงเท่ากับ 1/12 ของความถี่ของสัญญาณนาฬิกา

การทำงานเป็นเคาน์เตอร์

เมื่อทำงานเป็นเคาน์เตอร์ค่าของรีจิสเตอร์จะเพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิกจาก '1'เป็น '0' เกิดขึ้นที่ขาอินพุตทางฮาร์ดแวร์ของวงจรไทม์เมอร์/เคาน์เตอร์ ซึ่งก็คือขา T0(P3.4) และ T1(P3.5)

เมื่อสัญญาณอินพุตเปลี่ยนแปลงจาก '1' เป็น '0' เป็นเวลาหนึ่งไซเกิล ในไซเกิลต่อมาค่าของการนับจะเพิ่มขึ้นหนึ่งค่า และจะไปปรากฏในรีจิสเตอร์ของแมชชีนไซเกิลต่อไปหลังจากที่

ตรวจจับพบการเปลี่ยนแปลงที่ขาไทม์เมอร์อินพุตแล้ว หรือ ต้องใช้เวลาอย่างน้อย 2 แมชชีนไซเกิล หรือ 1/24 ของความถี่สัญญาณนาฬิกา ในการตรวจเจอสัญญาณอินพุต

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทม์เมอร์/เคาน์เตอร์

รีจิสเตอร์พื้นฐานที่เกี่ยวข้องกับการทำงานของไทม์เมอร์/เคาน์เตอร์ มีอยู่ด้วยกัน 3 ตัว ดังมีรายละเอียดดังนี้

1. รีจิสเตอร์ไทม์เมอร์

มีด้วยกัน 4 ตัวคือ TL0 มีแอดเดรสอยู่ที่ 8AH TH0 มีแอดเดรสอยู่ที่ 8CH TL1 แอดเดรสอยู่ที่ 8BH และ TH1 มีแอดเดรสอยู่ที่ 8DH รีจิสเตอร์ทั้ง 4 ตัวอยู่ในพื้นที่ของ รีจิสเตอร์ฟังก์ชันพิเศษ แต่ละตัวมีขนาด 8 บิต ในกรณีที่ใช้ไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิต จะทำการรวม TH และ TL ไว้ด้วยกัน โดยที่ TH จะเป็นบิตบนของข้อมูลรีจิสเตอร์ และ TL จะเป็นบิตล่างของข้อมูล รีจิสเตอร์

2. รีจิสเตอร์ควบคุมการทำงานของไทม์เมอร์/เคาน์เตอร์ (TCON : Timer/Counter Control Register)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 88H ในพื้นที่รีจิสเตอร์ SFR โดยมีการเรียงบิตดังรูปที่ 2-9 ซึ่งแต่ละบิตมีรายละเอียดการทำงานดังนี้

- TF1 (Timer 1 overflow flag) : เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของ รีจิสเตอร์ T1 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น
- TR1 (Timer 1 run control bit) ใช้ในการเปิดปิดการทำงานของไทม์เมอร์ 1 ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทม์เมอร์ 1 ทำงานต้องเซตบิตนี้ให้เป็น '1'
- TF0 (Timer 0 run control bit) เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจิสเตอร์ T0 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น
- TR0 (Timer 0 run control bit) ใช้ในการเปิดปิดการทำงานของไทม์เมอร์ 0 ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทม์เมอร์ 0 ทำงานต้องเซตบิตนี้ให้เป็น '1'
- IE1 (External Interrupt 1 edge flag) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 (INT1) ได้ และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น

- IT1 (Interrupt 1 type control bit) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 1 (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์ หากมีค่าเป็น '0' เลือกขอบขาลงของสัญญาณ หากเป็น '1' เลือกระดับลอจิกต่ำ
- IE0 (External Interrupt 0 edge flag) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 0 (INT0) ได้ และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น
- IT0 (Interrupt 0 type control bit) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 0 (INT0) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์ หากมีค่าเป็น '0' เลือกขอบขาลงของสัญญาณ หากเป็น '1' เลือกระดับลอจิกต่ำ

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

รูปที่ 2-9 รีจิสเตอร์ควบคุมการทำงานของไทม์เมอร์

3. รีจิสเตอร์เลือกโหมดการทำงานของไทม์เมอร์/เคาน์เตอร์ (TMOD : Timer/Counter Mode Control Register)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 88H ในพื้นที่รีจิสเตอร์ SFR แบ่งเป็น 2 ส่วนคือ 4 บิตล่างใช้ในการเลือกโหมดการทำงานของไทม์เมอร์ 0 และ 4 บิตบนใช้ในการเลือกโหมดการทำงานของไทม์เมอร์ 1 โดยมีการเรียงบิตดังรูป ที่ 2-10 แต่ละบิตมีรายละเอียดการทำงานดังนี้

- GATE ใช้เลือกลักษณะการควบคุมการทำงานของไทม์เมอร์/เคาน์เตอร์
  - '0' ไทม์เมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น '1' เรียกการควบคุมแบบนี้ว่า การควบคุมทางซอฟต์แวร์
  - '1' ไทม์เมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น '1' และสถานะลอจิกที่ขาอินพุตอินเตอร์รัปต์ INT0 และ INT1 เป็น '1' เรียกการควบคุมแบบนี้ว่า การควบคุมทางฮาร์ดแวร์
- CT (Timer or Counter selector) ใช้เลือกลักษณะการทำงานของไทม์เมอร์/เคาน์เตอร์

- '0' เลือกให้ทำงานเป็นไทม์เมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์
- '1' เลือกให้ทำงานเป็นเคาน์เตอร์ โดยรับสัญญาณจากภายนอกที่เข้ามาทางขา T0 หรือ T1

- M1,M0 (Mode selector bit) ใช้เลือกโหมดการทำงานของไทม์เมอร์/เคาน์เตอร์
  - '00' เลือกให้ทำงานในโหมดไทม์เมอร์/เคาน์เตอร์ 13 บิต มีการใช้งานรีจิสเตอร์ TL1 เพียง 5 บิต และ TH1 ครบ 8 บิต โดยเมื่อ TL1 นับครบ 32 ก็จะเพิ่มค่า TH1 ขึ้น และเมื่อทำการนับครบรอบ ก็จะทำการเซตบิต TF1 ในรีจิสเตอร์ TCON ส่วนในไทม์เมอร์/เคาน์เตอร์ 0 จะเหมือนกับไทม์เมอร์/เคาน์เตอร์ 1 ทุกประการ
  - '01' เลือกให้ทำงานในโหมดไทม์เมอร์/เคาน์เตอร์ 16 บิต มีการทำงานคล้ายกับ โหมด 0 แต่ใน TL1 จะใช้ครบ 8 บิต ทำให้มีขนาดการนับ 16 บิต ส่วนในไทม์เมอร์/เคาน์เตอร์ 0 จะเหมือนกับไทม์เมอร์/เคาน์เตอร์ 1 ทุกประการ
  - '10' เลือกให้ทำงานในโหมดไทม์เมอร์/เคาน์เตอร์ขนาด 8 บิตแบบตั้งอัตโนมัติ ในโหมดนี้จะแยกรีจิสเตอร์ออกเป็น 2 ตัว ตัวละ 8 บิต โดยรีจิสเตอร์ TL1 ทำหน้าที่เป็นตัวนับค่า ส่วน TH1 ใช้ในการเก็บค่าเริ่มต้นของการนับ เมื่อเริ่มต้นการทำงาน ค่าในรีจิสเตอร์ TH1 จะถูกส่งไปยังรีจิสเตอร์ TL1 ทำให้การนับค่าเกิดขึ้นตั้งแต่ค่า TL1 ไปจนถึง FFH แล้วจึงเซตบิต TF1 พร้อมกับทำการส่งค่าใน TH1 ไปยัง TL1 อีกครั้ง เรียกกระบวนการนี้ว่า การตั้งค่าอัตโนมัติ (reload) ส่วนในไทม์เมอร์/เคาน์เตอร์ 0 จะเหมือนกับไทม์เมอร์/เคาน์เตอร์ 1 ทุกประการ
  - '11' สำหรับไทม์เมอร์ 0 เลือกให้ทำงานในโหมดไทม์เมอร์/เคาน์เตอร์แยกส่วน โดยแยกออกเป็นไทม์เมอร์/เคาน์เตอร์ 8 บิต 2 ตัว รีจิสเตอร์ TL0 จะได้รับการควบคุมการเปิดปิดจากบิต TR0 ในรีจิสเตอร์ TH0 และรีจิสเตอร์ TH0 ซึ่งเป็นไทม์เมอร์/เคาน์เตอร์ 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1 ในรีจิสเตอร์ TCON สำหรับไทม์เมอร์ 1 เป็นการสั่งให้หยุดการทำงาน

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
ไทเมอร์ 1				ไทเมอร์ 0			

รูปที่ 2-10 รีจิสเตอร์เลือกโหมดการทำงานของไทม์เมอร์/เคาน์เตอร์

2.6 อินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ AT89S8252

การอินเตอร์รัปต์ เป็นชื่อเรียกกระบวนการที่เข้ามาขัดจังหวะการทำงานโดยปกติของไมโครคอนโทรลเลอร์ ภายในไมโครคอนโทรลเลอร์ AT89S8252 สามารถตอบสนองการเกิดอินเตอร์รัปต์ได้ 6 แหล่ง ประกอบด้วย การรับสัญญาณอินเตอร์รัปต์จากภายนอกผ่านทางขา INT0 และ INT1 สัญญาณอินเตอร์รัปต์จากไทม์เมอร์/เคาน์เตอร์ T0 และ T1 สัญญาณอินเตอร์รัปต์จากพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ และ สัญญาณอินเตอร์รัปต์จากไทม์เมอร์/เคาน์เตอร์ 2

เมื่อเกิดการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ แล้วมีการเอนาเบิลการตอบสนองต่ออินเตอร์รัปต์นั้นไว้ กระบวนการหลังจากนั้นซีพียูจะทำการกระโดดไปยังแอดเดรสในหน่วยความจำที่กำหนดไว้ ซึ่งเรียกว่า แอดเดรสอินเตอร์รัปต์เวกเตอร์ (Interrupt Address Vector) ดังนั้นจึงต้องมีการเขียนโปรแกรมย่อยการบริการอินเตอร์รัปต์ไว้ที่อินเตอร์รัปต์เวกเตอร์แอดเดรสนั้น โดยที่ค่าแอดเดรสของอินเตอร์รัปต์เวกเตอร์ของแต่ละอินเตอร์รัปต์ มีรายละเอียดดังนี้

- อินเตอร์รัปต์จากขา INT0 มีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 0003H
- อินเตอร์รัปต์จากไทม์เมอร์ 0 มีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 000BH
- อินเตอร์รัปต์จากขา INT1 มีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 0013H
- อินเตอร์รัปต์จากไทม์เมอร์ 1 มีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 001BH
- อินเตอร์รัปต์จากพอร์ตอนุกรมมีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 0023H
- อินเตอร์รัปต์จากไทม์เมอร์ 2 มีค่าแอดเดรสอินเตอร์รัปต์อยู่ที่ 002BH

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ AT89S8252

การอินเตอร์รัปต์ภายในไมโครคอนโทรลเลอร์ มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว คือ รีจิสเตอร์เอนาเบิลการอินเตอร์รัปต์ (IE : Interrupt Enable register) และ รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปต์ (IP : Interrupt Priority register)

1. รีจิสเตอร์เอนาเบิลการอินเตอร์รัปต์ (IE : Interrupt Enable register)

มีแอดเดรสอยู่ที่ A8H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ SFR มีขนาด 8 บิต สามารถเข้าถึงระดับบิตได้ รูปที่ 2-11 แสดงการเรียงลำดับของรีจิสเตอร์เอนาเบิลการอินเตอร์รัปต์ โดยมีรายละเอียดของแต่ละบิตดังนี้

- EA (Global enable/disable interrupt) ใช้ในการเอนาเบิลและดิสเอเบิลการตอบสนองการอินเตอร์รัปต์ทั้งหมด '0' เป็นการดิสเอเบิลการอินเตอร์รัปต์ แต่ถ้าเป็น '1' เป็นการเอนาเบิล การอินเตอร์รัปต์
- ET2 (Timer 2 interrupt enable) ใช้ในการเอนาเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวหรือการแคปเจอร์ในไทม์เมอร์/เคาน์เตอร์ สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- ES (Serial port interrupt enable) ใช้ในการเอนาเบิลการอินเตอร์รัปต์อันเนื่องมาจากการรับหรือส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- ET1 (Timer interrupt enable) ใช้ในการเอนาเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทม์เมอร์/เคาน์เตอร์ 1 สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- EX1 (External interrupt 1 enable bit) ใช้ในการเอนาเบิลการอินเตอร์รัปต์เนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT1 สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- ET0 (Timer 0 interrupt enable) ใช้ในการเอนาเบิลการอินเตอร์รัปต์เนื่องมาจากการโอเวอร์โฟลวในไทม์เมอร์/เคาน์เตอร์ 0 สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- EX0 (External interrupt 0 enable) ใช้ในการเอนาเบิลการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT0 สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
EA	-	ET2	ES1	ET1	EX1	ET0	EX0

รูปที่ 2-11 รีจิสเตอร์เอนาเบิลการอินเตอร์รัปต์

2. รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปต์ (IP : Interrupt Priority register)

มีแอดเดรสอยู่ที่ B8H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ SFR มีขนาด 8 บิต ทุกบิตสามารถเซ็ตหรือเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์ สามารถเข้าถึงระดับบิตได้ ใช้ในการเลือกลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า ต้องการให้ตอบสนองสัญญาณอินเตอร์รัปต์จากแหล่งกำเนิดใดเป็นลำดับก่อนหลัง ถ้าต้องการให้แหล่งกำเนิดใดมีความสำคัญสูง

สุด ให้กำหนดที่บิตนั้นเป็น '1' ดังรูปที่ 2-12 แสดงการจัดเรียงบิตลำดับความสำคัญ โดยมีรายละเอียดดังนี้

- PT2 (Timer 2 interrupt priority) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากโอเวอร์โฟลวหรือการแคปเจอร์ในไทม์เมอร์/เคาน์เตอร์ 2
- PS (Serial port interrupt priority bit) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการรับหรือส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์
- PT1 (Timer 1 interrupt priority bit) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทม์เมอร์/เคาน์เตอร์ 1
- PX1 (External interrupt 1 priority bit) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการสัญญาณภายนอกที่ป้อนเขามายังขา INT1
- PT0 (Timer 0 interrupt priority bit) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทม์เมอร์/เคาน์เตอร์ 0
- PX0 (External interrupt 0 priority bit) ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการสัญญาณภายนอกที่ป้อนเขามายังขา INT0

สำหรับบิต 6 และ 7 ของรีจิสเตอร์ IP ไม่มีการใช้งาน ต้องกำหนดให้เป็น '0' เสมอ

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

รูปที่ 2-12รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปต์

ลำดับความสำคัญของการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ในไมโครคอนโทรลเลอร์ AT89S8252

ในกรณีที่สัญญาณอินเตอร์รัปต์เกิดขึ้นพร้อมกันภายในไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์จะต้องมีการเลือกว่าจะเข้าไปให้บริการแก่อินเตอร์รัปต์ใด โดยปกติสามารถกำหนดได้ที่รีจิสเตอร์ IP แต่ในกรณีที่ มีการเซ็ตค่าบิตของรีจิสเตอร์กำหนดลำดับความสำคัญให้เป็น 1 ในทุกบิต ไมโครคอนโทรลเลอร์จะมีการเรียงลำดับความสำคัญจากมากไปน้อยดังนี้

- อินเตอร์รัปต์ภายนอกที่ขา INT0 หรือการเซ็ตของบิต IE0
- อินเตอร์รัปต์จากไทม์เมอร์ 0 หรือการเซ็ตของบิต TF0
- อินเตอร์รัปต์ภายนอกที่ขา INT1 หรือการเซ็ตของบิต IE1
- อินเตอร์รัปต์จากไทม์เมอร์ 1 หรือการเซ็ตของบิต TF1
- อินเตอร์รัปต์จากพอร์ตอนุกรม หรือการเซ็ตของบิต RI หรือ TI

- อินเตอร์รัปต์จากไทม์เมอร์ 2 หรือการเซ็ตของบิต TF2 หรือ EXF2

2.7 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ AT89S8252

ไมโครคอนโทรลเลอร์ AT89S8252 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD ซึ่งเป็นการสื่อสารแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัส

เป็นการรับส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

- บิตเริ่มต้น มีขนาด 1 บิต
- บิตข้อมูลอนุกรม มีขนาด 8 บิต
- บิตตรวจสอบพาริตี้ มีขนาด 1 บิต หรือ ไม่มี
- บิตหยุด มีขนาด 1 บิต

รูปที่ 2-13 แสดงรูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส จะมีสถานะหยุดรอก่อนที่จะทำการส่งและหลังจากการส่งเสร็จสิ้นแล้ว

รูปที่ 2-13 รูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน AT89S8252

มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว คือ รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม (SBUF : Serial data buffer register) และ รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม (SCON : Serial port Control Register)

1. รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ทำการส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ในไมโครคอนโทรลเลอร์โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และ รีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลยายังรีจิสเตอร์ SBUF ข้อมูลนั้น จะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ ขา P3.1 เมื่อมีการรับข้อมูลจากขา RxD หรือ ขา P3.0 ข้อมูลจะถูกนำมาเก็บไว้ในบัฟเฟอร์สำหรับรับข้อมูล แล้วจึงมีการอ่านข้อมูลจากบัฟเฟอร์ต่อไป

2. รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม

มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ SFR สามารถเข้าถึงในระดับบิตได้ รูปที่2-14 แสดงการจัดเรียงบิตของรีจีสเตอร์ โดยมีรายละเอียดดังนี้

- SM0-SM1 (Serial port mode bit 0 – 1) ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์
- SM2 ใช้ในการเอนาเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ ในโหมด 2 และ 3 ถ้าบิตนี้เป็น '1' บิต RI จะไม่แอกทีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น '0' (ข้อมูลบิตที่ 9 ไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอกทีฟถ้ายังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน
- REN (Enable serial reception) ใช้ในการเอนาเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซ็ตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซ็ตบิตนี้ให้เป็น '1'
- TB8 ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- RB8 ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น '0' ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งาน สามารถเซ็ตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์
- TI ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรส่งข้อมูลเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ สามารถเซ็ตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 เรียบร้อยแล้ว ในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิต

นี้จะเซ็ตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

- RI ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ สามารถเซ็ตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำงานรับข้อมูลบิตที่ 8 เรียบร้อยแล้ว ในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซ็ตเมื่อสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซ็ตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

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

รูปที่ 2-14 รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม

โหมดการทำงานของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ AT89S8252

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์สามารถเลือกโหมดการทำงานได้ถึง 4 โหมดคือ

- โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
- โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
- โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
- โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานสามารถเลือกได้จากรีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมที่บิต SM0 และ SM1

การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

การสื่อสารข้อมูลผ่านพอร์ตอนุกรมในโหมด 1 นั้นข้อมูลจะเข้าและออกจากไมโครคอนโทรลเลอร์ทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (Shift Clock) ในโหมด 0 นี้มีจำนวนข้อมูล 8 บิตโดยทำการรับและส่งข้อมูลในบิตที่มีนัยสำคัญต่ำที่สุด (LSB) ก่อน โดยค่า Baud Rate ถูกกำหนดไว้คงที่ ที่ 1/12 ของความถี่สัญญาณนาฬิกา

กระบวนการเริ่มต้นสำหรับการส่งข้อมูลผ่านทางพอร์ตสื่อสารอนุกรมนั้น ผู้เขียนโปรแกรมจะต้องนำข้อมูลที่จะส่งออกไปมาเก็บไว้ที่รีจิสเตอร์ SBUF จากนั้นข้อมูลในรีจิสเตอร์ SBUF ก็จะถูกเลื่อนออกที่ขา P3.0 หรือขา RxD ครั้งละหนึ่งบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD ไมโครคอนโทรลเลอร์จะส่งข้อมูลออกไปจนกระทั่งครบ 8 บิต จากนั้นบิต TI ในรีจิสเตอร์ SCON ก็จะเกิดการเซ็ตค่าให้เป็น 1 เพื่อแจ้งให้ทราบว่าได้ทำการส่ง

ข้อมูลครบแล้ว และถ้าหากอินเทอร์รัพท์ของพอร์ตอนุกรมได้รับการเซ็ตให้มีการยอมรับไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบด้วย

สำหรับกระบวนการในการรับข้อมูล จะต้องเคลียร์ค่าของบิต RI ก่อน จากนั้นข้อมูลก็จะถูกส่งเข้ามายังไมโครคอนโทรลเลอร์ผ่านทางขา RxD เช่นเดียวกัน โดยจังหวะการเลื่อนของข้อมูลเข้านั้นจะกำหนดโดยขา TxD เมื่อไมโครคอนโทรลเลอร์ได้รับข้อมูลจนครบทั้ง 8 บิตแล้ว บิต RI ก็จะได้รับการเซ็ตเพื่อแจ้งว่าได้เสร็จสิ้นกระบวนการรับข้อมูลแล้ว และเช่นเดียวกัน ถ้าอินเตอร์รัปต์สำหรับพอร์ตอนุกรมได้รับการยอมรับไว้ ก็จะทำให้เกิดการอินเตอร์รัปต์ขึ้นในระบบ

การทำงานในโหมด 0 ของพอร์ตสื่อสารอนุกรมในไมโครคอนโทรลเลอร์นี้จะใช้ประโยชน์ในการเชื่อต่อกับอุปกรณ์ IC ที่ทำหน้าที่เป็นรีจิสเตอร์ภายนอกสำหรับการขยายจำนวนพอร์ตอินพุตหรือเอาท์พุท แต่ในการใช้งานส่วนใหญ่ ผู้เขียนโปรแกรมก็ไม่ได้ใช้งานโหมด 0 นี้สักเท่าใดนัก ทั้งนี้เนื่องจากว่าตัวไมโครคอนโทรลเลอร์เองนั้นก็มีพอร์ตอินพุตเอาท์พุทอยู่ค่อนข้างพอเพียงต่อการใช้งานแล้ว และการติดต่อกับพอร์ตมาตรฐานก็สามารถทำได้ง่ายกว่ามาก

การทำงานในโหมด 1 ของวงจรพอร์ตอนุกรม

โหมดการสื่อสารข้อมูลผ่านพอร์ตอนุกรมในโหมด 1 ของไมโครคอนโทรลเลอร์ นั้นจะมีการรับและส่งข้อมูลทั้งหมด 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตนั้นประกอบด้วย บิตเริ่มต้น (Start Bit) ที่มีค่าเป็น 0 จำนวน 1 บิต ต่อด้วยบิตข้อมูลอีก 8 บิต โดยรับหรือส่งข้อมูลที่มีนัยสำคัญต่ำที่สุด (LSB) ก่อน และตามด้วยบิตหยุด (Stop Bit) ที่มีค่าเป็น 1 อีกจำนวน 1 บิต สำหรับการรับข้อมูลนั้น บิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON ในส่วนของค่า Baud Rate ในโหมดสื่อสารอนุกรมนี้จะได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทม์เมอร์ 1

กระบวนการส่งข้อมูลจะเริ่มด้วยการนำข้อมูลไปใส่ไว้ในรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูลลงในรีจิสเตอร์ SBUF จะทำให้วงจรควบคุมดำเนินกระบวนการส่งข้อมูลในรีจิสเตอร์ SBUF ตามจังหวะจากสัญญาณนาฬิกาการส่ง (TX Clock) เป็นตัวกำหนด สัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จากไทม์เมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปิดท้ายขนาด 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซ็ตค่าบิต TI ในรีจิสเตอร์ SCON หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการยอมรับไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบด้วย

หลังจากที่ไมโครคอนโทรลเลอร์ได้ทำงานในส่วนของโปรแกรมย่อยที่รองรับอินเตอร์รัปต์ (ISR) หรือส่งข้อมูลเรียบร้อยแล้ว ผู้เขียนโปรแกรมต้องสั่งให้โปรแกรมเคลียร์ค่าบิต TI ก่อน ทั้งนี้เพื่อให้โปรแกรมสามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมต่อๆ ไปได้

ในส่วนของการรับข้อมูลนั้น ไมโครคอนโทรลเลอร์จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจากค่า 1 เป็น 0 ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของค่าอัตราบอด เมื่อ

ตรวจจับพบ ไทม์เมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดค่าอัตราบอดจะรีเซ็ต และทำการเขียนข้อมูล 1FFH ไปยังชิฟรีจิสเตอร์ที่เป็นอินพุต

ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามามีค่าเป็น 1 หรือ 0 จะใช้ผลจากการสุ่ม โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สถานะ การสุ่มข้อมูลจะทำการสุ่มสถานะที่ 7, 8 และ 9 และหากตรวจดูว่า 2 ใน 3 ครั้งของการสุ่มข้อมูลมีค่าเป็นเท่าใด ก็จะตีความข้อมูลในบิตนั้นตามเสียงข้างมาก ยกตัวอย่างเช่น ถ้าสุ่มพบค่า 1 จำนวน 2 ครั้ง (จาก 3 ครั้ง) ไมโครคอนโทรลเลอร์ก็จะตีความว่าบิตข้อมูลที่ได้รับเข้ามานั้นมีค่าเป็น 1

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูล คือเริ่มต้นด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และจบด้วยบิตหยุดในทุกๆ การรับข้อมูล 1 บิต จะมีสัญญาณเลื่อน เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์ โดยการกำหนดจังหวะการรับข้อมูลจะใช้สัญญาณนาฬิกาสำหรับการรับข้อมูล (RX Clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย ซึ่งหมายความว่าไมโครคอนโทรลเลอร์ได้รับข้อมูลครบเรียบร้อยแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF พร้อมทั้งเซ็ตค่าบิต RB8 ในรีจิสเตอร์ SCON ซึ่งข้อมูลในบิต RB8 นี้ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นก็จะทำการเซ็ตบิต RI ในรีจิสเตอร์ SCON ด้วย และหากอินเตอร์รัปต์จากพอร์ตสื่อสารอนุกรมได้รับการยอมรับไว้ ก็จะทำให้เกิดการอินเตอร์รัปต์ขึ้นในระบบด้วย

หลังจากที่ไมโครคอนโทรลเลอร์ได้ทำงานในส่วนของโปรแกรมย่อยที่รองรับอินเตอร์รัปต์ (ISR) แล้วผู้เขียนโปรแกรมต้องสั่งให้โปรแกรมเคลียร์ค่าบิต RI ก่อน ทั้งนี้เพื่อให้โปรแกรมสามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมต่อๆ ไปได้

การสื่อสารข้อมูลผ่านพอร์ตอนุกรมในโหมด 1 นี้ได้รับความนิยมโดยนักพัฒนาระบบส่วนใหญ่มากที่สุด ทั้งนี้เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

การทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรม

การสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ในโหมด 2 และ 3 นั้นจะมีการสื่อสารข้อมูลทั้งหมด 11 บิต ประกอบด้วยStart Bit ซึ่งมีค่าเป็น 0 จำนวน 1 บิต บิตข้อมูล 8 บิต โดยจะทำการส่งข้อมูลในบิต LSB ก่อน บิตข้อมูลบิตที่ 9 ซึ่งเป็น Parity Bit และบิตสุดท้ายคือ Stop Bit ซึ่งมีค่าเป็น 1 จำนวน 1 บิต ในการส่งข้อมูล บิตข้อมูลบิตที่ 9 นั้นจะเก็บไว้ในบิต TB8 ของรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไปเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับค่า Baud Rate ในโหมดที่ 2 จะคงที่ โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 นั้นค่า Baud Rate จะสามารถปรับได้เหมือนกับในโหมด 1

บทที่ 3
ภาษาวีเอชดีแอล

วีเอชดีแอล ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC: Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (high level language) ที่ใช้ในการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (hierarchy) ได้ และสามารถที่จะเขียนได้หลายรูปแบบซึ่งจะกล่าวต่อไป จึงทำให้ภาษาวีเอชดีแอล เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาสามารถเปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยที่ยังไม่ต้องไปคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้นวีเอชดีแอลยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมด จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งาน

3.1 ประวัติความเป็นมาของภาษาวีเอชดีแอล

วิวัฒนาการของภาษาวีเอชดีแอล นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ ดีโอดี (DoD: Department of Defense) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร เป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อน เพราะเทคโนโลยีในขณะนั้นทำให้การพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่างล่าช้า ซึ่งเป็นสภาพที่ไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่จะเห็นได้ว่ามีวงจรดิจิตอลอิเล็กทรอนิกส์หลายวงจร ที่แต่เดิมถูกสร้างขึ้นมาจากชิ้น ถูกนำประกอบกันอยู่บนแผงวงจรไฟฟ้าที่มีขนาดใหญ่ แต่ในปัจจุบันสามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก (VLSI: Very Large Scale Integration) รวมอุปกรณ์ต่างๆ เหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำ ที่มีขนาดประมาณ 1-2 ตร.ซม. ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น (ความเร็วในการทำงานของวงจร) ตลอดจนความน่าเชื่อถือในการทำงาน และความคงทนต่อสภาพแวดล้อมสูง ขณะเดียวกันนั้นในวงการทหารได้มีการนำระบบคอมพิวเตอร์และอิเล็กทรอนิกส์ มาใช้ในระบบอาวุธอย่างแพร่หลาย ดังนั้นอุปกรณ์ที่มีใช้อยู่จึงไม่เหมาะสมกับเทคโนโลยีด้านอาวุธของประเทศคู่แข่งขัน การที่จะเปลี่ยนอุปกรณ์ใหม่เป็นสิ่งที่ต้องใช้งบประมาณมาก และก็จะประสบกับปัญหาเช่นเดิมคือ อุปกรณ์ใหม่ได้รับการพัฒนามานานแล้วเช่นกัน เพราะในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกร และเวลาสำหรับดำเนินการมาก ฉะนั้นทาง ดีโอดีจึงตั้งโครงการขึ้นมาเพื่อศึกษา วิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอลให้สามารถนำไปผลิตได้เร็วขึ้น และโครงการดังกล่าวมีชื่อว่า “Very High Speed Integrated Circuits” หรือ วีเอชเอสไอซี (VHSIC) ในระยะแรกนั้นโครงการเป็นความลับทางด้านความมั่งคงของประเทศ และอยู่ในความดูแลควบคุมของ United States International Traffic and Arms Regulations หรือ ไอทีเออาร์ (ITAR)

ในปี ค.ศ. 1983 ตามคำแนะนำของคณะทำงาน ("Woods Hole" workshop) ทาง ดีโอดีได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการวีเอชเอสไอซี ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรมการออกแบบฮาร์ดแวร์เรียกว่า "Hardware Description Language" หรือ เอชดีแอล (HDL) เริ่มต้นโครงการดีโอดีได้มอบหมายให้บริษัทไอบีเอ็มและบริษัทเท็กซัลอินสตูลเมนท์และบริษัทอินเตอร์เมทริกซ์เป็นผู้ศึกษาและพัฒนา การดำเนินการได้กระทำไปอย่างต่อเนื่อง และได้ผลเป็นที่น่าพอใจจนกระทั่งปี ค.ศ. 1985 ทางไอทีเออาร์ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหาร ออกจากโครงการนี้ ดังนั้นภาษาวีเอชดีแอล จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป จนกระทั่งทางไออีอีอี (IEEE) จึงได้รับภาษานี้เข้ามาศึกษาและประมาณปี ค.ศ. 1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 และมีชื่อเรียกว่าวีเอชดีแอลมาตรฐานนี้ก็ได้รับการปรับปรุงจนปัจจุบัน ได้ชื่อว่า IEEE 1076-1993 หรือ วีเอชดีแอล 1993 การที่ทางดีโอดีในขณะนั้น เป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ จึงมีผู้รับโครงการต่างๆ จากดีโอดี ไปดำเนินการด้านวิจัยและพัฒนามากเพื่อที่จะให้เป็นมาตรฐานเดียวกันหมด ทางดีโอดี จึงกำหนดว่า ในการส่งโครงการนั้นจะต้องเขียนอยู่ในรูปของภาษาวีเอชดีแอลเท่านั้น ซึ่งทำให้เกิดข้อดีต่อดีโอดีเองที่เป็นมาตรฐานเดียวกัน สามารถนำไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆ ระบบ

3.2 ส่วนประกอบต่างๆ ของภาษาวีเอชดีแอล

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะบนลงล่าง จะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆ ของรูปแบบภาษาวีเอชดีแอลเสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

- หน่วยการออกแบบเอนทิตี (Entity Design Unit)
- หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)
- หน่วยการออกแบบแพ็กเก็ต (Package Design Unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

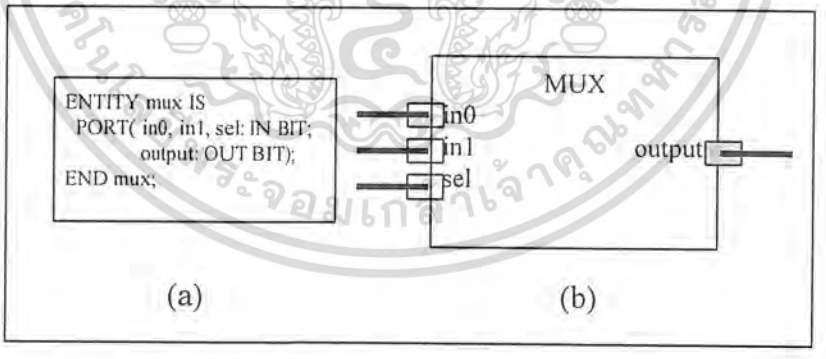
3.2.1 หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้น ที่เรียกว่า หน่วยการออกแบบเอนทิตี ในส่วนนี้ใช้กำหนดจุดเชื่อมต่อ ของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ ของข้อมูลที่ไหล่ผ่านจุดต่อเหล่านั้น รูปที่ 3-1 แสดงให้เห็นโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบเอนทิตี

```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name];
```

รูปที่ 3-1 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำ ENTITY และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ ส่วนนี้เรียกว่าส่วนหัว (entity header) และที่สำคัญคือ หน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาคเสมอ (;)

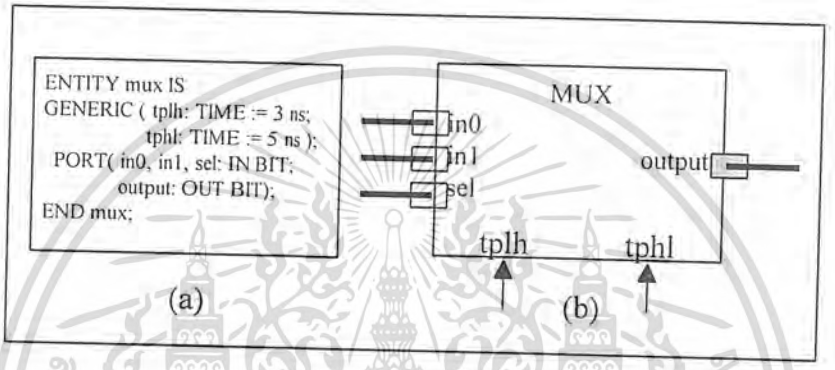


รูปที่ 3-2 แสดงรูปแบบของมัลติเพลกซ์
(a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อมประสาน

ในรูปที่ 3-2 เป็นหน่วยการออกแบบเอนทิตี ที่บรรยายอุปกรณ์ที่มีชื่อว่ามัลติเพลกซ์ หรือ MUX ในส่วนหัวของเอนทิตี มีการกำหนดจุดต่อ 4 จุดภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ in0, in1, sel ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการ

ไหลเข้าของข้อมูล (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่ 2-2 ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในรูปที่ 2-2 ส่วนประเภทของข้อมูลที่จะไหล เข้าและออก นั้นเป็นประเภท BIT ที่สามารถมีค่าได้เพียงสองค่าคือ '0' และ '1' เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามาถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆ ลงในส่วนหัวของเอนทิตีได้อีก เช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์ อันได้แก่ ค่าเวลาหน่วงแพร่กระจาย (Propagation delay time) พารามิเตอร์เหล่านี้ เรียกว่า เจนเนริก (Generic) ที่กำหนดด้วยคำสั่ง GENERIC จากตัวอย่างในรูปที่ 3-3



รูปที่ 3-3 รูปแบบมัลติเพลกซ์ที่ประกอบด้วยข้อมูลค่าเวลาหน่วงแพร่กระจาย
(a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อมประสาน

ในบางกรณีสามารถที่จะใช้ภาษาวีเอชดีแอล สร้างรูปแบบที่ปราศจากช่องทางไหล เข้าและออกของข้อมูล ได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบ สำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง คือ วีเอชดีแอลสำหรับการทดสอบเปรียบเทียบ (Test bench)

```
ENTITY test_bench IS
END test_bench;
```

รูปที่ 3-4 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

3.2.2 หน่วยการออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงานพฤติกรรมต่างๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออก ตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ ที่กำหนดใน หน่วยการออกแบบเอนทิตี รูปที่ 3-5 แสดงให้เห็นถึงโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบสถาปัตยกรรม

```
ARCHITECTURE identifier OF component_name IS
[declaration]
BEGIN
   specification of the functionality of the
   component in terms of its input lines and as
   influenced by physical and other parameters
END [identifier];
```

รูปที่ 3-5 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

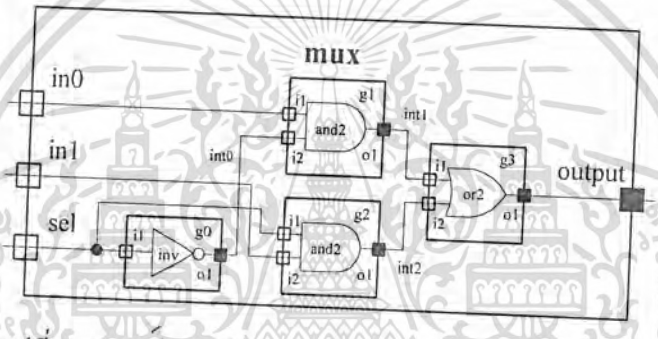
ส่วนของหน่วยการออกแบบสถาปัตยกรรม เริ่มต้นด้วยคำ ARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า ARCHITECTURE นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใดๆ (OF <entity design unit> IS) ส่วนที่อยู่ระหว่าง ARCHITECTURE และ BEGIN เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (architecture declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆ ที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ อาทิเช่นประเภท (type) ต่างๆ (ตัวอย่างเช่น bit, bit_vector), สัญญาณ (signal), ตัวคงที่ (constant), โปรแกรมย่อย (ได้แก่ function และ procedure) และอุปกรณ์ (component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง PORT) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGIN กับ END ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น หน่วยการออกแบบสถาปัตยกรรม จะต้องปิดท้ายด้วยคำสั่ง END และชื่อของสถาปัตยกรรมนั้นๆ ที่เป็นส่วนเผื่อเลือกโดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอล สามารถเขียนได้ในลักษณะต่างๆ ดังนี้

- ประเภทการไหลของข้อมูล (Dataflow description)
- ประเภทพฤติกรรม (Behavioral description)
- ประเภทโครงสร้าง (Structure description)
- ประเภทผสม (Mixed model description)

```
ARCHITECTURE data_flow OF mux IS
BEGIN
   output <= ((NOT sel) AND in0) OR (sel AND in1);
END data_flow;
```

รูปที่ 3-6 แสดงหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์

รูปที่ 3-6 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (*in0, in1*) กับข้อมูลที่ไหลออก (*output*) ประกอบด้วยชุดคำสั่งแบบแข่งขนานเพียงชุดเดียว ซึ่งเขียนเป็นประเภทการไหลของข้อมูลของ มัลติเพลกซ์ หรือ ระดับการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ (RTL: Register Transfer Level)



รูปที่ 3-7 แสดงโครงสร้างภายในสถาปัตยกรรมของมัลติเพลกซ์

รูปที่ 3-7 เป็นหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง โดยใช้ อินเวอร์เตอร์ (inv ที่ตำแหน่ง g0), แอนด์เกต 2 อินพุตจำนวน 2 ตัว (and2 ที่ตำแหน่ง g1 และ g2) และ ออร์เกต 2 อินพุต (or2 ที่ตำแหน่ง g3) มาสร้างตามฟังก์ชันบูลีนของรูปที่ 3-6

รูปที่ 3-8 และ 3-9 (ในหน้าถัดไป) เป็นการบรรยายส่วนสถาปัตยกรรมของมัลติเพลกซ์ในลักษณะแบบโครงสร้าง และแบบพฤติกรรม จากตัวอย่างที่ได้ยกมาจะเป็นว่า ในภาษาวีเอชดีแอลนั้น วงจรเดียวกันสามารถบรรยายได้หลายแบบซึ่งแต่ละแบบให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ในการบรรยายวงจรนั้น ผู้ออกแบบสามารถเลือกได้ว่าจะบรรยายในลักษณะใด ขึ้นอยู่กับความถนัด และ มุมมอง ยกตัวอย่างเช่น การบรรยายรถหนึ่งคันออกมาด้วยคำพูดนั้นสามารถบรรยายถึงโครงสร้างของรถ หรือ จะบรรยาย หน้าที่และลักษณะการทำงานของรถก็ได้ขึ้นอยู่กับความถนัด และ ความเหมาะสม

```
ARCHITECTURE struc OF mux IS
   COMPONENT inv
   PORT ( i1 : IN BIT ; o1 : OUT BIT );
    COMPONENT and2
    PORT ( i1, i2 : IN BIT ; o3 : OUT BIT );
    COMPONENT or2
    PORT ( i1, i2 : IN BIT ; o1 : OUT BIT );
END COMPONENT;
    SIGNAL int0, int1, int2 : BIT;
BEGIN
    g0 : inv   PORT MAP (i1 => sel, o1 => int0);
    g1 : and2 PORT MAP (i1 => in0, i2 => int0, o1 => int1);
    g3 : or2   PORT MAP (i1 => int1, i2 => int2, o1 => ouput);
END struc;
```

รูปที่ 3-8 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง

```
ARCHITECTURE behav OF mux IS
BEGIN
     PROCESS (in0, in1, sel)
     BEGIN
          IF (sel = '0') THEN output <= in0;
          ELSE output <= in1;
          END IF;
     END PROCESS;
END behav;
```

รูปที่ 3-9 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทพฤติกรรม

3.2.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กแก็ตได้ และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดยหน่วยการออกแบบเอนทิตี หน่วยการออกแบบสถาปัตยกรรม หรือ จากหน่วยการออกแบบแพ็ก

แก็ตอื่นๆ นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบมาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) จะถูกเก็บไว้ในแพ็กแก็ต ที่ทุกคนสามารถเข้าถึง โดยปกติแล้ว แพ็กแก็ตจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กแก็ต (Package declaration) และ ส่วนของบอดี้แพ็กแก็ต (Package body) เนื่องจาก แพ็กแก็ตถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กแก็ตไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอล สามารถกระทำได้ด้วยชุดคำสั่ง USE

- **PACKAGE DECLARATION**

ส่วนที่มีความสำคัญที่สุดของแพ็กแก็ต (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนการประกาศแพ็กแก็ตเพราะ เป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กแก็ตสำหรับนำไปใช้ภายนอกตัวของแพ็กแก็ตเอง สิ่งใดๆ ถูกประกาศในส่วนของ ส่วนบอดี้แพ็กแก็ต แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กแก็ต จะไม่สามารถถูกนำค่า และพฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตีคือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้ว แพ็กแก็ต สามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือ สัญญาณ เช่นเดียวกันกับ ส่วนบอดี้แพ็กแก็ต ที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กแก็ต แต่แพ็กแก็ต นั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE package_name IS
    Package_declarative_part
END package_name;
```

รูปที่ 3-10 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กแก็ต

- **PACKAGE BODY**

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ ที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (Subprogram) ทั้งหลายที่ชื่อของโปรแกรมย่อยนั้นๆ ที่ถูกประกาศไปในส่วนของการประกาศแพ็กแก็ต แล้วจะถูกเก็บไว้ในส่วนบอดี้แพ็กแก็ต ทั้งนี้รวมทั้ง การกำหนดค่าคงที่ต่างๆ อันได้แก่ตัวคงที่ที่ถูกประกาศชื่อก่อนในส่วนของการประกาศแพ็กแก็ต แต่ถูกกำหนดค่าในส่วนของบอดี้แพ็กแก็ต) ฉะนั้นส่วนบอดี้แพ็กแก็ต จึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กแก็ต ไม่มีการประกาศชื่อ ที่เป็นโปรแกรมย่อย หรือ ค่าคงที่ การเขียนบอดี้แพ็กแก็ตนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 3-11

```
PACKAGE BODY package_name IS
    declarative part
END package_name;
```

รูปที่ 3-11 โครงสร้างของบอดี้แพ็กเก็ต

3.2.4 หน่วยการออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนทิตีได้ เพียงหนึ่งเดียวเท่านั้น แต่ในขณะที่หน่วยการออกแบบเอนทิตี หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรม ที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมี หน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

```
CONFIGURATION identifier OF entity_name IS
    Configuration_declarative_part
END ;
```

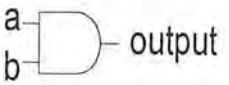
รูปที่ 3-12 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

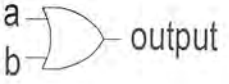
3.3 ภาษาวีเอชดีแอลเพื่อการสังเคราะห์

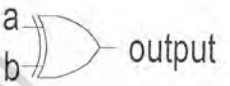
ภาษาวีเอชดีแอล เป็นภาษาที่เขียนเพื่อจำลองการทำงานของวงจร ซึ่งในบางรูปแบบการเขียนไม่สามารถที่จะนำไปสังเคราะห์ได้ทั้งหมด ดังนั้นถ้าต้องการเขียนเพื่อนำไปสังเคราะห์ ควรหลีกเลี่ยงรูปแบบต่างๆ ที่ไม่สามารถนำไปสังเคราะห์ได้ ในที่นี่ขึ้นอยู่กับความสามารถของโปรแกรมที่ใช้สังเคราะห์แต่ละโปรแกรม ดังนั้นในหัวข้อนี้จะแสดงตัวอย่างของการเขียนโมเดลในรูปแบบต่างๆ ที่สามารถนำไปสังเคราะห์ ซึ่งยึดหลักการเขียนตาม ViewSynthesis User's Guide ของโปรแกรม Viewlogic ซึ่งเป็นโปรแกรมที่ใช้ในสังเคราะห์วงจรทั้งหมดในการออกแบบไมโครคอนโทรลเลอร์ โดยแบ่งออกเป็น 2 กลุ่ม คือ เกตและฟลิปฟลอปประเภทต่างๆ

ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

```
SIGNAL a, b, c, d, input, output : vlbit_1d(3 DOWNTO 0);
SIGNAL sel  : vlbit_1d(1 DOWNTO 0);
SIGNAL enb : vlbit;
...
output <= a AND b;
...

output <= a OR b;
...

output <= a XOR b;
...

output <= NOT input;
...
output <= input WHEN enb='1' ELSE "ZZ";
...

WITH sel SELECT
        output <=   a WHEN "00" ELSE
                    b WHEN "01" ELSE
                    c;
```

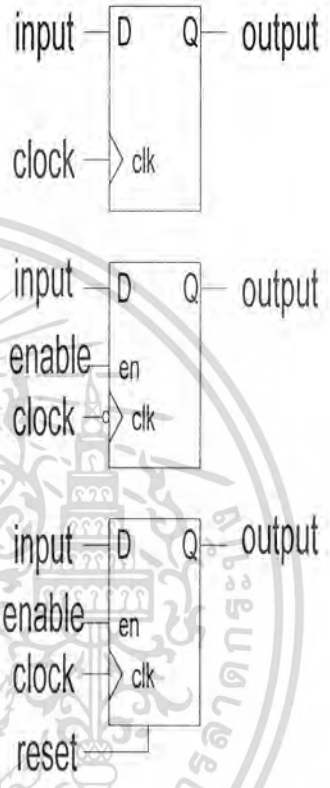






ตัวอย่างรูปแบบการเขียนฟลิปฟลอปพื้นฐาน

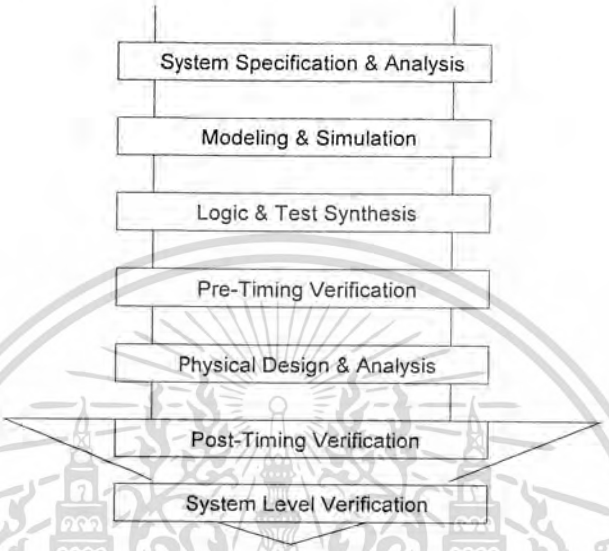
```
SIGNAL input, output : vlbit_1d(3 DOWNTO 0);
SIGNAL clock, enable, reset : vlbit;
...

PROCESS BEGIN
        WAIT UNTIL PRISING(clock);
                output <= input;
END PROCESS;
...
PROCESS BEGIN
        WAIT UNTIL PFALLING(clock);
                output <= input;
END PROCESS;

PROCESS BEGIN
WAIT UNTIL PRISING(clock) OR (reset='1');
  IF (reset = '1') THEN output <= "00";
  ELSIF (enble = '1') THEN ouput <= input;
  END IF;
END PROCESS;
```



3.4 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น วงจรรวม (ASIC: Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของของ บล็อกไดอะแกรมเสียก่อน ก่อนที่จะวิเคราะห์ให้ลึกถึงรายละเอียดต่อไปซึ่งภาษาวีเอชดีแอลนั้น อนุญาติให้อธิบายการทำงานของแต่ละบล็อก และวิเคราะห์การทำงาน แก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ และเพิ่มเติมในรายละเอียดที่ละชั้นนี้คือ หลักการออกแบบจากบนลงล่าง (Top-Down Design) ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Botom-Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เพราะเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ จำลองการทำงาน ตรวจสอบ

ความถูกต้อง ซึ่งใช้เวลามาก และถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนก็จะเป็นเรื่องที่ยากมากให้การออกแบบในลักษณะนี้ ดังนั้นการใช้ภาษาวีเอชดีแอลกับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรออกแบบที่จะสามารถออกแบบและพัฒนาวงจรที่มีซับซ้อนได้มากขึ้น และช่วยลดเวลาและค่าใช้จ่ายในการออกแบบ



รูปที่ 3-13 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 3-13 แสดงให้ถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย ก็เนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายๆ เทคโนโลยี เช่น พีแอลดี (PLD: Programmable Logic Device) อันได้แก่ พีแอลเอ (PLA: Programmable Logic Array), เอฟพีจีเอ (FPGA: Filed Programmable Gate Array), ซีพีแอลดี (CPLD: Cell Programmable Logic Device) เป็นต้น นอกนั้นยังมี เซมิคัสตัมไอซี (Semi-Custom IC) ได้แก่ เกตอะเรย์ (Gate array), เซลล์มาตรฐาน (Standard Cell) ขั้นตอนของการออกแบบจากบนลงล่างมีรายละเอียดดังนี้

1. ขั้นตอนการสร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา
2. ขั้นตอนการเขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษาวีเอชดีแอล หรือ ภาษาเอชดีแอลอื่นๆ สำหรับบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด
3. หลังจากที่ได้หลักการขั้นต้นพร้อมกับแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจร หรือสังเคราะห์ ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น ให้อยู่ในรูปของวง

จรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกต และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของเน็ทลิสท์ (Netlist) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในรูประดับเกตหรือเน็ทลิสท์แล้ว ข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงาน ในเรื่องของความถูกต้องของฟังก์ชันแล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่า อุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมี เวลาหน่วงของการแพร่กระจาย (Propagation delay time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ นาโนวินาที (10^{-9} นาที) แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชันต่างๆ จำนวน 10,000 เกต ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้
5. ขั้นตอนของการผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่ในรูปของวงจรรวม (ASIC)
6. หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่ออกแบบ จะประกอบด้วยอินพุตและเอาต์พุตแพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก
7. หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้ง เป็นการควบคุมคุณภาพของผลิตภัณฑ์

บทที่ 4
อุปกรณ์ซีพีแอลดี

ความก้าวหน้าของอุตสาหกรรมอิเล็คทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ มากมาย โดยมีแนวโน้มที่จะมีความซับซ้อน และ ขนาดที่ใหญ่ขึ้น เพื่อเพิ่มประสิทธิภาพ และ ระดับความเชื่อถือได้ของวงจรรวมให้สูงขึ้น แต่ค่าใช้จ่ายในการพัฒนาที่ลดลง

เมื่อสังเกตจากเทคโนโลยีไมโครโพรเซสเซอร์และหน่วยความจำปัจจุบันจะพบว่า ทุกๆ ครั้งที่มีการพัฒนาขึ้นทำให้เกิดช่องว่าง ระหว่างวงจรรวมและไอซีมาตรฐานมากขึ้น นอกจากนี้ การออกแบบวงจรดิจิตอลในปัจจุบันโดยใช้ภาษาบรรยายวงจรนั้น ทำให้การพัฒนาเป็นไปอย่างรวดเร็ว ดังนั้น อุปกรณ์ที่ใช้สำหรับการออกแบบวงจรดิจิตอลที่สามารถโปรแกรมได้ จึงเป็นสิ่งจำเป็น ซีพีแอลดีก็เป็นหนึ่งในนั้น

ซีพีแอลดี (CPLD) เป็นอุปกรณ์ที่สามารถโปรแกรมได้ ซึ่งมีโครงสร้างภายในคล้ายกับอุปกรณ์ประเภท พีเอแอล (PAL) แต่จะมีความสลับซับซ้อนมากกว่า มีเกทที่สามารถใช้ได้มากกว่า 800-12,800 ขึ้นอยู่กับเบอร์ และ ยังสนับสนุนการออกแบบด้วยภาษาบรรยายการทำงานของวงจร (Hardware Description Language , HDL) ได้หลายภาษา อันได้แก่ วีเอชดีแอล (VHDL), เอเบล (ABEL) และ เวอร์ริลอก (Verilog) ซึ่งอำนวยความสะดวกในการออกแบบให้แก่ผู้ใช้ที่มีความถนัดในแต่ละภาษาได้เป็นอย่างดี

4.1 ซีพีแอลดีตระกูล XC9500 ของบริษัทไซลิงก์

ซีพีแอลดีตระกูล XC9500 เป็นซีพีแอลดี รุ่นที่ 3 จากบริษัทไซลิงก์ (Xilinx Inc.) โดยมีจุดมุ่งหมายสำหรับผู้ผลิตที่ต้องการให้ระบบมีความสามารถในการทดสอบ และ โปรแกรมภายในระบบได้อย่างสมบูรณ์ เพื่อสนับสนุนทุกช่วงในวงจรชีวิตผลิตภัณฑ์ ตั้งแต่ การสร้างต้นแบบ , การรวมระบบ , การสร้าง และ การอัพเกรด

ซีพีแอลดีตระกูล XC9500 สามารถสนับสนุนการทำงานภายในระบบได้ทั้งหมด โดยใช้มาตรฐาน IEEE 1149.1 boundary scan (JTAG) และ เพิ่มความสามารถในการล็อกขา (Pin-locking) เพื่อป้องกันการความผิดพลาดในระหว่างการโปรแกรม สิ่งเหล่านี้ทำให้ซีพีแอลดีสามารถโปรแกรมในระบบได้ นอกจากนี้ ภายในยังใช้เทคโนโลยีแบบ FastFLASH ซึ่งเป็นหน่วยความจำแบบแฟลชที่มีความสามารถสูง สามารถรองรับการโปรแกรมใหม่ได้ 10,000 ครั้ง จากคุณสมบัติทั้งสองนี้ทำให้ซีพีแอลดีเหมาะสำหรับการใช้งานที่ต้องการการอัพเกรดอยู่บ่อยครั้ง

นอกจากนี้ซีพีแอลดีตระกูล XC9500 ยังมีค่าความหน่วงแพร่กระจาย (Propagation delay) ต่ำเพียง 5-15 ns เท่านั้น และ ยังมีขนาดให้เลือกถึง 9 เบอร์ ซึ่งมีจำนวนเกทตั้งแต่ 800-12,800 เกท ทำให้ซีพีแอลดีตระกูล XC9500 มีความสามารถ และ ความจุที่สูง

4.2 ความสามารถในการล็อกขา

ความสามารถในการล็อกขาในสถาปัตยกรรมของซีพีแอลดีตระกูลXC9500 เกิดจากองค์ประกอบ 3 อย่าง ดังนี้

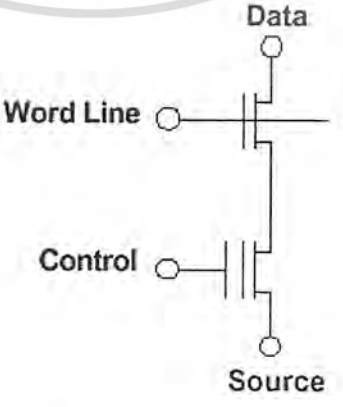
1. การใช้เมตริกซ์สวิตช์ FastCONNECT ซึ่งทำให้ทุกขา I/O และ ฟังก์ชั่นบล็อกสามารถเชื่อมต่อกันได้ทั้งหมด
2. จำนวนอินพุท (Fan-In) ของ ฟังก์ชั่นบล็อก มีจำนวนสูงสุด 36 อินพุท ทำให้จำนวนที่สามารถจ่ายให้กับฟังก์ชั่นบล็อกมีจำนวนมากขึ้น
3. มีตัวจัดตำแหน่งโปรดักเทอม ทำให้สามารถต่อเชื่อมสัญญาณลอจิกระหว่าง ฟังก์ชั่นบล็อกได้อย่างมีประสิทธิภาพ

4.3 เทคโนโลยี FastFLASH

เทคโนโลยี FastFLASH เป็นเทคโนโลยีเกี่ยวกับหน่วยความจำแบบแฟลชตัวแรกสำหรับ ซีพีแอลดี ที่ทำงานด้วยแรงดัน 5 โวลต์ สามารถโปรแกรมใหม่ได้ถึง 10,000 ครั้ง ซึ่งมากกว่า EEPROM ที่ใช้กับซีพีแอลดี แบบเดิมถึง 100 เท่า ความสามารถที่เพิ่มขึ้นนี้เนื่องมาจาก

1. ออกไซด์ที่ใช้ในการผลิตมีความหนาถึง 100 อังสตรอม ซึ่งมากกกว่า EEPROM ที่มีเพียง 80 อังสตรอม ความหนาของออกไซด์ที่เคลือบ ทำให้จำนวนครั้งที่สามารถโปรแกรมใหม่เพิ่มมากขึ้น
2. ใน FastFLASH ใช้แรงดันเพียง 10 mV/cm ในการโปรแกรมใหม่เท่านั้น ซึ่งต่างจาก EEPROM ที่ใช้ประมาณ 15-20 mV/cm เนื่องจากใช้แรงดันในการโปรแกรมน้อยกว่าทำให้ FastFLASH สามารถป้องกันการสึกหรอของออกไซด์ได้ดี
3. FastFLASH ใช้เวลาในการโปรแกรมประมาณ 100 us ซึ่งน้อยกว่า EEPROM ที่ใช้ 5-10 ms

นอกจากนั้น ขนาดเซลล์ของ FastFLASH ยังมีขนาดเล็กเมื่อเทียบกับ EEPROM ทำให้ขนาดของดายเล็กลง และ เมตริกซ์สวิตช์ FastCONNECT สามารถหาเส้นทางในการเชื่อมต่อได้ง่ายขึ้น รูปเซลล์ของ FastFLASH แสดงในรูปที่ 4-1



รูปที่ 4-1 เซลล์ของ FastFLASH

4.4 สมาชิกในตระกูล XC9500

ซีพีแอลดีตระกูล XC9500 มีอยู่ด้วยกัน 9 ตัว ซึ่งขาอุปกรณ์แต่ละตัวสามารถเข้ากันได้ ทำให้ผู้ใช้เกิดความคล่องตัวในการเลือกอุปกรณ์ที่เหมาะสมที่สุด สมาชิกของตระกูล XC9500 มีรายละเอียดดังตารางข้างล่าง

อุปกรณ์	มาโครเซลล์	จำนวนเกท	จำนวนรีจิสเตอร์	Propagation Delay
XC9536	36	800	36	5 ns
XC9572	72	1,600	72	7.5 ns
XC95108	108	2,400	108	7.5 ns
XC95144	144	3,200	144	7.5 ns
XC95180	180	4,000	180	10 ns
XC95216	216	4,800	216	10 ns
XC95288	288	6,400	288	10 ns
XC95432	432	9,600	432	12 ns
XC95576	576	12,800	576	15 ns

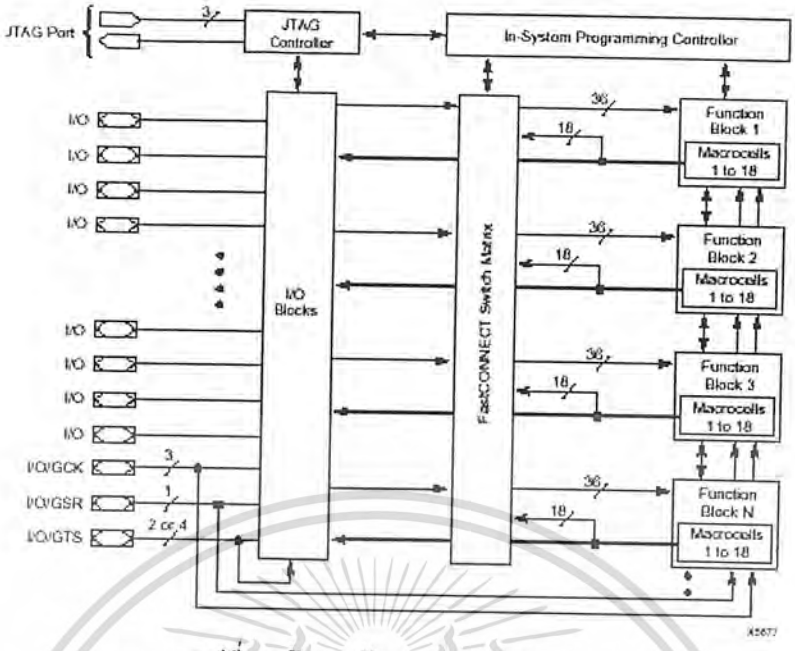
4.5 โครงสร้างของซีพีแอลดี

ภายในซีพีแอลดีประกอบไปด้วย ฟังก์ชั่นบล็อก จำนวนหลายตัวเชื่อมต่อกันอยู่ภายใน ผ่านเมตริกซ์สวิตช์ FastCONNECT โดยที่แต่ละ ฟังก์ชั่นบล็อก จะมีจำนวนมาโครเซลล์อยู่ภายในจำนวน 18 มาโครเซลล์ และ มีความเร็วสูงสุดจาก ขาถึงขา (Pin-to-Pin) อยู่ที่ 5 ns นอกจากนี้สัญญาณ I/O ยังสามารถเชื่อมต่อกับสัญญาณได้ 2 ระดับ คือ 5 โวลต์ และ 3.3 โวลต์

จากรูปที่ 4-2 จะเห็นว่าฟังก์ชั่นบล็อกที่มีความเร็วสูง ถูกเชื่อมต่อกับเมตริกซ์สวิตช์ FastCONNECT ที่อยู่ตรงกลาง โดยที่ขา I/O จะเชื่อมต่ออยู่รอบๆ FastCONNECT สัญญาณที่เข้าออกจากขาสัญญาณมาได้จากทั้งฟังก์ชั่นบล็อก และ FastCONNECT

4.6 เมตริกซ์สวิตช์ FastCONNECT

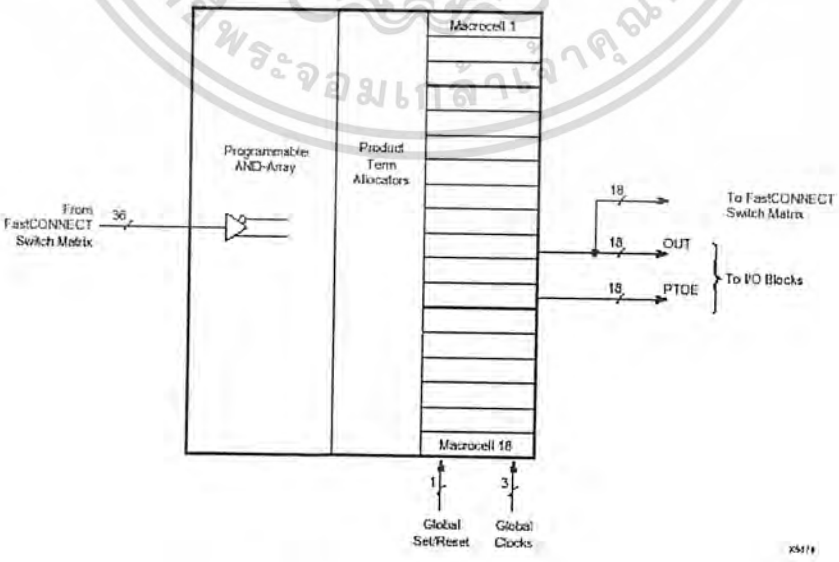
เมตริกซ์สวิตช์ FastCONNECT เชื่อมต่อสัญญาณความเร็วสูงเข้ากับฟังก์ชั่นบล็อก และ เชื่อมเอาต์พุทของทุกมาโครเซลล์ เข้ากับฟังก์ชั่นบล็อก ผ่านสวิตช์ภายใน ความสามารถในการเชื่อมต่อสูง เป็นกลไกสำคัญที่ทำให้การออกแบบสามารถเปลี่ยนแปลงได้ แม้แต่หลังจากที่อุปกรณ์ถูกติดตั้งลงบนบอร์ดแล้ว



รูปที่ 4-2 โครงสร้างภายในของ CPLD

4.7 ฟังก์ชั่นบล็อก

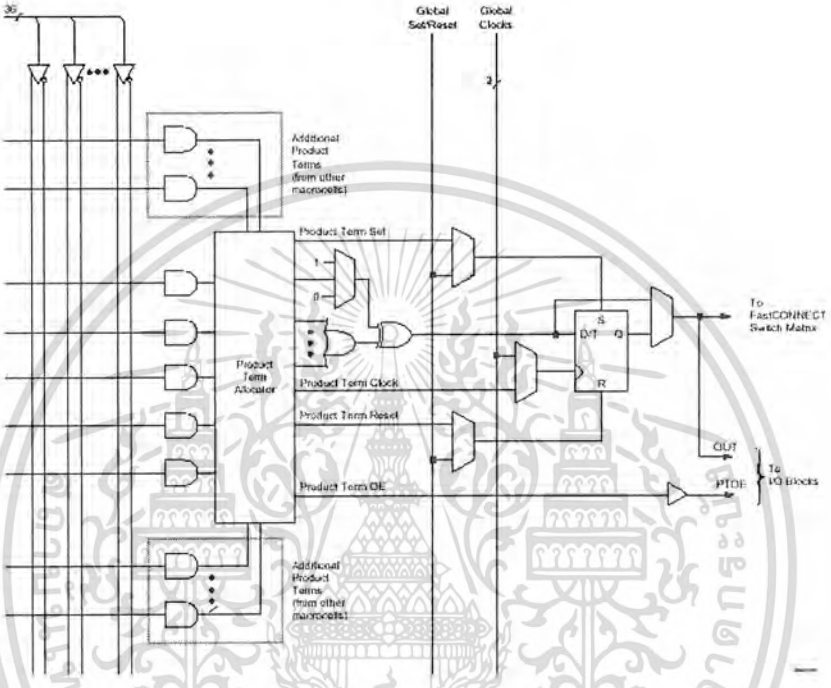
ฟังก์ชั่นบล็อกเป็นกลุ่มของมาโครเซลล์จำนวน 18 ตัว ฟังก์ชั่นบล็อก แต่ละตัวมีโปรดักเทอม จำนวน 90 ตัวที่สามารถเชื่อมต่อเข้ากับมาโครเซลล์ ทั้ง 18 ตัวนั้น ซึ่งสนับสนุนคุณสมบัติการล็อกขา และทำให้การออกแบบคล่องตัวมากขึ้น ประสิทธิภาพขึ้นอยู่กับซอฟต์แวร์ที่ช่วยจัดรูปแบบของโปรดักเทอม จำนวน 5 ตัวที่เข้ามา เอาต์พุทของมาโครเซลล์สามารถขับสัญญาณให้กับขาเอาต์พุท และ เป็นอินพุทของ FastCONNECT (ดูรูปที่ 4-3)



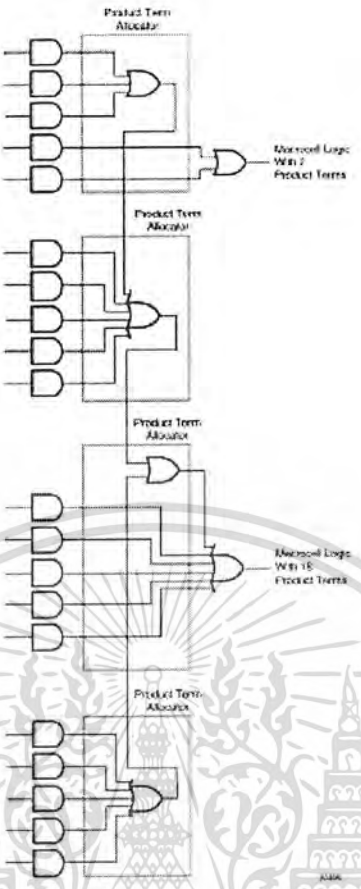
รูปที่ 4-3 โครงสร้างภายใน ฟังก์ชั่นบล็อก

4.8 มาโครเซลล์

ในโหมดปกติ จะมี 5 โปรดักเทอมที่ OR เข้าด้วยกันเพื่อขับสัญญาณให้กับดีฟลิปฟลอปภายในมาโครเซลล์ ดังที่แสดงในรูปที่ 4-4 รูปแบบภายในโดยมากจะเป็นการจำลองตัวเองเป็น XOR , แอดเดอร์ หรือ ลอจิกอินเวอร์ชั่น การจัดวางอีกลักษณะหนึ่งคือการส่งโปรดักเทอมไปยังมาโครเซลล์ที่อยู่รอบๆ เพื่อเพิ่มความสามารถในการรองรับลอจิกดังรูปที่ 4-5



รูปที่ 4-4 โครงสร้างภายในมาโครเซลล์



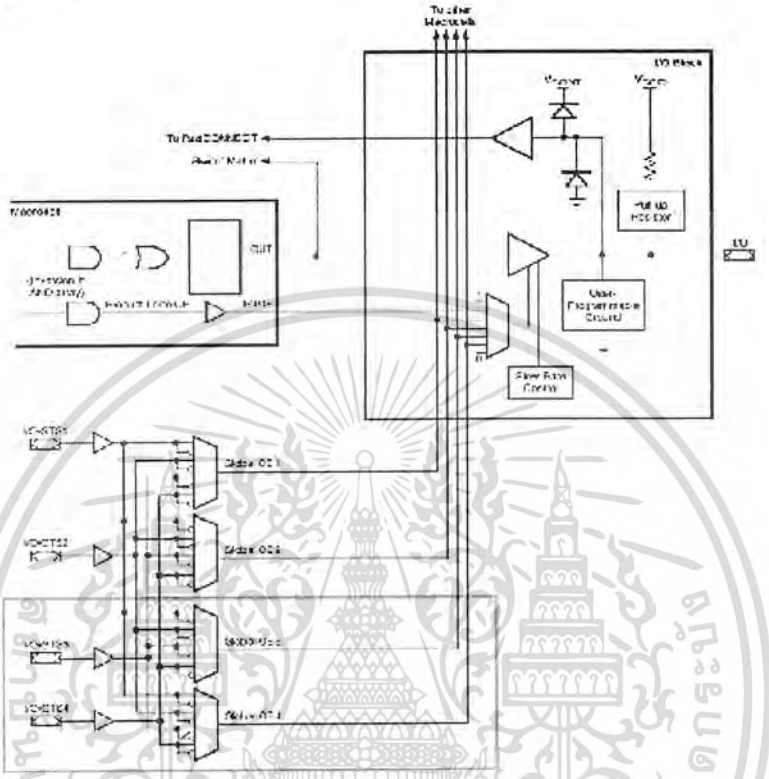
รูปที่ 4-5 การส่ง Product term ไปยังมาโครเซลล์ รอบๆ

ฟลิปฟลอปของตระกูล XC9500 สามารถเป็นได้ทั้ง ดีฟลิปฟลอป และ ทีฟลิปฟลอป ทำให้สามารถสร้างตัวนับโดยใช้เกทเพียงจำนวนไม่กี่เกท ตารางข้างล่าง แสดงจำนวน Product term ที่ใช้ในการสร้างฟังก์ชั่นพื้นฐาน

ฟังก์ชั่น	จำนวน Product term
Shift Register	2
Counters	2-4
n:1 MUX	n
Adder	6
Exclusive-OR	2
Storage Register	1

4.9 I/O Block

I/O Block (IOB) ทำหน้าที่เชื่อมต่อวงจรภายใน เข้ากับขาของซีพีแอลดี ในแต่ละ IOB จะประกอบด้วย อินพุทบัฟเฟอร์ , เอาต์พุทไดรเวอร์ , เอาต์พุทเอนาเบิล และ กราวน์ของเอาต์พุทที่สามารถโปรแกรมได้ ดังรูปที่ 4-6



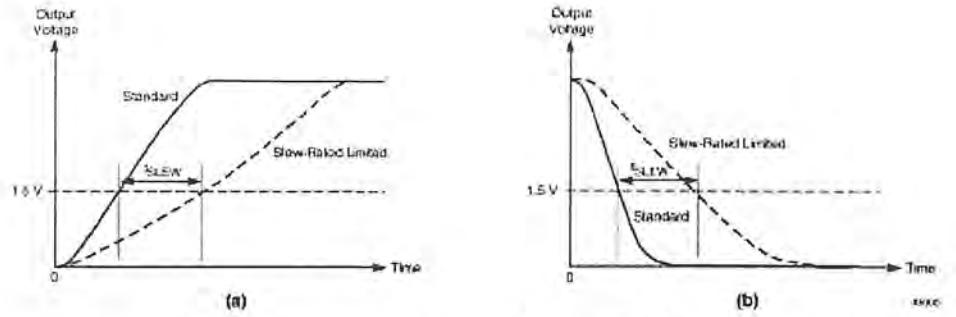
รูปที่ 4-6 I/O Block

อินพุทบัฟเฟอร์ สามารถทำงานได้กับวงจรประเภท CMOS , TTL ทั้งแบบ 5V และ 3.3V โดยที่อินพุทบัฟเฟอร์จะใช้แรงดันขนาด 5 V (VCCINT) เพื่อให้แน่ใจว่าแบ่งแยก (Threshold) ของอินพุทคงที่ และ ไม่เลื่อนไปมาเนื่องจากแรงดัน VCCIO

เอาต์พุทเอนาเบิล สามารถเลือกสร้างได้ 4 แบบ ได้แก่

1. สร้างจาก Product term ที่ได้จากมาโครเซลล์
2. สร้างจากสัญญาณเอาต์พุทเอนาเบิลร่วม
3. ต่อกับค่า '1' โดยตรง
4. ต่อกับค่า '0' โดยตรง

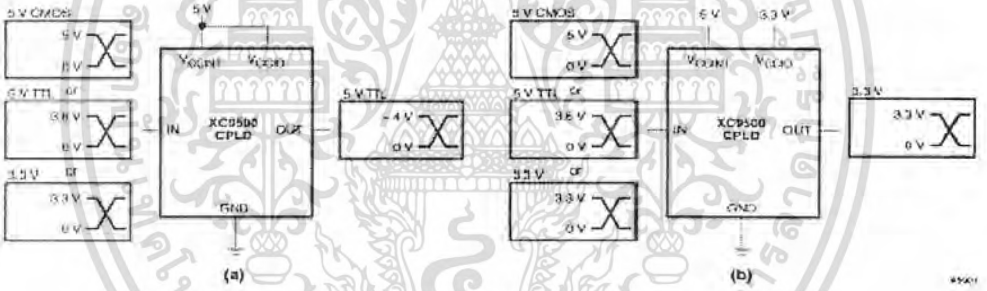
ภายในซีพีแอลดีมีสัญญาณเอาต์พุทเอนาเบิลร่วมอยู่ 2 เส้น มีชื่อว่า GTS แต่ละเอาต์พุทสามารถควบคุมค่า สลูเรท (Slew rate) แยกกันได้อย่างอิสระ ซึ่งสามารถโปรแกรมให้ ช่วงขอบของเอาต์พุทมีค่าช้าลง ทำให้สัญญาณรบกวนน้อยลงด้วย นอกจากนี้ยังสามารถโปรแกรมขาที่ต่อกับ IOB ให้เป็น กราวด์ เพื่อลดสัญญาณรบกวนจากอุปกรณ์ตัวอื่นในระบบได้ (ดูรูปที่ 4-7)



รูปที่ 4-7 การควบคุม Slew rate สำหรับ (a) ขาขึ้น (b) ขาลง

ที่ขา I/O จะมีพูลอัพรีจิสเตอร์ ต่อไว้เพื่อป้องกันไม่ให้เกิดสภาวะขาลอยในขณะที่อุปกรณ์ยังไม่ทำงานตามปกติ ตัวต้านทานดังกล่าวจะทำงานในระหว่างที่อุปกรณ์อยู่ในโหมดการโปรแกรม,ระหว่างที่อุปกรณ์เริ่มได้รับไฟเลี้ยง และ เมื่ออุปกรณ์อยู่ในสภาวะถูกลบหมดแล้ว นอกจากนั้น ตัวต้านทานจะอยู่ในสภาวะไม่ทำงาน

เอาต์พุทไดรเวอร์ของซีพีแอลดี สามารถทำงานได้กับแรงดันขนาด 5V TTL หรือ 3.3V TTL ได้ โดยการต่อขา VCCIO เข้ากับระดับแรงดันที่ต้องการ (ดูรูปที่ 4-8)

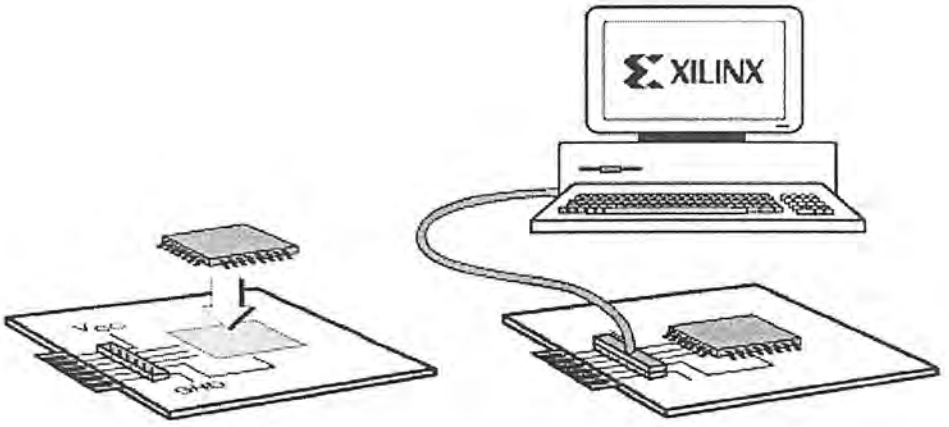


รูปที่ 4-8 การทำงานด้วยระดับแรงดัน (a) 5V (b) 3.3/5V

4.10 การโปรแกรมภายในระบบ

ซีพีแอลดีในตระกูล XC9500 สามารถถูกโปรแกรมได้ในขณะที่ติดตั้งอยู่บนแผงวงจรพิมพ์ โดยการใช้การต่อสัญญาณตามมาตรฐาน JTAG ขนาด 4 ขา ทำให้สามารถลดความยุ่งยากในขั้นตอนการออกแบบ และ อัพเกรด การโปรแกรมจะใช้สายดาวน์โหลด หรือ จะใช้คอมพิวเตอร์จำลองการทำงานของชุดคำสั่ง JTAG ก็ได้ (ดูรูปที่ 4-9)

ขา I/O ทุกขาจะเป็นแบบ ไตรเสตท (Tri-state) และ ถูกพูลให้เป็นลอจิก 'HIGH' ด้วยตัวต้านทานภายใน IOB ในระหว่าง โปรแกรมหากต้องการให้สัญญาณบางเส้นมีค่าเป็น 'LOW' จะต้องต่อความต้านทานพูลดาวน์ที่ขานั้น



รูปที่ 4-9 การโปรแกรมภายในระบบ (In-System Programming)

4.11 ความปลอดภัยในการออกแบบ

ซีพีแอลดีตระกูล XC9500 มีคุณสมบัติในการรักษาความปลอดภัยของการออกแบบ ซึ่งป้องกันไม่ให้ผู้อื่นสามารถอ่านข้อมูลภายในได้ ทำให้ผู้อื่นไม่สามารถก๊อปปี้ชิ้นงานไปใช้โดยไม่ได้รับอนุญาตได้ โดยการตั้งค่าที่ บิตป้องกันการอ่าน (Read security bit) นอกจากนี้ยังสามารถป้องกันการถูกลบเนื่องมาจากสัญญาณรบกวนได้

บิตป้องกันการเขียน (Write security bit) เพิ่มความสามารถในการป้องกันอุปกรณ์ถูกลบโดยไม่ได้ตั้งใจ ซึ่งเกิดจากขา JTAG ได้รับสัญญาณรบกวน เช่น ในระหว่างที่เปิดเครื่อง เป็นต้น ถ้าบิตนี้ถูกเซ็ตแล้ว วงจรป้องกันการเขียนจะทำงาน และ จะต้องป้อนแพทเทิร์นของข้อมูลที่เหมาะสมเข้าไปก่อนการเขียนจึงจะสามารถเขียนได้

4.12 โหมดประหยัดพลังงาน

มาโครเซลล์แต่ละตัวสามารถถูกเซ็ตโดยผู้ใช้ให้สามารถทำงานในโหมดประหยัดพลังงานได้ ซึ่งในการใช้งานทั่วไปนั้น ส่วนของวงจรที่ต้องการประสิทธิภาพในการทำงานสูง จะถูกเซ็ตให้ทำงานในโหมดปกติ ในขณะที่บางส่วนที่ไม่ได้ใช้งาน หรือ ไม่ต้องการประสิทธิภาพในการทำงานที่สูงมากนัก ก็อาจถูกเซ็ตให้อยู่ในโหมดประหยัดพลังงานเพื่อลดการใช้พลังงานในระบบลง มาโครเซลตัวที่อยู่ในโหมดประหยัดพลังงานจะมีค่าหน่วงเวลาที่มากขึ้น

โหมดประหยัดพลังงาน ไม่มีผลกับโปรดักเทอมตัวที่ใช้เป็นสัญญาณนาฬิกาให้กับเอาต์พุท และโปรดักเทอมที่ใช้เป็นสัญญาณเอาต์พุทเอนาเบิล

บทที่ 5
ระบบบัส I2C

สำหรับระบบควบคุมแบบดิจิตอล ที่ควบคุมด้วยไมโครคอนโทรลเลอร์ ขนาด 8 บิต ที่มีเกณฑ์ในการออกแบบดังต่อไปนี้

- ระบบประกอบด้วยไมโครคอนโทรลเลอร์ อย่างน้อย 1 ตัว กับอุปกรณ์ภายนอก เช่น หน่วยความจำ
- ต้องการให้การเชื่อมต่อระหว่างอุปกรณ์แต่ละตัวภายในประหยัดที่สุด
- ระบบไม่ต้องการอัตราการรับส่งข้อมูลที่สูงมาก
- ประสิทธิภาพโดยรวมขึ้นอยู่กับ การเลือกใช้อุปกรณ์ และ โครงสร้างของระบบบัส

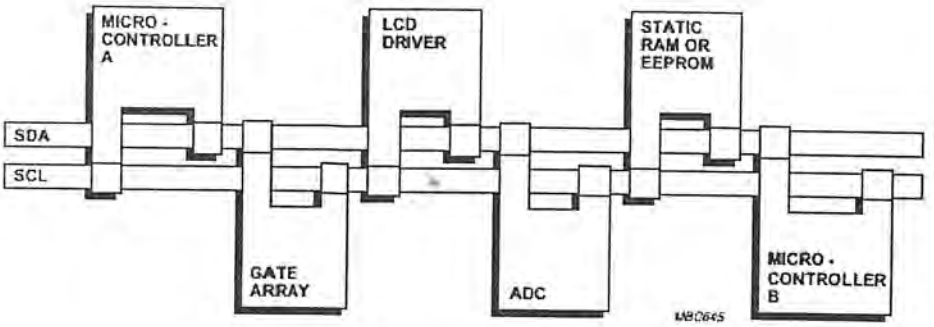
สำหรับระบบที่ได้กล่าวมา จำเป็นอย่างยิ่งที่ต้องใช้ระบบบัสที่มีการรับส่งข้อมูลแบบอนุกรมนั้น ถึงแม้ว่าอัตราการรับส่งข้อมูลจะน้อยกว่าระบบบัสแบบขนานก็ตาม แต่ระบบบัสแบบอนุกรมจะใช้สายสัญญาณน้อยกว่า ทำให้การเชื่อมต่อระหว่างอุปกรณ์ทำได้ง่าย และ ประหยัดกว่าการใช้บัสแบบขนาน

สำหรับอุปกรณ์ที่ต้องการเชื่อมต่อกับบัสแบบอนุกรม จำเป็นต้องมีโปรโตคอลสำหรับการติดต่อเพื่อป้องกันความสับสน หรือ การสูญหายของข้อมูล อุปกรณ์ที่มีความเร็วสูง ต้องสามารถติดต่อกับอุปกรณ์ที่มีความเร็วต่ำกว่าได้อย่างถูกต้อง ในกรณีที่มีอุปกรณ์หลายตัวต้องการใช้บัสพร้อมกัน จะต้องมีกลไกที่ตัดสินได้ว่าจะให้อุปกรณ์ตัวใดมีสิทธิใช้บัส สิ่งเหล่านี้ ทำให้เกิดระบบบัสแบบ I2C ขึ้นมา

5.1 แนวคิดของระบบบัส I2C

ระบบบัส I2C สามารถทำงานได้กับเทคโนโลยีการผลิต IC ทุกประเภท ไม่ว่าจะเป็น NMOS , CMOS หรือ Bipolar โดยการใช้สายสัญญาณเพียงสองเส้นคือ สายข้อมูลอนุกรม (Serial Data,SDA) และสายสัญญาณนาฬิกาอนุกรม (Serial Clock , SCL) ในการรับส่งข้อมูล อุปกรณ์แต่ละตัวจะมีหมายเลขแอดเดรสโดยเฉพาะ ไม่ว่าจะเป็น ไมโครคอนโทรลเลอร์ หรือ จอ LCD อุปกรณ์แต่ละตัวสามารถเป็นได้ทั้งตัวส่ง (Transmitter) หรือ ตัวรับ (Receiver) ขึ้นอยู่กับหน้าที่ ตัวอย่างเช่น จอ LCD สามารถเป็นเพียงผู้รับได้อย่างเดียว แต่หน่วยความจำสามารถเป็นได้ทั้งตัวส่ง และ ตัวรับ นอกจากนี้อุปกรณ์ยังสามารถเป็นได้ทั้งมาสเตอร์ หรือ สเลฟ โดยมาสเตอร์หมายถึงอุปกรณ์ที่เป็นตัวเริ่มต้นการรับส่งข้อมูล และ สร้างสัญญาณนาฬิกาสำหรับการสื่อสาร อุปกรณ์ตัวอื่นในขณะนั้นถือเป็นสเลฟ

ระบบบัส I2C เป็นบัสแบบมัลติมาสเตอร์ (Multi-master bus) กล่าวคือ ภายในระบบบัสนั้นสามารถมีอุปกรณ์ที่สามารถควบคุมบัสมากกว่าหนึ่งตัว ซึ่งโดยมากมักหมายถึงไมโครคอนโทรลเลอร์นั่นเอง ลองพิจารณารูปแบบการเชื่อมต่อในรูปที่ 5-1 จุดที่น่าสนใจอยู่ที่ ความสัมพันธ์ระหว่างมาสเตอร์ และสเลฟ และ อุปกรณ์ตัวส่ง และ ตัวรับ จะสังเกตุได้ว่า ความสัมพันธ์ที่ได้กล่าวมา ไม่จำเป็นจะต้องเป็นความสัมพันธ์แบบถาวร แต่ขึ้นอยู่กับทิศทางในการรับส่งข้อมูลในช่วงเวลานั้น ตัวอย่างเช่น



รูปที่ 5-1 ตัวอย่างระบบบัส I2C ที่มีไมโครคอนโทรลเลอร์สองตัว

สมมติว่า ไมโครคอนโทรลเลอร์ A ต้องการส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ B จะต้องทำตามขั้นตอนดังนี้

- ไมโครคอนโทรลเลอร์ A ซึ่งเป็นมาสเตอร์ จะชี้ไปยังไมโครคอนโทรลเลอร์ B ซึ่งเป็นสเลฟ
- ไมโครคอนโทรลเลอร์ A ซึ่งเป็นมาสเตอร์ และ ตัวส่ง ส่งข้อมูลไปให้กับไมโครคอนโทรลเลอร์ B ซึ่งเป็นสเลฟ และ ตัวรับ
- ไมโครคอนโทรลเลอร์ A ส่งสัญญาณยุติการส่งข้อมูล

แต่ถ้าไมโครคอนโทรลเลอร์ A ต้องการอ่านข้อมูลจากไมโครคอนโทรลเลอร์ B

- ไมโครคอนโทรลเลอร์ A ซึ่งเป็นมาสเตอร์ จะชี้ไปยังไมโครคอนโทรลเลอร์ B ซึ่งเป็นสเลฟ
- ไมโครคอนโทรลเลอร์ A ซึ่งเป็นมาสเตอร์ และ ตัวรับ จะอ่านข้อมูลจากไมโครคอนโทรลเลอร์ B ซึ่งเป็นสเลฟ และ ตัวส่ง
- ไมโครคอนโทรลเลอร์ A ส่งสัญญาณยุติการส่งข้อมูล

ในกรณีที่มีอุปกรณ์มาสเตอร์หลายตัวบนระบบบัส เป็นไปได้ที่อุปกรณ์เหล่านั้น จะเริ่มต้นส่งข้อมูลพร้อมกันทำให้เกิดความผิดพลาดขึ้นได้ เพื่อป้องกันความผิดพลาดนี้ จึงต้องมีกระบวนการในการตัดสินว่าจะให้อุปกรณ์ตัวใดเป็นผู้ใช้บัส กระบวนการนี้เรียกว่าการชี้ขาดเพื่อเข้าใช้บัส(Arbitration) ซึ่งรายละเอียดจะกล่าวภายหลัง

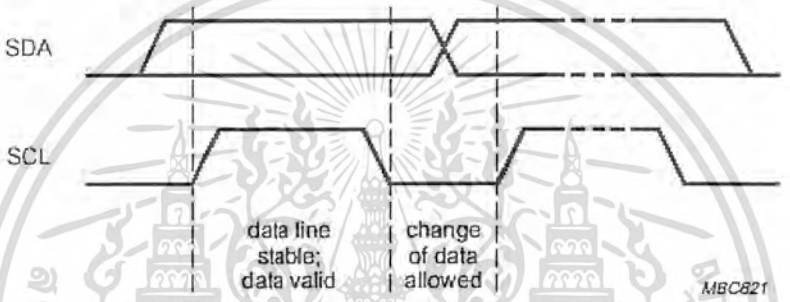
5.2 คุณลักษณะทั่วไปของระบบบัส I2C

ทั้ง SDA และ SCL เป็นสายสัญญาณแบบสองทิศทาง ซึ่งต่ออยู่กับแหล่งจ่ายไฟบวกดังรูปที่ 5-6 ในขณะที่บัสว่าง สายสัญญาณทั้งสองเส้นจะมีค่า HIGH ความเร็วในการรับส่งข้อมูลบนระบบบัส I2C เท่ากับ 100 kbit/s ในโหมดมาตรฐาน (Standard-mode) , 400 kbit/s ในโหมดความเร็ว (Fast-mode) และ 3.4 Mbit/s ในโหมดความเร็วสูง (High-speed mode) จำนวนอุปกรณ์ที่สามารถเชื่อมต่อกับระบบบัสขึ้นอยู่กับ

ค่าความจุไฟฟ้า (Capacitance) ซึ่งต้องมีค่าไม่เกิน 400 pF และ เพื่อให้สามารถทำงานกับเทคโนโลยีการผลิต IC ได้อย่างหลากหลาย ค่าของลอจิก '0' และ ลอจิก '1' จึงขึ้นอยู่กับค่าของ VDD ของวงจร

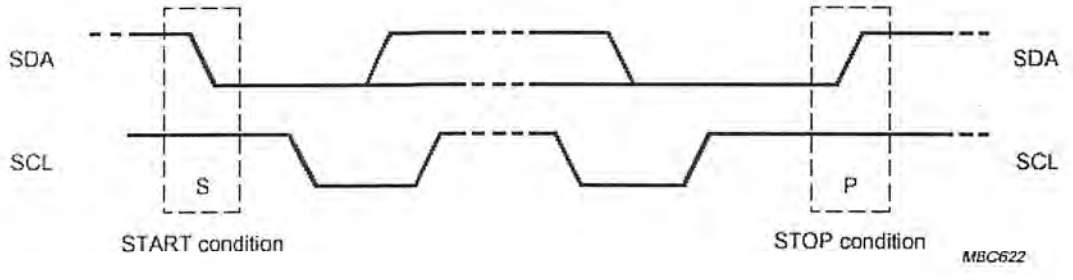
5.3 การรับส่งข้อมูลระดับบิต

เนื่องจากระบบบัส I2C สามารถเชื่อมต่ออุปกรณ์ที่ผลิตจากเทคโนโลยีต่างๆได้ เช่น CMOS , NMOS หรือ Bipolar จึงไม่มีการกำหนดค่าแรงดันของลอจิก '0' (LOW) และ '1' (HIGH) แต่จะขึ้นอยู่กับแรงดันของ VDD และ ใช้สัญญาณนาฬิกาสำหรับการแบ่งแยกข้อมูลแต่ละบิต สำหรับการรับส่งบิตที่เป็นข้อมูล ค่าของสายสัญญาณ SDA สามารถเปลี่ยนแปลงได้เมื่อค่าของ SCL มีค่าเป็น LOW เท่านั้น และ จะต้องคงที่เมื่อ SCL มีค่าเป็น HIGH (ดูรูปที่ 5-2)



รูปที่ 5-2 การรับส่งข้อมูลระดับบิต

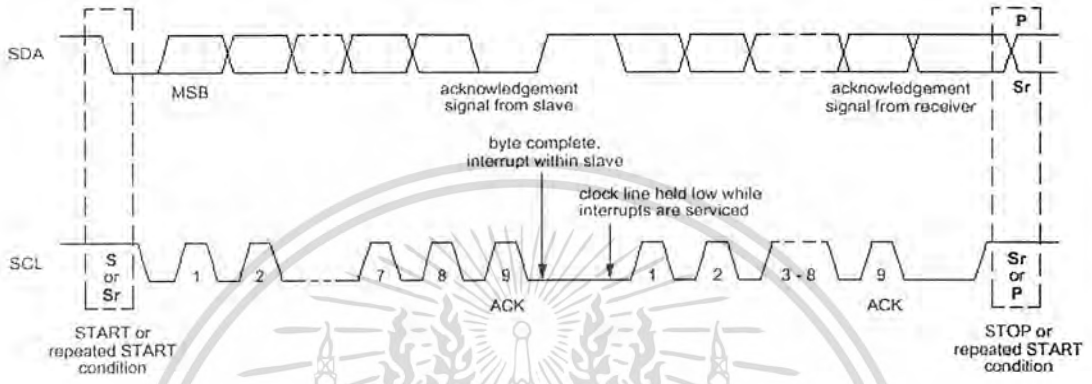
สำหรับการเริ่มต้นการรับส่งข้อมูล อุปกรณ์มาสเตอร์ต้องส่งเงื่อนไขการเริ่มต้น (Start condition,S) โดยการดึงค่าของ SDA จาก HIGH เป็น LOW ในขณะที่ SCL มีค่าเป็น HIGH หลังจากเงื่อนไขการเริ่มต้นแล้ว บัสจะไม่ว่าง (Busy) ถึงแม้ว่าจะเกิดเงื่อนไขการเริ่มต้นซ้ำอีกครั้งหนึ่งก็ตาม เมื่อต้องการยุติการส่งข้อมูล อุปกรณ์มาสเตอร์ตัวเดิม ก็ต้องส่งเงื่อนไขการยุติ (Stop condition,P) โดยการเปลี่ยนค่าของ SDA จาก HIGH เป็น LOW ในขณะที่ SCL มีค่าเป็น HIGH หลังจากเงื่อนไขการยุติ บัสก็จะว่างอีกครั้งหนึ่ง (ดูรูปที่ 5-3)



รูปที่ 5-3 เงื่อนไขการเริ่มต้น (START condition , S) และ เงื่อนไขการสิ้นสุด (STOP condition , P)

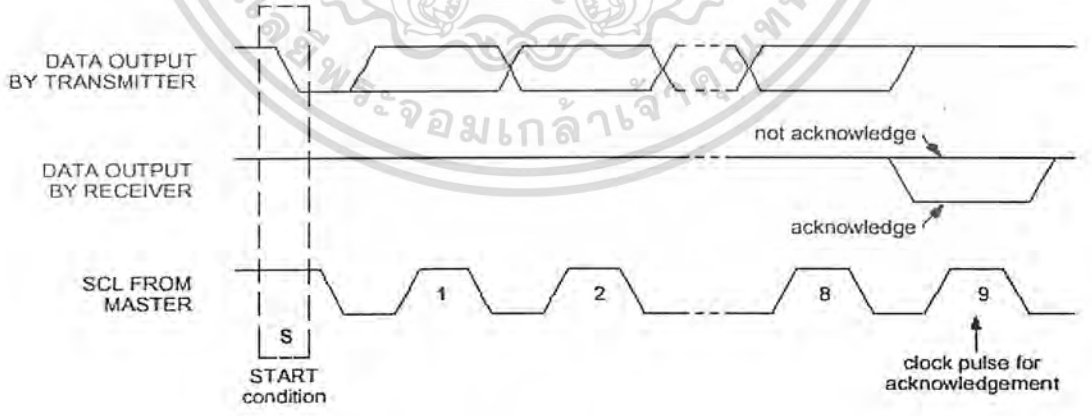
5.4 การรับส่งข้อมูลระดับไบต์

ขนาดของข้อมูลแต่ละไบต์บนระบบบัส I2C มีขนาด 8 บิต ซึ่งภายในการรับส่งข้อมูล 1 ครั้ง จะมีการรับส่งกี่ไบต์ก็ได้ การส่งแต่ละไบต์จะเริ่มจากการส่งบิตที่มีค่านัยสำคัญมากที่สุด (Most Significant bit,MSB) ก่อน (ดูรูปที่ 5-4) แต่ละไบต์จะถูกแยกออกจากกันโดยการส่งสัญญาณรับทราบ (Acknowledge,ACK) จาก อุปกรณ์ที่เป็นสเลฟกลับมา ในกรณีที่สเลฟไม่พร้อมที่จะส่งข้อมูล ก็สามารถคงค่าของ SCL เป็น LOW ไว้ก่อน เพื่อบอกให้อุปกรณ์ที่เป็นมาสเตอร์รอได้



รูปที่ 5-4 การรับส่งข้อมูลบนบัส I2C

เมื่อตัวส่งต้องการให้ตัวรับส่งสัญญาณตอบรับกลับมา ตัวส่งก็จะปล่อยสัญญาณ SDA ให้ขึ้นอยู่กับค่าของตัวรับ ตัวรับก็จะส่งสัญญาณตอบกลับมาโดยการดึงค่าของ SDA ลงและให้มีค่าเป็น LOW ในขณะที่ SCL มีค่าเป็น HIGH (ดูรูปที่ 5-5)



รูปที่ 5-5 สัญญาณรับทราบบน I2C

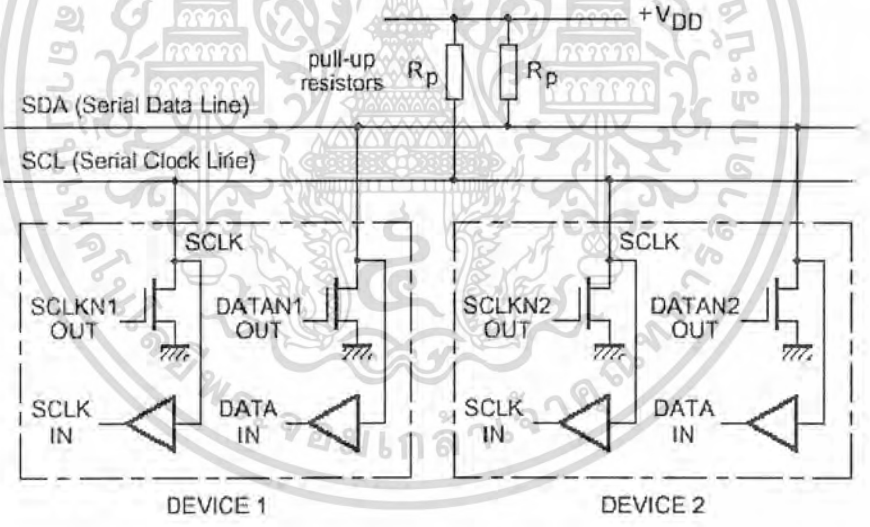
ในกรณีที่อุปกรณ์ตัวรับเป็นสเลฟ และ ไม่สามารถส่งสัญญาณตอบรับกลับมาได้ มาสเตอร์ก็สามารถส่งเงื่อนไขการเริ่มต้นซ้ำอีกครั้งหนึ่ง หรือ จะยุติการส่งข้อมูลโดยการส่งเงื่อนไขการยุติการส่งข้อ

มูลก็ได้ แต่ถ้าอุปกรณ์ตัวรับทำหน้าที่เป็นมาสเตอร์ และไม่สามารถส่งสัญญาณตอบรับกลับมาได้ ก็ไม่ต้องส่งสัญญาณตอบรับกลับไป อุปกรณ์ตัวรับที่เป็นสเลฟ ก็จะปล่อยสายสัญญาณ SDA ให้กับมาสเตอร์ เพื่อให้มาสเตอร์สามารถส่งเงื่อนไขการเริ่มต้นซ้ำ หรือ ส่งเงื่อนไขการยุติได้อีกครั้งหนึ่ง

5.5 กระบวนการชี้ขาดเพื่อเข้าใช้บัส

กระบวนการชี้ขาดเพื่อเข้าใช้บัส (Bus Arbitration) เป็นกระบวนการที่ใช้สำหรับหาอุปกรณ์ที่มีสิทธิเข้าใช้บัสในกรณีที่มีอุปกรณ์มาสเตอร์ สองตัวขึ้นไปต้องการเริ่มต้นการรับส่งข้อมูลพร้อมกัน สิ่งแรกที่ต้องพิจารณาในกระบวนการชี้ขาดเพื่อเข้าใช้บัส คือ การเข้าจังหวะกันของสัญญาณนาฬิกา (Clock synchronization)

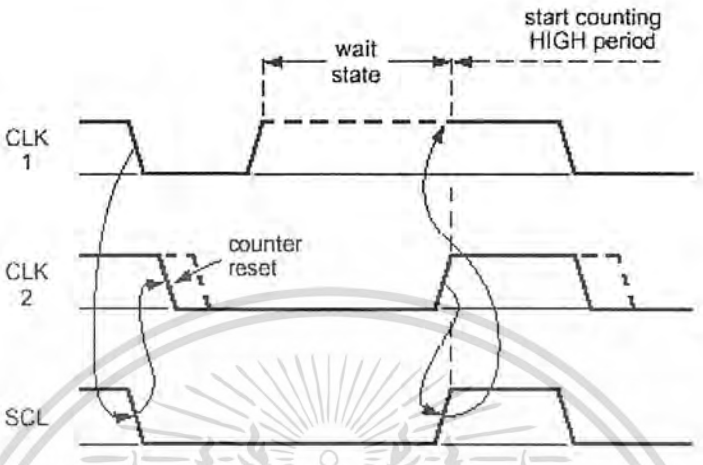
อุปกรณ์มาสเตอร์แต่ละตัวจะให้กำเนิดสัญญาณนาฬิกาสำหรับการสื่อสารเอง ซึ่งข้อมูลที่รับส่งคือเป็นค่าของ SDA ในขณะที่ SCL มีค่าเป็น HIGH ดังนั้นในกรณีที่มีอุปกรณ์มาสเตอร์หลายตัวเริ่มต้นการรับส่งข้อมูลพร้อมกัน จึงต้องมีการทำให้สัญญาณนาฬิกาที่ให้กำเนิดจากมาสเตอร์แต่ละตัว เข้าจังหวะกันโดยการเชื่อมต่อสัญญาณนาฬิกา (ในที่นี้หมายถึงสัญญาณนาฬิกาสำหรับการรับส่งข้อมูล) ของอุปกรณ์แต่ละตัวเข้ากับสายสัญญาณ SCL แบบ Wired-AND (ดูรูปที่ 5-6)



รูปที่ 5-6 การเชื่อมต่อแบบ Wired-AND

จากการเชื่อมสัญญาณนาฬิกาของอุปกรณ์ทุกตัวเข้ากับบัสแบบโดยการต่อแบบ Wired-AND ทำให้เมื่อเกิดการชนกันของข้อมูลระหว่างค่า HIGH กับ LOW แล้ว ค่าของสายสัญญาณจะมีค่าเป็น LOW หมายความว่า เมื่อใดก็ตามที่อุปกรณ์ตัวใดมีการเปลี่ยนแปลงสัญญาณนาฬิกาจาก HIGH เป็น LOW แล้วจะทำให้ค่าของ SCL มีค่าเป็น LOW ด้วย และ เมื่ออุปกรณ์ตัวนั้นเปลี่ยนค่าสัญญาณนาฬิกาของตัวเองจาก LOW เป็น HIGH แต่ถ้ายังมีอุปกรณ์บางตัวที่สัญญาณนาฬิกาของมันยังมีค่า LOW อยู่ ค่าของ SCL จะยังคงเป็น LOW อยู่ อุปกรณ์ตัวที่สัญญาณนาฬิกาของมันอยู่ในช่วง HIGH แล้ว จะต้องรอจนกว่าค่าของ SCL

จะเป็น HIGH จึงจะสามารถทำงานต่อไปได้ (ดูรูปที่ 5-7) จากที่ได้กล่าวมานั้น ทำให้สรุปได้ว่า สัญญาณ SCL ที่เข้าจังหวะกันแล้ว จะมีค่า LOW เท่ากับช่วงเวลาของสัญญาณนาฬิกาที่มีค่า LOW ที่ยาวที่สุด และจะมีค่า HIGH เท่ากับช่วงเวลาที่ของสัญญาณนาฬิกาที่สั้นที่สุด



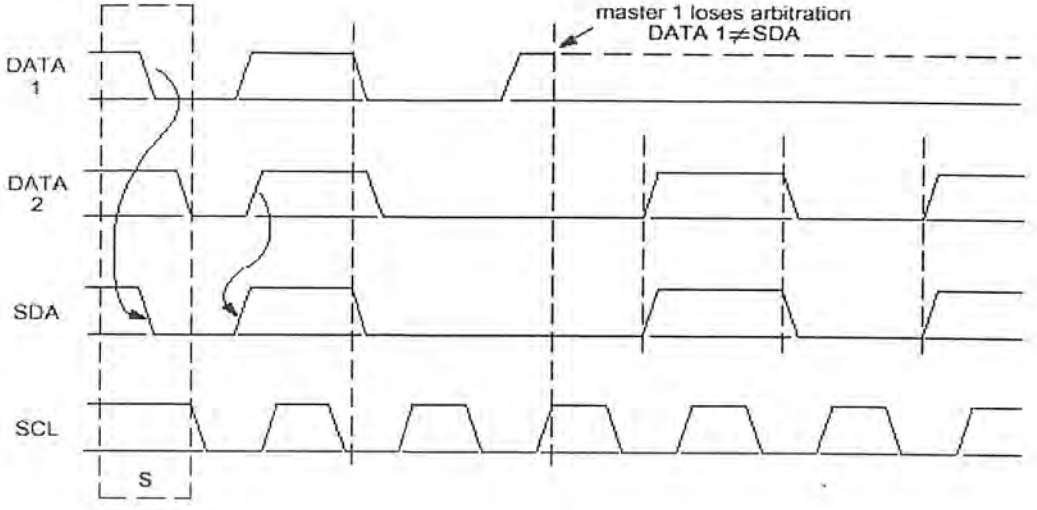
รูปที่ 5-7 การเข้าจังหวะของสัญญาณนาฬิกา

กระบวนการชี้ขาดเพื่อเข้าใช้บัสคล้ายกับการเข้าจังหวะของสัญญาณนาฬิกาที่ได้กล่าวมา เพียงแต่ว่าจะเป็นการพิจารณาค่าของสายสัญญาณ SDA ในขณะที่ SCL มีค่าเป็น HIGH เท่านั้น เมื่อเกิดการชนกันของข้อมูล มาสเตอร์ตัวที่ส่งค่า HIGH ออกมาในขณะที่มาสเตอร์ตัวอื่นส่งค่า LOW ก็จะสูญเสียสิทธิในการใช้บัส เนื่องจากการชนกันของข้อมูล HIGH กับ LOW จะมีผลให้ค่าบน SDA มีค่าเป็น LOW อุปกรณ์ตัวที่ส่งค่า HIGH ออกมา แต่อ่านค่าของสาย SDA ได้เป็น LOW ก็จะกลับไปสู่สถานะก่อนการส่งข้อมูล และ รอจนกว่าบัสจะว่างอีกครั้งหนึ่ง จึงจะเริ่มส่งข้อมูลใหม่

ในกรณีที่อุปกรณ์หลายตัวนั้น ส่งค่า LOW ออกมาเหมือนกัน กระบวนการชี้ขาดเพื่อเข้าใช้บัสก็จะเปรียบเทียบ ต่อไป เรื่อยๆ ตั้งแต่ช่วงการส่งแอดเดรส ถึงช่วงของการส่งข้อมูล จนกว่าจะสามารถแบ่งแยกได้ กล่าวคือ กระบวนการชี้ขาดเพื่อเข้าใช้บัสจะดำเนินไปพร้อมกับการส่งข้อมูล เมื่อมีอุปกรณ์ตัวใดที่ส่งข้อมูลออกมาเป็น HIGH ก่อน ตัวนั้นก็จะหมดสิทธิในการเข้าใช้บัส

ถ้าอุปกรณ์ที่เป็นได้ทั้งมาสเตอร์ และ สเลฟเมื่อเกิดกระบวนการชี้ขาดเพื่อเข้าใช้บัส และ มาสเตอร์ตัวนั้นสูญเสียสิทธิในการเข้าใช้บัส ในขณะที่กำลังส่งแอดเดรส จะต้องสลับกลับมาทำงานเป็นสเลฟทันที เนื่องจาก อาจเป็นไปได้ว่ามีอุปกรณ์มาสเตอร์ตัวอื่นกำลังต้องการติดต่อกับตัวมันอยู่ก็เป็นได้

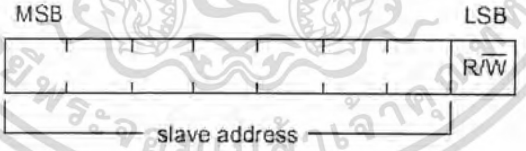
รูปที่ 5-8 แสดงกระบวนการชี้ขาดเพื่อเข้าใช้บัสที่มีอุปกรณ์มาสเตอร์สองตัว จะเห็นได้ว่า ในขณะที่ DATA1 มีค่าเป็น HIGH แต่ DATA2 มีค่าเป็น LOW ค่าของ SCL จะเป็น LOW (เนื่องจากการเชื่อมขาสัญญาณเข้ากับ SCL แบบ Wired-AND) ทำให้มาสเตอร์ที่ให้กำเนิด DATA1 สูญเสียสิทธิในการใช้บัส เนื่องจากอ่านค่าบน SCL ได้ไม่ตรงกับค่าที่ตัวเองส่งมา



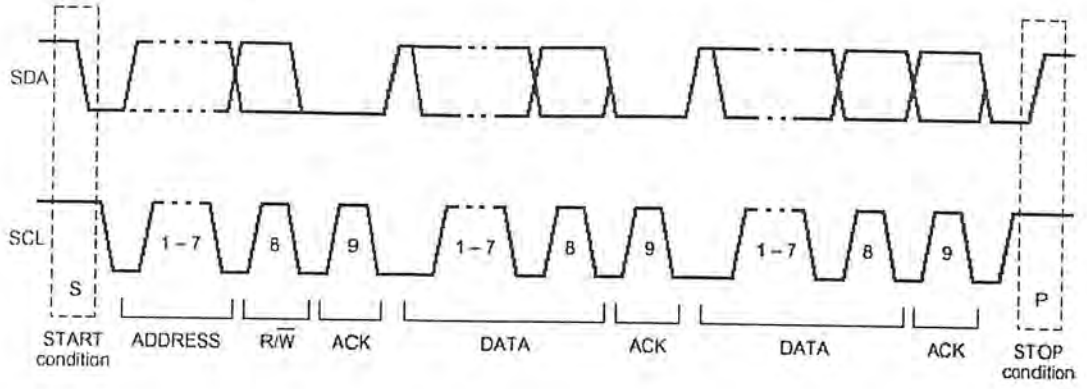
รูปที่ 5-8 แสดงกระบวนการ Arbitration และ การสูญเสียสิทธิในการเข้าใช้บัส

5.6 โปรโตคอลการรับส่งข้อมูลโดยใช้แอดเดรสแบบ 7 บิต

การรับส่งข้อมูลเริ่มจากการส่งเงื่อนไขการเริ่มต้น (S) แล้ว จะตามด้วยการส่งแอดเดรสของสเลฟที่ต้องการติดต่อ ซึ่งมีขนาด 7 บิต พร้อมกับบิตสุดท้าย (บิตที่ 8,R/$\overline{W}$) ซึ่งใช้สำหรับบอกทิศทางการรับส่งข้อมูล โดยค่า '0' หมายถึงการส่งข้อมูล (มาสเตอร์เขียนไปที่สเลฟ , WRITE) และ ค่า '1' หมายถึงการรับข้อมูล (มาสเตอร์อ่านจากสเลฟ , READ) (รายละเอียดของไบต์แรก ดูรูปที่ 5-9) จากนั้นตามด้วยการข้อมูลแต่ละไบต์ที่ต้องการส่งซึ่งแต่ละไบต์จะคั่นด้วยสัญญาณตอบรับ ของตัวรับ แล้วจบด้วยการรับส่งเงื่อนไขการยุติ (P) (ดูรูปที่ 5-10)



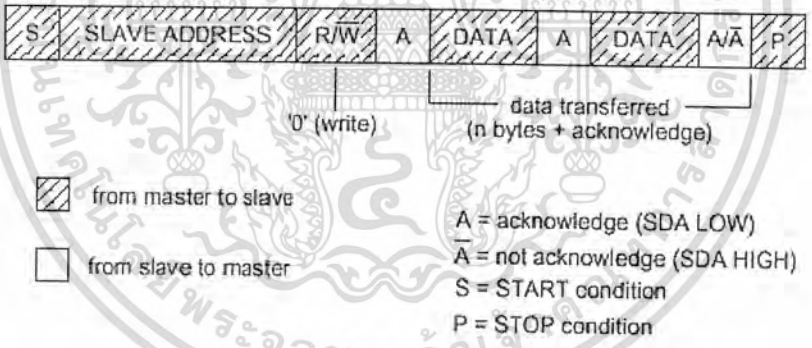
รูปที่ 5-9 ไบต์แรกของการรับส่งข้อมูลหลังจากเงื่อนไขการเริ่มต้น



รูปที่ 5-10 การรับส่งข้อมูลที่เสร็จสมบูรณ์

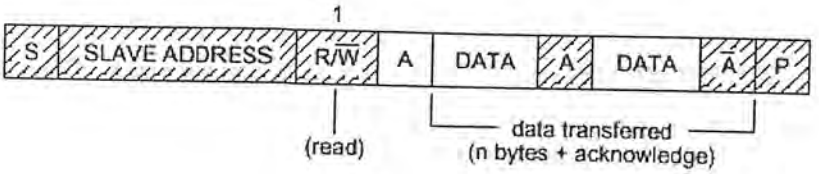
อย่างไรก็ตามหากมาสเตอร์ต้องการติดต่อกับอุปกรณ์สเลฟตัวอื่นก็สามารถส่งเงื่อนไขการเริ่มต้นซ้ำ (Sr) แล้วตามด้วยแอดเดรสของอุปกรณ์ตัวต่อไปได้ โดยที่ไม่จำเป็นต้องส่งเงื่อนไขการยุติก่อน นอกจากนั้นยังสามารถเปลี่ยนรูปแบบสลับกันระหว่างการเขียนกับการอ่าน ได้อีกด้วย รูปแบบของการรับส่งข้อมูลที่เป็นไปได้มีดังต่อไปนี้

- มาสเตอร์เป็นตัวส่งส่วนสเลฟเป็นตัวรับอย่างเดียว และ ไม่มีการเปลี่ยนทิศทางการสื่อสารตลอดการรับส่งข้อมูล (ดูรูปที่ 5-11)



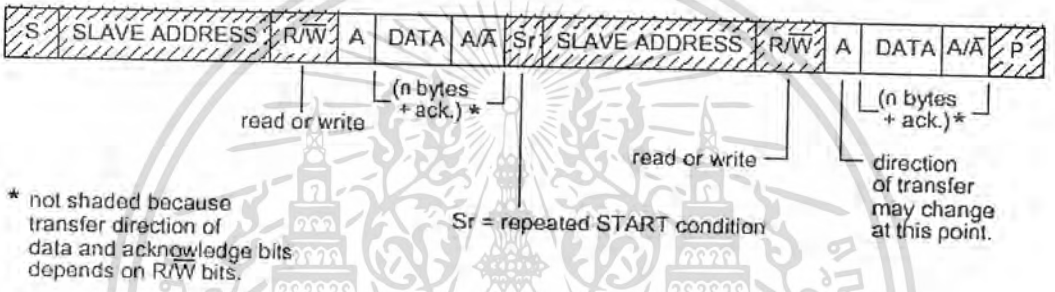
รูปที่ 5-11 มาสเตอร์เป็นตัวส่ง และ สเลฟเป็นตัวรับ ในการแอดเดรสแบบ 7 บิต (การเขียน,WRITE)

- มาสเตอร์เปลี่ยนไปทำหน้าที่อ่านข้อมูลจากสเลฟทันทีหลังจากที่มีการส่งสัญญาณรับทราบมาจาก สเลฟเป็นครั้งแรก เมื่อต้องการสิ้นสุดการอ่านข้อมูล หลังจากอ่านบิตสุดท้ายเป็นที่เรียบร้อย มาสเตอร์จะไม่ส่งสัญญาณรับทราบ แต่จะปล่อยค่าของ SDA เป็น HIGH แล้วส่งเงื่อนไขการยุติ (ดูรูปที่ 5-12)



รูปที่ 5-12 มาสเตอร์กลายเป็นตัวรับทันทีหลังจากส่งไบต์แรก (การอ่าน,READ)

- แบบผสม กล่าวคือ มาสเตอร์สามารถสลับการอ่าน และ เขียนสเลฟได้ โดยการส่งเงื่อนไขการเริ่มต้น พร้อมกับแอดเดรส ซึ่งแต่เปลี่ยนบิตสุดท้ายแทน ตัวอย่างในการรับส่งข้อมูลแบบนี้ เช่น การอ่านและเขียนข้อมูลจากหน่วยความจำสลับกัน (ดูรูปที่ 5-13)



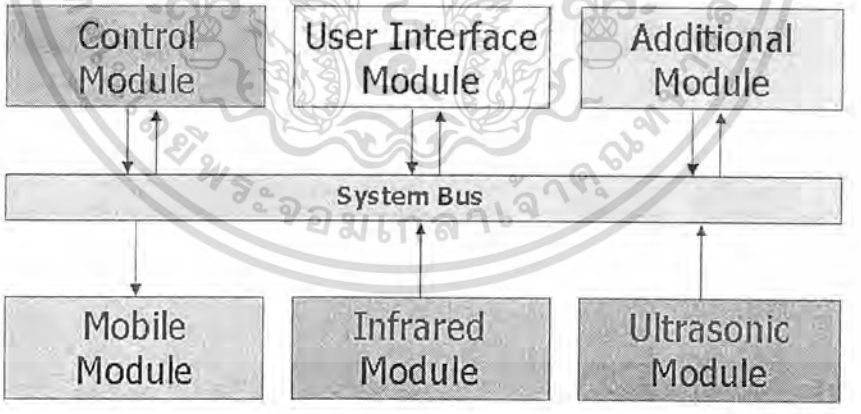
รูปที่ 5-13 การรับส่งข้อมูลแบบผสม

บทที่ 6
ขั้นตอนการออกแบบ

ชุดพัฒนาหุ่นยนต์ขนาดเล็กนี้จะประกอบด้วยโมดูลที่ทำงานแต่ละด้านแตกต่างกัน และมีโมดูลที่ทำหน้าที่ควบคุมการทำงานโมดูลอื่นๆ ที่ติดตั้งอยู่ภายในตัวหุ่น ซึ่งโมดูลแต่ละโมดูลออกแบบให้ใช้ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ในการควบคุมการทำงานของโมดูล และมีโมดูล I/O ที่ออกแบบโดยใช้ CPLD ในการควบคุมการทำงาน 1 โมดูลซึ่งเป็นโมดูลต้นแบบที่ได้ทดลองสร้างขึ้นมา โดยที่โมดูลหลักจะทำหน้าที่ส่งคำสั่งไปยังโมดูล I/O อื่นๆ ผ่านทางบัสของระบบที่ออกแบบขึ้น จากนั้นโมดูล I/O ที่ได้รับคำสั่งจะทำงานตามคำสั่งที่ได้รับต่อไปอย่างอิสระจนกว่าจะได้รับคำสั่งใหม่ ในกรณีที่โมดูลใดไม่จำเป็นในการทำงานของหุ่นยนต์ก็สามารถถอดออกได้ และ ในกรณีที่โมดูลที่จำเป็นต่อการทำงานของตัวหุ่นถูกถอดออกไป โมดูลควบคุมหลักจะสามารถละเลยการสั่งงานโมดูลนั้นๆ ไปได้ โดยยังทำงานกับโมดูลอื่นๆ ต่อไปได้อย่างปกติ

6.1 บล็อกไดอะแกรมของระบบ

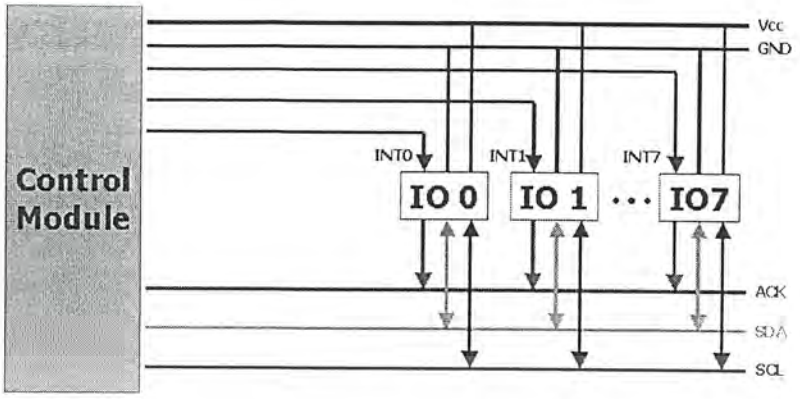
การทำงานของหุ่นจะประกอบด้วยโมดูลควบคุม 1 โมดูล และ มีโมดูล I/O ที่ทำงานอิสระสำหรับงานแต่ละด้านแตกต่างกัน ติดต่อรับส่งข้อมูลและคำสั่งโดยผ่านทางบัสของระบบโดยที่โปรโตคอลการติดต่อสื่อสารระหว่างโมดูลควบคุมหลักกับโมดูล I/O ถูกกำหนดขึ้นมาเฉพาะและใช้บัส I2C ในการรับส่งข้อมูลระหว่างโมดูล บล็อกไดอะแกร์มของตัวหุ่นแสดงดังรูปที่ 6-1



รูปที่ 6-1 บล็อกไดอะแกรมของตัวหุ่น

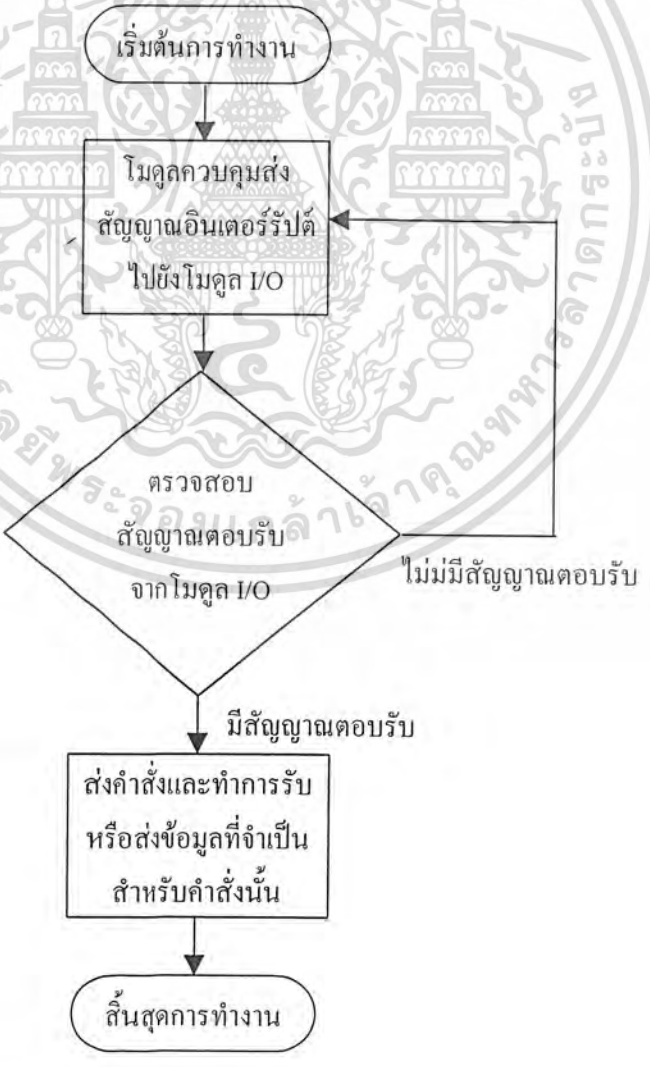
6.2 โปรโตคอลของบัสของระบบ

บัสของระบบประกอบไปด้วยสายส่งพลังงาน Vcc และ GND , สาย SDA และ SCL สำหรับรับ/ส่งข้อมูลแบบ I2C , สายสัญญาณอินเตอร์รัปต์ 8 เส้นสำหรับโมดูล I/O แต่ละโมดูล และสายสัญญาณตอบรับ ACK ร่วม 1 เส้น รวมทั้งหมด 13 เส้น โครงสร้างบัสของระบบแสดงในรูปที่ 6-2



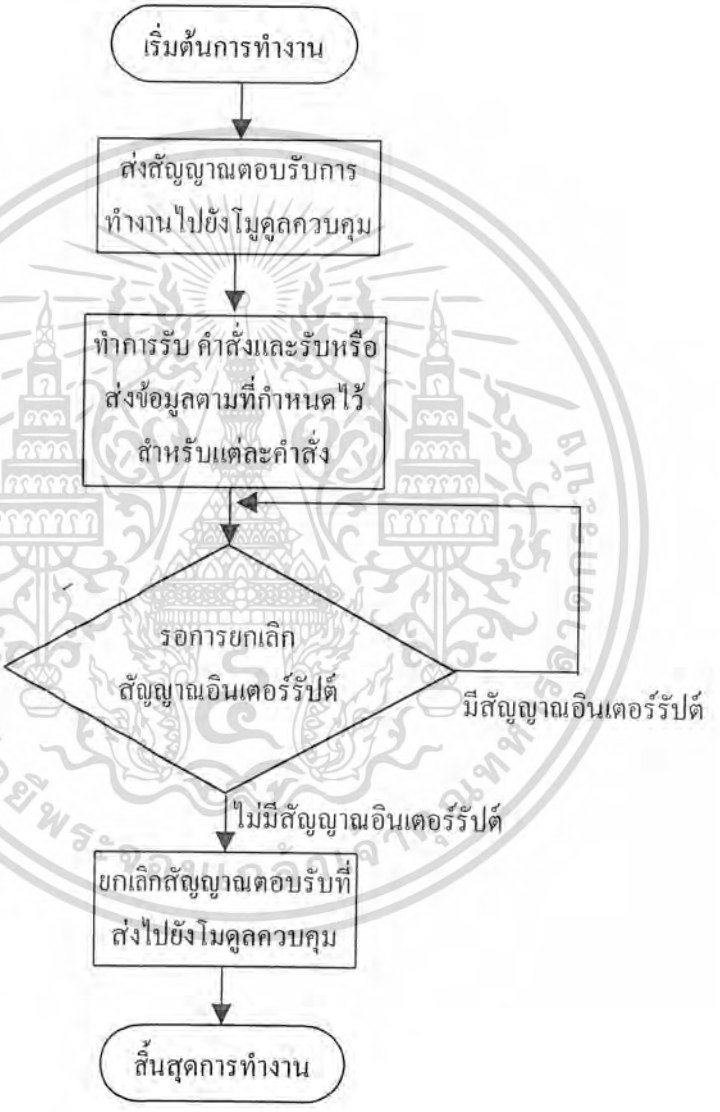
รูปที่ 6-2 โครงสร้างบัสของระบบ

6.2.1 ขั้นตอนในการสั่งงานและรับส่งข้อมูลบนระบบบัสของโมดูลควบคุม



ชุดคำสั่งที่ใช้สำหรับควบคุมการทำงานของโมดูล I/O จะมีลักษณะการทำงานโดยรวมดังโฟลชาร์ทข้างบน แต่จะมีการส่งสัญญาณอินเตอร์รัปต์และแอดเดรสของโมดูลแตกต่างกัน และ ข้อมูลที่ตามมาซึ่งอาจเป็นการส่งหรือรับข้อมูลจากโมดูล I/O

6.2.2 ขั้นตอนในการรับส่งข้อมูลของโมดูล I/O

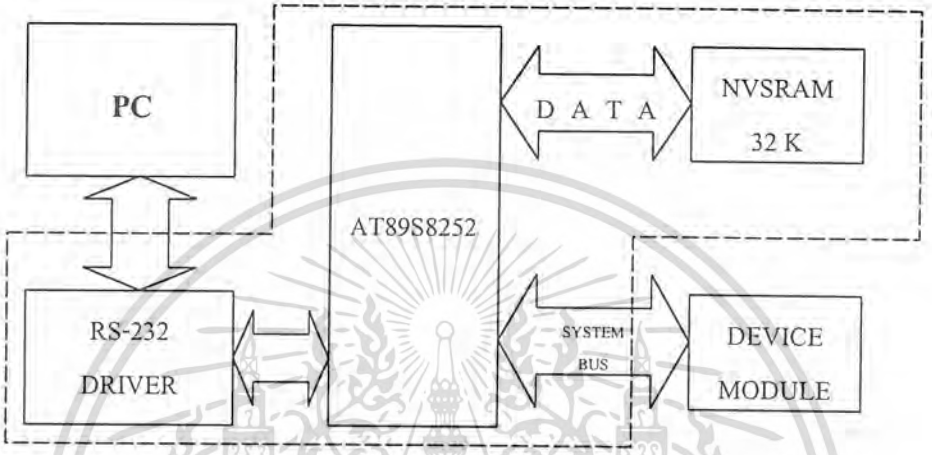


โมดูล I/O จะต่อขาอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์เข้ากับขาสัญญาณที่มาจากโมดูลควบคุมทำให้โปรแกรมส่วนของการรับคำสั่งของโมดูล I/O อยู่ภายในส่วนเฉพาะซึ่งจะมีการทำงานก็ต่อเมื่อมีการส่งสัญญาณอินเตอร์รัปต์มาเท่านั้น ในกรณีที่ไม่มีการส่งสัญญาณอินเตอร์รัปต์มา โมดูล I/O แต่ละโมดูลจะทำงานตามคำสั่งเดิมที่เคยได้รับจากโมดูลควบคุมหรือการทำงานตามที่ได้กำหนดไว้ของแต่ละโมดูล

6.3 โมดูลควบคุมหลัก

โมดูลควบคุมหลัก เป็นโมดูลที่ทำหน้าที่ควบคุมการทำงานของตัวหุ่น ซึ่งเป็นตัวที่ส่งข้อมูลและคำสั่งไปยังโมดูล I/O อื่นๆเพื่อให้ทำงาน รวมทั้งทำการรับข้อมูลที่โมดูล I/O อื่นส่งมาประมวลผลต่อไปโดยผ่านทางบัสของระบบ

โครงสร้างของโมดูลควบคุมหลักแสดงไว้ดังรูปที่ 6-3 ซึ่งมีรายละเอียดดังนี้

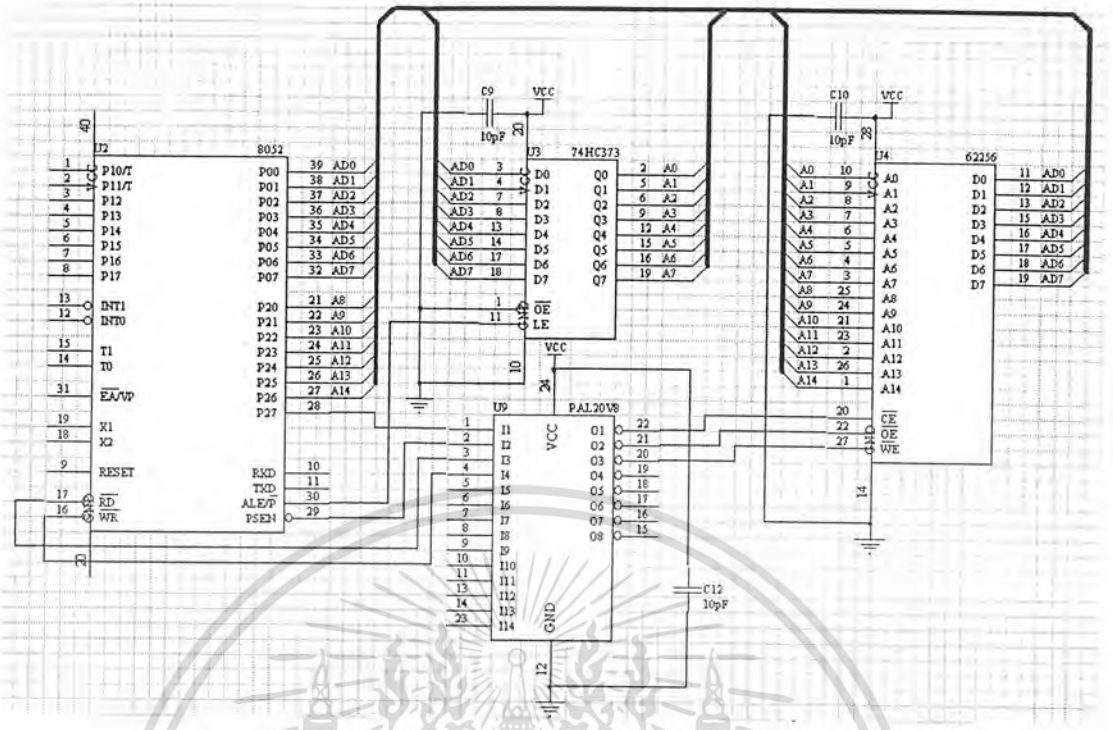


รูปที่ 6-3 โครงสร้างของโมดูลควบคุมหลัก

ไมโครคอนโทรลเลอร์ AT89S8252 ทำหน้าที่เป็นตัวประมวลผลหลักของตัวโมดูลโดยมีการเพิ่มหน่วยความจำชั่วคราวชนิดไม่สูญหาย (NVSRAM : Non Volatile SRAM) STK16C88 ขนาด 32 กิโลไบต์ ใช้สำหรับเก็บโปรแกรมที่ทำการโหลดมาจากเครื่องคอมพิวเตอร์ (PC) หรือทำการเก็บข้อมูล โดยใช้ไอซี HC74LS373 เป็นตัวแลซแอดเดรส คือทำหน้าเลือกสัญญาณแอดเดรสจากพอร์ต 0 ของไมโครคอนโทรลเลอร์ส่งไปยังขาแอดเดรสของหน่วยความจำก็ต่อเมื่อสัญญาณ ALE จากไมโครคอนโทรลเลอร์เป็นลอจิกต่ำเท่านั้น ดังรูปที่ 6-4

ไอซีโปรแกรมเกท PALCE20V8 ทำหน้าที่เลือกว่าจะอ้างอิงหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลจากหน่วยความจำชั่วคราว และอ้างอิงแอดเดรสของหน่วยความจำที่แอดเดรส 8000H ถึง FFFFH

การอ่านข้อมูลหรือโปรแกรมจากหน่วยความจำชั่วคราวทำได้โดยการเลือกสัญญาณจากขา PSEN และ RD เพียง 1 สัญญาณ เพื่อส่งไปยังขา OE ของหน่วยความจำชั่วคราวซึ่งไว้สำหรับอ่านข้อมูลจากหน่วยความจำชั่วคราว และการนำขา A15 จากพอร์ตที่ 2 บิตที่ 7 ของไมโครคอนโทรลเลอร์ ส่งไปยังขา CE ของหน่วยความจำเพื่อทำให้หน่วยความจำทำงาน ดังรูปที่ 6-4

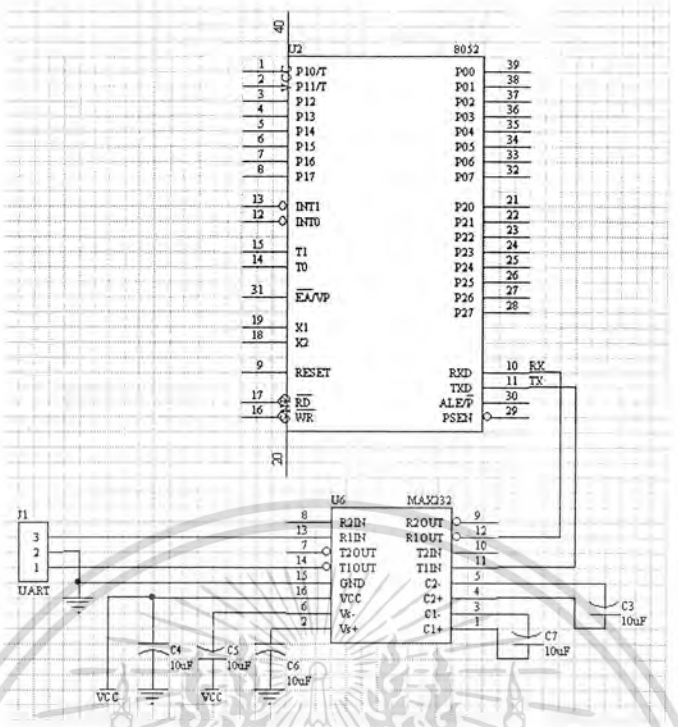


รูปที่ 6-4 การเชื่อมต่อกับ RAM ภายนอกโดยใช้ 74HC373 และ PAL20V8

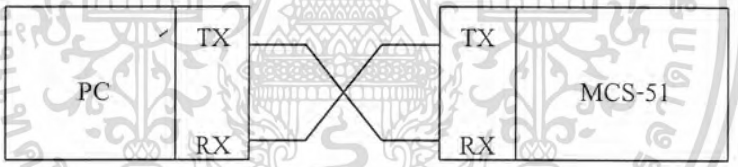
และมีไอซี MAX232 ซึ่งทำหน้าที่เป็นตัวแปลงระดับสัญญาณของข้อมูลแบบอนุกรมที่รับส่งระหว่างเครื่องคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ดังรูปที่ 6-5

การส่งข้อมูลแบบอนุกรมประกอบด้วยสายสัญญาณที่จำเป็น 3 เส้น ซึ่งประกอบด้วย สายส่งข้อมูล(Tx) สายรับข้อมูล(Rx) และ สายดิน(Ground) ซึ่งการรับส่งข้อมูลกับเครื่องคอมพิวเตอร์สายส่งข้อมูลของไมโครคอนโทรลเลอร์จะต้องส่งข้อมูลเข้าไปยังสายรับข้อมูลของเครื่องคอมพิวเตอร์ และสายรับข้อมูลของไมโครคอนโทรลเลอร์จะต่อเข้ากับสายส่งข้อมูลของเครื่องคอมพิวเตอร์ ดังรูปที่ 6-6

โมดูลควบคุมหลักจะทำงานตามโปรแกรมที่ได้บรรจุลงไปในไมโครคอนโทรลเลอร์ของโมดูล โดยที่การสั่งงานโมดูล I/O อื่นๆ ที่ต้องการจะสั่งงานโดยการเรียกใช้ชุดคำสั่งที่สร้างไว้ในคลังคำสั่งที่สร้างขึ้น รูปโมดูลควบคุมที่เสร็จสมบูรณ์แสดงในรูปที่ 6-7



รูปที่ 6-5 การเชื่อมต่อกับ MAX232 เพื่อใช้รับส่งข้อมูลแบบอนุกรม

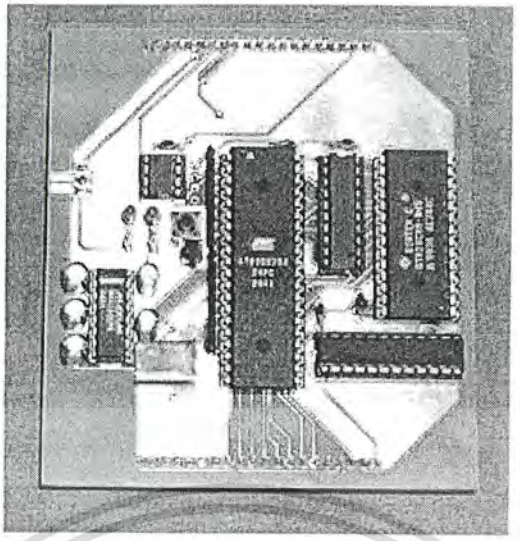


รูปที่ 6-6 การเชื่อมต่อสายสัญญาณอนุกรมระหว่างเครื่องคอมพิวเตอร์กับ MCS-51

ชุดคำสั่งที่มีออกแบบขึ้นเพื่อใช้สำหรับติดต่อสื่อสารกับโมดูล I/O แบ่งเป็นส่วนย่อยๆ ได้ดังนี้

1. ชุดคำสั่งสำหรับโมดูลขับเคลื่อน
2. ชุดคำสั่งสำหรับโมดูลตรวจสอบระดับพื้น
3. ชุดคำสั่งสำหรับโมดูลตรวจสอบสิ่งกีดขวาง
4. ชุดคำสั่งสำหรับโมดูลติดต่อกับผู้ใช้

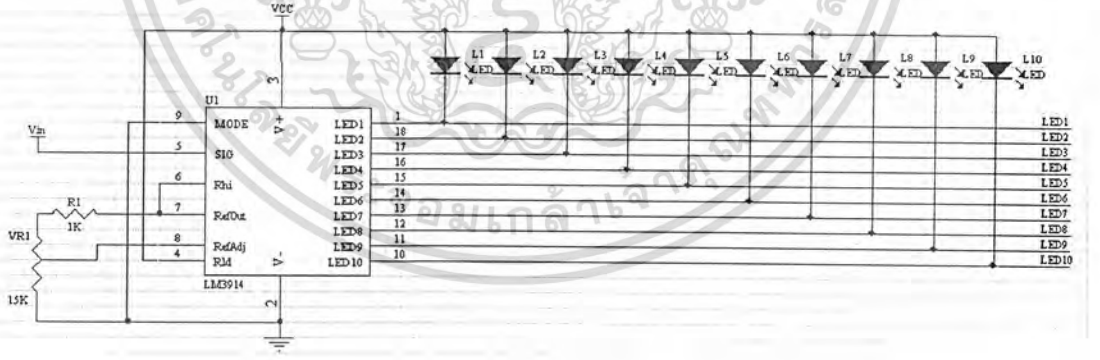
สำหรับชุดคำสั่งสำหรับแต่ละโมดูลที่ได้กล่าวมานี้จะอธิบายอีกครั้งในส่วนของการออกแบบโมดูล I/O แต่ละโมดูลอีกครั้ง



รูปที่ 6-7 โมดูลควบคุมหลักที่เสร็จสมบูรณ์

6.4 โมดูลขับเคลื่อน

ทำหน้าที่ขับเคลื่อนตัวหุ่นยนต์ ด้วยล้อ โดยใช้มอเตอร์เป็นอุปกรณ์ขับเคลื่อน อุปกรณ์ขับเคลื่อนมีอยู่ 2 ประเภท คือ เซอร์โวมอเตอร์ และ สเต็ปปิ้งมอเตอร์ โมดูลขับเคลื่อนจะรวมเอาแบตเตอรี่ และ วงจรวัดระดับพลังงานแบตเตอรี่เข้าไว้ภายในตัวโมดูลด้วย วงจรวัดระดับพลังงานแบตเตอรี่ประกอบด้วยไอซี LM3914 ทำหน้าที่วัดแรงดันและทำการส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ แสดงไว้ดังรูปที่ 6-8



รูปที่ 6-8 การใช้ LM3914 อ่านค่าแรงดันส่งให้ไมโครคอนโทรลเลอร์

ชุดคำสั่งที่ใช้ควบคุมการทำงานของโมดูลขับเคลื่อนมีดังนี้

1. คำสั่งเคลื่อนที่ไปข้างหน้า
2. คำสั่งเคลื่อนที่ถอยหลัง
3. คำสั่งหมุนตัวไปทางซ้าย

4. คำสั่งหมุนตัวไปทางขวา
5. คำสั่งหยุดการเคลื่อนที่
6. คำสั่งตั้งความเร็วในการเคลื่อนที่ (โมดูลขับเคลื่อนโดยใช้เซอร์โวมอเตอร์ ไม่มีคำสั่งนี้)
7. คำสั่งตั้งเวลาในการเคลื่อนที่
8. คำสั่งกำหนดลักษณะการป้อนพัลส์ให้สเต็ปปิ้งมอเตอร์ (มีเฉพาะโมดูลขับเคลื่อนโดยใช้สเต็ปปิ้งมอเตอร์เท่านั้น)

ชุดคำสั่งเหล่านี้สามารถใช้กับโมดูลขับเคลื่อนทั้ง 2 ประเภทได้ ดังนั้น เมื่อทำการเขียนโปรแกรมเพื่อใช้งานกับโมดูลขับเคลื่อนประเภทนึงแล้วสามารถแทนที่ด้วยโมดูลขับเคลื่อนประเภทอื่นได้

6.4.1 โมดูลขับเคลื่อนโดยใช้เซอร์โวมอเตอร์

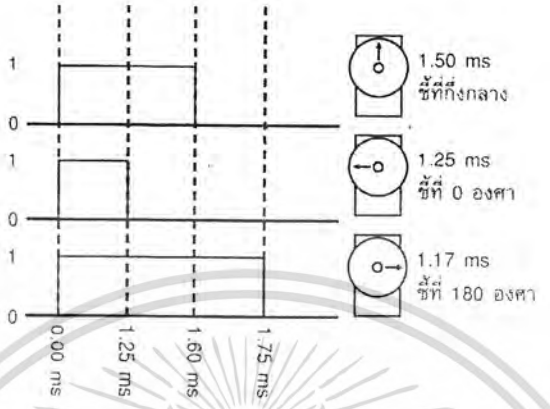
เซอร์โวมอเตอร์เป็นมอเตอร์ทดเฟืองขนาดเล็กโดยมีแกนส่งกำลัง 1 อัน โดยปกติเซอร์โวจะหมุนได้เพียง 180 องศาเท่านั้น หรือบางตัวอาจจะถึง 210 องศาขึ้นอยู่กับบริษัทผู้ผลิตและแกนส่งกำลังนี้สามารถที่จะควบคุมทิศทางที่หันไปหรือตำแน่งที่ต้องการได้ โดยการส่งรหัสควบคุมให้กับตัวเซอร์โว ตำแหน่งของแกนส่งสัญญาณดังกล่าวเข้าที่ขาอินพุตตลอดเวลา เซอร์โวสามารถนำไปประยุกต์ใช้งานกับอุปกรณ์ได้หลากหลาย รูปที่ 6-9 แสดงตัวอย่างเซอร์โวมอเตอร์ขนาดเล็กที่ใช้ทั่วไป

รูปที่ 6-9 ตัวอย่างเซอร์โวมอเตอร์ขนาดเล็กทั่วไป

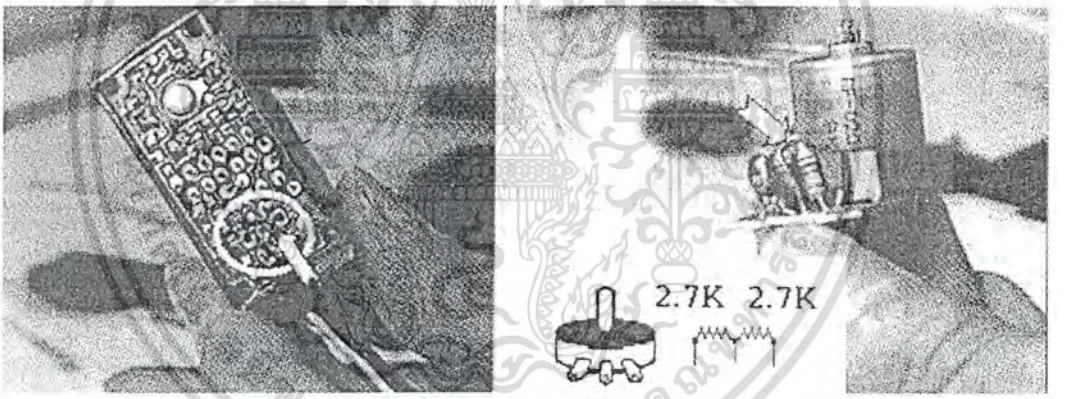
เซอร์โวมอเตอร์มีวงจรควบคุมการหมุดโดยใช้ตัวต้านทานปรับค่าได้หรือ VR ค่า 5 กิโลโอห์ม ตัวต้านทานตัวนี้จะติดอยู่กับแกนส่งกำลัง เพื่อใช้วัดระยะองศาของแกนหมุนโดยใช้ร่วมกับวงจรควบคุมการกำหนดมุมของเซอร์โว ถ้าแกนอยู่ในมุมที่ถูกต้องมอเตอร์ก็จะปิดเองอัตโนมัติ แต่ถ้าวงจรตรวจสอบพบว่ามุมยังไม่ถูกต้องมอเตอร์ก็จะหมุนไปในทางที่ถูกจนกระทั่งได้มุมที่ถูกต้อง และจะหยุดอยู่จนกว่าจะมีการป้อนข้อมูลเข้ามาใหม่ การกำหนดองศาของการหมุนทำโดยการส่งสัญญาณพัลส์ค่าระหว่าง 1.5 มิลลิวินาที ถึง 2 วินาที ดังรูปที่ 6-10 เป็นการแสดงลักษณะความกว้างของพัลส์ กับทิศทางที่เซอร์โวหมุนไป

ในการประยุกต์ใช้งานเป็นมอเตอร์ที่สามารถหมุนได้รอบทิศทางนั้น เราจะต้องทำการถอดตัวต้านทานปรับค่าออกและแทนด้วยตัวต้านทานค่าคงที่ที่ใกล้เคียงกันลงไปแทน ดังรูปที่ 6-11

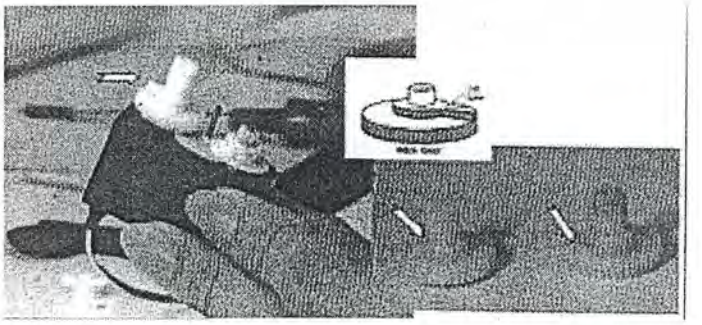
และ ทำการตัดขอบพลาสติกของชุดเฟืองทดที่ยื่นออกมาเพื่อกันการหมุนเกิน 180 องศา ดังรูปที่ 6-12



รูปที่ 6-10 ความกว้างของพัลส์ในการควบคุมทิศทางการหมุนของเซอร์โว

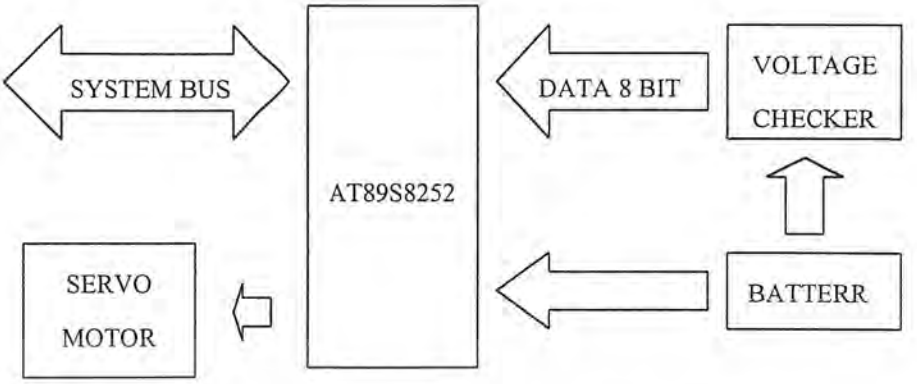


รูปที่ 6-11 การแทนที่ตัวต้านทานปรับค่าด้วยตัวต้านทานค่าคงที่ 2 ตัว



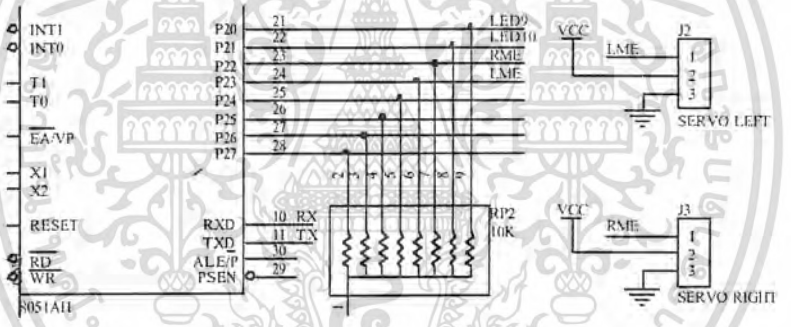
รูปที่ 6-12 แสดงการตัดรอยบากบนซี่เฟืองเพื่อให้สามารถหมุ่มได้มากกว่า 180 องศา

โครงสร้างการทำงานของโมดูลขับเคลื่อนโดยใช้เซอร์โวมอเตอร์แสดงไว้ดังรูปที่ 6-13



รูปที่ 6-13 โครงสร้างการทำงานของโมดูลขับเคลื่อนโดยใช้เซอร์โวมอเตอร์

ในการควบคุมการหมุนของเซอร์โวมอเตอร์นั้นทำโดยการสร้างสัญญาณพัลส์จากไมโครคอนโทรลเลอร์ไปเข้าสู่เซอร์โวมอเตอร์ดังรูปที่ 6-14

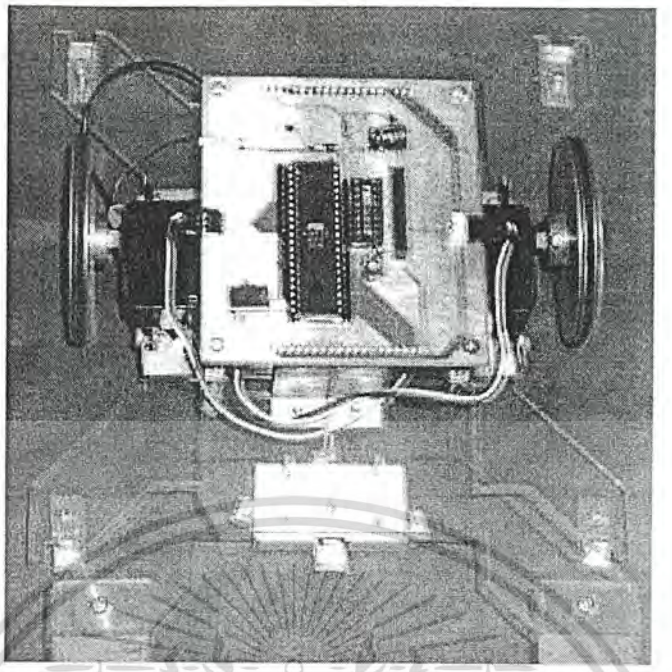


รูปที่ 6-14 การต่อสัญญาณพัลส์จากไมโครคอนโทลเลอร์ไปยังเซอร์โวมอเตอร์

การทำงานของโมดูลขับเคลื่อนโดยใช้เซอร์โวมอเตอร์แบ่งออกเป็น 3 ส่วนคือ

1. โค้ดหลักของโปรแกรมทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรที่เกี่ยวข้องกับการทำงานของโมดูล ตั้งเวลาการเกิดอินเตอร์รัปต์ให้กับอินเตอร์รัปต์ไทเมอร์ของไมโครคอนโทรลเลอร์ และกำหนดให้สามารถเกิดอินเตอร์รัปต์จากสัญญาณภายนอกได้
2. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากภายนอกเกิดขึ้นใช้สำหรับรับคำสั่งและรับส่งข้อมูลกับโมดูลควบคุมหลัก
3. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากไทเมอร์เกิดขึ้นใช้สำหรับสร้างพัลส์เพื่อควบคุมเซอร์โวมอเตอร์ให้หมุนตามคำสั่งที่ได้รับมาขณะนั้น

โมดูลขับเคลื่อนด้วยเซอร์โวมอเตอร์ที่เสร็จสมบูรณ์แสดงในรูปที่ 6-15



รูปที่ 6-15 โมดูลขับเคลื่อนด้วยเซอร์โวมอเตอร์ที่เสร็จสมบูรณ์

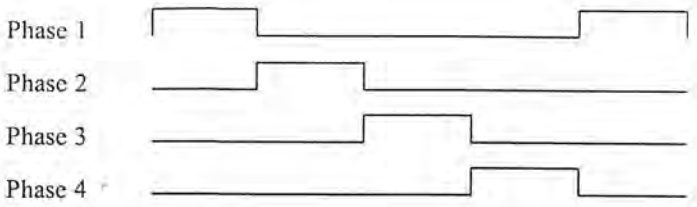
6.4.2 โมดูลขับเคลื่อนโดยใช้สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นมอ์เตอร์ชนิดหนึ่งซึ่งสามารถบังคับให้เพลาของมอเตอร์หมุนจากตำแหน่งเดมไปยังตำแหน่งถัดไปด้วยมุมค่าหนึ่งตามที่ต้องการอย่างแม่นยำ ทั้งในทิศทางทวนเข็มนาฬิกาและตามเข็มนาฬิกา ข้อจำกัดในการหมุนของเพลา คือจะหมุนได้ครั้งละ 1 มุมเท่านั้น โดยทั่วไปการหมุน 1 มุมนั้นจะได้ขนาด 1.8, 2 และ2.5 องศา

การควบคุมการทำงานของสเต็ปปิ้งมอเตอร์จะทำโดยการป้อนสัญญาณพัลส์เข้าไปกระตุ้นให้เกิดสนามแม่เหล็กเป็นช่วงๆ ต่อเนื่องกันไป การหมุนของสเต็ปปิ้งมอเตอร์จะขึ้นอยู่กับการขับกระแสเข้าทางขดลวดว่างต้องการทิศทางใด และยังสามารถควบคุมการหมุนหรือการเปลี่ยนแปลงมุมของเพลาได้อย่างแม่นยำ เนื่องจากเราใช้พัลส์ควบคุมการหมุนของมอเตอร์ ดังนั้นเราจึงสามารถทราบการขจัดของมอเตอร์ได้ด้วยการนับจำนวนพัลส์ที่ป้อนให้กับมอเตอร์นั้น (โดยการคูณค่ามุมของการเปลี่ยนไปใน 1 สเต็ป)

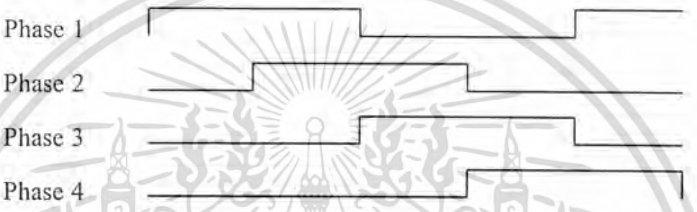
สเต็ปปิ้งมอเตอร์จะหมุนได้นั้นต้องมีการป้อนพัลส์กระตุ้นเข้าที่ลวดแต่ละเฟสของมอเตอร์ ซึ่งวิธีการป้อนพัลส์ที่ต่างกันนั้นสามารถทำได้ดังนี้

1. การป้อนพัลส์กระตุ้นเฟสเดียว (Single phase excitation) ซึ่งเป็นการป้อนพัลส์ให้กับขดลวดทีละขด ดังรูปที่ 6-16



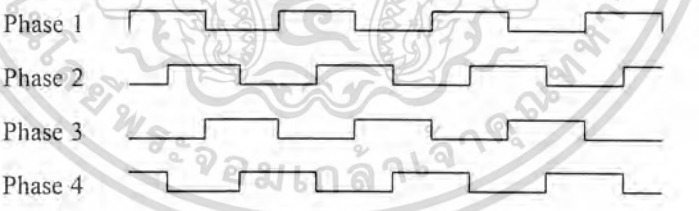
รูปที่ 6-16 การป้อนพัลส์กระตุ้นแบบเฟสเดียว

2. การป้อนพัลส์กระตุ้นสองเฟส (two phase excitation) เป็นการป้อนพัลส์ให้กับขดลวดทีละสองขด ดังรูปที่ 6-17



รูปที่ 6-17 การป้อนพัลส์กระตุ้นแบบสองเฟส

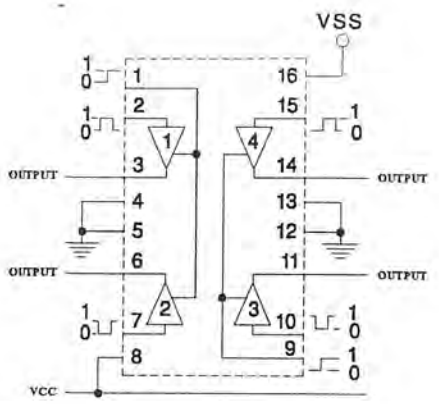
3. การป้อนพัลส์กระตุ้นแบบฮาล์ฟสเต็ป (half step excitation) การป้อนพัลส์แบบนี้จะทำให้มอเตอร์มีสเต็ปการหมุนที่ละเอียดขึ้น ดังรูปที่ 6-18



รูปที่ 6-18 การป้อนพัลส์กระตุ้นแบบครึ่งเฟส

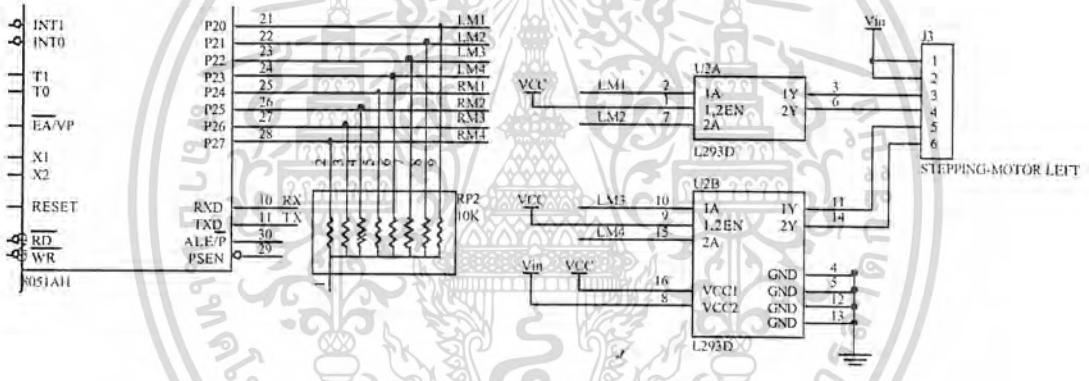
การป้อนพัลส์กระตุ้นแบบนี้มีความสำคัญมาก เพราะสามารถทำให้สเต็ปปิ้งมอเตอร์ที่มีความละเอียดในการหมุนต่ำสามารถมีความละเอียดในการหมุนเพิ่มขึ้นได้

วงจรที่ใช้ขับสเต็ปปิ้งมอเตอร์นี้ใช้เป็นไอซี L293D ภายในประกอบด้วยวงจรแปลงระดับลอจิกแบบทีทีแอลเป็นแรงดันเอาท์พุตตามแรงดันที่จ่ายให้กับไอซีเพื่อขับมอเตอร์จำนวน 4 ชุด ดังรูปที่ 6-19



รูปที่ 6-19 วงจรภายในของ L293D ประกอบด้วยวงจรขับจำนวน 4 ชุด

ในการขับสเต็ปปิ้งมอเตอร์นั้น จะต้องนำขาสัญญาณควบคุมทั้ง 4 เส้นต่อเข้ากับขาอินพุตของไอซี L293D แต่ละขาเพื่อขับขดลวดของสเต็ปปิ้งมอเตอร์แต่ละเฟส ดังรูปที่ 6-20

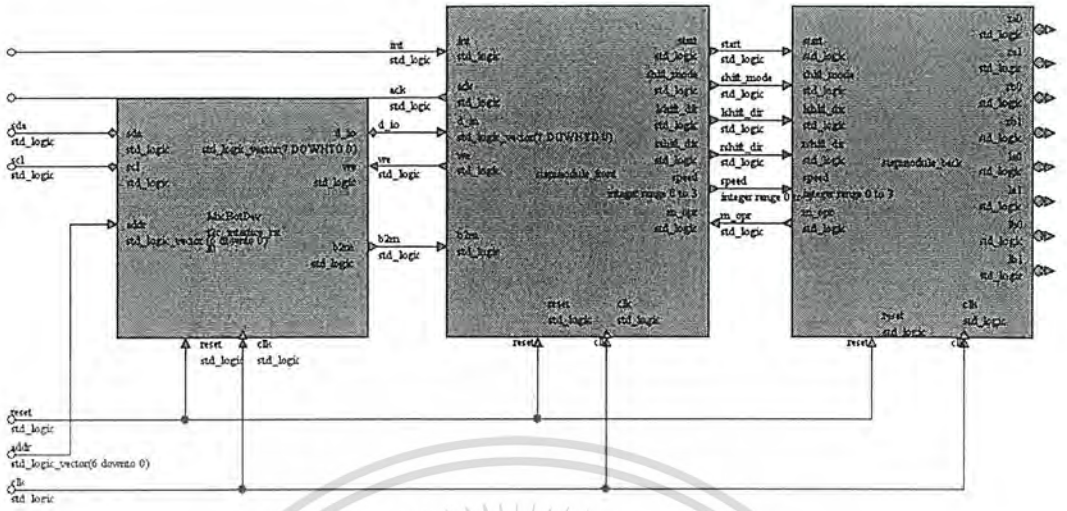


รูปที่ 6-20 การใช้ L293D ขับสัญญาณควบคุมสเต็ปปิ้งมอเตอร์แต่ละเฟส

โมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์นี้ มี 2 แบบคือแบบที่สร้างด้วยซีพีแอลดี และ แบบที่สร้างด้วยไมโครคอนโทรลเลอร์

6.4.2.1 โมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์สร้างด้วยซีพีแอลดี

โครงสร้างการทำงานของโมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์ที่สร้างด้วยซีพีแอลดี ประกอบไปด้วย 3 ส่วน ได้แก่ ส่วนรับข้อมูล I2C , ส่วนควบคุมส่วนหน้า และ ส่วนควบคุมส่วนหลัง ซึ่งแต่ละส่วนถูกสร้างอยู่บนซีพีแอลดีแต่ละตัว ดังรูปที่ 6-21



รูปที่ 6-21 โครงสร้างของโมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์ที่สร้างด้วยซีพีแอลดี

ส่วนรับข้อมูล I2C

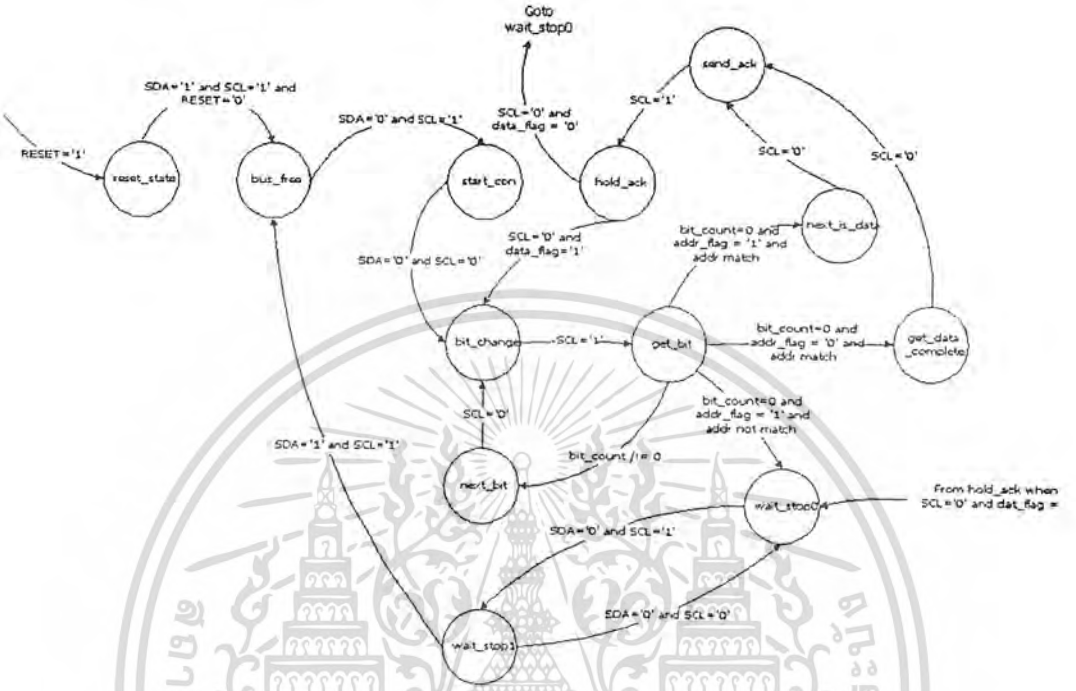
- Input/Output

สัญญาณ	ทิศทาง	คำอธิบาย
SDA	INOUT	สัญญาณข้อมูลบัส I2C
SCL	IN	สัญญาณนาฬิกาบัส I2C
ADDRESS(6..0)	IN	แอดเดรสของโมดูล
D_IO	OUT	ข้อมูลขนานส่งไปยังส่วนควบคุมส่วนหน้า
B2M	OUT	เปลี่ยนจาก LO เป็น HI เมื่อมีข้อมูลเข้ามา
WE	IN	Write Enable (Active Low)
RESET	IN	สัญญาณรีเซ็ตระบบ
CLK	IN	สัญญาณนาฬิกาของระบบ

- การทำงาน

หลังจากระบบรีเซ็ตค่าของตัวเอง เมื่อ SDA และ SCL เป็น HI ระบบจะไปอยู่ในสถานะบัสว่าง (bus_free) จากนั้นเมื่อมีการตรวจพบเงื่อนไขเริ่มต้น (SDA เป็น LO และ SCL เป็น HI) ระบบจะเริ่มต้นรับข้อมูลแต่ละบิตเข้ามาเมื่อสัญญาณ SCL มีการเปลี่ยนจาก LO เป็น HI ตามการรับส่งข้อมูลแบบ I2C (สถานะ bit_change,get_bit,next_bit ในรูป) และเมื่อรับข้อมูลครบ 1 ไบต์แล้ว จะตรวจสอบค่าที่รับได้กับแอดเดรสของอุปกรณ์ว่าตรงกันหรือไม่ ถ้าไม่ตรงจะไปยังสถานะรอการสิ้นสุด แต่หากตรงส่งสัญญาณตอบรับ (สถานะ send_ack,hold_ack) แล้วรับไบต์ต่อไป ซึ่งเป็นไบต์ข้อมูล เมื่อรับไบต์ที่สองเป็นที่เรียบ

ร้อยก็จะส่งค่านั้นไปยังขาสัญญาณ D_IO แล้วส่งสัญญาณ B2M เป็น HI เพื่อบอกให้ส่วนควบคุมส่วนหน้าทราบ แล้วกลับไปสู่สถานะรอการสิ้นสุด ส่วนสัญญาณ B2M จะกลับเป็น LO อีกครั้งเมื่อบัสว่าง แผนภาพสถานะของส่วนรับข้อมูล I2C แสดงในรูปที่ 6-22



รูปที่ 6-22 แผนภาพสถานะของส่วนรับข้อมูล I2C

ส่วนควบคุมส่วนหน้า

- Input/Output

สัญญาณ	ทิศทาง	คำอธิบาย
INT	IN	สัญญาณอินเตอร์รัปต์จากโมดูลควบคุมหลัก
ACK	OUT	สัญญาณตอบรับไปยังโมดูลควบคุมหลัก
B2M	IN	เปลี่ยนจาก LO เป็น HI เมื่อมีสัญญาณเข้า
D_IN (7..0)	IN	สัญญาณข้อมูลแบบขนาน
START	OUT	เป็น HI เมื่อสั่งให้มอเตอร์ทำงาน
SPEED (1..0)	OUT	ความเร็วของมอเตอร์มีค่าตั้งแต่ 0-3
LSHIFT_DIR	OUT	ทิศทางของมอเตอร์ซ้าย และ ขวา โดย LO คือ การหมุนทวนเข็มนาฬิกา
RSHIFT_DIR	OUT	และ HI คือการหมุนตามเข็มนาฬิกา

SHIFT_MODE	OUT	เป็น LO เมื่อต้องการขับแบบ Double Pharse เป็น HI เมื่อต้องการขับแบบ Single Pharse
M_OPR	IN	เปลี่ยนจาก LO เป็น HI เมื่อมีการหมุน 1 สเต็ป
WE	OUT	เป็น LO ตลอดเวลา
RESET	IN	สัญญาณรีเซ็ตระบบ
CLK	IN	สัญญาณนาฬิกาของระบบ

- การทำงาน

เมื่อเริ่มต้นการทำงาน ค่าของขาสัญญาณต่างๆจะมีค่าดังต่อไปนี้

START	LO
LSHIFT_DIR	LO
RSHIFT_DIR	HI
SPEED	2
SHIFT_MODE	LO
STEP_COUNT_FLAG	LO

ในกรณีที่ยังไม่มีการส่งข้อมูลใดๆจากโมดูลควบคุมหลักเข้ามา หากระบบถูกตั้งค่าให้มีการนับจำนวนสเต็ป ระบบจะอยู่ที่สถานะ IDLE ทุกครั้งที่มีการเปลี่ยนแปลงสัญญาณ M_OPR จาก LO เป็น HI จะนับจำนวนสเต็ปที่ตั้งค่าไว้ลงไปเรื่อยๆ จนหมด และ เมื่อจำนวนที่ตั้งไว้เป็น 0 ก็จะสั่งให้มอเตอร์หยุดหมุน

ในกรณีที่ระบบอยู่ในสถานะ IDLE และโมดูลควบคุมหลักส่งสัญญาณ INT เป็น LO เข้ามา ระบบจะตอบรับโดยการส่งสัญญาณ ACK เป็น LO กลับไป (สถานะ send_ack) จากนั้นจะรอจนกว่าส่วนรับข้อมูล I2C จะส่งสัญญาณ B2M เป็น HI ออกมา จึงจะอ่านค่าจากขาสัญญาณ D_IN เข้ามาทำงาน โดยคำสั่งที่เข้ามาจะมี 3 ประเภทคือ คำสั่งที่ไม่ต้องการพารามิเตอร์,คำสั่งที่มี 1 พารามิเตอร์ และ สัญญาณที่มี 2 พารามิเตอร์ คำสั่งแต่ละคำสั่งจะมีลักษณะการทำงานต่อไปนี้

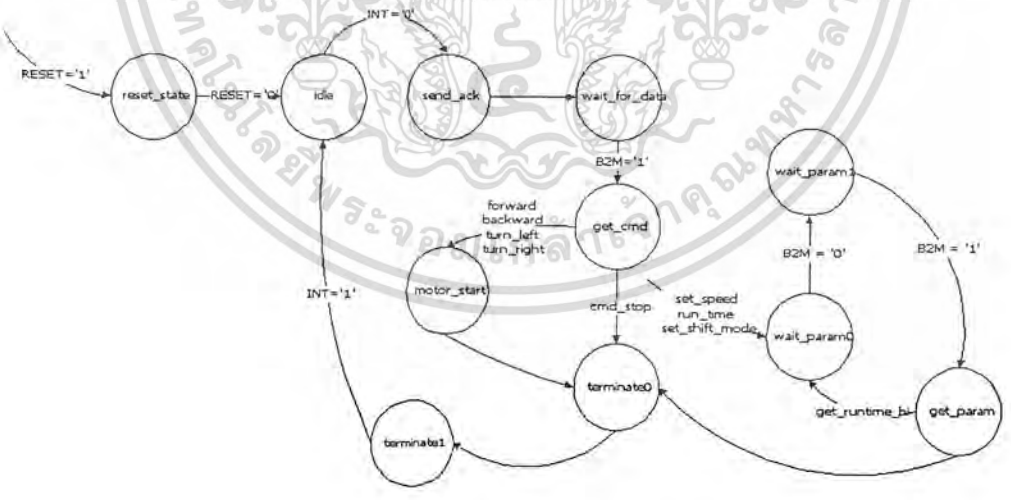
- คำสั่ง GO_FORWARD , GO_BACKWARD, TURN_LEFT , TURN_RIGHT และ STOP เป็นคำสั่งที่ไม่ต้องการพารามิเตอร์ โดยแต่ละคำสั่งมีผลการทำงานดังนี้

คำสั่ง	START	LSHIFT_DIR	RSHIFT_DIR
GO_FORWARD	HI	LO	HI
GO_BACKWARD	HI	HI	LO

TURN_LEFT	HI	HI	HI
TURN_RIGHT	HI	LO	LO
STOP	HI	X	X

- คำสั่ง SET_SPEED , SET_SHIFT_MODE เป็นคำสั่งที่ต้องการพารามิเตอร์ 1 ตัวโดยเมื่อรับคำสั่งเข้ามา ระบบจะตั้งค่า cmd_param เป็น cmd_set_speed หรือ cmd_shift_mode ตามลำดับ จากนั้นจะไปสู่สถานะ wait_param0 เพื่อรอรับค่าของพารามิเตอร์ต่อไป คำสั่งทั้งสอง มีผลดังต่อไปนี้
  - SET_SPEED จะตั้งค่า SPEED ตามพารามิเตอร์ที่ได้รับมา
  - SET_SHIFT_MODE จะตั้งค่าให้ SHIFT_MODE เป็น LO เมื่อพารามิเตอร์เป็น “00000001” แต่จะตั้งค่าให้ SHIFT_MODE เป็น HI เมื่อพารามิเตอร์มีค่าเป็นค่าอื่น
- คำสั่ง RUN_TIME เป็นคำสั่งที่รับพารามิเตอร์ 2 ตัวคือ RUNTIME_HI และ RUNTIME_LO โดยเมื่อรับคำสั่งนี้มาแล้ว ระบบจะตั้งค่า STEP_COUNT_FLAG เป็น HI และเริ่มนับถอยหลังเมื่อมีการหมุนของมอเตอร์ 1 สเต็ป

หลังจากรับคำสั่งเป็นที่เรียบร้อยแล้ว ระบบจะไปอยู่ที่สถานะ terminate0 ซึ่งเป็นสถานะที่จะส่งสัญญาณ ACK เป็น HI แล้วเปลี่ยนไปยังสถานะ terminate1 เพื่อรอให้โมดูลควบคุมหลักส่งสัญญาณ INT เป็น HI กลับมาแล้วกลับสู่สถานะ IDLE อีกครั้งหนึ่ง แผนภาพสถานะของส่วนควบคุมส่วนหน้าแสดงในรูปที่ 6-23



รูปที่ 6-23 แผนภาพแสดงสถานะของส่วนควบคุมส่วนหน้า

ส่วนควบคุมส่วนหลัง

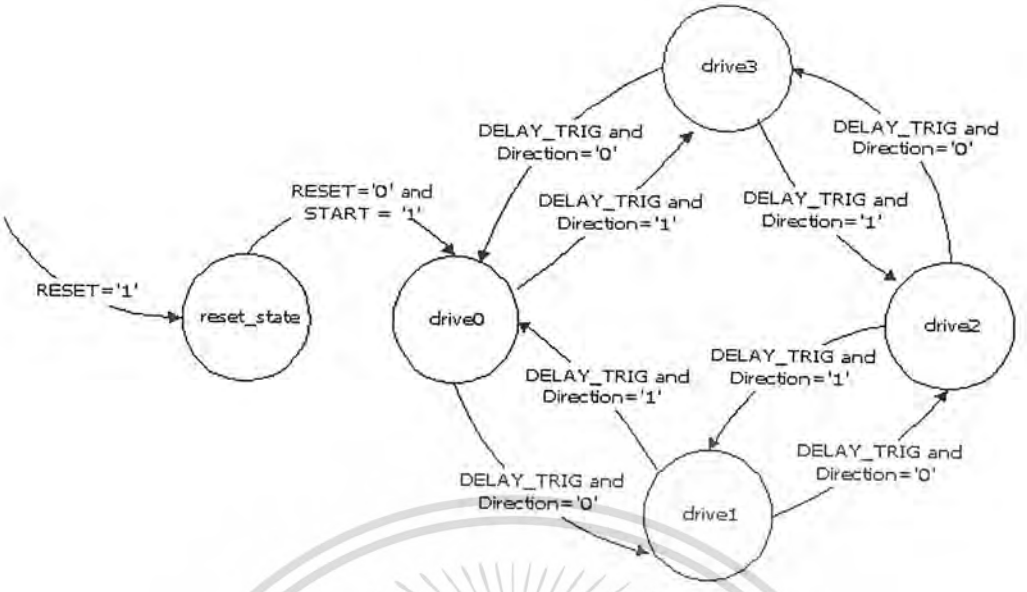
- Input/Output

สัญญาณ	ทิศทาง	คำอธิบาย
START	IN	มอเตอร์ทำงานเมื่อ START เป็น HI
LSHIFT_DIR	IN	ทิศทางของมอเตอร์ซ้าย และ ขวา โดย LO คือ การหมุนทวนเข็มนาฬิกา และ HI คือการหมุนตามเข็มนาฬิกา
RSHIFT_DIR	IN	
SPEED (1..0)	IN	ความเร็วของมอเตอร์มีค่าตั้งแต่ 0-3
SHIFT_MODE	IN	เป็น LO เมื่อต้องการขับแบบ Double Pharse เป็น HI เมื่อต้องการขับแบบ Single Pharse
M_OPR	OUT	เปลี่ยนจาก LO เป็น HI เมื่อมีการหมุน 1 สเต็ป
LA0,LA1,LB0,LB1	OUT	สัญญาณขับสเต็ปปิ้งมอเตอร์ซ้าย และ ขวา
RA0,RA1,RB0,RB1	OUT	
RESET	IN	สัญญาณรีเซ็ตระบบ
CLK	IN	สัญญาณนาฬิกาของระบบ

- การทำงาน

ส่วนควบคุมส่วนหลังทำหน้าที่ขับมอเตอร์เมื่อสัญญาณ START เป็น HI โดยภายในจะมีตัวนับตัวหนึ่ง ทำหน้าที่หารความถี่ของสัญญาณนาฬิกาที่เข้ามา โดยเมื่อตัวนับตัวนี้นับจนถึง 0 จะไปทำให้ตัวนับความเร็วมีค่าลดลง 1 และ เมื่อตัวนับความเร็วมีค่าถึง 0 เมื่อไหร่ ก็จะอ่านค่าของความเร็วจาก SPEED มาเก็บไว้ แล้วสั่งให้ สเตทแมชชีนของมอเตอร์ซ้าย และ ขวาเปลี่ยนสถานะตามรูปที่ 6-24 โดยแต่ละสถานะจะมีค่าของสัญญาณขับดังตารางต่อไปนี้

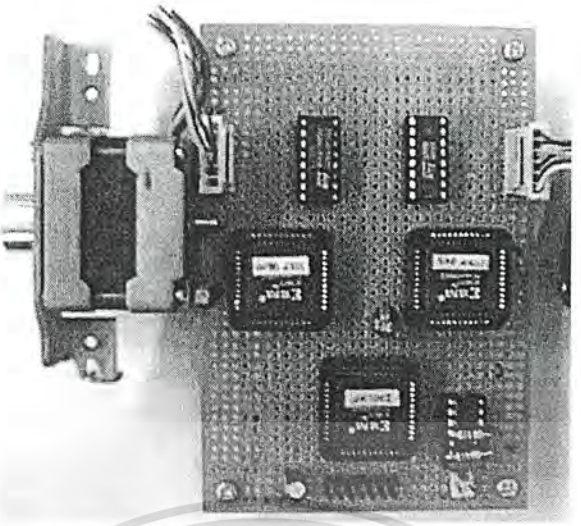
สถานะ	สัญญาณขับ (A0,A1,B0,B1)	
	SHIFT_MODE = LO	SHIFT_MODE = HI
drive0	0111	0011
drive1	1011	1001
drive2	1101	1100
drive3	1110	0110



รูปที่ 6-24 แผนภาพสถานะของการขับสเต็ปปิ้งมอเตอร์แต่ละข้าง

การทำงานของโมดูลทั้งหมด

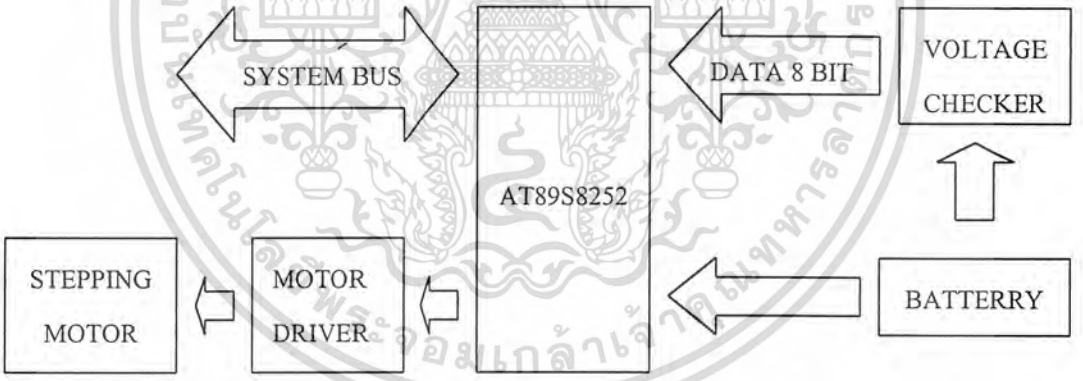
เมื่อเปิดเครื่อง โมดูลขับเคลื่อนจะตั้งค่าเริ่มต้นของให้กับตัวเอง โดยตั้งค่าของความเร็วไว้ที่ระดับ 2 และ ไม่มีการเคลื่อนที่ เมื่อมีสัญญาณ INT เป็น LO เข้ามายังส่วนควบคุมส่วนหน้า โมดูลก็จะตอบรับสัญญาณ ACK เป็น LO กลับไป โมดูลควบคุมหลักจะส่งข้อมูลลงในบัส I2C ส่วนรับข้อมูล I2C จะแปลงข้อมูลให้เป็นข้อมูลแบบขนาน แล้วส่งสัญญาณ B2M เป็น HI ไปให้ส่วนควบคุมส่วนหน้า ทำการถอดรหัสคำสั่ง แล้ว ตั้งค่าของแฟลกภายในตามคำสั่งที่รับมา พร้อมกับส่งสัญญาณควบคุมไปยังส่วนควบคุมส่วนหลัง เมื่อรับคำสั่งครบแล้ว โมดูลควบคุมหลักจะยกสัญญาณ INT ขึ้นเป็น HI ส่วนควบคุมส่วนหน้าก็จะยกสัญญาณ ACK เป็น HI ด้วย ในขณะที่ยังไม่มีคำสั่งเข้ามาจากโมดูลควบคุมหลัก โมดูลขับเคลื่อนจะทำงานตามแฟลกที่ตั้งไว้ รูปโมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์สร้างด้วยซีพีแอลดี แสดงในรูปที่ 6-25



รูปที่ 6-25 โมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์สร้างด้วยซีพีแอลดี

6.4.2.2 โมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์สร้างด้วย MCS-51

โครงสร้างการทำงานของโมดูลขับเคลื่อนโดยใช้สเต็ปปิ้งมอเตอร์แสดงไว้ดังรูปที่ 6-26



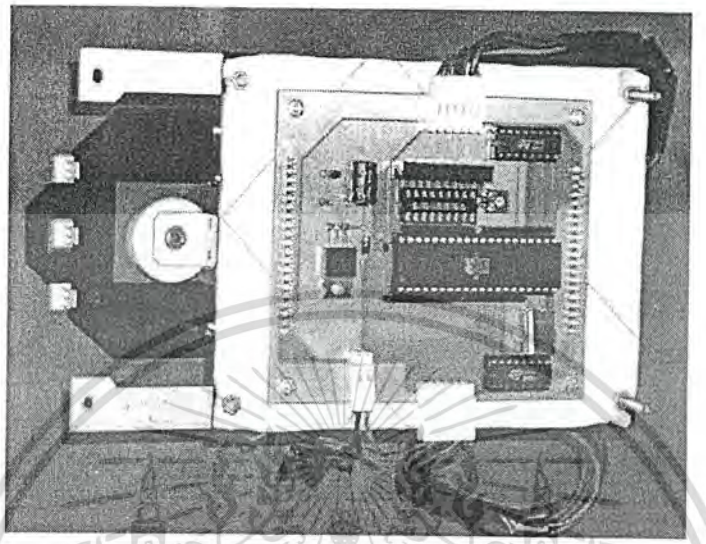
รูปที่ 6-26 โครงสร้างของโมดูลขับเคลื่อนโดยใช้สเต็ปปิ้งมอเตอร์สร้างด้วย MCS-51

การทำงานของโมดูลขับเคลื่อนโดยใช้สเต็ปปิ้งมอเตอร์แบ่งออกเป็น 3 ส่วนคือ

1. โค้ดหลักของโปรแกรมทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรที่เกี่ยวข้องกับการทำงานของโมดูล ตั้งเวลาการเกิดอินเตอร์รัปต์ให้กับอินเตอร์รัปต์ไทเมอร์ของไมโครคอนโทรลเลอร์ และกำหนดให้สามารถเกิดอินเตอร์รัปต์จากสัญญาณภายนอกได้
2. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากภายนอกเกิดขึ้นใช้สำหรับรับคำสั่งและรับส่งข้อมูลกับโมดูลควบคุมหลัก

3. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากไทเมอร์เกิดขึ้นใช้สำหรับสร้างพัลส์เพื่อขับขดลวดแต่ละเฟสภายในสเต็ปปิ้งมอเตอร์ให้หมุนตามคำสั่งที่ได้รับมาขณะนั้น

โมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์สร้างด้วย MCS-51 แสดงในรูปที่ 6-27



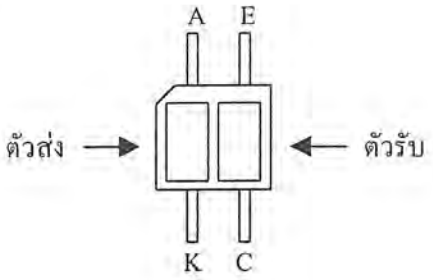
รูปที่ 6-27 โมดูลขับเคลื่อนโดยสเต็ปปิ้งมอเตอร์

6.5 โมดูลตรวจสอบระดับพื้นผิว

ไดโอดชนิดซิลิกอนโดยทั่วไปเมื่อได้รับการไบอัสกลับ จะเกิดกระแสรั่วไหลย้อนกลับผ่านไดโอด จะเกิดกระแสรั่วไหลย้อนกลับผ่านไดโอด ซึ่งกระแสรั่วไหลย้อนกลับและอิมพีแดนซ์ของรอยต่อพี-เอ็น นี้มีความไวต่อแสงมาก คือจะมีอิมพีแดนซ์สูงเมื่ออยู่ในที่มืด และมีอิมพีแดนซ์ต่ำเมื่ออยู่ในที่สว่าง

ไดโอดโดยทั่วไปจะหุ้มรอยต่อนี้ไว้ด้วยวัตถุทึบแสง แต่สำหรับโฟโต้ไดโอดเป็นไดโอดที่ทำขึ้นเพื่อให้เกิดปรากฏการณ์นี้โดยเฉพาะ ไดโอดชนิดนี้มีอยู่ 2 แบบ คือ ชนิดที่ตอบสนองต่อแสงที่มองเห็นได้ และชนิดที่ตอบสนองต่อแสงอินฟราเรด ซึ่งออปโต้คัปเปลอร์จะเป็นโฟโต้ไดโอดที่ตอบสนองต่อแสงอินฟราเรด

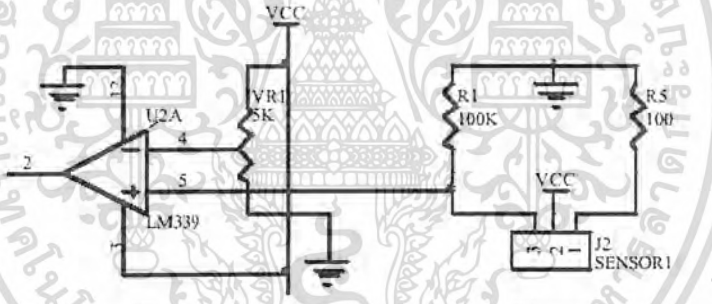
ออปโต้คัปเปลอร์ (OptoCoupler) เป็นอุปกรณ์ที่ประกอบด้วยแอลอีดี(LED) ซึ่งปกติเป็นชนิดอินฟราเรด และโฟโต้ทรานซิสเตอร์หรือโฟโต้ไดโอด ที่ถูกผลิตมาเป็นคู่กัน รวมอยู่ในตัวถังเดียวกันดังรูปที่ 6-28



รูปที่ 6-28 โครงสร้างทางกายภาพของออปโต้คัปเปลอร์

ดังนั้นเมื่อทำการจ่ายกระแสให้กับแอลอีดีส่งแสงอินฟราเรดออกมา หากมีวัตถุที่สามารถสะท้อนแสงอินฟราเรดกลับเข้าสู่โฟโต้ทรานซิสเตอร์ได้ จะทำให้ความต้านทานของโฟโต้ทรานซิสเตอร์ต่ำลง

การแปลงระดับแรงดันที่ตกคร่อมโฟโต้ทรานซิสเตอร์เข้าสู่ไมโครคอนโทรลเลอร์สามารถทำได้โดยการใช้ออปแอมป์เพื่อเปรียบเทียบแรงดันที่เหลือจากการตกคร่อมโฟโต้ทรานซิสเตอร์กับแรงดันอ้างอิงที่กำหนดขึ้น ดังรูปที่ 6-29

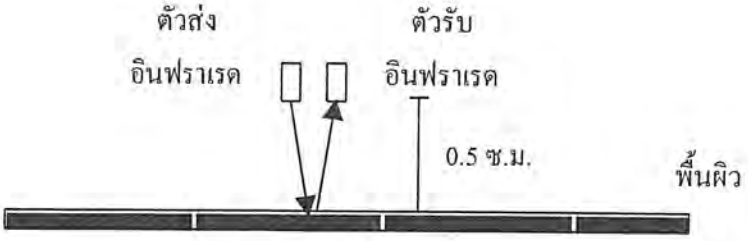


รูปที่ 6-29 วงจรเปรียบเทียบแรงดันที่ได้รับจากตัวรับอินฟราเรดกับแรงดันอ้างอิง

ซึ่งแรงดันอ้างอิงจะอยู่ระหว่างแรงดันที่เหลือจากการตกคร่อมโฟโต้ทรานซิสเตอร์เมื่อได้รับแสงกับแรงดันที่เหลือจากการตกคร่อมโฟโต้ทรานซิสเตอร์เมื่อไม่ได้รับแสง ดังนั้นเมื่อโฟโต้ทรานซิสเตอร์ได้รับแสงอินฟราเรดทำให้แรงดันที่เหลือจากการตกคร่อมโฟโต้ทรานซิสเตอร์มีค่ามากกว่าแรงดันอ้างอิง จะทำให้ผลของการเปรียบเทียบแรงดันของออปแอมป์ได้เป็น 5 โวลต์ หรือ ลอจิกสูง ในทางกลับกันเมื่อโฟโต้ทรานซิสเตอร์ไม่ได้รับแสงอินฟราเรดจะทำให้แรงดันที่เหลือจากการตกคร่อมโฟโต้ทรานซิสเตอร์มีค่าน้อยกว่าแรงดันอ้างอิง เป็นผลให้การเปรียบเทียบแรงดันของออปแอมป์ได้เป็น 0 โวลต์ หรือ ลงจิกต่ำ

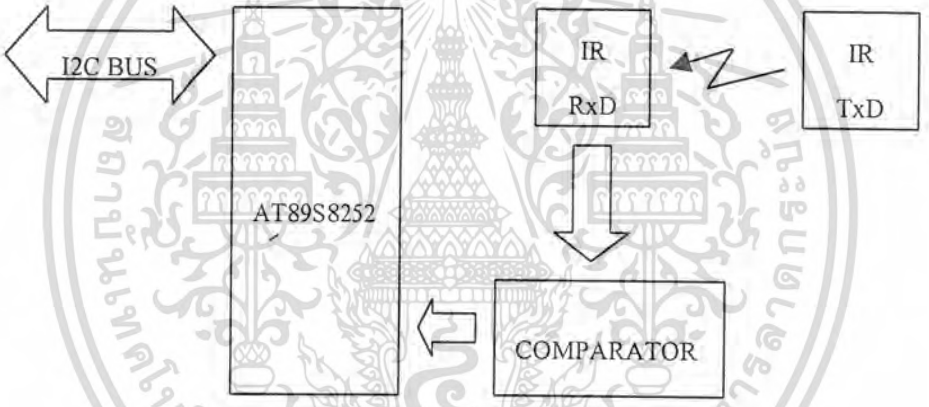
ในโมดูลตรวจสอบพื้นผิวนี้จะประกอบด้วยไอซีเปรียบเทียบแรงดัน จำนวน 3 ตัว แต่ละตัวภายในประกอบด้วยชุดเปรียบเทียบแรงดัน 4 ชุด ทำให้สามารถรองรับออปโต้คัปเปลอร์ได้ 12 ตัว

ในการติดตั้งออปโต้คัปเปลอร์เพื่อตรวจสอบระดับพื้นผิวจะสามารถติดตั้งให้สูงจากพื้นได้สูงสุดประมาณ 0.4 ซ.ม. บนพื้นสีขาว หรือน้อยกว่านี้ในกรณีที่เป็นพื้นสีอื่นที่มีความสามารถสะท้อนของแสงต่ำกว่านี้ และจะไม่มีการสะท้อนได้เลย บนพื้นผิวสีดำ ดังรูปที่ 6-30



รูปที่ 6-30 การติดตั้งออปโต้คัปเปลอร์เพื่อตรวจสอบระดับพื้นผิว

โครงสร้างการทำงานของโมดูลตรวจสอบระดับพื้นผิวแสดงไว้ดังรูปที่ 6-31



รูปที่ 6-31 โครงสร้างการทำงานของโมดูลตรวจสอบระดับพื้น

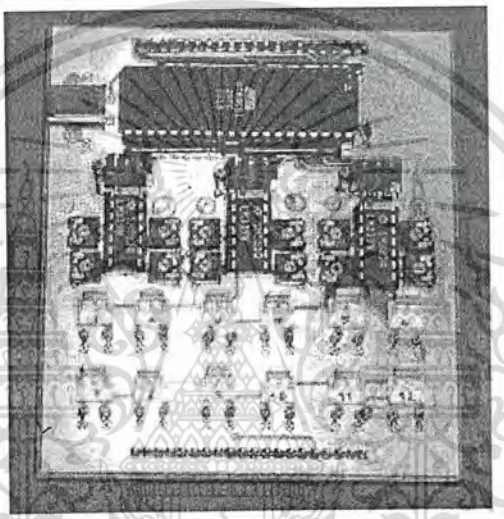
การทำงานของโมดูลตรวจสอบพื้นผิวแบ่งออกเป็น 2 ส่วนคือ

1. โค้ดหลักของโปรแกรมทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรที่เกี่ยวข้องกับการทำงานของโมดูล ตั้งเวลาการเกิดอินเตอร์รัปต์ให้กับอินเตอร์รัปต์ไทเมอร์ของไมโครคอนโทรลเลอร์ และกำหนดให้สามารถเกิดอินเตอร์รัปต์จากสัญญาณภายนอกได้ จากนั้นจะเป็นการวนตรวจสอบค่าสัญญาณตอบรับ ACK บนบัส และ ค่าจากการตรวจสอบพื้นผิวเพื่อทำการส่งค่ากลับไปยังโมดูลควบคุมหลักเมื่อมีการสั่งให้สามารถส่งข้อมูลกลับได้
2. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากภายนอกเกิดขึ้นใช้สำหรับรับคำสั่งและรับส่งข้อมูลกับโมดูลควบคุมหลัก

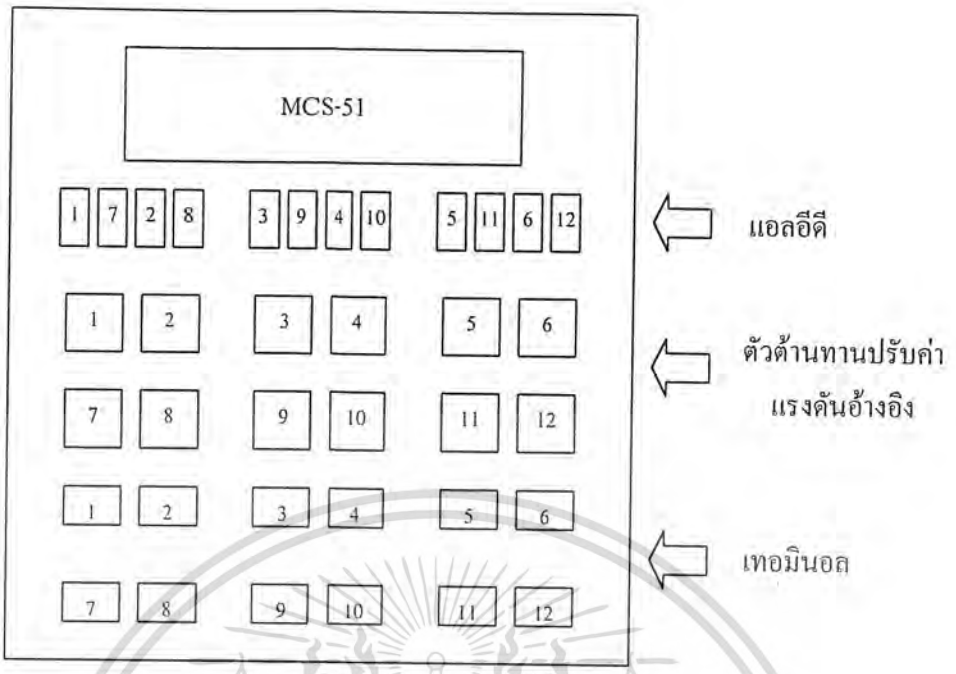
ชุดคำสั่งที่ใช้ควบคุมการทำงานของโมดูลตรวจสอบระดับพื้นผิวมีดังนี้

1. คำสั่งสำหรับอ่านค่าการตรวจสอบของอินฟราเรดแต่ละชุด
2. คำสั่งสำหรับอ่านค่าการตรวจสอบของอินฟราเรดทุกชุดพร้อมกัน
3. คำสั่งสำหรับขอมให้มีการส่งข้อมูลกลับเมื่อมีการเปลี่ยนแปลงข้อมูล
4. คำสั่งสำหรับยกเลิกการยอมให้มีการส่งข้อมูลกลับเมื่อมีการเปลี่ยนแปลงข้อมูล

โมดูลตรวจสอบระดับพื้นผิวแสดงไว้ดังรูปที่ 6-32 การปรับแต่งแรงดันอ้างอิงของชุดออปโต้คัปเปลอร์แสดงไว้ในรูปที่ 6-33 แอลอีดีจะของแต่ละตำแหน่งจะแสดงผลการเปรียบเทียบ ถ้าแอลอีดีสว่างแสดงว่าไม่มีการสะท้อนแสงกลับเข้าตัวออปโต้คัปเปลอร์ ถ้าแอลอีดีดับแสดงว่ามีการสะท้อนแสงกลับเข้าตัวออปโต้คัปเปลอร์



รูปที่ 6-32 โมดูลตรวจสอบพื้นผิว



รูปที่ 6-33 แสดงตำแหน่งของเทอมินอล ตัวต้านทานปรับค่าแรงดันอ้างอิง และแอลอีดีที่สัมพันธ์กัน

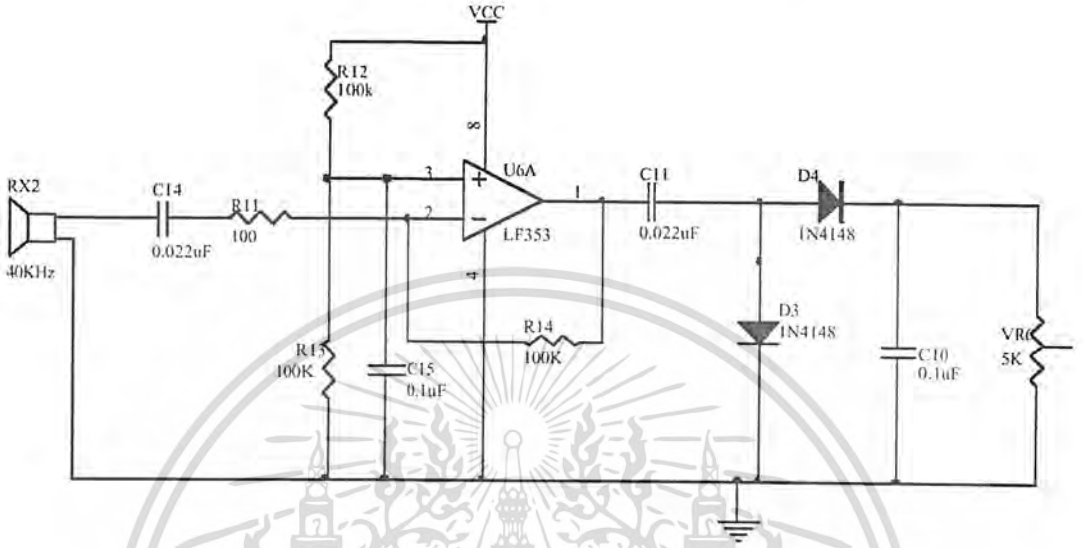
6.6 โมดูลตรวจสอบสิ่งกีดขวาง

คลื่นเสียงโดยทั่วไปมีการเคลื่อนที่แบบกระจายรอบทิศทาง เมื่อทำการเพิ่มความถี่ให้สูงขึ้นจะทำให้คลื่นเสียงเคลื่อนที่อย่างมีทิศทาง ซึ่งที่ความถี่ 40 กิโลเฮริทซ์ จะทำให้เสียงเดินทางเป็นเส้นตรงมากที่สุด จากทิศทางที่ส่งคลื่นโดยที่คลื่นเสียงจะแผ่ออกไปในลักษณะทรงกรวยที่มีมุมกว้างประมาณ 30 องศา จากแนวที่ส่งคลื่นเสียงออกไป ดังรูปที่ 6-34 ซึ่งความถี่เสียงระดับนี้ เรียกว่า อัลตราโซนิค (ultrasonic)

รูปที่ 6-34 แสดงลักษณะการเดินทางของคลื่นเสียงที่ความถี่ต่างกัน

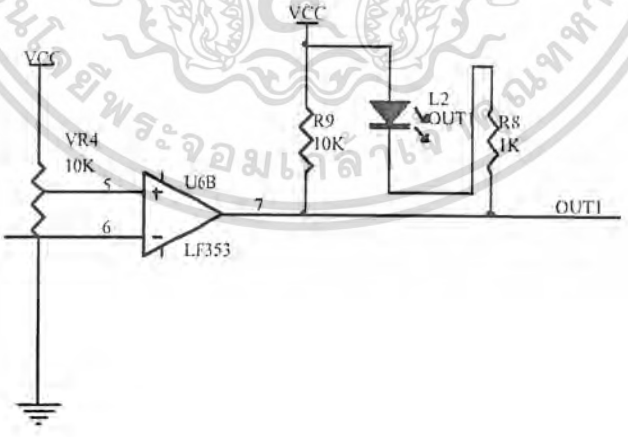
ธรรมชาติของคลื่นเมื่อเคลื่อนที่กระทบสิ่งกีดขวางจะสะท้อนกลับ และเมื่อคลื่นอัลตราโซนิคกระทบสิ่งกีดขวางแล้วสะท้อนกลับเข้าตัวรับคลื่นอัลตราโซนิคจะถูกเปลี่ยนเป็นแรงดันไฟฟ้ากระแสสลับ

ปริมาณน้อย ๆ ซึ่งแรงดันไฟฟ้าที่ได้จะขึ้นอยู่กับกำลังของคลื่นอัลตราโซนิคที่รับมาได้ ดังนั้นเราจำเป็นต้องแปลงแรงดันไฟฟ้าที่ได้จากตัวรับอัลตราโซนิคเป็นระดับแรงดันลอจิกให้กับไมโครคอนโทรลเลอร์ได้ซึ่งแบ่งวงจรออกเป็น 2 ส่วนคือส่วนแรกทำหน้าที่ขยายแรงดันไฟฟ้ากระแสสลับให้สูงขึ้นและตัดแรงดันซีกลบออกไปให้กลายเป็นไฟตรงเพื่อส่งไปยังส่วนที่สอง ดังรูปที่ 6-35



รูปที่ 6-35 วงจรขยายสัญญาณและกรองไฟกระแสสลับ

การทำงานของส่วนที่สองจะทำการเปรียบเทียบแรงดันที่ได้จากส่วนที่หนึ่งกับแรงดันอ้างอิงซึ่งอยู่ระหว่างแรงดันที่ได้จากส่วนที่หนึ่งเมื่อไม่ได้รับคลื่นอัลตราโซนิคกับแรงดันสูงสุดเมื่อได้รับคลื่นอัลตราโซนิค ดังรูปที่ 6-36

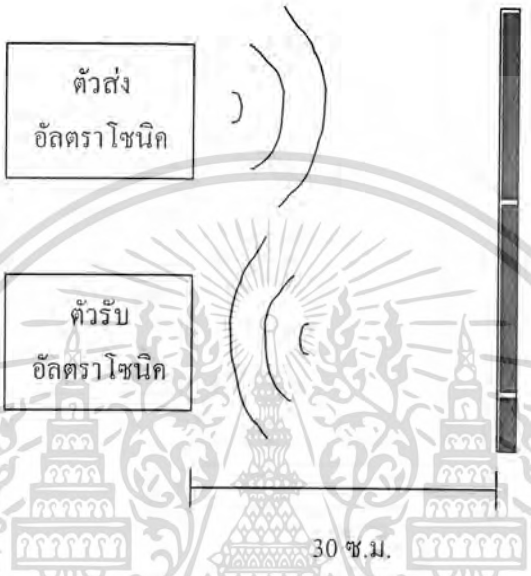


รูปที่ 6-36 วงจรเปรียบเทียบแรงดันที่ได้รับจากชุดขยายสัญญาณ

ในส่วนเปรียบเทียบแรงดันนี้ เมื่อแรงดันที่ได้รับจากส่วนขยายสัญญาณมีค่าสูงกว่าแรงดันอ้างอิงจะทำให้ออปแอมป์ที่เปรียบเทียบแรงดันให้แรงดันออกมาเป็น 0 โวลต์ หรือ ลอจิกต่ำ และในทางกลับกัน

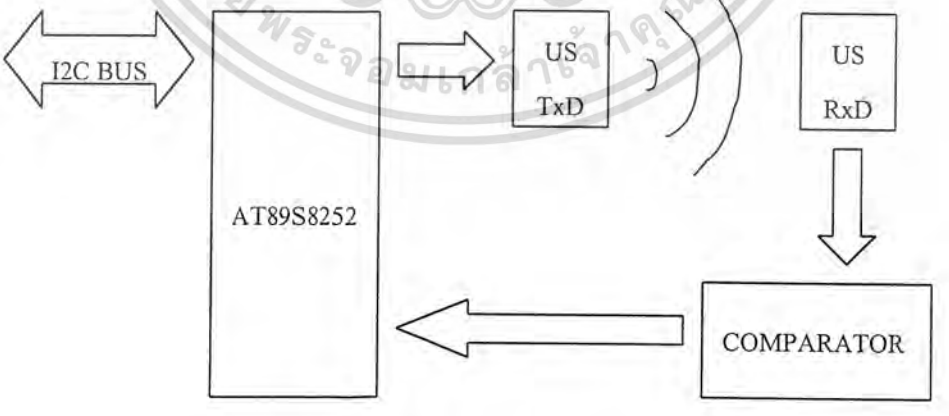
หากแรงดันที่ได้รับจากส่วนขยายสัญญาณมีค่าน้อยกว่าแรงดันอ้างอิงจะมีผลทำให้ออปแอมป์ที่เปรียบเทียบแรงดันให้ค่าแรงดันออกมาเป็น 5 โวลต์ หรือ ลอจิกสูง

โมดูลตรวจสอบสิ่งกีดขวางประกอบด้วยชุดอัลตราโซนิค 2 ชุด ใช้ตรวจสิ่งกีดขวางในระยะประมาณ 30 ซ.ม. ดังรูปที่ 6-37 โดยใช้ตัวรับส่งอัลตราโซนิคส่งคลื่นอัลตราโซนิคออกไป หากมีการสะท้อนกลับของคลื่นอัลตราโซนิคเข้าหาตัวรับคลื่นอัลตราโซนิคจะมีการส่งค่าลอจิก 0 ให้กับไมโครคอนโทรลเลอร์



รูปที่ 6-37 การติดตั้งตัวรับส่งอัลตราโซนิคเพื่อตรวจสอบสิ่งกีดขวาง

โครงสร้างการทำงานของโมดูลตรวจสอบสิ่งกีดขวางแสดงไว้ดังรูปที่ 6-38



รูปที่ 6-38 โครงสร้างการทำงานของโมดูลตรวจสอบสิ่งกีดขวาง

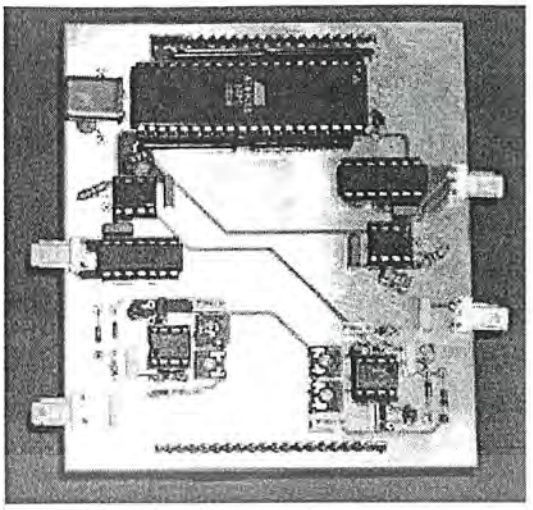
การทำงานของโมดูลตรวจสอบสิ่งกีดขวางแบ่งออกเป็น 2 ส่วนคือ

1. โค้ดหลักของโปรแกรมทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรที่เกี่ยวข้องกับการทำงานของโมดูล และกำหนดให้สามารถเกิดอินเตอร์รัปต์จากสัญญาณภายนอกได้ จากนั้นจะเป็นการส่งสัญญาณพัลส์ขนาดเล็ก ๆ เพื่อส่งคลื่นอัลตราโซนิคออกไป พร้อมกับเริ่มนับเวลา หากสามารถรับคลื่นอัลตราโซนิคได้ภายในช่วงเวลาที่ตั้งไว้ จะทำการเก็บค่าเวลาที่คลื่นสามารถเคลื่อนที่ไปกระทบวัตถุและสะท้อนกลับมาได้ หากการนับเวลาเกินที่กำหนดไว้จะถือว่าไม่มีวัตถุกีดขวางทิศทางที่ตรวจสอบ จากนั้นจะเป็นการวนตรวจสอบค่าสัญญาณตอบรับ ACK บนบัส และ ค่าเวลาจากการตรวจสอบเมื่อพบสิ่งกีดขวางเพื่อทำการส่งค่ากลับไปยังโมดูลควบคุมหลักเมื่อมีการสั่งให้สามารถส่งข้อมูลกลับได้
2. โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากภายนอกเกิดขึ้นใช้สำหรับรับคำสั่งและรับส่งข้อมูลกับโมดูลควบคุมหลัก

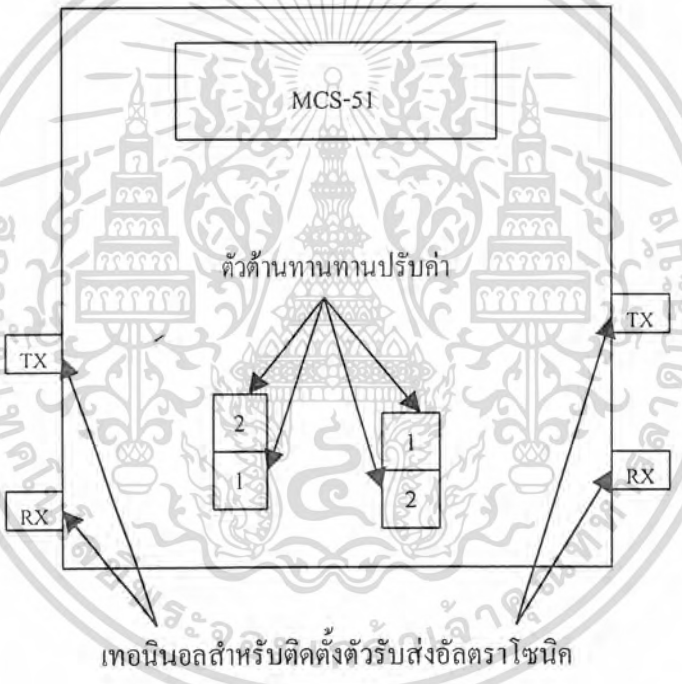
ชุดคำสั่งที่ใช้ควบคุมการทำงานของโมดูลตรวจสอบสิ่งกีดขวางมีดังนี้

1. คำสั่งสำหรับอ่านค่าระยะทางของอัลตราโซนิคแต่ละชุด
2. คำสั่งสำหรับอ่านค่าการตรวจสอบของอัลตราโซนิคทุกชุดพร้อมกัน
3. คำสั่งสำหรับยอมให้มีการส่งข้อมูลกลับเมื่อมีการเปลี่ยนแปลงข้อมูล
4. คำสั่งสำหรับยกเลิกการยอมให้มีการส่งข้อมูลกลับเมื่อมีการเปลี่ยนแปลงข้อมูล

โมดูลตรวจสอบสิ่งกีดขวางแสดงไว้ดังรูปที่ 6-39 การปรับแต่งมี 2 จุด อยู่ที่ ตัวต้านทานปรับค่าได้จำนวน 2 ตัว ที่วงจรส่วนรับคลื่นอัลตราโซนิค ตัวต้านทานตัวแรก จะทำหน้าที่ ปรับค่าแรงดันสูงสุดที่วงจรขยายสัญญาณส่งมา โดยจะปรับให้แรงดันที่ขาแท็บเมื่อเทียบกับกราวด์มีค่าประมาณ 2.5 ถึง 3 โวลต์ ตัวต้านทานตัวที่สอง จะทำหน้าที่ ปรับค่าแรงดันอ้างอิงเพื่อเปรียบเทียบกับแรงดันที่ได้จากวงจรขยาย โดยปรับค่าให้อยู่ระหว่างแรงดันที่ได้จากวงจรขยายเมื่อได้รับคลื่นอัลตราโซนิคกับค่าแรงดันเมื่อไม่ได้รับคลื่นอัลตราโซนิค ตำแหน่งของตัวต้านทานปรับค่าแสดงไว้ดังรูปที่ 6-40



รูปที่ 6-39 โมดูลตรวจสอบสิ่งกีดขวาง



รูปที่ 6-40 ตำแหน่งในการติดตั้งตัวรับส่งอัลตราโซนิคและตำแหน่งของตัวต้านทานปรับค่า

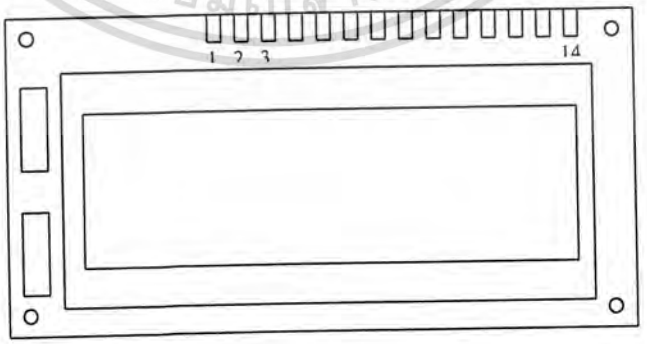
6.7 โมดูลติดต่อกับผู้ใช้

โมดูลแสดงผลแบบผลึกเหลวหรือแอลซีดี (LCD) เป็นโมดูลแสดงผลที่ใช้ลักษณะการปิดและเปิดตัวเองกับแสงซึ่งอยู่ในตัวกระจกบรรจุผลึก ภายในบรรจุไอซีระดับ LSI ใช้ทำหน้าที่ควบคุมการแสดงผลของโมดูลโดยทำการรับคำสั่งจากขาข้อมูลที่ต่อกับภายนอก แต่ละบริษัทอาจจะผลิตโมดูลแอลซีดีที่มีขนาดจำนวนตัวอักษรหรือจำนวนบรรทัดแตกต่างกันแต่ก็มีหลักการทำงานแบบเดียวกันทั้งหมด ตัวอย่างโมดูลแสดงผลแบบผลึกเหลวแสดงไว้ดังรูปที่ 6-41

รูปที่ 6-41 ตัวอย่างโมดูลแอลซีดีขนาด 16 ตัวอักษร 2 บรรทัด

การจัดขาสัญญาณโดยทั่วไปของโมดูลแอลซีดีแสดงไว้ดังรูปที่ 6-42 ซึ่งรายละเอียดการทำงานของแต่ละขามีดังนี้

ขา 1	GND	กราวด์
ขา 2	VCC	ไฟเลี้ยง 5 โวลต์
ขา 3	VO	ไฟเลี้ยงเพื่อปรับความเข้มของการแสดงผล
ขา 4	RS	แยกประเภทข้อมูลบนขาข้อมูลว่าเป็นคำสั่ง = 0 หรือ ข้อมูล = 1
ขา 5	R/W	เลือกว่าจะทำการอ่านข้อมูล = 1 หรือ เขียนข้อมูล = 0 กับโมดูลแอลซีดี
ขา 6	EN	ขาอีนาเบิลการทำงานของโมดูลแอลซีดี ทำงานที่ขอบสัญญาณขาลง
ขา 7	DB0	ขาข้อมูลบิตที่ 0
ขา 8	DB1	ขาข้อมูลบิตที่ 1
ขา 9	DB2	ขาข้อมูลบิตที่ 2
ขา 10	DB3	ขาข้อมูลบิตที่ 3
ขา 11	DB4	ขาข้อมูลบิตที่ 4
ขา 12	DB5	ขาข้อมูลบิตที่ 5
ขา 13	DB6	ขาข้อมูลบิตที่ 6
ขา 14	DB7	ขาข้อมูลบิตที่ 7



รูปที่ 6-42 การจัดเรียงขาสัญญาณโดยทั่วไปของโมดูลแอลซีดี

ขั้นตอนการทำงานของโมดูลแอลซีดี

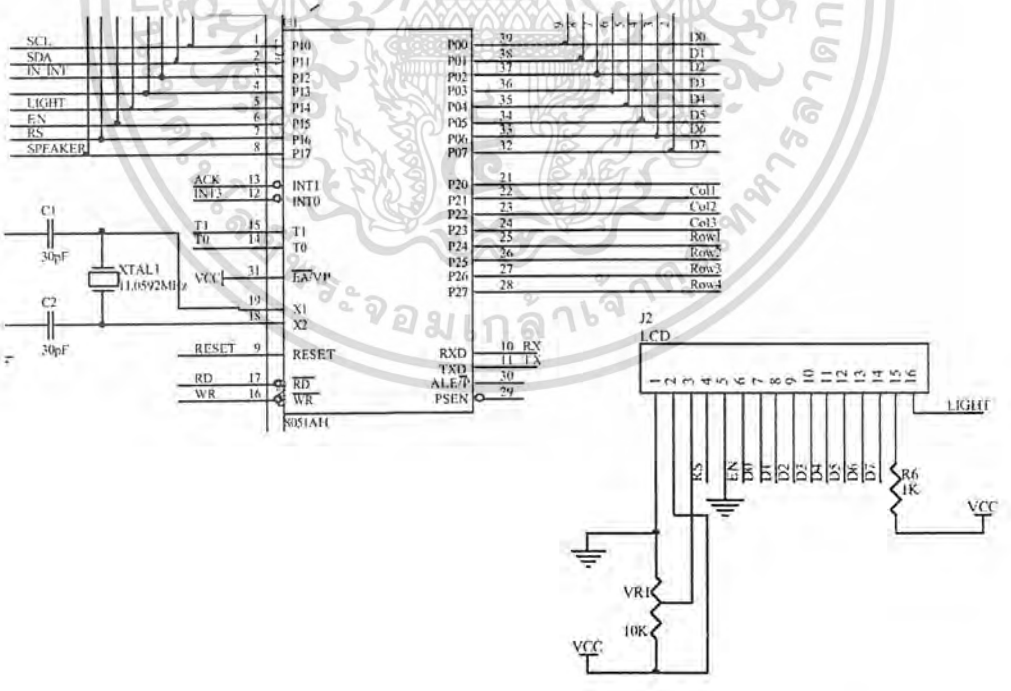
เมื่อเริ่มจ่ายไฟให้แก่โมดูลแอลซีดี จะใช้เวลาประมาณ 15 มิลลิวินาที เพื่อให้โมดูลแอลซีดีเตรียมความพร้อมก่อนจากนั้นจึงสามารถทำงานตามคำสั่งที่ส่งมายังโมดูลแอลซีดีได้

ในการติดต่อกับโมดูลแอลซีดีจะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายในโมดูลแอลซีดี แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือทำงานต่อไปได้ ซึ่งช่วงเวลาหน่วงเมื่อสั่งงานแต่ละคำสั่งจะใช้เวลาแตกต่างกันขึ้นอยู่กับความซับซ้อนของคำสั่งและการแสดงผลบนจอผลึกเหลว ขั้นตอนในการควบคุมการทำงานของจอแอลซีดีมีดังนี้

1. กำหนดว่าเป็นการอ่านข้อมูลจากจอแอลซีดีหรือเขียนข้อมูลไปยังจอแอลซี
2. กำหนดว่าข้อมูลบนขาข้อมูลเป็นคำสั่งหรือข้อมูลที่ต้องการแสดงผล
3. หากเป็นการเขียนข้อมูลจะวางข้อมูลขนาด 8 บิต ผ่านขาข้อมูลบิตที่ 0 ถึงบิตที่ 7
4. ทำการอีนาเบิลจอแอลซีดีโดยการเปลี่ยนระดับลอจิกที่ขา EN จากลอจิกสูงไปลอจิกต่ำ

ชุดคำสั่งที่ใช้สำหรับควบคุมการทำงานของจอแอลซีดีสามารถดูได้ในภาคผนวก ค

การเชื่อมต่อจอแอลซีดีเข้าภายในโมดูลติดต่อกับผู้ใช้จะนำขาข้อมูลของจอแอลซีดีต่อเข้ากับพอร์ต 0 ของไมโครคอนโทรลเลอร์ และขาควบคุมประกอบด้วย ขา EN ต่อเข้ากับพอร์ต 1 บิตที่ 5 ขา RS ต่อเข้ากับพอร์ต 1 บิตที่ 6 และขา R/W ต่อเข้ากับกราวด์ เพื่อทำการเขียนข้อมูลไปยังจอแอลซีดีตลอด ดังรูปที่ 6-43

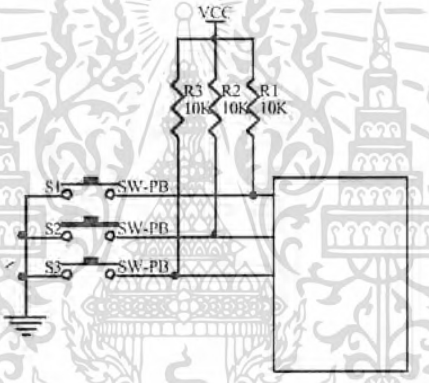


รูปที่ 6-43 การต่อขาสัญญาณเพื่อควบคุมกาทำงานของโมดูลแอลซีดี

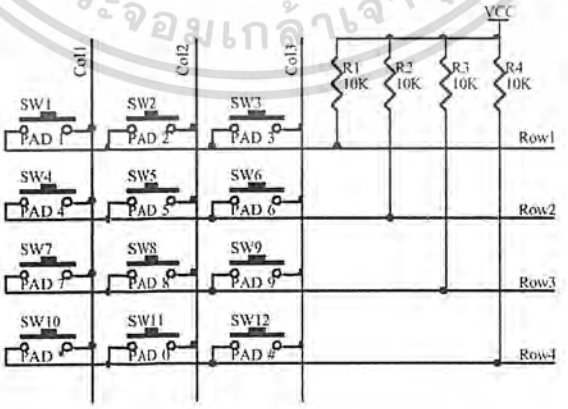
การรับข้อมูลจากผู้ใช้โดยการอ่านค่าหรือการรับค่าจากการกดสวิตช์ไปยังไมโครคอนโทรลเลอร์โดยทั่วไปสามารถทำได้ 2 แบบ คือ ต่อเข้าไฟเลี้ยงหรือกราวด์โดยตรงเมื่อมีการกดสวิตช์ตัวใดตัวหนึ่งดังรูปที่ 6-44

ข้อดีของการต่อสวิตช์ลักษณะนี้คือสามารถทำได้ง่ายและเขียนโปรแกรมเพื่อตรวจสอบได้ง่าย แต่เมื่อเราต้องการเพิ่มจำนวนสวิตช์มากขึ้นจะทำให้สิ้นเปลืองขาอินพุตของไมโครคอนโทรลเลอร์เพื่อทำการรับค่าจากสวิตช์แต่ละตัว

ดังนั้นจึงมีการออกแบบสวิตช์อีกแบบขึ้นมา ซึ่งเรียกว่า การต่อแบบเมตริกซ์ ดังรูปที่ 6-45 สวิตช์จะถูกต่อกันในแนวแกนตั้งและแกนนอน จะเรียกแนวตั้งว่า หลักหรือคอลัมน์ (column) และ แนวนอนว่า แถวหรือโรว์ (row) ดังนั้นค่าของสวิตช์จะต้องประกอบด้วยค่าตำแหน่งในแนวหลักและแถว ดังนั้นในการเขียนโปรแกรมจะมีความซับซ้อนเพิ่มมากขึ้น แต่จะประหยัดจำนวนสายสัญญาณที่ต่อเข้ากับไมโครคอนโทรลเลอร์ได้ ทั้งยังเพิ่มจำนวนสวิตช์ได้ง่ายโดยแก้ไขโปรแกรมเพียงเล็กน้อย โดยทั่วไปจะเรียกสวิตช์แบบเมตริกซ์นี้ว่า คีย์แพด (keypad)

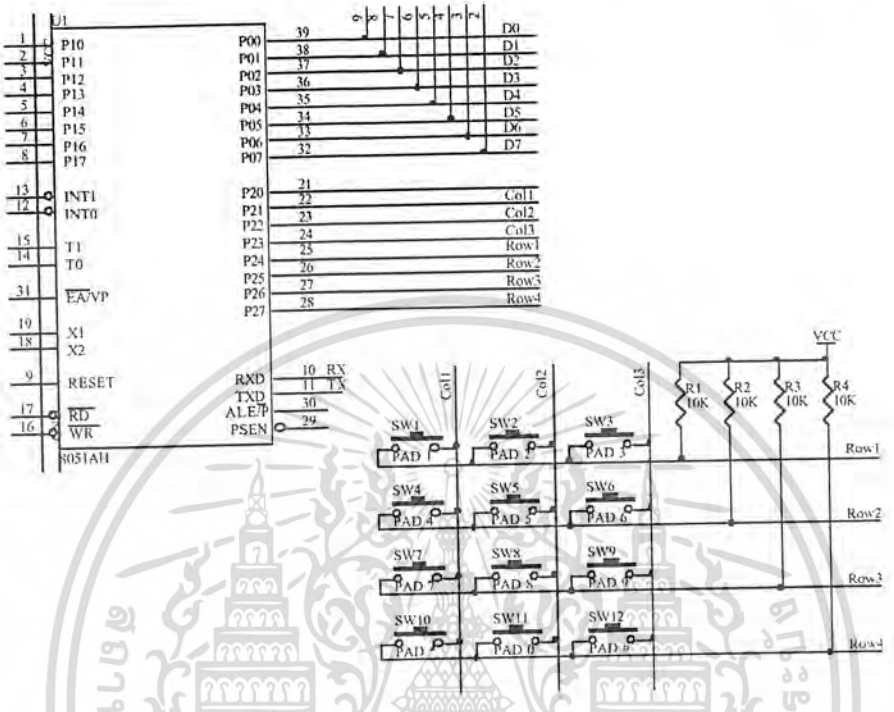


รูปที่ 6-44 การเชื่อมต่อสวิตช์เข้าโดยตรงกับขาไมโครคอนโทรลเลอร์



รูปที่ 6-45 การเชื่อมต่อสวิตช์แบบเมตริกซ์

การเชื่อมต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์ของโมดูลติดต่อกับผู้ใช้จะใช้ขา I/O ของพอร์ต 2 จำนวน 7 เส้น คือ บิตที่ 1 ถึง บิตที่ 3 สำหรับสายทางหลัก C0 C1 C2 และ บิตที่ 4 ถึง บิตที่ 7 สำหรับสายทางแถว R0 R1 R2 R3 ซึ่งค่าจากการกดจะถูกอ่านจากสายสัญญาณด้านแถว ดังนั้นจึงต้องมีการต่อตัวต้านทานพูลอัปเพื่อกำหนดสภาวะเริ่มต้นที่ขณะที่ไม่มีการกดคีย์ เมื่อมีการตรวจสอบการกด ดังรูปที่ 6-46



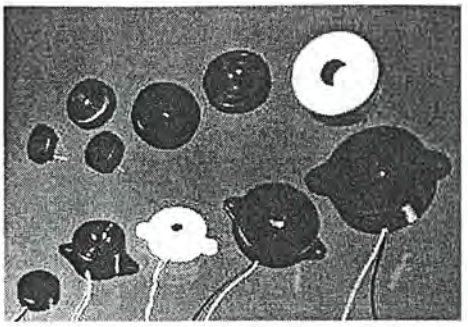
รูปที่ 6-46 การเชื่อมต่อสวิตช์แบบเมตริกซ์เข้ากับไมโครคอนโทรลเลอร์

ขั้นตอนในการอ่านค่าคีย์ที่ถูกกดมีดังนี้

1. ทำการส่งลอจิกต่ำไปยังขาสัญญาณด้านหลัก C0 C1 C2 ตามลำดับ
2. ในการส่งข้อมูลไปยังขาสัญญาณด้านหลักแต่ละครั้งจะมีการอ่านที่อยู่บนสายสัญญาณด้านแถวเข้ามา หากไม่มีการกดจะมีลอจิกด้านแถวจะเป็น 1 ทั้งหมด (1111)
3. ในกรณีที่มีการกดคีย์ใดคีย์หนึ่งค่าลอจิกที่อ่านได้จากด้านแถวจะไม่เป็น 1111จากนั้นจึงนำตำแหน่งของแถวที่อ่านได้ค่าลอจิกต่ำ และตำแหน่งของหลักที่มีการส่งลอจิกต่ำออกไป มาหาหมายเลขคีย์ที่มีการกดจริง แล้วจึงนำไปประมวลผลต่อไป

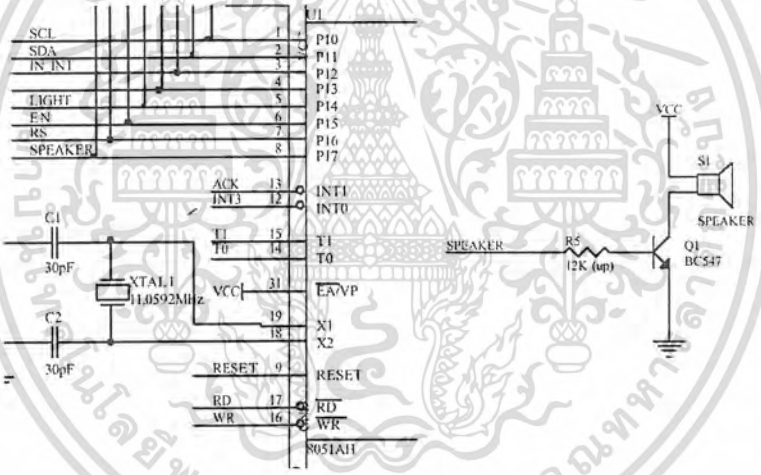
เช่นหากคีย์ "5" ถูกกด ค่าลอจิกด้านหลักที่ส่งออกไปขณะที่ตรวจสอบพบจะเป็น 101 และค่าลอจิกด้านแถวที่อ่านได้จะเป็น 1011 จากนั้นไมโครคอนโทรลเลอร์จะสรุปค่าที่ตรวจสอบได้ว่าเป็นคีย์ "5"

บัซเซอร์เป็นอุปกรณ์ที่กำเนิดเสียงตามความถี่ของสัญญาณที่ได้รับ มีขา 2 ขา ประกอบด้วยขาไฟเลี้ยงและขากราวด์ ในการกำหนดความถี่ของเสียงจะต้องทำการจ่ายแรงดันให้มีลักษณะเป็นลูกคลื่นตามค่าความถี่ที่ต้องกร ตัวอย่างบัซเซอร์แสดงดังรูปที่ 6-47



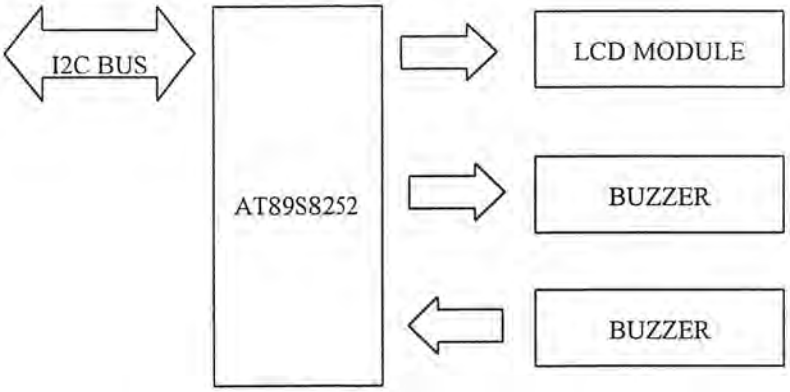
รูปที่ 6-47 ตัวอย่างบัซเซอร์ที่ใช้งานโดยทั่วไป

ในการใช้งานจะทำการต่อขาไฟเลี้ยงของบัซเซอร์เข้ากับไฟเลี้ยง และขากราวด์เข้ากับขา C ของทรานซิสเตอร์ชนิด NPN ซึ่งทำหน้าที่เป็นสวิตช์เปิดปิดการทำงานของบัซเซอร์ โดยที่ ขา B ของทรานซิสเตอร์ต่อเข้ากับบิตที่ 7 พอร์ต 1 ของไมโครคอนโทรลเลอร์ และ ขา E ของทรานซิสเตอร์ต่อลงกราวด์ เพื่อกระแสไหลผ่านบัซเซอร์ ทรานซิสเตอร์ และลงกราวด์ได้ครบวงจร ดังรูปที่ 6-48



รูปที่ 6-48 การต่อวงจรขยายสัญญาณเข้ากับไมโครคอนโทรลเลอร์

โครงสร้างการทำงานของโมดูลติดต่อกับผู้ใช้แสดงไว้ดังรูปที่ 6-49



รูปที่ 6-49 โครงสร้างการทำงานของโมดูลติดต่อกับผู้ใช้

การทำงานของโมดูลติดต่อกับผู้ใช้แบ่งออกเป็น 3 ส่วนคือ

1 โค้ดหลักของโปรแกรมทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรที่เกี่ยวข้องกับการทำงานของโมดูล และกำหนดให้สามารถเกิดอินเตอร์รัปต์จากสัญญาณภายนอกได้

2 โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากภายนอกเกิดขึ้นใช้สำหรับรับคำสั่งและรับส่งข้อมูลกับโมดูลควบคุมหลัก

3 โค้ดการทำงานเมื่อมีอินเตอร์รัปต์จากไทเมอร์เกิดขึ้นใช้สำหรับทำการสร้างความถี่ให้กับบัซเซอร์เพื่อกำเนิดเสียงตามความถี่ที่ต้องการ

ชุดคำสั่งที่ใช้ควบคุมการทำงานของโมดูลติดต่อกับผู้ใช้มีดังนี้

ส่วนแสดงผลผ่านทางจอแสดงผลแบบผลึกเหลว

1. แสดงผลตัวอักษรขนาด 1 ตัวอักษร
2. แสดงผลข้อความ
3. แสงดผลตัวเลขขนาด 8 บิต
4. แสดงผลตัวเลขขนาด 16 บิต
5. ย้ายตำแหน่งเคอร์เซอร์บนจอแสงผลแบบผลึกเหลว
6. ลบตัวอักษรบนหน้าจอแสดงผลแบบผลึกเหลว

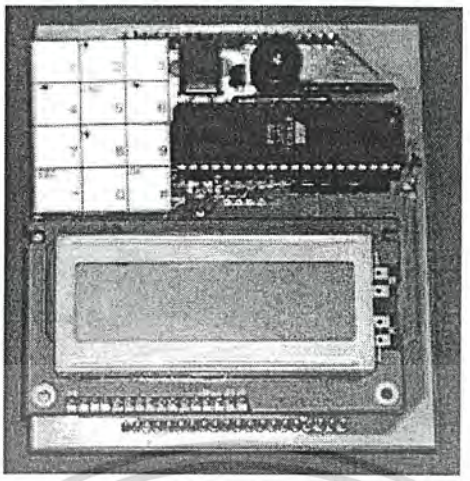
ส่วนรับข้อมูลจากคีย์แพด

1. รอรับค่าการกดคีย์
2. ตรวจสอบการกดคีย์โดยไม่รอการถูกกด

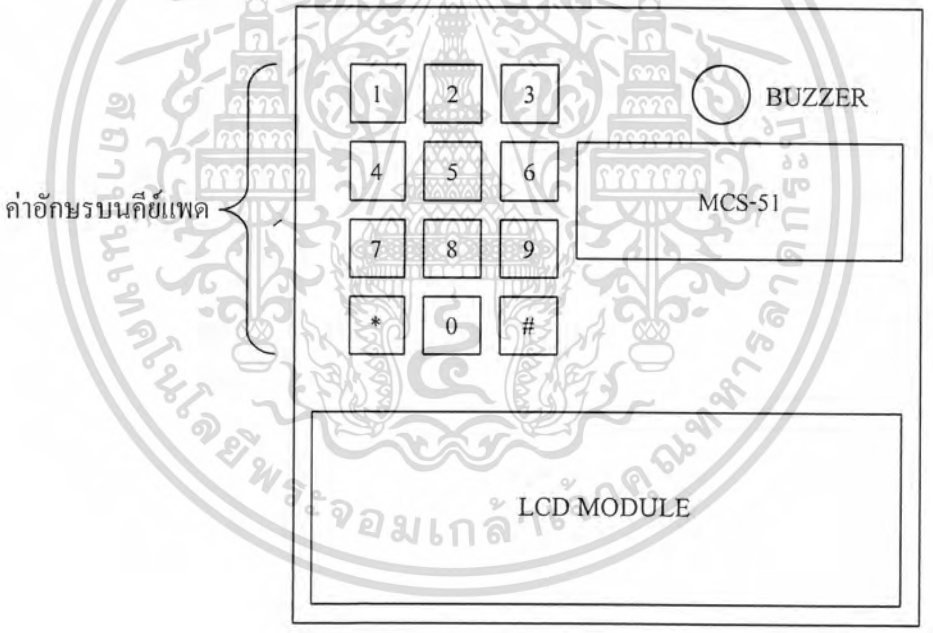
ส่วนกำเนิดเสียง

1. สร้างเสียงตามความถี่ที่กำหนด

โมดูลติดต่อกับผู้ใช้แสดงไว้ดังรูป 6-50 โดยที่อักษรประจำคีย์แพดแสดงไว้ดังรูปที่ 6-51



รูปที่ 6-50 โมดูลติดต่อกับผู้ใช้



รูปที่ 6-51 ค่าอักษรบนคีย์แพด

บทที่ 7
การทดสอบ

7.1 วัตถุประสงค์

ในการทดสอบโครงงานชุดพัฒนาหุ่นยนต์ขนาดเล็กนั้น จะแบ่งการทดสอบออกเป็นสองส่วน คือ การทดสอบแบบแยกโมดูล และ การทดสอบการนำไปสร้างเป็นหุ่นยนต์ โดยการทดสอบแต่ละแบบมีวัตถุประสงค์ดังนี้

7.1.1 การทดสอบแบบแยกโมดูล

เพื่อทดสอบการทำงานของแต่ละโมดูลทุกคำสั่ง ว่าสามารถทำงานได้อย่างถูกต้อง ตรงตามที่ต้องการหรือไม่ ทั้งโมดูลที่สร้างด้วย MCS-51 และ โมดูลที่สร้างด้วยซีพีแอลดี

7.1.2 การทดสอบการนำไปสร้างเป็นหุ่นยนต์

1. ทดสอบความสามารถในการนำชุดพัฒนาไปสร้างเป็นหุ่นยนต์ โดยการประกอบหุ่นยนต์ที่ใช้โมดูลต่างกัน
2. ทดสอบความสามารถในการเข้ากันได้ของโมดูลที่ทำหน้าที่คล้ายกัน เช่น โมดูลขับเคลื่อนทั้ง 3 แบบ

7.2 ขั้นตอนการทดสอบ

ในการทดสอบนั้น จะต้องเริ่มจากการทดสอบแบบแยกโมดูลเสียก่อน เพื่อให้แน่ใจว่าแต่ละโมดูลทำงานได้อย่างถูกต้อง จากนั้นจึงทดสอบโดนการสร้างเป็นหุ่นยนต์ต่อไป ลำดับในการทดสอบมีดังต่อไปนี้

1. ทดสอบการส่งข้อมูลผ่านบัส I2C จากโมดูลควบคุมหลักไปยังโมดูล I/O ต่าง ๆ
2. ทดสอบการรับข้อมูลผ่านบัส I2C จากโมดูล I/O ต่างๆ เข้าสู่โมดูลควบคุมหลัก
3. ทดสอบการทำงานของโมดูล I/O ทุกโมดูลโดยสั่งงานจากโมดูลควบคุมหลัก
4. ทดสอบการทำงานรวมของโมดูลเมื่อนำมาสร้างเป็นหุ่นยนต์ ซึ่งในที่นี้จะสร้างเป็นหุ่นยนต์ที่สามารถโปรแกรมได้ และ เดินหลบหลีกสิ่งกีดขวางได้ จากนั้นทดลองเปลี่ยนโมดูลขับเคลื่อนหลายๆแบบ

หุ่นยนต์ที่สามารถโปรแกรมได้

เป็นหุ่นยนต์ที่ผู้ใช้สามารถโปรแกรมลักษณะการเดินได้ สามารถหลบหลีกสิ่งกีดขวาง และ วิ่งโดยไม่ตกจากโต๊ะได้ โมดูลที่ใช้ในหุ่นยนต์ตัวนี้ ได้แก่ โมดูลติดต่อกับผู้ใช้ , โมดูลควบคุมหลัก , โมดูลอินฟราเรดใช้ในการตรวจสอบพื้นโต๊ะ , โมดูลอัลตราโซนิคสำหรับตรวจจับสิ่งกีดขวางที่อยู่ข้างหน้า และ โมดูลขับเคลื่อน เมื่อเริ่มต้นการทำงาน หุ่นยนต์จะแสดงหน้าจอรอรับ

การโปรแกรมของผู้ใช้ ผู้ใช้สามารถใส่รูปแบบการเดินของหุ่นยนต์ให้กับหุ่นยนต์ผ่านทางคีย์ที่อยู่บนโมดูลติดต่อกับผู้ใช้ ทั้งทิศทางการเคลื่อนที่ และ เวลา เมื่อป้อน ข้อมูลเรียบร้อยแล้ว หุ่นยนต์จะเดินตามรูปแบบที่โปรแกรมไว้ หากพบสิ่งกีดขวางที่อยู่ข้างหน้า หรือ พบว่าหุ่นกำลังจะตกจากโต๊ะ หุ่นยนต์สามารถเปลี่ยนทิศทางการเคลื่อนที่ได้ เมื่อเดินครบแล้ว จะหยุด

หุ่นยนต์เดินตามเส้น

เป็นหุ่นยนต์ที่เดินอยู่บนโต๊ะสีขาว ซึ่งมีเส้นสีดำเป็นทางเดิน โมดูลที่ใช้ในหุ่นยนต์ตัวนี้ได้แก่ โมดูลติดต่อกับผู้ใช้ , โมดูลควบคุมหลัก , โมดูลอินฟราเรดสำหรับการตรวจสอบเส้นทางการเดิน และ โมดูลขับเคลื่อน เมื่อเริ่มต้นการทำงาน หุ่นยนต์จะเคลื่อนที่ไปตามเส้นสีดำที่อยู่บนสนาม

7.3 ผลการทดสอบ

การทดสอบการทำงานของตัวหุ่นแบ่งเป็นการทดสอบ 2 ส่วนคือ การทดสอบการรับส่งข้อมูลบนบัส I2C เนื่องจากเป็นส่วนพื้นฐานในการรับส่งคำสั่งและข้อมูลระหว่างโมดูลควบคุมหลักกับโมดูลอื่น ๆ และ การทดสอบการทำงานของโมดูลต่าง ๆ

7.3.1 การทดสอบการรับส่งข้อมูล I2C

ผลการทดสอบการรับส่งข้อมูล I2C ของโมดูลควบคุมหลัก,โมดูลอินพุต และ เอาต์พุต พบว่าทำงานได้อย่างถูกต้อง โดยมีความเร็วในการรับ/ส่งข้อมูลอยู่ที่ประมาณ 70 kbit/s

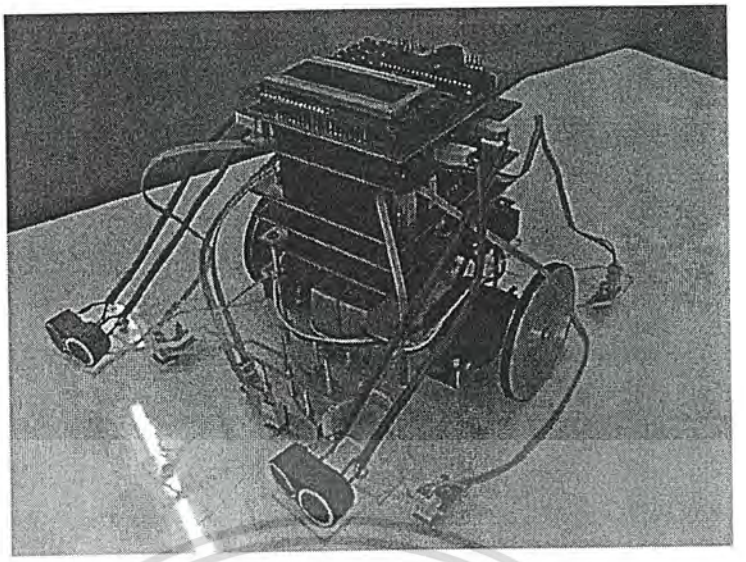
การรับข้อมูลของโมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์ที่สร้างด้วยซีพีแอลดีนั้น ไม่เสถียรเนื่องจากปัญหาด้านเสถียรภาพของตัวซีพีแอลดีเอง ส่วนการรับข้อมูลของโมดูลอื่นๆที่สร้างจาก MCS-51 นั้นมีเสถียรภาพดี

7.3.2 การทดสอบการทำงานของโมดูลอินพุต และ เอาต์พุต

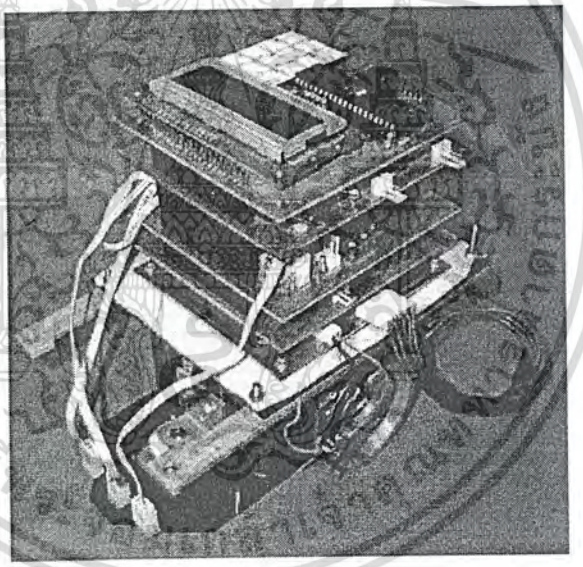
ผลการทดสอบการทำงานของโมดูลทุกโมดูลพบว่าทำงานได้อย่างถูกต้อง สามารถรองรับทุกคำสั่งที่แสดงในตารางข้างล่าง ส่วนการทำงานของโมดูลขับเคลื่อนด้วยสเตปปิ้งมอเตอร์ที่สร้างด้วยซีพีแอลดีสามารถรองรับทุกคำสั่งได้เหมือนกับโมดูลที่สร้างด้วย MCS-51 แต่มีปัญหาเรื่องความถูกต้องในการทำงาน

7.3.3 ผลการทดสอบการทำงานเมื่อสร้างเป็นหุ่นยนต์

การทดสอบการทำงานของหุ่นยนต์ที่สร้างขึ้นมาจากชุดพัฒนานั้น พบว่าทำงานตามหน้าที่ได้อย่างถูกต้อง และ สามารถเปลี่ยนโมดูลขับเคลื่อนไปใช้โมดูลที่ขับเคลื่อนด้วยมอเตอร์แบบอื่นได้ โดยที่ไม่จำเป็นต้องแก้ไขโปรแกรมภายใน



รูปที่ 7-1 การสร้างเป็นหุ่นยนต์ที่สามารถโปรแกรมได้



รูปที่ 7-2 การสร้างเป็นหุ่นยนต์เดินตามเส้น

7.4 สรุปผลการทดสอบ

1. ผลการทดสอบการรับส่งข้อมูลบนบัส I2C สามารถทำงานได้อย่างถูกต้อง แต่ยังไม่สามารถทำงานที่ความเร็วสูงสุดได้ เนื่องจากมีการหน่วงเวลาในการโปรแกรมเขียนขึ้น และ เพื่อป้องกันความผิดพลาดในการรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ของแต่ละโมดูล
2. ผลการทดสอบการส่งข้อมูลบนบัส I2C จากโมดูลควบคุมหลักไปยังโมดูล I/O ต่างๆ สามารถทำงานได้ถูกต้อง โมดูลต่าง ๆ สามารถรับข้อมูลได้ถูกต้อง โดยไม่มีการผิดโมดูล

3. ผลการทดสอบการรับข้อมูลบนบัส I2C จากโมดูล I/O ต่างๆ เข้าสู่โมดูลควบคุมหลักเมื่อมีการส่งคำสั่งให้ส่งข้อมูลกลับมาสามารถทำงานได้ถูกต้อง แต่ละโมดูลสามารถส่งข้อมูลหลาย ๆ ไบต์กลับมาได้ ตามที่ได้กำหนดจำนวนไบต์ที่ใช้รับส่งไว้

4. ผลการทดสอบการทำงานของทุกโมดูลเมื่อประกอบเป็นหุ่นสามารถทำงานได้สมบูรณ์ หุ่นสามารถทำงานตามโปรแกรมที่ได้บันทึกอยู่ในอยู่โมดูลควบคุมหลักได้อย่างถูกต้อง การตอบสนองการทำงานเกิดขึ้นทันทีที่มีการตรวจสอบพบการเปลี่ยนแปลง เช่น หุ่นจะหมุนตัวหลบทันทีเมื่อตรวจพบว่าไม่มีพื้นด้านหน้าที่กำลังเคลื่อนที่ไปนั้น
5. การเปลี่ยนโมดูลขับเคลื่อนเพื่อทดสอบความสามารถในการทำงานแทนที่ของโมดูลที่ทำหน้าที่คล้ายกันสามารถทำงานได้ถูกต้อง เช่น การแทนที่โมดูลขับเคลื่อนด้วยเซอร์โว ด้วยโมดูลขับเคลื่อนด้วยสเต็ปปิ้งมอเตอร์

7.5 บทวิจารณ์

เนื่องจากชุดพัฒนาหุ่นยนต์ขนาดเล็กนี้เป็นต้นแบบที่สร้างขึ้นจากแนวคิดในการแบ่งการทำงานส่วนต่างๆ ของหุ่นยนต์ ออกเป็นส่วน ๆ จึงทำให้พบปัญหาในการสร้างอยู่บ้าง รวมทั้งมีจุดเด่นในแนวทางการทำงานในลักษณะโมดูลอยู่ด้วย โดยสรุปแบ่งเป็นข้อๆ ได้ดังนี้

<u>จุดเด่น</u>

1. การทำงานของแต่ละโมดูลเป็นอิสระจากกัน โดยรับเฉพาะคำสั่งมาประมวลผลและทำงานต่อไปเอง
2. การเชื่อมต่อระหว่างโมดูลมีลักษณะเป็นชั้น ซ้อนกัน ซึ่งเป็นบัสรับส่งระหว่างโมดูล ทำให้สามารถสลับชั้นของโมดูลแต่ละโมดูล โดยไม่มีปัญหาในการทำงาน
3. สามารถนำเฉพาะโมดูลที่จำเป็นต่อการทำงานของตัวหุ่นเท่านั้นมาประกอบกันเพื่อทำงานได้

<u>จุดด้อย</u>

1. ใช้ไมโครคอนโทลเลอร์ 1 ตัวต่อ 1 โมดูลทำให้สิ้นเปลืองเกินไป ในโมดูลที่มีการทำงานน้อย
2. เนื่องจากซ็อกเก็ตที่ใช้ทำเป็นบัสที่เชื่อมต่อระหว่างโมดูล ไม่เหมาะสมเท่าที่ควร และปริมาณขาสัญญาณบนบัสของหุ่นมีมากเกินไป เนื่องจากข้อกำหนดในการรับส่งข้อมูลที่กำหนดขึ้น
3. ผู้ใช้งานจำเป็นต้องเคยศึกษาการเขียนโปรแกรมด้วยภาษาซีมาพอสมควร จึงทำให้การเขียนโปรแกรมยังต้องใช้ทักษะทางภาษาโปรแกรมพอสมควร
4. ไม่สามารถโหลดโปรแกรมจากเครื่องคอมพิวเตอร์ไปยังโมดูลควบคุมหลักโดยผ่านพอร์ตอนุกรมได้ ยังต้องใช้วิธีการบันทึกโปรแกรมโดยผ่านเครื่องบันทึกภายนอกอยู่

บทที่ 8
แนวทางการพัฒนาต่อ

เนื่องจากโปรเจ็คนี้เป็นเพียงต้นแบบ เพื่อใช้เป็นแนวทางในการพัฒนาและออกแบบชุดพัฒนาหุ่นยนต์ที่มีความสามารถมากขึ้น มีการทำงานอย่างมีประสิทธิภาพมากขึ้น มีความยืดหยุ่นสูง ซึ่งจากที่ได้ศึกษาค้นคว้าและสร้างชุดพัฒนาหุ่นขึ้นมาสามารถสรุปถึงแนวทางการพัฒนาออกมาดังนี้

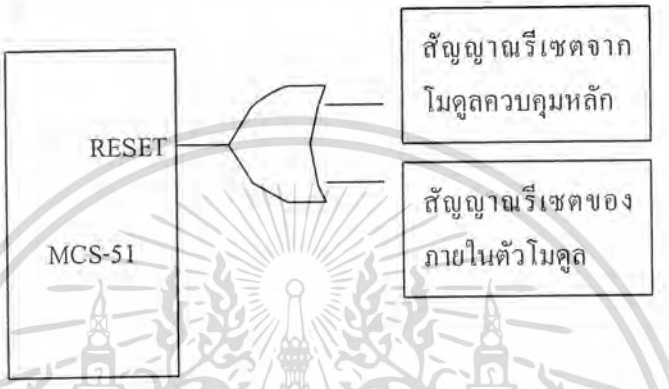
8.1 ข้อจำกัดของต้นแบบ

- เนื่องจากระบบบัสที่ใช้ในโครงงานยังไม่ใช่บัส I2C เต็มรูปแบบ ทำให้แต่ละโมดูลต้องใช้สัญญาณอินเตอร์รัปต์ในการทำงาน ทำให้โมดูล I/O ที่สามารถใช้ได้ถูกจำกัดอยู่ที่ 8 โมดูล (สัญญาณอินเตอร์รัปต์ 0-7)
- ชนิดของโมดูล I/O ในโครงงานนี้มีจำกัด เพียงแค่ โมดูลขับเคลื่อน , โมดูลอัลตราโซนิค , โมดูลอินฟราเรด และ โมดูลติดต่อกับผู้ใช้ เท่านั้น ทำให้หุ่นยนต์ที่สร้างได้มีความสามารถจำกัดในระดับหนึ่ง
- การทำงานของโมดูลอัลตราโซนิค ยังไม่เสถียร , ระยะที่สามารถตรวจจับได้มีระยะสั้น และ ระยะที่อ่านได้ยังไม่ละเอียดเท่าที่ควร
- เนื่องจากการส่งอินฟราเรดในโครงงานนี้ใช้การส่งแบบตลอดเวลา แล้ววัดแสงที่สะท้อนกลับ ไม่ใช่วิธีการส่งและรับเฉพาะช่วงความถี่ ทำให้ โมดูลอินฟราเรดทำงานผิดพลาดในที่ๆมีแสงสว่างมากเกินไป
- เนื่องจากการเขียนโปรแกรมควบคุมตัวหุ่นยนต์ใช้ภาษาซีในการเขียน ทำให้การเขียนโปรแกรมยุ่งยาก และ ผู้ใช้ต้องศึกษาการใช้งานภาษาซี
- การโปรแกรมตัวหุ่นยนต์ทำโดยการใช้เครื่องเบิร์น ทำให้เมื่อต้องการโปรแกรมใหม่ลงไป ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมา ซึ่งทำให้การทำงานไม่สะดวก

8.2 การพัฒนาทางด้านฮาร์ดแวร์

- การเชื่อมต่อกันระหว่างโมดูล ที่ใช้ซ็อกเก็ตมาดัดแปลง ไม่มีความเหมาะสมเพียงพอ ทำให้ความทนทานในการใช้งานต่ำลง และ อาจเชื่อมต่อแต่ละโมดูลได้ค่อนข้างลำบาก
- จำนวนขาสัญญาณบนบัส ที่มากเกินไป เนื่องจากในการออกแบบใช้ระบบรับส่งข้อมูลแบบบัส I2C แล้วดังนั้น หากทำการพัฒนาโปรแกรมที่ควบคุมการทำงานบนบัส I2C ได้สมบูรณ์แบบ จะลดปริมาณขาสัญญาณได้เนื่องจากไม่จำเป็นต้องใช้ขาอินเตอร์รัปต์ที่อยู่บนบัส และหากเพิ่มความเร็วในการรับส่งข้อมูลบนบัส I2C ให้เต็มความสามารถก็จะทำให้สามารถตอบสนองการทำงานของโมดูลที่อาจมีการพัฒนาขึ้นมาใหม่ที่ต้องการความเร็วในการรับส่งข้อมูลสูงได้อย่างเต็มที่

- โมดูลควบคุมหลักควรปรับปรุงรูปแบบการวางตัวอุปกรณ์บนบอร์ดให้มีความเหมาะสมมากขึ้น เนื่องจากอุปกรณ์บางตัวต้องมีการใช้งานบ่อยแต่ไม่สามารถใช้งานได้สะดวก เช่น สวิตช์รีเซต
- โมดูล I/O ทุกตัว ควรปรับปรุงส่วนของการรีเซตตัวเองเนื่องจากสัญญาณรีเซ็ตบนบัสจะมาจากโมดูลควบคุมหลัก ดังนั้นหากทำการติดตั้งโมดูลเข้าไปขณะที่ตัวหุ่นทำงานอยู่โมดูลนั้นจะทำงานไม่ถูกต้อง ซึ่งสามารถแก้ไขด้วยการเพิ่มวงจรรีเซตให้กับแต่ละโมดูล โดยที่สัญญาณรีเซตที่เข้าไปยังไมโครคอนโทรลเลอร์ที่ควบคุมโมดูลจะมาจากการทำกระบวนการตรรกยะแบบหรือ ระหว่างสัญญาณรีเซตจากโมดูลควบคุมหลักกับสัญญาณรีเซตของตัวโมดูลเอง ดังรูป



- โมดูลขับเคลื่อนควรทำการแบ่งแยกแหล่งจ่ายพลังงานที่ให้แก่ตัวมอเตอร์กับที่ให้แก่โมดูลต่าง ๆ ออกจากกัน เพื่อป้องกันการกระชากไฟทำให้แรงดันตกลงไปต่ำกว่าที่ไมโครคอนโทรลเลอร์จะสามารถทำงานได้ ทำให้เกิดการรีเซตตัวเองขึ้น รวมทั้งควรปรับปรุงวงจรตรวจระดับแรงดันแบตเตอรี่ให้มีประสิทธิภาพมากขึ้น
- เทอร์มินอลที่ใช้ต่อเข้ากับชุดออปโต้คัปเปลอร์ที่ติดตั้งอยู่ในโมดูลตรวจสอบระดับพื้นผิวควรเปลี่ยนให้มีขนาดเล็กลงเนื่องจากขนาดเทอร์มินอลที่สูงนั้นจะทำให้ความสูงของโมดูลสูงกว่าความยาวของช็อกเก็ตที่เชื่อมต่อระหว่างโมดูลทำให้ไม่สามารถติดตั้งโมดูลอยู่ด้านบนของโมดูลตรวจสอบระดับพื้นผิวได้
- โมดูลอัลตราโซนิคควรปรับปรุงส่วนส่งให้มีกำลังในการส่งสัญญาณให้สูงมากขึ้น ในส่วนรับควรเพิ่มกำลังการขยายสัญญาณที่ได้รับจากตัวรับอัลตราโซนิครวมทั้งเปลี่ยนส่วนเปรียบเทียบแรงดันให้เหมาะสม เนื่องจาก ส่วนขยายสัญญาณและส่วนเปรียบเทียบแรงดันนั้น ใช้ ออปแอมป์ชุดเดียวกันทำให้การทำงานในส่วนเปรียบเทียบแรงดันทำงานไม่ถูกต้องในบางครั้ง
- โมดูลติดต่อกับผู้ใช้ควรทำการปรับปรุงวงจรขยายสัญญาณเสียงที่ขับลำโพงเปียซโซให้มีกำลังมากขึ้น
- ทำการสร้างโมดูลขับเคลื่อนที่มีลักษณะเป็นขา เพื่อใช้ในสภาพแวดล้อมที่การใช้ล้อในการเคลื่อนที่ทำได้ยาก
- ทำการสร้างโมดูลรับส่งข้อมูลกับโมดูลรับส่งตัวอื่นหรือกับเครื่องคอมพิวเตอร์ โดยใช้คลื่นวิทยุ

8.3 การพัฒนาทางด้านซอฟต์แวร์

- พัฒนาโปรแกรมในส่วนที่เกี่ยวกับการรับส่งข้อมูลบนบัส I2C ให้ทำงานได้สมบูรณ์แบบ เพื่อลดขาสัญญาณอินเตอร์รัปต์ของโมดูล I/O ต่างๆ และ เพิ่มความเร็วในการรับส่งข้อมูลโดยการลดการหน่วงเวลาในการรับส่งข้อมูลลงให้มากที่สุด
- สร้างโปรแกรมมอนิเตอร์ไว้ภายในโมดูลควบคุมหลัก เพื่อให้การบันทึกโปรแกรมสามารถทำได้สะดวกและรวดเร็วขึ้น
- พัฒนาชุดคำสั่งของโมดูลขับเคลื่อนให้มีความละเอียดมากขึ้น เช่นคำสั่งที่สั่งให้ล้อด้านใดด้านหนึ่งหมุนเท่านั้น หรือ กำหนดความเร็วในการหมุนของแต่ละล้อแยกออกจากกัน เพื่อให้เกิดการเคลื่อนที่ที่เป็นธรรมชาติมากขึ้น
- พัฒนาชุดคำสั่งของโมดูลติดต่อกับผู้ใช้ให้มีความสามารถมากขึ้นเช่น สามารถให้จอแอลซีดี แสดงข้อความแบบเลื่อนได้ หรือ มีชุดเสียงพื้นฐานเก็บไว้ให้เรียกใช้ได้ทันที
- พัฒนาระบบการเขียนโปรแกรมแบบใช้ภาพ (Visual IDE) เพื่อให้ผู้ใช้งานสามารถเรียนรู้การทำงานและเขียนโปรแกรมเพื่อควบคุมหุ่นได้ง่ายขึ้น โดยไม่ต้องศึกษาภาษาซีมาก่อน รวมทั้งทำการพัฒนาชุดจำลองการทำงานของโปรแกรมที่ได้ออกแบบขึ้น เพื่อจำลองการทำงานโดยที่ยังไม่ทำบันทึกโปรแกรมการทำงานลงบนตัวหุ่น

ภาคผนวก ก

การใช้งาน C51 Compiler

ในการเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์โดยทั่วไปจะเขียนเป็นภาษาแอสแซมบลีซึ่งเป็นภาษาระดับต่ำ จากนั้นจึงทำการแอสแซมเบิลโค้ดเพื่อให้ได้เป็นไบนารีโค้ดที่ใช้บันทึกลงไปในไมโครคอนโทรลเลอร์ เพื่อกำหนดการทำงานของไมโครคอนโทรลเลอร์

ภาษาแอสแซมบลีเป็นภาษาระดับต่ำ ซึ่งคำสั่งของภาษาแอสแซมบลีแต่ละคำสั่งจะถูกแปลงไปเป็นไบนารีโค้ดของคำสั่งนั้นๆ จึงเปรียบได้ว่าใช้ ชุดอักขระแทนความหมายของคำสั่งที่เป็นไบนารีเพื่อให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมให้มีการทำงานตามที่ต้องการได้ง่ายกว่าการเขียนเป็นไบนารีโค้ดเพื่อบันทึกลงไมโครคอนโทรลเลอร์ ซึ่งข้อดีของการเขียนโปรแกรมด้วยภาษาแอสแซมบลีนั้น ก็คือสามารถรู้ระยะเวลาในการทำงานของโปรแกรมที่เขียนขึ้น เนื่องจากว่า คำสั่งแต่ละคำสั่งที่เขียนนั้น จะมีระยะเวลาในการประมวลผลที่แน่นอน โดยดูได้จากคู่มือคำสั่งของตัวไมโครคอนโทรลเลอร์ รวมทั้งยังทำให้สามารถปรับขนาดของโปรแกรมให้มีขนาดเล็ก หรือมีการทำงานที่เร็ว ได้เนื่องจากการแอสแซมเบิลโค้ดเป็นเพียงการแทนที่คำสั่งที่เขียนด้วยภาษาแอสแซมบลีด้วยไบนารีโค้ดที่ตรงกันเท่านั้น ดังนั้น ผู้เขียนโปรแกรมจึงสามารถเลือกเขียนโปรแกรมให้มีจำนวนโค้ดที่น้อย หรือ มีการทำงานที่เร็วได้ตามต้องการ แต่เนื่องจาก คำสั่งของภาษาแอสแซมบลีจะแปลงเป็นไบนารีโค้ดโดยตรงจึงทำให้การเขียนโปรแกรมที่มีการทำงานที่ซับซ้อน หรือมีขนาดใหญ่นั้น ก็ทำได้ลำบาก การแก้ไข และการพัฒนาที่ตามมา ก็จะเป็นปัญหามากขึ้นด้วยเช่นกัน ดังนั้นผู้ที่เขียนโปรแกรมจะต้องมีทักษะในการเขียนโปรแกรมด้วยภาษาแอสแซมบลีพอที่จะลำดับการทำงานของโปรแกรม และเขียนโปรแกรมตามที่ต้องการได้

ภาษาซีเป็นภาษาระดับสูงที่พัฒนาขึ้นมาเพื่อใช้เขียนโปรแกรมที่ทำงานกับเครื่องไมโครคอมพิวเตอร์โดยทั่วไป ซึ่งลักษณะของภาษาระดับสูงคือคำสั่งของภาษาระดับสูงจะคอมไพล์ (แปลงไป) เป็นชุดคำสั่งในระดับไบนารีโค้ดหลาย ๆ คำสั่ง ทำให้ลดปริมาณโค้ดที่เขียนขึ้น รวมทั้งยังทำความเข้าใจในตัวโปรแกรมได้ง่ายขึ้นด้วย จึงได้มีการพัฒนาภาษาซีขึ้นมาเพื่อใช้ในการเขียนโปรแกรมที่ควบคุมการงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 หรือเรียกว่า C51 Compiler

ก.1 การคอมไพล์โค้ดโดยใช้ C51 Compiler

การคอมไพล์โค้ดภาษาซี ด้วย C51 Compiler แบ่งออกเป็น 2 ส่วน คือ

1. การแปลโค้ดภาษาซีเป็นออบเจ็กโค้ด หรือ โค้ดคำสั่งระดับไบนารีที่ยังไม่ได้รวมเข้ากับคำสั่งในคลังคำสั่งที่เหมาะสม โดยใช้โปรแกรม c51.exe ดังนี้

   c51.exe filename_in

   filename_in เป็นโปรแกรมภาษาซีที่เขียนขึ้นมา

   จะได้ไฟล์ออบเจ็กต์ที่มีชื่อเดียวกับ filename_in แต่มีนามสกุลเป็น .obj

2. การรวมโค้ดของโปรแกรมเข้ากับโค้ดในคลังคำสั่งที่เหมาะสมกับลักษณะของโปรแกรม เพื่อได้เป็นไบนารีโค้ดที่สามารถทำงานได้ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยใช้โปรแกรม l51.exe ดังนี้

l51.exe file_obj, other_obj, library.lib file_bin CODE(?C_C51STARTUP(0040H), 0080H) XDATA(8000H) NOOVERLAY

file_obj เป็นไฟล์ออบเจ็กต์โค้ด

other_obj เป็นไฟล์ออบเจ็กต์โค้ดอื่นๆ ที่ต้องการรวมเข้าด้วยกัน

library.lib เป็นไฟล์คลังคำสั่งมาตรฐาน ที่จำเป็นต้องรวมเข้าไปด้วยกัน แบ่งเป็น 3 ไฟล์ ตามรูปแบบของโปรแกรมดังนี้

C51S.LIB ใช้เมื่อรูปแบบของโปรแกรมเป็นแบบ SMALL

C51C.LIB ใช้เมื่อรูปแบบของโปรแกรมเป็นแบบ COMPACT

C51L.LIB ใช้เมื่อรูปแบบของโปรแกรมเป็นแบบ LARGE

file_bin เป็นไฟล์ไบนารีโค้ดที่ใช้บันทึกลงบนไมโครคอนโทรลเลอร์ตระกูล MCS-51

CODE(?C_C51STARTUP(0040H),0080H) เป็นการระบุตำแหน่งการเริ่มทำงานของโปรแกรมที่ตำแหน่ง 0080H และเซ็กเมนต์ของโค้ดเริ่มที่ 0040H

XDATA(8000H) เป็นการระบุตำแหน่งเริ่มต้นในการอ้างอิงหน่วยความจำภายนอก

โดยทั่วไปแล้วการนำไบนารีโค้ดบันทึกลงบนไมโครคอนโทรลเลอร์ตระกูล MCS-51 นั้นจะใช้เครื่องบันทึกโปรแกรม ซึ่งมีขายตามท้องตลาด ซึ่งเครื่องบันทึกนี้จะรับโค้ดที่ต้องการบันทึกลงบนไมโครคอนโทรลเลอร์ในรูปของไฟล์แบบเลขฐาน 16 ซึ่งใน C51 Compiler มีโปรแกรมที่ใช้สำหรับแปลงไบนารีโค้ดไปเป็นไฟล์แบบเลขฐาน 16 อยู่ด้วย คือ oha51.exe ดังนี้

oha51.exe file_bin file_hex

file_bin เป็นไฟล์ไบนารีโค้ดที่ใช้บันทึกลงบนไมโครคอนโทรลเลอร์ตระกูล MCS-51

file_hex เป็นไฟล์แบบเลขฐาน 16 ที่ให้เครื่องบันทึกอ่านเพื่อบันทึกโปรแกรมลงบนไมโครคอนโทรลเลอร์ตระกูล MCS-51

ก.2 คำสั่งในภาษาซีของ C51 Compiler

ภาษาซีที่ใช้เขียนเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีรูปแบบภาษาที่เหมือนกับภาษาซีที่ใช้สำหรับเขียนโปรแกรมบนไมโครคอมพิวเตอร์โดยทั่วไป แต่จะมีการเพิ่มเติมบางส่วนเข้าไป เพื่อให้เหมาะสมกับการทำงานกับไมโครคอนโทรลเลอร์ตระกูล MCS-51

ในส่วนของคำสั่งโดยทั่วไปของภาษาซีสามารถดูได้จากคู่มือภาษาซีโดยทั่วไป จะขอกล่าวถึงส่วนที่เพิ่มเติมเข้ามาเท่านั้น

ก.2.1 ส่วนของการกำหนดลักษณะของโปรแกรม

ต้องกำหนดไว้ที่บรรทัดแรกของโปรแกรมใช้สำหรับกำหนดลักษณะของโปรแกรมที่ทำงานบนไมโครคอนโทรลเลอร์โดยใช้คำสั่ง #pragma ตามด้วย ชนิดของโปรแกรมดังนี้

SMALL	อ้างอิงโค้ดภายใน 64 กิโลไบต์ และ หน่วยความจำชั่วคราวภายใน 256 ไบต์
COMPACT	อ้างอิงตัวแปรจากหน่วยความจำภายนอกด้วยพอร์ต 0 และ 2 และอ้างอิงคำสั่งโดยอ้อม
LARGE	อ้างอิงตัวแปรและข้อมูลต่างจากหน่วยความจำภายนอกโดยอ้างอิงผ่าน @DPTR

ก.2.2 ส่วนของการประกาศชนิดข้อมูล

ชนิดข้อมูลที่เพิ่มเติมเข้ามามีดังนี้

sbit	ใช้ประกาศตัวแปรให้มีขนาดเพียงบิตเดียว ที่ใช้พื้นที่ของหน่วยความจำที่เข้าถึงแบบบิตได้
sfr	ใช้ประกาศตัวแปรให้เป็นรีจิสเตอร์พิเศษ ที่ใช้พื้นที่ของหน่วยความจำรีจิสเตอร์พิเศษ
bdata	ใช้ประกาศตัวแปรขนาด 8 บิต ที่ใช้พื้นที่ของหน่วยความจำที่เข้าถึงแบบบิตได้
idata	ใช้ประกาศตัวแปรให้ใช้พื้นที่หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์
xdata	ใช้ประกาศตัวแปรให้ใช้พื้นที่หน่วยความจำภายนอกตัวไมโครคอนโทรลเลอร์

ก.2.3 ส่วนของการทำงานเมื่อเกิดอินเตอร์รัปต์

การใช้งานสามารถทำได้โดยใช้คำสั่ง interrupt ตามด้วยเลขเบอร์อินเตอร์รัปต์ที่บอกชนิดของอินเตอร์รัปต์ตามหลัง ฟังก์ชัน ที่ต้องการให้ทำงานเมื่อเกิดอินเตอร์รัปต์ประเภทต่างๆ ขึ้น

ตัวเลขที่ตามหลังคำสั่ง interrupt มีดังนี้

เบอร์ 0 หมายถึง อินเตอร์รัปต์ภายนอกเบอร์ 0

เบอร์ 1 หมายถึง อินเตอร์รัปต์ไทเมอร์เบอร์ 0

เบอร์ 2 หมายถึง อินเตอร์รัปต์ภายนอกเบอร์ 1

เบอร์ 3 หมายถึง อินเตอร์รัปต์ไทเมอร์เบอร์ 1

เบอร์ 4 หมายถึง อินเตอร์รัปต์พอร์ตอนุกรม

ตัวอย่างการประกาศฟังก์ชันให้ทำงานเมื่ออินเตอร์รัปต์เกิดขึ้น

```
void int0_interrupt_routine interrupt 0
{
……….
}
```

ในส่วน void int0_interrupt_routine เป็นการประกาศชื่อฟังก์ชัน ซึ่งจะเกิดการทำงานภายในฟังก์ชันเมื่อ อินเตอร์รัปต์ เบอร์ 0 หรือ อินเตอร์รัปต์ภายนอกเบอร์ 0 เกิดขึ้น

ภาคผนวก ข

ชุดคำสั่งสำหรับควบคุมโมดูลต่างๆ

การพัฒนาโปรแกรมที่ใช้เพื่อควบคุมไมโครโรบอตจะพัฒนาบนภาษาซีที่ออกแบบพิเศษเพื่อให้สามารถคอมไพล์โค้ดไปเป็นคำสั่งภาษาเครื่องของไมโครคอนโทรลเลอร์ในตระกูล MCS51 ได้ โดยมีชุดคำสั่งและแอดเดรสที่ใช้เชื่อมต่อกับ module ต่างๆ ดังนี้

โมดูลขับเคลื่อน

ใช้ขาสัญญาณอินเตอร์รัปต์ INT_0 ในการเรียกโมดูลทำงาน โดยอ้างอิงแอดเดรสบนบัส I2C เป็น 0x01 เนื่องจากใช้เพียง 7 บิต หรือ 0x02 ในโหมด 8 บิต เพิ่มบิตต่ำสุดเข้าไปเป็นบิตสำหรับเลือกการอ่านหรือเขียนข้อมูล

รหัสคำสั่ง	0x01
คำสั่ง	void MT_go_forward()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	สั่งงานให้เคลื่อนที่ไปด้านหน้า

รหัสคำสั่ง	0x02
คำสั่ง	void MT_go_backward()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	สั่งงานให้เคลื่อนที่ถอยหลัง

รหัสคำสั่ง	0x03
คำสั่ง	void MT_turn_left()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	สั่งงานให้เคลื่อนที่หมุนตัวไปทางซ้าย

รหัสคำสั่ง	0x04
คำสั่ง	void MT_turn_right()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	สั่งงานให้เคลื่อนที่หมุนตัวไปทางขวา

รหัสคำสั่ง	0x05
คำสั่ง	void MT_stop_move()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	สั่งงานให้หยุดการเคลื่อนที่

รหัสคำสั่ง	0x06
คำสั่ง	void MT_set_speed(unsigned char speed)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ค่าความเร็วในการเคลื่อนที่ speed
การทำงาน	กำหนดความเร็วในการเคลื่อนที่ตามค่าความเร็วที่กำหนด

รหัสคำสั่ง	0x07
คำสั่ง	void MT_set_run_time(unsigned int time)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ค่าระยะเวลาในการเคลื่อนที่ time
การทำงาน	กำหนดระยะเวลาในการเคลื่อนที่ตามระยะเวลาที่กำหนด

รหัสคำสั่ง	0x08
คำสั่ง	void STEPPING_set_single_phase_mode()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	กำหนดให้การเลื่อนเฟสของสเต็ปปิ้งมอเตอร์เป็นแบบ 1 เฟส

รหัสคำสั่ง	0x08
คำสั่ง	void STEPPING_set_double_phase_mode()
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	กำหนดให้การเลื่อนเฟสของสเต็ปปิ้งมอเตอร์เป็นแบบ 2 เฟส

โมดูลติดต่อกับผู้ใช้

ใช้ขาสัญญาณอินเตอร์รัปต์ INT_3 ในการเรียกโมดูลทำงาน โดยอ้างอิงแอดเดรสบนบัส I2C เป็น 0x04 เนื่องจากใช้เพียง 7 บิต หรือ 0x08 ในโหมด 8 บิต เพิ่มบิตต่ำสุดเข้าไปเป็นบิตสำหรับเลือกการอ่านหรือเขียนข้อมูล

รหัสคำสั่ง	0x01
คำสั่ง	void LCD_write_char(unsigned char ch)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ตัวอักษรที่ต้องการแสดง ch
การทำงาน	แสดงตัวอักษรบนจอแอลซีดีในตำแหน่งที่เคอร์เซอร์อยู่

รหัสคำสั่ง	-
คำสั่ง	void LCD_write_byte(unsigned char in)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ข้อมูล 1 ไบต์ที่ต้องการแสดงเป็นค่าตัวเลข
การทำงาน	แสดงค่าตัวเลขของ in บนจอแอลซีดีในตำแหน่งที่เคอร์เซอร์อยู่

รหัสคำสั่ง	-
คำสั่ง	void LCD_write_int(unsigned int in)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ข้อมูล 1 เวิร์ด หรือ 2 ไบต์ที่ต้องการแสดงเป็นค่าตัวเลข
การทำงาน	แสดงค่าตัวเลขของ in บนจอแอลซีดีในตำแหน่งที่เคอร์เซอร์อยู่

รหัสคำสั่ง	0x01
คำสั่ง	void LCD_write_string(unsigned char* str)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	สายอักขระที่ต้องการแสดงผล
การทำงาน	แสดงผลข้อมูลสายอักขระบนจอแอลซีดีในตำแหน่งที่เคอร์เซอร์อยู่

รหัสคำสั่ง	0x02
คำสั่ง	void LCD_gotoxy(unsigned char x,y)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ค่าตำแหน่งคอลัมน์ x และ บรรทัด y
การทำงาน	ย้ายตำแหน่งเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ที่ x บรรทัดที่ y

รหัสคำสั่ง	0x03
คำสั่ง	void LCD_clear(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	ลบตัวอักษรบนจอแอลซีดีแล้วย้ายเคอร์เซอร์ไปที่มุมบนซ้ายของจอ

รหัสคำสั่ง	0x07
คำสั่ง	unsigned char KBD_get_key(void)
ชนิดข้อมูลที่คืนกลับ	ค่าคีย์ที่ถูกกด
พารามิเตอร์	ไม่มี
การทำงาน	อ่านค่าคีย์ที่ถูกกดขณะนั้นโดยรอการกดคีย์

รหัสคำสั่ง	0x08
คำสั่ง	unsigned char KBD_scan_key(void)
ชนิดข้อมูลที่คืนกลับ	ค่าคีย์ที่ถูกกดขณะนั้น
พารามิเตอร์	ไม่มี
การทำงาน	อ่านค่าคีย์ที่ถูกกดขณะนั้นโดยไม่มีการรอการกดคีย์

รหัสคำสั่ง	0x05
คำสั่ง	void SND_gen_freq(unsigned int freq)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ค่าความถี่ที่ต้องการสร้าง
การทำงาน	สร้างความถี่ตามค่าความถี่ที่กำหนด

รหัสคำสั่ง	0x06
คำสั่ง	void SND_stop_freq(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	หยุดการสร้างความถี่

โมดูลตรวจสอบระดับพื้นผิว

ใช้ขาสัญญาณอินเตอร์รัปต์ INT_1 ในการเรียกโมดูลทำงาน โดยอ้างอิงแอดเดรสบนบัส I2C เป็น 0x02 เนื่องจากใช้เพียง 7 บิต หรือ 0x04 ในโหมด 8 บิต เพิ่มบิตต่ำสุดเข้าไปเป็นบิตสำหรับเลือกการอ่านหรือเขียนข้อมูล

รหัสคำสั่ง	ตามค่าตัวแปร no
คำสั่ง	unsigned char IR_get_data_no(unsigned char no)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ชุดอินฟราเรดที่ต้องการอ่านค่า
การทำงาน	อ่านค่าการตรวจสอบระดับพื้นผิวจากชุดอินฟราเรดที่ต้องการ

รหัสคำสั่ง	0x0F
คำสั่ง	void IR_get_data(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลการตรวจสอบพื้นผิวจากชุดอินฟราเรดทั้งหมด เก็บไว้ในตัวแปร infrared_data_hi และ infrared_data_lo เป็นไบต์สูง และไบต์ต่ำตามลำดับ เรียงชุดอินฟราเรดตามบิต

รหัสคำสั่ง	L_ADDRESS (0x55)
คำสั่ง	void IR_get_sample(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลตัวอย่างจากโมดูลอินฟราเรดจำนวน 2 ไบต์เก็บไว้ในตัวแปร infrared_data_hi และ infrared_data_lo

รหัสคำสั่ง	0x11
คำสั่ง	void IR_enable_interrupt(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	ยอมให้โมดูลอินฟราเรดสามารถส่งสัญญาณอินเตอร์รัปต์กลับมายังโมดูลควบคุมในกรณีที่ตรวจสอบไม่พบพื้นผิวได้

รหัสคำสั่ง	0x12
คำสั่ง	void IR_disable_interrupt(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	ยกเลิกความสามารถในการส่งสัญญาณอินเตอร์รัปต์กลับมายังโมดูลควบคุมในกรณีที่ตรวจสอบไม่พบพื้นผิวได้

โมดูลตรวจสอบสิ่งกีดขวาง

ใช้ขาสัญญาณอินเตอร์รัปต์ INT_2 ในการเรียกโมดูลทำงาน โดยอ้างอิงแอดเดรสบนบัส I2C เป็น 0x03 เนื่องจากใช้เพียง 7 บิต หรือ 0x06 ในโหมด 8 บิต เพิ่มบิตต่ำสุดเข้าไปเป็นบิตสำหรับเลือกการอ่านหรือเขียนข้อมูล

รหัสคำสั่ง	0x01
คำสั่ง	void US_get_data(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลระยะของสิ่งกีดขวางที่ตรวจจับได้จากชุดอัลตราโซนิคทั้ง 2 ชุด โดยเก็บไว้ในตัวแปร ultrasonic01 และ ultrasonic02

รหัสคำสั่ง	0x02
คำสั่ง	unsigned char US_get_data01(void)
ชนิดข้อมูลที่คืนกลับ	ค่าระยะทางที่อ่านได้จากชุดอัลตราโซนิกชุดที่ 1 มีขนาด 8 บิต
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลระยะของสิ่งกีดขวางที่ตรวจจับได้จากอัลตราโซนิคชุดที่ 1 โดยคืนค่าระยะทางกลับมาจากฟังก์ชัน

รหัสคำสั่ง	0x03
คำสั่ง	unsigned char US_get_data02(void)
ชนิดข้อมูลที่คืนกลับ	ค่าระยะทางที่อ่านได้จากชุดอัลตราโซนิคชุดที่ 2 มีขนาด 8 บิต
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลระยะของสิ่งกีดขวางที่ตรวจจับได้จากอัลตราโซนิคชุดที่ 1 โดยคืนค่าระยะทางกลับมาจากฟังก์ชัน

รหัสคำสั่ง	L_ADDRESS (0x55)
คำสั่ง	void US_get_sample(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	อ่านข้อมูลตัวอย่างจากโมดูลอัลตราโซนิคจำนวน 2 ไบต์เก็บไว้ในตัวแปร ultrasonic01 และ ultrasonic02

รหัสคำสั่ง	0x11
คำสั่ง	void US_enable_interrupt(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	ยอมให้โมดูลอัลตราโซนิคสามารถส่งสัญญาณอินเตอร์รัปต์กลับมายังโมดูลควบคุมในกรณีที่ตรวจสอบเจอสิ่งกีดขวางได้

รหัสคำสั่ง	0x12
คำสั่ง	void US_disable_interrupt(void)
ชนิดข้อมูลที่คืนกลับ	ไม่มี
พารามิเตอร์	ไม่มี
การทำงาน	ยกเลิกความสามารถในการส่งสัญญาณอินเตอร์รัปต์กลับมายังโมดูลควบคุมในกรณีที่ตรวจสอบเจอสิ่งกีดขวางได้

ภาคผนวก ค

รายละเอียดคำสั่งของโมดูลแอลซีดี

1. CLEAR DISPLAY

RS	R/W	DB7	----------	DB0
0	0	0	0	0	0	0	0	0	1

คำสั่งนี้จะเป็นเขียนช่องว่างเข้าไปในหน่วยความจำของโมดูลแอลซีดี และเซ็ตตำแหน่งของเคอร์เซอร์กลับไปที่ตำแหน่งบนซ้ายของจอ

2. RETURN HOME

RS	R/W	DB7	----------	DB0
0	0	0	0	0	0	0	0	1	X

คำสั่งนี้เป็นการเซ็ตตำแหน่งของเคอร์เซอร์กลับไปที่ตำแหน่งบนซ้ายของจอ

3. ENTRY MODE SET

RS	R/W	DB7	----------	DB0
0	0	0	0	0	0	0	0	I/D	S

บิต I/D เป็นตัวกำหนดว่าเมื่ออ่านหรือเขียนหน่วยความจำของโมดูลแอลซีดีแล้วจำให้ค่าตำแหน่งเพิ่มหรือลด โดยถ้า

1 = เพิ่มขึ้นหนึ่ง

0 = ลดลงหนึ่ง

บิต S เป็นตัวกำหนดการแสดงผล โดยถ้า

S = 1 จะเป็นการใส่ข้อมูลแล้วเคอร์เซอร์อยู่กับที่ข้อมูลจะถูกดันไปทางซ้าย

S = 0 ข้อมูลจะอยู่กับที่ตัวเคอร์เซอร์จะถูกดันไปทางขวามือ

4. DISPLAY ON/OFF CONTROL

RS	R/W	DB7	----------						DB0
0	0	0	0	0	0	1	D	C	B

บิต D เป็นบิตควบคุมการเปิดปิดของจอ โดยถ้า

D = 1 เปิด

D = 0 ปิด

บิต C เป็นบิตควบคุมการแสดงผลของเคอร์เซอร์ โดยถ้า

C = 0 ไม่แสดง

C = 1 แสดง

บิต B เป็นบิตควบคุมการกระพริบของเคอร์เซอร์ โดยถ้า

B = 1 มีการกระพริบ

B = 0 ไม่มีการกระพริบ

5. CURSOR OR DISPLAY SHIFT

RS	R/W	DB7	----------						DB0
0	0	0	0	0	1	S/C	R/L	X	X

เป็นคำสั่งที่กำหนดให้ตำแหน่งของเคอร์เซอร์หรือข้อมูลไปแสดงทางซ้ายหรือขวาโดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน โดยถ้า

S/C	R/L	
0	0	ทำการย้ายเคอร์เซอร์จากตำแหน่งเดิมไปซ้ายมือ 1 ตำแหน่ง
0	1	ทำการย้ายเคอร์เซอร์จากตำแหน่งเดิมไปขวามือ 1 ตำแหน่ง
1	0	ทำการดันตัวอักษรที่แสดงผลไปทางซ้าย
1	1	ทำการดันตัวอักษรที่แสดงผลไปทางขวา

6. FUNCTION SET

RS	R/W	DB7	----------						DB0
0	0	0	0	1	DL	N	F	X	X

บิต DL เป็นการเลือกการติดต่อว่าจะให้เป็นแบบ 8 บิต หรือ 4 บิต โดยถ้า

DL = 0 เป็น 4 บิต

DL = 1 เป็น 8 บิต

บิต N เป็นการเลือกจำนวนบรรทัดในการแสดงผล โดยถ้า

N = 0 แสดงผลเพียง 1 บรรทัด

N = 1 แสดงผลมากกว่า 1 บรรทัดสำหรับโมดูลที่สามารถแสดงผลได้มากกว่า 1 บรรทัด

บิต F เป็นการเลือกขนาดตัวอักษรในการแสดงผล โดยถ้า

F = 0 แสดงผลแบบ 5x7 จุด

F = 1 แสดงผลแบบ 5x10 จุด

7. SET CG RAM ADDRESS

RS R/W DB7 ---------- DB0

0	0	0	1	A	A	A	A	A	A

กำหนดตำแหน่งของหน่วยความจำเก็บรูปแบบตัวอักษรที่ต้องการอ่านหรือเขียน

8. SET DD RAM ADDRESS

RS R/W DB7 ---------- DB0

0	0	1	A	A	A	A	A	A	A

กำหนดตำแหน่งของหน่วยความจำแสดงผลที่ต้องการอ่านหรือเขียน โดยที่จำนวนแอดเดรสที่อ้างอิงบนจอจะขึ้นอยู่กับ บิต N ด้วย โดยถ้า

บิต N = 0 แอดเดรสจะอยู่ในช่วง 00H – 40H

บิต N = 1 แอดเดรสจะอยู่ในช่วง 00H – 27H สำหรับบรรทัดที่ 1 และ 40H - 67H สำหรับบรรทัดที่ 2

9. READ BUSY FLAG AND ADDRESS

RS R/W DB7 ---------- DB0

0	1	BF	A	A	A	A	A	A	A

อ่านค่าแฟลก BUSY ซึ่งเป็นตัวบอกว่าโมดูลแอลซีดีพร้อมรับคำสั่งต่อไปหรือไม่ โดยถ้า

บิต BF = 1 อยู่ในขบวนการทำงาน

บิต BF = 0 พร้อมรับข้อมูลหรือคำสั่งต่อไปได้

10. WRITE DATA TO CG or DD RAM

RS	R/W	DB7							DB0
1	0	D	D	D	D	D	D	D	D

เขียนข้อมูลเข้าไปยังหน่วยความจำแสดงผลหรือหน่วยความจำเก็บรูปแบบตัวอักษร ในตำแหน่งปัจจุบันที่ได้รับจากคำสั่ง SET CG or DD RAM ADDRESS เมื่อเขียนข้อมูลแล้วแอดเดรสของหน่วยความจำจะเพิ่มหรือลดลงอัตโนมัติ ขึ้นอยู่กับการเซ็ตใน ENTRY MODE

11. READ DATA FROM CG or DD RAM

RS	R/W	DB7							DB0
1	1	D	D	D	D	D	D	D	D

อ่านข้อมูลจากหน่วยความจำแสดงผลหรือหน่วยความจำเก็บรูปแบบตัวอักษร ในตำแหน่งปัจจุบันที่ได้รับจากคำสั่ง SET CG or DD RAM ADDRESS

บรรณานุกรม

หนังสืออ้างอิง

[1] สุนทร วิทูสุรพจน์ : "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051" บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) 2537

[2] นายไอซี : บทความเรื่อง "พื้นฐานของทรานซิสเตอร์" จาก "รวมบทความ ทฤษฎี และการประยุกต์ใช้งาน อุปกรณ์สารกึ่งตัวนำ" บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน) 2538

[3] "DOT MATRIX LCD MODULE" บริษัท อีทีที จำกัด

[4] Ott, D. E. and Wilderotter, T. J. A Designer's Guide to VHDL Synthesis. Klower Academic. 1994.

[5] Xilinx Inc. The Programmable Logic Data Book. Xilinx Inc. 1994.

โฮมเพจอ้างอิง

[1] www.atmel.com : 8051 Architecture

http://www.atmel.com/atmel/products/prod20.htm

[2] www.xilinx.com : Xilinx Data Book 2000

http://www.xilinx.com/partinfo/databook.htm