ตัวควบคุมเวลาจริงแบบอะแดปทีฟ พีไอดี โดยบอร์ดประมวลผลสัญญาณดิจิตอล

Real-Time Adaptive PID Controller Based on DSP Board

โดย

นาย ธีรวุฒิ ธนสารโสภิณ

นาย ปริญญา วัฒนนุกูลชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์เชิงกล

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ตัวควบคุมเวลาจริงแบบอะแดปทีฟ พีไอดี
โดยบอร์ดประมวลผลสัญญาณดิจิตอล
Real-Time Adaptive PID Controller Based on DSP Board

จัดทำโดย

นาย ธีรวุฒิ ธนสารโสภิณ

นาย ปริญญา วัฒนนุกูลชัย

อาจารย์ที่ปรึกษา

รศ. สุเธียร เกียรติสุนทร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์เชิงกล

ภาควิชาวิศวกรรมระบบควบคุมสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

สาขาวิศวกรรมอิเล็กทรอนิกส์เชิงกล

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ตัวควบคุมเวลาจริงแบบอะแดปทีฟ พีไอดี โดยบอร์ดประมวลผลสัญญาณดิจิตอล

Real-Time Adaptive PID Controller Based on DSP Board

ผู้จัดทำ

นาย ธีรวุฒิ ธนสารโสภิณ รหัส 44010223

นาย ปริญญา วัฒนนุกูลชัย รหัส 44010293

..... อาจารย์ที่ปรึกษา

(รศ. สุเธียร เกียรติสุนทร)

ตัวควบคุมเวลาจริงแบบอะแดปทีฟ พีไอดี โดยบอร์ดประมวลผลสัญญาณดิจิตอล

Real-Time Adaptive PID Controller Based on DSP Board

โดย
นาย ธีรวุฒิ ธนสารโสภิณ
นาย ปริญญา วัฒนนุกูลชัย

อาจารย์ที่ปรึกษา
รศ. สุเธียร เกียรติสุนทร

ปีการศึกษา 2547

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการศึกษาโครงสร้างสถาปัตยกรรมภายในและการประยุกต์ใช้งานอุปกรณ์ประมวลผลสัญญาณดิจิตอล บนบอร์ดดีเอสเคที่มีไมโครโปรเซสเซอร์ทีเอ็มเอส320ซี31 เป็นตัวประมวลผลกลาง โดยจะเน้นการเชื่อมต่อสัญญาณอนาลอกผ่านพอร์ตอนุกรมของดีเอสพี โดยประยุกต์ใช้บอร์ดเป็นตัวควบคุมความเร็วของมอเตอร์กระแสตรง ด้วยวิธีการควบคุมแบบอะแดปทีฟ พีไอดี ตัวควบคุมพีไอดีจะมีการปรับค่าพารามิเตอร์อัตโนมัติเพื่อให้ได้เอาท์พุตที่มีสมรรถนะตามต้องการ

Abstract

This project is the study of the architecture of DSP and its application on the DSK (DSP Starter Kit) Board. The study emphasizes the analog interface through serial port of DSP. The DSP board application is using it as an adaptive PID controller in order to control the speed of DC motor which the DSK automatically updates the PID controller parameters to achieve the desired system output.

สารบัญ

ภาคผนวก

กิตติกรรมประกาศ

เอกสารอ้างอิง

สารบัญรูปภาพ

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากในปัจจุบัน ในการควบคุมระบบใดๆ ก็ตามโดยส่วนใหญ่จะเป็นการควบคุมลักษณะกึ่งอัตโนมัติ กล่าวคือในระบบการควบคุมต้องมีตัวควบคุม ควบคุมแทนมนุษย์ แต่อย่างไรก็ตามก็ต้องคอยมีการปรับแต่งค่าพารามิเตอร์ของตัวควบคุมอยู่เสมอ เนื่องจากระบบต่าง ๆ ค่าพารามิเตอร์ของระบบจะมีการเปลี่ยนแปลงตามสภาพแวดที่ไม่อาจทำนายได้จากทั้งภายในและภายนอกระบบ ซึ่งในการปรับแต่งต้องอาศัยผู้มีประสบการณ์ในการปรับแต่งค่าพารามิเตอร์ เพื่อให้ระบบมีสมรรถภาพดีที่สุดและถ้าผู้ปรับแต่งค่าพารามิเตอร์ไม่มีประสบการณ์ก็จะทำให้เสียเวลาอย่างมาก ในการ ปรับแต่งค่าพารามิเตอร์

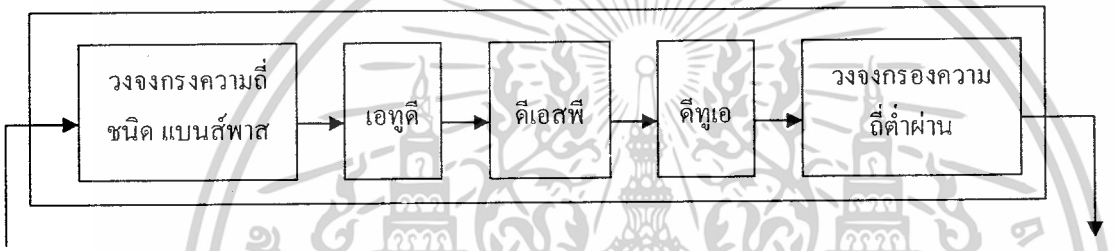
ดังนั้นในปริญญานิพนธ์ฉบับนี้ จะเป็นการประยุกต์ใช้งานของบอร์ด ดีเอสเค ซึ่งมี ดีเอสพี เบอร์ ทีเอ็มเอส320ซี31-50 เป็นตัวประมวลผลเป็นตัวควบคุม ในการควบคุมความเร็วของ ดีซี มอเตอร์ โดยใช้วิธีการควบคุมแบบอะแดปทีฟ พีไอดี (adaptive PID)

ความหมายของระบบอะแดปทีฟ (adaptive system) นั้นมีการอธิบายไว้หลากหลายความหมาย แต่โดยทั่วไปแล้ว ระบบอะแดปทีฟ (adaptive system)นั้นจะหมายถึง ระบบซึ่งมีความสามารถในการปรับตัวของระบบ ตามสภาพแวดล้อมที่เปลี่ยนแปลงไป ถึงแม้ว่าการเปลี่ยนแปลงที่เกิดขึ้น จะเกิดขึ้นจากภายในระบบ หรือจากภายนอกระบบก็ตาม ตัวอย่างของการเปลี่ยนแปลงของระบบตามสภาพแวดล้อม เช่น ระบบการประมวลสัญญาณไฟฟ้า โดยการใช้อุปกรณ์ทางอนาลอกนั้น มักนำตัวต้านทาน ตัวเก็บประจุ และทรานซิสเตอร์ นำมาประกอบเป็นวงจรเพื่อจัดรูปสัญญาณตามที่เราต้องการ ข้อด้อยที่หลีกเลี่ยงไม่ได้คือ การที่ค่าของอุปกรณ์อาจเปลี่ยนแปลงไปได้ตามอุณหภูมิ (สภาพแวดล้อมภายนอกของระบบ) ซึ่งจะส่งผลให้การทำงานผิดพลาดไปด้วย โดยเฉพาะอย่างยิ่งถ้าทำงานที่ความถี่สูง การประกอบวงจรจะเป็นไปด้วยความยากลำบาก เนื่องจากสัญญาณรบกวน และการเดินทางของคลื่นไฟฟ้าในความถี่สูงๆนั้นจะเดินทางไปที่ผิวของวัตถุ

ในปัจจุบันการประมวลสัญญาณไฟฟ้านั้น ได้นำ คอมพิวเตอร์มาใช้ในการประมวลซึ่งสามารถทำได้ง่ายขึ้น สามารถตัดข้อเสียของอุปกรณ์เรื่องอุณหภูมิได้ เพราะสามารถออกแบบวงจรด้วย การเขียนโปรแกรมขึ้นมาได้ และสามารถเปลี่ยนแปลงค่าพารามิเตอร์ต่างๆ ได้ง่ายโดยเปลี่ยนแปลงค่าในโปรแกรมเท่านั้น ไม่ต้องเปลี่ยนแปลงอุปกรณ์ภายนอก สามารถจำลองดูผลการทำงาน บันทึกผลไว้ประมวลผลครั้งต่อไปได้และหากเป็นการกระทำกับ สัญญาณที่มีความถี่สูงๆ ก็สามารถใช้อุปกรณ์เชื่อมต่อกับคอมพิวเตอร์เพื่อประมวลผลโดย ฮาร์ดแวร์ ซึ่งจะมีความเร็วสูงมาก เช่น ดีเอสพี มีลักษณะเป็น ไมโครโปรเซสเซอร์ ซึ่งสามารถทำงานได้ทั้งโหมด

ไมโครโปรเซสเซอร์ และโหมด ไมโครคอนโทรลเลอร์ โดย ดีเอสพี นี้เป็นไมโครโปรเซสเซอร์ ที่มีความเร็วสูงมาก มีชุดคำสั่งพิเศษ มีสถาปัตยกรรมภายในเหมาะสมสำหรับการประมวลสัญญาณทางตัวเลข ซึ่งสามารถนำไปประยุกต์ใช้ในงานทางด้าน การควบคุมระบบต่างๆได้

ในการประมวลสัญญาณอนาลอก ของ ดีเอสพี นี้จะต้องมีการเชื่อมต่อ ไอซี ทีแอลซี 32040 ซึ่งจะมีอยู่ในบอร์ด ดีเอสเค แล้วโดยใน ทีแอลซี 32040 ประกอบด้วย อินพุต / เอาท์พุต อนาลอก ผ่าน อาร์ซีเอแจ็คหรือ ขา 3 ของตัวเชื่อมต่อ JP3, เอทูดีและ ดีทูเอ ขนาด 14 บิต และมีอัตราการแซมปลิ้งสูงสุด 20กิโลเฮิรต์ซ. วงจรกรองความถี่ด้านอินพุต ชนิดแบนด์พาส โดยสามารถบายพาสได้. วงจรกรองความถี่ด้านเอาท์พุตเป็นชนิดวงจรกรองความถี่ต่ำผ่าน โดยมีการเชื่อมต่อสัญญาณอนาลอก ดังรูป



รูปที่ 1-1 โครงสร้างของบอร์ดดีเอสเคที่จะนำไปควบคุมกระบวนการ

ซึ่งในปริญญานิพนธ์ฉบับนี้ได้มีการออกแบบตัวควบคุมโดยที่ตัวควบคุมแบบพีไอดี ได้มีการเจาะจงค่าโพลของระบบที่จุดกำเนิดตัวหนึ่งที่ระนาบตัวแปรเชิงซ้อน s (s-plane) หรือ เท่ากับจุด 1 บนระนาบตัวแปรเชิงซ้อน z (z-plane) และ ซีโร่ที่เปลี่ยนค่าได้ 2 ตัว และเนื่องจากค่าโพลที่ถูกกำหนดเจาะจงไว้แล้ว ดังนั้นสิ่งที่เราจะต้องทำก็คือเราจะต้องทำที่ซีโรที่ปรับค่าได้2ตัว

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาโครงสร้าง การทำงาน และการเขียนโปรแกรม ของบอร์ด DSK
- เพื่อศึกษาทฤษฎีการควบคุมแบบ adaptive
- เพื่อศึกษาทฤษฎีการควบคุมแบบ PID
- เพื่อศึกษาผลตอบสนองของระบบที่มีการควบคุมแบบ PID
- เพื่อศึกษาผลตอบสนองของระบบที่มีการควบคุมแบบ Adaptive PID
- เพื่อศึกษาการประมวลผลสัญญาณอนาลอก และ ดิจิตอล

1.3 ประโยชน์ที่คาดว่าจะได้รับ

- มีความรู้ความสามารถในการประยุกต์ใช้งานบอร์ด DSK
- มีความรู้ความเข้าใจในระบบควบคุมแบบ PID
- มีความรู้ความเข้าใจในระบบควบคุมแบบ Adaptive PID
- มีความรู้ความเข้าใจในการประมวลผลสัญญาณอนาลอก และดิจิตอล

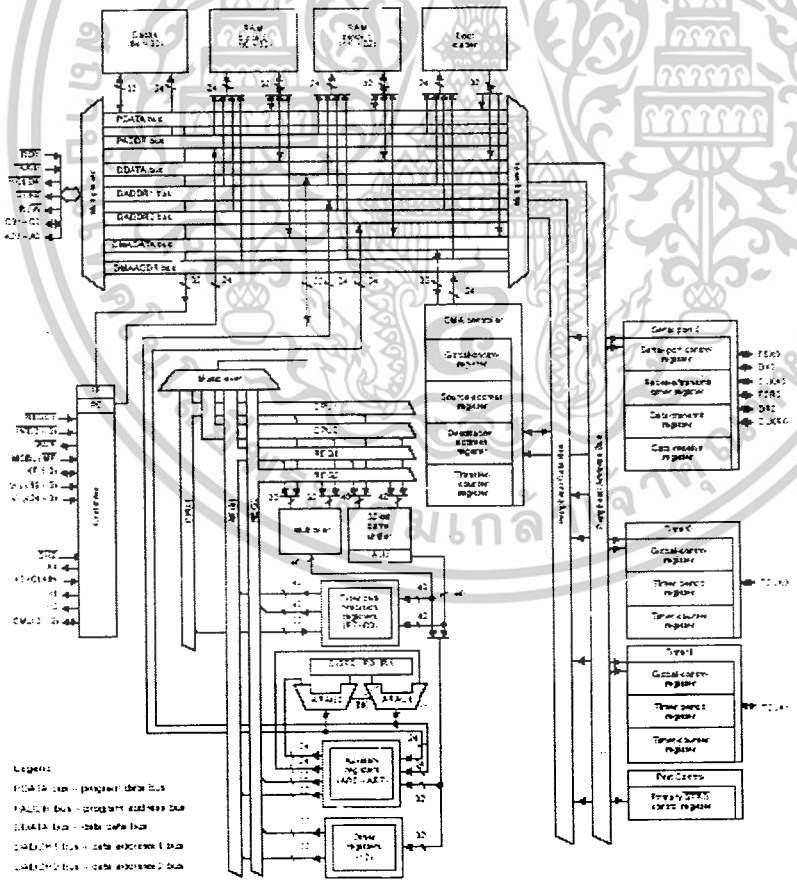
บทที่ 2
ทฤษฎี

ในบทนี้จะกล่าวถึงโครงสร้างของส่วนบอร์ด ดีเอสเคและวิธีการคำนวณหาค่าพารามิเตอร์ของตัวควบคุมแบบ Adaptive PID ดังนี้

2.1 โครงสร้างบอร์ดดีเอสเค

2.1.1 สถาปัตยกรรม TMS320C3x

สถาปัตยกรรม TMS320C3x สามารถตอบสนองความต้องการพื้นฐานของขั้นตอนวิธีคำนวณที่ซับซ้อน และเน้นการแก้ปัญหาทั้งทางด้านฮาร์ดแวร์และซอฟท์แวร์ หน่วยอิงดรรชนีมีพิสัยแบบพลวัต (dynamic range) ที่กว้างและเที่ยงตรง หน่วยความจำบนชิปขนาดใหญ่ ระบบแบบขนานดีกรีสูง และตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง ทำให้ TMS320C3x มีสมรรถภาพสูง



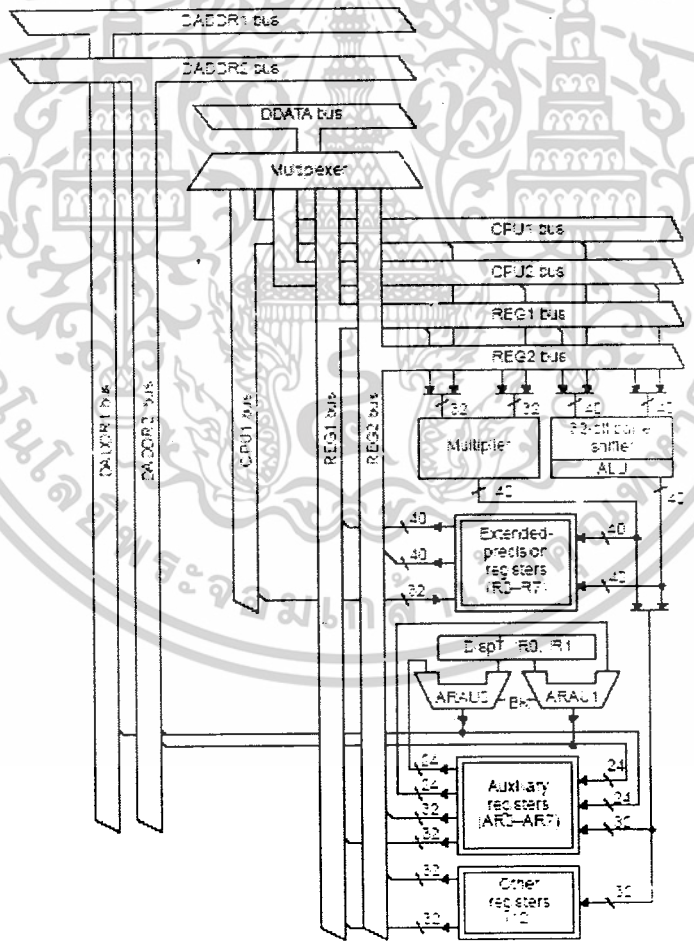
รูปที่ 2-1 บล็อกไดอะแกรมของ TMS320C3x

2.1.1.1 หน่วยประมวลผลกลาง (Central Processing Unit : CPU)

ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- ตัวคูณเลขอิงดรรชนี/จำนวนเต็ม (Floating-point/integer multiplier)
- หน่วยคำนวณและตรรกะ (Arithmetic logic unit : ALU) สำหรับไว้ดำเนินการเลขอิงดรรชนี จำนวนเต็ม และตรรกะ
- ตัวเลื่อนหรือหมุนเวิร์ดข้อมูล (barrel shifter) 32 บิต
- บัสภายใน (CPU1/CPU2 และ REG1/REG2)
- หน่วยคำนวณรีจิสเตอร์ช่วย (Auxiliary register arithmetic units : ARAUs)
- แฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง (CPU register file)

รูปที่ 2-2 แสดงอุปกรณ์ต่างๆ ของหน่วยประมวลผลกลาง โดยจะตามหลังด้วยคำอธิบาย



รูปที่ 2-2 หน่วยประมวลผลกลาง

2.1.1.1.1 ตัวคูณ (Multiplier)

ตัวคูณจะทำการคูณแบบรอบเดียว โดยเป็นค่าจำนวนเต็ม 24 บิต และค่าอิงดรรชนี 32 บิต วิธีการคำนวณเลขอิงดรรชนีของ TMS320C3x จะให้ทำงานที่ความเร็วไม่อิงดรรชนี (fixed-point) ผ่านทางรอบคำสั่ง 50 นาโนวินาที และระบบแบบขนานดีกรีสูง

เมื่อตัวคูณทำการคูณเลขอิงดรรชนี อินพุตจะเป็นเลขอิงดรรชนี 32 บิต และได้ผลลัพธ์เป็นเลขอิงดรรชนี 40 บิต

เมื่อตัวคูณทำการคูณเลขจำนวนเต็ม อินพุตจะเป็นเลขจำนวนเต็ม 24 บิต และได้ผลลัพธ์เป็นเลขจำนวนเต็ม 32 บิต

2.1.1.1.2 หน่วยคำนวณและตรรกะ (ALU)

หน่วยคำนวณและตรรกะ แสดงการจัดการข้อมูลจำนวนเต็ม 32 บิต ตรรกะ 32 บิต และอิงดรรชนี 40 บิต แบบรอบเดียว รวมทั้งการเปลี่ยนจำนวนเต็มและอิงดรรชนี แบบรอบเดียว ผลลัพธ์ของหน่วยคำนวณและตรรกะจะยังคงเป็นเลขจำนวนเต็ม 32 บิตหรืออิงดรรชนี 40 บิตเสมอ ส่วนตัวเลื่อนหรือหมุนเวิร์ดข้อมูล จะใช้เพื่อเลื่อนหรือหมุนบิตจนถึง 32 บิตซ้ายหรือขวาภายในรอบเดียว

บัสภายใน CPU1/CPU2 และ REG1/REG2 บรรจุ 2 ตัวถูกดำเนินการ (operand) จากหน่วยความจำ และบรรจุ 2 ตัวถูกดำเนินการจากแฟ้มรีจิสเตอร์ ดังนั้นจึงสามารถคูณแบบขนาน และบวก/ลบเลขจำนวนเต็มหรือเลขอิงดรรชนี 4 ตัวได้ในรอบเดียว

2.1.1.1.3 หน่วยคำนวณรีจิสเตอร์ช่วย (ARAUs)

ARAU0 และ ARAU1 สามารถสร้างคำนวณรีจิสเตอร์ 2 ตำแหน่งในรอบเดียว หน่วยคำนวณรีจิสเตอร์ช่วย ทำงานแบบขนานด้วยตัวคูณและALU ซึ่งทั้ง 2 ตัวนี้สนับสนุนการเข้าถึงตำแหน่งด้วยการแทนที่ (addressing with displacements) รีจิสเตอร์ดรรชนี (index register : IR0 และ IR1) การเข้าถึงตำแหน่งแบบวนรอบ (circular addressing) และการเข้าถึงตำแหน่งแบบกลับบิต (bit-reversed addressing)

2.1.1.1.4 แฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง (CPU register file)

TMS320C3x จะจัดให้รีจิสเตอร์ 28 ตัวอยู่ในแฟ้มรีจิสเตอร์ที่มีทางเข้าออกหลายทาง ซึ่งติดกับหน่วยประมวลผลกลาง ส่วนตัวนับโปรแกรม (Program Counter : PC) จะไม่รวมอยู่ในรีจิสเตอร์ 28 ตัวข้างต้น รีจิสเตอร์ทั้งหมดนี้สามารถทำงานโดยตัวคูณหน่วยคำนวณและตรรกะ และสามารถใช้เป็นรีจิสเตอร์อเนกประสงค์ 32 บิต นอกจากนี้รีจิสเตอร์ยังมีฟังก์ชันพิเศษ เช่น รีจิสเตอร์เพิ่มเติม (extended-precision) 8 ตัวจะเหมาะสำหรับเก็บเลขผลลัพธ์แบบอิง

ดรรชนี รีจิสเตอร์ช่วย 8ตัว จะสนับสนุนโหมดการเข้าถึงตำแหน่งโดยอ้อม (indirect addressing mode) ที่หลากหลาย และสามารถใช้เป็นรีจิสเตอร์จำนวนเต็ม 32บิตทั่วไป และ รีจิสเตอร์แบบตรรกะ รีจิสเตอร์ที่เหลือจะจัดให้แต่ละฟังก์ชันของระบบ เช่น การเข้าถึงตำแหน่ง การจัดการสแตก (stack) สถานะของตัวประมวลผล การอินเทอร์รัพต์ และบล็อก (block) ที่ทำซ้ำ

ชื่อของรีจิสเตอร์และฟังก์ชันที่กำหนด จะอยู่ในตาราง 2-1 ตามตารางจะบรรยายฟังก์ชันของแต่ละรีจิสเตอร์หรือกลุ่มของรีจิสเตอร์ไว้อย่างย่อๆ

ตาราง 2-1 แสดงรีจิสเตอร์ของหน่วยประมวลผลกลาง

Register name	Assigned Function
R0 – R7	Extended – precision register 0 – 7
AR0 – AR7	Auxiliary register 0 – 7
DP	Data – page pointer
IR0	Index register 0
IR1	Index register 1
BK	Block size
SP	System stack pointer
ST	Status register
IE	CPU / DMA Interrupt enable
IF	CPU interrupt flags
IOF	I/O flags
RS	Repeat start address
RE	Repeat end address
RC	Repeat counter

R0 – R7	รีจิสเตอร์เพิ่มเติม (extended–precision register) 0–7 มีความสามารถในการจัดเก็บและสนับสนุนการจัดการเลขจำนวนเต็ม32บิต และเลขอิงดรรชนี 40 บิต
AR0 - AR7	รีจิสเตอร์ช่วย (auxillary register) 0-7 สามารถเข้าถึงได้โดยหน่วยประมวลผลกลาง และแก้ไขโดยหน่วยคำนวณรีจิสเตอร์ช่วย 2ตัวโดยที่ฟังก์ชันพื้นฐานของรีจิสเตอร์ช่วยทำให้เกิดตำแหน่ง 24บิต ซึ่งสามารถใช้เป็นตัวนับลูป (loop counter) หรือ เป็นรีจิสเตอร์อเนกประสงค์ 32บิต ซึ่งสามารถแก้ไขโดยตัวคูณและหน่วยเลขคณิตแบบตรรกะ
DP	ตัวชี้เพจของข้อมูล (data page pointer) คือ รีจิสเตอร์ 32บิต โดยที่ 8บิตล่างถูกใช้เป็นโหมดการเข้าถึงตำแหน่งโดยตรง (direct addressing mode) เสมือนเป็นตัวชี้ตำแหน่งเพจของข้อมูล เพจของข้อมูลมีความยาว64K เวิร์ด (word) จากทั้งหมด 256 เพจ
IR0, IR1	รีจิสเตอร์ดรรชนี 32 บิต จะเก็บค่าให้กับหน่วยคำนวณรีจิสเตอร์ช่วย เพื่อคำนวณตำแหน่งที่ชี้
BK	รีจิสเตอร์ขนาดบล็อก (block size register) 32บิต ถูกใช้โดยหน่วยคำนวณรีจิสเตอร์ช่วยในการเข้าถึงตำแหน่งแบบวนรอบ (circular addressing) เพื่อระบุขนาดของบล็อกข้อมูล (data block)
SP	ตัวชี้สแตกของระบบ (system stack pointer) คือ รีจิสเตอร์ 32บิต ซึ่งเก็บตำแหน่งของสแตกบนสุด (top stack) ซึ่ง SP จะชี้อยู่ที่ตัวสุดท้ายที่ถูกพุช (push) ลงมา (การพุชนั้นก่อนพุช ค่าของ SP จะถูกเพิ่มก่อนทุกครั้ง ส่วนการป็อป (pop) จะทำการป็อปก่อนจึงจะลดค่าใน SP) ค่า SP เปลี่ยนแปลงตามการอินเทอร์รัพต์ แทร็ป (trap) เรียก (call) รีเทิร์น (return) คำสั่งพุช และคำสั่งป็อป
ST	รีจิสเตอร์สถานะ (status register) เก็บข้อมูลทั่วไปที่เกี่ยวข้องกับสถานะของหน่วยประมวลผลกลาง
IE	รีจิสเตอร์อินเทอร์รัพต์หน่วยประมวลผลกลาง/การเข้าถึงหน่วยความจำโดยตรง (CPU/DMA interrupt enable register) เป็น รีจิสเตอร์ 32บิต บิตอินเทอร์รัพต์หน่วยประมวลผลกลาง (CPU interrupt enable bit) คือ บิต 10-0 บิตอินเทอร์รัพต์การเข้าถึงหน่วยความจำโดยตรง (DMA interrupt enable bit) คือ บิต 26-16 โดย 1 = ได้ (enable) 0 = ไม่ได้ (disable)
IF	รีจิสเตอร์อินเทอร์รัพต์แฟลกของหน่วยประมวลผลกลาง (CPU interrupt flag register) เป็น รีจิสเตอร์ 32บิต โดย 1 = อินเทอร์รัพต์ 0 = ไม่อินเทอร์รัพต์

IOF รีจิสเตอร์แฟลกของอินพุต/เอาท์พุต (I/O flag register) ควบคุมฟังก์ชันของขาภายนอกที่มีหน้าที่พิเศษ (XF0 และ XF1) โดยที่ขาเหล่านี้อาจถูกกำหนดเป็นอินพุตหรือเอาท์พุตก็ได้

RC ตัวนับการทำซ้ำ (repeat counter) เป็น รีจิสเตอร์32บิต ใช้ระบุจำนวนครั้งที่บล็อกของรหัส (code) ที่ทำซ้ำ เมื่อบล็อกที่ทำซ้ำกำลังทำงาน

RS รีจิสเตอร์ตำแหน่งเริ่มต้นการทำซ้ำ (repeat start address register)32บิต เก็บตำแหน่งเริ่มต้นของบล็อกของหน่วยความจำโปรแกรมที่ทำซ้ำ

RE รีจิสเตอร์ตำแหน่งจบการทำซ้ำ (repeat end address register) 32บิต เก็บตำแหน่งสุดท้ายของบล็อกที่ทำซ้ำ

PC ตัวนับโปรแกรม (program counter) เป็น รีจิสเตอร์ 32บิต เก็บตำแหน่งของคำสั่งต่อไปที่จะไปนำมา (fetch) ถึงแม้ PC จะไม่ใช่ส่วนของแฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง แต่ก็สามารถแก้ไขโดยคำสั่งที่แก้ไขสายงานโปรแกรม

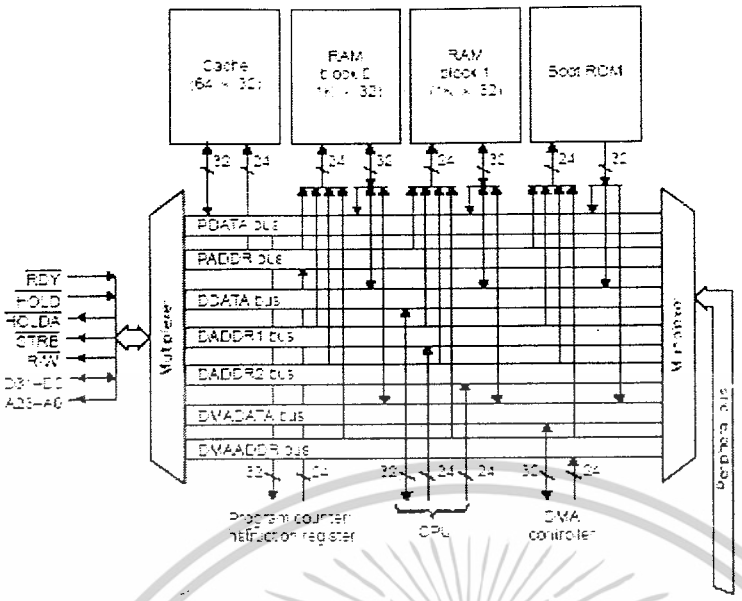
2.1.1.2 การจัดหน่วยความจำของ TMS320C31

TMS320C3x มีพื้นที่ของหน่วยความจำทั้งหมดเป็น 16M (million) 32บิต เวิร์ด โดยบรรจุโปรแกรม ข้อมูล และเนื้อที่ของอินพุต/เอาท์พุตอยู่ภายใน ดังนั้นสามารถเก็บตาราง (table) สัมประสิทธิ์ (coefficient) รหัสของโปรแกรม (program code) หรือข้อมูล ไว้ได้ทั้งใน RAM หรือ ROM ก็ได้ ในกรณีนี้จะทำให้สามารถใช้หน่วยความจำได้มากที่สุด และ จัดพื้นที่หน่วยความจำได้ตามต้องการ

2.1.1.2.1 RAM, ROM และ Cache

รูปที่ 2-3 แสดงการจัดการหน่วยความจำของ TMS320c3x โดยใน TMS320C31 มี RAM 2 ตัว แต่ละตัวมีขนาด 1K x 32 บิต ไม่มี ROM และมี Cache ขนาด 64 x 32 บิต เพื่อเก็บคำสั่งที่ใช้บ่อยๆ (ทำให้ทำงานได้ดีขึ้น)

TMS320C31 นี้มีการแยกหน่วยความจำข้อมูลกับหน่วยความจำโปรแกรมออกจากกัน และทำงานโดยวิธีไปป์ไลน์ (pipeline) จึงทำให้มีความเร็วสูงขึ้น การแยกบัสของโปรแกรม (program bus), บัสของข้อมูล (data bus) และบัสของการเข้าถึงหน่วยความจำโดยตรง (DMA bus) ทำให้สามารถไปนำโปรแกรมมา อ่านข้อมูล และทำการเข้าถึงหน่วยความจำโดยตรงได้พร้อมๆกัน เช่น ในขณะที่หน่วยประมวลผลกลางกำลังจัดการกับข้อมูลอยู่ใน RAM ตัวหนึ่งอยู่ ก็ยังสามารถไปนำโปรแกรมมาโดยเข้าถึงหน่วยความจำโดยตรงผ่าน RAM อีกตัวหนึ่งได้ภายในรอบเดียว



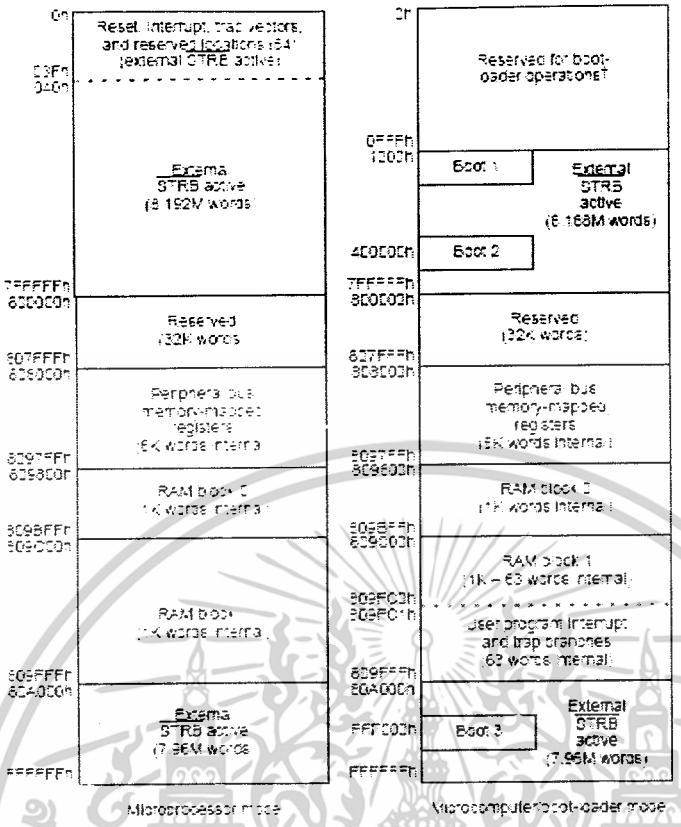
รูปที่ 2-3 การจัดการหน่วยความจำ

2.1.1.2.2 การแทนข้อมูล (Memory Map) ของ TMS320C31

การแทนข้อมูลขึ้นอยู่กับการตั้งค่าขา MCBL/$\overline{MP}$ ว่าอยู่ในโหมดใดของฟังก์ชันดังนี้

- โหมดไมโครโปรเซสเซอร์ (MC/$\overline{MP}$ หรือ MCBL/$\overline{MP}$ = 0)
- โหมดไมโครคอมพิวเตอร์ (MC/$\overline{MP}$ หรือ MCBL/$\overline{MP}$ = 1)

การแทนข้อมูลที่ได้จะคล้ายๆ กัน ดังรูปที่ 2-4



รูปที่ 2-4 การแทนข้อมูลของ TMS320C31

2.1.1.3 การจัดการบัสภายใน

ส่วนใหญ่ TMS3320C3x จะมีสมรรถภาพสูงเพราะบัสภายใน และการแยกแบบขนานของบัสของโปรแกรม (PADDR และ PDATA) บัสของข้อมูล (DADDR1, DADDR2 และ DDATA) และบัสของการเข้าถึงหน่วยความจำโดยตรง (DMAADDR และ DMADATA) จะสามารถไปนำโปรแกรมมาแบบขนาน การเข้าถึงข้อมูล และ การเข้าถึงหน่วยความจำโดยตรง โดยบัสเหล่านี้จะต่อกับพื้นที่ทั้งหมด (หน่วยความจำบนชิป หน่วยความจำนอกชิป และ อุปกรณ์ภายนอกบนชิป) รูปที่ 2-3 แสดงบัสภายในเหล่านี้ และการติดต่อกับบล็อกของหน่วยความจำบนชิปและนอกชิป

ตัวนับโปรแกรมต่ออยู่กับบัสตำแหน่งโปรแกรม 24 บิต (PADDR) ส่วนรีจิสเตอร์คำสั่ง (Instruction register: IR) ต่ออยู่กับบัสข้อมูลของโปรแกรม 32 บิต (PDATA) โดยที่บัสเหล่านี้สามารถไปนำเวิร์ดคำสั่งเดี่ยวมาได้ทุกๆ รอบของเครื่อง

บัสตำแหน่งของข้อมูล 24 บิต (DADDR1 และ DADDR2) และ บัสของข้อมูล 32 บิต (DDATA) จะสนับสนุนการเข้าถึงหน่วยความจำของข้อมูลทั้ง 2 ในทุกรอบของเครื่อง บัส DDATA จะเก็บข้อมูลให้หน่วยประมวลผลกลางผ่านไป บัส CPU1 และ CPU2 จากนั้นบัส

CPU1 และ CPU2 จะเก็บตัวถูกดำเนินการของหน่วยความจำข้อมูล 2 ตัวไปให้ตัวคูณ, หน่วยคำนวณและตรรกะ และแฟ้มรีจิสเตอร์ทุกรอบเครื่อง เช่นเดียวกันกับ บัสของรีจิสเตอร์ REG1 และ REG2 ก็จะเก็บข้อมูล 2 ค่าจากแฟ้มรีจิสเตอร์ไปให้ตัวคูณ และหน่วยคำนวณและตรรกะทุกรอบของเครื่อง รูปที่ 2-2 แสดงบัสภายในไปยังส่วนของหน่วยประมวลผลกลาง

ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA controller) ถูกสนับสนุนโดยบัสของตำแหน่งDMA ADDR 24 บิต และบัสของข้อมูลDMA DATA 32บิต บัสเหล่านี้จะอนุญาตให้การเข้าถึงหน่วยความจำโดยตรงเป็นแบบขนาน โดยใช้บัสของข้อมูล และบัสของโปรแกรม

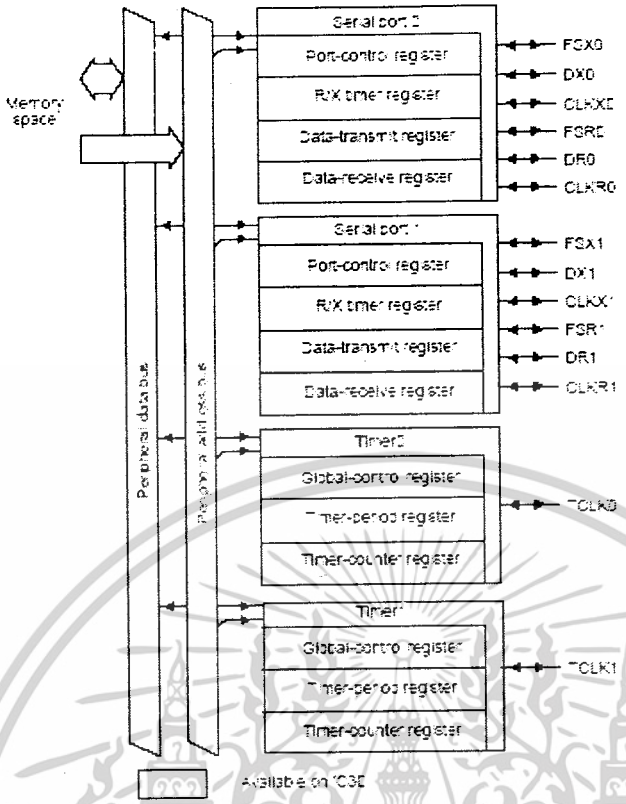
2.1.1.4 อุปกรณ์รอบนอก (Peripherals)

รีจิสเตอร์การแทนข้อมูลจะควบคุมอุปกรณ์ภายนอกของ TMS320C3x ทั้งหมดโดยใช้บัสของอุปกรณ์รอบนอกซึ่งเป็นบัสของข้อมูล 32บิต และบัสของตำแหน่ง 24บิต ที่จะยอมให้ติดต่อกับอุปกรณ์รอบนอกโดยตรง อุปกรณ์รอบนอกของ TMS320C31 ประกอบด้วยตัวจับเวลา (timer) 2 ตัว และพอร์ตอนุกรม (serial port) 1 ตัว รูปที่ 2-5 แสดงอุปกรณ์รอบนอกกับบัสและสัญญาณที่เกี่ยวข้อง

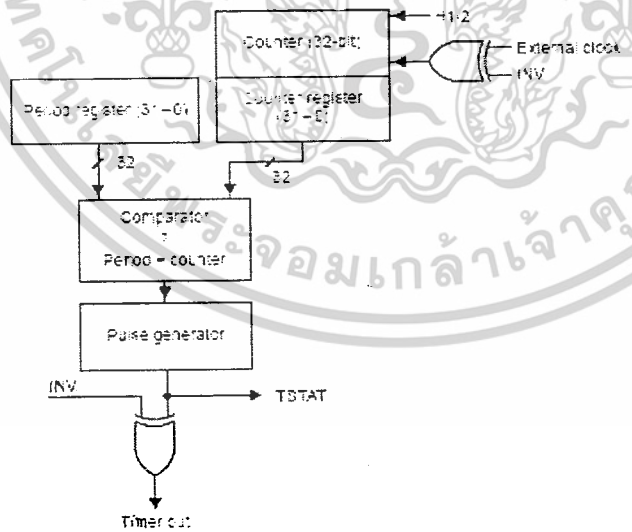
2.1.1.4.1 ตัวจับเวลา (Timer)

ตัวจับเวลา 2 ตัวของ TMS320C3x เป็นตัวจับเวลาอเนกประสงค์ (general-purpose timer) / ตัวนับเหตุการณ์ (event-counter) 32 บิตที่มีโหมดด้านสัญญาณ 2 ตัว และสัญญาณนาฬิกา (clocking) ภายในหรือภายนอก (ดูรูปที่ 2-6) สามารถใช้ตัวจับเวลา เพื่อส่งสัญญาณให้ TMS320C3x หรือโลกภายนอกเป็นช่วงๆ ตามที่กำหนด หรือนับเหตุการณ์ภายนอก

การใช้สัญญาณนาฬิกาภายในจะทำให้สามารถใช้ตัวจับเวลาเพื่อส่งสัญญาณให้ตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (A/D converter) ภายนอกไปเริ่มต้นการเปลี่ยนแปลง หรือสามารถอินเทอร์รัพต์ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงของ TMS320C3x ให้เริ่มต้นส่งย้ายข้อมูล การอินเทอร์รัพต์ตัวจับเวลาเป็นการอินเทอร์รัพต์ภายในแบบหนึ่ง ส่วนการใช้สัญญาณนาฬิกาภายนอกจะทำให้ตัวจับเวลาสามารถนับเหตุการณ์ภายนอกและอินเทอร์รัพต์หน่วยประมวลผลกลางหลังจากระบุจำนวนเหตุการณ์ ตัวจับเวลาแต่ละตัวจะมีขาอินพุต/เอาท์พุต (I/O pin) ซึ่งสามารถใช้เสมือนเป็นสัญญาณนาฬิกาอินพุต (input clock) ให้ตัวจับเวลา สัญญาณนาฬิกาเอาท์พุต (output clock) หรือขาอินพุต/เอาท์พุตอเนกประสงค์ (general-purpose I/O pin)



รูปที่ 2-5 อุปกรณ์ภายนอกกับบัสและสัญญาณที่เกี่ยวข้อง



รูปที่ 2-6 บล็อกไดอะแกรมของตัวจับเวลา

การแทนที่ข้อมูลสำหรับตัวจับเวลา แสดงในรูปที่ 2-7

Register	Peripheral address	
	Timer 0	Timer 1
Timer Global Control	808020h	808030h
Reserved	808021h	808031h
Reserved	808022h	808032h
Reserved	808023h	808033h
Timer Counter	808024h	808034h
Reserved	808025h	808035h
Reserved	808026h	808036h
Reserved	808027h	808037h
Timer Period	808028h	808038h
Reserved	808029h	808039h
Reserved	80802Ah	80803Ah
Reserved	80802Bh	80803Bh
Reserved	80802Ch	80803Ch
Reserved	80802Dh	80803Dh
Reserved	80802Eh	80803Eh
Reserved	80802Fh	80803Fh

รูปที่ 2-7 ตำแหน่งการแทนข้อมูลของตัวจับเวลา

ในแต่ละตัวจับเวลา จะมีรีจิสเตอร์การแทนข้อมูล 3 ชนิด ดังนี้

- รีจิสเตอร์ควบคุมส่วนกลาง (Global-control register) กำหนดโหมดการทำงาน (operating mode) ของตัวจับเวลา ตรวจสอบสถานะของตัวจับเวลา และ ควบคุมฟังก์ชันต่างๆ ของขาอินพุต/เอาท์พุตของตัวจับเวลา
- รีจิสเตอร์คาบเวลา (Period register) ระบุความถี่ทางสัญญาณของตัวจับเวลา
- รีจิสเตอร์ตัวนับ (Counter register) เก็บค่าในขณะปัจจุบันของตัวนับที่เพิ่มค่า (incrementing counter) ซึ่งสามารถเพิ่มค่าตัวจับเวลา โดยการนับขอบขาขึ้นหรือขอบขาลงของตัวนับสัญญาณนาฬิกาอินพุต (input clock counter) ถ้า

ตัวนับเป็น 0 จะสามารถทำให้เกิดการอินเทอร์รัพต์ภายในได้เมื่อค่าของมันเท่ากับรีจิสเตอร์คาบเวลา โดยตัวสร้างสัญญาณพัลส์ (pulse generator) จะสร้างสัญญาณนาฬิกาภายนอก 2 ตัวคือ สัญญาณพัลส์ หรือ สัญญาณนาฬิกา

2.1.1.4.1.1 รีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา (Timer Global-Control Register)

เป็นรีจิสเตอร์ 32 บิต บรรจุบิตควบคุมพอร์ตและทั้งหมด สำหรับตัวจับเวลา ตาราง 2-2 อธิบายบิตของรีจิสเตอร์ ชื่อ และฟังก์ชัน โดย บิต 3-0 เป็น บิตควบคุมพอร์ต; บิต 11-6 เป็นบิตควบคุมส่วนกลางของตัวจับเวลา รูปที่ 2-8 แสดงรีจิสเตอร์ 32 บิต จะสังเกตได้ว่าเมื่อตั้งใหม่ (reset) บิตทุกบิตจะถูกตั้งค่าเป็น 0 ยกเว้น DATIN ที่ถูกตั้งค่าด้วยค่าที่ได้จาก TCLK

R = Read, W = Write, xx = reserved bit, read as 0

รูปที่ 2-8 รีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา

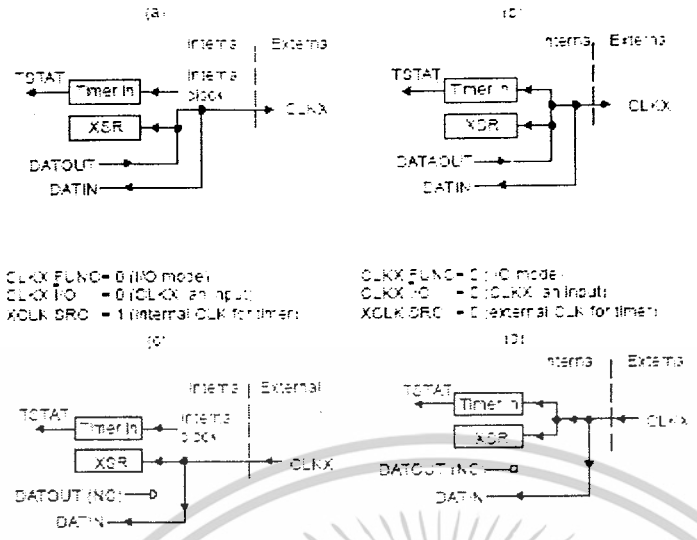
ตาราง 2-2 สรุปบิตของรีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา

Bit	Name	Reset Value	Function
0	FUNC	0	FUNC ควบคุมฟังก์ชันของ TCLK ถ้า FUNC = 0 แล้ว TCLK จะถูกกำหนดคล้ายพอร์ตอินพุต/เอาท์พุตแบบดิจิตอลอเนกประสงค์ ถ้า FUNC = 1 แล้ว TCLK จะถูกกำหนดคล้ายขาตัวจับเวลา (ดูรูปที่ 2-9 สำหรับอธิบายความสัมพันธ์ระหว่าง FUNC และ CLKSRC)
1	I/O	0	ถ้า FUNC = 0 และ CLKSRC = 0 แล้ว TCLK ถูกกำหนดคล้ายกับขาอินพุต/เอาท์พุตอเนกประสงค์ ในกรณีที่ I/O = 0 แล้ว TCLK ถูกกำหนดคล้ายขาอินพุตอเนกประสงค์ ถ้า I/O = 1 แล้ว TCLK ถูกกำหนดเสมือนขาเอาท์พุต

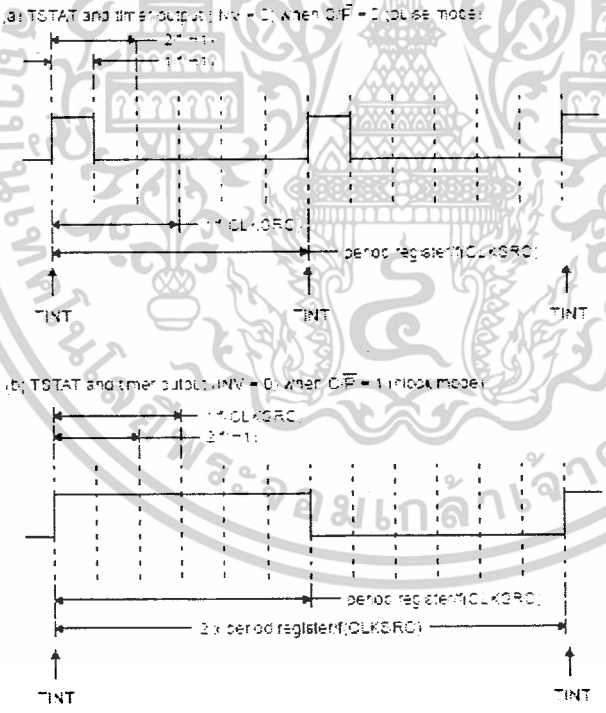
2	DATOUT	0	DATOUT เป็นตัวกำหนด TCLK เมื่อ TMS320C3x อยู่ในโหมดพอร์ตอินพุต/เอาท์พุต สามารถใช้ DATOUT เช่นเดียวกับอินพุตไปสู่ตัวจับเวลา
3	DATIN	x'	อินพุตข้อมูลบน TCLK หรือ DATOUT การเขียนไม่มีผลใดๆ
5-4	Reserved	0-0	สำรอง
6	GO	0	บิต GO ตั้งและเริ่มการทำงานของตัวจับเวลา-ตัวนับใหม่ เมื่อ GO = 1 และตัวจับเวลาไม่ถูกคงค่า (hold) ไว้ ตัวนับจะถูกทำให้เป็น 0 และเริ่มเพิ่มค่าในขอบขาขึ้นถัดไปของสัญญาณนาฬิกาอินพุตของตัวจับเวลา โดยที่บิต GO นี้จะถูกทำให้ว่าง (clear) บนขอบขาขึ้นเดิม ถ้า GO = 0 จะไม่มีผลต่อตัวจับเวลา
7	$\overline{\text{HLD}}$	0	ตัวนับจะคงค่าสัญญาณ เมื่อบิตนี้เป็น 0 โดยที่ตัวนับ จะไม่ทำงานและถูกคงค่าในสถานะปัจจุบัน ถ้าตัวจับเวลากำลังกำหนด TCLK แล้ว สถานะของ TCLK ถูกคงค่าด้วยตัวนับภายในที่ถูกหารด้วย 2 ถูกคงค่าด้วย ดังนั้นตัวนับจะสามารถดำเนินต่อไปได้เมื่อ $\overline{\text{HLD}}$ ถูกตั้งค่าเป็น 1 สามารถอ่านและแก้ไขรีจิสเตอร์ของตัวจับเวลา ในขณะที่ตัวจับเวลากำลังถูกคงค่าได้ แต่ RESET มีลำดับความสำคัญ (priority) มากกว่า $\overline{\text{HLD}}$ ตาราง 2-3 แสดงผลของการเขียน GO และ $\overline{\text{HLD}}$
	$\overline{\text{C/P}}$	0	ควบคุมโหมดสัญญาณนาฬิกา/สัญญาณพัลซ์ เมื่อ $\overline{\text{C/P}}$ = 1 โหมดสัญญาณนาฬิกา จะถูกเลือกและสัญญาณแฟลกของ TSTAT และเอาท์พุตภายนอกจะมีค่ารอบการทำงาน (duty cycle) 50% เมื่อ $\overline{\text{C/P}}$ = 0 แฟลกสถานะและเอาท์พุตภายนอกจะถูกกระตุ้น (active) สำหรับ 1 รอบของ H1 ระหว่างแต่ละคาบเวลาของตัวจับเวลา (ดูรูปที่ 2-10)
9	CLKSRC	0	แหล่งกำเนิด (Source) เฉพาะของสัญญาณนาฬิกาของตัวจับเวลา เมื่อ CLKSRC = 1, สัญญาณนาฬิกาภายในเกี่ยวข้องกับความถี่ที่เท่ากับ 1.5 ของความถี่ของ H1 ถูกใช้ในการเพิ่มของตัวนับ โดยที่บิต INV ไม่มีผลกับ

			แหล่งกำเนิดสัญญาณนาฬิกาภายใน เมื่อ CLKSRC = 0 คุณสามารถใช้สัญญาณภายนอก จากขา TCLK เพื่อเพิ่มตัวนับ. สัญญาณนาฬิกาภายนอกจะถูกทำให้สอดคล้องกัน (synchronize) กับภายใน ดังนั้นการอนุญาตให้แหล่งกำเนิดสัญญาณนาฬิกาที่ไม่สอดคล้องจากภายนอกที่ซึ่งไม่เกินค่าสูงสุดที่กำหนด เป็นความถี่สัญญาณนาฬิกาภายนอกจะน้อยกว่า f(H1) / 2 (ดูรูปที่ 2-9 สำหรับความสัมพันธ์ FUNC และ CLKSRC)
10	INV	0	บิตควบคุมอินเวอร์เตอร์ (Inverter) ถ้าสัญญาณนาฬิกาภายนอกถูกใช้และ INV = 1 แล้วสัญญาณนาฬิกาภายนอกจะถูกเปลี่ยน (invert) เสมือนเป็นตัวนับ ถ้าเอาท์พุตของตัวกำเนิดสัญญาณพัลซ์ถูกกำหนดเส้นทางไปสู่ TCLK และ INV = 1 แล้วจะเอาท์พุตจะถูกเปลี่ยนก่อนถูกส่งไป TCLK ถ้า INV = 0 แล้วจะไม่มีการเปลี่ยนแสดงบนอินพุต::รือเอาท์พุตของตัวจับเวลา โดยที่บิต INV จะมีผลเมื่อ TCLK ถูกใช้ในโหมดพอร์ตอินพุต/เอาท์พุต
11	TSTAT	0	บิตแสดงสถานะของตัวจับเวลา โดยจะติดตามเอาท์พุตของขา TCLK ที่ไม่ถูกเปลี่ยน แฟลกนี้จะตั้งค่าอินเทอร์รัพต์ โดยการส่งผ่านจาก 0 เป็น 1 ในการเขียนไม่มีผล
31-12	Reserved	0-0	สำรอง

x'=0 or 1



รูปที่ 2-9 โหมดของตัวจับเวลาที่กำหนดโดย CLKSRC และ FUNC



รูปที่ 2-10 ช่วงเวลาของตัวจับเวลา (Timer Timing)

อัตราของการส่งสัญญาณของตัวจับเวลากำหนดโดยความถี่ของสัญญาณนาฬิกาอินพุตของตัวจับเวลาและรีจิสเตอร์คาบเวลาของตัวจับเวลา ดังสมการข้างล่างโดยมีผลทั้งสัญญาณนาฬิกาของตัวจับเวลาภายในหรือภายนอกก็ได้

f (pulse mode) = f (timer clock) / period register

f (clock mode) = f (timer clock) / (2xperiod register)

ตาราง 2-3 ผลลัพธ์ของการเขียนโดยระบุค่าของ GO และ $\overline{HLD}$

GO	$\overline{HLD}$	RESULT
0	0	ทุกๆ การทำงานของตัวจับเวลาจะถูกคงค่าไว้ และไม่มีการแสดงค่าตั้งใหม่
0	1	ตัวจับเวลา เกิดจากสภาวะก่อนการเขียน
1	0	ทุกๆ การทำงานของตัวจับเวลาจะถูกคงค่า รวมถึงการทำให้เป็น 0 ของตัวนับ และบิต GO จะไม่ถูกทำให้ว่างจนกว่าตัวจับเวลาจะไม่ถูกคงค่าไว้
1	1	ตัวจับเวลา ตั้งค่าใหม่ และเริ่มการทำงาน

2.1.1.4.1.2 การอินเทอร์รัพต์ตัวจับเวลา (Timer interrupts)

การอินเทอร์รัพต์ตัวจับเวลา จะถูกสร้างเมื่อใดก็ตามที่บิต TSTAT ของรีจิสเตอร์ควบคุมตัวจับเวลาเปลี่ยนจาก 0 เป็น 1 โดยที่ความถี่ของการอินเทอร์รัพต์ จะขึ้นอยู่กับว่าจะตั้งค่าให้ตัวจับเวลาอยู่ในโหมดใด

- ในโหมดสัญญาณพัลซ์. ความถี่อินเทอร์รัพต์ จะเป็นตามสมการ

  f (interrupt) = f (timer clock) / period register

  โดยที่ f (interrupt) = timer frequency

  f (timer clock) = interrupt frequency

- ในโหมดสัญญาณนาฬิกา. ความถี่อินเทอร์รัพต์ จะเป็นตามสมการ

  f (interrupt) = f (timer clock) / (2xperiod register),

  โดยที่ f (interrupt) = timer frequency

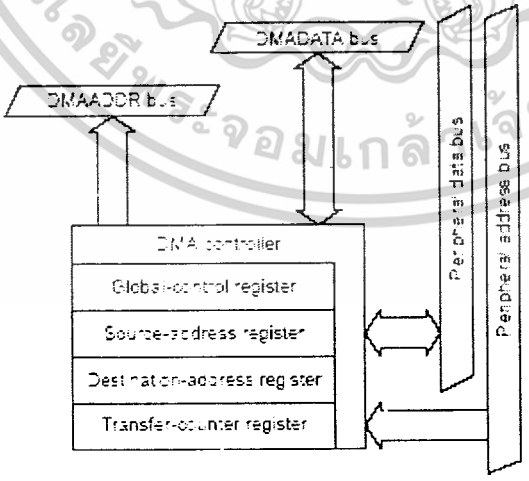
  f (timer clock) = interrupt frequency

2.1.1.4.3 พอร์ตอนุกรม (Serial Port)

พอร์ตอนุกรม 2 ทิศทางทั้ง 2 ตัวจะไม่ขึ้นต่อกัน โดยมีส่วนประกอบของรีจิสเตอร์ควบคุม ที่ควบคุมแต่ละพอร์ตเหมือนกัน แต่ละพอร์ตอนุกรมสามารถกำหนดให้ส่งผ่านข้อมูลต่อเวิร์ด 8 16 24 หรือ 32บิต ส่วนสัญญาณนาฬิกาสำหรับแต่ละพอร์ตอนุกรม เกิดได้จากทั้งภายในและภายนอก ขาของพอร์ตอนุกรมสามารถกำหนดให้เป็นตัวจับเวลาได้ด้วย ส่วนโหมดพิเศษ: แฮนด์เชค (handshake) จะอนุญาตให้ TMS320C3x ทำการติดต่อพอร์ตอนุกรมได้อย่างไม่มีการผิดพลาด

2.1.1.4.4 การเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access: DMA)

ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงบนชิป สามารถอ่านหรือเขียนลงตำแหน่งต่างๆ ใน แผนที่หน่วยความจำ โดยไม่เข้าไปแทรกแซงการทำงานของหน่วยประมวลผลกลาง ดังนั้น TMS320C3x สามารถอินเตอร์เฟสกับหน่วยความจำภายนอกและอุปกรณ์ภายนอก โดยผ่านทางหน่วยประมวลผลหลักซึ่งข้อมูลนั้นจะไม่ถูกลดทอนลง ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง จะบรรจุตัวสร้างตำแหน่ง (address generator) ของตัวเอง รีจิสเตอร์ต้นกำเนิดและจุดหมาย และตัวนับการส่งย้ายข้อมูล (transfer counter) บัสของตำแหน่งและบัสของข้อมูลของการเข้าถึงหน่วยความจำหน้าที่พิเศษ จะลดขนาดความขัดแย้งระหว่างหน่วยประมวลผลกลาง และตัวควบคุมการเข้าถึงหน่วยความจำให้น้อยลง การทำงานของการเข้าถึงหน่วยความจำประกอบด้วยบล็อก หรือเวิร์ดเดียว ส่งย้ายไปยังหรือไปจากหน่วยความจำ รูปที่ 2-11 แสดง ตัวควบคุมการเข้าถึงหน่วยความจำกับบัสที่สัมพันธ์กัน



รูปที่ 2-11 ตัวควบคุมการเข้าถึงหน่วยความจำ

2.1.2 โครงสร้างของ TMS320C3x DSP Starter Kit

2.1.2.1 โครงสร้างของ DSK

- TMS329C31 เป็น DSP อิงดรรชนี
- คำสั่งมีเวลาครบรอบ (cycle time) 40 นาโนวินาที 50 ล้านคำสั่งอิงดรรชนีต่อวินาที (MFLOPS) 25 ล้านคำสั่งต่อวินาที (MIPS)
- อินเตอร์เฟสพอร์ตเครื่องพิมพ์แบบขนาน (parallel printer port interface) ที่เป็นมาตรฐานหรือพัฒนาแล้ว เพื่อต่อกับเครื่องคอมพิวเตอร์แม่ (host PC^{TM}) และ TMS320C31 ติดต่อกับโปรแกรมของเครื่องคอมพิวเตอร์ได้
- รับข้อมูลอนาลอกโดยผ่านวงจรอินเตอร์เฟสแบบอนาลอก (AIC) TLC32040
  - อัตราตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (ADC) และตัวแปลงสัญญาณสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (DAC) สามารถเปลี่ยนแปลงพิสัยแบบพลวัตได้ 14 บิตใน 20000 ตัวอย่างต่อวินาที
  - มีฟิลเตอร์สร้างเอาท์พุตใหม่ และสามารถบายพาส (bypass) ได้ ตัวเก็บประจุแบบสวิตช์ (switched-capacitor) ป้องกันสัญญาณรบกวน เป็นอินพุตฟิลเตอร์
- ใช้คอนเนคเตอร์ปลั๊กแบบ RCA มาตรฐาน สำหรับอนาลอกอินพุตและเอาท์พุต
- คอนเนคเตอร์อีมูเลเตอร์ XDS510
- คอนเนคเตอร์เพิ่มเติมที่กำหนดขา TMS320C31 ทั้งหมดสำหรับใช้กับบอร์ด DSK ตัวลูก

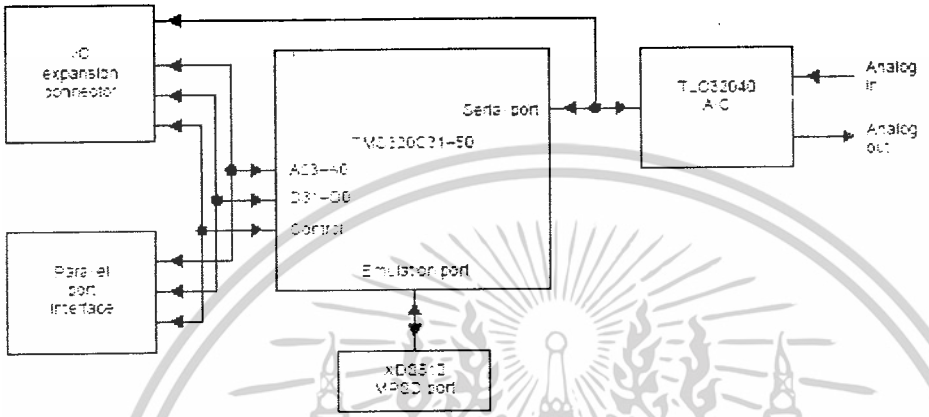
2.1.2.2 ภาพรวมของ DSK

รูปที่ 2-12 แสดงบล็อกไดอะแกรมของฮาร์ดแวร์ TMS320C3x DSK. รูปที่ 2-13 แสดงส่วนประกอบพื้นฐานของ DSK ดังนี้

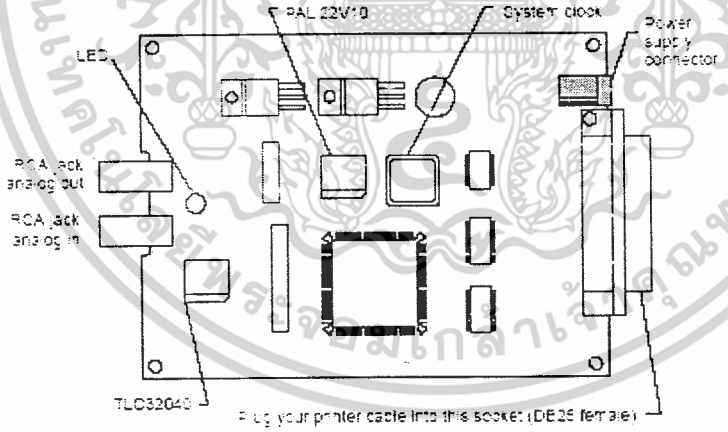
- DSP รุ่น TMS320C31
- วงจรอินเตอร์เฟสแบบอนาลอก TLC32040
- คอนเนคเตอร์เพิ่มเติม
- สัญญาณนาฬิกาของระบบ (System clock)
- อินเตอร์เฟสพอร์ตของเครื่องพิมพ์แบบขนาน
- ไฟแสดงผลสามสี (Tri-color LED)

เส้นทางสัญญาณทั้งหมดของ TMS320C3x จะไปยังคอนเนคเตอร์เพิ่มเติม โดยคอนเนคเตอร์เพิ่มเติมจะรวมทั้ง เฮดเดอร์ (header) 32 ขา 4 ตัว บล็อกจัมเพอร์ (jumper block) 11 ขา 1 ตัว และเฮดเดอร์ XDS510 12 ขา

วงจรอินเตอร์เฟสแบบอนาลอก TLC32040 อินเตอร์เฟสกับพอร์ตอนุกรมของ TMS320C3x โดยบล็อกจัมเพอร์จะอนุญาตให้เคลื่อนย้ายการติดต่อไปตามพอร์ตอนุกรมเพื่อไปยังการ์ด DSK ตัวลูกที่จะจ่ายไฟ ใช้ คอนเนคเตอร์อาร์ซีเอ 2 ตัวเป็นอนาลอกอินพุตและเอาท์พุตของบอร์ด



รูปที่ 2-12 บล็อกไดอะแกรมของ TMS320C3x



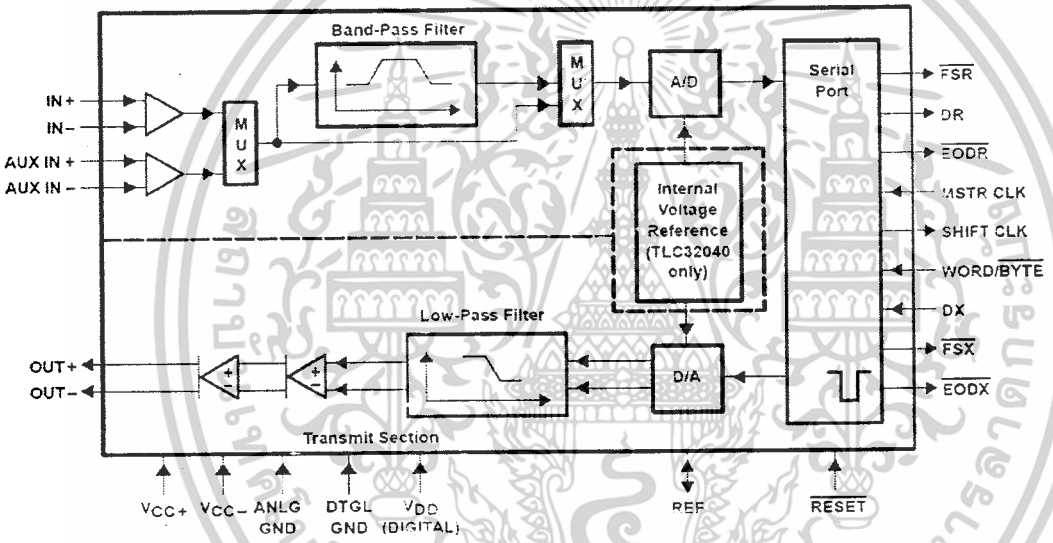
รูปที่ 2-13 แสดงส่วนประกอบพื้นฐานของ DSK

2.1.2.3 วงจรเชื่อมต่อสัญญาณอนาลอก

การใช้ดีเอสพีในการประยุกต์ใช้งานนั้นจะต้องมีส่วนประกอบที่สำคัญคือ ตัวเชื่อมที่ต่อสัญญาณอนาลอกเพื่อจะติดต่อสัญญาณอินพุตอนาลอกและสัญญาณเอาท์พุตอนาลอก ด้านอินพุตจะมีวงจรกรองและวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลที่มีช่วงความถี่ผ่านไม่เกินครึ่งหนึ่งของความถี่ชักตัวอย่าง (Sampling frequency) ซึ่งตามทฤษฎีชักค่าตัวอย่าง ความถี่ที่ใช้ชักตัวอย่าง ต้องมีค่าอย่างน้อยมากกว่า 2 เท่าของความถี่สูงสุดในสัญญาณ

$$Fs > 2f$$

วงจรเชื่อมต่อสัญญาณอนาลอก (Analog interface circuit ; AIC) ของ DSK คือ TLC32040 ซึ่งจะเชื่อมต่อกับพอร์ตอนุกรมของ TMS320C31 มีลักษณะดังรูปที่ 2-14



รูปที่ 2-14 แสดงบล็อกไดอะแกรมของวงจรเชื่อมต่อสัญญาณอนาลอก

ลักษณะและส่วนประกอบที่สำคัญดังนี้ คือ

- มี 2 อินพุต (IN+ IN-, AUX IN– AUX IN-)

  IN สามารถต่อใช้งานได้โดยใช้งานผ่านตัวคอนเนคเตอร์ RCA บนบอร์ด DSK ส่วนAUX IN สามารถต่อใช้งานผ่านขา 3 จาก JP3 โดยการเลือกใช้งานระหว่าง IN และ AUX IN สามารถเลือกใช้ได้จากบิตควบคุมที่จะกล่าวถึงต่อไป

- มี 1 เอาท์พุต (OUT+ , OUT -)

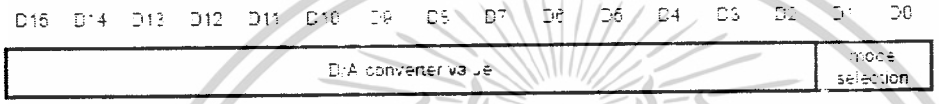
  OUT สามารถต่อใช้งานได้โดยใช้งานผ่านตัวคอนเนคเตอร์ RCA บนบอร์ดDSKเช่นเดียวกับส่วนอินพุต

- ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล และตัวแปลงสัญญาณดิจิตอลเป็นอนาลอก ขนาด 14 บิต
- วงจรกรองความถี่อินพุตและวงจรกรองความถี่เอาท์พุต (Switched-capacity antialiasing input filter & Reconstruction output filter)
- วงจรกรองความถี่อินพุตเป็นวงจรกรองความถี่ที่ทำให้ช่วงความถี่ผ่าน (bandpass filter) ที่จะให้ความถี่ผ่าน ในช่วง 200 HZ < f < 3.4 kHz สามารถเลือกใช้งานหรือไม่ก็ได้โดยผ่านการควบคุมด้วยบิตควบคุม นอกจากนี้วงจรกรองความถี่อินพุตยังสามารถปรับความถี่สำหรับความถี่ cut off หรือ bandwidth ที่ได้ออกแบบเอาไว้ โดยจะขึ้นอยู่กับความถี่ช่วงชักตัวอย่าง วงจรกรองความถี่เอาท์พุต เป็นวงจรกรองความถี่แบบความถี่ต่ำผ่าน (low-pass filter) โดยจะยอมให้มีความถี่ต่ำกว่า 3.4 kHz ผ่าน โดยไม่สามารถปรับค่าต่างๆ ได้
- ใช้เทคโนโลยี CMOS
- ความถี่ชักตัวอย่างสามารถโปรแกรมได้จนถึง 20 kHz
- ความถี่ Master clock เป็นตัวที่กำหนดสัญญาณทุกอย่างในวงจรเชื่อมต่อสัญญาณอนาลอก Master clock จะได้รับสัญญาณมาจากตัวจับเวลา 0 ของ TMS320C31
- Shift Clock เป็นสัญญาณที่ได้มาจากการหาร Master clock ด้วย 4 ใช้สำหรับเป็นสัญญาณนาฬิกาของการส่งข้อมูลแบบอนุกรมของ วงจรเชื่อมต่อสัญญาณอนาลอก และ TMS320C31
- ขา DR เป็นขาที่ใช้ในการส่งข้อมูลจากวงจรเชื่อมต่อสัญญาณอนาลอก ไปที่พอร์ตอนุกรมของ TMS320C31 โดยการส่งเป็นจังหวะเดียวกับความถี่ Shift clock
- ขา DX เป็นขาที่ใช้ในการรับข้อมูลจากพอร์ตอนุกรมของ TMS320C31 ไปที่วงจรเชื่อมต่อสัญญาณอนาลอก โดยการส่งเป็นจังหวะเดียวกับความถี่ Shift clock
- ขา FSR สัญญาณของ FSR จะเป็นสถานะต่ำระหว่างการส่งข้อมูล เมื่อ FSR เป็นสถานะต่ำ TMS320C31 จะเริ่มการรับบิตจากวงจรเชื่อมต่อสัญญาณอนาลอก ผ่าน DR ของวงจรเชื่อมต่อสัญญาณอนาลอก บิต MSB ของ DR จะถูกส่งการที่ FSR จะเป็นสถานะต่ำ FSR จะไม่เกิดขึ้นหลังจากการติดต่อแบบ Secondary
- ขา FSX สัญญาณของ FSX จะเป็นต่ำเมื่อพอร์ตอนุกรมของ TMS320C31 เริ่มการส่งบิตไปยังวงจรเชื่อมต่อสัญญาณอนาลอก ผ่าน DX ของวงจรเชื่อมต่อสัญญาณอนาลอก ในการติดตั้งทั้งหมด FSX จะมีสถานะต่ำระหว่างการส่ง
- รีเซต ฟังก์ชันรีเซตใช้สำหรับการกำหนดค่าเริ่มต้นให้กับ TA. TA'. TB. RA. RA '. RB และรีจิสเตอร์ควบคุม ฟังก์ชันรีเซตนี้จะกำหนดค่าเริ่มต้นของการติดต่อแบบอนุกรมระหว่างวงจรเชื่อมต่อสัญญาณอนาลอก และ DSP หลังจากมี negative-going pulse มาที่ขา RESET รีจิสเตอร์ของวงจรเชื่อมต่อสัญญาณอนาลอกจะถูกกำหนดค่าทำให้สร้างอัตราการแปลงข้อมูลเท่ากับ 8- kHz และมี Master clock เท่ากับ 5.184 MHz รีเซตฟังก์ชั่นรีเซตใช้สำหรับกำหนดค่าเริ่มต้นให้กับ TA. TA'.

TB. RA, RA `. RB และรีจิสเตอร์ควบคุม ฟังก์ชั่นรีเซตนี้จะกำหนดค่าเริ่มต้นของการติดต่อต่อแบบอนุกรมระหว่าง วงจรเชื่อมต่อสัญญาณอนาลอก และ DSP หลังจากมี negative-going pulse มาที่ขา RESET รีจิสเตอร์ของวงจรเชื่อมต่อสัญญาณอนาลอก จะถูกกำหนดค่าทำให้สร้างอัตราการแปลงข้อมูลเท่ากับ 8 kHz และมีmaster clock เท่ากับ 5.184 MHz

2.1.2.4 การใช้งานและควบคุมวงจรเชื่อมต่อสัญญาณอนาลอก

การส่งข้อมูลเกิดขึ้นผ่าน DR และ DX ซึ่งเป็นรีจิสเตอร์อนุกรมของ วงจรเชื่อมต่อสัญญาณอนาลอก ซึ่งการส่งข้อมูลแต่ละครั้งจะมีรูปแบบการส่งดังรูปที่ 2-15 คือ



รูปที่ 2-15 รูปแบบของบิตในการส่งข้อมูลของ DR หรือ DX

LSBs	Mode
00	Tx Counter A = TA, Rx Counter A = RA Tx Counter B = TB, Rx Counter B = RB
01	Tx Counter A = TA + TA, Rx Counter A = RA + RA Tx Counter B = TB, Rx Counter B = RB
10	Tx Counter A = TA + TA, Rx Counter A = RA + RA Tx Counter B = TB, Rx Counter B = RB
11	Tx Counter A = TA, Rx Counter A = RA Tx Counter B = TB, Rx Counter B = RB

รูปที่ 2-16 แสดงบิตควบคุม D0. D1 ของการส่งข้อมูลในแต่ละแบบ

จากรูปที่ 2- 16 บิต DO และ D1 ใช้สำหรับควบคุมฟังก์ชันการทำงานของการสื่อสาร โดยถ้า D1 , D0 มีค่าเท่ากับ 0 (โหมดที่ 1) จะเป็นการส่งแบบปกติ แต่ถ้ามีค่าเท่า1 ทั้งสองตัว (โหมด 4) ทำให้เกิดการร้องขอการส่งแบบ secondary ซึ่งทำให้เกิดการส่งแบบ secondary ตามมาหลังจาก shift clock 4 ลูก ซึ่งการส่งแบบ secondary เป็นการส่งคำสั่งควบคุมเพื่อกำหนดค่าเริ่มต้น และควบคุมวงจรเชื่อมต่อสัญญาณอนาลอก โดยจะส่งหนึ่งครั้ง และจะกลับไปเป็นการส่งแบบปกติตามเดิม ส่งในกรณีที่มีตัวใดตัวหนึ่งเป็น 0 และอีกตัวเป็น 1 จะทำให้เกิดการเปลี่ยนความถี่การชักค่าตัวอย่างในคาบถัดไป ส่วนการส่งในโหมด 2 และ 3 ใช้เพื่อเป็นการเพิ่ม หรือลดความถี่ชักค่าตัวอย่างโดยการลด หรือเพิ่มคาบการชักตัวอย่าง ซึ่งจะใช้อย่างมากในการประยุกต์ใช้กับโมเด็ม

การส่งแบบ Secondary จะส่งรีจิสเตอร์ A ซึ่งประกอบด้วย TA. RA รีจิสเตอร์ B ซึ่งประกอบด้วย TB. RB รีจิสเตอร์ A` ซึ่งประกอบด้วย TA`. RA` และบิตควบคุม ซึ่งจะควบคุมการทำงานของวงจรเชื่อต่อสัญญาณอนาลอก ค่า TA. TB สามารถนำมาใช้ดังนี้

- ออกแบบค่าความถี่ของวงจรกรองสัญญาณอินพุต (Switched-capacitor filter frequency) ได้โดยคิดจาก

SCF Clock Frequency = Master Clock Frequency/ (2*TA)

เพื่อให้วงจรกรองความถี่อินพุทมีการทำงานตามคุณลักษณะของฟังก์ชั่นถ่ายโอน จะต้องมีความถี่ Switched-capacitor clock เท่ากับ 288 kHz ซึ่งจะทำให้มี bandwidth หรือความถี่ cut-off เท่ากับ 3600 Hz

- ออกแบบค่าความถี่การชักค่าตัวอย่าง(Sampling Frequency)

Sampling Frequency = Master Clock Frequency/ (2*TA*TB)

ส่วนค่า Shift Clock หาได้จาก

Shift Clock = Master Clock/4

สำหรับค่า TA และ TB จะใช้ในส่วนของการแปลงสัญญาณอนาลอกเป็นดิจิตอล ส่วน RA และRB จะใช้ในทำนองเดียวกันในส่วนของการแปลงสัญญาณดิจิตอลเป็นอนาลอกรูปแบบการส่งของ Secondary Communication คือ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	TA register value (unsigned)					X	X	RA register value (unsigned)					0	0
X	TA register value (signed 2s complement)						X	RA register value (signed 2s complement)						0	1
X	TB register value (unsigned)						X	RB register value (unsigned)						1	0
X	X	X	X	X	X	X	X	Control register						1	1

ตารางที่ 2-6 แสดงรูปแบบการส่งแบบ Secondary

บิตควบคุมมีดังนี้ คือ

D2 = 0/1 ใช้/ไม่ใช้ วงจรกรองสัญญาณอินพุต

D3 = 0/1 ใช้/ไม่ใช้ ฟังก์ชันวนลูป

D4 = 0/1 ใช้/ไม่ใช้ AUX IN + และ AUX IN -

D5 = 0/1 ซิงโครนัส / อะซิงโครนัส การส่ง และ การรับ

D6 = 0/1 บิตควบคุมอัตราการขยาย ซึ่งจะมีค่าดังรูปที่ 2-16

INPUT CONFIGGURATIONS	CONTROL REGISTER BITS d6	CONTROL REGISTER BITS d7	ANALOG INPUT	A/D CONVERSION RESULT
ANALOG INPUT	1	1	± 3 V	Half scale
	0	0	± 3 V	Half scale
	1	0	± 3 V	Full scale
	0	1	± 1.5V	Full scale

รูปที่ 2-17 แสดงบิตควบคุมการขยาย

2.2 หลักการและทฤษฎีที่ใช้ในการออกแบบการหาค่าพารามิเตอร์ของตัวควบคุมแบบ Adaptive PID

2.2.1 ความรู้เบื้องต้นเกี่ยวกับระบบควบคุมป้อนกลับ

หลักการของระบบควบคุมป้อนกลับนั้น เกิดมานานกว่า 2000 ปีแล้ว แต่ไม่ได้ถูกนำมาใช้ในอุตสาหกรรม จนกระทั่งเมื่อประมาณ 200 ปีที่แล้ว เจมส์ วัตต์ ได้นำเอาหลักการดังกล่าวมาทำการสร้างเครื่องควบคุมความเร็วให้กับเครื่องจักรไอน้ำ จากนั้นอุตสาหกรรมมากมาย ได้หันมาให้ความสนใจและนำมาใช้กันอย่างแพร่หลายจนถึงปัจจุบัน ระบบควบคุมแบบป้อนกลับนี้อาจะเรียกอีกอย่างหนึ่งว่า "ระบบควบคุมวงปิด หรือ ระบบควบคุมแบบอัตโนมัติ"

ระบบควบคุมทางอุตสาหกรรมแบบป้อนกลับโดยทั่วไปประกอบด้วยส่วนประกอบสำคัญ 4 ส่วน ดังนี้

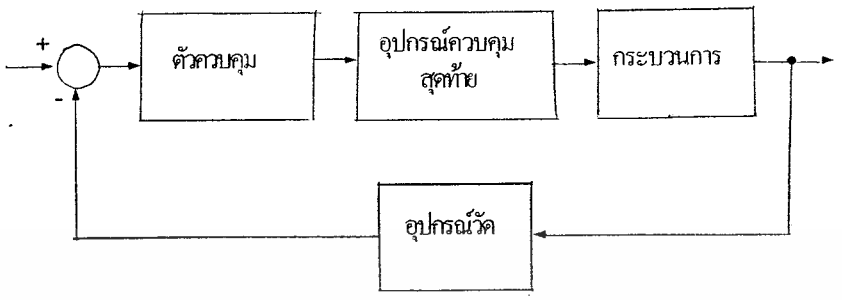
2.2.1.1 ตัวควบคุม (controller) เป็นเครื่องมือหรืออุปกรณ์ที่ใช้ในการสร้างสัญญาณควบคุม เพื่อทำหน้าที่ควบคุมให้กระบวนการที่ต้องการควบคุมมี เอาท์พุตหรือผลตอบสนองเป็นไปตามความต้องการ โดยทั่วไปตัวควบคุมมีหลายชนิด เช่น ตัวควบคุมแบบ เปิด-ปิด ตัวควบคุมแบบพี ตัวควบคุมแบบไอ ตัวควบคุมแบบดี หรือ การใช้ตัวควบคุมหลายๆชนิดร่วมกัน เช่น ตัวควบคุมแบบพีไอ ตัวควบคุมแบบพีดี หรือ ตัวควบคุมแบบพีไอดี เป็นต้น

2.2.1.2 อุปกรณ์ควบคุมสุดท้าย (Final Control Element) คือ อุปกรณ์ที่ทำหน้าที่ปรับสภาวะของกระบวนการ ด้วยการเปลี่ยนแปลงค่าตัวแปรปรับกระบวนการ ตามคำสั่งหรือสัญญาณที่ได้รับจากตัวควบคุม อุปกรณ์ควบคุมสุดท้ายนั้นมีอยู่หลายชนิดด้วยกัน เช่น วาล์วควบคุม อินเวิตเตอร์ หรือแอคทูเอเตอร์ ทั่วไป ๆ

2.2.1.3 กระบวนการ (Plant or Process) หมายถึง ระบบหรือกระบวนการทางฟิสิกส์ที่ต้องการควบคุมให้มีสถานะเป็นไปตามต้องการ เช่น การควบคุมความเร็วของ มอเตอร์ การควบคุมตำแหน่งของ มอเตอร์ การควบคุมระดับของเหลว การควบคุมอุณหภูมิ เป็นต้น

2.2.1.4 อุปกรณ์วัด (Measuring Instrements) หมายถึง อุปกรณ์ที่ใช้วัดค่าทางฟิสิกส์ของเอาท์พุตของระบบ แล้วเปลี่ยนไปเป็นปริมาณที่เราต้องการ เพื่อใช้เป็นป้อนสัญญาณกลับให้กับระบบ อุปกรณ์วัดทั่วไปมีหลายชนิด เช่น ทาโคมิเตอร์ใช้วัดค่าความเร็วรอบให้ออกมาอยู่ในรูปของสัญญาณไฟฟ้า เป็นต้น

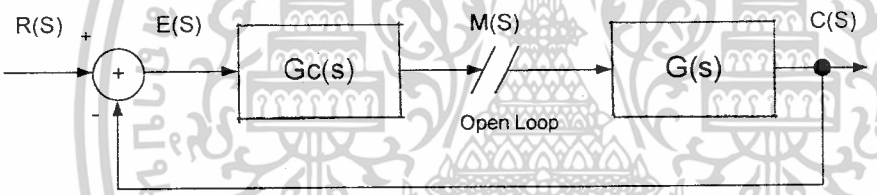
โดยระบบควบคุมป้อนกลับทั่วๆ ไป จะมี บล็อกไดอะแกรม ดังรูป 2-18



รูปที่ 2-18 ระบบควบคุมป้อนกลับทั่วๆไป

2.2.2 วิธีการควบคุมแบบต่าง

2.2.2.1 ตัวควบคุมแบบ PID (PID Controller)



รูปที่ 2-19 บล็อกไดอะแกรมแสดงค่าความคลาดเคลื่อน

จากรูป ที่ 2-19 แสดงถึงค่าความคลาดเคลื่อน ซึ่งเป็นผลต่างของค่าอินพุท R และค่าเอาท์พุท C ค่าความคลาดเคลื่อนนี้จะถูกส่งไปที่ตัวควบคุมแบบพีไอดี และตัวควบคุมจะคำนวณสัญญาณความคลาดเคลื่อน และส่งสัญญาณ M เพื่อไปควบคุมระบบ โดยM จะมีค่าดังนี้

$$M(t) = K_p\, e(t) + K_i \int e(t)dt + K_d\, (de(t)/dt) \qquad \text{----(1)}$$

หาฟังก์ชันถ่ายโอนของตัวควบคุมแบบพีไอดี จะได้

$$G_c(s) = M(s)/E(s) = Kp + K_i/s + K_d s = (K_d s^2 + K_p s + K_i)/s \qquad \text{--- (2)}$$

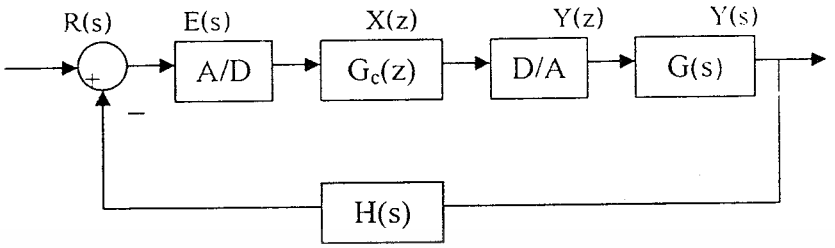
เมื่อ K_p คือ ค่าขยาย proportional

K_i คือ ค่าขยาย integral

K_d คือ ค่าขยาย derivative

2.2.3.2 การควบคุมแบบดิจิตอล (Digital Controller)

การควบคุมแบบดิจิตอลมีบล็อกไดอะแกรมดังรูป



รูปที่ 2-20 บล็อกแสดงการควบคุมแบบดิจิตอล

จากรูป R (s) คือ อินพุทของระบบ หรือ set point ของระบบ

C(s) คือ เอาท์พุทของระบบ

$G_c(z)$ คือ ฟังก์ชั่นถ่ายโอนของตัวควบคุมแบบดิจิตอล

G(s) คือ ฟังก์ชั่นถ่ายโอนของกระบวนการ

H(s) คือ ฟังก์ชั่นถ่ายโอนของอุปกรณ์วัด

ตัวควบคุมแบบดิจิตอลพีไอดี จะมีสมการดังนี้

$$G_c(z) = K_p + K_i \frac{Tz+1}{2z-1} + K_d \frac{z-1}{Tz}$$

$$= \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} \qquad \text{---(3)}$$

เมื่อ $a_0 = K_p + K_i \frac{T}{2} + \frac{K_d}{T}$

$a_1 = -K_p + K_i \frac{T}{2} - \frac{2K_d}{T}$

$a_2 = \frac{K_d}{T}$

โดยที่ K_p K_i K_d คือ ค่าพารามิเตอร์proportional integral และ derivative ของตัวควบคุม และ T คือ sampling period และเพื่อความสะดวกในการเขียนโปรแกรม เราจึงกำหนดให้

$$G_c(z) = G_1(z)\frac{M(z)}{M(z)} = \frac{Y(z)}{X(z)}$$

ที่ซึ่ง X (z), Y (z) เป็นอินพุทและเอาท์พุทของตัวควบคุมบน z-domain ดังนั้นได้

$$Y(z) = \left(a_0 + a_1 z^{-1} + a_2 z^{-2}\right)M(z) \qquad \text{--- (4)}$$

และ

$$X(z) = \left(1 - z^{-1}\right)M(z) \qquad \text{---(5)}$$

ใช้การแปลงแซดแบบผกผันจะได้

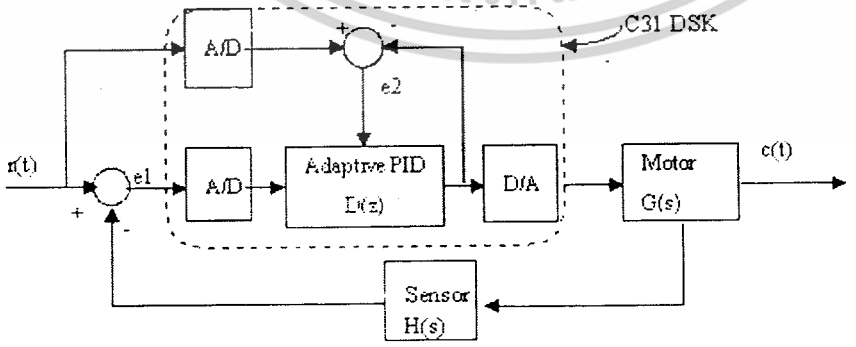
$$y(k) = a_0 m(k) + a_1 m(k-1) + a2m(k-2) \qquad \text{--- (6)}$$

และ

$$m(k) = x(k) + m(k-1) \qquad \text{--- (7)}$$

จากสมการจะเห็นได้ว่า เมื่อได้เอาท์พุทใหม่ เอาท์พุทจะถูกส่งกลับไปคำนวณค่าความคลาดเคลื่อนอีกครั้ง และตัวควบคุมก็จะหาสัญญาณควบคุมใหม่เป็นเช่นนี้ตลอดไป

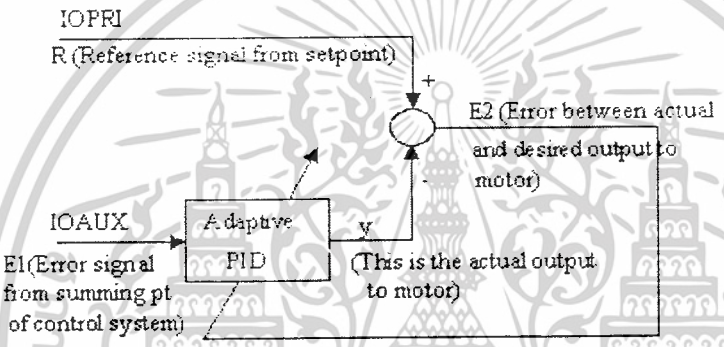
2.2.3.3การควบคุมแบบ Adaptive PID (Adaptive PID Controller)



รูปที่ 2-21 แสดงการควบคุมแบบ Adaptive PID

จากแผนภาพของระบบควบคุมแบบ Adaptive จะประกอบด้วย 2 ส่วนหรือสองลูป คือ ส่วนแรก เป็นส่วนกำหนดค่าเริ่มต้นและค่าที่ปรับปรุงของค่าพารามิเตอร์พีไอดี ส่วนส่วนที่สอง เป็นส่วนที่ปรับปรุงค่าพารามิเตอร์ของตัวควบคุม ซึ่งกระบวนการนี้จะดำเนินต่อไปจนกระทั่งสัญญาณerror e2 เข้าใกล้ค่า ศูนย์

ในรายละเอียดของตัวตัวส่วนควบคุมซึ่งมีบอร์ดดีเอสเคเป็นศูนย์กลางนั้น จะมีอินพุทสองทางที่เข้ามา ทางแรกหรือPrimary Input (IOPRI) และอีกทางก็คือ Auxiliary Input (IOAUX) ซึ่งลายละเอียดที่ของส่วนควบคุมจากรูปที่ 2-26 จะถูกแสดงในรูปที่ 2-27 โดยที่ E1 และ y เป็นอินพุท และเอาท์พุท ของตัวควบคุมพีไอดีดังรูป



รูปที่ 2-22 แสดงรายละเอียดของตัวควบคุมแบบ Adaptive PID

ดังนั้นจะได้ E2 เท่ากับ

$$E_2(z) = R(z) - Y(z) = R(z) - \frac{\left[a_0 + a_1 z^{-1} + a_2 z^{-2}\right]}{1 - z^{-1}} E_1(z) \quad \text{--- (8)}$$

เมื่อพิจารณาฟังก์ชั่นวัตถุประสงค์ในรูปกำลังสอง(Quadratic objective function) ของสัญญาณความผิดพลาด $E_2(z)$ จะได้

$$J(a_0, a_1, a_2) = \frac{(E_2)^2}{2}$$

$$= \frac{\left\{\left[R(z) - \left(\frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}}\right) E_1(z)\right]^2\right\}}{2} \quad \text{--- (9)}$$

หากวัตถุประสงค์ในการควบคุม เพื่อปรับค่าหาพารามิเตอร์ของตัวควบคุมซึ่งทำให้ฟังก์ชั่นวัตถุประสงค์นี้มีค่าน้อยที่สุด(Minimizing) พิจารณาอนุพันธ์ย่อยอันดับหนึ่งของฟังก์ชั่นวัตถุประสงค์เทียบกับพารามิเตอร์ของตัวควบคุม จะได้

$$\frac{\partial J}{\partial a_0} = -2R\left(\frac{1}{1-z^{-1}}\right)E_1 + 2\left(\frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1-z^{-1}}\right)\left(\frac{1}{1-z^{-1}}\right)E_1^2$$

$$\frac{\partial J}{\partial a_1} = -2R\left(\frac{z^{-1}}{1-z^{-1}}\right)E_1 + 2\left(\frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1-z^{-1}}\right)\left(\frac{z^{-1}}{1-z^{-1}}\right)E_1^2$$

$$\frac{\partial J}{\partial a_2} = -2R\left(\frac{z^{-2}}{1-z^{-1}}\right)E_1 + 2\left(\frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1-z^{-1}}\right)\left(\frac{z^{-1}}{1-z^{-1}}\right)E_1^2 \quad \text{--- (10)}$$

เมื่อใช้การประมาณแบบอันดับหนึ่ง จะได้สมการ Negative gradient อย่างง่ายที่อยู่ในรูปผลคูณของสัญญาณอ้างอิง R(s) และ สัญญาณค่าความผิดพลาด $E_1(s)$ แต่จากผลการทดลองที่อ้างอิงมาแสดงให้เห็นว่าถ้าเราใช้ ทั้ง E_1และE_2 จะทำให้การconvergence ดีกว่า ดังนั้น วิธีการที่ปรับปรุง Gradient จึงถูกใช้เป็นไปตามนี้เพื่อใช้ในการปรับเปลี่ยนค่าพีไอดีพารามิเตอร์ ซึ่งข้อสรุปอันนี้ก็ตรงกับ general adaptive mechanism โดยอ้างอิงจากหนังสือ Servo Motor and Motion Control Using Digital Signal Processors ของ Y.Dote ดังนี้

new parameter = old parameter + (bounded step size) *(function of input) * (function of error)

หรือเขียนในรูป

$$a_n(k+1) = a_n + \beta e_2(k) e_1(k-n) \quad n = 0, 1, 2 \quad \text{--- (11)}$$

บทที่ 3

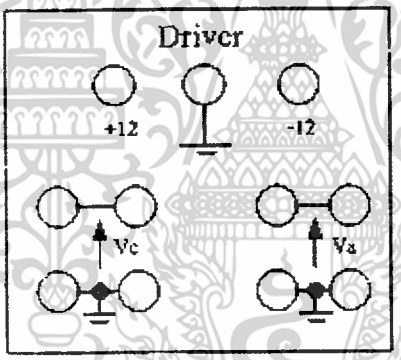
การออกแบบการทดลอง

ในการทดลองนี้จะเป็นการประยุกต์ใช้งาน บอร์ด ดีเอสเค ไปควบคุมกระบวนการ โดยกระบวนการในการทดลองนี้ เป็นการควบคุมความเร็วของมอเตอร์กระแสตรง โดยให้มีการควบคุมแบบหาค่าพารามิเตอร์ของการควบคุมแบบAdaptive PID

3.1 กระบวนการที่จะทำการควบคุม

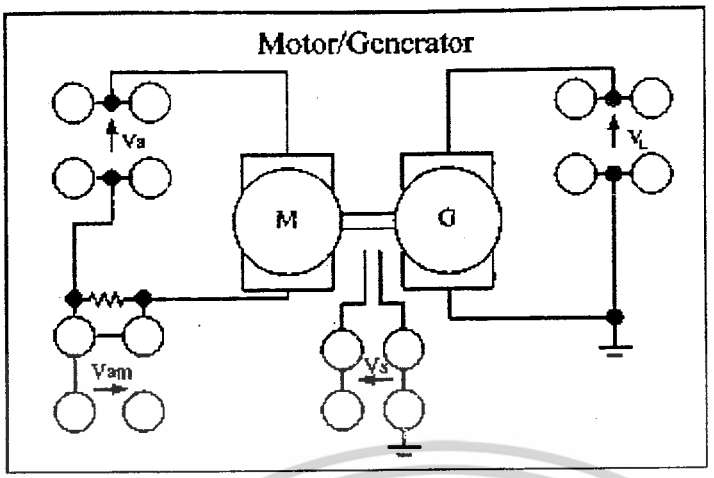
กระบวนการในการทดลองนี้ จะเป็นการควบคุมความเร็วของ มอเตอร์กระแสตรง โดยมีส่วนประกอบของกระบวนการดังนี้

3.1.1 ชุดวงจรขับมอเตอร์ เป็นวงจรที่ใช้ไฟเลี้ยง +12 กับ -12โวลต์ และใช้ Vc เป็นส่วนที่รับคำสั่งจาก เอาท์พุตของวงจรขยายสัญญาณที่ออกจากตัวควบคุม



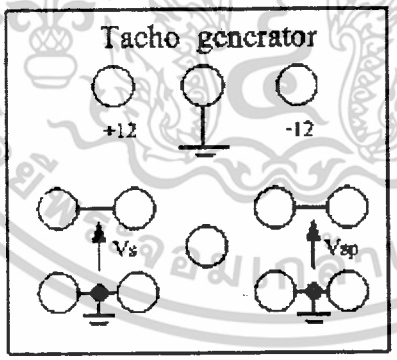
รูปที่3-1 ชุดวงจรขับมอเตอร์

3.1.2 ชุดมอเตอร์และเจนเนอเรเตอร์ ประกอบด้วยมอเตอร์มีโหลดเป็นเจนเนอเรเตอร์ โดยเมื่อมอเตอร์หมุนก็จะทำให้เกิดการสร้างแรงดันไฟฟ้าเกิดขึ้น และนำแรงดันไฟฟ้านี้ไปต่อกับชุดของโหลด สำหรับดูผลการตอบสนอง เมื่อมีการเปลี่ยนแปลงที่โหลด



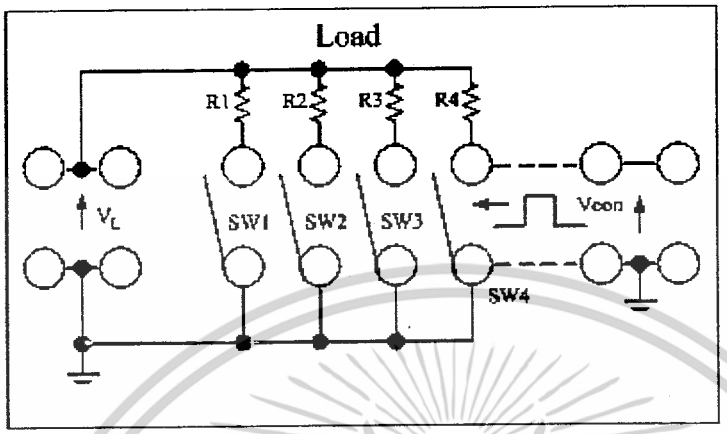
รูปที่3-2 ชุดมอเตอร์และเจนเนอเรเตอร์

3.1.3 ชุดทาโคเจนเนอเรเตอร์ ชุดทาโคเจนเนอเรเตอร์นี้จะมี สัญญาณอินพุตเป็นสัญญาณ Vs ซึ่งเป็นสัญญาณซายน์ (sine) ที่ถูกสร้างออกมาจากเจนเนอเรเตอร์ที่ต่ออยู่กับมอเตอร์ โดยสัญญาณซายน์นี้จะแปรผันโดยตรงกับความเร็วรอบของมอเตอร์ที่หมุน และชุดแทคโคเจนเนอเรเตอร์นี้จะมีสัญญาณเอาท์พุตเป็นสัญญาณดีซี โดยสัญญาณดีซีจะแปรผันโดยตรงกับความถี่ของสัญญาณซายน์อินพุตที่เข้ามา กล่าวคือถ้าสัญญาณ Vs มีความถี่สูง ค่าVsp ก็จะมีค่าสูงตามไปด้วย และเราจะนำสัญญาณ Vsp นี้เป็นสัญญาณที่ใช้ในการป้อนกลับ



รูปที่ 3-3 ชุดทาโคเจนเนอเรเตอร์

3.1.4 ชุดโหลด ประกอบด้วยชุดตัวต้านทานซึ่งเมื่อผลักสวิตซ์จะทำให้ค่าระดับโวลต์ของ V_L ลดลงทำให้มอเตอร์หมุนช้าลงด้วย



รูปที่ 3-4 ชุดโหลด

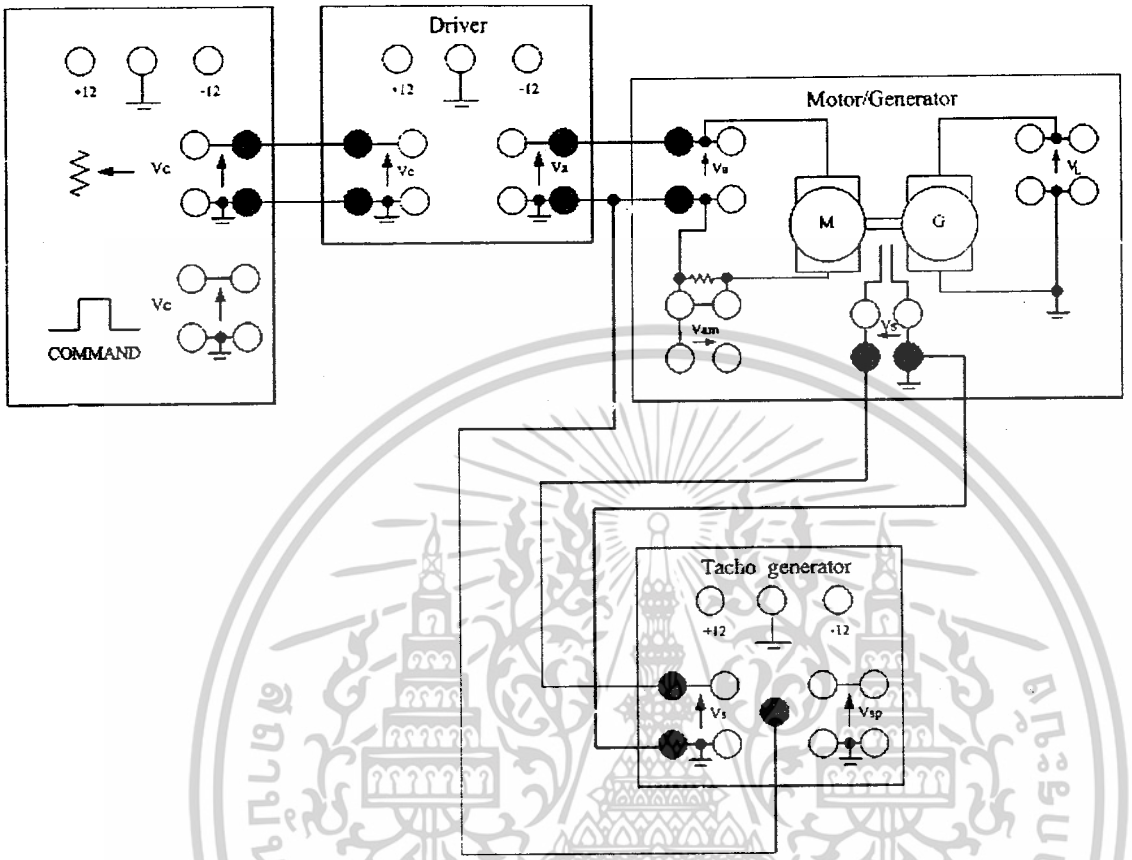
3.2 วงจรที่ใช้ในการทดลอง

สัญญาณเอาท์พุตที่ได้จากบอร์ด มีค่าอยู่ระหว่าง – 3 โวลต์ ถึง 3 โวลต์ แต่เนื่องจากเราต้องการนำสัญญาณที่ได้มาควบคุมกระบวนการ เราจึงจำเป็นต้องขยายย่านการทำงานของสัญญาณเอาท์พุต

จากการทดลองดังรูป 3-5ใส่ค่าอินพุต 0- 12 โวลต์ ให้กับกระบวนการ(V_c) จะได้ เอาท์พุท (V_{sp}) ดังนี้

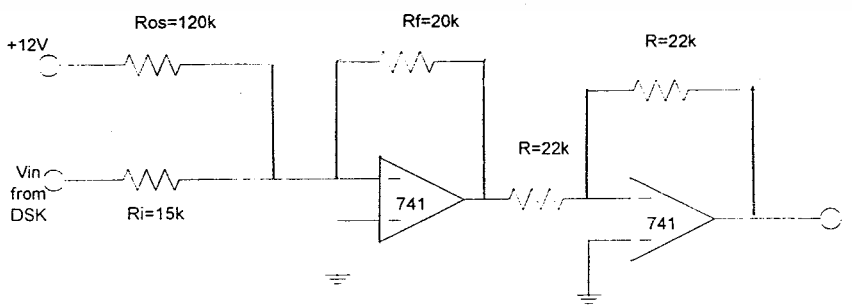
I/P (V_c)	O/P (V_{sp})
0	0
1	0
2	0
3	1.1
4	1.8
5	2.45
6	3
7	3.5
8	3.6
9	3.6
10	3.6
11	3.6
12	3.6

ตารางที่ 3-1 ผลของความสัมพันธ์ระหว่างอินพุต V_c และเอาท์พุต V_{sp}

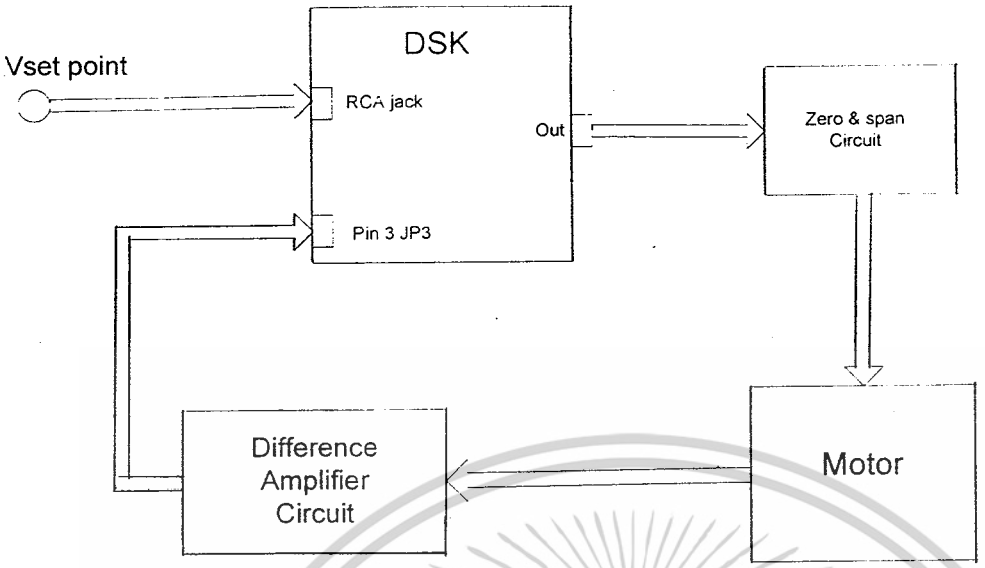


รูปที่ 3-5 วงจรใช้ในการทดลอง หาย่านการทำงานของกระบวนการ

จากผลการทดลองเราต้องการ O/P (V_{sp}) ในช่วง 0- 3 โวลต์ (ควบคุมการหมุนทางบวกเท่านั้น) ดังนั้นเราจึงต้องการ I/P (V_c) ในช่วง 2-6 โวลต์ เพราะฉะนั้นจึงออกแบบวงจร Zero & Span ดังนี้ ดังนี้

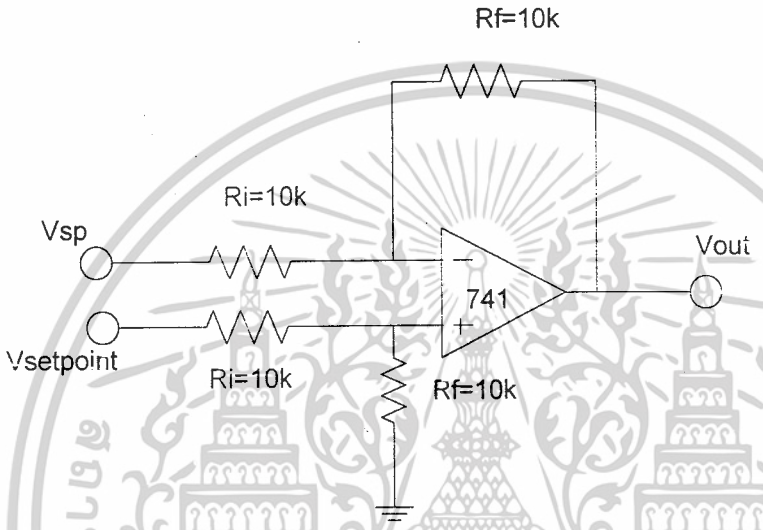


รูปที่ 3-6 วงจรขยายสัญญาณควบคุมแบบ (Zero & Span)



รูปที่ 3-8 การต่อวงจรที่ใช้ในการทดลอง

หลังจากสัญญาณนี้ผ่านชุดขับมอเตอร์(Driver) ชุดมอเตอร์และเจนเนอเรเตอร์ (moter and generator) ชุดทาโคเจนเนอเรเตอร์ (tacho generator) ก็จะได้ V_{sp} ซึ่งเราจะนำสัญญาณที่ได้นี้ไปลบกับสัญญาณset point ที่ตั้งไว้ได้เป็นสัญญาณ e_1 เพื่อส่งไปเข้าบอร์ดที่ขา 3 ของ JP3 วงจรที่ใช้ในที่นี้ก็คือวงจร difference amplifierดังรูป



รูปที่ 3-7 วงจรขยายสัญญาณผลต่าง (difference amplifier)

ในวงจรขยายวงจรขยายสัญญาณควบคุมแบบ(Zero & Span) และวงจรขยายสัญญาณผลต่าง (difference amplifier) ได้ใช้ออปแอมป์เบอร์ 741 ขนาด 8 ขา เนื่องจากเป็นเบอร์ที่มีการใช้งานทั่วไป

3.3 การประยุกต์ใช้บอร์ด ดีเอสเค ตามหลักการและทฤษฎี

ในการทดลองนี้จะนำบอร์ด DSK ไปเป็นตัวควบคุมกระบวนการ โดยให้มีการควบคุมแบบ PID และแบบ Adaptive PID ตามหลักการและทฤษฎีที่ได้กล่าวไว้แล้วในบทที่2 โดยมีรูปวงจรดังรูป

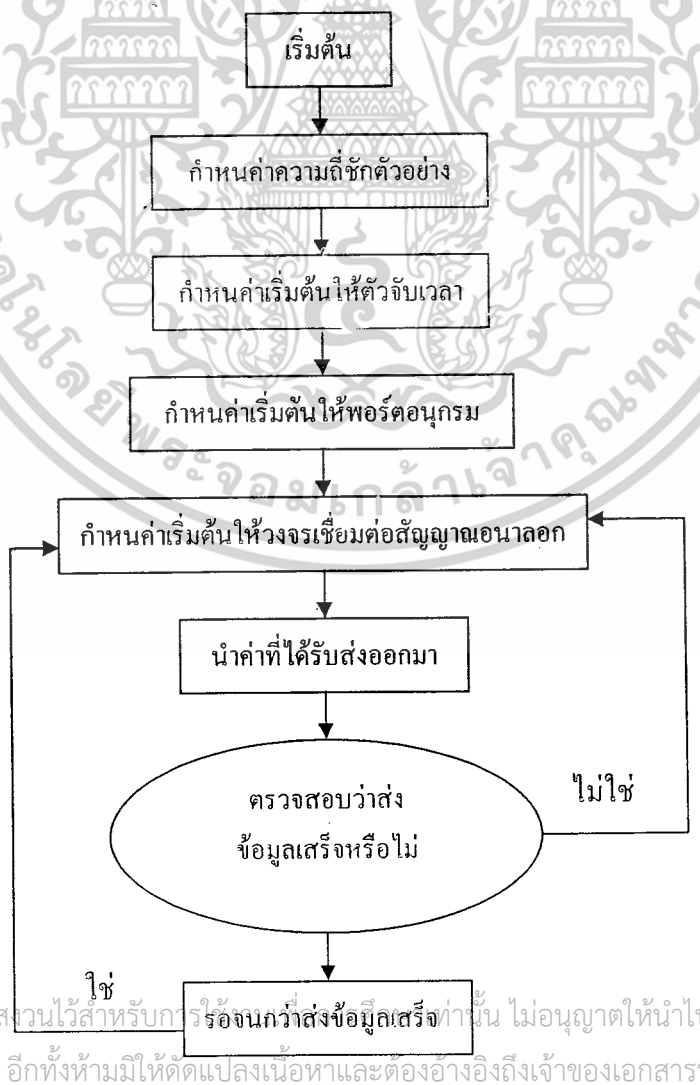
บทที่ 4
ผลการทดลอง

บทนี้ประกอบไปด้วยผลการทดลองการใช้วงจรเชื่อมต่ออนาลอก ผลการทดลองการควบคุมแบบพีไอดี และผลการทดลองการควบคุมแบบอะแดปทีฟ พีไอดี ดังนี้

4.1 ผลการทดลองการใช้วงจรเชื่อมต่อสัญญาณอนาลอก

4.1.1 การทดลองรับค่าเข้าบอร์ดดีเอสเค และส่งค่าเดิมออกมาจากบอร์ดดีเอสเค

โปรแกรมนี้จะรับค่าอินพุทที่คอนเนคเตอร์RCAอินพุท และส่งค่าเดิมออกไปที่คอนเนคเตอร์RCA เอาท์พุท ซึ่งสามารถที่จะใส่อินพุทเป็นสัญญาณsine wave ,square wave หรือ ramp wave ก็ได้ แต่ควรมีค่าความถี่อยู่ระหว่าง 500Hz จนถึง 3 kHz ซึ่งเอาท์พุทออกมาจะมีค่าตามอินพุท แต่จะมีความล่าช้าของสัญญาณอยู่บ้าง มีผังการทำงานดังต่อไปนี้

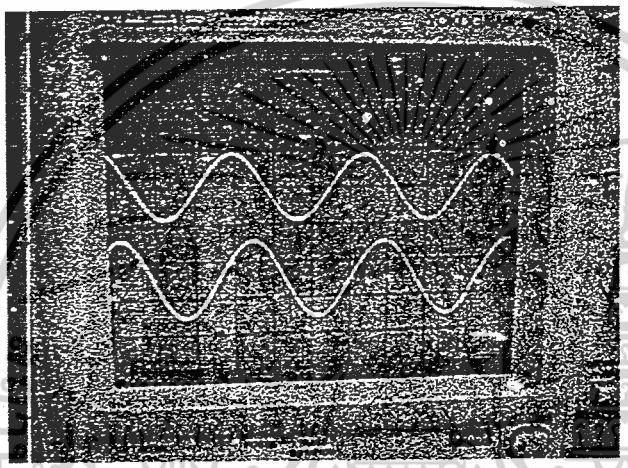


รูปที่ 4- 1 ผังการทำงานของโปรแกรมรับค่า และส่งค่าเดิมออก

4.1.2 การทดลองผลของความถี่สัญญาณอินพุทที่มีผลต่อการล้าหลังของสัญญาณ

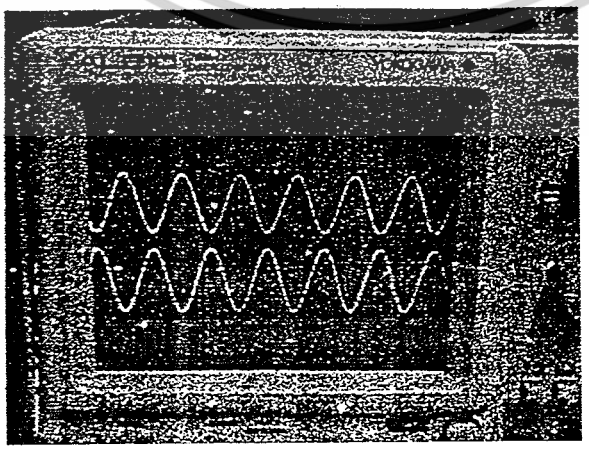
การทดลองจะทำการรับสัญญาณ sine wave ที่มีความถี่ต่างกันผ่านท่างวงจรเชื่อมต่อสัญญาณอนาลอก แล้วส่งออก จากนั้นหาการล้าหลังระหว่างสัญญาณที่เป็นอินพุทกับเอาท์พุท ซึ่งผลการทดลองได้ดังรูป

4.1.2.1 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 300 Hz ใช้ความถี่การชักตัวอย่างเป็น 8 Hz ขนาด เวลาต่อช่อง เท่ากับ 1 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



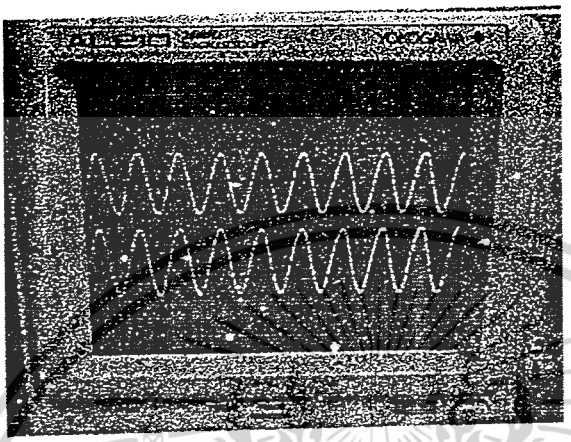
รูปที่ 4-2 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 300 Hz

4.1.2.2 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 600 Hz ใช้ความถี่การชักตัวอย่างเป็น 8 Hz ขนาด เวลาต่อช่อง เท่ากับ 1 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



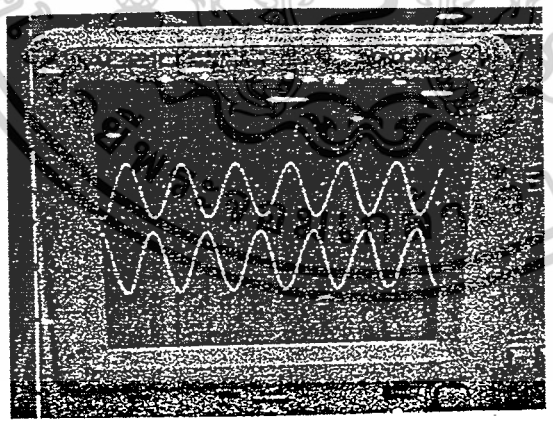
รูปที่ 4-3 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 600 Hz

4.1.2.3 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 900 Hz ใช้คามถี่การชักตัวอย่างเป็น 8 Hz ขนาด เวลาต่อช่อง เท่ากับ 1 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



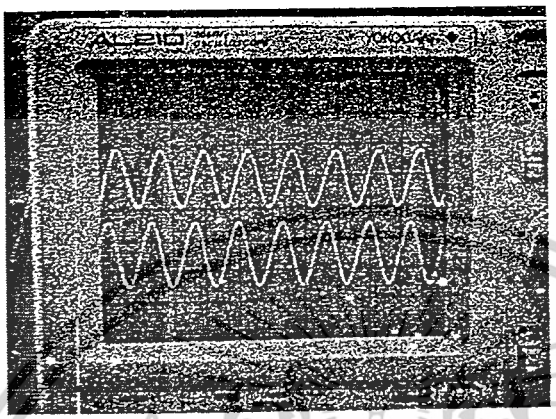
รูปที่ 4-4 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 900 Hz

4.1.2.4 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1200 Hz ใช้คามถี่การชักตัวอย่างเป็น 8 Hz ขนาด เวลาต่อช่อง เท่ากับ 0.5 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



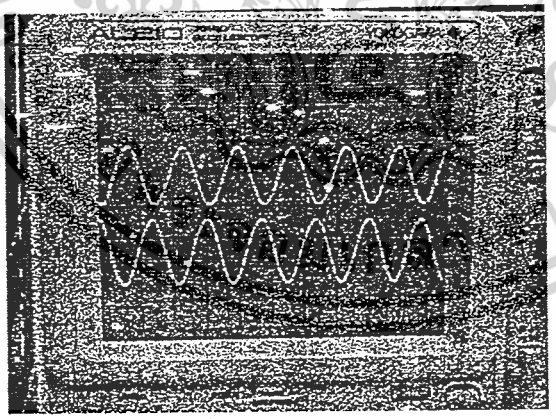
รูปที่ 4-5 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1200 Hz

4.1.2.5 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1500 Hz ใช้คามถี่การชักตัวอย่างเป็น 8 Hz ขนาด เวลาต่อช่อง เท่ากับ 0.5 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



รูปที่ 4-6 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1500 Hz

4.1.2.6 การทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1200 Hz ใช้คามถี่การชักตัวอย่างเป็น 20 Hz ขนาด เวลาต่อช่อง เท่ากับ 0.5 ms สัญญาณด้านบนเป็นอินพุท สัญญาณด้านล่างเป็นเอาท์พุท



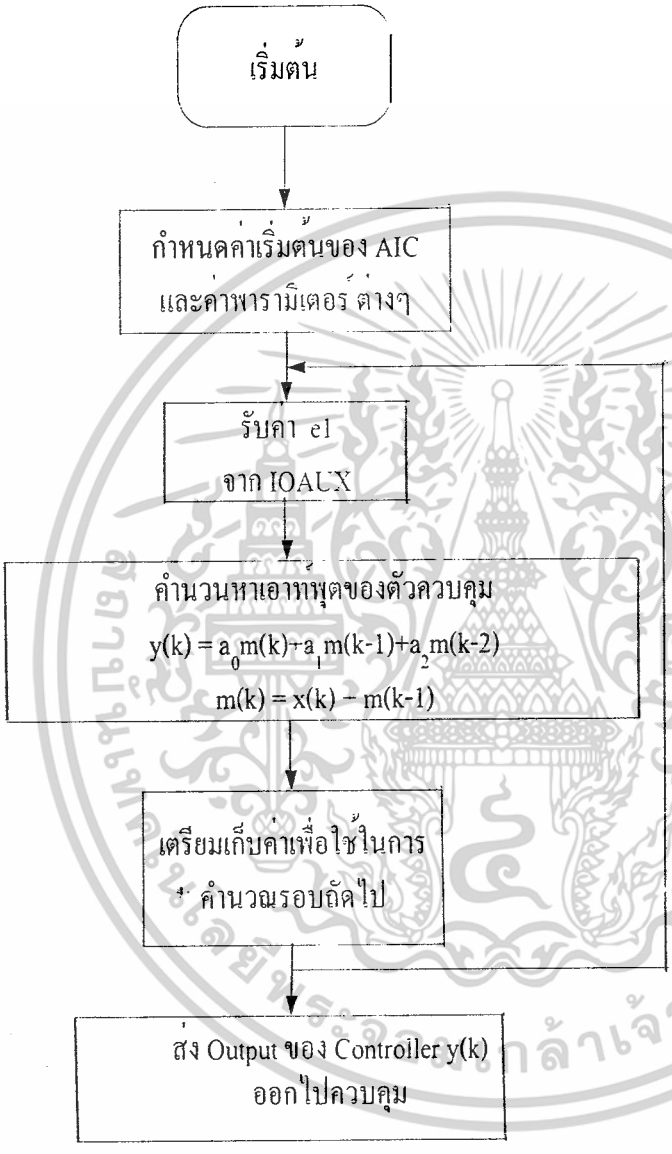
รูปที่ 4-7 ผลการทดลองเมื่อความถี่ของสัญญาณอินพุทเป็น 1200 Hz

จากผลการทดลองจะเห็นได้ว่า ความถี่ของสัญญาณอินพุทมีผลต่อการล้าหลังระหว่างสัญญาณอินพุทกับสัญญาณเอาท์พุท โดยเมื่อความถี่มากขึ้นจะทำให้เกิดการล้าหลังมากขึ้นจนกระทั่งถึงค่าความถี่ค่าหนึ่งจะทำให้เกิดการล้าหลังจนครบลูก เช่นที่ความถี่ 900 Hz เป็นต้น

จากผลการทดลองที่ 4.1.2.4 และการทดลองที่ 4.1.2.6 ซึ่งเป็นการทดลองโดยใช้ความถี่ของสัญญาณอินพุตเท่ากันคือ 1200 Hz แต่ใช้ค่าความถี่ชักตัวอย่างไม่เท่ากัน จากรูปที่ 4-5 และรูปที่ 4-7 จะเห็นได้ว่าค่าความถี่ชักตัวอย่างไม่มีผลต่อการล้าหลังของสัญญาณ

4.2 ผลการทดลองการควบคุมแบบ PID

การเขียนโปรแกรมควบคุมแบบพีไอดี มีผังงานดังนี้



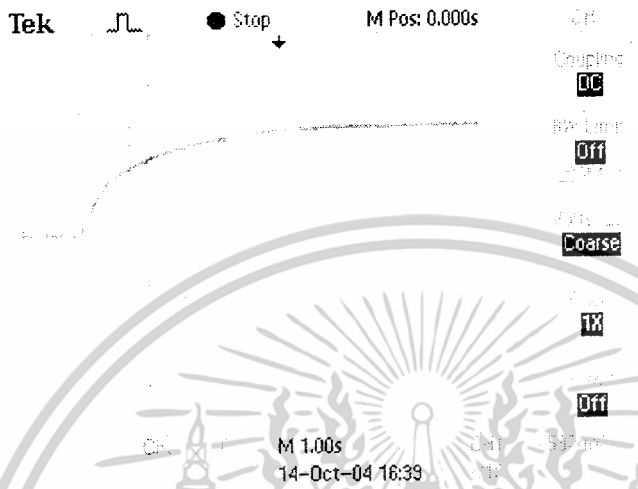
รูปที่ 4-8 แสดงผังงานในการเขียนโปรแกรมตัวควบคุมแบบพีไอดี

ซึ่งเขียนได้เป็นโปรแกรมดังนี้

```
*CONTROL2.ASM - PID controller using the C31 DSK
        .start      ".text", 0x809900          ; starting address of text
        .start      ".data", 0x809C00          ; starting address of data
        .include    "AICCOM31.ASM"             ; include AIC communication routines
        .entry      BEGIN                      ; start of code
        .text                                  ; assemble into text
BEGIN   LDP         @COEFF_ADDR                ; init to data page 128
CALL    AICSET                                 ; initialize AIC
PID     LDI         @COEFF_ADDR, AR0           ; AR0 points to coefficients address
        LDI         @DLY_ADDR, AR1             ; AR1 points to addr of delay samples
        CALL        AICIO_P                    ; call AIC for polling
        FLOAT       R6, R3                     ; stage input
        MPYF3       *AR0++,*AR1--, R0          ; b [0]*dly[0]
        LDI         STAGES-1, RC               ; initialize stage counter
        MPYF3       *AR0++,*AR1--, R1          ; b [1]*dly[1]
||      SUBF3       R0, R3, R3                 ; input-b [0]*dly[0]
        MPYF3       *AR0++,*AR1--, R0          ; a [1]*dly[0]
||      SUBF3       R1, R3, R2                 ; dly=input-b[0]*dly[0]-b[1]*dly[1]
        MPYF3       *AR0++,*AR1--, R1          ; a [2]*dly[1]
        ADDF3       R0, R1, R3                 ; a [2]*dly[1]+a[1]*dly[0]
        LDF         *AR1, R4                   ; dly[2]
||      STF         R2,*AR1++                  ; dly[0] = dly
        MPYF3       R2,*AR0++, R2              ; dly*a[0]
||      STF         R4,*AR1++                  ; dly[1] = dly[0]
        MPYF3       *AR0++,*AR1--, R0          ; b [0]*dly[0]
||      ADDF3       R2, R3, R3                 ; controller output
        FIX         R3, R7                     ; convert output to integer
        BR          PID
        .data                                  ;b[0] b[1] a[1] a[2] a[0]
COEFF   .float      -1.0000E+0, 0.0000E+0, -0.2700E+0, 0.2025E+0, 2.2275E+0
DLY     .float      0, 0                       ; init delay var for each stage
STAGES  .set 1                                 ; number of stages
COEFF_ADDR .word    COEFF                      ; address of COEFF
DLY_ADDR .word      DLY                        ; address of DELAY
AICSEC  .word       162Ch, 1h, 3872h, 63h      ; AIC config data, Fs = 10 kHz
        .end                                   ; end
```


รูป 4-9 โปรแกรมแกรมภาษาAssembly สำหรับการควบคุมแบบ PID

จากโปรแกรมจะเห็นได้ว่าเมื่อกำหนดค่า K_p = 1.35, K_i = 13.5, Kd = 0.02025 ซึ่งเท่ากับ a_0 = 2.2275, a_1 = -0.27, $a2$ = 0.2025 และมีความถี่การสุ่มเท่ากับ 10 kHz เมื่อนำมาทดสอบกับสัญญาณระดับ เพื่อดูผลการควบคุม จะได้รูปกราฟดังรูปที่ 4-10



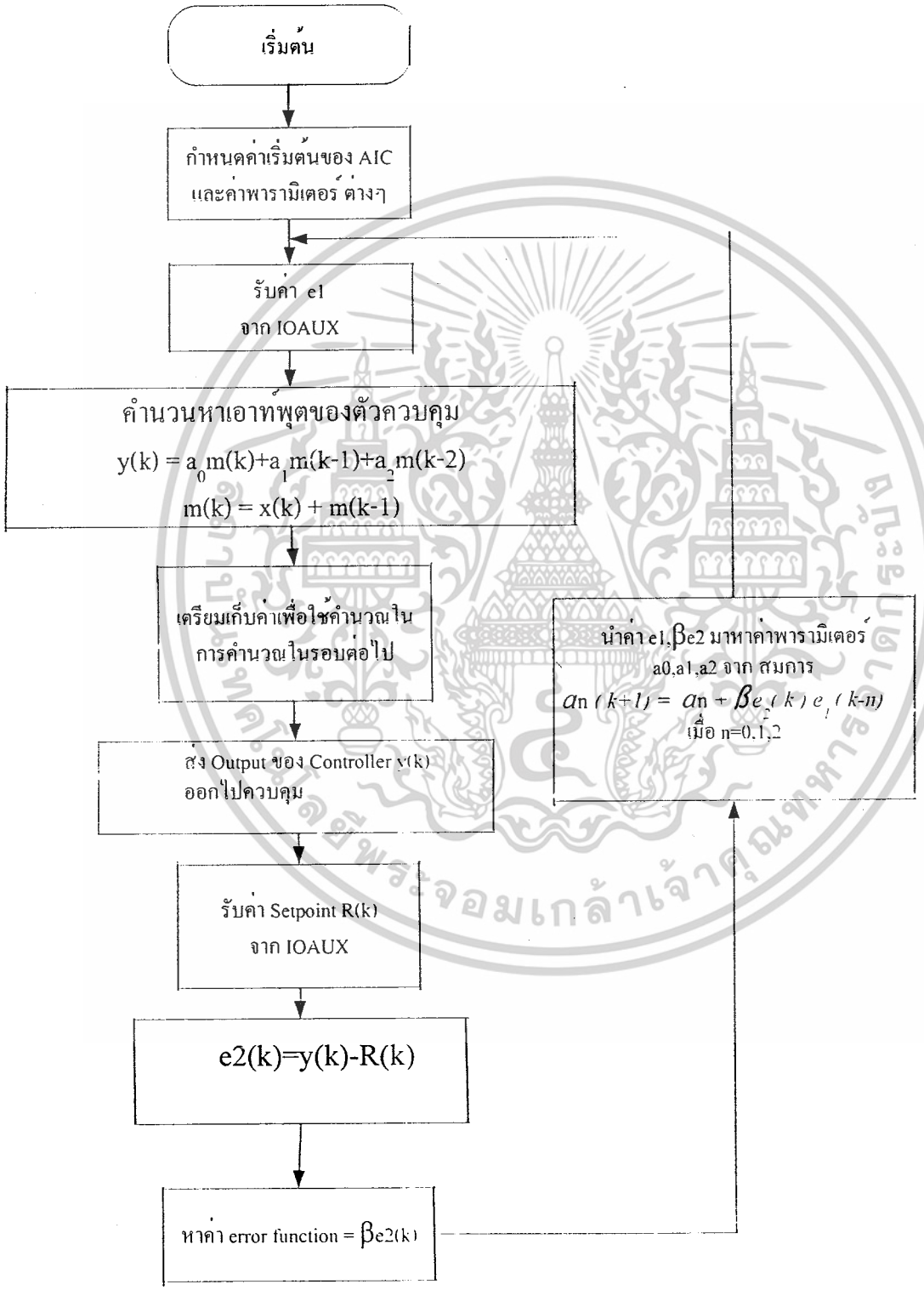
รูปที่ 4-10 ผลตอบสนองต่อสัญญาณอินพุตอ้างอิงแบบขั้นบันไดที่ระดับ 2 โวลต์ ของระบบควบคุมแบบพีไอดี

รูปที่ 4-11 ผลตอบสนองต่อสัญญาณอินพุตอ้างอิงแบบขั้นบันไดที่ระดับ 2 โวลต์ ของระบบควบคุมแบบพีไอดี และการใส่โหลดที่ค่า 1 kΩ

ผลการควบคุมแบบพีไอดีดังรูป 4-11 เมื่อมีการสับสวิตซ์โหลด ความเร็วรอบจะเปลี่ยนเล็กน้อยเพียงช่วงเวลาหนึ่ง ๆ แล้วจะกลับมาเป็นความเร็วรอบที่เท่าเดิม

4.3 ผลการทดลองการควบคุมแบบ Adaptive PID

การเขียนโปรแกรมควบคุมแบบพีไอดี มีผังงานดังนี้



รูปที่ 4-12 แสดงผังงานในการเขียนโปรแกรมตัวควบคุมแบบAdaptive PID

ซึ่งเขียนเป็นโปรแกรมได้ดังนี้

```
;ADAPTPID.ASM - Adaptive PID controller using the C31 DSK
              .start ".text", 0x809900       ; starting address of text
              .start ".data", 0x809C00       ; starting address of data
              .include "AICCOM31.ASM"        ; include AIC comm routines
              .entry BEGIN                   ; start of code
              .text                          ; assemble into text
BEGIN         LDP     @COEFF_ADDR            ; init to data page 128
              CALL    AICSET                 ; initialize AIC
              LDI     @ERR_ADDR, AR6         ; error function address
              LDI     @XNB_ADDR, AR2         ; AR2=bottom addr of input xn to filter
              LDI     LENGTH, BK             ; BK=length of circular buffer
PID           CALL    IOAUX                  ; get input error signal
              FLOAT   R6, R3                 ; stage input
              STF     R3,*AR2++%             ; store newest sample
              LDI     @COEFF_ADDR, AR0       ; AR0 points to coefficients address
              LDI     @DLY_ADDR, AR1         ; AR1 points to addr of delay samples
              MPYF3   *AR0++,*AR1++, R0      ; b [0]*dly[0]=b1u(n-1)
              MPYF3   *AR0--,*AR1--, R1      ; b [1]*dly[1]=b2u(n-2)
||            SUBF3   R0, R3, R3             ; input-b[0]*dly[0];R3=x(n)-b1u(n-1)
              MPYF3   *AR0--,*AR1++, R0      ; a [1]*dly[0];R0=a1u(n-1)
||            SUBF3   R1,R3,R2               ; u(n)=xn-b[0]*dly[0]-b[1]*dly[1]=x(n)-b1u(n-1)-b2u(n-2)
              MPYF3   *AR0++,*AR1--, R1      ; a [2]*dly[1]=R1=a2u(n-2)
              ADDF3   R0,R1,R3               ; a[2]*dly[1]+a[1]*dly[0];R3=a1u(n-1)+a2u(n-2)
              LDF     *AR1, R4               ; dly[0];R4=u(n-1)
||            STF     R2,*AR1++              ; dly [0] = dly; u (n-1) updated to -->u (n)
              MPYF3   R2,*AR0--, R2          ; dly*a [0]; R2=a0u (n); point to a0 to adapt
||            STF     R4,*AR1++              ; dly [1] = dly[0] ;u(n-2)<-u(n-1) to update
              ADDF3   R2, R3, R3             ; controller out;y=a0u(n)+a1u(n-1)+a2u(n-2)
              FIX     R3, R7                 ; convert to integer for output
              CALL    IOPRI                  ; get reference desired signal d
              FLOAT   R6, R4                 ; R4=reference desired signal d
              SUBF3   R3, R4, R0             ; R0=error signal=d-y
              MPYF    @BETA, R0              ; ERR function=e*beta
              STF     R0,*AR6                ; store error function
              CALL    ADAPT                  ; call ADAPT subroutine
              BR      PID                    ; branch back/repeat with new input sample
;ADAPTATION ROUTINE
ADAPT         MPYF3   *AR6,*AR2++%, R0       ; error function*x (n-(N-1)) ->R0=erfx(n-2)
              LDF     *AR0, R3               ; w (N-1) -> R3=a2
              ADDF3   R3, R0, R2             ; w (n-1-i) +erf*x(n-(N-1-i));R2=a2+erfx(n-2)
              STF     R2,*AR0--              ; store/upgrade a2 coeff
              MPYF3   *AR6,*AR2++%, R0       ; erf*x(n-(N-1-i))->R0=erfx(n-1)
              LDF     *AR0, R3               ; load subsequent w(k) ->R3=a1
              ADDF3   R3,R0,R2               ; w(n+1)=w(n)+erf*x(n);R2=a1+erfx(n-1)
              STF     R2,*AR0++ (2)          ; store/upgrade a1 coeff;then points to a0
              MPYF3   *AR6,*AR2++%,R0        ; R0=erfx(n);erf*newest sample in circ buffer
              LDF     *AR0, R3               ; R3=a0
              ADDF3   R3, R0, R2             ; R2=a0+erfx (n)
              STF     R2, *AR0               ; store/upgrade a0
              RETS                           ; b[0] b[1] a[1] a[2] a[0]
COEFF         .float  -1.0000E+0, 0.0000E+0, -1.0000E+0, 0.0000E+0, 1.0000E+0
DLY           .float 0, 0                    ; init delay var for each stage
COEFF_ADDR    .word   COEFF                  ; address of COEFF
DLY_ADDR      .word   DLY                    ; address of DELAY
XNB_ADDR      .word   XN+LENGTH-1            ; bottom addr of cir buffer for error signal xn
ERR_ADDR      .word   ERR_FUNC               ; address of error function
ERR_FUNC      .float  0                      ; initialize error function
BETA  .float  10E-14                         ; rate of adaptation constant
AICSEC        .word   162Ch, 1h, 244Ah, 73h  ; AIC config data, Fs = 16/2 kHz
LENGTH        .set    3                      ; Length of circular buffer for xn
              .brstart "XN_BUFF",16          ; align on 16-word boundary
XN            .sect "XN_BUFF"                ; section for buffer
              .loop   LENGTH                 ; loop length (3) times
              .float 0                       ; initialize buffer to zero
              .endloop                       ; end of loop
              .end                           ; end
```

รูปที่ 4-13 โปรแกรมแกรมภาษาAssembly สำหรับการควบคุมแบบ Adaptive PID

โปรแกรมADAPTPID.ASM ในรูป 4- 13 อยู่บนพื้นฐานของการรวมกันระหว่างโปรแกรม IIR filter และโปรแกรม Adaptive filter สำหรับการกำจัดสัญญาณรบกวน จากหนังสือ *Digital Signal Processing Laboratory Experiments Using C and theTMS320C31 DSK* ซึ่งในตัวโปรแกรมจะประกอบด้วยโปรแกรมย่อยสองส่วนใหญ่ๆ คือ โปรแกรมย่อย PID กับ โปรแกรมย่อย ADAPT ในส่วนของโปรแกรมย่อย PID จะรับอินพุท e1(k) เพื่อนำไปคำนวณหาเอาท์พุทของตัวควบคุมแบบ PID y(k) โดยใช้พาพารามิเตอร์พีไอดีที่ได้คำนวณไว้แล้ว โปรแกรมย่อย PID ยังคำนวณหาค่าความผิดพลาดe_2ระหว่างค่าอ้างอิงR(k)กับค่าเอาท์พุทของตัวควบคุมy(k)ด้วย นอกจากนั้นส่วนโปรแกรมย่อย ADAPT จะทำหน้าที่ปรับค่าพารามิเตอร์PID คือ a_0, a_1, a_2 โดยใช้สมการ

$$a_n(k+1) = a_n + \beta e_2(k)\, e_1(k-n) \quad n = 0, 1, 2$$

ในโปรแกรมนี้ต้องใช้อินพุททั้งสองทาง คือ ทั้ง ทาง Primary input (IOPRI) คือ ทาง RCA JACK และ ทาง Secondary input (IOAUX) คือ ทางขา 3 ของ JP3 ซึ่ง การเรียกใช้งานทาง Secondary input นี้สามารถเรียกใช้ได้จากการเซตโปรแกรมในส่วนของ AICSEC

ในโปรแกรม ADAPTPID.ASM จะมีการเรียกส่วนโปรแกรมย่อยเพื่อใช้ในการเชื่อมต่อสัญญาณอนาลอกAICCOM31.ASM (มีในภาคผนวก) รวมอยู่ด้วยในโปรแกรม ซึ่งเป็นโปรแกรมการกำหนดรีจิสเตอร์ที่เป็นอินพุท และเอาท์พุท ได้แก่ R6 และ R7 ตามลำดับ และในการรันโปรแกรมในที่นี้ เราจะรันบนโปรแกรม Code Explore

สำหรับค่าเริ่มต้น b_0 และ b_1 ที่ถูกฟิกส์ค่าไว้ที่ -1 และ 0 เสมอ นั้น เพราะเรากำหนดว่าให้ตัวควบคุมPIDนี้เป็นแบบฟิกส์โพล ดังนั้น ค่า b_0 และ b_1 จึงไม่ได้ถูกปรับเปลี่ยน ส่วนค่าพารามิเตอร์ a_0, a_1, a_2 จะมีค่าเท่ากับ 1. 0 . 0 ซึ่งค่าจะเปลี่ยนไปตามโปรแกรม ซึ่งเราจะไม่สามารถเลือกค่าพารามิเตอร์ทั้งสามนี้เท่ากับ 0 ทั้งหมดได้ เพราะจะทำให้ลูปการควบคุมถูกเปิด

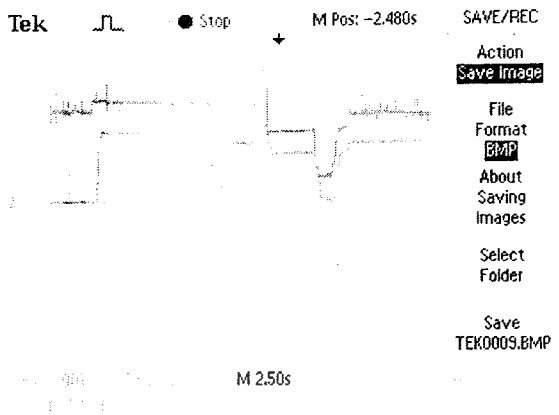
จากวงจรในรูป 3-8 สามารถหาโมเดลทางคณิตศาสตร์ของกระบวนการซึ่งในที่นี้คือมอเตอร์ซึ่งเราจะได้โมเดลเป็น

$$G(s) = \frac{A}{s+B} = \frac{72.5}{s+12.5}$$

เมื่อ 1/B คือ ค่าเวลาคงตัว= 0.8 วินาที

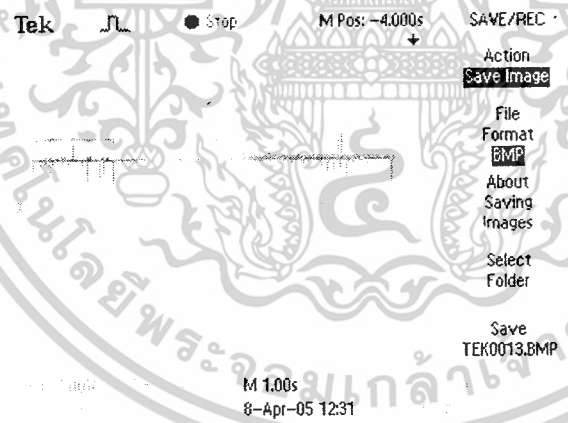
A/B คือ ค่า DC gain (steady-state gain) = 5.8 เท่า

จากสมการจะเห็นได้ว่ามอเตอร์มีโพลอยู่ที่ -12.5 และเมื่อเรารันโปรแกรมโดยเราตั้ง set point ที่ค่าต่างๆ ให้กับระบบในรูป3-8 และ กำหนดให้Adaptation rate = $2.5 * 10^{-12}$ จะได้ผลการทดลองดังรูปที่ 4-14



รูปที่ 4-14 ผลตอบสนองต่อสัญญาณอินพุตที่เปลี่ยนแปลง
ของระบบควบคุมแบบอะแดปทีฟพีไอดี

จากรูปที่ 4-14 สัญญาณเส้นบนแสดงค่าสัญญาณอินพุตที่ตั้งไว้ ส่วนสัญญาณกราฟเส้นล่าง แสดงค่าสัญญาณเอาท์พุตที่ออกจากชุดมอเตอร์ ซึ่งผลที่ได้แสดงให้เห็นว่ากระบวนการมีการติดตาม สัญญาณอินพุตที่ใส่เข้าไป แต่ยังมีออฟเซตและช่วงเวลาเดดไทม์(dead time)เกิดขึ้นอีกด้วย



รูปที่ 4-15 ผลตอบสนองของระบบเมื่อการสับสวิตซ์โหลด
ของระบบควบคุมแบบอะแดปทีฟพีไอดี

จากรูปที่ 4-15 สัญญาณเส้นบนแสดงค่าสัญญาณเอาท์พุตที่ออกมาจากชุดมอเตอร์ ส่วนเส้นกราฟด้านล่างแสดงถึงค่าสัญญาณเอาท์พุตที่ออกจากบอร์ดดีเอสเคที่นำมาใช้เป็นสัญญาณควบคุมมอเตอร์ ซึ่งผลการทดลองเมื่อใส่โหลดเข้าไปในระบบ จะเห็นได้ว่ากราฟเส้นสัญญาณเอาท์พุตที่ออกมาจากชุดมอเตอร์นั้นจะตกลงมาค่าหนึ่งคงที่ตลอดที่มีการใส่โหลด ส่วนสัญญาณเอาท์พุตที่ออกมาจากบอร์ดดีเอสเคมีระดับสัญญาณคงที่ตลอด

บทที่5
บทสรุปและวิจารณ์

5.1 สรุปผลการทดลอง

5.1.1 สรุปผลการทดลองการใช้วงจรเชื่อมต่อสัญญาณอนาลอก

การทดสอบผลของความถี่ที่มีผลต่อการล้าหลังของสัญญาณ(Lag) จากการทดลองพบว่าเมื่อความถี่ของสัญญาณอินพุตมากขึ้น จะทำให้มีการล้าหลังกันระหว่างสัญญาณอินพุทกับสัญญาณเอาท์พุทเพิ่มมากขึ้นด้วย ซึ่งสาเหตุอันเนื่องมาจากผลของวงจรกรองสัญญาณความถี่ต่ำผ่านทางด้านเอาท์พุทของวงจรเชื่อมต่ออนาลอกของบอร์ดดีเอสเค

นอกจากนี้ได้มีการทดสอบผลของความถี่ในการชักตัวอย่างยังไม่มีผลกับการล้าหลังของสัญญาณด้วย

5.1.2 สรุปผลการทดลองการควบคุมแบบ PID

การหาค่าพารามิเตอร์ที่เหมาะสมของตัวควบคุม พีไอดี เป็นสิ่งจำเป็นเพื่อให้ได้การควบคุมที่ดีที่สุด ซึ่งจะทำให้เราได้ระบบที่มีสมรรถนะหรือข้อกำหนดตามที่ต้องการ ข้อกำหนด (Specification) ต่างๆ เช่น ความเที่ยงตรง. ความเร็วของผลตอบสนอง. เสถียรภาพสัมพัทธ์

จากผลการทดลองเมื่อมีการสับสวิตซ์โหลด ความเร็วรอบจะเปลี่ยนเล็กน้อยเพียงช่วงเวลาหนึ่ง แล้วจะกลับมาเป็นความเร็วรอบที่เกือบเท่าเดิม จะเห็นได้ว่า Steady-state error ถูกกำจัดให้หมดไป ซึ่งทำให้ระบบเข้าสู่ความเร็วที่ตั้งไว้ได้ ซึ่งแสดงให้เห็นว่าค่าพารามิเตอร์ที่นำมาใช้มีความเหมาะสมกับระบบ

5.1.3 สรุปผลการทดลองการควบคุมแบบ Adaptive PID

จากผลการทดลองเมื่อมีการเปลี่ยนค่าอินพุตที่ตั้งไว้ ระบบจะมีการติดตามสัญญาณอินพุตที่เปลี่ยนไป แต่ยังมีออฟเซตและช่วงเวลาเดดไทม์(dead time)เกิดขึ้นด้วย และเมื่อเราดูค่าพารามิเตอร์ของระบบ จะเห็นได้ว่าค่าพารามิเตอร์ไม่มีการเปลี่ยนแปลง ซึ่งแสดงให้เห็นว่าระบบไม่มีการอะแดปทีฟตามที่ได้ออกแบบ

5.2 ปัญหาที่เกิดขึ้น

1. ตัวRegulator ของ board DSK จะร้อนมากขณะทำงาน

2. ช่วงแรงดันVoltage ที่ออกมาจากบอร์ด (0-3V) กับ ช่วงแรงดันvoltageที่จะนำไปใช้ drive motor(2-6V) ต่างกัน

3. การอินเทอร์เฟสระหว่างบอร์ดดีเอสเคกับเครื่องคอมพิวเตอร์ ตัวบอร์ดจะไม่สามารถทำงานบนวินโดว์เวอร์ชั่นใหม่ๆได้

4. ในการทดลองการควบคุมแบบอะแดปทีฟพีไอดี ยังไม่สามารถควบคุมได้ตามต้องการ

5. วงจรอนาลอกอินเทอร์เฟสมีสัญญาณรบกวนจากบอร์ด

5.3 แนวทางการแก้ไขปัญหา

1. ต้องต่อโหลดภายนอกให้น้อยกว่า50mA หรือเลือกใช้power supply ที่ได้มาตรฐานที่จ่ายไฟอยู่ในช่วง 7-12 Vdc หรือ 6-9 Vac ที่กระแสอย่างน้อย 400- 1500 mA

2. ออกแบบและต่อวงจร Zero & Span เข้าช่วยในการเชื่อมต่อสัญญาณอนาลอกระหว่างบอร์ดDSKและชุดทดลอง

3. เวลาใช้บอร์ดให้ใช้บนระบบปฏิบัติการวินโดว์ 98 ลงมา

4. อาจมีการแก้ไขโปรแกรมและออกแบบกระบวนการ โดยการอนุมานกระบวนการให้เป็นรูปแบบของกระบวนการอันดับหนึ่งที่มีการหน่วงเวลา (FOPDT)

5. ต่อวงจรโดยใช้ไอซีที่ลดผลต่อสัญญาณรบกวนจากสัญญาณภายนอกมาใช้แทน

ภาคผนวก

ภาคผนวก

```
;-----------------------------------------------------------------------------------------------------------
; โปรแกรมภาษาซี การควบคุมแบบ Adaptive PID
;-----------------------------------------------------------------------------------------------------------

/*ADAPTPID.C -Real-Time Adaptive PID controller algorithm */
#include "aiccomc2.c" /*Include AIC comm routines*/
int AICSEC[4] = {0x162C,0x1,0x244A,0x73 }; /*AIC config data, Fs = 16/2 kHz */
#define beta 10e-14 /* Rate of convergence */
float a[3]={1.0,0.0}; /* Numerator coefficients */
float b[2]= {-1.0} ; /* Denominator coefficients*/
float dly[2] = {0}, input[2]={0},yn=0; /* Global variables */
void pid(float xn) /* Standard PID controller based on a real-time IIR algorithm*/
{
float un;
un = xn - b[0] * dly[0] - b[1] * dly[1]; /* Calculate yn*/
yn = a[2]*dly[1] + a[1]*dly[0] + a[0]*un;
dly[1] = dly[0]; /* Update the delay samples*/
dly[0] = un;
}
void adapt(float error_function, float xn) /* Adaptation routine */
{
a[2] = a[2] + (error_function * input[1]); /* Update the numerator coefficients */
a[1] = a[1] + (error_function * input[0]);
a[0] = a[0] + (error_function * xn);
input[1]=input[0]; /* Update the adaptors delays */
input[0]=xn;
}
void c_int05() /* Interrupt routine to service incoming and outgoing CODEC data */
{
float control_error,desired,error_func;
```

```
control_error = ((float) UPDATE_AUXSAMPLE(yn)): /* Get sample from IOAUX*/
desired = (float) UPDATE_PRISAMPLE(yn): /* Get sample from IOPRI */
pid(control_error): /* Call standard real-time PID */
error_func = (desired - yn) * beta; /* Calculate error function */
adapt(error_func.control_error); /* Call adaptation routine */
}
void main()
{
AICSET_I(): /* Setup CODEC and wait for interrupt */
while (1) {} /*infinite loop*/
}
```

```
;-------------------------------------------------------------------------------
; โปรแกรม AICCOM31.ASM
;-------------------------------------------------------------------------------
*AICCOM31.ASM - AIC COMMUNICATION ROUTINES - POLLING OR INTERRUPT
      .data             ;assemble into data section
PBASE  .word  808000h     ;peripheral base address
SETSP  .word  0E970300h   ;serial port set-up data
ATABLE .word  AICSEC      ;SP0 AIC init table address
      .text             ;assemble into text section
AICSET PUSH   AR0         ;save AR0
      PUSH   AR1         ;save AR1
      PUSH   R0          ;save R0
      PUSH   R1          ;save R1
      LDI    @PBASE,AR0  ;AR0 -> 808000h
      LDI    1,R0        ;timer CLK=H1/2*(AIC master CLK)
      STI    R0,*+AR0(28h) ;timer period reg(TCLK0=6.25 MHZ)
      LDI    03C1h,R0    ;init timer global register
      STI    R0,*+AR0(20h) ;reset timer
      LDI    62h,IOF     ;AIC reset = 0
      LDI    @ATABLE,AR1 ;AR1 -> AIC init data
      RPTS   99          ;repeat next instr 100 times
      NOP               ;keep IOF low for a while
      LDI    131h,R0     ;X & R port control register data
      STI    R0,*+AR0(42h) ;FSX/DX/CLKX=SP operational pins
      STI    R0,*+AR0(43h) ;FSR/DR/CLKR=SP operational pins
      LDI    @SETSP,R0   ;RESET->SP:16 bits,ext clks,std mode
      STI    R0,*+AR0(40h) ;FSX=output&INT enable SP global reg
      LDI    0,R0        ;R0 = 0
      STI    R0,*+AR0(48h) ;clear serial port XMIT register
      OR     06h,IOF     ;bring AIC out of reset
      LDI    03h,RC      ;RC=3 to transmit 4 values
      RPTB   SECEND      ;repeat 4 data transmit of sec com
```

```
    CALL   TWAIT        ;wait for data transmit
    LDI    03h,R0       ;value for secondary XMIT request
    STI    R0,*+AR0(48h) ;secondary XMIT request to AIC
    CALL   TWAIT        ;wait for data transmit
    LDI    *AR1++(1),R0  ;AR1 -> next AIC init data
SECEND STI   R0,*+AR0(48h) :DTR = current AIC data
    POP    R1           ;restore R1
    POP    R0           ;restore R0
    POP    AR1          ;restore AR1
    POP    AR0          ;restore AR0
    RETS                ;return from subroutine
AICSET_I                ;---CONFIG FOR INTERRUPT ----------
    CALL   AICSET       ;call AICSET routine
    LDI    0h,IF        ;clear IF register
    OR     10h,IE       ;enable EXINT0 CPU interrupt
    OR     2000h,ST     ;global interrupt enable
    RETS                ;return from subroutine
;--------------------TRANSMIT WAIT ROUTINE-------------------------
TWAIT  PUSH   AR0       ;save AR0
    PUSH   R0           ;save R0
    LDI    @PBASE,AR0   ;AR0 -> 0808000h
TW1    LDI    *+AR0(40h),R0 ;R0=content of SP global control reg
    AND    02h,R0       ;see if transmit buffer is ready
    BZ     TW1          ;if not ready, try again
    POP    R0           ;restore R0
    POP    AR0          ;restore AR0
    RETS                ;return from subroutine
;--------------------AIC TRANSFER ROUTINE--------------------------
AICIO_I LDI    R7,R6        ;copy output to modify for AIC
    LSH    2,R6         ;two LSB must=0 for primary AIC comm
IO     PUSH   AR0          ;save AR0
    LDI    @PBASE,AR0   ;AR0 -> 0808000h
```

```
    STI   R6,*+AR0(48h)  ;DTR = next data for AIC D/A
    LDI   *+AR0(4Ch),R6  ;R6 = DRR data from AIC A/D
    LSH   16,R6          ;left shift for sign extension
    ASH   -18,R6         ;right shift keeping sign
    POP   AR0            ;restore AR0
    RETS                 ;return from subroutine
;----------------------AIC POLLING ROUTINE----------------------------
AICIO_P CALL  TWAIT      ;wait for data to be transferred
    CALL  AICIO_I        ;call AIC transfer routine
    RETS                 ;return from subroutine
SW_IO  PUSH   AR0        ;save AR0
    LDI   @PBASE,AR0     ;AR0 -> 0808000h
    LDI   R7,R6          ;copy output to modify for AIC
    LSH   2,R6           ;prepare for secondary AIC com
    OR    03h,R6         ;set two LSB for secondary com
    CALL  TWAIT          ;wait for data to be transferred
    CALL  IO             ;call AIC transfer routine
    CALL  TWAIT          ;wait for data to be transferred
    STI   R1,*+AR0(48h)  ;DTR = next data for AIC control
    POP   AR0            ;restore AR0
    RETS                 ;return from subroutine
;SUBROUTINES FOR PRIMARY OR AUXILIARY INPUT
IOPRI  PUSH   R1         ;save R1
    LDI   063h,R1        ;load secondary com data into R1
    CALL  SW_IO          ;call IO routine to switch inputs
    POP   R1             ;restore R1
    RETS                 ;return from subroutine
IOAUX  PUSH   R1         ;save R1
    LDI   073h,R1        ;load secondary com data into R1
    CALL  SW_IO          ;call IO routine to switch inputs
    POP   R1             ;restore R1
    RETS                 ;return from subroutine
```

;----------

; โปรแกรม **Loop.asm**

;----------

```
;LOOP.ASM - LOOP PROGRAM. CALLS AIC ROUTINES IN AICCOM31.ASM
            .start      ".text".0x809900        ;starting address for text
            .start      ".data".0x809C00        :starting address for data
            .include    "AICCOM31.ASM"  ;AIC communication routines
            .data                               :data section
AICSEC      .word   162Ch,1h,4892h.67h  ;Fs = 8 kHz
            .text                               :text section
            .entry          BEGIN               :start of code
BEGIN       LDP     AICSEC      ;init to data page 128
            CALL        AICSET      ;init AIC
LOOP        CALL        AICIO_P             :R6 = input, R7 = output
            LDI     R6,R7           ;output R7=new input in R6
            BR      LOOP        ;loop continuously
        .end                                :end
```