การควบคุมที่เวลาจริง

Realtime Workshop

โดย

นาย พูนพิพัฒน์ ศรีสุพรรณ

นาย โอพาร อมรวัฒนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมที่เวลาจริง

Realtime workshop

ผู้จัดทำ

1. นาย พูนพิพัฒน์ ศรีสุพรรณ 40010532
2. นาย โอฬาร อมรวัฒนา 40011053

อาจารย์ที่ปรึกษา

(รศ. สุเธียร เกียรติสุนทร)

การควบคุมที่เวลาจริง

นาย พูนพิพัฒน์ ศรีสุพรรณ 40010532
นาย โอฬาร อมรวัฒน่า 40011053
อาจารย์ที่ปรึกษา อ. สุเธียร เกียรติสุนทร
ปีการศึกษา 2543

บทคัดย่อ

โครงงานนี้เรียบเรียงขึ้น เพื่อจุดประสงค์ที่จะเพิ่มประสิทธิภาพของฟังก์ชั่นซิมูลิ้งค์ (Simulink) ในโปรแกรมแมทแลป (Matlab) ซึ่งเป็นฟังก์ชั่นที่ใช้วิเคราะห์ระบบจำลองที่มีประสิทธิภาพสูงให้สามารถวิเคราะห์ระบบจริง(Real System) ได้ โดยใช้โปรแกรมภาษาซี(C) เป็นตัวกลางเชื่อมโยงให้ซิมูลิ้งค์และระบบจริงส่งผ่านตัวแปรซึ่งกันและกันได้ โดยการสร้างบล็อคที่เรียกว่าซีเม็ค เอส-ฟังก์ชั่น (C-Mex as S-function) ขึ้นในซิมูลิ้งค์ ซึ่งมีลักษณะเหมือนซิมูลิ้งค์บล็อคทั่วไป แต่มีคุณสมบัติพิเศษในด้านของการติดต่อกับอุปกรณ์ภายนอกโดยผ่านทางพอร์ตขนาน (Parallel Port) โดยได้รับข้อมูลที่ถูกแปลงสัญญาณแล้วจาก อะนาลอกเป็นดิจิตอล โดยบอร์ดอินพุต-เอาท์พุต (I / Oboard)

Abstract

This project is Realtime Workshop. Its idea is to connect simulink program to the real system outside by using C language to link and transfer all the signals and parameters into the simulink program or out to the real system passing through A/D and D/A converter of interface card.

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เป็นที่ทราบกันดีว่าในปัจจุบันโปรแกรม แมทแลป เป็นโปรแกรมที่สามารถพัฒนา วิเคราะห์และออกแบบระบบควบคุมได้อย่างมีประสิทธิภาพ และเป็นที่นิยมใช้กันอย่างแพร่หลาย โดยเฉพาะการใช้ซิมูลิ้งค์ในการออกแบบ วิเคราะห์ระบบเพราะทำได้ง่าย ด้วยการสร้างโมเดลขึ้นมาเป็นบล็อคโดยใช้การปฏิบัติการณ์คลิกและลากของเมาส์ (Click and Drag Mouse Operation) แต่เป็นที่น่าเสียดายที่สามารถทำการวิเคราะห์ได้เพียงระบบจำลองเท่านั้น (Simulated System) เท่านั้น ซึ่งหมายความว่าผู้ใช้ต้องทำการจำลองระบบจริงก่อนทำการวิเคราะห์ ซึ่งอาจทำให้เกิดความคลาดเคลื่อนของตัวแปรต่างๆ โดยเฉพาะระบบที่มีความละเอียดอ่อนสูงหรือระบบที่ตัวแปรมีการเปลี่ยนแปลงค่าตลอดเวลา อาจไม่สามารถทำการวิเคราะห์ได้เลย แต่ถ้าเราสามารถเชื่อมโยงให้ซิมูลิ้งค์ ติดต่อกับระบบจริงได้โดยตรงที่เวลาใกล้เคียงความจริง ก็สามารถนำตัวแปรต่างที่เกิดขึ้นจริงในเวลานั้นเข้ามาทำการวิเคราะห์ได้ทันทีเป็นการเพิ่มประสิทธิภาพของซิมูลิ้งค์ในโปรแกรมแมทแลปให้สูงขึ้น

ในการออกแบบระบบควบคุมระบบหนึ่งนั้น ยกตัวอย่างเช่นระบบควบคุมมอเตอร์ (Motor Control) ทำได้ช้ามาก เพราะจะต้องเสียเวลาในการออกแบบตัวควบคุม โดยการจำลองระบบที่ต้องการควบคุมลงไปในซิมูลิ้งค์ก่อนจึงจะทำการออกแบบตัวควบคุมได้ และเมื่อได้ตัวควบคุมที่เป็นที่พอใจแล้วก็ต้องนำมาออกแบบวงจรควบคุมก่อน จึงจะสามารถนำไปต่อทดลองกับระบบจริงได้ ซึ่งจะต้องทำกลับไปกลับมาทำให้เสียเวลาเป็นอย่างมาก ดังนั้นเราจึงควรนำเอาเอาท์พุตจากระบบจริงเข้ามาคำนวณ ผลในซิมูลิ้งค์โดยตรงจึงจะให้ตัวควบคุมที่ถูกต้องกว่า แต่แมทแลปไม่สามารถติดต่อกับระบบภายนอกได้ด้วยตัวเอง เราจึงจำเป็นต้องสร้างซีเม็ด เอสฟังก์ชั่นบล็อค ขึ้นมาเพื่อเป็นตัวช่วยในการส่งผ่านตัวแปรต่างๆเข้ามาที่ ซิมูลิ้งค์ของโปรแกรมแมทแลปได้ทันที

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาการจำลองระบบทั้งแบบซิมูเลชั่นและแบบเรียลไทม์

1.2.2 ศึกษาการเขียนโปรแกรมในรูปแบบ ซีเมค เอสฟังก์ชั่น เพื่อใช้ในการติดต่อกับภายนอก

1.2.3 เพื่อให้เข้าใจขั้นตอนการติดต่อกับระบบภายนอกโดใช้บอร์ดอินเตอร์เฟส

1.3 ขอบเขตของโครงงาน

สร้างบล็อคไดอะแกรมบนหน้าต่างซิมูลิ้งค์ในแบบเรียลไทม์ และรับค่าสัญญาณจากภายนอกผ่านทางบอร์ดอินเตอร์เฟสได้

1.4 วิธีการดำเนินงาน

1.4.1ฮาร์ดแวร์

ในส่วนของวงจรของบอร์ดอินเตอร์เฟสประกอบด้วยวงจร ซีโร่สแปน (ZERO-SPAN) วงจรอินเตอร์เฟสแบบ IC และวงจรภาคจ่ายไฟ

1.4.2 ซอฟต์แวร์

ในส่วนของซอฟต์แวร์นั้นใช้โปรแกรมซีเม็คไฟล์(C-Mex files) ในการติดต่อ และใช้ส่วนของแมทแลปในการแสดงผล

บทที่ 2
ทฤษฎีและหลักการ

2.1 โปรแกรมแมทแลป

โปรแกรมที่มีประสิทธิภาพสูงในการคำนวณด้านเทคนิค เป็นการรวมกันของการคำนวณ การมองเห็นได้และการเขียนโปรแกรม ในรูปแบบที่ง่ายต่อการใช้ โดยที่ปัญหาและการแก้ปัญหาถูกแสดงในรูปแบบทางคณิตศาสตร์ ลักษณะการใช้ทั่วไปของแมทแลป ได้แก่

- คณิตศาสตร์ และการคำนวณ
- การพัฒนาทางด้านอัลกอลิทึม
- การจำลองระบบ และการสร้างโมเดล
- การวิเคราะห์ข้อมูล การสำรวจ และการแสดงระบบโดยใช้ภาพ
- กราฟฟิคทางด้านวิทยาศาสตร์และทางด้านวิศวกรรม
- การพัฒนาและการสร้างการติดต่อกับผู้ใช้ในแบบกราฟฟิค

แมทแลปเป็นระบบปฏิสัมพันธ์ (interactive system) ที่ส่วนประกอบของข้อมูลพื้นฐานเป็นแบบอาร์เรย์ซึ่งไม่มีมิติ ทำให้สามารถแก้ปัญหาการคำนวณทางด้านเทคนิคโดยเฉพาะการคำนวณโดยใช้สูตรในทางเมตริกซ์และเวคเตอร์ ซึ่งต้องใช้การเขียนโปรแกรมในรูปแบบภาษา สเกลาร์นอนอินเทอร์เรคทีฟ (scalar noninteractive) เช่น ภาษาซีและ ฟอร์แทรน

แมทแลป จะมีชุดของคำสั่งเฉพาะที่ใช้ในการแก้ปัญหาเรียกว่า ทูลบ็อกซ์ (toolboxes) ซึ่งมีความสำคัญอย่างมากต่อผู้ใช้ ซึ่งผู้ใช้จะได้รับเทคนิคจากการเรียนรู้และพัฒนา ทูลบ็อกซ์จะรวบรวมเอ็มไฟล์ ซึ่งเป็นตัวช่วยให้แมทแลปสามารถแก้ปัญหาที่หลากหลายได้ เช่น การประมวลสัญญาณ (signal processing) ,การควบคุมระบบ (control system) ,ฟัซซี่ ลอจิก (fuzzy logic) ,เวฟเลทส (wave lets), การจำลองระบบ(simulation) ฯลฯ

2.2 ซิมูลิ้งค์

ซิมูลิ้งค์เป็นซอฟต์แวร์ ที่ใช้สำหรับการสร้างโมเดล , การจำลอง และการวิเคราะห์ระบบแบบไดนามิค ซึ่งสามารถใช้ได้กับ ระบบที่เป็นเชิงเส้นและไม่เชิงเส้นโดยสร้างโมเดลในรูปแบบระบบเวลาต่อเนื่อง ,ระบบเวลาไม่ต่อเนื่อง และระบบไฮบริด โดยที่ระบบเป็นแบบมัลติเรต (multirate) ได้

สำหรับการสร้างโมเดลซิมูลิงค์ มีการติดต่อกับผู้ใช้แบบกราฟฟิค(GUI) โดยการสร้างโมเดลเป็นบล็อคได้ง่ายโดยใช้เม้าส์ ซิมูลิ้งค์จะมีประเภทของบล็อคต่างได้แก่ ซิ้งค์ (sinks), แหล่งกำเนิด (sources) , ส่วนประกอบเชิงเส้นและไม่เชิงเส้น (linear and nonlinear components) และอุปกรณ์เชื่อมต่อต่างๆ (connectors) และคุณก็สามารถสร้างบล็อคขึ้นมาได้เองอีกด้วย

เมื่อสร้างบล็อคแล้วก็จะสามารถจำลองระบบโดยใช้วิธีต่างๆ ทั้งจากเมนูของซิมูลิ้งค์เอง หรือใส่เข้ามาทางหน้าต่างของแมทแลป และสามารถใช้สโคปหรือบล็อคแสดงผลต่างๆในการดูผลไปพร้อมกับที่ระบบจำลองทำงานอยู่ หรืออาจทำการเปลี่ยนแปลงตัวแปรต่างๆได้ทันที

2.3 เรียลไทม์ เวอร์คชอพ(Realtime Workshop)

มีหน้าที่สร้างโปรแกรมภาษาซี โดยอัตโนมัติให้กับซิมูลิ้งค์ มันจะสร้างรหัสภาษาซีขึ้นมาโดยตรงจากบล็อคไดอะแกรมของซิมูลิ้งค์ และจะสร้างโปรแกรมอัตโนมัติที่สามารถทำงานในเวลาจริงในสิ่งแวดล้อมที่ต่างๆ เช่น การทำงานกับไมโครคอนโทรลเลอร์และตัวประมวลสัญญาณดิจิตอล ซึ่งการทำจะต้องใช้ความรู้ในด้านต่างๆดังนี้คือ

2.4 เรียลไทม์วินโดว์ทาร์เก็ต(Real Time Window Target)

คือซอฟต์แวร์ที่อนุญาตให้ภาษาซีที่ถูกสร้างโดยเรียลไทม์เวอร์คชอพ ทำงานได้ในเวลาจริงบนคอมพิวเตอร์ โดยที่ในคอมพิวเตอร์จะต้องมีโปรแกรมแมทแลป ,ซิมูลิ้งค์ และเรียล-ไทม์เวอร์คชอพการประยุกต์ใช้ทั่วๆไปของเรียลไทม์วินโดว์ทาร์เก็ต ได้แก่ การควบคุมในเวลาจริง ,การประมวลผลสัญญาณและ การจำลองแบบฮาร์ดแวร์อินเดอะลูป (Hardware in The Loop) รหัสภาษาซีที่สร้างขึ้น จะทำงานได้เร็วบน วินโดวส์ 95,98และ เอ็นที (NT) ขณะที่ทำงานกับโมเดลในหมวดเวลาจริง เรียล-ไทม์วินโดว์ทาร์เก็ต จะจับข้อมูลในช่วงเวลาที่กำหนดจากช่องสัญญาณหนึ่งช่องหรือมากกว่านั้น และใช้ข้อมูลที่จับได้นั้นเป็นอินพุทให้กับโมเดลในซิมูลิ้งค์ที่เขียนขึ้น เมื่อทำการประมวลข้อมูลเรียบร้อยแล้ว ข้อมูลจะถูกส่งข้อมูลไปยังภายนอกผ่านทางช่องสัญญาณเอาต์พุทของอินพุทเอาต์พุทบอร์ด เรียล-ไทม์วินโดว์ทาร์เก็ต ใช้ความสามารถที่มีอยู่ใน เรียล-ไทม์ ของ เคอร์เนลพิเศษ สำหรับสนับสนุนแต่ละปฏิบัติการณ์ของวินโดว์ส และสำหรับความเร็วของการคอมไพล์(compile) ซีโปรแกรม มันจะให้บล็อคคำสั่ง ซึ่งมีจำนวนมากกว่า 60 บล็อคคำสั่งที่ใช้ในการติดต่อกับฮาร์ดแวร์ สัญญาณบางทีถูกจับและถูกบันทึกข้อมูลลงในสโคป โดยใช้การติดต่อระบบภายนอกของซิมูลิ้งค์ เพื่อที่จะทำให้คุณสามารถสังเกตพฤติกรรมของระบบ เรียล-ไทม์ ได้

เรียลไทม์วินโดว์ทาร์เก็ต ประกอบด้วยชุดของแหล่งกำเนิด, เลขฐานสองสำหรับไดรเวอร์ไลบรารี่(driver libraries) ของอุปกรณ์อินพุทเอาต์พุท , ตารางที่ใช้ในการสร้างไฟล์ และ เมคส์-ไฟล์ อินเตอร์เฟส (Mex-files interface)หลังจากที่ได้สร้างบล็อคไดอะแกรมโมเดลในซิมูลิ้งค์และใช้ เรียล-ไทม์ เวอร์คชอพ สร้าง เรียลไทม์ โมเดล โดยการคลิกที่ปุ่มบิ้ว(build) ที่หน้าเรียลไทม์ เวอร์คชอพ ของ ไดอะลอกบล็อค ซิมูลิ้งค์พารามิเตอร์ หน่วยของ เมคส์-ไฟล์ อินเตอร์เฟส จะถูกใช้เพื่ออนุญาต ให้โหมดการทำงานภายนอกของซิมูลิ้งค์ ทำการส่งค่าตัวแปรใหม่ให้ เรียลไทม์ โมดูล และทำการนำสัญญาณ เรียลไทม์ โมดูล กลับมา เรายังสามารถแสดงผลสัญญาณนี้ออกทางสโคปของซิมูลิ้งค์ได้

2.5 การเริ่มต้นการปฏิบัติการณ์ realtime

เมื่อการสร้างเรียลไทม์ โมเดลได้สำเร็จแล้วเราสามารถทดสอบได้โดยการทดสอบการทำงานในที่เวลาจริงและดูผลที่เกิดขึ้นจากการปฏิบัติการณ์ เริ่มต้นการปฏิการณ์ rtvdp โมเดล โดยไปเลือกคำสั่ง เอ็กซ์เทอร์นอล(external) ในเมนูของซิมูเลชั่น และเลือก คำสั่ง คอนเน็คท์ทูทาร์เก็ต(Connect to target) และจะเห็นว่าสามารถที่จะเลือกคำสั่งสตาร์ทเรียลไทม์โค้ด (Start realtime code) จากนั้นจะสังเกตเส้นกราฟปรากฏขึ้นที่สโคปของซิมูลิ้งค์ ค่อยๆเคลื่อนที่สอดคล้องกับเวลาจริง หลังจากนั้นทำการทดลองนี้อีกครั้งโดยการทดสอบการทำงานของโมเดลนี้ขึ้นตรงกับซิมูลิ้งค์โดยไม่ใช้เรียลไทม์วินโดว์ทาร์เก็ต จะเห็นว่าการทำงานจะไม่เป็นไปตามเวลาจริง

2.6 เอส-ฟังก์ชั่น(S-Functions) คืออะไร

เอสฟังก์ชั่นเป็นการบรรยายภาษาคอมพิวเตอร์ ของระบบไดนามิค ซึ่งสามารถถูกเขียนโดยใช้ แมทแลปหรือภาษาเอสฟังก์ชั่นของซี(C language S-function) จะถูกคอมไพล์ขณะที่เม็ค-ไฟล์กำลังใช้เม็คยูทิลิตี้(Mex utility) (ถูกบรรยายอยู่ใน The Application Program Interface Guide ในขณะที่เม็คไฟล์ อื่นๆจะติดต่ออยู่ตลอดเวลาในแมทแลป เมื่อจำเป็น

เอส-ฟังก์ชั่น จะใช้ซินแทกซ์พิเศษ เพื่อที่คุณสามารถติดต่อกับสมการการแก้ปัญหาของซิมูลิ้งค์ (Simulink's Equation solvers) การติดต่อแบบนี้จะคล้ายคลึงกันมากกับ การติดต่อระหว่างผู้แก้ปัญหาและการสร้าง ซิมูลิ้งค์บล็อค

รูปแบบของเอสฟังก์ชั่น จะเป็นรูปแบบง่ายมาก และสามารถใช้กับระบบต่อเนื่อง,ไม่ต่อเนื่องและไฮบริด เป็นสาเหตุให้ ซิมูลิ้งค์โมเดล ทั้งหมดมีความใกล้เคียงกับ เอสฟังก์ชั่น

เอส-ฟังก์ชั่น จะถูกใช้เสมือนเป็นโมเดลหนึ่งในซิมูลิ้งค์โดยสามารถเรียกขึ้นมาได้จาก เอส-ฟังก์ชั่น บล็อค ในไลบรารี่ย่อย (sublibrary) การใช้เอส-ฟังก์ชั่น บล็อค ของไดอะลอกบอกซ์เพื่อที่จะระบุชื่อของช่องว่างของเอส-ฟังกชั่นที่เว้นไว้ ซึ่งอธิบายตามภาพด้านล่าง

ในตัวอย่างนี้โมเดล จะประกอบด้วย 2 เอสฟังก์ชั่นบล็อค ซึ่งทั้งคู่จะอ้างอิงในแหล่งกำเนิดไฟล์ เดียวกัน (mysfun ,สามารถเป็นได้ทั้งซีเม็คไฟล์ และเอ็มไฟล์) ถ้าทั้ง ซีเม็คไฟล์ และเอ็มไฟล์ ปรากฏด้วยชื่อเดียวกันซีเม็คไฟล์ จะเป็นส่วนที่ถูกเรียกใช้

คุณสามารถใช้แมสกิ้ง(masking) ของซิมูลิ้งค์ อำนวยความสะดวก เพื่อที่จะสร้างไดอะลอกบอกซ์ และ ไอคอน(Icons) สำหรับ เอสฟังก์ชั่นบล็อค ของคุณ และ แมสไดอะลอกบอกซ์ สามารถที่จะระบุการเพิ่มด้วย พารามิเตอร์ สำหรับเอส-ฟังก์ชั่น การบรรยายของการเพิ่ม พารามิเตอร์และแมสกิ้ง ดูใน Using Simulink

2.7 เมื่อไหร่ที่จะใช้ S-Functions

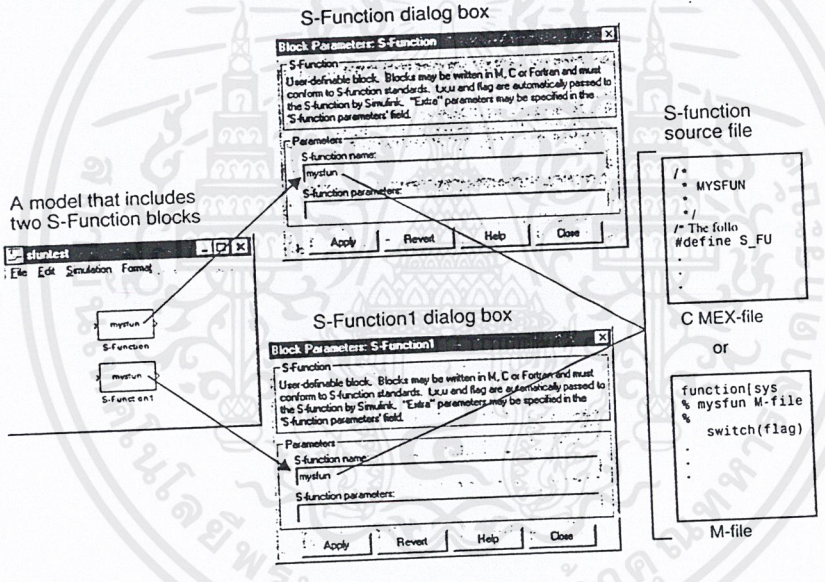
เป็นการง่ายที่สุดของการใช้ เอส-ฟังก์ชั่น สร้างซิมูลิ้งค์บล็อค คุณสามารถใช้ เอส-ฟังก์ชั่น สำหรับการใช้ที่หลากหลาย รวมถึง

- การเพิ่มบล็อคที่มีจุดประสงค์ใหม่ไปยัง ซิมูลิ้งค์
- การร่วมมือกกับซีโปรแกรม ในการซิมูเลชั่น
- อธิบายระบบที่เป็นรูปแบบสมการทางคณิตศาสตร์
- ใช้ กราฟฟิค เอนิเมชั่น(Graphical animations)

ข้อได้เปรียบของการใช้ เอส-ฟังก์ชั่น ที่ว่า คุณสามารถสร้างบล็อคจุดประสงค์ทั่วไป และคุณสามารถใช้หลายๆ ครั้ง ใน 1 โมเดล ในการเปลี่ยนแปลงค่าต่างๆในส่วนของแต่ละบล็อค

2.8 ใช้เอส-ฟังก์ชั่นได้อย่างไร

แต่ละบล็อคในซิมูลิ้งค์ โมเดลจะเป็นไปตามลักษณะทั่วๆไปของ เวคเตอร์อินพุต u, เวคเตอร์เอาต์พุท yและเวคเตอร์ของสถานะต่างๆx แสดงตามภาพ



รูปที่ 2-1 แสดงความสัมพันธ์ระหว่าง เอสฟังก์ชั่นบล็อค ,ไดอะลอกบอกซ์,และซอสไฟล์ที่ให้คำจำกัดความของพฤติกรรมของบล็อค

2.9 ซีเมค เอสฟังก์ชั่น (C-Mex S-Function)

ในเอ็มไฟล์ เอสฟังก์ชั่น,ซิมูลิ้งค์ประกอบด้วยเวคเตอร์สถานะใน 2 ส่วนคือ สถานะต่อเนื่องและไม่ต่อเนื่อง สถานะต่อเนื่องจะจับจองในส่วนแรกของเวคเตอร์และไม่ต่อเนื่องจะจับจองส่วนที่2 ส่วนบล็อคที่ไม่มีสถานะ ก็คือ x ที่เป็นเวคเตอร์ว่างเปล่า

สถานะการจำลองและ เอสฟังก์ชั่นรูทีน (Simulation stages & S-Functions Routines)

ซิมูลิ้งค์จะทำการเรียกซึ่งระหว่างการเจาะจงสถานะของซิมูเลชั่นในแต่ละบล็อคในโมเดล ไปยังมันโดยตรงเพื่อจะปฏิบัติหน้าที่ เช่น คำนวณเอาต์พุทของมัน อัพเดทค่าของระบบไม่ต่อเนื่อง หรือคำนวณอนุพันธ์(deriavatives) ส่วนการเรียกเพิ่มเติมคือคำสั่งการเริ่มต้นและจบของการซิมูเลชั่น เพื่อต้องปฏิบัติหน้าที่เริ่มต้นและสิ้นสุด

รูปด้านร่างเป็นการแสดงว่าซิมูลิ้งค์ จะทำการซิมูเลชั่นอย่างไร อย่างแรกซิมูลิ้งค์จะทำการเริ่มต้นโมเดล(การเริ่มต้นแต่ละบล็อค รวมถึงเอส-ฟังก์ชั่น) และซิมูเลชั่นลูปที่ซึ่งแต่ละเส้นทางจะผ่านตามลูปซึ่งจะอ้างถึง ซิมูเลชั่น สเตป ระหว่างแต่ละซิมูเลชั่นสเตป ,ซิมูลิ้งค์จะปฏิบัติการณ์ เอส-ฟังก์ชั่น บล็อคของคุณอย่างต่อเนื่องจนกระทั่งการ ซิมูเลชั่น เรียบร้อย

ซิมูลิ้งค์ จะทำการเรียกซ้ำไปยัง เอสฟังก์ชั่น ในโมเดลของคุณ ในระหว่างการเรียกนี้ ซิมูลิ้งค์เรียกเอสฟังก์ชั่น รูทีน ,ปฏิบัติตามหน้าที่ที่ถูกต้องการแต่ละสถานะหน้าที่รวมถึง

- Initialization ก่อเริ่มลูปแรกของการซิมูเลชั่น ซิมูลิ้งค์จะทำการเริ่มต้นเอสฟังก์ชั่น ระหว่างสถานะนี้ ซิมูลิ้งค์

-Initializes the Simstruct (a simulation structure)– ซึ่งประกอบด้วยข้อมูลเกี่ยวกับเอสฟังก์ชั่น

-เซตตัวเลขและขนาดของ อินพุทและเอาต์พุทพอร์ต

-เซตของบล็อคการสุ่มค่า

-จัดพื้นที่การจัดเก็บและขนาดของอาร์เรย์

-ทำการคำนวณของค่า next sample hit

-ทำการคำนวณเอาต์พุทในช่วงของเวลาหลัก หลังจากการเรียกนี้สมบูรณ์เอาต์พุทพอร์ตทั้งของบล็อคจะมีค่าของช่วงเวลาในปัจจุบัน

-Update สถานะไม่ต่อเนื่องในช่วงเวลาหลัก ในการเรียกนี้จะปฏิบัติการณ์ 1 ครั้งช่วงเวลาการทำ

-Integration จะใช้กับโมเดลที่ต่อเนื่องและ/หรือ nonsampled zero crossings ถ้าเอสฟังก์ชั่น ของคุณแป็นตัวเนื่อง ซิมูลิ้งค์จะเรียกส่วนของเอาต์พุทและอนุพันธ์ของเอสฟังก์ชั่น

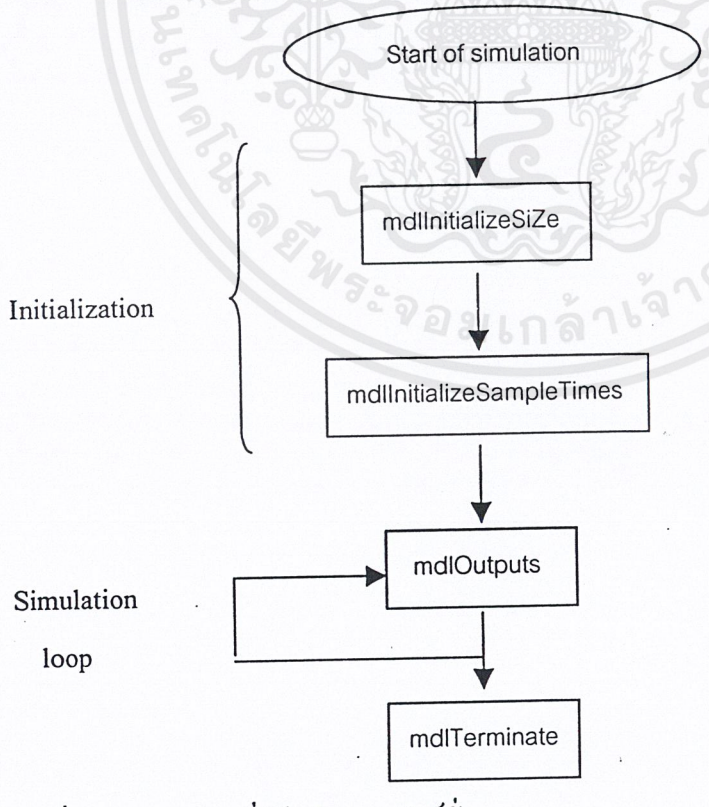
ของคุณที่ไมเนอร์ไทม์เสตป ซึ่งซิมูลิ้งค์สามารถคำนวณสถานะของเอสฟังก์ชั่น

ซีเมคไฟล์จะให้ข้อมูลเกี่ยวกับโมเดลไปยังซิมูลิ้งค์ระหว่างซิมูเลชั่น ในขณะที่ดำเนินการซิมูเลชั่น ซิมูลิ้งค์ ODE SOLVER และเมคไฟล่จะติดต่อกันเพื่อจะทำงานตามหน้าที่ที่เจาะจงไว้ หน้าที่นี้ได้แก่การนิยามค่าเริ่มต้นและบล็อคคาแรกเตอร์ริสติคส์ และคำนวณอนุพันธ์,สถานะไม่ต่อเนื่อง, เอาต์พุท

ซีเมคไฟล์ของเอสฟังก์ชั่นจะมีโครงสร้างและการปฏิบัติการณ์เหมือนกับเอ็มไฟล์ แต่จะต่างกันในเรื่องของการปฏิบัติการณ์

2.9.1 การเขียน ซีเม็คเอสฟังก์ชั่นพื้นฐาน

ในส่วนจะกล่าวถึงการเขียนซีเมคเอสฟังก์ชั่นพื้นฐาน ซึ่งจะเป็นส่วนที่ต้องการในเอสฟังก์ชั่นรูทีน แต่อย่างไรก็ตามผู้ใช้สามารถทำให้โปรแกรมซับซ้อนอย่างไรก็ได้ ในที่นี้จะขอกล่าวถึงโปรแกรม timestwo (code อยู่ในภาคผนวก) ซึ่งโปรแกรม timestwo ของเอสฟังก์ชั่นจะมีส่วนที่เอสฟังก์ชั่นต้องการตามรูปด้านล่างนี้



รูปที่ 2-2 แสดงส่วนที่ใช้อยู่ในเอสฟังก์ชั่น

เพื่อที่จะใช้เอสฟังก์ชั่นในซิมูลิ้งค์ จะต้องสร้าง source file เพื่อที่จะเรียกใช้ timestwo.c โดย

พิมพ์

mex timestwo.c

ในที่นี้จะขอกล่าวถึงรูทีนเอสฟังก์ชั่น ระหว่างการซิมูเลชั่นของ timestwo

- mdlInitializeSizes ซิมูลิ้งค์จะเรียกรูทีนนี้ขณะที่อีดิตโมเดล เพื่อที่จะกำหนดจำนวนของอินพุทและเอาต์พุทพอร์ต ซิมูลิ้งค์จะเรียกมันขณะที่เริ่มต้นการซิมูเลชั่นเพื่อ ร้องขอขนาดของพอร์ตและวัตถุประสงค์อื่น
- mdlInitializeSampletimes ซิมูลิ้งค์จะเรียกรูทีนนี้ เพื่อกำหนดขนาดของการสุ่มของเอสฟังก์ชั่นุ
- mdlOutputs เป็นการคำนวณของค่าเอาต์พุท mdlOutputs จะรับค่าอินพุทและจะคูณมันด้วย2 สำหรับโปรแกรมtimestwo รูทีนนี้จะถูกเรียกขณะที่มีการซิมูเลชั่นลูปแต่ละเวลา ซึ่งจะต้องมีการอัพเดทในtimestwo
- mdlterminate จะแสดงในส่วนท้ายสุดของโปรแกรมแต่ใน timestwo จะไม่มีการแสดงของหน้าที่นี้เนื่องจากรูทีนนี้ว่างเปล่า

สำหรับข้อมูลเพิ่มเติมของ timestwo จะต้องการ

- การนิยามอันได้แก่ ชื่อของเอสฟังก์ชั่น(timestwo) และเอสฟังก์ชั่นจะต้องอยู่ในการจัดรูปแบบ level2 หลังจากนิยามในสองหัวข้อนี้แล้วจะยกตัวอย่าง simstruct.h ซึ่งจะเรียกว่าส่วน เฮดไฟล์ (head file) ซึ่งใช้ในการเข้าถึง SimStruct และ API ของ แมทแลป :

  #define S_FUNCTION_NAME timestwo

  #define S_FUNCTION_LEVEL 2

  #include “simstruct.h”

- mdlInitializeSizes แสดงได้โดยตัวอย่างโปรแกรม timestwo
  - Zero paraters หมายถึง เอสฟังก์ชั่นพารามิเตอร์ ของไดอะลอกบอกซ์จะต้องว่างถ้ามันมีตัวแปรซิมุลิ้งค์อยู่จะแสดงผลมาว่าเกิดการไม่เข้ากัน (mismatch)
  - หนึ่งอินพุทและหนึ่งเอาต์พุท ความกว้างของพอร์ตอินพุทและเอาต์พุทสามารถเปลี่ยนแปลงขนาดได้ สิ่งนี้บอกด้วยว่าซิมูลิ้งค์ใช้คุณแต่ละอุปกรณ์แต่ละสัญญาณอินพุทกับเอสฟังก์ชั่นโดยคูณกับ2 และ ใส่ผลลัพธ์ลงในสัญญาณเอาต์พุท การที่ขนาดของมันเปลี่ยนแปลงได้ จะทำให้เอสฟังก์ชั่นของกรณีนี้ มีความกว้างของอินพุทและเอาต์พุทเท่ากัน
  - หนึ่งการสุ่มเวลา ควรจะต้องระบุค่าของ sample time ใน mdlInitializeSampletime

- รหัสที่ถูกยกเว้นการระบุโค้ดนี้จะ ทำให้การจัดการของเอสฟังก์ชั่นคุณเร็วขึ้นจะต้องสนใจในเรื่องของการระบุใน option ด้วย โดยทั่วไปแล้วค่าเอสฟังก์ชั่นของคุณไม่สามารถติดต่อกับแมทแลปได้ คุณควรจะระบุใน option นี้

- mdlInitializeSampletime

  - จะถูกยับยั้งจากบล็อคผลักดันหมายถึงเอสฟังก์ชั่นจะทำงานได้เมื่อได้รับอินพุทจากบล็อคที่ติดต่อกับอินพุทพอร์ตของบล็อคเอสฟังก์ชั่น

-mdloutput

-การคำนวนตัวเลข mdloutput ได้บอกกับสัญญาณซิมูลิ้งค์ คูณสัญญาณอินพุทด้วยสอง

-การเข้าถึงสัญญาณอินพุท ใช้

InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(s,0); ซึ่ง Uptrs เป็นเวคเตอร์ตัวหนึ่งของพอยเตอร์จะต้องถูกใช้ในรูป uPtrs(i)

-การเข้าถึงสัญญาณเอาต์พุทใช้

real_T*Y=ssGetOutputPortRealSignalPtrs(s,0);

-ลูปของเอสฟังก์ชั่นจะกว้างกว่าความกว้างของสัญญาณที่กำลังผ่านเข้าไปในบล็อคเพื่อที่จะ เข้าถึงบล็อคสัญญาณคุณสามารถตรวจสอบพอร์ตความกว้างพอร์ตอินพุทและพอร์ตเอาต์พุทได้ ซึ่งความกว้างของเอาต์พุทสามารถหาค่าได้

- mdlterminate

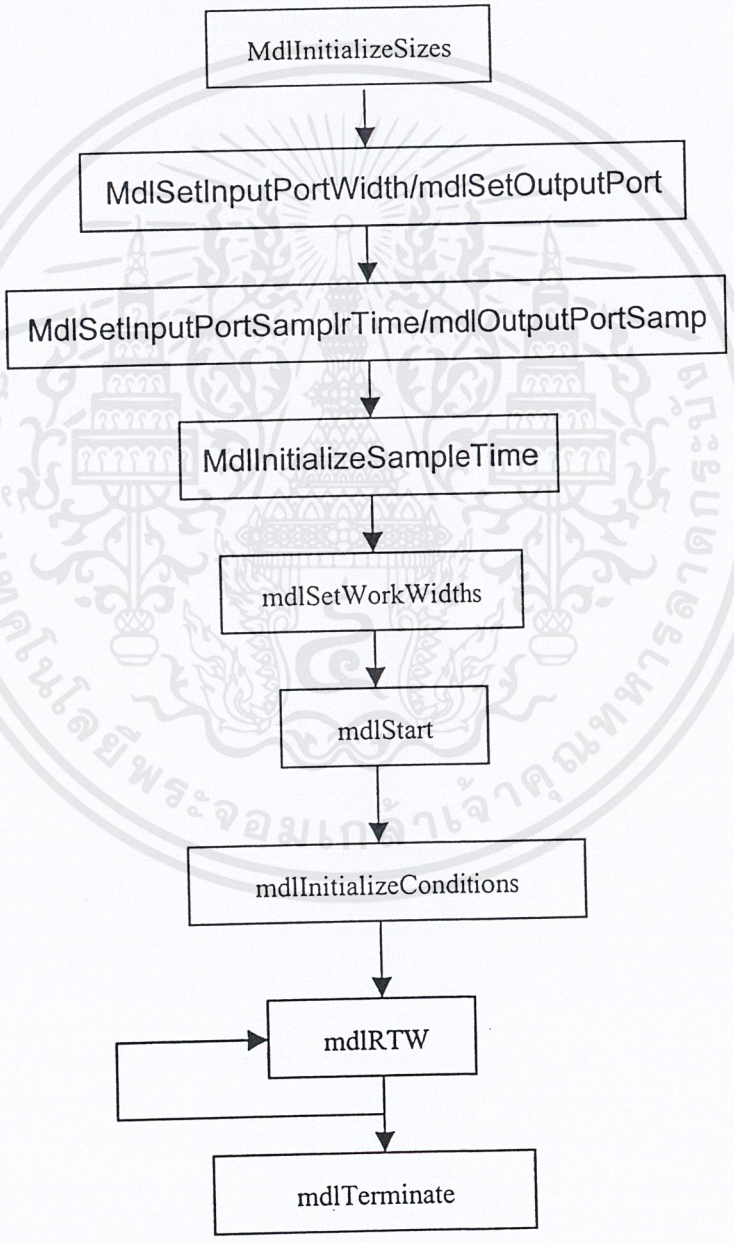
  - ซึ่งเป็นเอสฟังก์ชั่นรูทีน อย่างไรก็ตาม timestwo ฟังก์ชั่นไม่จำเป็นต้องแสดงดังนั้นรูทีนนี้จึงว่าง

สุดท้ายของเอสฟังก์ชั่นจะบอกรหัสที่ติดต่อกับตัวอย่างเพื่อเข้าสู่ซิมูลิ้งค์และเรียลไทม์เวอร์คชอพ

```
#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

2.9.2โครงสร้างของการเรียกแบบเปลี่ยนได้สำหรับโหมดภายนอก

เมื่อรันโปรแกรมซิมูลิงค์ในโหมดภายนอก ลำดับการเรียกของเอสฟังก์ชันรูทีนจะเปลี่ยนแปลง รูปข้างล่างนี้แสดงลำดับที่ถูกต้องของโหมดภายนอก



รูปที่ 2-3 ลำดับการเรียกแบบเอสฟังก์ชันเมื่อมีการรันซิมูลิงค์ในโหมดภายนอก

ซิมูลิงค์ที่เรียกว่า mdlRTW เมื่อเข้าสู่โหมดภายนอก การเข้าสู่โหมดภายนอกแต่ละครั้ง หรือเมื่อคุณเลือก**อัพเดต ไดอะแกรม** ภายใต้ Edit เมนูของรูปแบบของคุณ จะทำให้พารามิเตอร์เปลี่ยนไป

2.9.3การแสดงข้อมูลของเอสฟังก์ชัน

บล็อกของเอสฟังก์ชันมีสัญญาณอินพุทและสัญญาณเอาท์พุท , พารามิเตอร์ , สถานะภายในบวกกับพื้นที่ทำงานทั่วไปอื่นๆ โดยทั่วไป บล็อกอินพุทและเอาท์พุทจะถูกเขียนไปยัง และถูกอ่านจาก I/O เวคเตอร์บล็อก อินพุทต่างๆสามารถมาจาก

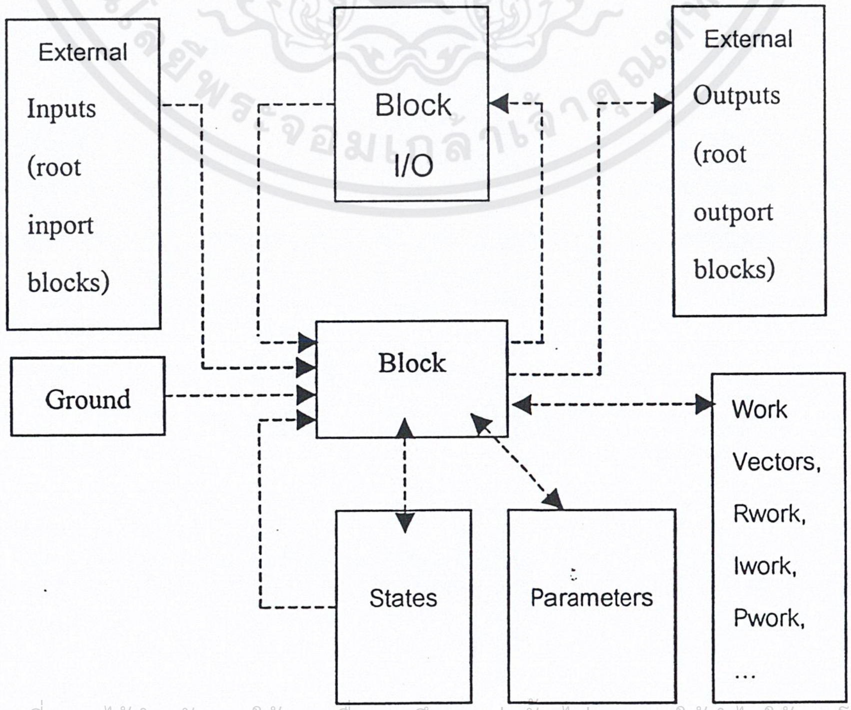
- อินพุทภายนอกโดยผ่านทางบล็อกของรากของพอร์ทภายใน
- กราวนด์ ถ้าสัญญาณอินพุทไม่ได้ถูกเชื่อมต่อ กรือถูกต่อลงกราวนด์

บล็อกเอาท์พุทสามารถไปถึงเอาท์พุทภายนอกได้โดยผ่านทางบล็อกของรากของพอร์ทภายนอก สิ่งที่เพิ่มเติมคือ สัญญาณอินพุทและสัญญาณเอาท์พุท เอสฟังก์ชันสามารถมี :

- สถานะที่ต่อเนื่อง
- สถานะที่ไม่ต่อเนื่อง
- พื้นที่ทำงานอื่นๆ เช่น ค่าจริง , เลขจำนวนเต็ม , หรือเวคเตอร์ตัวชี้

บล็อกของเอสฟังก์ชันสามารถถูกทำเป็นพารามิเตอร์โดยการผ่านพารามิเตอร์ไปยังบล็อกเหล่านั้น และการใช้ไดอะลอกบอกซ์ของบล็อกของเอสฟังก์ชัน

รูปข้างล่างนี้แสดงแมปปิ้งทั่วไประหว่างข้อมูลแบบต่างๆกัน



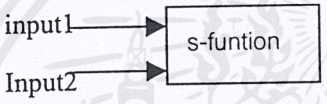
รูปที่ 2-4 แสดงความสัมพันธ์ระหว่างเอสฟังก์ชั่นและข้อมูลของเอสฟังก์ชั่น

ความยาวของสัญญาณต่างและเวคเตอร์จะมีลักษณะ mdlInitializeSizes รูทีนและความยาวของมันสามารถเข้าไปในรูทีนเอสฟังก์ชั่นจะถูกเรียกระหว่างลูปซิมูเลชั่น

ระหว่างซิมูเลชั่นลูป การเข้าถึงสัญญาณอินพุทจะถูกแสดงดังต่อไปนี้

= ssGetInputPortRealSignalPtrs InputRealPtrsType uPtrs (s,portindex);

สิ่งนี้คืออาร์เรย์ของพอยเตอร์ซึ่งพอร์ตอินเดคเริ่มที่0 จะมีหนึ่งอันสำหรับพอร์ตอินพุทเท่านั้นการเข้าถึงอุปกรณ์ของสัญญาณนี้คุณจะต้องใช้ uPtrs (element) สามารถแสดงให้เห็นโดย



รูปที่ 2-5 แสดงถึงการเข้าถึงสัญญาณ

2.9.4 การเข้าสัญญาณอินพุทของพอร์ต

ในส่วนนี้กล่าวถึงจะทำอย่างไรจึงจะเข้าถึงสัญญาณอินพุททั้งหมดของพอร์ต และเขียนมันลงในเอาต์พุทพอร์ต สัญญาณเอาต์พุทของพอร์ตอยู่ในรูปแบบฟอร์มของเวคเตอร์ดังนั้นวิธีที่ถูกต้องในการเข้าถึงอุปกรณ์อินพุทและเขียนมันสู่สัญญาณเอาต์พุท (สมมติว่าพอร์ตอินพุทและพอร์ตเอาต์มีความกว้างเท่ากัน) จะต้องใช้โค้ดเหล่านี้คือ

int_T element;

int_T portwidth = ssGetInputPortWidth(s,input PortIndex);

ssGetInputPortRealSignalPtrs InputRealPtrsType uPtrs (s,portindex);

real_T*Y=ssGetOutputPortRealSignalPtrs(s,outputPortIdx)

ความผิดพลาดโดยทั่วไป จะพยายามและเข้าถึงสัญญาณอินพุททางพอยเตอร์ ตัวอย่างเช่น

real_T*U = *uPtrs ซึ่งแบบนี้จะผิด

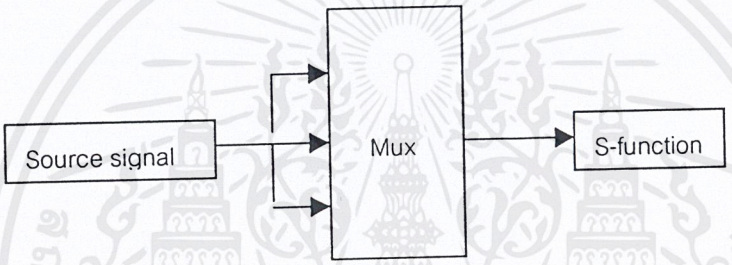
หลังจากเริ่มต้นของ uptrs และการแทนเข้าไปในส่วนภายในของลูป

- y++ = * u++ ซึ่งแบบนี้ก็จะผิด

การแปลงรหัสสำหรับเมดไฟล์ อาจจะชนกับซิมูลิ้งค์ได้ อันเนื่องมาจากการเข้าไปยังหน่วยความจำที่สูญหายได้ซึ่งจะขึ้นอยู่กับว่าคุณได้สร้างรูปแบบของคุณอย่างไร การเข้าถึงสัญญาณอินพุทเกิดการชนกันอย่างผิดปกติ จะเกิดเมื่อสัญญาณที่เข้า เอสฟังก์ชั่นของคุณไม่ติดกัน สัญญาณข้อมูลที่

ไม่ติดกันนี้จะเกิดเมื่อสัญญาณผ่านเข้าสู่ บล็อคที่ติดต่อกันแบบเห็นได้จริง เช่นบล็อคของซีเล็คเตอร์ หรือมัลติเพลกเซอร์

เพื่อพิสูจน์ว่าคุณกำลังเข้าสู่สัญญาณอินพุทที่ถูกต้องสัญญาณจะลอกเลียนแบบไปยังพอร์ตอินพุทของแต่ละอันของเอสฟังก์ชั่นของคุณเมื่อขั้นตอนนี้ถูกกระทำเสร็จโดยการสร้าง mux บล็อคกับจำนวนพอร์ตอินพุท = ความกว้างของสัญญาณที่ต้องการ ที่กำลังเข้าสู่ เอสฟังก์ชั่นของคุณ แหล่งขับเคลื่อนควรจะติดกับอินพุทพอร์ตแต่ละอันแสดงได้ดังภาพ



รูปที่ 2-6 แสดงถึงการเข้าถึงสัญญาณอินพุท

2.9.5 การตรวจสอบและการประมวลตัวแปรเอสฟังก์ชั่น

คุณสามารถให้พารามิเตอร์ยังเอสฟังก์ชั่น ซึ่งสามารถถูกเปลี่ยนโดยใช้การติดต่อกับเอสฟังก์ชั่นพารามิเตอร์ ของ ไดอะลอกบอกซ์ ถ้าคุณนิยามพารามิเตอร์ ข้างล่างเป็นขั้นตอนเมื่อคุณจะสร้างเอสฟังก์ชั่น

1. กำหนดคำสั่ง ซึ่งพารามิเตอร์จะระบุในไดอะลอกบอกซ์
2. ใน mdlInitializeSizes ฟังก์ชั่น ใช้ ssSetNumSFcnParams สำหรับบอกซิมูลิ้งค์ถึงจำนวนตัวแปรที่ถูกผ่านเข้าไปในเอสฟังก์ชั่น ระบุเอสเป็นเหมือน อาร์กิวเมนต์ตัวแรก และจำนวนของตัวแปรที่คุณจำกัดความให้เป็นอาร์กิวเมนต์ตัวที่สอง
3. การเข้าถึงอาร์กิวเมนต์อินพุทเหล่านี้ในเอสฟังก์ชั่นที่กำลังใช้ ssGetSFcnParam macro ระบุเอสเป็นอาร์กิวเมนต์แรกและตำแหน่งของตัวแปรในรายการที่ถูกเข้าไปยังไดอะลอกบอกซ์ (จะเป็นตำแหน่งแรก) เปรียบเสมือนอาร์กิวเมนต์ที่สอง

เมื่อคุณทำการซิมูเลชั่นระบุชื่อตัวแปรหรือค่าในตัวแปรเอสฟังก์ชั่นของไดอะลอกบอกซ์ ลำดับของชื่อและค่าตัวแปรจะเหมือนกัน เพื่อที่จะเรียงลำดับตามที่คุณกำหนดไว้ในขั้นตอนที่1 ถ้าคุณกำหนดชื่อ คุณไม่จำเป็นต้องใช้ชื่อเดียวกันในเมดไฟล์

2.9.6 การใช้เอสฟังก์ชั่นกับเรียลไทม์เวอร์ชอพ

โดยทั่วไปคุณสามารถใช้เอสฟังก์ชั่นในเรียลไทม์เวอร์ชอพ อย่างไรก็ตามในบางกรณีต้องการการปรับปรุงเหล่านี้เพื่อเอสฟังก์ชั่น

-การกำหนดชื่อของรูปแบบพิเศษถูกใช้ในการสร้างเอสฟังก์ชั่นของคุณ

-การกำหนด RTWdata สำหรับเอสฟังก์ชั่น

-การเพิ่ม mdlRTW ฟังก์ชั่น

2.9.7 ชื่อโมดูลสำหรับ RTW Build's

ถ้าเอสฟังก์ชั่นของคุณถูกสร้างด้วยโมดูลจำนวนมากคุณจำเป็นต้องแบ่งการสร้างชื่อกระบวนการของโมดูลที่เพิ่มเข้าไป คุณจะสามารถทำขั้นตอนนี้ได้โดยคุณต้องผ่านเรียลไทม์เวอร์คชอพ เทมเพลตเมคไฟล์หรือเพื่อให้สะดวกมากยิ่งขึ้นให้ใช้ set_param MATLAB command สำหรับตัวอย่างถ้าเอสฟังชั่นของคุณใช้หลายโมดูลต้องทำดังต่อไปนี้

```
mex sfun_main.c Sfun_module1.c Sfun_module2.c
```

ต่อมาระบุชื่อของโมดูล โดยไม่ต้องใช้คำสั่ง

```
set_param(sfun_block,-'SfunctionModule','sfun_module1 sfun_module2')
```

ตัวแปรสามารถเปลี่ยนแปลงได้

```
module=('sfun_module1 sfun_module2'
set_param(sfun_block,-'SfunctionModule','modules')
```

2.9.8 เอสฟังก์ชั่น RTWdata สำหรับการสร้างรหัสด้วยRTW

มีคุณสมบัติของบล็อคอย่างหนึ่งถูกเรียกว่า RTWdata ซึ่งสามารถถูกใช้โดยทีแอลซี เมื่อทำเอสฟังก์ชั่น RTWdata จะเป็นโครงสร้างหนึ่งของตัวอักษร ซึ่งคุณสามารถติดต่อได้กับบล็อค มันจะถูกบันทึกและแทนที่ในโหมดmodel.rtw เมื่อสร้างรหัสขึ้นสำหรับตัวอย่างคำสั่งกลุ่มนี้ของแมทแลปคือ

```
mydata.field1 ='information for field1';
mydata.field2 ='information for field2';
set_param(gcb,'RTWdata',mydata)
Get_param(gcb'RTWdata')
```

ได้ผลลัพธ์ดังนี้

```
Ans=field1: 'information for field1'
```

Field2:'information for field2'

ใน model.rtw สำหรับ บล็อคเอสฟังก์ชั่นจะมีข้อมูลเหล่านี้คือ

```
Block {
        Type            "S-Function"
        RTWdata{
          field1              "information for field1"
          field2              "information for field2"
mdlRTW
```

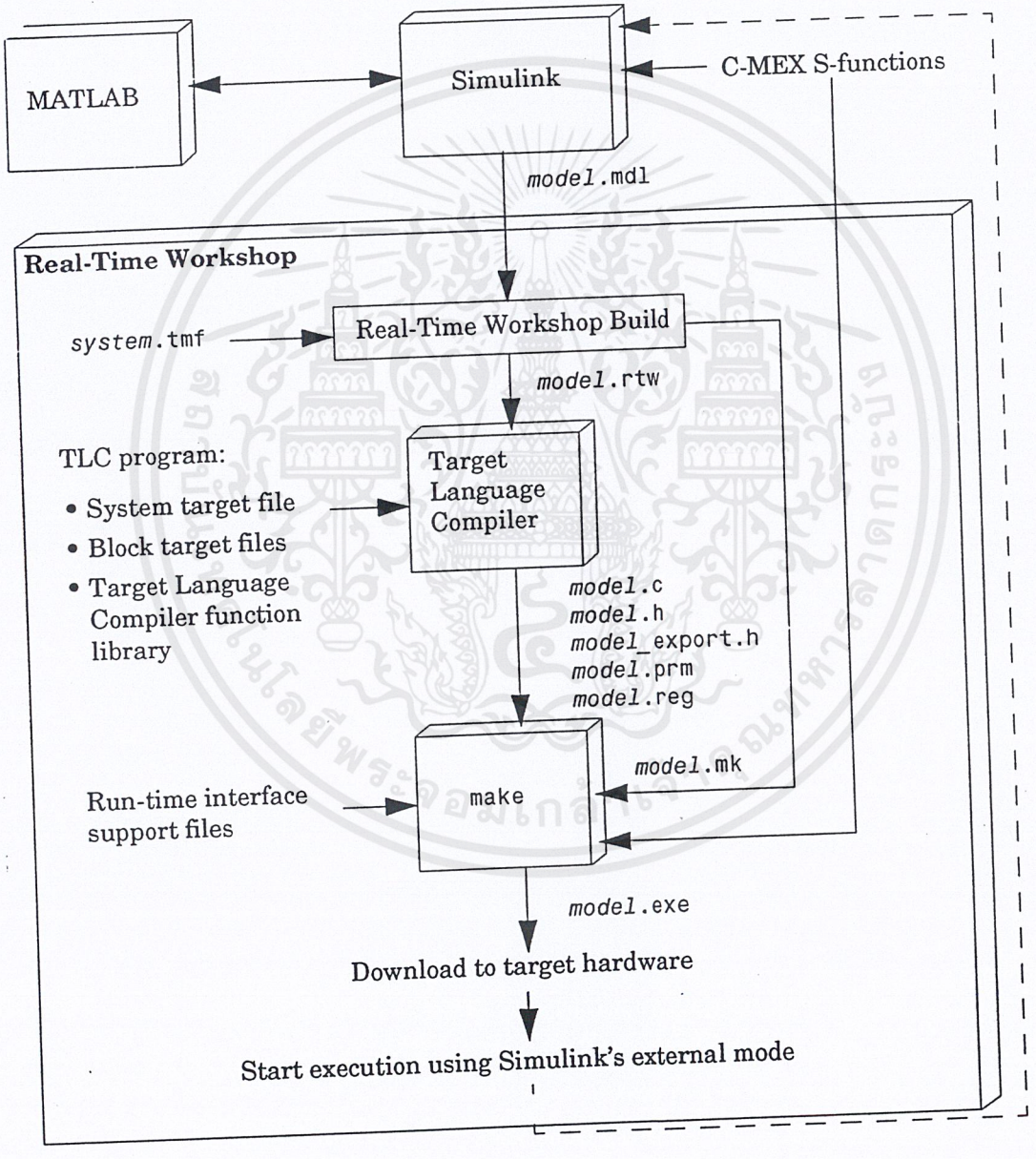
mdlRTW รูทีนจะช่วยย่อยเอสฟังก์ชั่นของคุณในรหัสที่สร้างขึ้น การย่อในรหัสของคุณในเรียลไทม์เวอร์คชอพโดยทั่วไปเมื่อเสร็จขั้นตอนนี้สำหรับการแสดงผลคุณอาจจะถูกต้องการอินลายฟังก์ชั่นของคุณอีกครั้งหนึ่งถ้าคุณมี mdlProcessParameters รูทีน หรือถ้าเอสฟังก์ชั่นคุณมี'ซิมูเลชั่นโหมด, และเรียลไทม์โหมดเช่น ฮาร์ดแวร์ I/O เอสฟังก์ชั่น ซึ่งประมวลอุปกรณ์อินพุทเอาต์พุทในซิมูลิ้งค์และติดต่อด้วยอุปกรณ์อินพุทเอาต์พุทในเรียลไทม์

2.10 วิธีการใช้เรียลไทม์เวอร์คชอพ

เรียลไทม์เวอร์คชอพ(Realtime Workshop) ใช้กับ แมทแลปและซิมูลิ้งค์ ผลิตโค้ดโดยตรงจากซิมูลิ้งค์โมเดลและสร้างโปรแกรมโดยอัตโนมัติ ซึ่งสามารถทำงานในหลายรูปแบบได้แก่ระบบเวลาจริงและการจำลองแบบสแตนอโลน(stand alone simulation) โดยสามารถทำงานได้ในที่ความเร็วสูงบนเครื่องจักรหลักและกับคอมพิวเตอร์ภายนอก

การประยุกต์ใช้งานของเรียลไทม์เวอร์คชอพ

- การควบคุมที่เวลาจริง (realtime control) เราสามารถออกแบบระบบควบคุมโดยแมทและซิมูลิ้งค์ และสร้างโค้ดจากบล็อคไดอะแกรมโมเดล คอมไพล์และดาวน์โหลดโค้ดนี้เข้าสู่ฮาร์ดแวร์เป้าหมายได้โดยตรง
- การประมวลผลสัญญาณที่เวลาจริง เราสามารถประมวลผลสัญญาณที่สร้างโดยแมทแลปและซิมูลิ้งค์ การสร้างโค้ดจากบล็อคไดอะแกรมของคุณสามารถคอมไพล์และดาวน์โหลดไปยังฮาร์ดแวร์เป้าหมายได้
- การจำลองแบบฮาร์ดแวร์ อินเดอะลูป (Hardware in the loop) คุณสามารถสร้างซิมูลิ้งค์โมเดล ซึ่งใช้ในระบบจริงๆหรือระบบไดนามิค และสัญญาณจริงเพื่อใช้ในการติดต่อเปลี่ยนแปลงที่เวลาจริง



รูปที่ 2-7 สถาปัตยกรรมของเรียลไทม์เวอร์คชอพ

2.10.1 โค้ดที่ถูกสร้างขึ้น

ถูกสร้างจากซิมูลิ้งค์โมเดลใดๆใน ไม่ว่าเป็นระบบต่อเนื่อง,ไม่ต่อเนื่องและไฮบริดก็ตาม มีลักษณะเป็นรหัสทางภาษาซี ซิมูลิ้งค์บล็อคทั้งหมดถูกแปลงเป็นโปรแกรมโดยอัตโนมัติ โดยเราต้องเขียนบล็อคนี้เป็นซีเม็ค เอส-ฟังก์ชั่นเป็นแบบ ถ้าให้ต้องการจะทำงานกับเรียลไทม์เวอร์คชอพ

เรียลไทม์เวอร์คชอพ ประกอบด้วยชุดของไฟล์เป้าหมายซึ่งถูกคอมไพล์โดยทาร์เก็ต แลงเกวจคอมไพเลอร์ (ทีแอลซี) และกลายเป็น ANSI C CODE โดยก่อนหน้านี้ไฟล์เป้าหมายเป็นแอสกี้ที่อธิบายว่าเราจะแปลงโมเดลของซิมูลิ้งค์เป็นโค้ด ได้อย่างไร เราสามารถรวมซีเม็คเอสฟังก์ชั่นรวมกับ โค้ดที่ถูกสร้างขึ้น ไว้ในโปรแกรมทำงาน และเรายังสามารถเขียนทาร์เก็ตไฟล์ให้กับซีเม็คเอสฟังก์ชั่นเพื่อที่จะ อินลาย(inline) เอส-ฟังก์ชั่นนั้น

2.10.2 ทีแอลซี (Target language Compiler)

ในการสร้างโค้ดจำเป็นต้องมีทีแอลซีเพื่อใช้เป็นตัวแปลงคำอธิบายโมเดลที่สร้างโดยเรียลไทม์เวอร์คชอพให้กลายเป็นโค้ดที่มีเป้าหมายเฉพาะและคำอธิบายโมเดลนั้นจะถูกบันทึกในไฟล์แอสกี้โดยมีชื่อว่า model.rtw คอมไพเลอร์จะอ่านไฟล์ model.rtw ขึ้นมา และปฏิบัติการณ์โปรแกรมทีแอลซี ซึ่งมีชุดของทาร์เก็ตไฟล์ (.tlc) ลงไป โปรแกรมทีแอลซีบอกให้รู้ว่าจะแปลงไฟล์ model.rtwให้เป็นโค้ดที่ถูกสร้าง(generated code)

โปรแกรมทีแอลซีประกอบด้วย

- ไฟล์หลัก ซึ่งเรียกว่าซิสเต็มทาร์เก็ตไฟล์(system target files)

-ชุดของบล็อคทาร์เก็ตไฟล์ซึ่งบ่งชี้ว่าจะแปลงแต่ละบล็อคในโมเดลให้กลายเป็นโค้ดที่เฉพาะเจาะจงได้อย่างไร

- ทีแอลซี ฟังกชั่นไลบรารี่ เป็นชุดของไลบรารี่ที่ทีแอลซีใช้ในการแปลงไฟล์ model.rtw

2.10.3 เมคยูทิลิตี้ (make utility)

(make)เป็นเครื่องมือในการคอมไพล์และเชื่อมโค้ดที่ถูกสร้าง เพื่อสร้างการปฏิบัติงาน เราจะสามารถกำหนดรูปร่างของเมคได้โดยไปทำการเปลี่ยนแปลงที่ซิสเต็ม เทมเพลต เมคไฟล์ (system template makefile)

2.10.4 เอส-ฟังก์ชั่น

เอสฟังก์ชั่นสามารถทำให้เราสร้างโค้ดได้เองลงในซิมูลิ้งค์ เราสามารถใส่เอสฟังก์ชั่นโค้ดโดยตรงลงในโค้ดที่ถูกสร้างได้ เรียกวิธีนี้ว่าอินลายนิ่ง

ทีแอลซี เป็นหัวใจสำคัญในการสร้างขึ้นมาใช้ในตัวของมันเอง โดยวิธีการสร้างนี้เกี่ยวข้องกับอินลายนิ่งเอสฟังก์ชั่น

2.10.5 ขั้นตอนการสร้างในเรียลไทม์เวอร์คชอพ

สร้างไฟล์ model.rtw ขึ้นมาซึ่งเป็นตัวแทนของซิมูลิ้งค์บล็อคในไฟล์นี้จะมีข้อมูลต่างเช่น ค่าของตัวแปร, ความกว้างของเวคเตอร์,เวลาในการสุ่มค่าและลำดับการปฏิบัติการณ์ของบล็อคโมเดลนั้น หลังจากที่ได้ไฟล์นี้แล้วต้องร้องขอให้ทีแอลซี มาแปลงไฟล์นี้ให้เป็นโค้ดที่เฉพาะเจาะจง โดยทีแอลซีจะเริ่มอ่านไฟล์ model.rtw หลังจากนั้นจะคอมไพล์ และปฏิบัติการณ์กับคำสั่งในทาร์เก็ตไฟล์ ผลลัพธ์ของทีแอลซีจะได้ซอสโค้ด(source code) ที่เป็นฟังก์ชั่นของซิมูลิ้งค์บล็อค

ขั้นตอนไปคือการสร้างไฟล์ system.mk ขึ้นจากเทมเพลทเมคไฟล์(system.tmf) ไฟล์ model.mk ถูกสร้างโดยการบันทึกหัวข้อของ system.tmf และขยายออก หลังจากได้ model.mk คำสั่งเมค จะสร้างการปฏิบัติการณ์ และยังสามารถดาวน์โหลดการปฏิบัติการณ์ลงในฮาร์ดแวร์เป้าหมาย และหลังจากดาวน์โหลดแล้วถ้าเราใช้ โหมดภายนอก(external mode) คุณยังสามารถติดต่อนำสัญญาณกลับมาที่ซิมูลิ้งค์เพื่อปรับแต่งค่าต่างๆในขณะที่ทำงานได้

2.10.6 ความคิดพื้นฐานที่ควรรู้ในเรียลไทม์เวอร์คชอพ

2.10.6.1 เรียลไทม์ทั่วไป (generic realtime)

- สิ่งแวดล้อมสำหรับการจำลองโมเดลที่เป็นขั้นตอนตายตัว ในโหมดเดี่ยว หรือ โหมดหลายหน้าที่ (single mode or multitasking mode)
- ในความที่สามารถแสดงค่าของโค้ดได้
- เป็นจุดเริ่มต้นของการทำเป้าหมาย

2.10.6.2 เป้าหมาย

ในการใช้เรียลไทม์เวอร์คชอพเราต้องตัดสินใจว่าจะนำโค้ดที่ถูกสร้างไปวางไว้ในสิ่งแวดล้อมใด ในสิ่งแวดล้อมนั้นถูกเรียกว่าทาร์เก็ต โฮส (host) ที่ซึ่งแมทแลป ซิมูลิ้งค์และเรียลไทม์เวอร์คชอพทำงานอยู่ ในการใช้เครื่องมือ build ที่อยู่บนโฮสเราจะได้โค้ดและการปฏิบัติการณ์ที่ทำงานบนระบบของเป้าหมาย

เราจำเป็นต้องนิยามว่า เป้าหมายที่ใช้เป็นชนิดใด ตารางข้างล่างแสดงตัวอย่างของเป้าหมาย, ไฟล์เป้าหมายของระบบและเทมเพลตเมคไฟล์ของแต่ละสิ่งแวดล้อม

Target	System Target File	Template Makefile
Generic real-time	grt.tlc	grt_default.tmf
Embedded-C	ert.tlc	ert_default.tmf
DOS (4GW) real-time	drt.tlc	drt_watc.tmf
Tornado (VxWorks) real-time	tornado.tlc	tornado.tmf
Rapid Simulation	rsim.tlc	rsim_default.tmf

ตารางที่ 2-1 ตารางแสดงทาร์เก็ต ไฟล์ที่ใช้ และเทมเพลตเมคไฟล์ที่ใช้

2.10.6.3 ไฟล์ทีแอลซี (target language compiler file)

เป็นไฟล์ที่ใช้สำหรับคอมไพล์และปฏิบัติการณ์ ซึ่งจะอธิบายถึงการสร้างโค้ดให้กับเป้าหมายของเรา เรียลไทม์เวอร์คชอพใช้ไฟล์ทีแอลซีแปลงซิมูลิ้งค์โมเดลให้เป็นโค้ด

ซิสเต็มทาร์เก็ตไฟล์ เป็นทางเข้าของโปรแกรมทีแอลซี เพื่อสร้างการปฏิบัติการณ์ส่วนบล็อคคทาร์เก็ตไฟล์จะบอกว่าโค้ดมองหาบล็อคซิมูลิ้งค์ได้อย่างไร

2.10.6.4 กระบวนการการสร้าง

กระบวนการนี้ถูกควบคุมด้วย make_rtw ซึ่งจะถูกร้องข้อเมื่อคุณกดปุ่มบิ้วในเรียลไทม์เวอร์คชอพ โดยอันดับแรก make_rtw จะแปลงบล็อคไดอะแกรมเพื่อสร้างไฟล์ model.rtw และอันดับต่อไป make_rtw จะบอกให้ทีแอลซีมาสร้างโค้ด เราจะต้องกำหนดซิสเต็มทาร์เก็ตไฟล์ที่อยู่บนหน้าเรียลไทม์เวอร์คสำหรับทีแอลซีด้วย ต่อจากนั้น make_rtw จะสร้างเมคไฟล์ model.mk ขึ้นมาจากเทมเพลตเมคไฟล์(template make files) ในหน้าต่างเรียลไทม์เวอร์คชอพ และสุดท้ายถ้าโฮสที่เราใช้งานอยู่ตรงกันกับ โฮสในเทมเพลทเมคไฟล์ เมคจะถูกเรียกขึ้นเพื่อสร้างโปรแกรมจากโค้ด

2.10.6.5 เทมเพลตเมคไฟล์

เรียลไทม์เวอร์คชอพใช้เทมเพลตเมคไฟล์ทำการสร้างปฏิบัติการณ์จากโค้ด เพื่อให้สะดวกเทมเพลตเมคไฟล์จะมีนามสกุลเป็น.tmf และมีชื่อตามเป้าหมาย ตัวอย่างเช่น grt_unix.tmf เป็นเทมเพลตเมคไฟล์สำหรับ unix ไฟล์ที่ถูกสร้างจากเทมเพลตเมคไฟล์จะเลียนแบบแต่ละบรรทัดจากเทมเพลตเมคไฟล์และขยายโทเค่น (token) เข้าไปในเมคไฟล์ ชื่อของไฟล์ที่ถูกสร้างขึ้นคือmodel.mk

และไฟล์ model.mk จะผ่านไปยังเครื่องมือเมคเพื่อสร้างการปฏิบัติการณ์จากชุดของไฟล์เหล่านั้น เครื่องมือ จะแสดงวันที่ของการเกี่ยวเนื่องกันระหว่างเป้าหมายกับซีไฟล์ และจะสร้างไฟล์เป้าหมายใหม่ถ้าจำเป็น

2.10.6.6 รูปร่างลักษณะของเทมเพลตเมคไฟล์

คุณสามารถกำหนดกระบวนการสร้างโดยการเปลี่ยนแปลงที่เทมเพลตเมคไฟล์ อาจทำได้โดยการก๊อปปี้กระบวนการนั้นไดเรคทอรี่ของตัวที่ทำงานอยู่และแก้ไขใหม่ หรืออีกทางหนึ่งคุณสามารถกำหนดการปฏิบัติการณ์ของเทมเพลตเมคไฟล์ โดยไปกำหนดที่ตัวเลือกของการสร้างให้กับการสร้างคำสั่ง make_rtw ดังตัวอย่าง

make_rtw OPT_OPTS=-g

2.10.6.7 การระบุพารามิเตอร์ของโมเดล

คุณจะสามารถเปลี่ยนแปลงพารามิเตอร์ของโมเดลที่ควบคุมการเรื่องการจำลองระบบเช่น เวลาเริ่มและเวลาหยุดโดยไปเลือกที่ ไดอะลอกบอกซ์ซิมูเลชั่นพารามิเตอร์ (Simulation parameter) พารามิเตอร์เหล่านี้ถูกใช้โดยตรงกับการสร้างโค้ดและโปรแกรม ดังนั้นก่อนที่จะสร้างโค้ดและโปรแกรม จะต้องแน่ใจว่าพารามิเตอร์ได้ถูกเซตอย่างถูกต้องในไดอะลอกบอกซ์ซิมูเลชั่นของพารามิเตอร์

2.11 แมสกิ้ง (masking)

เป็นการซิมูลิ้งค์หรือเชื่อมต่อกันของไดอะลอกบอกซ์ และไอคอน(หน่วยย่อย)หรือเป็นระบบย่อย ลักษณะของแมสกิ้ง

1.ในโมเดลของคุณสามารถแทนด้วยไดอะลอกบอกซ์ หลายตัวในระบบย่อยเพียงตัวเดียว สามารถเปิดทีละบล็อคและได้ค่าพารามิเตอร์บนพารามิเตอร์บน แมส ไดอะลอกบอกซ์และในระบบย่อยของแมส

2. มีคำอธิบายและช่วยแนะนำผู้ใช้ในบล็อคคำบรรยาย(block description),คำบรรยายของพารามิเตอร์ (parameter field labels) และ ตัวช่วย (help text)

3. ให้กำหนดคำสั่งและประมวลผลในค่าพารามิเตอร์

4. สร้างบล็อคย่อย

5. ทำการปรับปรุงระบบย่อยโดยใช้การซ่อนไว้หลังการอินเตอร์เฟส

การแซมเปิ้ลแมสของระบบย่อย

ระบบย่อยอย่างง่ายๆ เป็นสมการเส้นตรง Y=mx+b โดยแสดงการเปิดบล็อคระบบย่อยในวินโดว์ซึ่งสมการ mx+b ประกอบด้วย บล็อคเกนคือ ความชัน (m)ค่าคงที่ที่จุดตัดแกนคือ b เราสามารถดับเบิ้ลคลิก ในการเปิดระบบย่อยซึ่งภายในพารามิเตอร์จะประกอบด้วย 2 ค่าที่ตั้งไว้ และเราสามารถสร้างค่าพารามิเตอร์เองได้เรียกค่าเหล่านี้ว่า แมสพารามิเตอร์

2.11.1 สร้างแมสสำหรับระบบย่อยโดย

1.กำหนดค่าแมสไดอะลอกบอกซ์พารามิเตอร์ เช่นตัวอย่างของความชันและค่าคงที่

2.กำหนดชื่อตัวแปร เพื่อสำหรับเก็บค่าของแต่ละพารามิเตอร์

3.กรอกข้อมูลของบล็อคประกอบด้วย การอธิบายและตัวช่วยบล็อค

4.ทำการวาดภาพด้วยคำสั่งและสร้างบล็อคไอคอน

5.เติมคำสั่งและตัวแปรต่างๆในดรออิ้งคอมแมนด์

2.11.2 การสร้างแมสไดอะลอกบอกซ์พรอมพต์ (Mask Dialog Box Prompts)

เราสามารถเลือกระบบย่อยบล็อค และเลือกการสร้างแมสจากเมนู อีดิต(Edit)ส่วนบนของไดอะลอกบล็อคจะมีหน้าอินนิเชี่ยลไลเซชั่น ในหนานี้จะประกอบด้วย

1.ค่าพรอมต์คือข้อความเลเบลที่ใช้แทนค่าพารามิเตอร์

2.ชนิดของการควบคุม(control type) คือ รูปแบบของการควบคุมค่าพารามิเตอร์โดยการใส่ค่าหรือเลือกค่า

3.ตัวแปรคือชื่อของตัวแปรที่จะเก็บค่าพารามิเตอร์นั้น

โดยปกติแล้วเราสามารถอ้างอิงค่าพารามิเตอร์หลายๆตัวในพรอมต์เช่นความชันก็กำหนดสโลปพารามิเตอร์และค่าคงที่กำหนดเป็นจุดตัด ทั้งค่าความชันและค่าคงที่ล้วนอยู่ในหมวดอีดิต ซึ่งผู้ใช้สามารถเปลี่ยนแปลงค่าและสามารถเก็บค่าตัวแปรไว้ในแมสเวอร์คสเปซ(maskworkspace) จะเก็บเฉพาะตัวแปรเท่านั้น เช่นความชันมีตัวแปรเป็น m ซึ่งจะมีค่าความชันมากจากแมสเวอร์คสเปซและจะแสดงในส่วนของอีดิตเตอร์(editor) หลังจากที่คุณสร้างค่าพารามิเตอร์และปิดแล้วสามารถดับเบิ้ลคลิกเพื่อใส่ค่าตัวเลขเช่น 3 ลงในความชัน 2 ลงในค่าคงที่

- การสร้างบล็อคการบรรยายและตัวช่วย สองส่วนที่จะอยู่ในส่วนของหน้าของเอกสาร
- การสร้างบล็อคไอคอนการสร้างบล็อคย่อย mx+b ซึ่งมีการซิมูลิ้งค์ภายในระบบย่อย ซึ่งค่าไอคอนนี้เราสามารถพลอตเส้นตรงที่มีความชัน และในหน้าไอคอนประกอบด้วย

1.ดรออิ้ง คอมแมนด์ (Drawing commands)

1.1the drawing command เป็นการพลอตเส้นจากจุด (0,0) ถึง (0,m)

1.2สามารถเข้าถึงตัวแปรโดยการใส่ค่าต่างๆลงในความชัน ซึ่งไอคอนจะมีการพลอตให้โดยอัตโนมัติ

2.ไอคอน พรอพเพอร์ตี้ ประกอบด้วยแบบปกติกับการจัดที่กำหนดจุดโดยจาก(0,0)ไปยัง (1,1)

สรุป ในการสร้างต้องประกอบด้วย

1. กำหนดค่าไดอะลอกบอกซ์พรอมต์ (ค่าพารามิเตอร์ต่างๆ)และตัวอักษร

2. กำหนดการบรรยายแมสบล็อค (คำอธิบายฟังก์ชั่นนการทำงาน) และ ตัวช่วยช่วยแนะนำการใช้งาน

3 กำหนดคำสั่งและสร้างแมสบล็อคไอคอน(หน่วยย่อย)

2.11.3 แมสอีดิตเตอร์ (mask editor)

รูปภายนอกบล็อคการเลือกบล็อคและสร้างแมสจากเมนูอีดิตซึ่งในส่วนของเดอะแมสอีดิตเตอร์ประกอบด้วย 3 เพจต่างๆกันดังนี้

1. อินนิเชี่ยลไลเซชั่น เพจ คุณต้องกำหนดค่าพารามิเตอร์พรอมต์ ชื่อตัวแปรของค่าพารามาเตอร์และคำสั่ง
2. ไอคอนเพจ สามารถกำหนดบล็อคไอคอน
3. หน้าเอกสาร โดยกำหนดชนิดของแมสกำหนดคำอธิบายบล็อคและส่วนช่วยการใช้บล็อคและมี 5 ปุ่มด้านล่างของ แมสอีดิตเตอร์

- ปุ่มแอพพลายด์ สร้างและเปลี่ยนแปลงข้อมูลในทุกหน้าในการเปิดอีดิตเตอร์
- ปุ่มรีเวอร์ต การแทนค่าให้กลับมาสู่ค่ามาตรฐาน
- ปุ่มอันแมส ไม่กระทำการแมสและปิดแมสอีดิตเตอร์ ข้อมูลที่ได้รับการสร้างหรือการเซตไว้จะไป กระทำการเมื่อได้มีการปิดและไม่สามารถค้นหาอีกได้
- ปุ่มช่วยเหลือ แสดงเนื้อหาในบทนี้
- ปุ่มปิด(close) เมื่อทำการเซตแมสทุกหน้าแล้วก็ต้องปิดแมสอีดิตเตอร์ด้วย

ในการมองระบบภายใต้แมส ปราศจาก อันแมสกิ้ง (unmasking) การเลือกบล็อค และเลือก Look under Masking จากเมนูอีดิตถ้าบล็อคอยู่ในระบบย่อย กำลังที่จะไม่มีผลเกิดขึ้น

2.11.4 การเริ่มต้นหน้า (The Initialization page)

ผู้ใช้สามารถกำหนดค่าพารามิเตอร์ในระบบแมสและสามารถสร้างแมสให้ติดต่อกับผู้ใช้โดยค่าพารามิเตอร์ใน หน้าเริ่มต้นจากตัวอย่าง mx+bใน การเริ่มต้นหน้าประกอบด้วย

1. พรอมพต์ และกลุ่มตัวแปร การกำหนดข้อมูลพรอมพต์จะช่วยให้ผู้ใช้เต็มหรือเลือกค่าพารามิเตอร์และจะแสดงรายการพรอมพต์ เมื่อตัวแปรถูกกำหนดค่าแล้วเลือกรูปแบบการควบคุม และเก็บค่าในตัวแปร ถ้ามีการกำหนดชนิดเป็นการประมาณค่า(Evaluate) ค่าสตริงเกน จะถูกตีค่าโดยแมทแลปและผลจะถูกกำหนดค่าตัวแปร ถ้าชนิดเป็นลิเทอรัลสตริง(literal string) จะไม่ถูกตีค่าโดยแมทแลปแต่จะให้สตริงที่มีข้อความว่า 'gain' ถ้าคุณต้องการกำหนดสตริงเพื่อตีค่าในการเลือกชนิดลิเทอรอล โดยใช้คำสั่ง eval matlab ใน อินนิเชี่ยลไลซ์คอมมานด์(initialization command)
2. การสร้างฟิวด์พรอมพต์ (field prompt)

ในการสร้างพรอมพต์แรกในรายการ โดยการเอนเทอร์พรอมพต์(enter prompt) ลงในพรอมพต์ฟิวด์(prompt field) ตัวแปรจะมีค่าพารามิเตอร์ในฟิวด์ตัวแปรและเลือกรูปแบบการควบคุมและกำหนดชนิด

3. การแทรกพรอมพต์

1) เลือกพรอมพต์ และต้องการแทรกพรอมพต์ใหม่โดยการคลิกปุ่ม Add ของปุ่มทางซ้ายมือ

2)ใสค่าพรอมพต์ลงใน พรอมพต์ฟิวด์ใส่ตัวแปรโดยคงค่าพารามิเตอร์ในฟิวด์ตัวแปร

4. การแก้ไข พรอมพต์

1)เลือกพรอมพต์ในรายการพรอมพต์,ชื่อตัวแปร,รูปแบบควบคุมและชนิดใช้ในฟิวด์ตามรายการ

2)แก้ไขค่า เมื่อคลิกเมาส์ไปข้างนอกแล้วกดเอนเทอร์หรือรีเทิร์นคีย์ ในการติดต่อพรอมพต์ให้ทันสมัย

5.การลบพรอมพต์

1)เลือกพรอมพต์ที่ต้องการลบ

2)คลิกปุ่มดีลีต(delete) ทางด้านซ้ายรายการ

6.การย้ายพรอมพต์

1)เลือกพรอมพต์ที่ต้องการย้าย

2)ย้ายค่าขึ้นไป 1 ตำแหน่งคลิกอัพ หรือย้ายค่าลงมา1ตำแหน่งคลิกปุ่มดาวน์

Defalut ค่าของ แมสบล็อคพารามิเตอร์ (Masked Block parameters)

การบันทึกบล็อคไว้ในส่วนของบล็อคไลบรารี่ตามลำดับดังนี้

1)ลากบล็อคไปไว้ใน บล็อคไลบรารี่

2)เปิดบล็อคแล้วเติมค่าที่ต้องการและปิดไดอะลอกบอกซ์

3)บันทึกบล็อคไลบราลี่

คำสั่ง Initialization

คำสั่งของตัวแปรจะถูกกำหนดในแมสเวอร์คสเปซ ตัวแปรนี้จะถูกดำเนินตามคำสั่ง Initialization กำหนดแมสโดยเขียนบล็อคในระบบย่อยเขียนคำสั่งในการวาดบล็อคไอคอน ซิมูลิ้งค์จะดำเนินการเมื่อ

1.โมเดลถูกโหลด

2.การซิมูเลชั่นจะเริ่มหรือบล็อคไดอะแกรมจะถูกสร้างขึ้น

3.ไอคอนของบล็อคจะถูกวาดหรือพล็อตตามคำสั่ง initialization

คำสั่งจะผิดพลาดในแมทแลปพิจารณาที่ functions,ตัวกระทำ,ตัวแปรและจะไม่สามารถเข้าถึงเวอร์คสเปซ เมื่อมีเซมิโคลอนบนคำสั่งวินโดว์

แมส เวอร์คสเปซ

ซิมูลิ้งค์ จะสร้างพื้นที่ถูกเรียกว่าเวอร์คสเปซ เมื่อแมสค์มีคำสั่ง initialization หรือมีพรอมพต์และตัวแปร แมสบล็อคไม่สามารถเข้าถึงฐานเวอร์คสเปซหรือเวอร์คสเปซอื่นๆ

ในเวอร์คสเปซประกอบด้วยตัวแปรในพารามิเตอร์และตัวแปรจากคำสั่ง initialization ตัวแปรเหล่านี้ในเวอร์คสเปซจะถูกเข้าถึงโดยแมสบล็อค ถ้าบล็อคเป็นระบบย่อยหนึ่งเราสามารถเข้าถึงทุกบล็อคในระบบย่อย แมสเวอร์คสเปซที่เป็นอนาลอกจะมีการใช้เอ็ม-ไฟล์ ฟังก์ชั่น คิดสำนวนการพูดลงในไดอะลอกบอกซ์ของ อันเดอร์ลายอิ้ง บล็อก และคำสั่ง initialization บนอีดิตเตอร์ ของฟังก์ชั่น เอ็มไฟล์ การใช้อนาลอกจะอยู่ในรูปฟังก์ชั่น

2.12 ความรู้เบื้องต้นเกี่ยวกับพอร์ตขนาน

การประมวลผลข้อมูลเพื่องานควบคุมนั้น สิ่งแรกจะต้องมีส่วนของสัญญาณอินพุต ซึ่งอาจจะมาจากวงจรตรวจจับต่างๆ ผ่านวงจรภาคหน้าเพื่อเปลี่ยนรูปแบบสัญญาณอินพุตให้เหมาะสมกับการเชื่อมต่อกับคอมพิวเตอร์ เมื่อข้อมูลอินพุตถูกส่งเข้าสู่คอมพิวเตอร์แล้ว คอมพิวเตอร์จะทำการประมวลผลข้อมูลที่ได้มาเหล่านั้นให้อยู่ในรูปแบบที่เหมาะสมก่อนที่จะส่งออกไปยังภายนอกผ่านอุปกรณ์เอาต์พุต ซึ่งอาจจะเป็นการส่งออกไปยังจอภาพ หรือส่งออกไปยังจุดเชื่อมต่ออื่นๆ เพื่อควบคุมอุปกรณ์เอาต์พุตต่อไป

การเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกทั้งส่วน ของภาคอินพุต และภาคเอาต์พุต สามารถทำได้หลายวิธี

- เชื่อมต่อผ่านทางการ์ดอินเตอร์เฟส ซึ่งวิธีการเสียบหรือติดตั้งการ์ดลงในสล็อตภายในเครื่องคอมพิวเตอร์
- เชื่อมต่อผ่านทางพอร์ตอนุกรม
- เชื่อมต่อผ่านทางพอร์ตขนาน
- เชื่อมต่อผ่านระบบมาตรฐานอื่น เช่นพอร์ต USB

ทำไมถึงเลือกใช้พอร์ตขนาน

เมื่อเทียบการใช้การ์ดอินพุตเอาต์พุตที่ต้องติดตั้งอยู่ภายในเครื่องคอมพิวเตอร์แล้วพอร์ตขนานมีข้อได้เปรียบอยู่หลายประการดังนี้

- ในด้านความปลอดภัย การที่ต้องถอดฝาเครื่องคอมพิวเตอร์ออกมาเพื่อเสียบการ์ดเชื่อมต่อลงในสล็อตของคอมพิวเตอร์อาจจะทำให้เกิดความเสียหายกับส่วนอื่นๆ ของคอมพิวเตอร์ได้ ถ้าผู้ที่ใช้งานไม่มีความชำนาญหรือเกิดการต่อวงจรที่ผิดพลาด
- ในด้านการเข้ากันได้กับคอมพิวเตอร์ส่วนใหญ่ การเชื่อมต่อโดยใช้การ์ดที่เสียบลงในสล็อตไม่สามารถใช้กับคอมพิวเตอร์ในปัจจุบันได้ทุกรุ่น
- ข้อจำกัดพื้นที่ คอมพิวเตอร์บางเครื่องมีการเสียบการ์ดเชื่อมต่อตัวอื่นๆ อยู่แล้ว อาทิ การ์ดเสียง การ์ดโมเด็ม เป็นต้น จนไม่มีสล็อตเหลือพอสำหรับการเสียบการ์ดต่อเพิ่มเติม
- ความสะดวกในการใช้งาน การเชื่อมต่อทางพอร์ตขนานสามารถทำได้ง่ายๆ เพียงต่อสายสำหรับเชื่อมต่อเข้ากับคอนเน็กเตอร์ DB-25 ของพอร์ตขนาน
- ความเร็วในการสื่อสารข้อมูลกับพอร์ตขนาน มีความเร็วเท่ากับการติดต่อกับระบบบัสโดยตรง และมีความเร็วมากกว่าการติดต่อผ่านทางพอร์ตอนุกรม

2.12.1ความรู้เบื้องต้นของพอร์ตขนาน

พอร์ตขนาน สาเหตุที่มีชื่อนี้ เนื่องจากการถ่ายทอดข้อมูลของพอร์ตนี้เป็นแบบขนาน การประมวลผลส่วนใหญ่จะมีขนาด 8 บิต

ลักษณะทางกายภาพของพอร์ตขนาน

เพื่อให้เข้าถึงการนำเอาพอร์ตขนานไปใช้งาน ก่อนอื่นต้องทำความเข้าใจก่อนว่า ปกตินั้นการสั่งพิมพ์รายงานจากคอมพิวเตอร์ไปยังพอร์ตขนาน กับเครื่องพิมพ์ เริ่มจากสัญญาณพอร์ต data ถูกส่งออกไปยังเครื่องพิมพ์ พร้อมทั้งส่งสัญญาณสโตรบ (strobe) ออกไปด้วยเพื่อให้เครื่องรับรู้ว่ามีการส่งข้อมูลใหม่มาที่ขาดาต้า (data) แล้วจากนั้นคอมพิวเตอร์จะต้องรอการตอบกลับจากเครื่องพิมพ์นั่นคือเครื่องพิมพ์จะสร้างสัญญาณบีซี่ (busy) หรือเพื่อบอกว่าเครื่องพิมพ์ยังไม่พร้อมที่จะรับข้อมูลใหม่ จนกระทั่งเมื่อเครื่องพิมพ์พร้อม เครื่องพิมพ์จะสร้างสัญญาณ ack ส่งไปยังคอมพิวเตอร์เพื่อแจ้งว่า พร้อมที่จะรับข้อมูล

นอกจากสัญญาณทั้งสามแล้วส่วนใหญ่การติดต่อกับเครื่องพิมพ์ยังต้องมีสัญญาณอื่นๆ ร่วมด้วย เนื่องจากเครื่องพิมพ์ต้องทำหน้าที่ ถึง 3 อย่างด้วยกันคือ รับข้อมูลจากคอมพิวเตอร์,พิมพ์ข้อมูลที่รับเข้ามา และตอบสนองต่อการใช้งานของผู้ใช้

พอร์ตขนานของคอมพิวเตอร์ยังแยกย่อยออกเป็นอีก 3 พอร์ตได้แก่ พอร์ตเอาต์พุตที่ทำหน้าที่ส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ พอร์ตเอาต์พุตสำหรับสัญญาณสโตรบ และรีเซ็ตพอร์ตอินพุตสำหรับการอ่านค่าสัญญาณแอคโนเลจ(Acknowledge),บีซี่ และสัญญาณ Error จากเครื่องพิมพ์

โดยปกติพอร์ตขนานออกแบบมาให้มีสายสัญญาณอยู่ทั้งหมด 17 เส้น สายสัญญาณเหล่านี้จะมีรีจิสเตอร์ 3 ตัวควบคุมการทำงานดังนี้

1. พอร์ตเอาต์พุตสำหรับสัญญาณข้อมูล 8 เส้นมีรีจิสเตอร์ดาต้าควบคุม
2. พอร์ตอินพุตสำหรับการอ่านค่าสถานะต่าง ๆจากภายนอกมีอยู่ด้วยกัน 5 เส้น ใช้รีจิสเตอร์สเตตัส(STATUS) ในการควบคุม
3. พอร์ตเอาต์พุตสำหรับส่งสัญญาณควบคุมไปยังอุปกรณ์ภายนอก มีอยู่ด้วยกัน 4 เส้นใช้รีจิสเตอร์คอนโทรลในการควบคุม

การติดต่อกับพอร์ตขนานจะต้องมีการอ้างแอดเดรส ตำแหน่งแอดเดรสที่ใช้อ้างถึงจะเป็นตำแหน่ง A0-A9 และใช้ขา IOR และ IOW สำหรับเป็นตัวเลือกว่าต้องการอ่านหรือเขียน รีจิสเตอร์ตัวใด จากการดีโค้ดแอดเดรส A0-A9 นี้เองทำให้ได้สัญญาณออกมาเพื่อไปควบคุมหรืออีนาเบิลวงจรบัฟเฟอร์ต่างๆดังนี้

Datawrite สัญญาณอีนาเบิลสำหรับนำข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Data ของพอร์ตขนาน

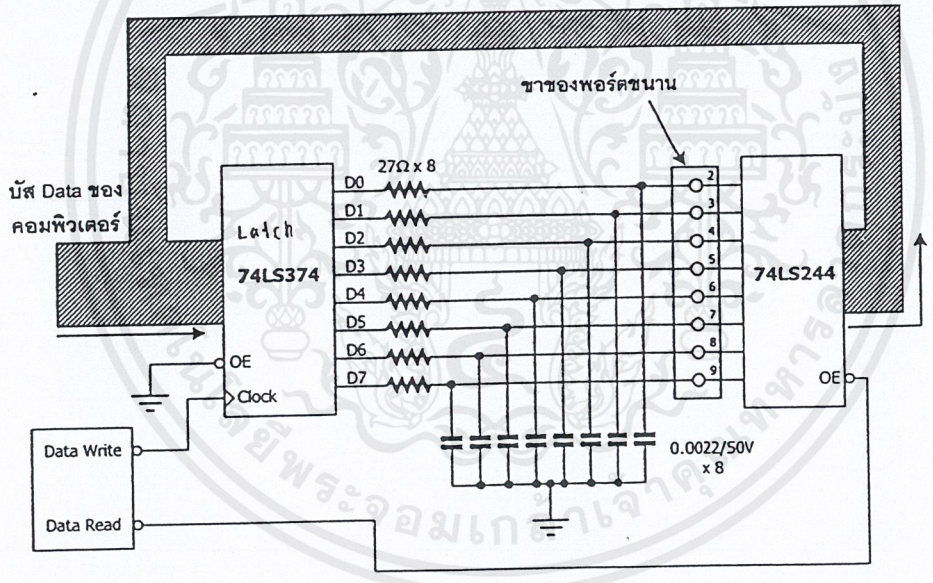
Dataread สัญญาณอีนาเบิลสำหรับอ่านข้อมูลจากขา Data ของพอร์ตขนานมาเก็บไว้ในบัส Data

Controlwrite สัญญาณอีนาเบิลสำหรับนำข้อมูลที่อยู่ในบัส Data ไปออกที่ขา Control ของพอร์ตขนาน สำหรับพอร์ตนี้นอกจากจะส่งข้อมูลออกไปยังพอร์ตขนานแล้ว ยังทำหน้าที่อีนาเบิลการอินเตอร์รัปต์ของการเปลี่ยนแปลงสัญญาณที่พอร์ต Status อีกด้วย

Controlread สัญญาณอีนาเบิลสำหรับอ่านค่าข้อมูลจากขา Control มาเก็บไว้ในบัส Data

Statusread สัญญาณอีนาเบิลสำหรับอ่านค่าข้อมูลจากขาพอร์ต status มาเก็บไว้ในบัส Data

พอร์ตดาต้า



รูปที่ 2-8 พอร์ตดาต้า

จากรูป แสดงให้เห็นว่าพอร์ตดาต้าประกอบไปด้วยบัฟเฟอร์ 1 ตัว และไอซีแลตช์อีก 1 ตัว เมื่อคอมพิวเตอร์ต้องการส่งข้อมูล คอมพิวเตอร์จะเขียนข้อมูลไปยังไอซีแลตช์ 1 ตัว ทั้ง 8 บิต เอาต์พุตของไอซีแลตช์ 1 คือ D0-D7 ซึ่งเอาต์พุตนี้จะไปปรากฏอยู่ที่พอร์ตขนานในตำแหน่งขา 2 ถึง 9

สำหรับบัฟเฟอร์สำหรับการอ่านข้อมูลกลับได้แก่เบอร์ 74LS244 ซึ่งเมื่อต้องการอ่านค่าคอมพิวเตอร์จะส่งสัญญาณ DataRead ออกมาเพื่ออีนาเบิลไอซี 74LS244 สำหรับพอร์ตขนานแบบมาตรฐาน พอร์ต DATA จะต้องใช้เพื่อการส่งค่าออกเอาต์พุตเท่านั้น แต่สำหรับพอร์ตขนานที่มีการสื่อสารสองทิศทาง สามารถอ่านค่าจากพอร์ต Data ได้ด้วย แต่ก่อนที่จะอ่านค่าต้องจำไว้เสมอว่าจะต้องป้อนค่าเอาต์พุตให้มีค่าลอจิก “1”ทั้งหมดก่อน

พอร์ต Control

DB-25	รีจิสเตอร์	ทิศทาง	ตัวแทนบิต	เมื่อขาสัญญาณ	หน้าที่การทำงาน
1	Control	Out	$\overline{C0}$	$\overline{STROBE}$	แอกทีฟ "0" ส่งค่าออกไปเพื่อบอกว่าที่ขาตาต้ามีข้อมูลแล้ว
2-9	Data	Out	D1-D8	DATA1-DATA8	สำหรับพอร์ตขนานมาตรฐานเดิมขานี้ทำหน้าที่เป็นขาส่งข้อมูลเอาต์พุตเท่านั้น สำหรับในปัจจุบันขานี้รับข้อมูลอินพุตได้ด้วย
10	Status	In	S6	nACK	เป็นพัลส์ลอจิก "0" ที่ส่งมาจากเครื่องพิมพ์เพื่อบอกว่าได้รับข้อมูลที่ส่งไปแล้ว
11	Status	In	$\overline{S7}$	$\overline{BUSY}$	เป็นสัญญาณแจ้งมาจากเครื่องพิมพ์ว่ายังไม่พร้อมรับข้อมูล
12	Status	In	S5	PE	แจ้งกระดาษหมด
13	Status	In	S4	SELECT	แจ้งว่าเครื่องพิมพ์ต่ออยู่
14	Control	Out	$\overline{C1}$	$\overline{AUTO\ FEED}$	สั่งเครื่องพิมพ์ให้เลื่อนบันทัด
15	Status	In	S3	$\overline{ERROR}$	สัญญาณจากเครื่องพิมพ์มายังคอมพิวเตอร์เพื่อแสดงข้อผิดพลาดจากการพิมพ์
16	Control	Out	C2	$\overline{INIT}$	รีเซตเครื่องพิมพ์โดยให้ลอจิก "0"
17	Control	Out	$\overline{C3}$	$\overline{SELECT\text{-}IN}$	ส่งสัญญาณไปยังเครื่องพิมพ์เพื่อแจ้งว่าต้องการเลือกเครื่องพิมพ์เครื่องนี้
18-25				GND	กราวด์

ตารางที่ 2-2 สถานะของขาคอนเนคเตอร์แสดง

พอร์ต Control ใช้สำหรับคอมพิวเตอร์ควบคุมเครื่องพิมพ์ จากตารางที่ 2-1 จะเห็นว่าพอร์ตควบคุม ประกอบไปด้วยบิตเอาต์พุต 4 บิตที่ต่อออกไปยังเครื่องพิมพ์ ส่วนบิตอีนาเบิลอินเตอรัปต์ไม่ได้ถูกต่อออกไป เอาต์พุตของพอร์ตควบคุมมีอินเวอร์เตอร์แบบคอลเลคเตอร์เปิดต่อร่วมอยู่ด้วย โดยเอาต์พุตเหล่านี้จะถูกพูลอัปไว้ด้วยตัวต้านทานค่า 4.7 กิโลโอห์ม สำหรับบิต C2 จะผ่านอินเวอร์เตอร์ถึงสองตัวทำให้เอาต์พุตของบิต C2 ไม่มีการกลับสถานะลอจิก

สถานะของพอร์ตควบคุมสามาถอ่านค่ากลับได้โดยการใช้บัฟเฟอร์เบอร์ 74LS240 ซึ่งเอาต์พุตของ 74LS40 มีอินเวอร์เตอร์อยู่ภายใน ทำให้ค่าที่อ่านได้ตรงกับค่าที่ส่งออกไป การควบคุมการ

อ่านและเขียนข้อมูลกับพอร์ต ควบคุม คอมพิวเตอร์จะส่งข้อมูลที่ขา Control write และ Control Read

เนื่องจากเอาต์พุตของพอร์ต Control เป็นแบบคอลเล็กเตอร์เปิด ดังนั้นผู้ใช้งานสามารถใช้พอร์ตนี้ในการอ่านค่าสัญญาณอินพุตจากภายนอกได้ โดยก่อนที่จะอ่านค่าจะต้องทำให้ขาพอร์ตที่ต้องการอ่านค่ามีลอจิก "1"เสียก่อน

พอร์ตแสดงสถานะหรือพอร์ต Status

พอร์ตStatus เป็นพอร์ตที่คอมพิวเตอร์ใช้สำหรับการอ่านค่าสถานะจากเครื่องพิมพ์ จะสังเกตได้ว่ามีขาสัญญาณอยู่ทั้งหมด 5 สัญญาณ ด้วยกันและจะเรียกชื่อเป็น S3,S4,S5,S6 และS7 ซึ่งตัวเลขนั้นหมายถึงตำแหน่งบิตของขาเหล่านี้ภายในรีจิสเตอร์ Status นั่นเอง สำหรับบิต S 7 จะมีข้อแตกต่างจากบิตอื่นๆ ที่เมื่อสัญญาณจากภายนอกส่งเข้ามาแล้วจะไม่ผ่านอินเวอร์เตอร์ ในขณะที่ขาอื่นๆ ผ่านอินเวอร์เตอร์ทั้งหมด ดังนั้นเมื่อข้อมูลผ่านจากขาอินพุตไปยัง 74LS240 ซึ่งเอาต์พุตมีการกลับสถานะทำให้บิต S7 เป็นบิตเดียวที่การกลับสถานะนอกจากนี้ในการใช้งานถ้าต้องการให้มีการสร้างสัญญาณอินเตอร์รัปต์จากขอบขา S6 สามารถกำหนดค่าได้จากพอร์ตControl บิต4

การนำพอร์ตขนานไปใช้งาน

สำหรับพอร์ตขนานแบบมาตรฐาน ผู้ใช้งานสามารถนำพอร์ตอินพุต 5 บิตพอร์ตเอาต์พุต 4 บิต และพอร์ตเอาต์พุตอีก 8 บิต ไปใช้งานได้โดยตรง โดยที่ 4 บิตของพอร์ตเอาต์พุตหรือพอร์ต Control นั้นสามารถดัดแปลงให้ใช้งานเป็นพอร์ตอินพุตขนาด 4 บิตได้ด้วยดังนั้นผู้ใช้งานจึงสามารถนำสัญญาณจากพอร์ตขนานที่มีมากถึง 17 เส้นไปใช้งานในการควบคุมโดยใช้ระดับสัญญาณ TTL

การเขียนโปรแกรมติดต่อกับพอร์ตขนาน

2.12.2 การติดต่อพอร์ตขนาน กับระบบบัสแบบ I^2C

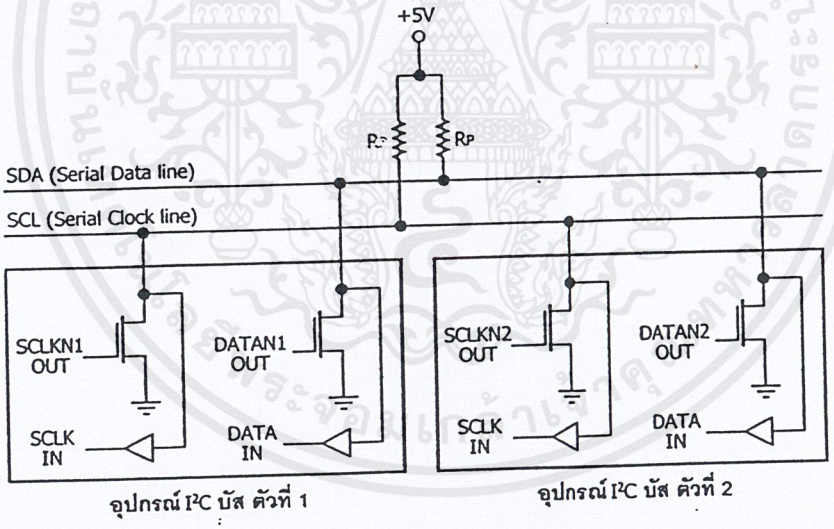
I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีโมดูลสามารถติดต่อสั่งงานและควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือสายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไปส่วนการกำหนดแอดเดรสหรือตำแหน่ง

สำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบน บัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือ SDA ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรม หรือ SCL ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองว่า สาย SDA และSCL

คุณสมบัติโดยทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด หรือคอลเลกเตอร์เปิด ดังแสดงรายละเอียดในรูปที่ 2 - 2

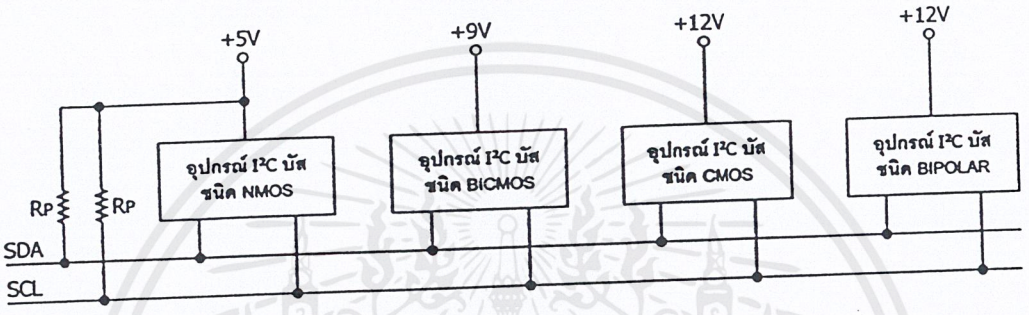


รูปที่ 2 – 9 การต่อใช้งานอุปกรณ์ผ่าน ตัวต้านทานพูลอัปบนบัส I^2C

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาที ในโหมดปกติ และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง อุปกรณ์ที่ต่อรวมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 พิโคฟารัด การเข้าถึงอุปกรณ์บนบัส I^2C ใช้การเข้าถึงข้อมูลได้ 2 แบบ คือ 7 บิต หรือ 10 บิต

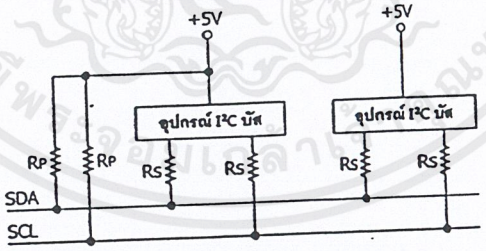
ข้อเด่นอีกประการหนึ่งของบัส I^2C คือสามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจจะใช้ไฟเลี้ยง +5V ในขณะที่อีกตัว

หนึ่งใช้ไฟเลี้ยง +12V การต่อรวมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกันและต้องต่อตัวต้านทานพูลอัป เข้ากับแรงดัน +5 V เสมอ



รูปที่ 2-10 แสดงการเชื่อมต่ออุปกรณ์ภายนอกที่ใช้ไฟเลี้ยงต่างกัน

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องต่อตัวต้านทานอนุกรมกับขา SDA และ SCL เรียกว่า RS ก่อนต่อเข้าสู่บัส I^2C ดังแสดงในรูปที่ 2-4 หลักการของบัส I^2C



รูปที่ 2-11 แสดงการต่อตัวต้านทานเพื่อป้องกันไฟกระชาก

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้นดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล เพื่อผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะขออธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูลเรียกว่า ตัวส่ง อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่งบางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่าสเลฟ

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น
2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

สภาวะที่เกิดขึ้นบนบัส I^2C มีด้วยกัน 5 สภาวะ

1. บัสว่าง สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้
2. เริ่มการถ่ายทอดข้อมูล เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น
3. หยุดการถ่ายทอดข้อมูล เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด
4. ข้อมูลดำรงอยู่บนบัส สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น “0”หรือ “1” ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะทีสาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น
5. รับรู้ข้อมูล เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตรับรู้ มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส

6. อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังหรือกำลังติดต่ออยู่ในขณะนั้นก็จะสร้างบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

บทที่ 3
การสร้างและการคำนวณ

3.1 การสร้างเจนเนอริคเรียลไทม์(generic realtime)

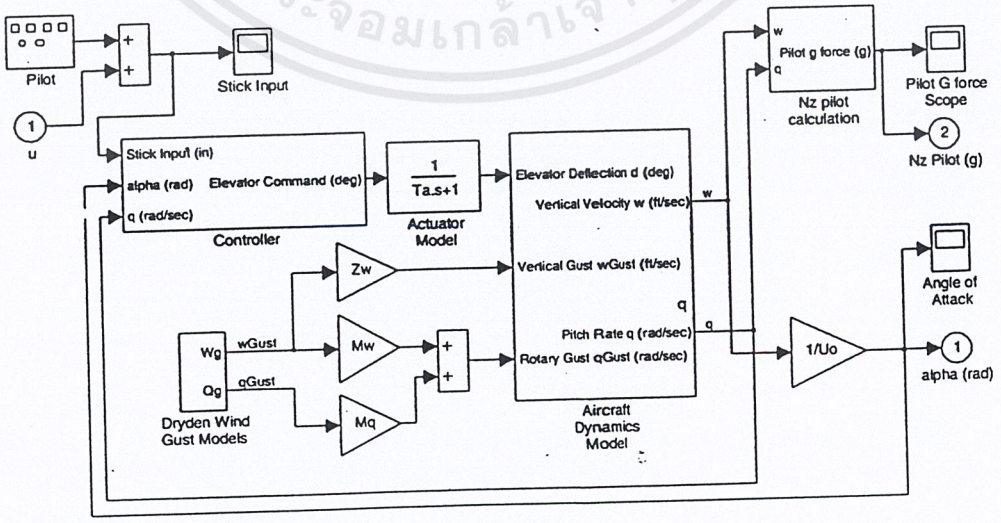
ต่อไปนี้เป็นการทดลองเพื่ออธิบายวิธีการสร้างโค้ดภาษาซีจากซิมูลิ้งค์โมเดลและสร้างโปรแกรมเจนเนอริคเรียลไทม์ ซึ่งโปรแกรมจะมีลักษณะดังนี้

- มีการปฏิบัติการณ์แบบโปรแกรมสแตนอโลน กล่าวคือไม่ขึ้นกับเวลาและเหตุการณ์ภายนอก
- มีการบันทึกข้อมูลในแมทไฟล์ สำหรับการวิเคราะห์ต่อไป
- ถูกสร้างในสิ่งแวดล้อมแบบพีซี (PC)

การทดลองนี้ยังแสดงให้เห็นถึงการใช้ดาต้าลอกกิ้ง(Data logging) ในการทำให้โค้ดที่สร้างขึ้นใช้งานได้ โดยดาต้าลอกกิ้งจะทำให้เราสามารถเปรียบเทียบผลลัพธ์ของระบบจากบล็อคของซิมูลิ้งค์กับข้อมูลที่ถูกเก็บไว้โดยโปรแกรมที่สร้างจากโค้ด

3.1.1 ตัวอย่างโมเดลการควบคุมเครื่องบินของซิมูลิ้งค์

ในการแสดงโมเดลให้พิมพ์ที่บรรทัดคำสั่งของแมทแลปว่า F-14 จะมีโมเดลเกิดขึ้นที่ซิมูลิ้งค์ดังรูป

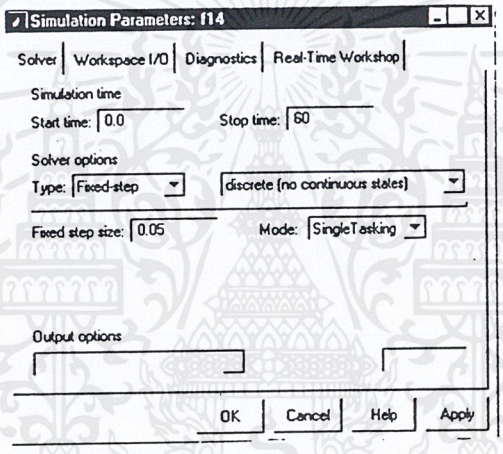


รูปที่ 3-1 ซิมูลิ้งค์โมเดลของเครื่องบิน F14

จะเห็นว่าในโมเดลมีบล็อคอินพุทของตัวกำเนิดสัญญาณ ที่มาจากคันบังคับของนักบิน ซึ่งสามารถดูสัญญาณได้ที่สโคปโดยระบบนี้มีเอาต์พุทเป็นมุมของปีกและแรงจี (G Force) ของนักบินโดยดูสัญญาณได้จากสโคปเช่นกัน โดยสโคปมีจุดประสงค์เพื่อสังเกตการปฏิบัติงานของโมเดลนี้

3.1.2 การตั้งค่าของโปรแกรม

หลังจากที่โมเดลถูกแสดงขึ้นแล้วให้เลือกที่คำสั่ง พารามิเตอร์ จากเมนู ซิมูเลชั่น จะแสดงหน้าโซ้ลเวอร์(solver) ดังรูป



รูปที่ 3-2 การตั้งค่าซิมูลิ้งค์พารามิเตอร์ของ F14

จากนั้นให้เซตตัวเลือกต่างๆ , เลือกเทมเพลตเมคไฟล์,เลือกโค้ดและสร้างโปรแกรม ในครั้งแรกที่เปิดไดอะลอกบอกซ์ขึ้นมา จะมีค่าที่ไม่ตรงกับค่าพารามิเตอร์ที่ต้องการในการสร้างการปฏิบัติการณ์เจนเนอริคเรียลไทม์ ดังนั้นจึงจำเป็นต้องแก้ไขค่าให้ตรงกับค่าในโมเดลที่ต้องใช้กับเรียลไทม์เวอร์คชอพ สำหรับการทดลองนี้ให้เซตค่าดังนี้

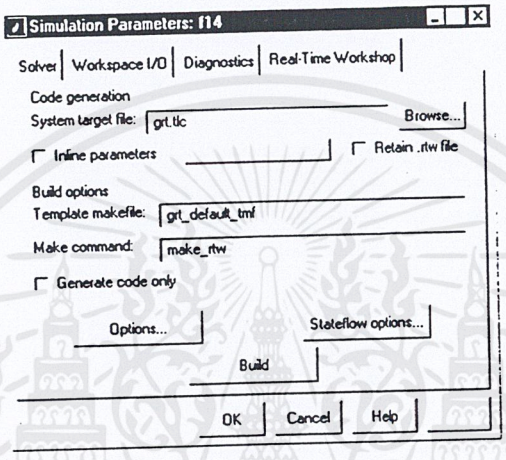
1. ที่หน้าโซลเวอร์ เซตที่ช่อง **Solver option type** ว่า Fixed-step และเลือก โซลเวอร์ ode5
2. ให้เซตที่ช่อง**Fixed Step Size** = 0.05

ขั้นตอนการสร้างโปรแกรม

เราจะสร้างโปรแกรมได้ที่ หน้าเรียลไทม์เวอร์คชอพ ในไดอะลอกบอกซ์ซิมูเลชั่นพารามิเตอร์

3.1.3 ขั้นตอนการสร้างโปรแกรม

เราจะสร้างโปรแกรมได้ที่ หน้าเรียลไทม์เวอร์คชอพ ในไดอะลอกบอกซ์ซิมูเลชั่นพารามิเตอร์



รูปที่ 3-3 หน้าต่างการสร้าง F14 แบบเรียลไทม์เวอร์คชอพ

ให้กดปุ่ม BUILD เพื่อสร้างซี โค้ด จาก F-14 โมเดล คำสั่งสร้างจะเรียกไฟล์ซิสเต็มทาร์เก็ตที่ชื่อ grt.tlc ซึ่งไฟล์นี้ใช้เทมเพลตเมคไฟล์ที่ชื่อ grt_default_tmf เพื่อมาสร้างเมคไฟล์ และเรียลไทม์เวอร์คชอพก็จะใช้เมคไฟล์ที่สร้างขึ้นมาสร้างโปรแกรม

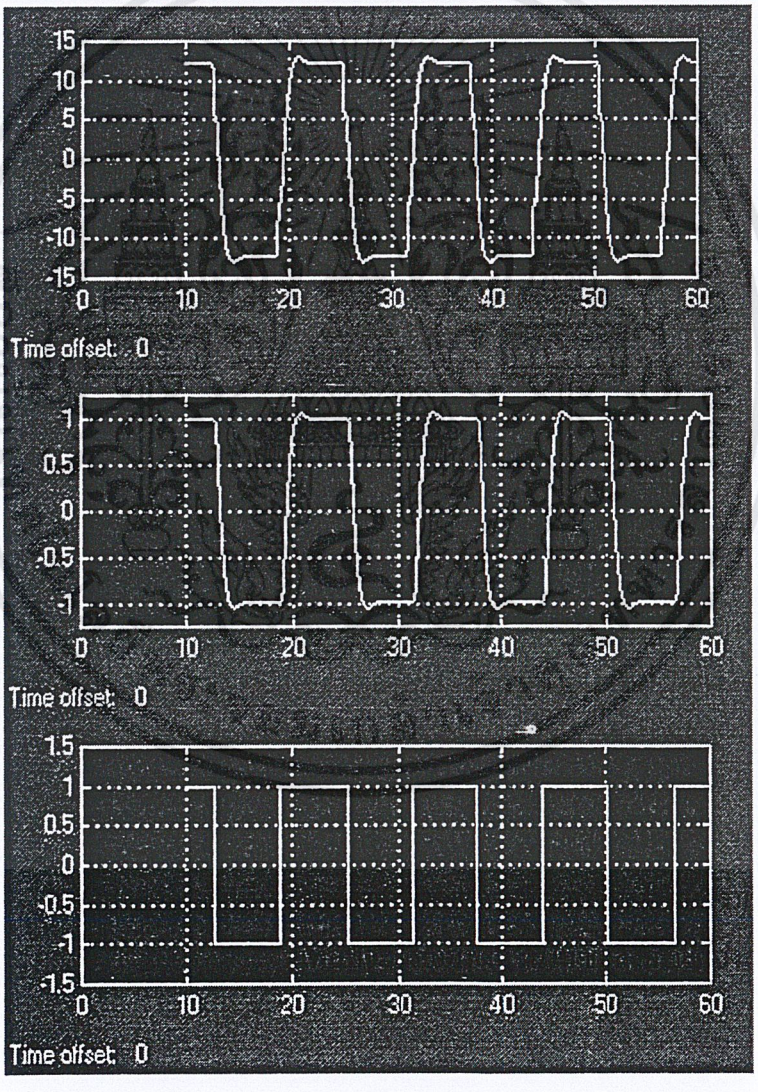
ตอนที่เราคลิกปุ่ม build เรียลไทม์เวอร์คชอพจะเรียกคำสั่งเมค ซึ่งจะมีการเปลี่ยนแปลงดังนี้

- จะแปลงบล็อคไดอะแกรมเพื่อสร้างไฟล์ model.rtw ในที่นี้ คือ f14.rtw
- จะเรียกทีแอลซีมาทำหน้าที่แปลโปรแกรมทีแอลซีและปฏิบัติการณ์กับ model.rtw เพื่อสร้างโค้ด
- สร้างเมคไฟล์ model.mkในที่นี้คือ f14.mk จากเทมเพลทเมคไฟล์ grt_vc.tmf
- ถ้าซิมูลิ้งค์ทำงานที่เดียวกับที่ระบุในเทมเพลทเมคไฟล์แล้ว ก็จะถือว่าโปรแกรมได้ถูกสั่ง ถ้าไม่เช่นนั้นกระบวนการจะหยุด หลังจากการสร้างโมเดลโค้ดแล้วจะบอกให้กำหนดโฮสใหม่ให้ตรงกับเป้าหมาย

การกำหนดโฮสมี3ทางเลือก คือ HOST=PC,UNIX or ANY ถ้าต้องการเป้าหมายที่แตกต่างจากเครื่องที่ทำงานอยู่ให้เซตโฮสเป็น ANY ที่เทมเพลตเมคไฟล์

หลังจากการปฏิบัติการณ์ที่ปุ่มbuild เสร็จสิ้นลง เรียลไทม์เวอร์คชอพจะสร้างไฟล์ต่างดังนี้

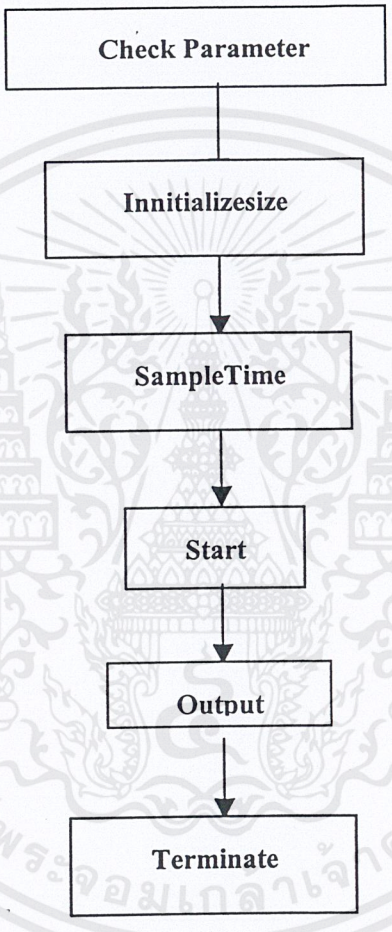
- f14.c เป็นซีโคดแบบสแตนอโลน
- f14.h เป็นไฟล์เฮดเดอร์ซึ่งประกอบด้วยข้อมูลเกี่ยวกับสถานะของตัวแปร
- f14_export.h เป็นเฮดเดอร์ไฟล์ที่ประกอบด้วยข้อมูลเกี่ยวกับสัญญาณและพารามิเตอร์ที่ถูกส่งออก
- f14.reg เป็นไฟล์ที่ประกอบด้วย ฟังก์ชั่นเกี่ยวกับการลงทะเบียนโมเดลในโมเดล F14
- f14.exe เป็นตัวปฏิบัติการณ์ เรียลไทม์



รูปที่ 3-4 แสดงกราฟของแรงจีของนักบิน,แรงปะทะที่ปีกและอินพุทจากคันบังคับของการทดลอง F14 ตามลำดับ

3.2 โปรแกรมที่ใช้ในการติดต่อในการ์ดอินเตอร์เฟส

โปรแกรมที่ใช้ติดต่อกับการ์ดอินเตอร์เฟส ซึ่งจะเป็นส่วนที่ต้องการในเอสฟังก์ชั่นรูทีน ของโปรแกรม Ctrlcard.c จะมีการทำงานเป็นไปตามโฟลตชาร์ตข้างล่างนี้



รูปที่ 3-5 โฟลว์ชาร์จของการทำงานของโปรแกรมที่ใช้ติดต่อระหว่างการ์ดอินเตอร์เฟส

เพื่อที่จะใช้เอสฟังก์ชั่นในซิมูลิ้งค์ จะต้องสร้าง source file เพื่อที่จะเรียกใช้ ctrlcard.c โดยพิมพ์

```
mex ctrlcard.c
```

3.2.1 รูทีนของเอสฟังก์ชั่น ระหว่างการซิมูเลชั่นของ ctrlcard.c

1. mdlCheckParameters ซิมูลิ้งค์จะเรียกรูทีนนี้ในการตรวจสอบค่าพารามิเตอร์ที่ผ่านการ์ดอินเตอร์เฟสเข้ามา เพื่อทำให้ช่วงของค่าเป็นไปตามที่กำหนด เนื่องจากการส่งค่าอาจจะมีการผิดพลาดหรือมีข้อผิดพลาด ซึ่งในที่จะตรวจสอบค่าให้อยู่ในช่วง 0 ถึง 5 โวลต์

2. mdlinitializeSizes ซิมูลิ้งค์จะเรียกรูทีนนี้ขณะที่อีดิตโมเดล เพื่อจะกำหนดจำนวนอินพุทและเอาต์พุทพอร์ต ซิมูลิ้งค์จะเรียกมันขณะที่เริ่มต้นการซิมูเลชั่นเพื่อร้องขอขนาดของพอร์ตและวัตถุประสงค์อื่น

3. mdlSampletimes ซิมูลิ้งค์จะเรียกรูปทีนนี้เพื่อกำหนดขนาดของการสุ่มเอสฟังก์ชั่น

4. mdlStart เป็นการเริ่มต้นการทำงานของโปรแกรมซึ่งส่วนนี้จะมีการกำหนดพอยเตอร์ที่ใช้สำหรับการชี้ข้อมูลของอาร์เรย์ทั้งในส่วนที่รับเข้าและส่งออก

5. mdlOutput เป็นการคำนวณของค่าเอาต์พุท ซึ่งจะรับค่าอินพุทและจะมีการแปลงอยู่ในช่วงที่เราต้องการคือ จาก 0 – 5 เป็น –5 – 5 โวลต์ รูทีนนี้จะถูกเรียกในขณะที่มีการซิมูเลชั่นลูปแต่ละเวลาซึ่งต้องมีการอัพเดทใน ctrlcard.

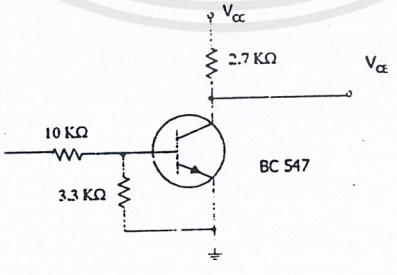
6. mdlterminate จะแสดงในส่วนท้ายสุดของโปรแกรมแต่ใน ctrlcard จะไม่มีการแสดงของหน้าที่นี้เนื่องจากรูทีนว่างเปล่า

3.3 การสร้างการ์ดอินเตอร์เฟส

การ์ดอินเตอร์เฟส มีช่องรับสัญญาณเข้า(analog input) 4 ช่อง และมีช่องส่งสัญญาณ(analog output) 2 ช่อง โดยมีส่วนประกอบที่สำคัญ 5 ส่วนคือ

3.3.1 ไอซี 74HC541 ทำหน้าที่เป็นบัฟเฟอร์ข้อมูล

3.3.2 วงจรสวิชต์ on/off



รูปที่ 3-6 แสดงวงจรสวิชต์ on/off

ตามปกติทรานซิสเตอร์มีช่วงการทำงานทั้งหมด 3 สถานะคือ active,cut off,saturate แต่เราจะเลือกใช้เพียง 2 สถานะคือ cut-off กับ saturate เพื่อทำหน้าที่เป็น switch on-off โดย V_{cc} ทำหน้าที่เป็น switch จากกฎของเคียชโฮฟ

$$V_{ce}=V_{cc} - I_c*R_L$$

ถ้า on ให้ I_B=0 ดังนั้น I_C=0

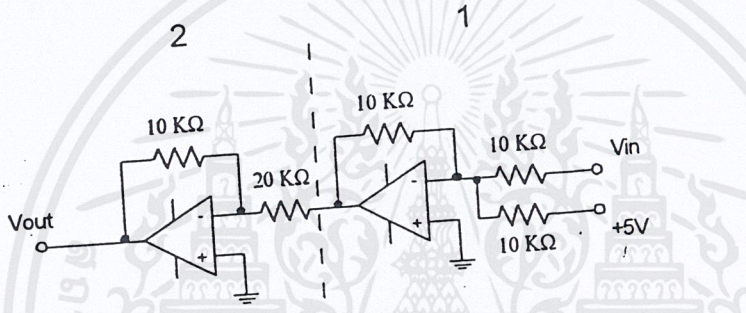
$V_{CE} = V_{CC}$

ถ้า off ให้ I saturate ได้

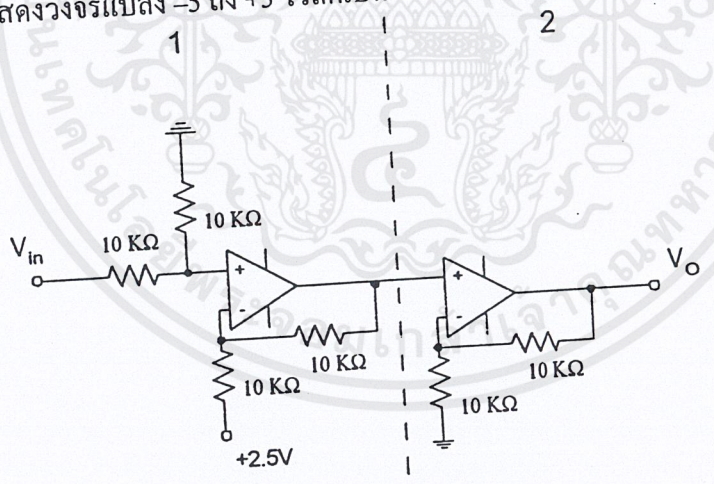
$V_{CE} = V_{CC} - I_C * R_L = 0$

3.3.3 วงจรปรับแต่งแรงดัน

สัญญาณที่ออกมาจากคอมพิวเตอร์อยู่ในช่วง 0-5 โวลต์แต่ในการใช้ระบบควบคุมต้องการสัญญาณที่อยู่ในช่วง +5 ถึง –5 โวลต์ จึงจำเป็นต้องใช้วงจรปรับแต่งแรงดัน



รูปที่ 3-7 แสดงวงจรแปลง –5 ถึง +5 โวลต์เป็น 0 ถึง 5 โวลต์



รูปที่ 3-8 แสดงวงจรแปลง 0 ถึง 5 V เป็น –5 ถึง 5 V

3.3.4ไอซี A/Dและ D/A PCF8591

ทำหน้าที่แปลงสัญญาณดิจิตอลขนาด 8 บิทเป็นอนาลอก 0 ถึง 5 โวลต์ ออกสู่ภายนอกและสามารถแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลขนาด 8 บิทเข้าสู่คอมพิวเตอร์ได้

3.3.5 ไอซี PCF8574 และ ไอซี DAC0800

PCF85747 เป็นอุปกรณ์เพิ่มช่องสัญญาณเอาต์พุตสำหรับระบบบัสแบบ I^2C ซึ่งถูกใช้บนการ์คอินเตอร์เฟสนี้ทำให้สามารถเพิ่มจำนวนช่องสัญญาณเอาต์พุตได้ตามต้องการ

ส่วน DAC0800 ใช้ในการแปลงสัญญาณจาก สัญญาณดิจิตอลเป็นสัญญาณอะนาลอกในรูป Voltage ออกสู่ภายนอก ทางช่อง Vo2

3.4 การรับและส่งสัญญาณในรูป Voltage ผ่านการ์ดอินเตอร์เฟส

3.4.1 การส่งค่า Voltage ออกผ่านช่อง Vo1 และ Vo2

1. ส่งสัญญาณออกทางช่อง Vo1 โดยผ่าน IC PCF8951 ที่ Address &H44

```
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H44)
Call Ack
Call Send8BIT( แรงดันที่ต้องการส่งออก * 51)
Call Ack
Call I2CStop
```

2. ส่งสัญญาณออกทางช่อง Vo2โดยผ่าน IC PCF8574 ที่ Address &H72

```
Call I2CStart
Call Send8BIT(&H72)
Call Ack
Call Send8BIT( แรงดันที่ต้องการส่งออก * 51)
Call Ack
Call I2CStop
```

3.4.2 การรับค่าสัญญาณ Voltage จากภายนอกเข้ามาทางช่อง Vin1,Vin2,Vin3 และ Vin4

จะผ่าน IC PCF8591 โดยมีชุดคำสั่งดังนี้

```
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
```

```
V1 = (DAT * 5) / 256
Call MAck
V2 = (DAT * 5) / 256
Call MAck
V3 = (DAT * 5) / 256
Call MAck
V4 = (DAT * 5) / 256
Call Ack
Call I2CStop
```

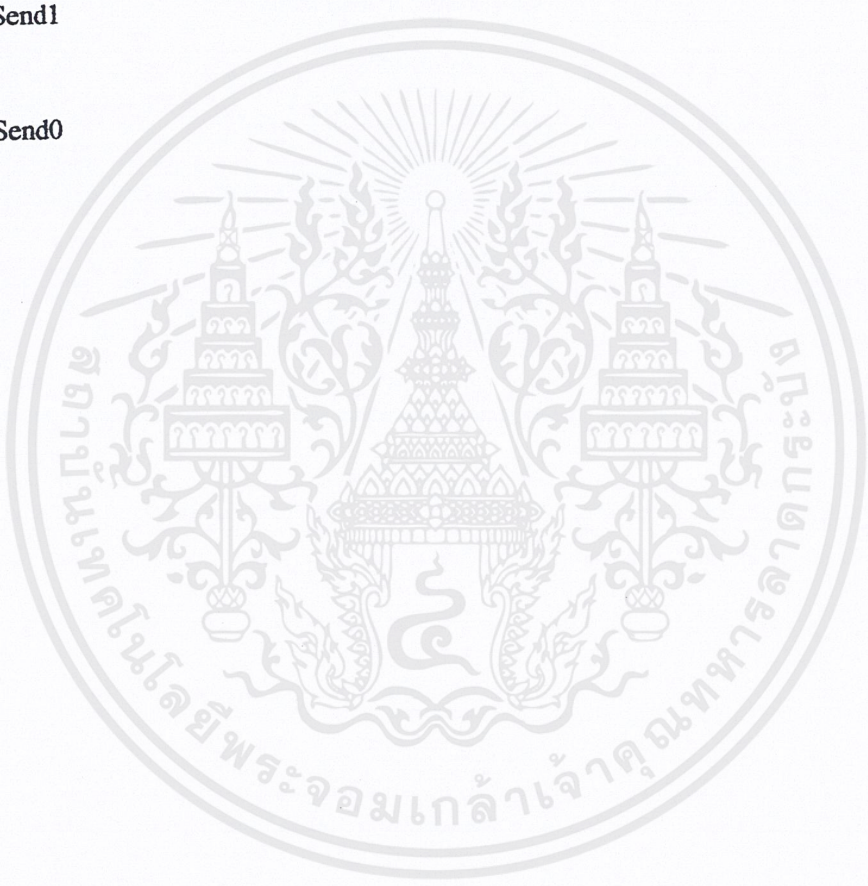
V1,V2,V3 และ V4 คือ ค่า Voltage ที่รับเข้ามา

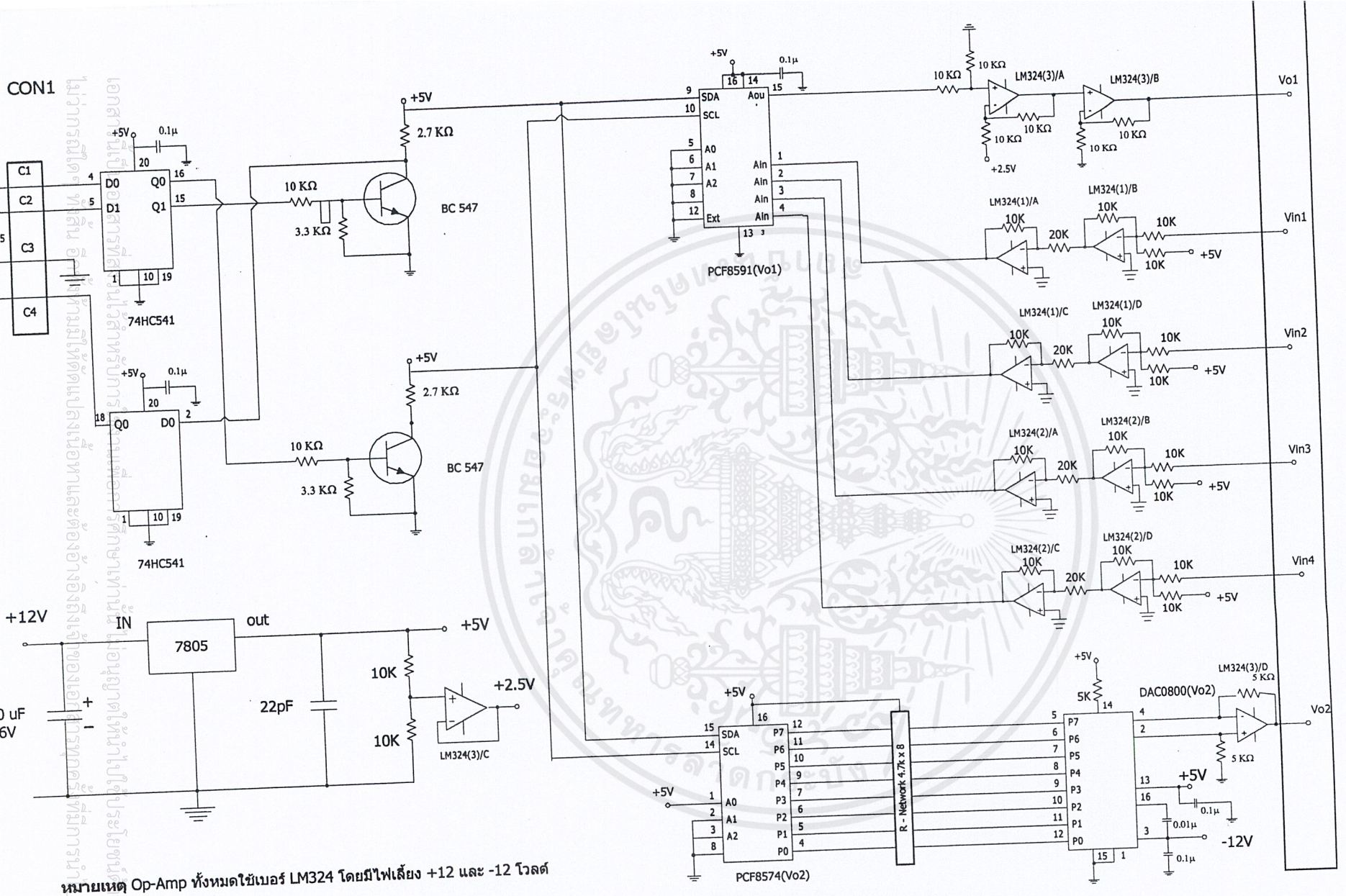
โดยที่ มี Sub Function ที่ใช้ร่วมดังต่อไปนี้

```
Private Sub I2CStart()
Out &H37A, Inp(&H37A) Or 1    'SDA=1
Out &H37A, Inp(&H37A) Or 2    'SCL=1
Out &H37A, Inp(&H37A) And &HFE   'SDA=0
Out &H37A, Inp(&H37A) And &HFD   'SCL=0
End Sub

Private Sub I2CStop()
Out &H37A, Inp(&H37A) And &HFE   'SDA=0
Out &H37A, Inp(&H37A) Or 2 'SCL=1
Out &H37A, Inp(&H37A) Or 1    'SDA=1
End Sub

Private Sub Send0()
Out &H37A, Inp(&H37A) And &HFE   'SDA=0
Out &H37A, Inp(&H37A) Or 2   'SCL=1
Out &H37A, Inp(&H37A) And &HFD 'SCL=0
```

```
End Sub

Private Sub Send1()
Out &H37A, Inp(&H37A) Or 1    'SDA=1
Out &H37A, Inp(&H37A) Or 2    'SCL=1
Out &H37A, Inp(&H37A) And &HFD  'SCL=0
End Sub

Private Sub Ack()
Out &H37A, Inp(&H37A) Or 1   'SDA=1
Out &H37A, Inp(&H37A) Or 2    'SCL=1
Out &H37A, Inp(&H37A) And &HFD   'SCL=0
End Sub
Private Sub MAck()
Out &H37A, Inp(&H37A) And &HFE   'SDA=0
Out &H37A, Inp(&H37A) Or 2     'SCL=1
Out &H37A, Inp(&H37A) And &HFD  'SCL=0
Out &H37A, Inp(&H37A) Or 1     'SDA=1
End Sub

Private Function DAT()
 For I = 7 To 0 Step -1
  Out &H37A, Inp(&H37A) Or 1    'SDA=1
  Out &H37A, Inp(&H37A) Or 2    'SCL=1
    If (Inp(&H379) And &H80) <> &H80 Then    'Read SDA
      DAT1 = 2 ^ I Or DAT1
    End If
  Out &H37A, Inp(&H37A) And &HFD       'SCL=0
 Next I
 DAT = DAT1                    'Data 8 Bit
```

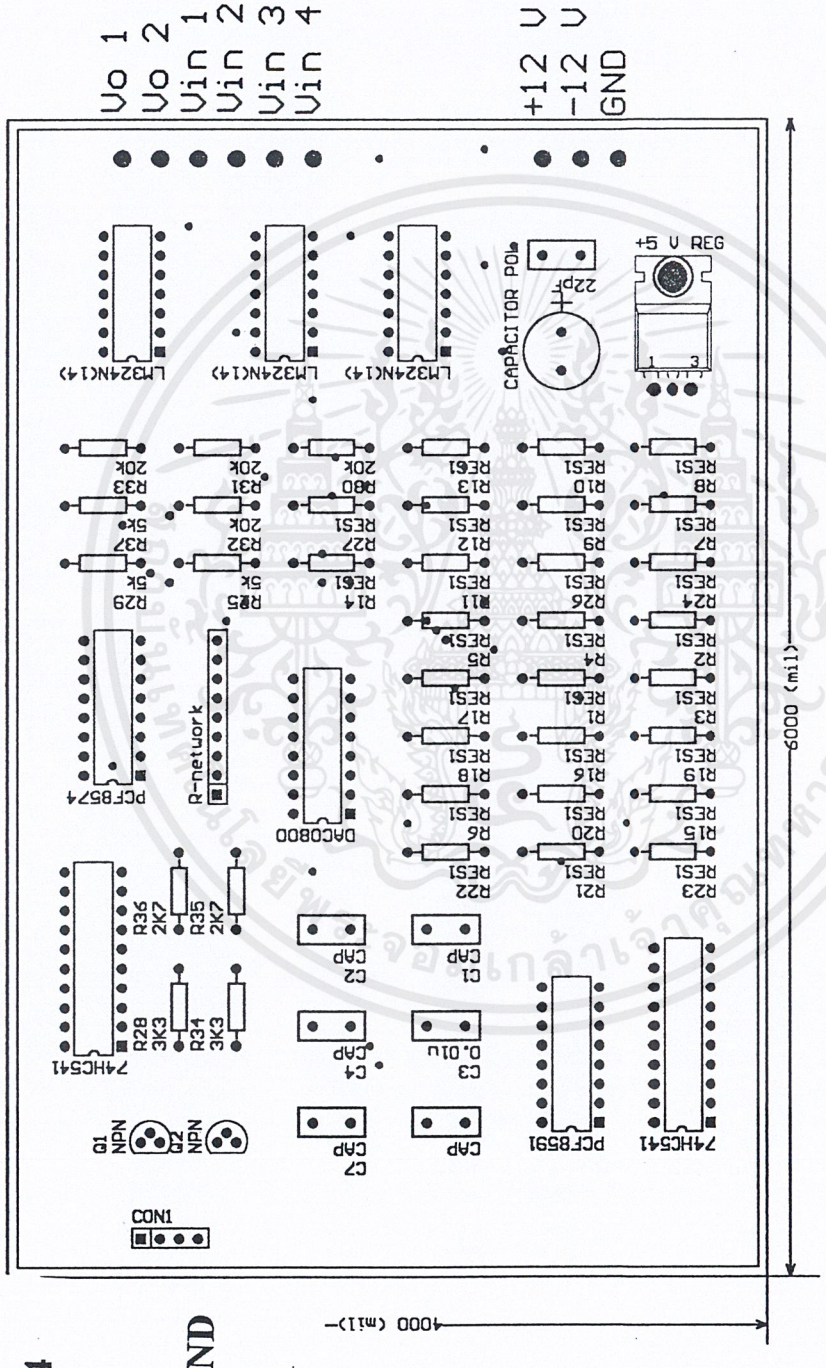
```
End Function

Private Sub Send8BIT(Add As Integer)
 For I = 7 To 0 Step -1              ' Loop 7 Cycle
    If (Add And 2 ^ I) = 2 ^ I Then  'Test Bit 0 OR 1
      Call Send1
    Else
      Call Send0
    End If
 Next I
End Sub
```





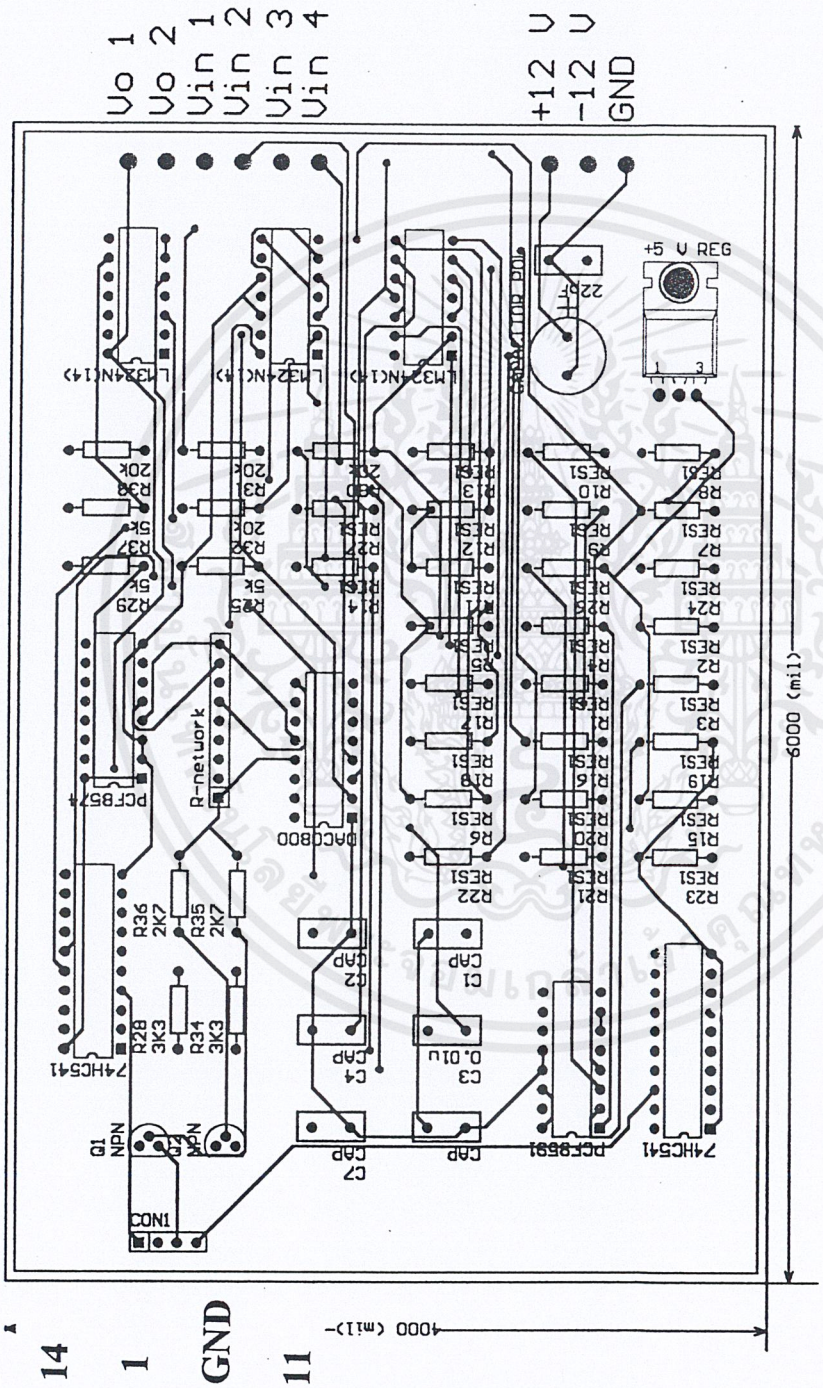
หมายเหตุ Op-Amp ทั้งหมดใช้เบอร์ LM324 โดยมีไฟเลี้ยง +12 และ -12 โวลต์

รูปที่ 3-9 แสดงบล็อกไดอะแกรมของบอร์ดอินเตอร์เฟสที่สมบูรณ์

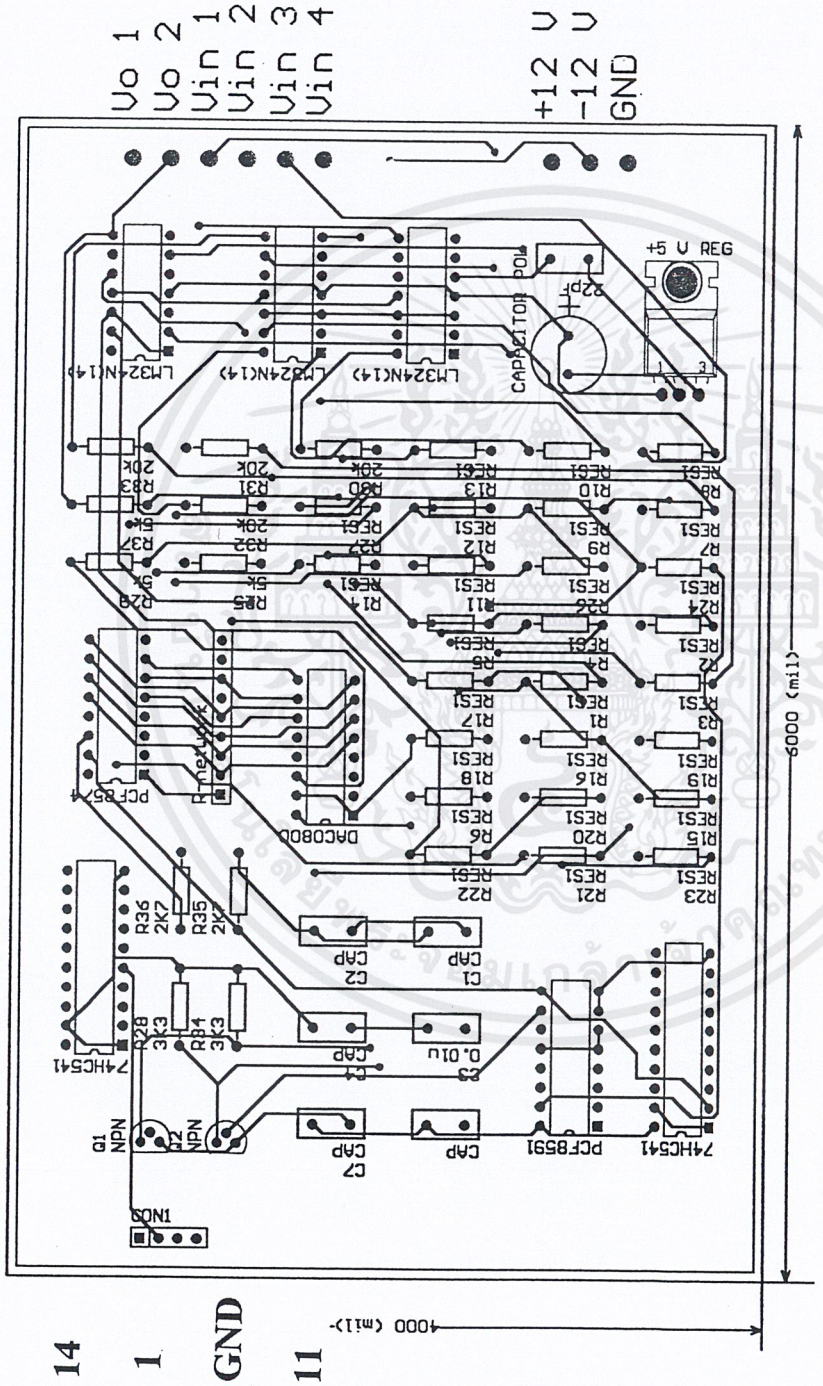


รูปที่ 3-10 แสดงการวางอุปกรณ์ในขั้นตอนการออกแบบ

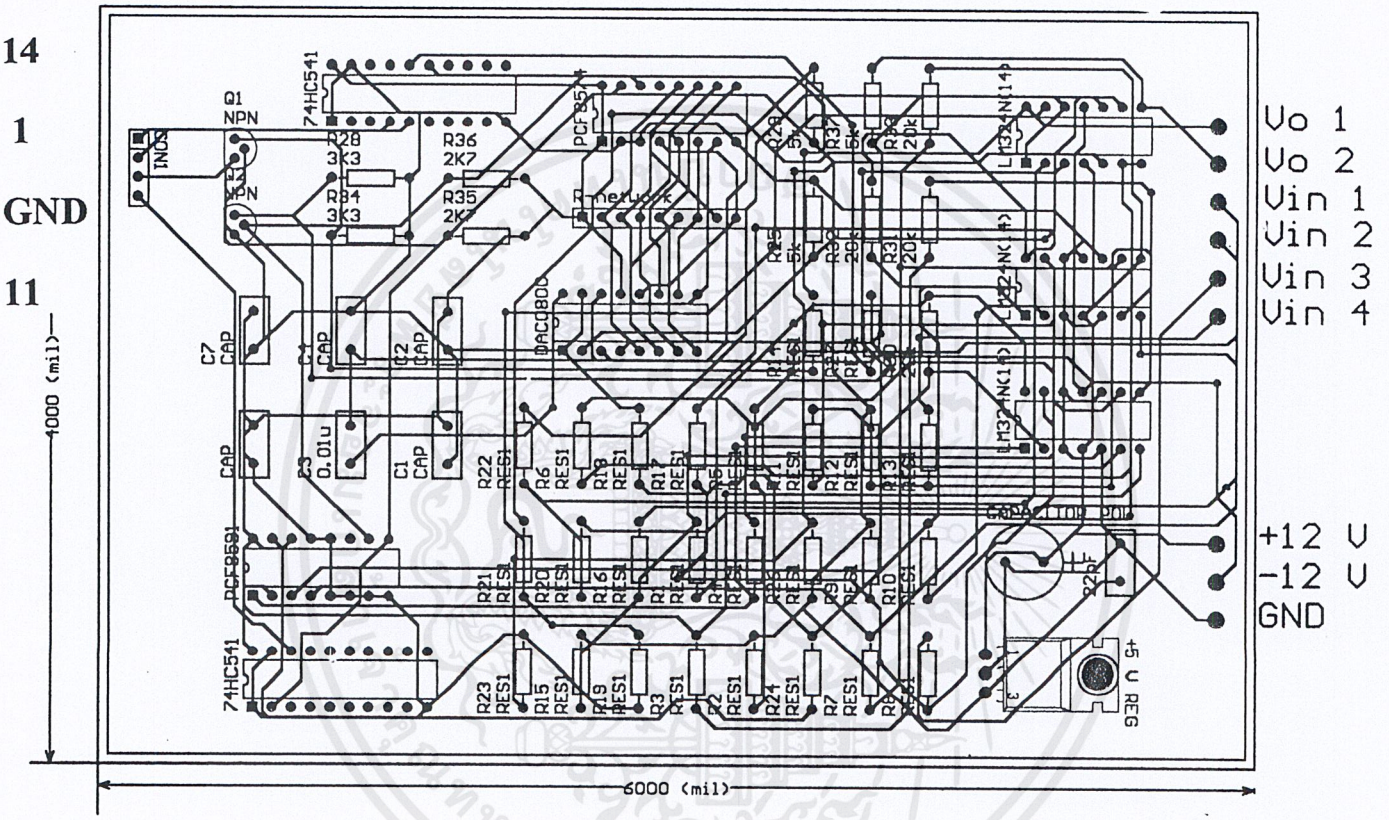
14 1 GND 11 FROM DB25



รูปที่ 3-11 แสดงลายปริ้นด้านบนในขั้นตอนการออกแบบ



รูปที่ 3-12 แสดงลายปริ้นด้านล่างในขั้นตอนการออกแบบ

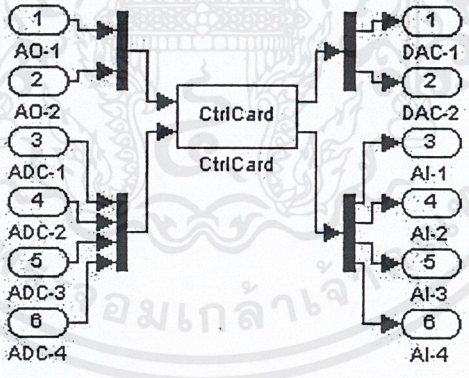


รูปที่ 3-13แสดงลายปริ๊นที่เสร็จสมบูรณ์ในขั้นตอนการออกแบบ

บทที่ 4
การทดลองและผลการทดลอง

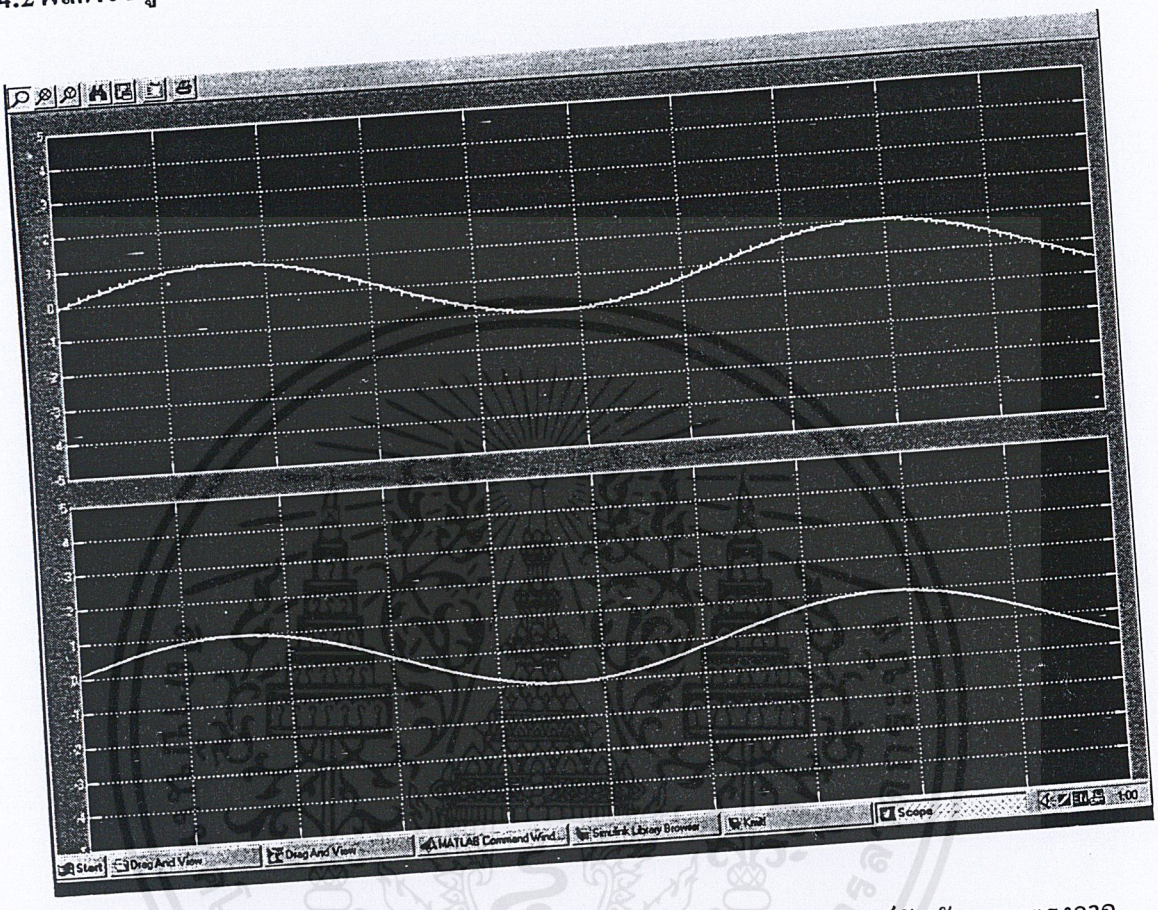
4.1 การทดลองการรับค่าสัญญาณ

การทดลองนี้เป็นการทดลอง รับค่าสัญญาณซายน์แบบโดยตรง กับสัญญาณที่ผ่านบล็อคเรียลไทม์เวอร์คชอพ แล้ว

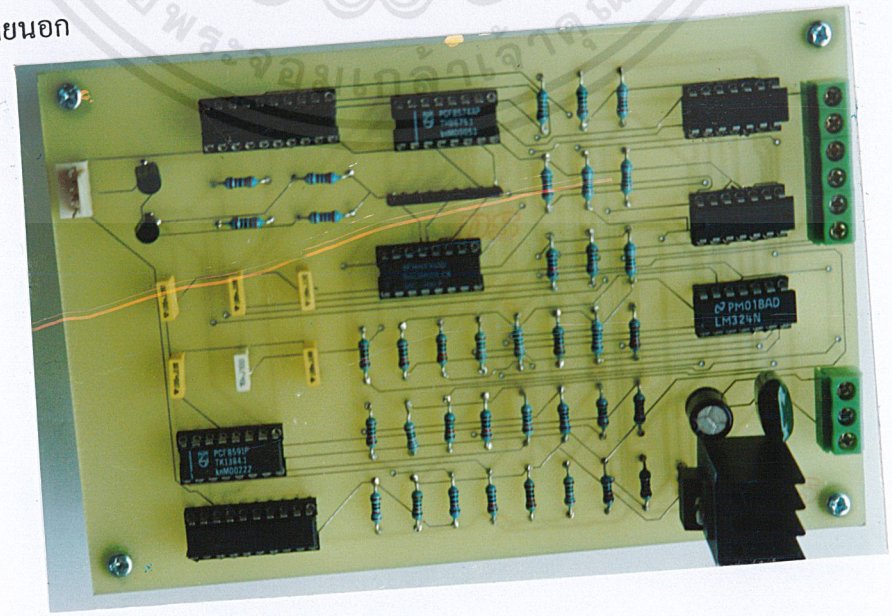


รูปที่ 4-1 แสดงบล็อคไดอะแกรมของตัวรับส่งค่าสัญญาณ

4.2 ผลการซิมูเลชั่นแบบเรียลไทม์ในโหมดภายนอก(external mode)



รูปที่4-2 กราฟแสดงการเปรียบเทียบระหว่างสัญญาณที่ผ่านบล็อคเรียลไทม์กับสัญญาณตรงจากระบบภายนอก



รูปที่ 4-3 ฮาร์ดแวร์ของการอินเตอร์เฟส

บทที่5
บทสรุปและบทวิจารณ์

5.1 สรุปผลการปฏิบัติงาน

โครงงานชุดเรียลไทม์เวอร์คชอพมีจุดประสงค์เพื่อรับสัญญาณจากระบบภายนอก เข้ามาแสดงผลในโมเดลของซิมูลิ้งค์ เพื่อนำไปวิเคราะห์แล้วนำไปใช้งานต่อไป

เริ่มต้นการทำงาน จากการออกแบบวงจรของการ์ดอินเตอร์เฟส โดยได้มีการปรึกษาอาจารย์ที่ปรึกษา ในการใช้วัสดุอุปกรณ์ต่างๆ หลังจากนั้นได้เริ่มศึกษาในส่วนของการเขียนโปรแกรมภาษาซีเพื่อเป็นพื้นฐานในการศึกษาวิธีการเขียน ซีเมค เอสฟังก์ชั่นในการติดต่อกับระบบภายนอกและแสดงผลของสัญญาณของระบบภายนอก

ผลที่เกิดขึ้นคือสามารถออกแบบการ์ดอินเตอร์เฟส และเขียนซอฟต์แวร์ที่ใช้ในการติดต่อนำสัญญาณจากภายนอกเข้ามาแสดงผลได้

5.2 ปัญหาในการทำงานและแนวทางแก้ไข

ในการดำเนินโครงงานนี้ประสบปัญหาหลายประการซึ่งได้มีการแก้ไขดังต่อไปนี้

1. ไอซีที่เลือกใช้ในการ์ดอินเตอร์เฟสเป็นไอซีที่มีขายและใช้น้อยมากในท้องตลาด จึงทำให้เสียเวลาในการสั่งจองล่วงหน้า ควรมีการแก้ไขโดย ใช้ไอซีที่เป็นไอซีพื้นฐานให้มากที่สุด

2.เป็นโครงงานใหม่ที่ค่อนข้างยากต่อการศึกษาและเข้าใจ ทำให้เกิดความเข้าใจผิดในเนื้อหาหรือมีการศึกษาที่ไม่ตรงประเด็นทำให้เกิดการเสียเวลาเป็นอย่างมาก ควรแก้ไขโดยการปรึกษากับอาจารย์หรือผู้รู้ให้มากขึ้น

5.3 ผลที่ได้รับจากการโครงงาน

จากการศึกษาทางทฤษฎีเพียงอย่างเดียวนั้นอาจไม่เพียงพอ เพราะเมื่อนำมาทดลองใช้งานจริงย่อมมีปัญหา และเงื่อนไขในสภาวะต่างๆมากมาย ซึ่งการที่จะประสบความสำเร็จนั้นบางครั้งต้องใช้ประสบการณ์ในการทำงานนั้นเป็นอย่างมาก หรือบางครั้งอาจได้จากการทดลองและนำมาวิเคราะห์ควบคู่กันไปกับแนวทางทฤษฎีที่ได้ศึกษา ซึ่งจากการทำโครงงานนี้ ทำให้ผู้ทำได้เรียนรู้สิ่งต่างๆมากมายมากขึ้น ซึ่งในบางส่วนอาจจะไม่พบในทางทฤษฎีเลย หรือบางส่วนของทฤษฎีก็เป็น

ความรู้ใหม่ที่ผู้ทำไม่เคยศึกษามาก่อน รวมถึงปัญหาต่างๆที่จะต้องมีการแก้ไขให้ถูกส่วนลุล่วงไปได้ ทำให้สามารถนำไปใช้ในการทำงานจริง หลังจากจบการศึกษา

ภาคผนวก ก.

```
/*
 *
 * Kmitl.c
 *
 *
 * Real-Time Workshop code generation for Simulink model "Kmitl.".
 *
 *
 * Model Version              : 1.65
 * RTW file version           : 3.0 $Date: 1999/09/30 23:33:29 $
 * RTW file generated on      : Tue Oct 31 10:48:10 2000
 * TLC version                : 1.0 (Jan 15 1999)
 * C source code generated on : Tue Oct 31 10:48:11 2000
 *
 *
 * Relevant TLC Options:
 *   InlineParameters = 0
 *   RollThreshold = 5
 *   CodeFormat = RealTime
 *
 * Simulink model settings:
 *   Solver     : FixedStep
 *   StartTime  : 0.0 s
 *   StopTime   : 999999.0 s
 *   FixedStep  : 0.0 s
 */

#include <math.h>
#include <string.h>
```

```
#include "Kmitl.h"
#include "Kmitl.prm"

real_T Kmitl_RGND = 0.0;              /* real_T ground */

/* Start the model */
void MdlStart(void)
{
  /* start code */

  /* Level2 S-Function Block: <S1>/CtrlCard (CtrlCard) */
  {
    SimStruct *rts = ssGetSFunction(rtS, 0);
    sfcnStart(rts);
  }

  /* Level2 S-Function Block: <S3>/CtrlPID (CtrlPID) */
  {
    SimStruct *rts = ssGetSFunction(rtS, 1);
    sfcnStart(rts);
  }

  /* state initialization */

  /* TransferFcn Block: <S2>/motor shaft */
  rtX.c.s2_motor_shaft[0] = 0.0;
  rtX.c.s2_motor_shaft[1] = 0.0;

  /* Derivative Block: <S2>/speed */
  rtRWork.s2_speed.TimeStampA = rtInf;
```

```
 rtRWork.s2_speed.TimeStampB = rtInf;

 /* TransferFcn Block: <S2>/armature circuit */
 rtX.c.s2_armature_circuit = 0.0;
}

/* Compute block outputs */
void MdlOutputs(int_T tid)
{
 /* local block i/o variables */
 real_T rtb_temp7;

 if (ssIsContinuousTask(rtS, tid)) {  /* Sample time: [0.0, 0.0] */
  /* Sin Block: <Root>/Sine Wave */
  rtB.root_Sine_Wave = rtP.root_Sine_Wave_Amp *
   sin(rtP.root_Sine_Wave_Freq * ssGetTaskTime(rtS,tid) + rtP.root_Sine_Wave_Phase);
 }

 if (ssIsSampleHit(rtS, 1, tid)) {   /* Sample time: [0.001, 0.0] */
  /* Level2 S-Function Block: <S1>/CtrlCard (CtrlCard) */
  {
   SimStruct *rts = ssGetSFunction(rtS, 0);
   sfcnOutputs(rts, tid);
  }

  /* Level2 S-Function Block: <S3>/CtrlPID (CtrlPID) */
  {
   SimStruct *rts = ssGetSFunction(rtS, 1);
   sfcnOutputs(rts, tid);
  }
```

```
}

if (ssIsContinuousTask(rtS, tid)) {  /* Sample time: [0.0, 0.0] */
  /* TransferFcn Block: <S2>/motor shaft */
  {
    rtB.s2_motor_shaft =
rtP.s2_motor_shaft_C[0]*rtX.c.s2_motor_shaft[0]+rtP.s2_motor_shaft_C[1]*rtX.c.s2_motor_shaf
t[1];
  }

  /* Derivative Block: <S2>/speed */
  {
    real_T t = ssGetTaskTime(rtS,tid);
    real_T timeStampA = rtRWork.s2_speed.TimeStampA;
    real_T timeStampB = rtRWork.s2_speed.TimeStampB;

    if (timeStampA >= t && timeStampB >= t) {
      rtb_temp7 = 0.0;
    } else {
      real_T deltaT;
      real_T *lastBank = &rtRWork.s2_speed.TimeStampA;
      if (timeStampA < timeStampB) {
        if (timeStampB < t) {
          lastBank += 2;
        }
      } else if (timeStampA >= t) {
        lastBank += 2;
      }
      deltaT = t - *lastBank++;
      rtb_temp7 = (rtB.s2_motor_shaft - *lastBank++) / deltaT;
```

```
    }
   }

   /* Gain Block: <S2>/back emf. */
   rtb_temp7 *= rtP.s2_back_emf_Gain;

   /* Sum Block: <S2>/Sum */
   rtB.s2_Sum = 0.0 - rtb_temp7;

   /* TransferFcn Block: <S2>/armature circuit */
   {
    rtb_temp7 = rtP.s2_armature_circuit_C*rtX.c.s2_armature_circuit;
   }

   /* Gain Block: <S2>/motor gain */
   rtb_temp7 *= rtP.s2_motor_gain_Gain;

   /* Sum Block: <S2>/Sum1 */
   rtB.s2_Sum1 = rtb_temp7 + 0.0;
  }
 }

/* Perform model update */
void MdlUpdate(int_T tid)
{
  if (ssIsContinuousTask(rtS, tid)) {   /* Sample time: [0.0, 0.0] */
    /* Derivative Block: <S2>/speed */
    {
      real_T timeStampA = rtRWork.s2_speed.TimeStampA;
      real_T timeStampB = rtRWork.s2_speed.TimeStampB;
```

```
    real_T *lastBank = &rtRWork.s2_speed.TimeStampA;

    if (timeStampA != rtInf) {
     if (timeStampB == rtInf) {
      lastBank += 2;
     } else if (timeStampA >= timeStampB) {
      lastBank += 2;
     }
    }
    *lastBank++ = ssGetTaskTime(rtS,tid);
    *lastBank++ = rtB.s2_motor_shaft;
   }
  }
}

/* Compute model derivatives */
void MdlDerivatives(void)
{
 /* TransferFcn Block: <S2>/motor shaft */
 {
  real_T *dx = ssGetdX(rtS)+0;

  dx[0] = rtB.s2_Sum1;
  dx[0] +=
rtP.s2_motor_shaft_A[0]*rtX.c.s2_motor_shaft[0]+rtP.s2_motor_shaft_A[1]*rtX.c.s2_motor_sha
ft[1];
  dx[1] = rtX.c.s2_motor_shaft[0];
 }

 /* TransferFcn Block: <S2>/armature circuit */
```

```
{
  real_T *dx = ssGetdX(rtS)+2;

  dx[0] = rtB.s2_Sum;
  dx[0] += rtP.s2_armature_circuit_A*rtX.c.s2_armature_circuit;
 }
}

/* Terminate function */
void MdlTerminate(void)
{
 /* Level2 S-Function Block: <S1>/CtrlCard (CtrlCard) */
 {
  SimStruct *rts = ssGetSFunction(rtS, 0);
  sfcnTerminate(rts);
 }

 /* Level2 S-Function Block: <S3>/CtrlPID (CtrlPID) */
 {
  SimStruct *rts = ssGetSFunction(rtS, 1);
  sfcnTerminate(rts);
 }
}

#include "Kmitl.reg"

/* [EOF] Kmitl.c */
```



```
/*
```

```
* Kmitl.h
*
* Real-Time Workshop code generation for Simulink model "Kmitl.mdl".
*
* Model Version              : 1.65
* RTW file version           : 3.0 $Date: 1999/09/30 23:33:29 $
* RTW file generated on      : Tue Oct 31 10:48:10 2000
* TLC version                : 1.0 (Jan 15 1999)
* C source code generated on : Tue Oct 31 10:48:11 2000
*
* Relevant TLC Options:
*   InlineParameters = 0
*   RollThreshold = 5
*   CodeFormat = RealTime
*
* Simulink model settings:
*   Solver     : FixedStep
*   StartTime  : 0.0 s
*   StopTime   : 999999.0 s
*   FixedStep  : 0.0 s
*/

#ifndef _RTW_HEADER_FILE_
#define _RTW_HEADER_FILE_

#ifndef TRUE
# define TRUE (1)
#endif
#ifndef FALSE
# define FALSE (0)
```

```
#endif

#define MODEL_NAME              Kmitl
#define NMODES                  (0)             /* Number of block mode elements */
#define NSAMPLE_TIMES           (2)             /* Number of sample times */
#define NINPUTS                 (0)             /* Number of model inputs */
#define NOUTPUTS                (0)             /* Number of model outputs */
#define NSTATES                 (3)             /* Number of states (total) */
#define NDSTATES                (0)             /* Number of discrete states */
#define NBLOCKIO                (7)             /* Number of data output port signals */
#define NUM_ZC_EVENTS           (0)             /* Number of zero-crossing events */
#define fcn_call_T              real_T

#ifndef NCSTATES
# define NCSTATES (3)           /* Number of continuous states */
#elif NCSTATES != 3
# error Invalid specification of NCSTATES defined in compiler command
#endif

#define assert(exp)

#include "simstruc.h"
#include "rtlibsrc.h"
#include "dt_info.h"
#include "Kmitl_export.h"

/* include Real-Time Windows Target header file */

#include "rtwintgt.h"
```

```
/*
 * Block I/O Structure
 *
 * Note: Individual field names are derived from the signal name when
 * present, otherwise, field names are derived from the source block name
 * with an optional port number appened to the block name if the block has
 * multiple output ports. The comment to the right of structure field
 * contains the signal source block name.
 *
 */

typedef struct BlockIO_tag {
  real_T root_Sine_Wave;          /* <Root>/Sine Wave */
  real_T s1_CtrlCard_o1[2];       /* <S1>/CtrlCard */
  real_T s1_CtrlCard_o2[4];       /* <S1>/CtrlCard */
  real_T s3_CtrlPID;              /* <S3>/CtrlPID */
  real_T s2_motor_shaft;          /* <S2>/motor shaft */
  real_T s2_Sum;                  /* <S2>/Sum */
  real_T s2_Sum1;                 /* <S2>/Sum1 */
} BlockIO;


/*
 * Default Parameters Structure
 *
 * Note: The parameters structure contains all the block parameters
 * in the model.  Individual field names are comprised of the block and
 * parameter name.
 *
 */
```

```
typedef struct Parameters_tag {
  real_T s1_CtrlCard_P6Size[2];       /* Computed Parameter: P6Size
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
                                       */
  real_T s1_CtrlCard_P6;              /* Computed Parameter: P6
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
                                       */
  real_T s1_CtrlCard_P7Size[2];       /* Computed Parameter: P7Size
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
                                       */
  real_T s1_CtrlCard_P7;              /* Computed Parameter: P7
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
                                       */
  real_T s1_CtrlCard_P8Size[2];       /* Computed Parameter: P8Size
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
                                       */
  real_T s1_CtrlCard_P8;              /* Computed Parameter: P8
                                       * External Mode Tunable: yes
                                       * Referenced by block:
                                       * <S1>/CtrlCard
```

```
                        */
real_T s1_CtrlCard_P9Size[2];       /* Computed Parameter: P9Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P9;              /* Computed Parameter: P9
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P10Size[2];      /* Computed Parameter: P10Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_0P10;            /* Computed Parameter: P10
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P1Size[2];       /* Computed Parameter: P1Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P1;              /* Computed Parameter: P1
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
```

```
                        */
real_T s1_CtrlCard_P2Size[2];       /* Computed Parameter: P2Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P2[2];           /* Computed Parameter: P2
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P3Size[2];       /* Computed Parameter: P3Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P3[2];           /* Computed Parameter: P3
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P4Size[2];       /* Computed Parameter: P4Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P4[2];           /* Computed Parameter: P4
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
```

```
                        */
real_T s1_CtrlCard_P5Size[2];       /* Computed Parameter: P5Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s1_CtrlCard_P5[2];           /* Computed Parameter: P5
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S1>/CtrlCard
                        */
real_T s2_armature_circuit_A;       /* Computed Parameter: A
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S2>/armature circuit
                        */

real_T s2_armature_circuit_C;       /* Computed Parameter: C
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S2>/armature circuit
                        */

real_T s2_back_emf_Gain;            /* Variable: s2_back_emf_Gain
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S2>/back emf.
                        */
real_T s2_motor_gain_Gain;          /* Variable: s2_motor_gain_Gain
                        * External Mode Tunable: yes
```

```
                        * Referenced by block:
                        * <S2>/motor gain
                        */
real_T s2_motor_shaft_A[2];         /* Computed Parameter: A
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S2>/motor shaft
                        */

real_T s2_motor_shaft_C[2];         /* Computed Parameter: C
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S2>/motor shaft
                        */

real_T s3_CtrlPID_P1Size[2];        /* Computed Parameter: P1Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S3>/CtrlPID
                        */
real_T s3_CtrlPID_P1;               /* Computed Parameter: P1
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S3>/CtrlPID
                        */
real_T s3_CtrlPID_P2Size[2];        /* Computed Parameter: P2Size
                        * External Mode Tunable: yes
                        * Referenced by block:
                        * <S3>/CtrlPID
                        */
```

```
real_T s3_CtrlPID_P2[2];          /* Computed Parameter: P2
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <S3>/CtrlPID
                                   */
real_T s3_CtrlPID_P3Size[2];      /* Computed Parameter: P3Size
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <S3>/CtrlPID
                                   */
real_T s3_CtrlPID_P3;             /* Computed Parameter: P3
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <S3>/CtrlPID
                                   */
real_T s3_CtrlPID_P4Size[2];      /* Computed Parameter: P4Size
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <S3>/CtrlPID
                                   */
real_T s3_CtrlPID_P4;             /* Computed Parameter: P4
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <S3>/CtrlPID
                                   */
real_T root_Sine_Wave_Amp;        /* Expression: 1
                                   * External Mode Tunable: yes
                                   * Referenced by block:
                                   * <Root>/Sine Wave
                                   */
```

```
real_T root_Sine_Wave_Freq;          /* Expression: 1
                                     * External Mode Tunable: yes
                                     * Referenced by block:
                                     * <Root>/Sine Wave
                                     */
real_T root_Sine_Wave_Phase;         /* Expression: 0
                                     * External Mode Tunable: yes
                                     * Referenced by block:
                                     * <Root>/Sine Wave
                                     */
} Parameters;

/*
 * States Structure
 *
 * Note: Individual field names are derived from the block name.
 *
 */

 /* continuous states */
 typedef struct States_Cont_tag {
  real_T s2_motor_shaft[2];
  real_T s2_armature_circuit;
 } States_Cont;

 typedef struct States_tag {
  States_Cont c;
 } States;

 /*
```

```
 * State Derivatives Structure
 *
 * Note: Individual field names are derived from the block name.
 *
 */
typedef struct StateDerivatives_tag {
 real_T s2_motor_shaft[2];
 real_T s2_armature_circuit;
} StateDerivatives;

/*
 * Real-Work (RWork) Structure
 *
 * Note: Individual field names are derived from either 1) the
 * block's DefineRWork function, 2) Simulink's RWorkDefine
 * record, or 3) the block name.
 *
 */

typedef struct R_Work_tag {
 struct {
   real_T TimeStampA;
   real_T LastUAtTimeA;
   real_T TimeStampB;
   real_T LastUAtTimeB;
 } s2_speed;
} R_Work;

/*
 * Pointer-Work (PWork) Structure
```

```
 *
 * Note: Individual field names are derived from either 1) the block's
 * DefinePWork function, 2) Simulink's PWorkDefine record, or
 * 3) the block name.
 *
 */

typedef struct P_Work_tag {
  struct {
    void *LoggedData;
  } root_Scope;
} P_Work;

/* non-finites */
extern real_T rtInf;

#if defined(MULTITASKING)
# error Model (Kmitl) was built in \
  SingleTasking solver mode, however the MULTITASKING define is \
  present. If you have multitasking (e.g. -DMT or -DMULTITASKING) \
  defined on the RTW page of Simulation parameter dialog, please \
  remove it and on the Solver page, select solver mode \
  MultiTasking. If the Simulation parameter dialog is configured \
  correctly, please verify that your template makefile is \
  configured correctly.
#endif

/*
 * Comments are included throughout the generated code which allow you to
 * trace directly back to the approprate location in model. The basic format
```

```
* is <system>/block_name, where system is the system number (uniquely assigned
* by Simulink) and block_name is the name of the block.
*
* Use the MATLAB open_system command to trace the generated code back
* to the model.  For example,
*
* open_system('<S3>')    - opens system 3
* open_system('<S3>/Kp') - opens and selects block Kp which resides in S3
*
* Here is the system hierarchy for this model
*
* <Root> : Kmitl
* <S1>   : Kmitl/KMITL CtrlBoard
* <S2>   : Kmitl/KMITL MotorSim
* <S3>   : Kmitl/KMITL PID
*/

#endif                                 /* _RTW_HEADER_FILE_ */

/* [EOF] Km



/*
 * Kmitl_export.h
 *
 * Real-Time Workshop code generation for Simulink model "Kmitl.mdl".
 *
 * Model Version              : 1.65
 * RTW file version           : 3.0 $Date: 1999/09/30 23:33:29 $
 * RTW file generated on      : Tue Oct 31 10:48:10 2000
```

```
 * TLC version          : 1.0 (Jan 15 1999)
 * C source code generated on : Tue Oct 31 10:48:11 2000
 *
 * Relevant TLC Options:
 *   InlineParameters = 0
 *   RollThreshold = 5
 *   CodeFormat = RealTime
 *
 * Simulink model settings:
 *   Solver     : FixedStep
 *   StartTime  : 0.0 s
 *   StopTime   : 999999.0 s
 *   FixedStep  : 0.0 s
 */

#ifndef _RTW_EXPORT_Kmitl_
#define _RTW_EXPORT_Kmitl_

#endif                          /* _RTW_EXPORT_Kmitl */

/* [EOF] Kmitl_export.h */



/*
 * File : timestwo.c
 * Abstract:
 *      An example C-file S-function for multiplying an input by 2,
 *      y = 2*u
 *
```

```
 * See simulink/src/sfuntmpl.doc
 *
 * Copyright (c) 1990-1998 by The MathWorks, Inc. All Rights Reserved.
 * $Revision: 1.3 $
 */


#define S_FUNCTION_NAME  timestwo
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"


/* Function: mdlInitializeSizes ===========================================
 * Abstract:
 *   Setup sizes of the various vectors.
 */
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      return; /* Parameter mismatch will be reported by Simulink */
   }

   if (!ssSetNumInputPorts(S, 1)) return;
   ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
   ssSetInputPortDirectFeedThrough(S, 0, 1);

   if (!ssSetNumOutputPorts(S,1)) return;
   ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
```

```
    ssSetNumSampleTimes(S, 1);

    /* Take care when specifying exception free code - see sfuntmpl.doc */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *   Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
  ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
  ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs
=======================================================
 * Abstract:
 *   y = 2*u
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
  int_T             i;
  InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
  real_T            *y    = ssGetOutputPortRealSignal(S,0);
  int_T             width = ssGetOutputPortWidth(S,0);
```

```
    for (i=0; i<width; i++) {
        *y++ = 2.0 *(*uPtrs[i]);
    }
}


/* Function: mdlTerminate
=============================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
}


#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

โปรแกรมของการติดต่อการ์ดอินเตอร์เฟส ctrlcard.c

```
/* $Date: 2000/10/03
 * $Revision: 1.0
 * $Author: Suthian KIATSUNTHORN
 *
 * File: CtrlCard.c
 *
```

```
 * Abstract:
 *      S-function device driver for the analog input/output interface card
 *      of Control Engineering Department, KMITL.
 *
 */

#define S_FUNCTION_NAME CtrlCard
#define S_FUNCTION_LEVEL 2

#include  <stdlib.h>     /* malloc(), free(), strtoul() */
#include "C:\matlabr11\simulink\include\simstruc.h"    /* the simstruct access macros */


/*=================================================================
======*
 * System Parameters

*=================================================================
====*/
#define SAMPLE_TIME_BASE                0.0001      /* 0.1 msec. */

/*=================================================================
======*
 * Number of S-function Parameters and macros to access from the SimStruct *

*=================================================================
=====*/
#define NUM_PARAMS              (10)
#define PRINTER_PORT_PARAM      (ssGetSFcnParam(S,0))
#define ANALOG_INPUT_PARAM      (ssGetSFcnParam(S,1))
```

```
#define DIGITAL_INPUT_PARAM        (ssGetSFcnParam(S,2))
#define ANALOG_OUTPUT_PARAM        (ssGetSFcnParam(S,3))
#define DIGITAL_OUTPUT_PARAM       (ssGetSFcnParam(S,4))
#define SAMPLE_TIME_PARAM          (ssGetSFcnParam(S,5))
#define MODEL_STEP_SIZE_PARAM      (ssGetSFcnParam(S,6))
#define HARDWARE_INTERFACE         (ssGetSFcnParam(S,7))
#define HOLD_DEVICE_1              (ssGetSFcnParam(S,8))
#define HOLD_DEVICE_2              (ssGetSFcnParam(S,9))


/*======================*
 * S-function User Data *
 *======================*/

typedef struct {
   uint_T   printer_port;
   real_T   adc_analog_min[4];
   real_T   adc_analog_max[4];
   int_T    adc_digital_min[4];
   int_T    adc_digital_max[4];
   real_T   dac_analog_min[2];
   real_T   dac_analog_max[2];
   int_T    dac_digital_min[2];
   int_T    dac_digital_max[2];
   uint_T   sample_time;
   uint_T   model_step_count;
   uint_T   model_step_size;
   int_T    hardware_interface;
   real_T   last_signal[2];
   real_T   next_signal[2];
```

```
    int_T     hold_device[2];
} CONTROL_CARD_Info;



/*====================*
 * S-function methods *
 *====================*/


#define MDL_CHECK_PARAMETERS   /* Change to #undef to remove function */
#if defined(MDL_CHECK_PARAMETERS)
/*
 Function: mdlCheckParameters
=============================================
*/
static void mdlCheckParameters(SimStruct *S)
{
     static char_T errMsg[256];
    boolean_T allParamsOK = 1;
      int number;
    /*
     * ADC Analog Range
     */
    number = mxGetNumberOfElements(ANALOG_INPUT_PARAM);
    if (allParamsOK==1 && number!=2 && number!=4 && number!=6 && number!=8) {
       strcpy(errMsg,"ADC Analog Range must be 2,4,6 or 8 elements vector
[min1,max1,min2,max2,...,min4,max4].\n");
      allParamsOK = 0;
    }
```

```
    if (allParamsOK==1 && number>=2 && ((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[0])>=((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[1])) {
      strcpy(errMsg,"The specified ADC Analog Range #1 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number>=4 && ((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[2])>=((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[3])) {
      strcpy(errMsg,"The specified ADC Analog Range #2 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number>=6 && ((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[4])>=((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[5])) {
      strcpy(errMsg,"The specified ADC Analog Range #3 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number==8 && ((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[6])>=((real_T)
 mxGetPr(ANALOG_INPUT_PARAM)[7])) {
      strcpy(errMsg,"The specified ADC Analog Range #4 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    /*
     * ADC Digital Range
     */
    number = mxGetNumberOfElements(DIGITAL_INPUT_PARAM);
    if (allParamsOK==1 && number!=2 && number!=4 && number!=6 && number!=8) {
```

```
    strcpy(errMsg,"ADC Digital Range must be 2,4,6 or 8 elements vector
[min1,max1,min2,max2,...,min4,max4].\n");
    allParamsOK = 0;
   }
   if (allParamsOK==1 && number>=2 && ((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[0])>=((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[1])) {
     strcpy(errMsg,"The specified ADC Digital Range #1 is not supported by I/O board.\n");
     allParamsOK = 0;
   }
   if (allParamsOK==1 && number>=4 && ((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[2])>=((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[3])) {
     strcpy(errMsg,"The specified ADC Digital Range #2 is not supported by I/O board.\n");
     allParamsOK = 0;
   }
   if (allParamsOK==1 && number>=6 && ((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[4])>=((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[5])) {
     strcpy(errMsg,"The specified ADC Digital Range #2 is not supported by I/O board.\n");
     allParamsOK = 0;
   }
   if (allParamsOK==1 && number>=8 && ((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[6])>=((int_T)
mxGetPr(DIGITAL_INPUT_PARAM)[7])) {
     strcpy(errMsg,"The specified ADC Digital Range #2 is not supported by I/O board.\n");
     allParamsOK = 0;
   }
   /*
    * DAC Analog Range
```

```
    */
    number = mxGetNumberOfElements(ANALOG_OUTPUT_PARAM);
    if (allParamsOK==1 && number!=2 && number!=4) {
      strcpy(errMsg,"DAC Analog Range must be 2 or 4 elements vector
[min1,max1,min2,max2].\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number>=2 && ((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[0])>=((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[1])) {
      strcpy(errMsg,"The specified DAC Analog Range #1 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number==4 && ((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[2])>=((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[3])) {
      strcpy(errMsg,"The specified DAC Analog Range #2 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    /*
     * DAC Digital Range
     */
    number = mxGetNumberOfElements(DIGITAL_OUTPUT_PARAM);
    if (allParamsOK==1 && number!=2 && number!=4) {
      strcpy(errMsg,"DAC Digital Range must be 2 or 4 elements vector
[min1,max1,min2,max2].\n");
      allParamsOK = 0;
    }
```

```
    if (allParamsOK==1 && number>=2 && ((int_T)
   mxGetPr(DIGITAL_OUTPUT_PARAM)[0])>=((int_T)
   mxGetPr(DIGITAL_OUTPUT_PARAM)[1])) {
      strcpy(errMsg,"The specified DAC Digital Range #1 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    if (allParamsOK==1 && number==4 && ((int_T)
  mxGetPr(DIGITAL_OUTPUT_PARAM)[2])>=((int_T)
  mxGetPr(DIGITAL_OUTPUT_PARAM)[3])) {
      strcpy(errMsg,"The specified DAC Digital Range #2 is not supported by I/O board.\n");
      allParamsOK = 0;
    }
    /*
     * Sample Time
     */
    if (allParamsOK==1 && mxGetNumberOfElements(SAMPLE_TIME_PARAM)!=1) {
     strcpy(errMsg,"Sample Time must be a scalar.\n");
     allParamsOK = 0;
   }
    if (allParamsOK==1 && ((uint_T) mxGetPr(SAMPLE_TIME_PARAM)[0])<=0) {
    strcpy(errMsg,"Sample Time must be a positive number.\n");
    allParamsOK = 0;
  }
  /*
   * Model Step Size
   */
  if (allParamsOK==1 && mxGetNumberOfElements(MODEL_STEP_SIZE_PARAM)!=1) {
   strcpy(errMsg,"Model Step Size must be a scalar.\n");
   allParamsOK = 0;
  }
```

```
    if (allParamsOK==1 && ((uint_T) mxGetPr(MODEL_STEP_SIZE_PARAM)[0])<=0) {
     strcpy(errMsg,"Model Step Size must be a positive number.\n");
     allParamsOK = 0;
    }
    if (!allParamsOK) {
     ssSetErrorStatus(S, errMsg);
    }
}
#endif /* MDL_CHECK_PARAMETERS */


/*
 Function: mdlInitializeSizes ===============================================
 */
static void mdlInitializeSizes(SimStruct *S)
{    int_T i;
    ssSetNumSFcnParams(S, NUM_PARAMS);
    if (ssGetNumSFcnParams(S)==ssGetSFcnParamsCount(S)) {
     mdlCheckParameters(S);
     if (ssGetErrorStatus(S)!=NULL) {
      return;       /* Error reported in mdlCheckParameters */
     }
    } else {
     return; /* Parameter mismatch will be reported by Simulink */
    }

    /* None of this s-functions's parameters are tunable during simulation */
    for (i=0;i<NUM_PARAMS;i++) {
     ssSetSFcnParamNotTunable(S, i);
    }
```

```
    ssSetNumSampleTimes(S,1);
    ssSetNumInputPorts(S,2);
    ssSetInputPortWidth(S,0,2);
    ssSetInputPortWidth(S,1,4);
    ssSetNumOutputPorts(S,2);
    ssSetOutputPortWidth(S,0,2);
    ssSetOutputPortWidth(S,1,4);
    ssSetInputPortDirectFeedThrough(S,0,1);
    ssSetInputPortDirectFeedThrough(S,1,1);
}



/*
 Function: mdlInitializeSampleTimes ========================================
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{   uint_T ts;
    ts = ((real_T) mxGetPr(MODEL_STEP_SIZE_PARAM)[0]);
   ssSetSampleTime(S,0,SAMPLE_TIME_BASE*ts);
   ssSetOffsetTime(S,0,0.0);
}



#define MDL_START                /* Change to #undef to remove function */
#if defined(MDL_START)
/* Function: mdlStart
=============================================================
 *
```

```
 */
static void mdlStart(SimStruct *S)
{    int_T num;
   CONTROL_CARD_Info   *CtrlCardInfo   = ssGetUserData(S);
  /*
   * Initialize Control Card Info (pointer saved in the user data)
   */
   if (CtrlCardInfo != NULL) {
    free(CtrlCardInfo);
   }
   if ((CtrlCardInfo = malloc(sizeof(CONTROL_CARD_Info))) == NULL) {
    ssSetErrorStatus(S,"Control Card Info Memory Allocation Error\n");
    return;
   }

    num = ((uint_T) mxGetPr(PRINTER_PORT_PARAM)[0]);
    switch (num) {
      case 1:
        CtrlCardInfo->printer_port = 0x0378;
        break;
      case 2:
        CtrlCardInfo->printer_port = 0x03bc;
        break;
      case 3:
        CtrlCardInfo->printer_port = 0x0278;
    }

    num = mxGetNumberOfElements(ANALOG_INPUT_PARAM);
    CtrlCardInfo->adc_analog_min[0] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[0]);
    CtrlCardInfo->adc_analog_max[0] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[1]);
```

```
if (num>2) {
  CtrlCardInfo->adc_analog_min[1] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[2]);
  CtrlCardInfo->adc_analog_max[1] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[3]);
} else {
  CtrlCardInfo->adc_analog_min[1] = CtrlCardInfo->adc_analog_min[0];
  CtrlCardInfo->adc_analog_max[1] = CtrlCardInfo->adc_analog_max[0];
}
if (num>4) {
  CtrlCardInfo->adc_analog_min[2] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[4]);
  CtrlCardInfo->adc_analog_max[2] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[5]);
} else {
  CtrlCardInfo->adc_analog_min[2] = CtrlCardInfo->adc_analog_min[1];
  CtrlCardInfo->adc_analog_max[2] = CtrlCardInfo->adc_analog_max[1];
}
if (num>6) {
  CtrlCardInfo->adc_analog_min[3] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[6]);
  CtrlCardInfo->adc_analog_max[3] = ((real_T) mxGetPr(ANALOG_INPUT_PARAM)[7]);
} else {
  CtrlCardInfo->adc_analog_min[3] = CtrlCardInfo->adc_analog_min[2];
  CtrlCardInfo->adc_analog_max[3] = CtrlCardInfo->adc_analog_max[2];
}

num = mxGetNumberOfElements(DIGITAL_INPUT_PARAM);
CtrlCardInfo->adc_digital_min[0] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[0]);
CtrlCardInfo->adc_digital_max[0] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[1]);
if (num>2) {
  CtrlCardInfo->adc_digital_min[1] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[2]);
  CtrlCardInfo->adc_digital_max[1] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[3]);
} else {
  CtrlCardInfo->adc_digital_min[1] = CtrlCardInfo->adc_digital_min[0];
```

```
        CtrlCardInfo->adc_digital_max[1] = CtrlCardInfo->adc_digital_max[0];
       }
       if (num>4) {
        CtrlCardInfo->adc_digital_min[2] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[4]);
        CtrlCardInfo->adc_digital_max[2] = ((int_T) mxGetPr(DIGITAL_INPUT_PARAM)[5]);
       } else {
        CtrlCardInfo->adc_digital_min[2] = CtrlCardInfo->adc_digital_min[1];
        CtrlCardInfo->adc_digital_max[2] = CtrlCardInfo->adc_digital_max[1];
       }
       if (num>6) {
        CtrlCardInfo->adc_digital_min[3] = ((real_T) mxGetPr(DIGITAL_INPUT_PARAM)[6]);
        CtrlCardInfo->adc_digital_max[3] = ((real_T) mxGetPr(DIGITAL_INPUT_PARAM)[7]);
       } else {
        CtrlCardInfo->adc_digital_min[3] = CtrlCardInfo->adc_digital_min[2];
        CtrlCardInfo->adc_digital_max[3] = CtrlCardInfo->adc_digital_max[2];
       }

      num = mxGetNumberOfElements(ANALOG_OUTPUT_PARAM);
      CtrlCardInfo->dac_analog_min[0] = ((real_T) mxGetPr(ANALOG_OUTPUT_PARAM)[0]);
      CtrlCardInfo->dac_analog_max[0] = ((real_T) mxGetPr(ANALOG_OUTPUT_PARAM)[1]);
      if (num>2) {
       CtrlCardInfo->dac_analog_min[1] = ((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[2]);
       CtrlCardInfo->dac_analog_max[1] = ((real_T)
mxGetPr(ANALOG_OUTPUT_PARAM)[3]);
      } else {
       CtrlCardInfo->dac_analog_min[1] = CtrlCardInfo->dac_analog_min[0];
       CtrlCardInfo->dac_analog_max[1] = CtrlCardInfo->dac_analog_max[0];
      }
```

```
    num = mxGetNumberOfElements(DIGITAL_OUTPUT_PARAM);
    CtrlCardInfo->dac_digital_min[0] = ((int_T) mxGetPr(DIGITAL_OUTPUT_PARAM)[0]);
    CtrlCardInfo->dac_digital_max[0] = ((int_T) mxGetPr(DIGITAL_OUTPUT_PARAM)[1]);
    if (num>2) {
     CtrlCardInfo->dac_digital_min[1] = ((int_T) mxGetPr(DIGITAL_OUTPUT_PARAM)[2]);
     CtrlCardInfo->dac_digital_max[1] = ((int_T) mxGetPr(DIGITAL_OUTPUT_PARAM)[3]);
    } else {
     CtrlCardInfo->dac_digital_min[1] = CtrlCardInfo->dac_digital_min[0];
     CtrlCardInfo->dac_digital_max[1] = CtrlCardInfo->dac_digital_max[0];
    }
    CtrlCardInfo->sample_time = ((uint_T) mxGetPr(SAMPLE_TIME_PARAM)[0]);
    if (CtrlCardInfo->sample_time==0) CtrlCardInfo->sample_time = 1;
    CtrlCardInfo->model_step_size = ((uint_T) mxGetPr(MODEL_STEP_SIZE_PARAM)[0]);
    if (CtrlCardInfo->model_step_size==0) CtrlCardInfo->model_step_size = 1;
    CtrlCardInfo->hardware_interface = ((int_T) mxGetPr(HARDWARE_INTERFACE)[0]);
    CtrlCardInfo->hold_device[0] = ((int_T) mxGetPr(HOLD_DEVICE_1)[0]);
    CtrlCardInfo->hold_device[1] = ((int_T) mxGetPr(HOLD_DEVICE_2)[0]);

  CtrlCardInfo->model_step_count= CtrlCardInfo->sample_time;
    CtrlCardInfo->last_signal[0] = 0.0;
    CtrlCardInfo->last_signal[1] = 0.0;
    CtrlCardInfo->next_signal[0] = 0.0;
    CtrlCardInfo->next_signal[1] = 0.0;

  ssSetUserData(S, (void*) CtrlCardInfo);

#if defined(MATLAB_MEX_FILE)
    if (CtrlCardInfo->hardware_interface==0 || ssGetSimMode(S) ==
SS_SIMMODE_NORMAL) {
```

```
    mexPrintf("\nControl Ladkrabang Analog Input/Output Interface Card: Hardware Access
Disabled\n");
   }
#endif

}
#endif /* MDL_START */



/*
 * Function: mdlOutputs
=============================================================
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{   int_T  hardware,num,ai,ao;
   real_T analog_min,analog_max;
   real_T digital_min,digital_max;
     real_T signal,dac,adc;
     real_T last,range,dt;
   real_T                *DAC = ssGetOutputPortRealSignal(S,0);
     real_T                 *AI = ssGetOutputPortRealSignal(S,1);
   InputRealPtrsType          AO = ssGetInputPortRealSignalPtrs(S,0);
   InputRealPtrsType          ADC = ssGetInputPortRealSignalPtrs(S,1);
   CONTROL_CARD_Info   *CtrlCardInfo = ssGetUserData(S);

     if (CtrlCardInfo->hardware_interface!=0 && ssGetSimMode(S) !=
SS_SIMMODE_NORMAL) {
    hardware = 1;
   } else {
```

```
 hardware = 0;
}
if (CtrlCardInfo->sample_time>1) {
    CtrlCardInfo->model_step_count += 1;
    if (CtrlCardInfo->model_step_count>=CtrlCardInfo->sample_time) {
     CtrlCardInfo->model_step_count = 0;
    }
  }
  for (num=0;num<1;num++) {
  analog_min = CtrlCardInfo->dac_analog_min[num];
    analog_max = CtrlCardInfo->dac_analog_max[num];
  digital_min = (real_T)(CtrlCardInfo->dac_digital_min[num]);
    digital_max = (real_T)(CtrlCardInfo->dac_digital_max[num]);
    signal = *AO[num];
/*
 * hold device
 */
    if (CtrlCardInfo->sample_time>1) {
        switch (CtrlCardInfo->hold_device[num]) {
          case 1:
             if (CtrlCardInfo->model_step_count>0) {
               signal = 0.0;
             }
             break;
          case 2:
            if (CtrlCardInfo->model_step_count==0) {
               CtrlCardInfo->last_signal[num] = signal;
             } else {
               signal = CtrlCardInfo->last_signal[num];
             }
```

```
                break;
            case 3:
              if (CtrlCardInfo->model_step_count==0) {
                CtrlCardInfo->last_signal[num] = CtrlCardInfo->next_signal[num];
                CtrlCardInfo->next_signal[num] = signal;
              } else {
                last = CtrlCardInfo->last_signal[num];
                range = CtrlCardInfo->next_signal[num]-CtrlCardInfo->last_signal[num];
                dt = ((real_T)(CtrlCardInfo->model_step_count))/((real_T)(CtrlCardInfo->sample_time));
                signal = last+range*dt;
              }
            }
        }
    /*
     * end of hold device
     */
        if (signal>analog_max) signal = analog_max;
        if (signal<analog_min) signal = analog_min;
        ao = (int_T)(digital_min+(digital_max-digital_min)*(signal-analog_min)/(analog_max-analog_min));
        if (ao>digital_max) ao = digital_max;
        if (ao<digital_min) ao = digital_min;
        if (hardware==1) {
/*          DACSetOutput(num,ao); */
        }
        dac = analog_min+(((real_T)(ao))-digital_min)*(analog_max-analog_min)/(digital_max-digital_min);
        if (dac>analog_max) dac = analog_max;
        if (dac<analog_min) dac = analog_min;
```

```
        DAC[num] = dac;
        DAC[1] = 0;
       }

       for (num=0;num<4;num++) {
      analog_min = CtrlCardInfo->adc_analog_min[num];
       analog_max = CtrlCardInfo->adc_analog_max[num];
     digital_min = (real_T)(CtrlCardInfo->adc_digital_min[num]);
       digital_max = (real_T)(CtrlCardInfo->adc_digital_max[num]);
       if (hardware==1) {
/*        ai = ADCConversion(num);  */
       } else {
          signal = *ADC[num];
        if (signal>analog_max) signal = analog_max;
        if (signal<analog_min) signal = analog_min;
          ai = (int_T)(digital_min+(digital_max-digital_min)*(signal-analog_min)/(analog_max-
analog_min));
       }
       if (ai>digital_max) ai = digital_max;
       if (ai<digital_min) ai = digital_min;
       adc = analog_min+(((real_T)(ai))-digital_min)*(analog_max-analog_min)/(digital_max-
digital_min);
       if (adc>analog_max) adc = analog_max;
       if (adc<analog_min) adc = analog_min;
       AI[num] = adc;
     }
     if (CtrlCardInfo->sample_time>1) {
     ssSetUserData(S, (void*) CtrlCardInfo);
     }
}
```

```
/* Function: mdlTerminate
=============================================================
 *
 */
static void mdlTerminate(SimStruct *S)
{
   CONTROL_CARD_Info *CtrlCardInfo   = ssGetUserData(S);
   free(CtrlCardInfo);
   ssSetUserData(S,NULL);
}



#undef MDL_RTW  /* Change to #undef to remove function */
#if defined(MDL_RTW) && (defined(MATLAB_MEX_FILE) || defined(NRT))
static void mdlRTW(SimStruct *S)
{
}
#endif /* MDL_RTW */
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "C:\matlabr11\simulink\include\simulink.c"      /* MEX-file interface mechanism */
#else
#include "c:\matlabr11\rtw\c\src\cg_sfun.h"       /* Code generation registration function */
#endif
```

ภาคผนวก ข.

8-bit A/D and D/A converter PCF8591

1 FEATURES

- Single power supply
- Operating supply voltage 2.5 V to 6 V
- Low standby current
- Serial input/output via I^2C-bus
- Address by 3 hardware address pins
- Sampling rate given by I^2C-bus speed
- 4 analog inputs programmable as single-ended or differential inputs
- Auto-incremented channel selection
- Analog voltage range from V_{SS} to V_{DD}
- On-chip track and hold circuit
- 8-bit successive approximation A/D conversion
- Multiplying DAC with one analog output.

2 APPLICATIONS

- Closed loop control systems
- Low power converter for remote data acquisition
- Battery operated equipment
- Acquisition of analog values in automotive, audio and TV applications.

3 GENERAL DESCRIPTION

The PCF8591 is a single-chip, single-supply low power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I^2C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I^2C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I^2C-bus.

The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I^2C-bus.

4 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCA8591P	DIP16	plastic dual in-line package; 16 leads (300 mil); long body	SOT38-1
PCA8591T	SO16	plastic small outline package; 16 leads; body width 7.5 mm	SOT162-1

5 BLOCK DIAGRAM

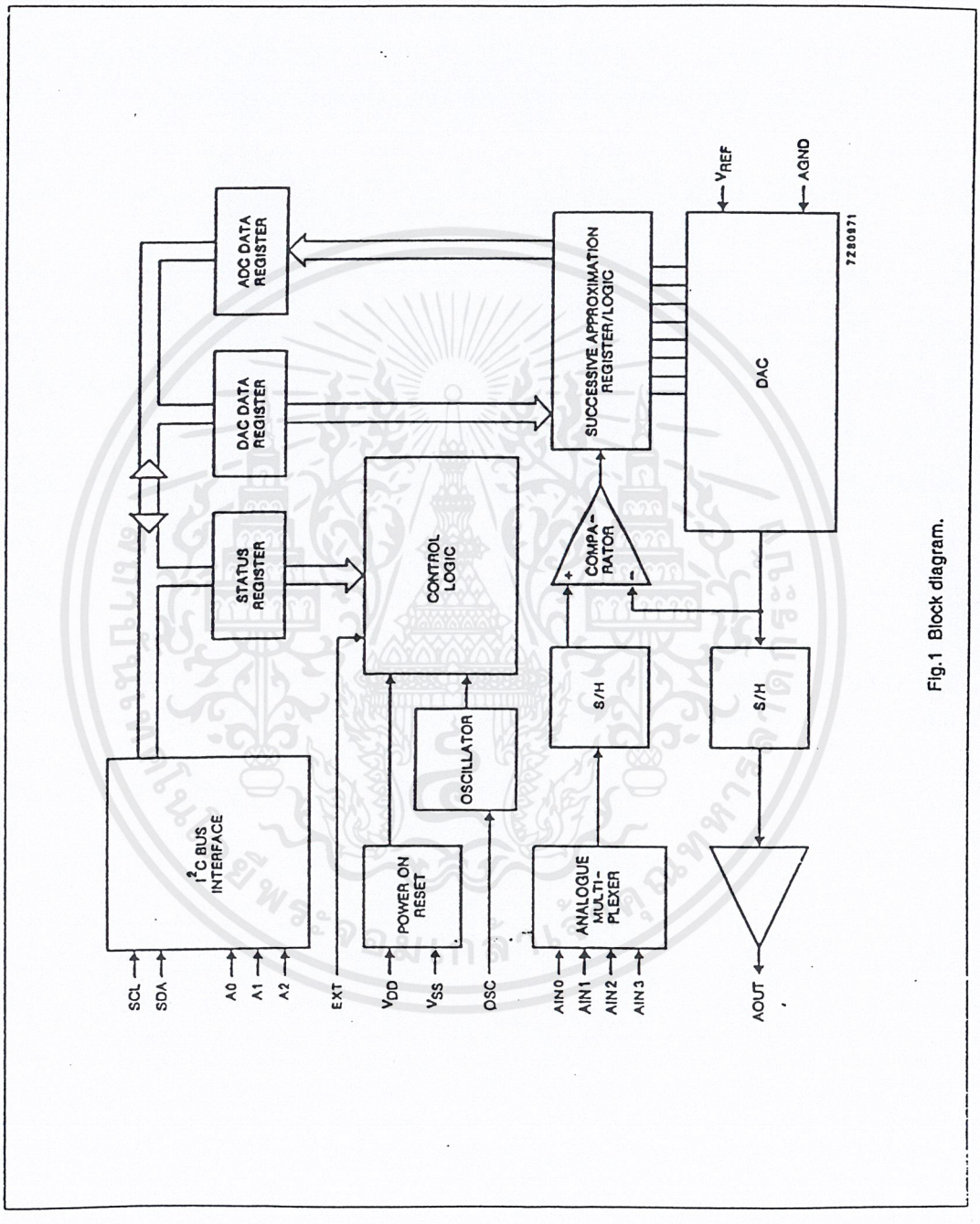


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
AIN0	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I^2C-bus data input/output
SCL	10	I^2C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

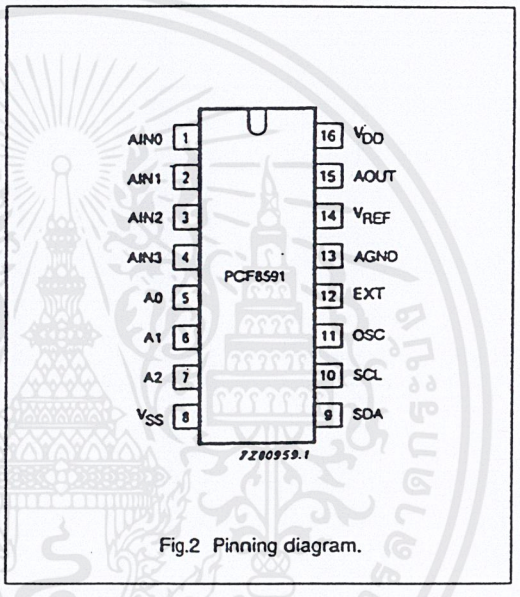


Fig.2 Pinning diagram.

7 FUNCTIONAL DESCRIPTION

7.1 Addressing

Each PCF8591 device in an I²C-bus system is activated by sending a valid address to the device. The address consists of a fixed part and a programmable part. The programmable part must be set according to the address pins A0, A1 and A2. The address always has to be sent as the first byte after the start condition in the I²C-bus protocol. The last bit of the address byte is the read/write-bit which sets the direction of the following data transfer (see Figs 3, 15 and 16).

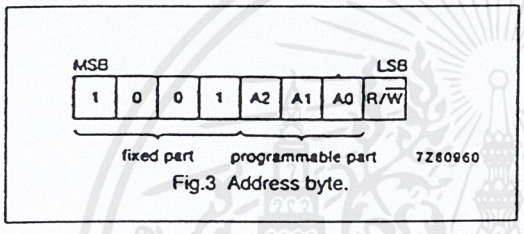


Fig.3 Address byte.

7.2 Control byte

The second byte sent to a PCF8591 device will be stored in its control register and is required to control the device function.

The upper nibble of the control register is used for enabling the analog output, and for programming the analog inputs as single-ended or differential inputs. The lower nibble selects one of the analog input channels defined by the upper nibble (see Fig.4). If the auto-increment flag is set the channel number is incremented automatically after each A/D conversion.

If the auto-increment mode is desired in applications where the internal oscillator is used, the analog output enable flag in the control byte (bit 6) should be set. This allows the internal oscillator to run continuously, thereby preventing conversion errors resulting from oscillator start-up delay. The analog output enable flag may be reset at other times to reduce quiescent power consumption.

The selection of a non-existing input channel results in the highest available channel number being allocated. Therefore, if the auto-increment flag is set, the next selected channel will be always channel 0. The most significant bits of both nibbles are reserved for future functions and have to be set to 0. After a Power-on reset condition all bits of the control register are reset to 0. The D/A converter and the oscillator are disabled for power saving. The analog output is switched to a high-impedance state.

7.3 D/A conversion

The third byte sent to a PCF8591 device is stored in the DAC data register and is converted to the corresponding analog voltage using the on-chip D/A converter. This D/A converter consists of a resistor divider chain connected to the external reference voltage with 256 taps and selection switches. The tap-decoder switches one of these taps to the DAC output line (see Fig.5).

The analog output voltage is buffered by an auto-zeroed unity gain amplifier. This buffer amplifier may be switched on or off by setting the analog output enable flag of the control register. In the active state the output voltage is held until a further data byte is sent.

The on-chip D/A converter is also used for successive approximation A/D conversion. In order to release the DAC for an A/D conversion cycle the unity gain amplifier is equipped with a track and hold circuit. This circuit holds the output voltage while executing the A/D conversion.

The output voltage supplied to the analog output AOUT is given by the formula shown in Fig.6. The waveforms of a D/A conversion sequence are shown in Fig.7.

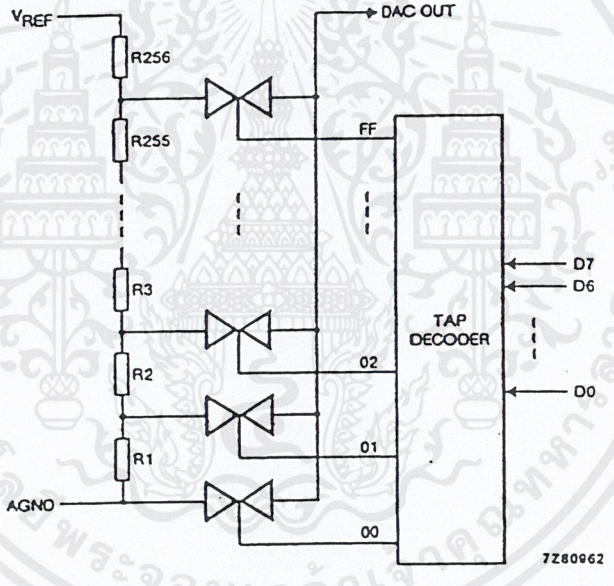


Fig.5 DAC resistor divider chain.

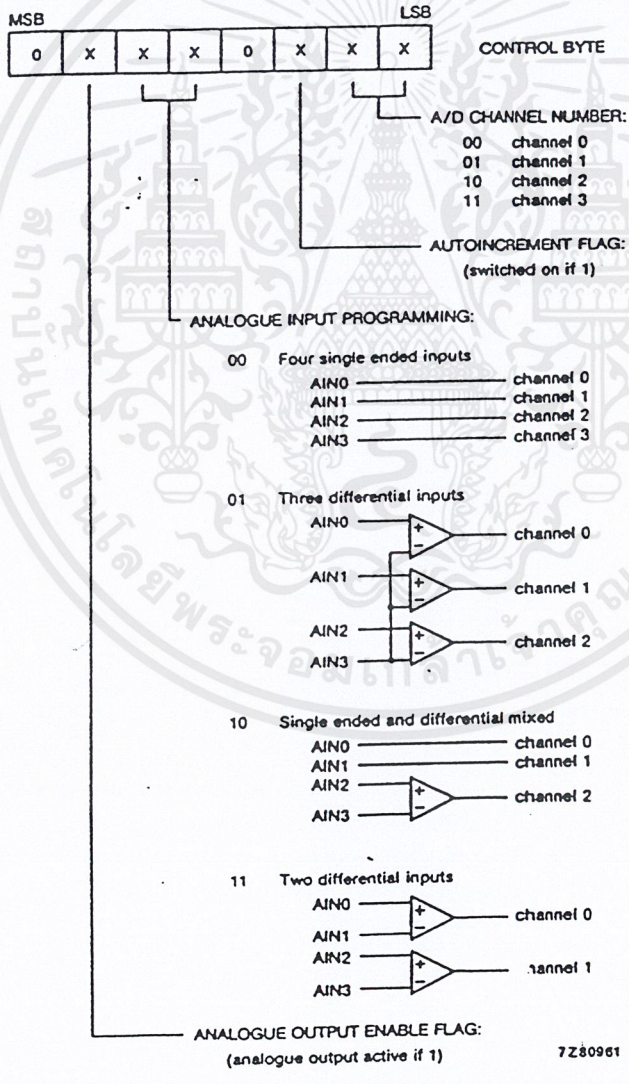


Fig.4 Control byte.

4 BLOCK DIAGRAM

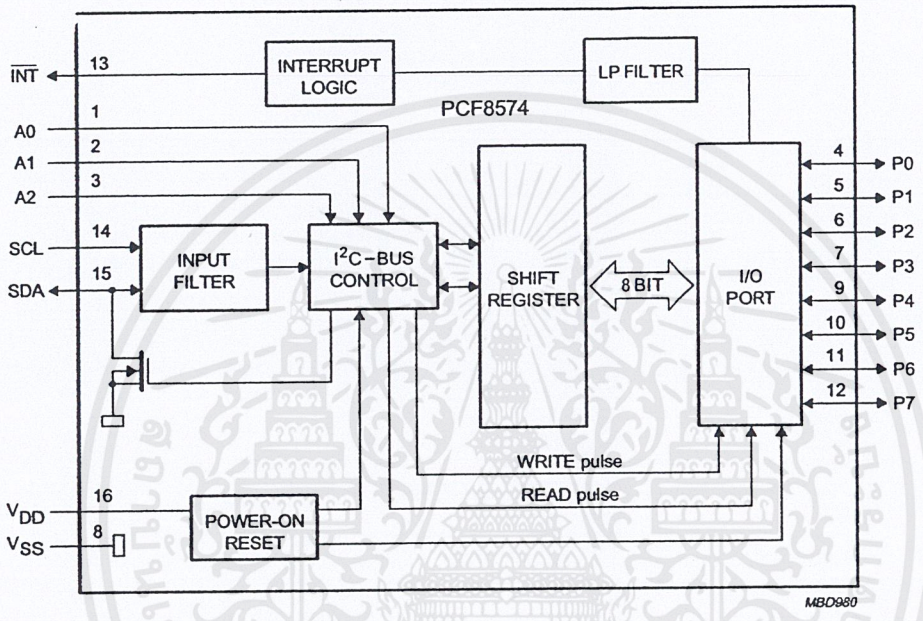


Fig.1 Block diagram (SOT38-1 and SOT162-1).

5 PINNING

SYMBOL	PIN		DESCRIPTION
	DIP16; SO16	SSOP20	
A0	1	6	address input 0
A1	2	7	address input 1
A2	3	9	address input 2
P0	4	10	quasi-bidirectional I/O 0
P1	5	11	quasi-bidirectional I/O 1
P2	6	12	quasi-bidirectional I/O 2
P3	7	14	quasi-bidirectional I/O 3
V_{SS}	8	15	supply ground
P4	9	16	quasi-bidirectional I/O 4
P5	10	17	quasi-bidirectional I/O 5
P6	11	19	quasi-bidirectional I/O 6
P7	12	20	quasi-bidirectional I/O 7
$\overline{INT}$	13	1	interrupt output (active LOW)
SCL	14	2	serial clock line
SDA	15	4	serial data line
V_{DD}	16	5	supply voltage
n.c.	–	3	not connected
n.c.	–	8	not connected
n.c.	–	13	not connected
n.c.	–	18	not connected

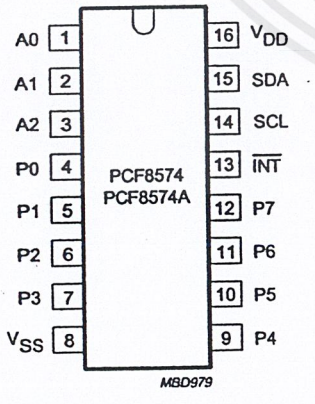


Fig.2 Pin configuration (DIP16; SO16).

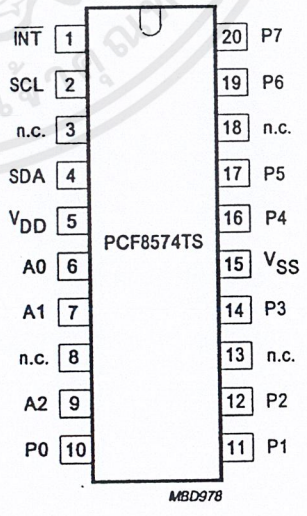


Fig.3 Pin configuration (SSOP20).

6 CHARACTERISTICS OF THE I²C-BUS

The I²C-bus is for 2-way, 2-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor when connected to the output stages of a device. Data transfer may be initiated only when the bus is not busy.

6.1 Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as control signals (see Fig.4).

6.2 Start and stop conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH is defined as the start condition (S).
A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the stop condition (P) (see Fig.5).

6.3 System configuration

A device generating a message is a 'transmitter', a device receiving is the 'receiver'. The device that controls the message is the 'master' and the devices which are controlled by the master are the 'slaves' (see Fig.6).

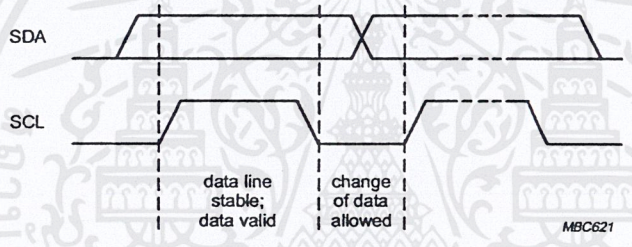


Fig.4 Bit transfer.

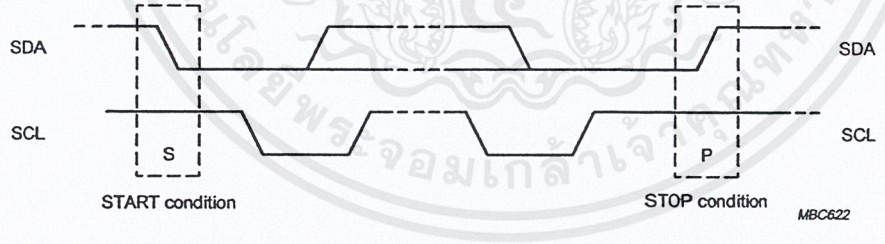


Fig.5 Definition of start and stop conditions.

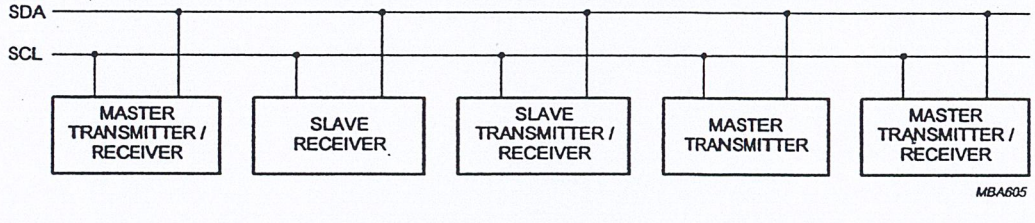


Fig.6 System configuration.

6.4 Acknowledge

The number of data bytes transferred between the start and the stop conditions from transmitter to receiver is not limited. Each byte of eight bits is followed by one acknowledge bit. The acknowledge bit is a HIGH level put on the bus by the transmitter whereas the master generates an extra acknowledge related clock pulse.

A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse, set-up and hold times must be taken into account.

A master receiver must signal an end of data to the transmitter by **not** generating an acknowledge on the last byte that has been clocked out of the slave. In this event the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

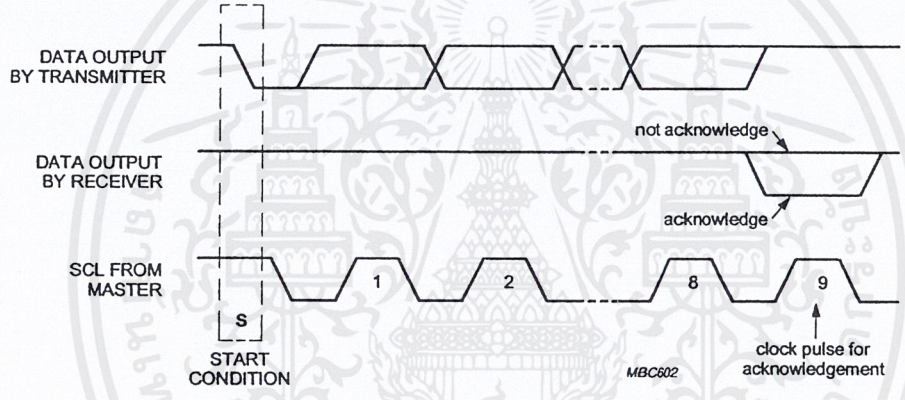


Fig.7 Acknowledgment on the I²C-bus.

7 FUNCTIONAL DESCRIPTION

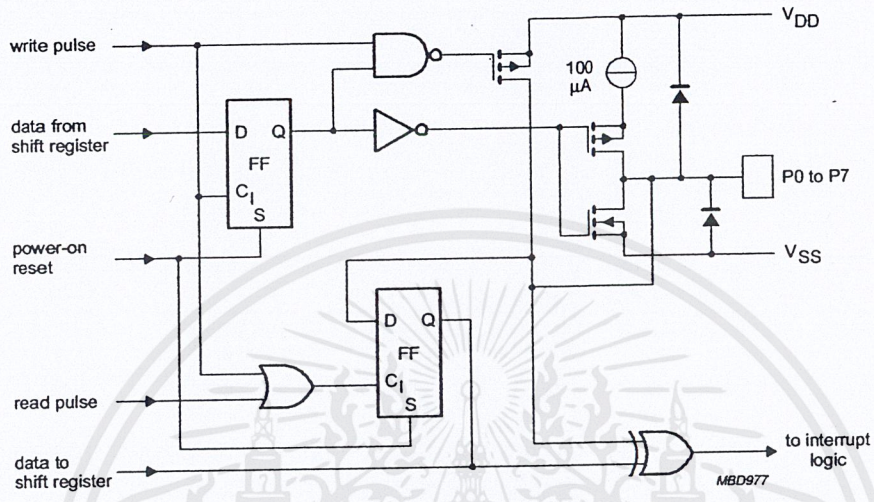
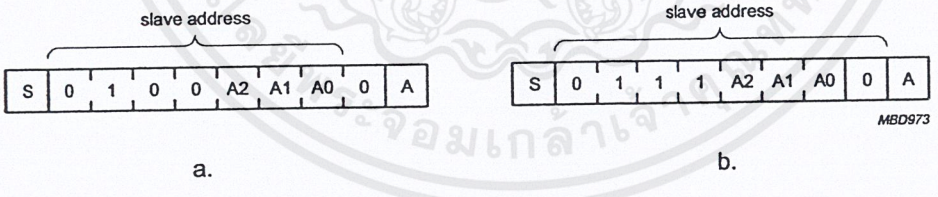


Fig.8 Simplified schematic diagram of each I/O.

7.1 Addressing

For addressing see Figs 9, 10 and 11.



(a) PCF8574.
(b) PCF8574A.

Fig.9 PCF8574 and PCF8574A slave addresses.

Each of the PCF8574's eight I/Os can be independently used as an input or output. Input data is transferred from the port to the microcontroller by the READ mode (see Fig.11). Output data is transmitted to the port by the WRITE mode (see Fig.10).

8 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage	−0.5	+7.0	V
V_I	input voltage	V_{SS} − 0.5	V_{DD} + 0.5	V
I_I	DC input current	−	±20	mA
I_O	DC output current	−	±25	mA
I_{DD}	supply current	−	±100	mA
I_{SS}	supply current	−	±100	mA
P_{tot}	total power dissipation	−	400	mW
P_O	power dissipation per output	−	100	mW
T_{stg}	storage temperature	−65	+150	°C
T_{amb}	operating ambient temperature	−40	+85	°C

9 HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC12 under *"Handling MOS Devices"*.

10 DC CHARACTERISTICS

V_{DD} = 2.5 to 6 V; V_{SS} = 0 V; T_{amb} = −40 to +85 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply						
V_{DD}	supply voltage		2.5	−	6.0	V
I_{DD}	supply current	operating mode; V_{DD} = 6 V; no load; $V_I = V_{DD}$ or V_{SS}; f_{SCL} = 100 kHz	−	40	100	µA
I_{stb}	standby current	standby mode; V_{DD} = 6 V; no load; $V_I = V_{DD}$ or V_{SS}	−	2.5	10	µA
V_{POR}	Power-on reset voltage	V_{DD} = 6 V; no load; $V_I = V_{DD}$ or V_{SS}; note 1	−	1.3	2.4	V
Input SCL; input/output SDA						
V_{IL}	LOW level input voltage		−0.5	−	+0.3V_{DD}	V
V_{IH}	HIGH level input voltage		0.7V_{DD}	−	V_{DD} + 0.5	V
I_{OL}	LOW level output current	V_{OL} = 0.4 V	3	−	−	mA
I_L	leakage current	$V_I = V_{DD}$ or V_{SS}	−1	−	+1	µA
C_i	input capacitance	$V_I = V_{SS}$	−	−	7	pF

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I/Os						
V_{IL}	LOW level input voltage		−0.5	−	$+0.3V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7V_{DD}$	−	$V_{DD} + 0.5$	V
$I_{IHL(max)}$	maximum allowed input current through protection diode	$V_I \geq V_{DD}$ or $V_I \leq V_{SS}$	−	−	±400	μA
I_{OL}	LOW level output current	V_{OL} = 1 V; V_{DD} = 5 V	10	25	−	mA
I_{OH}	HIGH level output current	$V_{OH} = V_{SS}$	30	−	300	μA
I_{OHt}	transient pull-up current	HIGH during acknowledge (see Fig.14); $V_{OH} = V_{SS}$; V_{DD} = 2.5 V	−	−1	−	mA
C_i	input capacitance		−	−	10	pF
C_o	output capacitance		−	−	10	pF
Port timing; $C_L \leq 100$ pF (see Figs 10 and 11)						
t_{pv}	output data valid		−	−	4	μs
t_{su}	input data set-up time		0	−	−	μs
t_h	input data hold time		4	−	−	μs
Interrupt $\overline{INT}$ (see Fig.13)						
I_{OL}	LOW level output current	V_{OL} = 0.4 V	1.6	−	−	mA
I_L	leakage current	$V_I = V_{DD}$ or V_{SS}	−1	−	+1	μA
TIMING; $C_L \leq 100$ pF						
t_{iv}	input data valid time		−	−	4	μs
t_{ir}	reset delay time		−	−	4	μs
Select inputs A0 to A2						
V_{IL}	LOW level input voltage		−0.5	−	$+0.3V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7V_{DD}$	−	$V_{DD} + 0.5$	V
I_{LI}	input leakage current	pin at V_{DD} or V_{SS}	−250	−	+250	nA

Note

1. The Power-on reset circuit resets the I²C-bus logic with $V_{DD} < V_{POR}$ and sets all I/Os to logic 1 (with current source to V_{DD}).

MAXIMUM RATINGS*

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	– 0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	– 1.5 to V_{CC} + 1.5	V
V_{out}	DC Output Voltage (Referenced to GND)	– 0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 25	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 50	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP†	750	mW
	SOIC Package†	500	
T_{stg}	Storage Temperature	– 65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		°C
	(Plastic DIP or SOIC Package)	260	
	(Ceramic DIP)	300	

This device contains protection circuitry to guard against damage due to high static voltages or electric fields. However, precautions must be taken to avoid applications of any voltage higher than maximum rated voltages to this high–impedance circuit. For proper operation, V_{in} and V_{out} should be constrained to the range GND ≤ (V_{in} or V_{out}) ≤ V_{CC}.

Unused inputs must always be tied to an appropriate logic voltage level (e.g., either GND or V_{CC}). Unused outputs must be left open.

* Maximum Ratings are those values beyond which damage to the device may occur. Functional operation should be restricted to the Recommended Operating Conditions.

† Derating — Plastic DIP: – 10 mW/°C from 65° to 125°C
Ceramic DIP: – 10 mW/°C from 100° to 125°C
SOIC Package: – 7 mW/°C from 65° to 125°C

For high frequency or heavy load considerations, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)		2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)		0	V_{CC}	V
T_A	Operating Temperature, All Package Types		– 55	+ 125	°C
t_r, t_f	Input Rise and Fall Time (Figure 1)	V_{CC} = 2.0 V	0	1000	ns
		V_{CC} = 4.5 V	0	500	
		V_{CC} = 6.0 V	0	400	

DC ELECTRICAL CHARACTERISTICS (Voltages Referenced to GND)

Symbol	Parameter	Test Conditions	V_{CC} V	Guaranteed Limit			Unit
				– 55 to 25°C	≤ 85°C	≤ 125°C	
V_{IH}	Minimum High–Level Input Voltage	V_{out} = 0.1 V or V_{CC} – 0.1 V $\|I_{out}\|$ ≤ 20 μA	2.0	1.5	1.5	1.5	V
			4.5	3.15	3.15	3.15	
			6.0	4.2	4.2	4.2	
V_{IL}	Maximum Low–Level Input Voltage	V_{out} = 0.1 V or V_{CC} – 0.1 V $\|I_{out}\|$ ≤ 20 μA	2.0	0.3	0.3	0.3	V
			4.5	0.9	0.9	0.9	
			6.0	1.2	1.2	1.2	
V_{OH}	Minimum High–Level Output Voltage	V_{in} = V_{IH} or V_{IL} $\|I_{out}\|$ ≤ 20 μA	2.0	1.9	1.9	1.9	V
			4.5	4.4	4.4	4.4	
			6.0	5.9	5.9	5.9	
		V_{in} = V_{IH} or V_{IL} $\|I_{out}\|$ ≤ 4.0 mA	4.5	3.98	3.84	3.70	
		$\|I_{out}\|$ ≤ 5.2 mA	6.0	5.48	5.34	5.20	
V_{OL}	Maximum Low–Level Output Voltage	V_{in} = V_{IH} or V_{IL} $\|I_{out}\|$ ≤ 20 μA	2.0	0.1	0.1	0.1	V
			4.5	0.1	0.1	0.1	
			6.0	0.1	0.1	0.1	
		V_{in} = V_{IH} or V_{IL} $\|I_{out}\|$ ≤ 4.0 mA	4.5	0.26	0.33	0.40	
		$\|I_{out}\|$ ≤ 5.2 mA	6.0	0.26	0.33	0.40	
I_{in}	Maximum Input Leakage Current	V_{in} = V_{CC} or GND	6.0	± 0.1	± 1.0	± 1.0	μA
I_{CC}	Maximum Quiescent Supply Current (per Package)	V_{in} = V_{CC} or GND I_{out} = 0 μA	6.0	8	80	160	μA

NOTE: Information on typical parametric values can be found in Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

AC ELECTRICAL CHARACTERISTICS (C_L = 50 pF, Input $t_r = t_f$ = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit			Unit
			– 55 to 25°C	≤ 85°C	≤ 125°C	
f_{max}	Maximum Clock Frequency (50% Duty Cycle) (Figures 1 and 4)	2.0 4.5 6.0	6.0 30 35	4.8 24 28	4.0 20 24	MHz
t_{PLH}, t_{PHL}	Maximum Propagation Delay, Clock to Q (Figures 1 and 4)	2.0 4.5 6.0	175 35 30	220 44 37	265 53 45	ns
t_{PHL}	Maximum Propagation Delay, Reset to Q (Figures 2 and 4)	2.0 4.5 6.0	205 41 35	255 51 43	310 62 53	ns
t_{TLH}, t_{THL}	Maximum Output Transition Time, Any Output (Figures 1 and 4)	2.0 4.5 6.0	75 15 13	95 19 16	110 22 19	ns
C_{in}	Maximum Input Capacitance	—	10	10	10	pF

NOTES:
1. For propagation delays with loads other than 50 pF, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).
2. Information on typical parametric values can be found in Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

		Typical @ 25°C, V_{CC} = 5.0 V	
C_{PD}	Power Dissipation Capacitance (Per Package)*	140	pF

* Used to determine the no–load dynamic power consumption: $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$. For load considerations, see Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

TIMING REQUIREMENTS (Input $t_r = t_f$ = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit			Unit
			– 55 to 25°C	≤ 85°C	≤ 125°C	
t_{su}	Minimum Setup Time, A1 or A2 to Clock (Figure 3)	2.0 4.5 6.0	50 10 9	65 13 11	75 15 13	ns
t_h	Minimum Hold Time, Clock to A1 or A2 (Figure 3)	2.0 4.5 6.0	5 5 5	5 5 5	5 5 5	ns
t_{rec}	Minimum Recovery Time, Reset Inactive to Clock (Figure 2)	2.0 4.5 6.0	5 5 5	5 5 5	5 5 5	ns
t_w	Minimum Pulse Width, Clock (Figure 1)	2.0 4.5 6.0	80 16 14	100 20 17	120 24 20	ns
t_w	Minimum Pulse Width, Reset (Figure 2)	2.0 4.5 6.0	80 16 14	100 20 17	120 24 20	ns
t_r, t_f	Maximum Input Rise and Fall Times (Figure 1)	2.0 4.5 6.0	1000 500 400	1000 500 400	1000 500 400	ns

NOTE: Information on typical parametric values can be found in Chapter 2 of the Motorola High–Speed CMOS Data Book (DL129/D).

PIN DESCRIPTIONS

INPUTS

A1, A2 (Pins 1, 2)

Serial Data Inputs. Data at these inputs determine the data to be entered into the first stage of the shift register. For a high level to be entered into the shift register, both A1 and A2 inputs must be high, thereby allowing one input to be used as a data–enable input. When only one serial input is used, the other must be connected to V_{CC}.

Clock (Pin 8)

Shift Register Clock. A positive–going transition on this pin shifts the data at each stage to the next stage. The shift register is completely static, allowing clock rates down to DC in a continuous or intermittent mode.

OUTPUTS

$Q_A - Q_H$ (Pins 3, 4, 5, 6, 10, 11, 12, 13)

Parallel Shift Register Outputs. The shifted data is presented at these outputs in true, or noninverted, form.

CONTROL INPUT

Reset (Pin 9)

Active–Low, Asynchronous Reset Input. A low voltage applied to this input resets all internal flip–flops and sets Outputs $Q_A - Q_H$ to the low level state.

SWITCHING WAVEFORMS

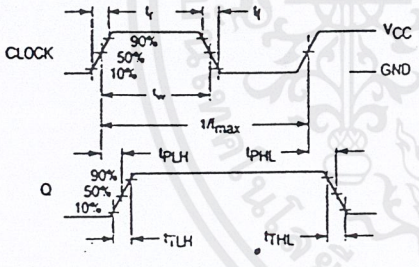


Figure 1.

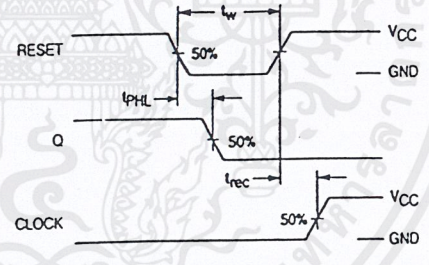


Figure 2.

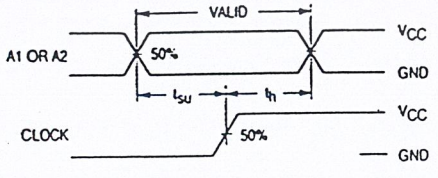
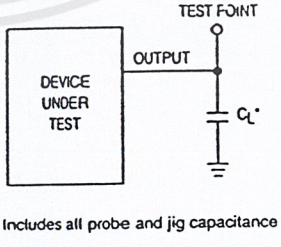


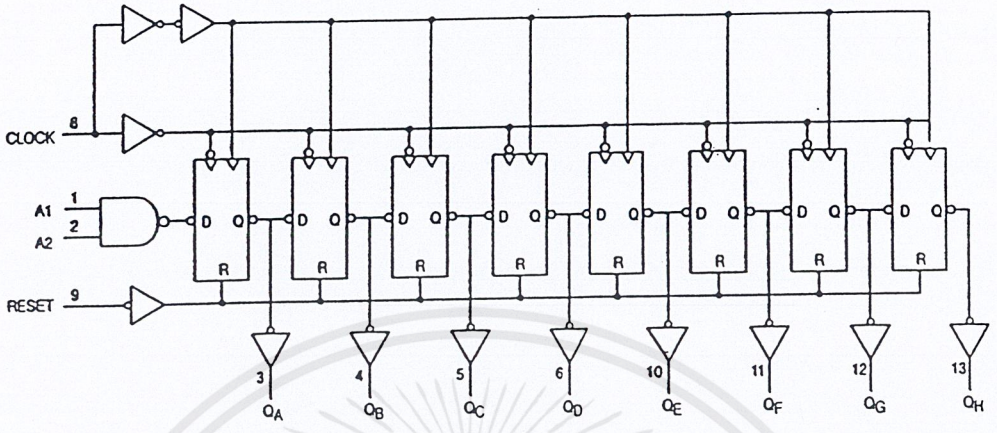
Figure 3.



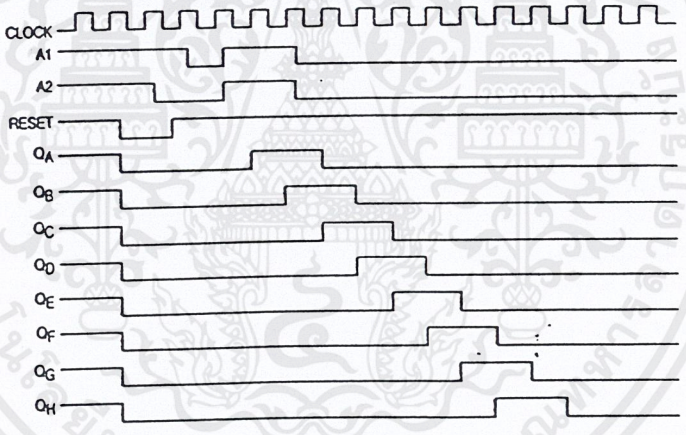
* Includes all probe and jig capacitance

Figure 4. Test Circuit

EXPANDED LOGIC DIAGRAM



TIMING DIAGRAM



FAIRCHILD
SEMICONDUCTOR TM

www.fairchildsemi.com

LM2902,LM324/LM324A

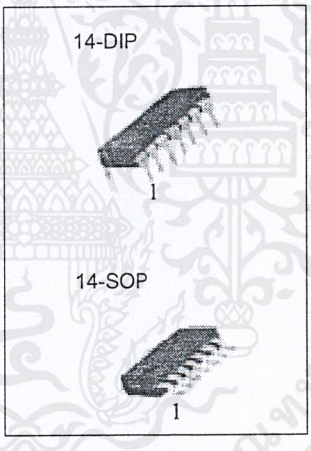
Quad Operational Amplifier

Features

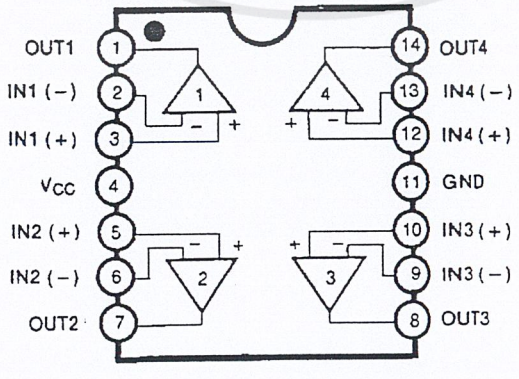
- Internally frequency compensated for unity gain
- Large DC voltage gain: 100dB
- Wide power supply range:
  LM324/LM324A : 3V~32V (or ±1.5 ~ 15V)
  LM2902: 3V~26V (or ±1.5V ~ 13V)
- Input common-mode voltage range includes ground
- Large output voltage swing: 0V to V_{CC} -1.5V
- Power drain suitable for battery operation

Description

The LM324/LM324A,LM2902 consist of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide voltage range. Operation from split power supplies is also possible so long as the difference between the two supplies is 3 volts to 32 volts. Application areas include transducer amplifier, DC gain blocks and all the conventional OP amp circuits which now can be easily implemented in single power supply systems.

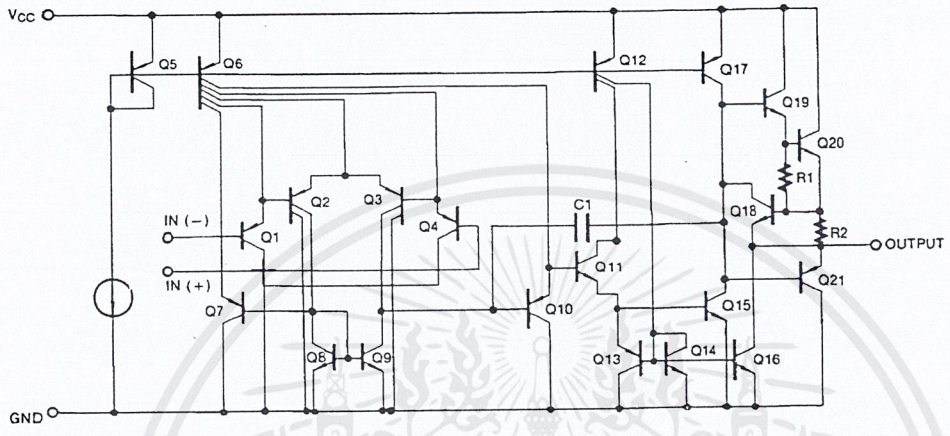


Internal Block Diagram



Schematic Diagram

(One Section Only)



Absolute Maximum Ratings

Parameter	Symbol	LM324/LM324A	LM2902	Unit
Power Supply Voltage	V_{CC}	±16 or 32	±13 or 26	V
Differential Input Voltage	$V_{I(DIFF)}$	32	26	V
Input Voltage	V_I	-0.3 to +32	-0.3 to +26	V
Output Short Circuit to GND	-	Continuous	Continuous	-
Power Dissipation	P_D	570	570	mW
Operating Temperature Range	T_{OPR}	0 ~ + 70	-40 ~ + 85	°C
Storage Temperature Range	T_{STG}	-65 ~ + 150	-65 ~ + 150	°C

Electrical Characteristics

(V_{CC}=5.0V, V_{EE}=GND, T_A=25 °C, unless otherwise specified)

Parameter	Symbol	Conditions	LM324			LM2902			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{CM} = 0V to V_{CC} = 1.5V $V_{O(P)}$ = 1.4V, R_S = 0Ω	-	1.5	7.0	-	1.5	7.0	mV
Input Offset Current	I_{IO}	-	-	3.0	50	-	3.0	50	nA
Input Bias Current	I_{BIAS}	-	-	40	250	-	40	250	nA
Common-Mode Input Voltage Range	$V_{I(R)}$	Note1	0	V_{CC}-1.5	-	0	-	V_{CC}-1.5	V
Supply Current	I_{CC}	R_L = ∞,V_{CC} = 30V (all Amps)	-	1.0	3	-	1.0	3	mA
		R_L = ∞,V_{CC} = 5V (all Amps) (V_{CC} = 26V for LM2902)	-	0.7	1.2	-	0.7	1.2	mA
Large Signal Voltage Gain	G_V	V_{CC} = 15V, R_L≥2KΩ $V_{O(P)}$ = 1V to 11V	25	100	-	-	100	-	V/mV
Output Voltage Swing	$V_{O(H)}$	Note1 R_L = 2KΩ	26	-	-	22	-	-	V
		R_L = 10KΩ	27	28	-	23	24	-	V
	$V_{O(L)}$	V_{CC} = 5V, R_L≥10KΩ	-	5	20	-	5	100	mV
Common-Mode Rejection Ratio	CMRR	-	65	75	-	50	75	-	dB
Power Supply Rejection Ratio	PSRR	-	65	100	-	50	100	-	dB
Channel Separation	CS	f = 1KHz to 20KHz	-	120	-	-	120	-	dB
Short Circuit to GND	I_{SC}	-	-	40	60	-	40	60	mA
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V V_{CC} = 15V, $V_{O(P)}$ = 2V	20	40	-	20	40	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V, $V_{O(P)}$ = 2V	10	13	-	10	13	-	mA
		$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V,$V_{O(R)}$ = 200mV	12	45	-	-	-	-	μA
Differential Input Voltage	$V_{I(DIFF)}$		-	-	V_{CC}	-	-	V_{CC}	V

Note.

1. V_{CC}=30V for LM324 , V_{CC} = 26V for LM2902

Electrical Characteristics

(V_{CC} = 5.0V, V_{EE} = GND, unless otherwise specified)
The following specification apply over the range of 0 °C ≤ T_A ≤ +70 °C for the LM324 ; and the - 40°C ≤ T_A ≤ +85 °C for the LM2902

Parameter	Symbol	Conditions	LM324			LM2902			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{ICM} = 0V to V_{CC} = 1.5V $V_{O(P)}$ = 1.4V, R_S = 0Ω	-	-	9.0	-	-	10.0	mV
Input Offset Voltage Drift	$\Delta V_{IO}/\Delta T$	-	-	7.0	-	-	7.0	-	μV/ °C
Input Offset Current	I_{IO}	-	-	-	150	-	-	200	nA
Input Offset Current Drift	$\Delta I_{IO}/\Delta T$	-	-	10	-	-	10	-	pA/ °C
Input Bias Current	I_{BIAS}	-	-	-	500	-	-	500	nA
Common-Mode Input Voltage Range	$V_{I(R)}$	Note1	0	-	V_{CC}-2.0	0	-	V_{CC}-2.0	V
Large Signal Voltage Gain	G_V	V_{CC} = 15V, R_L ≥ 2.0KΩ $V_{O(P)}$ = 1V to 11V	15	-	-	15	-	-	V/mV
Output Voltage Swing	$V_{O(H)}$	Note1, R_L = 2KΩ	26	-	-	22	-	-	V
		Note1, R_L = 10KΩ	27	28	-	23	24	-	V
	$V_{O(L)}$	V_{CC} = 5V, R_L≥10KΩ	-	5	20	-	5	100	mV
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V V_{CC} = 15V, $V_{O(P)}$ = 2V	10	20	-	10	20	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V, $V_{O(P)}$ = 2V	5	8	-	5	8	-	mA
Differential Input Voltage	$V_{I(DIFF)}$	-	-	-	V_{CC}	-	-	V_{CC}	V

Note.
1. V_{CC}=30V for LM324 , V_{CC} = 26V for LM2902

Electrical Characteristics

(V_{CC}=5.0V, V_{EE} = GND, T_A=25 °C, unless otherwise specified)

Parameter	Symbol	Conditions	LM324A			Unit
			Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{CM} = 0V to V_{CC} = 1.5V $V_{O(P)}$ = 1.4V, R_S = 0 Ω	-	1.5	3.0	mV
Input Offset Current	I_{IO}	-	-	3.0	30	nA
Input Bias Current	I_{BIAS}	-	-	40	100	nA
Input Common-Mode Voltage Range	$V_{I(R)}$	V_{CC} = 30V	0	-	V_{CC}-1.5	V
Supply Current (All Amps)	I_{CC}	V_{CC} = 30V	-	1.5	3	mA
		V_{CC} = 5V	-	0.7	1.2	mA
Large Signal Voltage Gain	G_V	V_{CC} = 15V, R_L≥ 2 KΩ $V_{O(P)}$ = 1V to 11V	25	100	-	V/mV
Output Voltage Swing	$V_{O(H)}$	Note1, R_L = 2 KΩ	26	-	-	V
		Note1, R_L = 10 KΩ	27	28	-	V
	$V_{O(L)}$	V_{CC} = 5V, R_L≥ 10 KΩ	-	5	20	mV
Common-Mode Rejection Ratio	CMRR	-	65	85	-	dB
Power Supply Rejection Ratio	PSRR	-	65	100	-	dB
Channel Separation	CS	f = 1KHz to 20KHz	-	120	-	dB
Short Circuit to GND	I_{SC}	-	-	40	60	mA
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V V_{CC} = 15V	20	40	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V, $V_{O(P)}$ = 2V	10	20	-	mA
		$V_{I(+)}$ = 0v, $V_{I(-)}$ = 1V V_{CC} = 15V, $V_{O(P)}$ = 200mV	12	50	-	µA
Differential Input Voltage	$V_{I(DIFF)}$	-	-	-	V_{CC}	V

Electrical Characteristics

(V_{CC} = 5.0V, V_{EE} = GND, unless otherwise specified)
The following specification apply over the range of 0 °C ≤ T_A ≤+70 °C for the LM324A

Parameter	Symbol	Conditions	LM324A			Unit
			Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{CM} = 0V to V_{CC} = 1.5V $V_{O(P)}$ = 1.4V, R_S = 0Ω	-	-	5.0	mV
Input Offset Voltage Drift	$\Delta V_{IO}/\Delta T$	-	-	7.0	30	μV/ °C
Input Offset Current	I_{IO}	-	-	-	75	nA
Input Offset Current Drift	$\Delta I_{IO}/\Delta T$	-	-	10	300	pA/ °C
Input Bias Current	I_{BIAS}	-	-	40	200	nA
Common-Mode Input Voltage Range	$V_{I(R)}$	V_{CC} = 30V	0	-	V_{CC}-2.0	V
Large Signal Voltage Gain	G_V	V_{CC} = 15V, R_L≥ 2.0KΩ	15	-	-	V/mV
Output Voltage Swing	$V_{O(P-P)}$	V_{CC} = 30V, R_L = 2KΩ	26	-	-	V
		V_{CC} = 30V, R_L = 10KΩ	27	28	-	
		V_{CC} = 5V, R_L≥ 10KΩ	-	5	20	mA
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V V_{CC} = 15V	10	20	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V	5	8	-	mA
Differential Input Voltage	$V_{I(DIFF)}$	-	-	-	V_{CC}	V

Typical Performance Characteristics

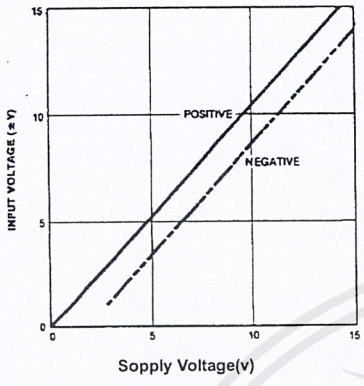


Figure 1. Input Voltage Range vs Supply Voltage

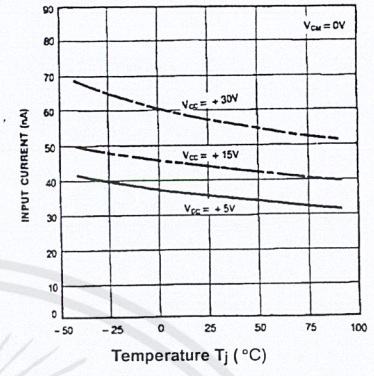


Figure 2. Input Current vs Temperature

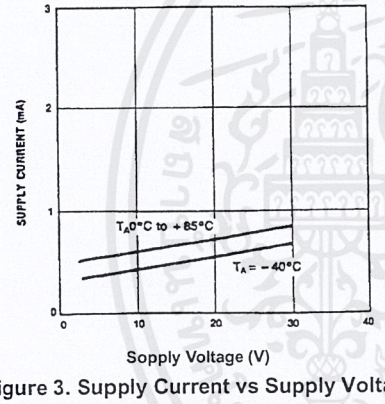


Figure 3. Supply Current vs Supply Voltage

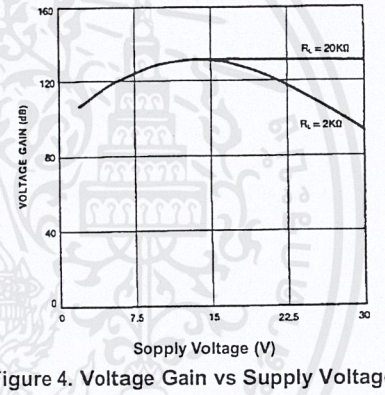


Figure 4. Voltage Gain vs Supply Voltage

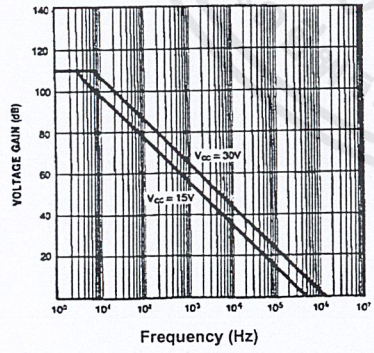


Figure 5. Open Loop Frequency Response

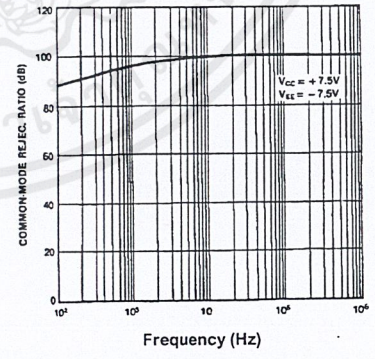


Figure 6. Common mode Rejection Ratio

Typical Performance Characteristics (continued)

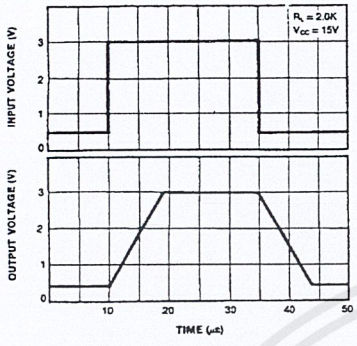


Figure 7. Slew Rate

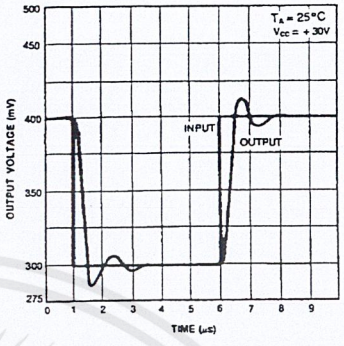


Figure 8. Voltage Follower Pulse Response

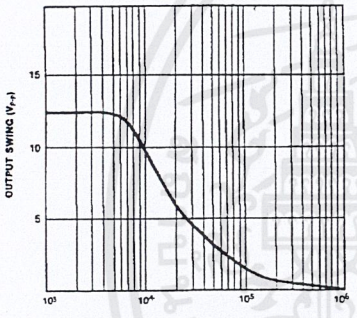


Figure 9. Large Signal Frequency Response

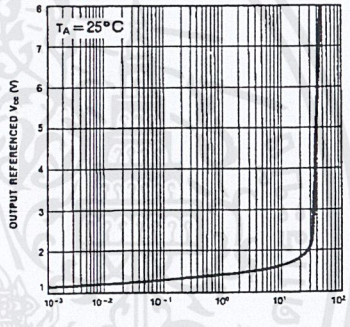


Figure 10. Output Characteristics vs Current Sourcing

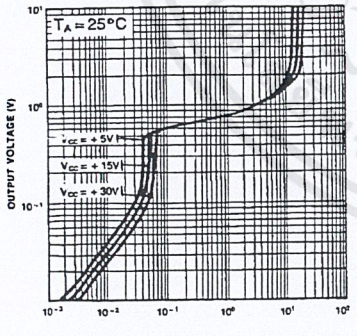


Figure 11. Output Characteristics vs Current Sinking

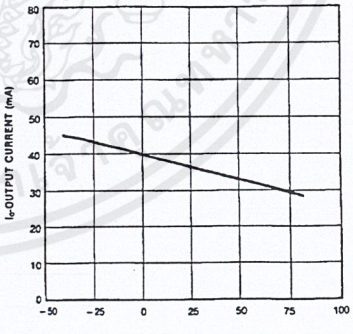
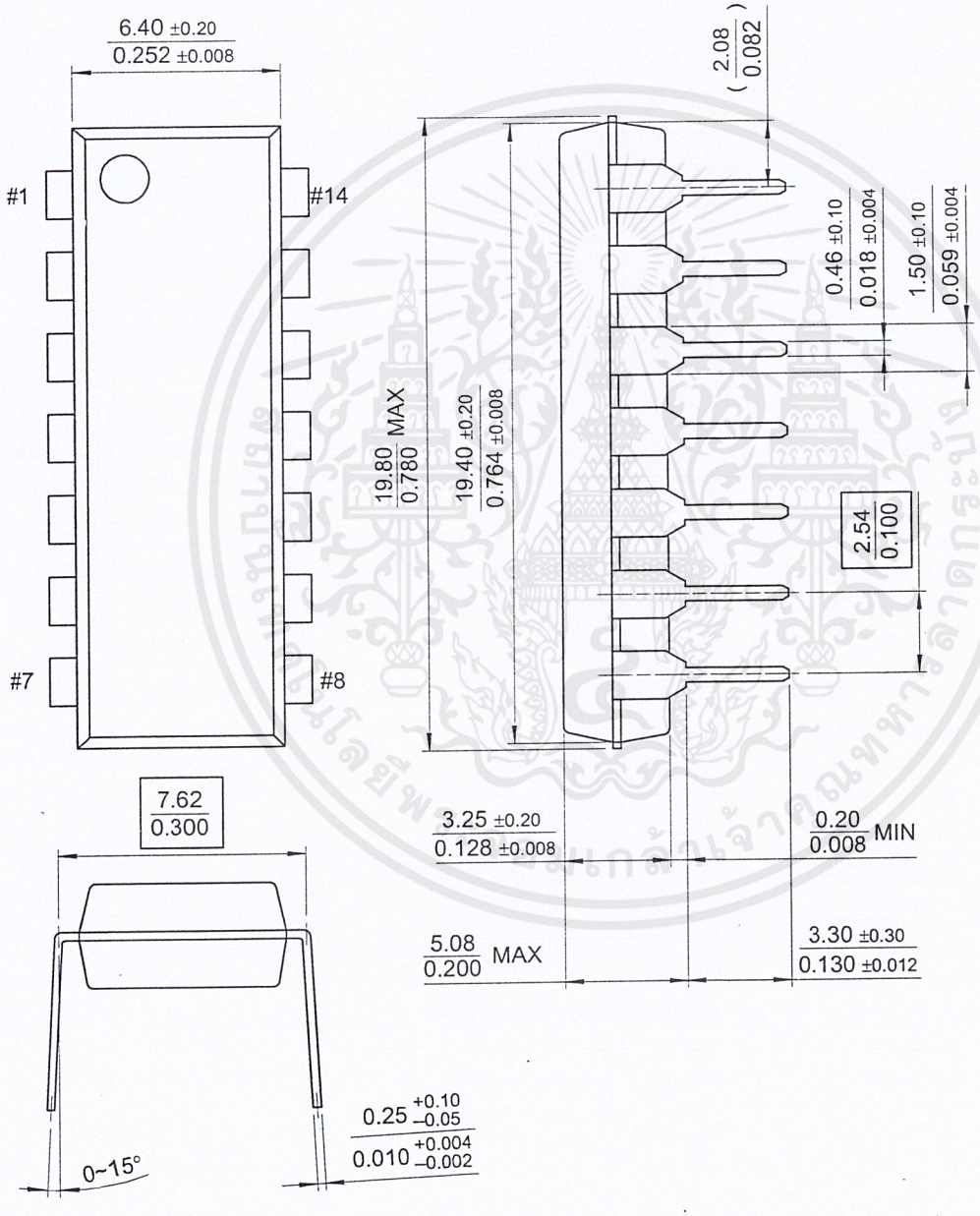


Figure 12. Current Limiting vs Temperature

Mechanical Dimensions

Package

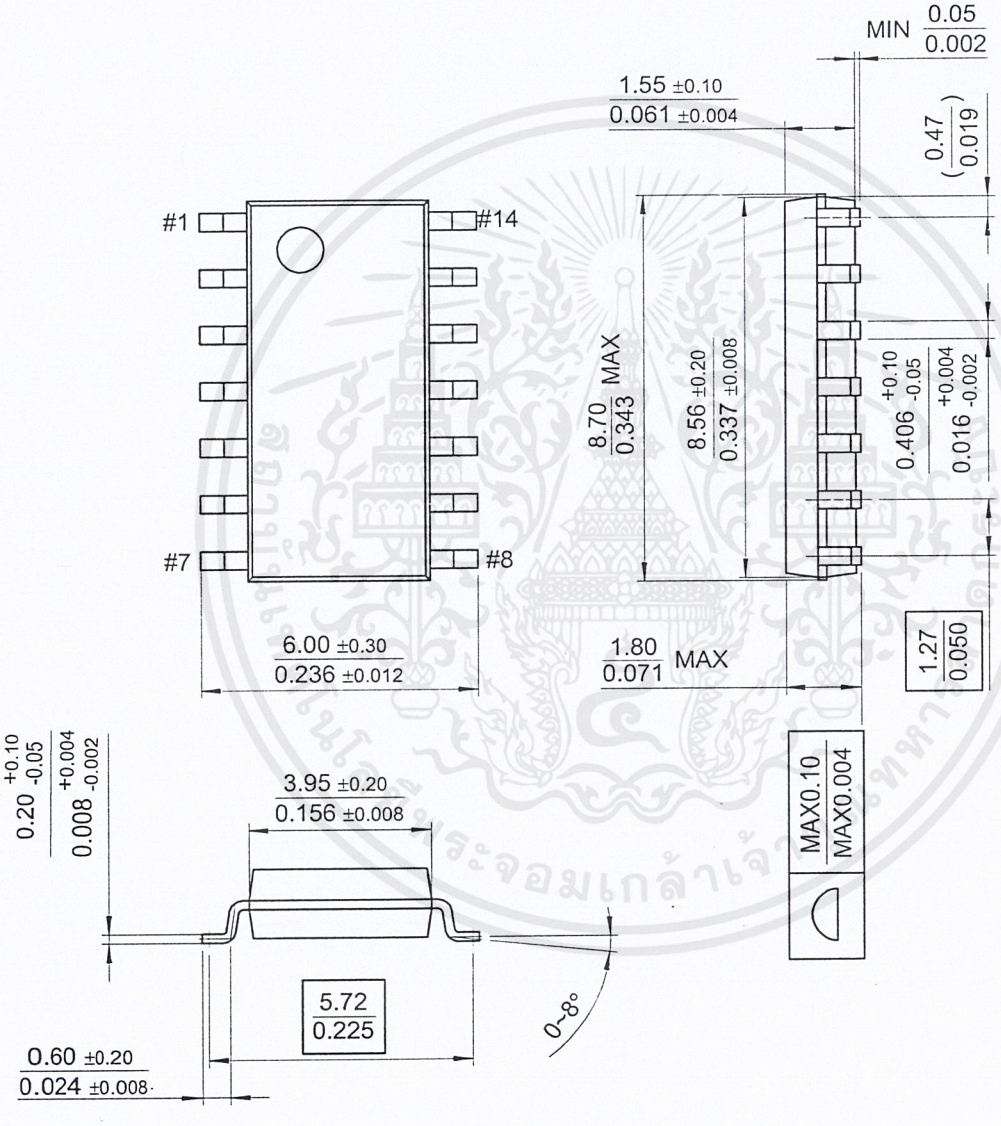
14-DIP



Mechanical Dimensions (Continued)

Package

14-SOP



Ordering Information

Product Number	Package	Operating Temperature
LM324N	14-SOP	0 ~ + 70 °C
LM324AN		
LM324M	14-SOP	
LM324AM		
LM2902N	14-DIP	-40 ~ + 85 °C
LM2902M	14-SOP	

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR INTERNATIONAL. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.

2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

7/13/00 0.0m 001
Stock#DSxxxxxxxx

June 1999

National Semiconductor

DAC0800/DAC0802
8-Bit Digital-to-Analog Converters

General Description

The DAC0800 series are monolithic 8-bit high-speed current-output digital-to-analog converters (DAC) featuring typical settling times of 100 ns. When used as a multiplying DAC, monotonic performance over a 40 to 1 reference current range is possible. The DAC0800 series also features high compliance complementary current outputs to allow differential output voltages of 20 Vp-p with simple resistor loads as shown in *Figure 1.* The reference-to-full-scale current matching of better than ±1 LSB eliminates the need for full-scale trims in most applications while the nonlinearities of better than ±0.1% over temperature minimizes system error accumulations.

The noise immune inputs of the DAC0800 series will accept TTL levels with the logic threshold pin, V_{LC}, grounded. Changing the V_{LC} potential will allow direct interface to other logic families. The performance and characteristics of the device are essentially unchanged over the full ±4.5V to ±18V power supply range; power dissipation is only 33 mW with ±5V supplies and is independent of the logic input states.

The DAC0800, DAC0802, DAC0800C and DAC0802C are a direct replacement for the DAC-08, DAC-08A, DAC-08C, and DAC-08H, respectively.

Features

- Fast settling output current: 100 ns
- Full scale error: ±1 LSB
- Nonlinearity over temperature: ±0.1%
- Full scale current drift: ±10 ppm/°C
- High output compliance: −10V to +18V
- Complementary current outputs
- Interface directly with TTL, CMOS, PMOS and others
- 2 quadrant wide range multiplying capability
- Wide power supply range: ±4.5V to ±18V
- Low power consumption: 33 mW at ±5V
- Low cost

Typical Applications

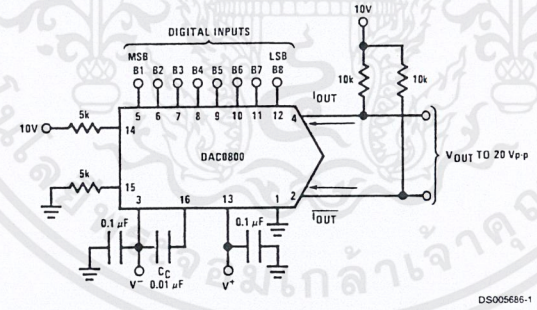


FIGURE 1. ±20 V_{P-P} Output Digital-to-Analog Converter (Note 5)

Ordering Information

Non-Linearity	Temperature Range	Order Numbers				
		J Package (J16A) (Note 1)		N Package (N16E) (Note 1)		SO Package (M16A)
±0.1% FS	0°C ≤ T_A ≤ +70°C	DAC0802LCJ	DAC-08HQ	DAC0802LCN	DAC-08HP	DAC0802LCM
±0.19% FS	−55°C ≤ T_A ≤ +125°C	DAC0800LJ	DAC-08Q			
±0.19% FS	0°C ≤ T_A ≤ +70°C	DAC0800LCJ	DAC-08EQ	DAC0800LCN	DAC-08EP	DAC0800LCM

Note 1: Devices may be ordered by using either order number.

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage ($V^+ - V^-$)	±18V or 36V
Power Dissipation (Note 3)	500 mW
Reference Input Differential Voltage (V14 to V15)	V^- to V^+
Reference Input Common-Mode Range (V14, V15)	V^- to V^+
Reference Input Current	5 mA
Logic Inputs	V^- to V^- plus 36V
Analog Current Outputs (V_S- = −15V)	4.25 mA
ESD Susceptibility (Note 4)	TBD V
Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C

Operating Conditions (Note 2)

	Min	Max	Units
Temperature (T_A)			
DAC0800L	−55	+125	°C
DAC0800LC	0	+70	°C
DAC0802LC	0	+70	°C

Electrical Characteristics

The following specifications apply for V_S = ±15V, I_{REF} = 2 mA and $T_{MIN} \le T_A \le T_{MAX}$ unless otherwise specified. Output characteristics refer to both I_{OUT} and $\overline{I_{OUT}}$.

Symbol	Parameter	Conditions	DAC0802LC			DAC0800L/ DAC0800LC			Units
			Min	Typ	Max	Min	Typ	Max	
	Resolution		8	8	8	8	8	8	Bits
	Monotonicity		8	8	8	8	8	8	Bits
	Nonlinearity				±0.1			±0.19	%FS
t_s	Settling Time	To ±½ LSB, All Bits Switched "ON" or "OFF", T_A=25°C		100	135				ns
		DAC0800L					100	135	ns
		DAC0800LC					100	150	ns
tPLH, tPHL	Propagation Delay Each Bit	T_A=25°C		35	60		35	60	ns
	All Bits Switched			35	60		35	60	ns
TCI_{FS}	Full Scale Tempco			±10	±50		±10	±50	ppm/°C
V_{OC}	Output Voltage Compliance	Full Scale Current Change <½ LSB, R_{OUT}>20 MΩ Typ	−10		18	−10		18	V
I_{FS4}	Full Scale Current	V_{REF}=10.000V, R14=5.000 kΩ R15=5.000 kΩ, T_A=25°C	1.984	1.992	2.000	1.94	1.99	2.04	mA
I_{FSS}	Full Scale Symmetry	$I_{FS4}-I_{FS2}$		±0.5	±4.0		±1	±8.0	µA
I_{ZS}	Zero Scale Current			0.1	1.0		0.2	2.0	µA
I_{FSR}	Output Current Range	V^-=−5V	0	2.0	2.1	0	2.0	2.1	mA
		V^-=−8V to −18V	0	2.0	4.2	0	2.0	4.2	mA
	Logic Input Levels								
V_{IL}	Logic "0"	V_{LC}=0V			0.8			0.8	V
V_{IH}	Logic "1"		2.0			2.0			V
	Logic Input Current	V_{LC}=0V							
I_{IL}	Logic "0"	−10V≤V_{IN}≤+0.8V		−2.0	−10		−2.0	−10	µA
I_{IH}	Logic "1"	2V≤V_{IN}≤+18V		0.002	10		0.002	10	µA
V_{IS}	Logic Input Swing	V^-=−15V	−10		18	−10		18	V
V_{THR}	Logic Threshold Range	V_S=±15V	−10		13.5	−10		13.5	V
I_{15}	Reference Bias Current			−1.0	−3.0		−1.0	−3.0	µA
dI/dt	Reference Input Slew Rate	(*Figure 11*)	4.0	8.0		4.0	8.0		mA/µs
$PSSI_{FS+}$	Power Supply Sensitivity	4.5V≤V^+≤18V		0.0001	0.01		0.0001	0.01	%/%
$PSSI_{FS-}$		−4.5V≤V^-≤18V I_{REF}=1mA		0.0001	0.01		0.0001	0.01	%/%

Electrical Characteristics (Continued)

The following specifications apply for V_S = ±15V, I_{REF} = 2 mA and $T_{MIN} \le T_A \le T_{MAX}$ unless otherwise specified. Output characteristics refer to both I_{OUT} and $\overline{I_{OUT}}$.

Symbol	Parameter	Conditions	DAC0802LC			DAC0800L/ DAC0800LC			Units
			Min	Typ	Max	Min	Typ	Max	
	Power Supply Current	V_S=±5V, I_{REF}=1 mA							
I+				2.3	3.8		2.3	3.8	mA
I−				−4.3	−5.8		−4.3	−5.8	mA
		V_S=5V, −15V, I_{REF}=2 mA							
I+				2.4	3.8		2.4	3.8	mA
I−				−6.4	−7.8		−6.4	−7.8	mA
		V_S=±15V, I_{REF}=2 mA							
I+				2.5	3.8		2.5	3.8	mA
I−				−6.5	−7.8		−6.5	−7.8	mA
P_D	Power Dissipation	±5V, I_{REF}=1 mA		33	48		33	48	mW
		5V,−15V, I_{REF}=2 mA		108	136		108	136	mW
		±15V, I_{REF}=2 mA		135	174		135	174	mW

Note 2: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

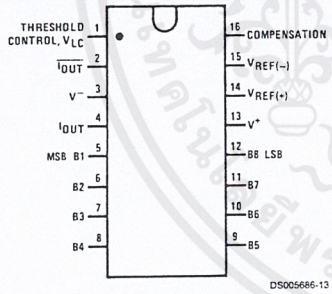
Note 3: The maximum junction temperature of the DAC0800 and DAC0802 is 125°C. For operating at elevated temperatures, devices in the Dual-In-Line J package must be derated based on a thermal resistance of 100°C/W, junction-to-ambient, 175°C/W for the molded Dual-In-Line N package and 100°C/W for the Small Outline M package.

Note 4: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 5: Pin-out numbers for the DAC080X represent the Dual-In-Line package. The Small Outline package pin-out differs from the Dual-In-Line package.

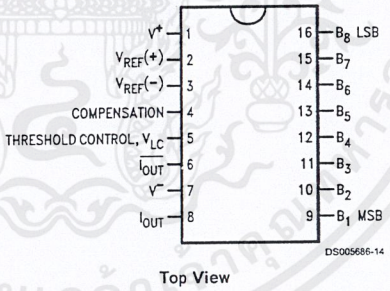
Connection Diagrams

Dual-In-Line Package



Top View

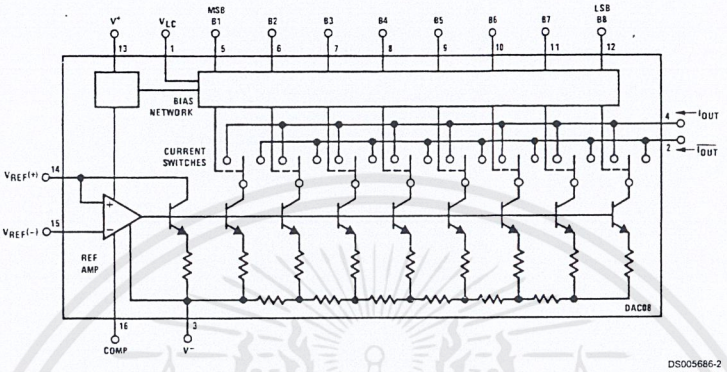
Small Outline Package



Top View

See Ordering Information

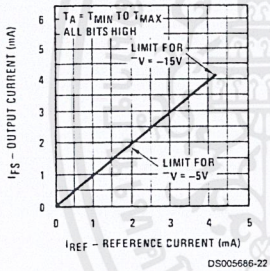
Block Diagram (Note 5)



DS005686-2

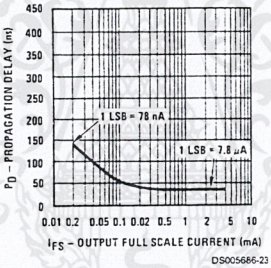
Typical Performance Characteristics

Full Scale Current vs Reference Current



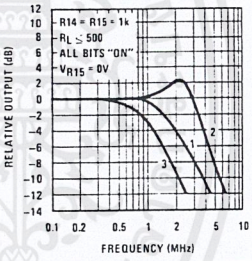
DS005686-22

LSB Propagation Delay vs I_{FS}



DS005686-23

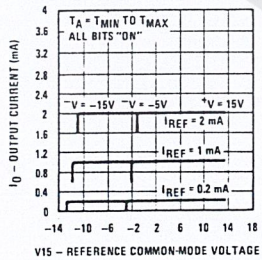
Reference Input Frequency Response



DS005686-24

Curve 1: C_C=15 pF, V_{IN}=2 Vp-p centered at 1V.
Curve 2: C_C=15 pF, V_{IN}=50 mVp-p centered at 200 mV.
Curve 3: C_C=0 pF, V_{IN}=100 mVp-p centered at 0V and applied through 50Ω connected to pin 14.2V applied to R14.

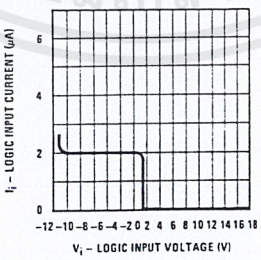
Reference Amp Common-Mode Range



DS005686-25

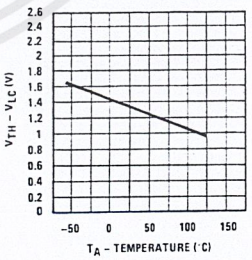
Note. Positive common-mode range is always (V+) − 1.5V.

Logic Input Current vs Input Voltage



DS005686-26

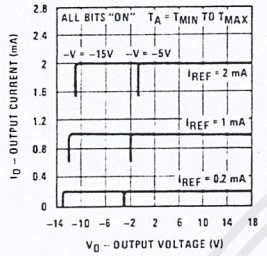
V_{TH} — V_{LC} vs Temperature



DS005686-27

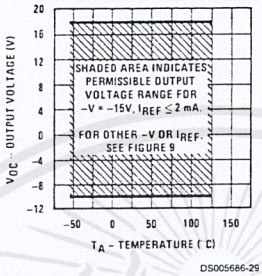
Typical Performance Characteristics (Continued)

Output Current vs Output Voltage (Output Voltage Compliance)



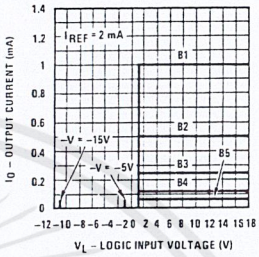
DS005686-28

Output Voltage Compliance vs Temperature



DS005686-29

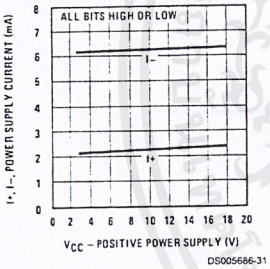
Bit Transfer Characteristics



DS005686-30

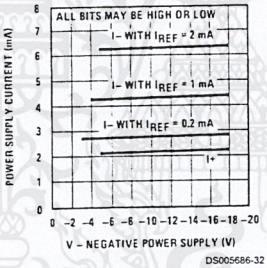
Note. B1–B8 have identical transfer characteristics. Bits are fully switched with less than ½ LSB error, at less than ±100 mV from actual threshold. These switching points are guaranteed to lie between 0.8 and 2V over the operating temperature range (V_{LC} = 0V).

Power Supply Current vs +V



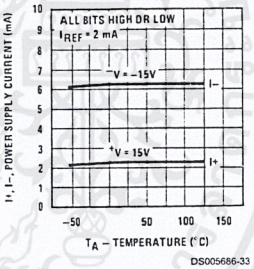
DS005686-31

Power Supply Current vs −V



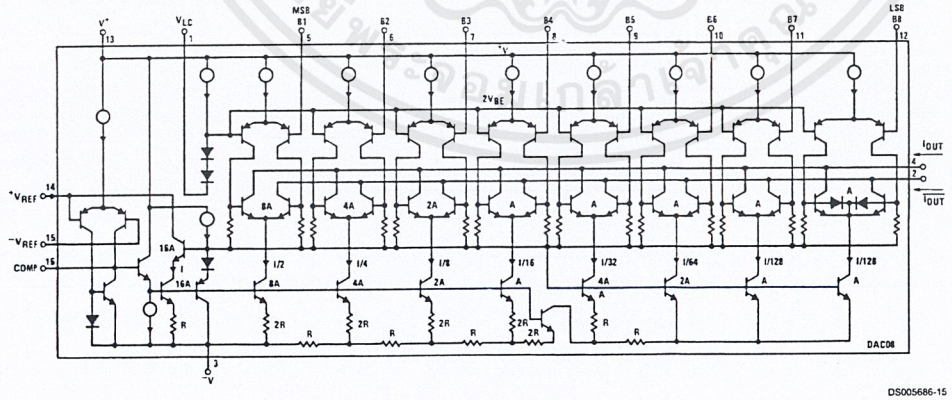
DS005686-32

Power Supply Current vs Temperature



DS005686-33

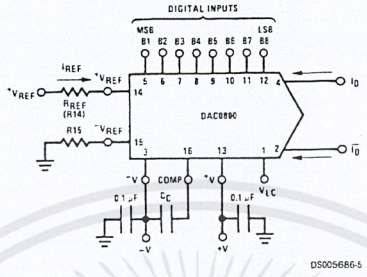
Equivalent Circuit



DS005686-15

FIGURE 2.

Typical Applications

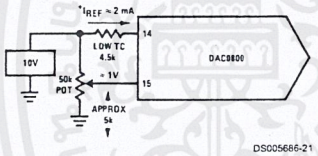


DS005686-5

$$I_{FS} \approx \frac{+V_{REF}}{R_{REF}} \times \frac{255}{256}$$

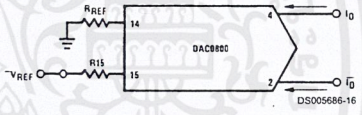
$I_O + \bar{I}_O = I_{FS}$ for all logic states
For fixed reference, TTL operation, typical values are:
V_{REF} = 10.000V
R_{REF} = 5.000k
R15 = R_{REF}
C_C = 0.01 µF
V_{LC} = 0V (Ground)

FIGURE 3. Basic Positive Reference Operation (Note 5)



DS005686-21

FIGURE 4. Recommended Full Scale Adjustment Circuit (Note 5)

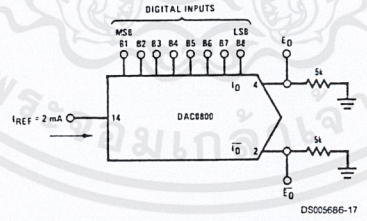


DS005686-16

$$I_{FS} \approx \frac{-V_{REF}}{R_{REF}} \times \frac{255}{256}$$

Note. R_{REF} sets I_{FS}; R15 is for bias current cancellation

FIGURE 5. Basic Negative Reference Operation (Note 5)

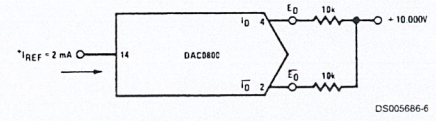


DS005686-17

	B1	B2	B3	B4	B5	B6	B7	B8	I_O mA	$\bar{I}_O$mA	E_O	$\bar{E}_O$
Full Scale	1	1	1	1	1	1	1	1	1.992	0.000	−9.960	0.000
Full Scale−LSB	1	1	1	1	1	1	1	0	1.984	0.008	−9.920	−0.040
Half Scale+LSB	1	0	0	0	0	0	0	1	1.008	0.984	−5.040	−4.920
Half Scale	1	0	0	0	0	0	0	0	1.000	0.992	−5.000	−4.960
Half Scale−LSB	0	1	1	1	1	1	1	1	0.992	1.000	−4.960	−5.000
Zero Scale+LSB	0	0	0	0	0	0	0	1	0.008	1.984	−0.040	−9.920
Zero Scale	0	0	0	0	0	0	0	0	0.000	1.992	0.000	−9.960

FIGURE 6. Basic Unipolar Negative Operation (Note 5)

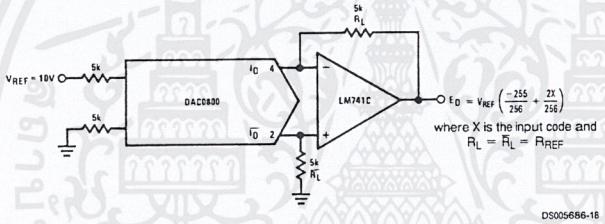
Typical Applications (Continued)



DS005686-6

	B1	B2	B3	B4	B5	B6	B7	B8	E_o	$\bar{E}_o$
Pos. Full Scale	1	1	1	1	1	1	1	1	−9.920	+10.000
Pos. Full Scale−LSB	1	1	1	1	1	1	1	0	−9.840	+9.920
Zero Scale+LSB	1	0	0	0	0	0	0	1	−0.080	+0.160
Zero Scale	1	0	0	0	0	0	0	0	0.000	+0.080
Zero Scale−LSB	0	1	1	1	1	1	1	1	+0.080	0.000
Neg. Full Scale+LSB	0	0	0	0	0	0	0	1	+9.920	−9.840
Neg. Full Scale	0	0	0	0	0	0	0	0	+10.000	−9.920

FIGURE 7. Basic Bipolar Output Operation (Note 5)

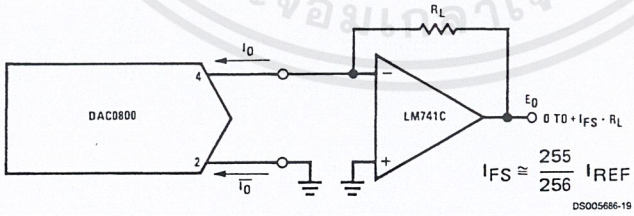


DS005686-18

If $R_L = \bar{R}_L$ within ±0.05%, output is symmetrical about ground

	B1	B2	B3	B4	B5	B6	B7	B8	E_o
Pos. Full Scale	1	1	1	1	1	1	1	1	+9.960
Pos. Full Scale−LSB	1	1	1	1	1	1	1	0	+9.880
(+)Zero Scale	1	0	0	0	0	0	0	0	+0.040
(−)Zero Scale	0	1	1	1	1	1	1	1	−0.040
Neg. Full Scale+LSB	0	0	0	0	0	0	0	1	−9.880
Neg. Full Scale	0	0	0	0	0	0	0	0	−9.960

FIGURE 8. Symmetrical Offset Binary Operation (Note 5)

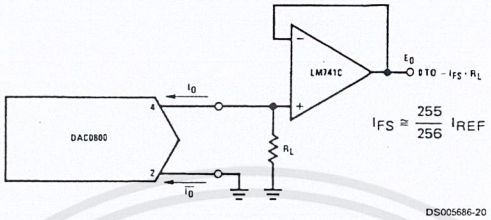


DS005686-19

For complementary output (operation as negative logic DAC), connect inverting input of op amp to $\bar{I}_O$ (pin 2), connect I_O (pin 4) to ground.

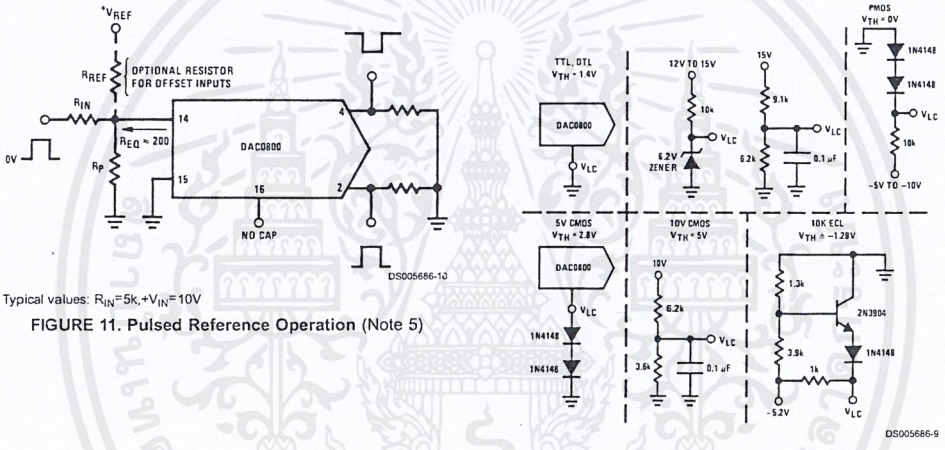
FIGURE 9. Positive Low Impedance Output Operation (Note 5)

Typical Applications (Continued)



For complementary output (operation as a negative logic DAC) connect non-inverting input of op am to $\overline{I_O}$ (pin 2); connect I_O (pin 4) to ground.

FIGURE 10. Negative Low Impedance Output Operation (Note 5)

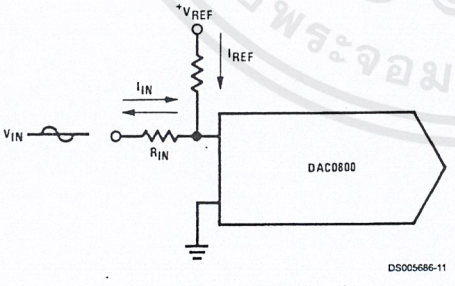


Typical values: R_{IN}=5k,+V_{IN}=10V

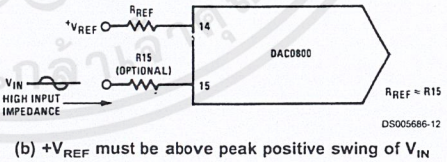
FIGURE 11. Pulsed Reference Operation (Note 5)

$V_{TH} = V_{LC} + 1.4V$
15V CMOS, HTL, HNIL
V_{TH} = 7.6V
Note. Do not exceed negative logic input range of DAC.

FIGURE 12. Interfacing with Various Logic Families



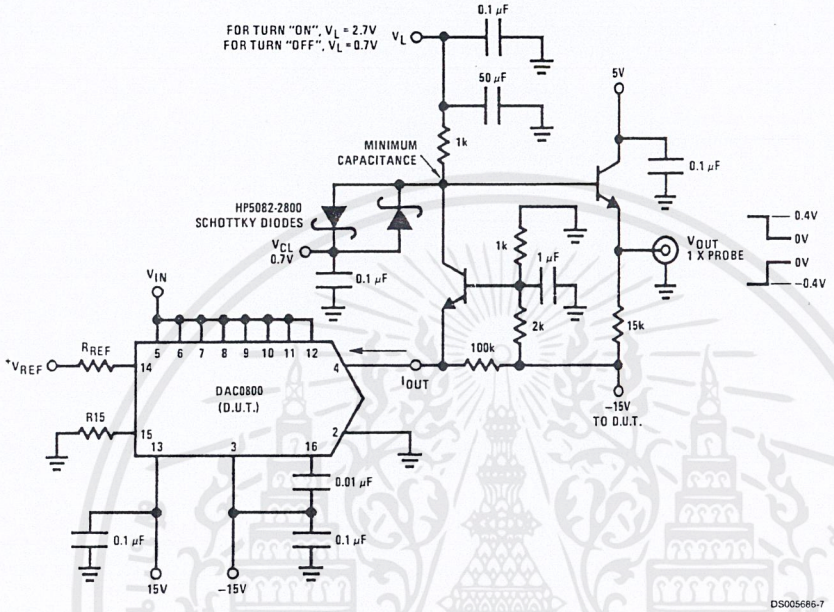
(a) $I_{REF} \geq$ peak negative swing of I_{IN}



(b) $+V_{REF}$ must be above peak positive swing of V_{IN}

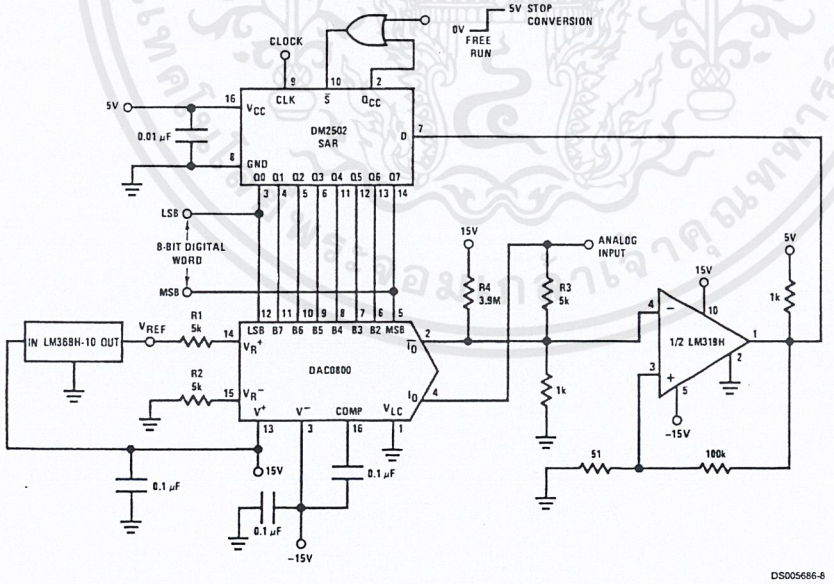
FIGURE 13. Accommodating Bipolar References (Note 5)

Typical Applications (Continued)



DS005686-7

FIGURE 14. Settling Time Measurement (Note 5)

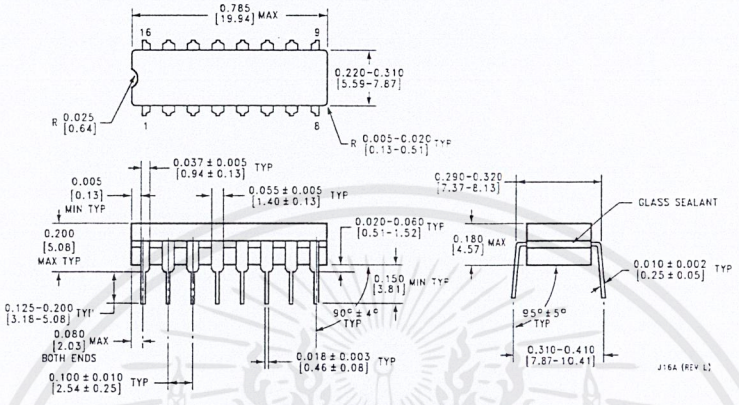


DS005686-8

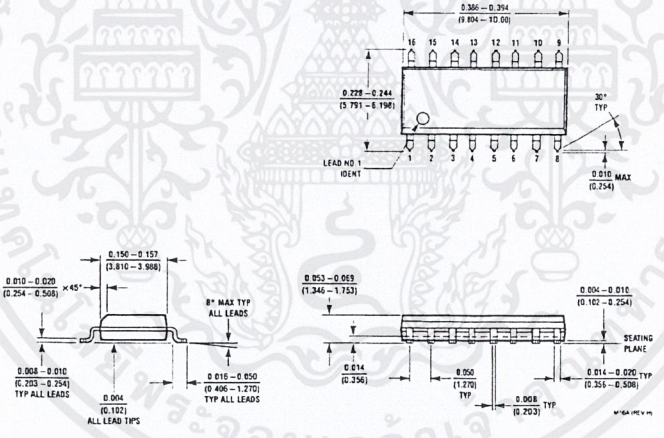
Note. For 1 µs conversion time with 8-bit resolution and 7-bit accuracy, an LM361 comparator replaces the LM319 and the reference current is doubled by reducing R1, R2 and R3 to 2.5 kΩ and R4 to 2 MΩ.

FIGURE 15. A Complete 2 µs Conversion Time, 8-Bit A/D Converter (Note 5)

Physical Dimensions inches (millimeters) unless otherwise noted

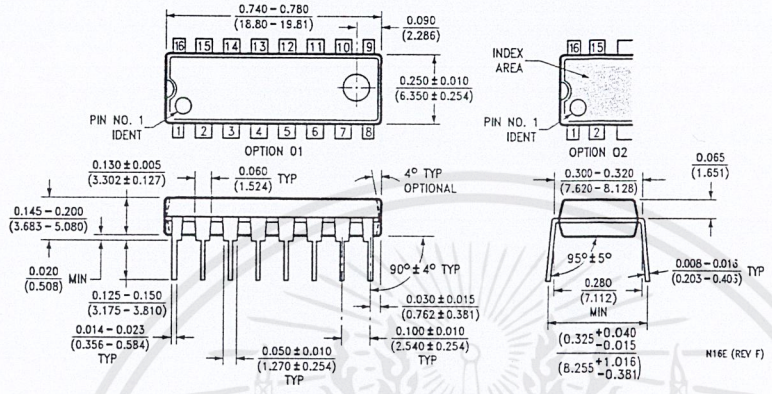


Molded Small Outline Package (SO)
Order Numbers DAC0800LCM,
or DAC0802LCM
NS Package Number M16A



Molded Small Outline Package (SO)
Order Numbers DAC0800LCM,
or DAC0802LCM
NS Package Number M16A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Molded Dual-In-Line Package
Order Numbers DAC0800, DAC0802
NS Package Number N16E

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ จะไม่สำเร็จลุล่วงลงได้ถ้าไม่ได้รับความเอาใจใส่ และความช่วยเหลือด้วยดี จากอาจารย์ที่ปรึกษา อ.สุเธียร เกียรติสุนทร และขอขอบคุณ ย.ถาวร เบญจนราสุทธิ์เป็นอย่างมากสำหรับคำปรึกษาที่ดี ยอมสละเวลาทุกครั้งแม้นว่าจะไม่ว่างเพียงใด ให้ความเป็นกันเองกับนักศึกษา

ขอขอบคุณ บิดา มารดา ที่ทำให้ข้าพเจ้าได้ลืมตาดูโลก ให้การศึกษา เลี้ยงดู เอาใจใส่ จนมีวันนี้ขึ้นมาได้ ขอขอบคุณแผ่นดิน ทรัพยากรธรรมชาติ ที่ทำให้ข้าพเจ้ากินดีอยู่ดี ขอขอบคุณธรรมชาติที่ให้ข้าพเจ้าได้เรียนรู้ชีวิตและสิ่งต่างๆที่ผ่านเข้ามาในชีวิต ขอขอบคุณ ทศ,โอ๊ต,ปุ๊ก,ติ๊ด,หุย,โอ๋ ที่ช่วยเหลือทำให้รายงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี และสุดท้ายคือเพื่อนๆภาคคอนโทรลทุกๆคนและทุกๆคนที่เกี่ยวข้องที่ให้รอยยิ้ม เสียงหัวเราะ คำปลอบใจดีๆ

ขอขอบคุณคณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ทำให้ข้าพเจ้าได้รับสิ่งดีๆ ในรั้วมหาวิทยาลัย ได้รับความรู้และประสบการณ์ต่างๆมากมาย จากอาจารย์ที่ร่วมกันประสิทธิ์ประสาทวิชาความรู้ให้

นาย พูนพิพัฒน์ ศรีสุพรรณ

นาย โอฬาร อมรวัฒนา

หนังสืออ้างอิง

1.THE MATH WORKS Inc,"Real Time Workshop for Use with Simulink ",pp.1-1 to 3-41

2.THE MATH WORKS Inc,"Simulink Dynamic System Simulaton for Matlab",pp.1-1to1-13 and 3-2 to 3-68

THE MATH WORKS Inc,"Realtime Window Target For Use with Real Time Workshop",pp.1-2 to 1-4

กฤษดา ใจเย็น,ชัยวัฒน์ ลิ้มพรจิตรวิไล," เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตขนาน" ,หน้า 1-19 และ หน้า 91-93

อาจารย์ ธันวา ศรีประมง,"การเรียนรู้การเขียนภาษาซี ",หน้า1-230