PHASE-LOCKED CONTROLLER

โดย

นายจักรภาษณ์ วิศวกุล

นายจิระศักดิ์ ปิยพสุนทรา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2534

ปริญญานิพนธ์ปีการศึกษา ๒๕๓๔

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง PHASE-LOCKED CONTROLLER

คณะผู้จัดทำ

จักรภาษณ์ วิศวกุล

จิระศักดิ์ ปิยพสุนทรา

..... อาจารย์ที่ปรึกษา

(ดร.โยธิน เปรมปราณีรัชต์)

..... กรรมการสอบ

( )

..... กรรมการสอบ

( )

..... กรรมการสอบ

( )

บทคัดย่อ

PHASE LOCKED CONTROLLER เป็นการเลียนแบบการทำงานของ Phase Locked Loop (PLL) โดยใช้คอมพิวเตอร์ส่วนบุคคล เพื่อหาคุณสมบัติของโปรแกรมที่จำลองการทำงานของเฟสดีเทคเตอร์แบบตรวจจับเฟสและความถี่ โดยโปรแกรมที่ใช้มีสองอัลกอริทึม คือแบบที่ใช้วิธีการเปิดตารางการเปลี่ยนสถานะ และแบบที่ใช้สมการลอจิก แยกเป็นสอลโปรแกรม โดยจะนำโปรแกรมเฟสดีเทคเตอร์นี้ไปควบคุมความถี่ที่ได้จาก Voltage Control Oscillator โดยต่อวงจรในลักษณะครบลูฟตามบล็อคไดอะแกรมของเฟสล็อคลูฟ เพื่อใช้เป็นแนวทางในการจำลองการทำงานของเฟสล็อคลูฟด้วยคอมพิวเตอร์ หรือใช้โปรแกรมเฟสดีเทคเตอร์ดังกล่าวให้เป็นส่วนหนึ่งของระบบควบคุมหรือระบบอื่นๆ ซึ่งผลการทดลองของโปรแกรมนี้ ได้เสนอไว้ในปริญญานิพนธ์นี้

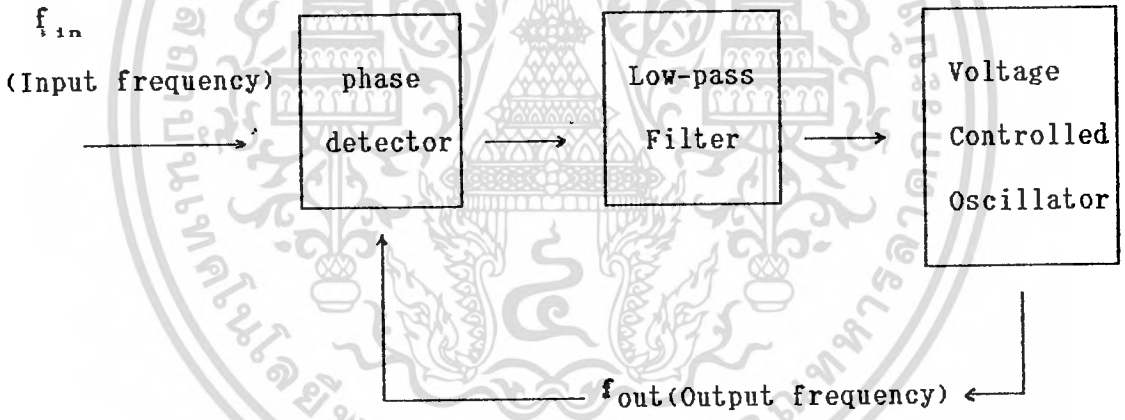
ABSTRACT

Phase locked controller is the simulation of Phase Detector in using personal computer. Main purpose is to search for characteristic of program that simulate phase and frequency detector. There is two way to obtain the same purpose in programing. The first is uses Look-Up Table method and another is Logic Equation of phase detector. This phase detector program uses to control frequency from Voltage Control Oscillator. In the experiment, program is a part of phase locked loop circuit.

The result may being the new way to simulation of phase locked loop by personal computer. In other way, this program can be used as an element of control or other system. The experiment result is also presented in this thesis.

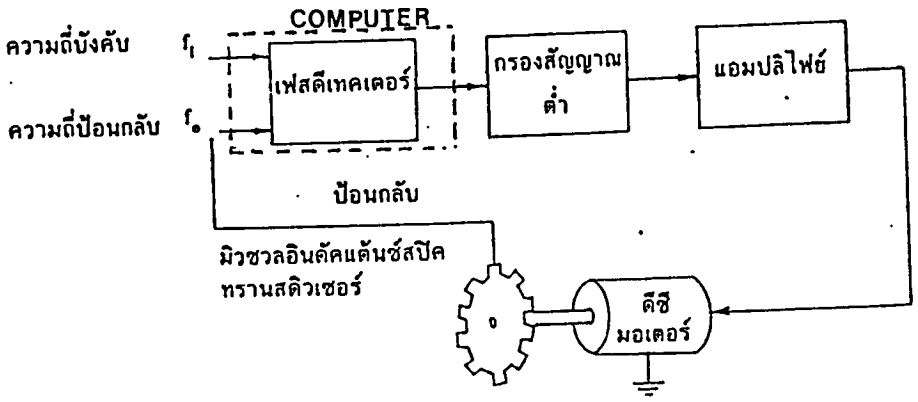
บทนำ

เนื่องด้วยในปัจจุบัน ความต้องการระบบควบคุมที่มีความเที่ยงตรงสูง มีเสถียรภาพมีเพิ่มมากขึ้น ประกอบกับวงจรรวมไอซีของวงจรต่าง ๆ ก็มีแพร่หลายมากขึ้น ดังนั้น การพัฒนาและวิจัยจึงมีผลต่อการเปลี่ยนแปลงในวงการวิศวกรรมมาก เครื่องมือที่จะนำมาใช้ในการวิเคราะห์และจำลองการทำงานของระบบต่าง ๆ ที่นิยมกันมากคือคอมพิวเตอร์ส่วนบุคคล ซึ่งสะดวกต่อการปรับปรุงและแก้ไขเป็นอย่างมาก หลักการของเฟสล็อคลูฟมีการนำมาใช้นานแล้วในระบบสื่อสาร ต่อมามีการประยุกต์ใช้กับการควบคุมความเร็วมอเตอร์ โดยอ้างถึงความแม่นยำสูง ส่วนประกอบที่สำคัญของเฟสล็อคลูฟคือ Phase Detector (PD), Low Pass Filter (LPF) และ Votage Controlled Oscillator (VCO) ดังรูปที่ ๑



รูปที่ ๑ บล็อกไดอะแกรมของเฟสล็อคลูฟ

หากเราแทนส่วนของวงจรกำเนิดสัญญาณควบคุมโดยแรงดันด้วยมอเตอร์และเอนโคดเดอร์พร้อมกันนั้น ก็เลียนแบบส่วนของตัวตรวจจับเฟสด้วยซอฟท์แวร์ โดยใช้โปรแกรมภาษาเอสแซมบลี ดังนั้น จะได้บล็อคไดอะแกรมคร่าว ๆ ของระบบควบคุมมอเตอร์ด้วยคอมพิวเตอร์ส่วนบุคคล(Personnal Computer) ดังรูปที่ ๒



รูปที่ ๒ บล็อกไดอะแกรมระบบควบคุมความเร็วมอเตอร์ด้วยคอมพิวเตอร์ส่วนบุคคล

จะเห็นได้ว่าตัวตรวจสอบเฟส มีความสำคัญต่อคุณสมบัติของเฟสล็อคลูปมาก การปรับปรุงโครงสร้างการทำงานจะมีผลต่อโครงสร้างของโปรแกรม แต่เนื่องจากตัวตรวจสอบเฟสมีมากมายหลายแบบ มีคุณสมบัติแตกต่างกันไป จึงมีการเลือกโดยใช้หลักเกณฑ์ ดังนี้

๑. ให้ช่วงพิสัยการล็อคกว้าง

๒. สามารถนำมาเขียนเป็นโปรแกรมแล้วใช้เวลาน้อยที่สุดในการทำงาน

ในวิทยานิพนธ์นี้ ได้เลือกตัวตรวจสอบเฟสที่สองของไอซ์เบอร์ ๔๐๔๔ ซึ่งเป็นคุณสมบัติของวงจรซีเควลเชียลประเภทหนึ่ง

สารบัญ

สารบัญภาพ

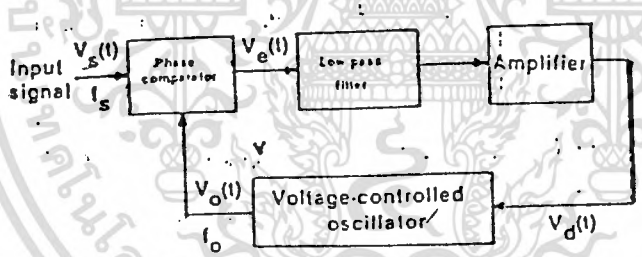
สารบัญตาราง

บทที่ ๑

หลักการของเฟสล็อคลูฟ

เฟสล็อคลูฟ(PLL) เป็นวงจรอิเล็กทรอนิกส์เซอร์โวลูฟ ประกอบด้วยตัวเปรียบเทียบเฟส โลว์พาสฟิสเตอร์ และโวลท์เตจคอนโทรลออสซิเลเตอร์ (VCO) เฟสล็อคลูฟจะควบคุมให้ VCO สร้างความถี่ขึ้นสอดคล้อง(synchronize)กับสัญญาณเข้า ปัจจุบันนี้เฟสล็อคลูฟจะปรากฏออกมาในรูปของวงจรรวม(ไอซี)ที่มีลักษณะเดียวกับวงจรรวมออปแอมป์ ซึ่งสามารถนำไปประยุกต์ใช้งานต่าง ๆ ได้มาก ทั้งการประมวลสัญญาณแบบอนาลอกและดิจิตอล

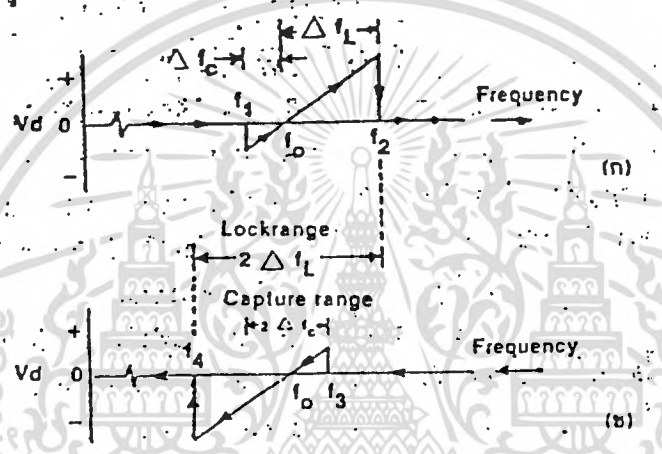
เฟสล็อคลูฟ คือระบบที่มีการป้อนความถี่ย้อนกลับประกอบด้วยตัวเปรียบเทียบเฟสและความถี่ ตัวกรองความถี่ต่ำผ่านและเออร์เรอร์แอมปลิไฟเออร์ ซึ่งอยู่ทางที่สัญญาณเดินไปหน้าและ VCO อยู่ในทางป้อนกลับ แผนภาพของระบบเฟสล็อคลูฟอย่างง่าย ๆ แสดงได้ดังในรูปที่ ๑.๑ อย่างไรก็ตามหลักการทำงานเบื้องต้นของเฟสล็อคลูฟ สามารถอธิบายได้ดังต่อไปนี้



รูปที่ ๑.๑ บล็อคไดอะแกรมของเฟสล็อคลูฟ

ขณะที่ยังไม่มีสัญญาณเข้าไปในระบบ แรงดันควบคุม(control voltage) $V_d(t)$ จะเท่ากับศูนย์ VCO จะทำงานโดยตั้งความถี่ไว้ที่ f_o เรียกว่า "free-running frequency" ถ้าสัญญาณเข้าไปใน ระบบเฟสคอมพาราเตอร์จะทำการเปรียบเทียบเฟสและความถี่ของสัญญาณอินพุตกับ VCO และผลิตแรงดันคลาดเคลื่อน $V_e(t)$ ซึ่งสัมพันธ์กับความแตกต่างของเฟส และความถี่ระหว่างสัญญาณทั้งสอง แรงดันคลาดเคลื่อนนี้จะถูกกรองและขยายส่งไปควบคุม VCO โดยแรงดันควบคุม $V_d(t)$ จะไปบังคับความถี่ VCO ให้เปลี่ยนไปในทิศทางที่จะลดความถี่ที่แตกต่างระหว่าง f_o และสัญญาณที่เข้า ถ้าความถี่ของสัญญาณที่เข้ามาใกล้เคียงกับ f_o จากการป้อนกลับของ PLL

ทำให้ PLL ชิงโครไนส์หรือล็อคกับสัญญาณที่เข้ามา ขณะที่ทำการล็อคนั้น ความถี่ VCO จะเท่ากับสัญญาณอินพุตแต่เฟสยังต่างกันอยู่ ความแตกต่างของเฟส ϕ_e มีความจำเป็นต่อการผลิตแรงดันคลาดเคลื่อนที่จะไปคอยปรับความถี่ VCO จากค่าฟรีรันนิ่ง ให้เท่ากับความถี่ที่เข้ามา f_e ดังนั้น PLL จะยังคงรักษาสภาพการล็อค การที่ระบบสามารถที่จะปรับตัวได้เอง ทำให้ PLL สามารถติดตามความถี่ที่เปลี่ยนไปของสัญญาณที่เข้าไปให้อยู่ในสภาพล็อคเช่นเดิม ช่วงของความถี่ซึ่ง PLL สามารถติดตามการล็อคกับสัญญาณที่เข้ามา เรียกว่า "lock range" ของระบบ ช่วงความถี่นี้จะมากกว่าช่วงความถี่ที่ PLL สามารถทำการล็อคอย่างแท้จริงกับสัญญาณอินพุต ช่วงความถี่หลังนี้เรียกว่า "capture range" ของระบบ ช่วงการแคปเจอร์ นี้ขึ้นอยู่กับ แบนด์เอคของฟิลเตอร์ และอัตราขยายลูฟบิด (K_v) ของระบบทั้งหมด



แสดงรูปที่ ๑.๒ Lock range และ Capture range

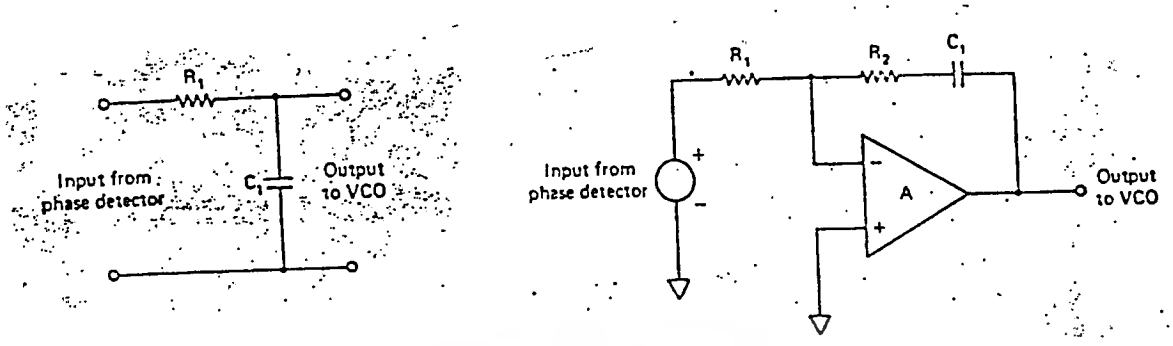
ตัวกรองความถี่ต่ำผ่าน (Low-Pass Filter)

ตัวกรองนี้ จะกรองสิ่งรบกวนและส่วนที่เป็นความถี่สูงทิ้งไป ควบคุมพฤติกรรมทางพลศาสตร์ (Dynamic) ของเฟสล็อคลูฟเช่น แบนวิดท์, ผลตอบสนองชั่วขณะ เป็นต้น ตัวกรองนี้ที่ใช้ในเฟสล็อคลูฟ มีได้หลายประเภท เช่น

๑. Passive Filter

๒. Active Filter

๓. Digital Filter



PASSIVE ACTIVE

รูปที่ ๑.๓ แสดงตัวกรองความถี่ต่ำผ่านประเภทต่าง ๆ

การทำงานของลูฟ ตัวกรองความถี่ต่ำผ่าน มีหน้าที่ ๒ อย่างคือ

๑. การลดค่าคลาดเคลื่อน ที่เป็นความถี่สูงที่ออกจากเฟสคอมพาราเตอร์ โดยการใช้คุณสมบัติการก่าจัดสัญญาณรบกวน

๒. ทำหน้าที่เหมือนกับ short-term memory สำหรับ PLL และจะแคปเจอร์กับสัญญาณใหม่อีกที เมื่อระบบหลุดออกจากการล็อค เนื่องจากสัญญาณรบกวนในช่วงทรานเซี้ยนท์

เนื่องจากตัวกรองความถี่ต่ำผ่าน ลดค่าแรงดันคลาดเคลื่อนของความถี่สูงระหว่างลูฟ มันเป็นตัวควบคุมแคปเจอร์โดยตรง และคุณสมบัติผลตอบสนองชั่วขณะของ PLL

การลดช่วงกว้างของ ฟิลเตอร์ จะส่งผลไปยังการทำงานของระบบคือ

๑. ขบวนการ แคปเจอร์ จะช้าลงและฟูลอินไทม์ จะเพิ่มขึ้น

๒. ช่วง แคปเจอร์ จะลดลง

๓. คุณสมบัติทาง interference-rejection ของ PLL จะดีขึ้น เพราะว่าแรงดันคลาดเคลื่อน เนื่องจากความถี่ของสัญญาณรบกวนจะถูกลดลงไป

๔. ผลตอบสนองชั่วขณะของ PLL ต่อการเปลี่ยนทันทีของสัญญาณเข้าใน ช่วงความถี่แคปเจอร์ จะอยู่ในลักษณะภายใต้การแดมพ์ (underdamped)

ในระบบเฟสล็อคลูฟ ต้องการพิสัยแคปเจอร์ที่กว้างมากในการล็อคกับสัญญาณอินพุต แต่พิสัยแคปเจอร์ที่กว้าง จะทำให้ระบบเฟสล็อคลูฟไวต่อสัญญาณรบกวน ในกรณีทั่วไป ควรจะเลือกพิสัยแคปเจอร์ที่เหมาะสมเพื่อให้กำจัดสัญญาณรบกวนได้ดี และล็อคกับสัญญาณอินพุตได้ในช่วงกว้าง

คุณสมบัติของ VCO มีความสำคัญ ดังต่อไปนี้

๑. แปลงความถี่เป็นโวลท์เตจของระบบเฟสล็อคลูฟ

๒. ความชันของโวลท์เตจเอาต์พุต

๓. ความเป็นเชิงเส้นของการแปลงความถี่เป็นโวลท์เตจ

บทที่ ๒

ตัวเปรียบเทียบเฟส (Phase Comparator)

ในระบบเฟสล็อคลูฟ จะประกอบไปด้วยส่วนหลัก ๆ คือ ตัวกรองความถี่ต่ำผ่าน VCO และตัวเปรียบเทียบเฟส ซึ่งตัวเปรียบเทียบเฟสนี้ทำขึ้นมาแล้ว ก็จะต้องใช้ หรือทำการคำนวณให้อยู่ในเงื่อนไขที่ตัวเปรียบเทียบจะยอมรับได้ ความยุ่งยากซับซ้อนของตัวเปรียบเทียบเฟส ก็จะมีผลต่อความสามารถในการเปรียบเทียบ และสภาวะการเข้าล็อคของวงจร

ตัวเปรียบเทียบเฟส ที่จะกล่าวดังต่อไป จะขอกล่าวเฉพาะ แบบที่เป็นดิจิตอล คือ สัญญาณเข้ามาจะสนใจ แค่มีสภาวะเป็น "๐" หรือ "๑" เท่านั้น อื่น ๆ นอกจากนี้ จะไม่นำมาคิดคำนวณ

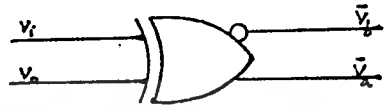
แบบ Exclusive-OR

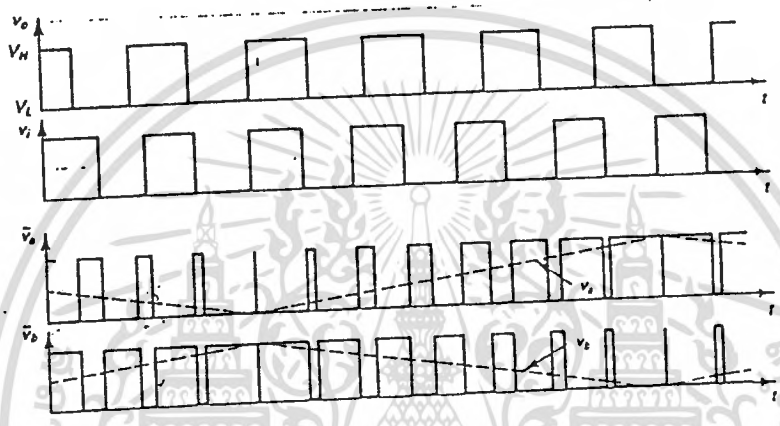
จะเป็นการนำสัญญาณอ้างอิง กับสัญญาณเปรียบเทียบมาทำ Exclusive-OR กับเอาต์พุตที่ออกมาจากตัวเปรียบเทียบเฟส จะเป็น "๐" ถ้าระดับของสัญญาณที่เข้ามาเหมือนกัน และจะเป็น "๑" ถ้ามีความแตกต่างกันของระดับสัญญาณ การตรวจจับของ Exclusive-OR จะเห็นได้ว่าใช้ระดับสัญญาณเป็นหลัก เมื่อพิจารณาสัญญาณแบบดิจิตอลที่มีคาบใด ๆ จะมีการทำงานคือ ระดับเปลี่ยนจาก "๐" ไปเป็น "๑" และจะคงค่า "๑" ไประยะหนึ่ง แล้วยังกลับมาเป็น "๐" เพราะสัญญาณให้ความสำคัญต่อระดับ ฉะนั้นค่า Duty Cycle ของสัญญาณ ก็จะมีผลต่อเอาต์พุต ดังนั้นสัญญาณที่จะนำมาเป็นสัญญาณอ้างและเปรียบเทียบ ควรจะมี Duty Cycle ๕๐ % จากรูป ถ้าสัญญาณมีความถี่เท่ากันและเฟสเดียวกัน จะได้เอาต์พุตที่ออกมาจะเป็น "๐" ตลอดค่าศักดาเฉลี่ยที่จะนำไปเข้า VCO เป็นศูนย์ และถ้าเฟสของสัญญาณต่างกัน ๑๘๐ องศา สัญญาณจากตัวเปรียบเทียบจะออกมาเป็น "๑" ซึ่งจะเห็นได้ว่า เมื่อลูฟอยู่ในสภาวะล็อค สัญญาณอ้างอิงกับสัญญาณเปรียบเทียบจะมีเฟสต่างกัน ๙๐ องศา

เนื่องจากตัวเปรียบเทียบเฟส เป็นแบบดิจิตอล ก็หมายความรวมถึงผลที่ออกมาจากตัวเทียบเฟสด้วย ระดับสัญญาณที่ออกมาก็จะเป็นบวกเท่านั้น คืออยู่ในช่วง ๐-V_{cc} ตัว VCO ของระบบที่จะใช้สัญญาณ ๐-๕ โวลท์ ดังนั้น ความถี่กลางของสัญญาณ f_o จะเกิดขึ้น เมื่อศักดาขาเข้าของ VCO เป็นครึ่งหนึ่งของ V_{cc} แทนที่จะเป็นศูนย์โวลท์

ตัวเปรียบเทียบแบบ Exclusive-OR จะมีข้อดีคือ ใช้แค่เกทตัวเดียว ก็สามารถเทียบได้

ทำได้ง่าย แต่สัญญาณที่ออกมาจากตัวเปรียบเทียบ หลังจากผ่านตัวกรองแล้ว ก็ยังคงเป็นลูกคลื่นอยู่ และจะเกิดการเลื่อนเฟสไป ๙๐ องศา เมื่อลูฟอยู่ในสภาวะล็อค





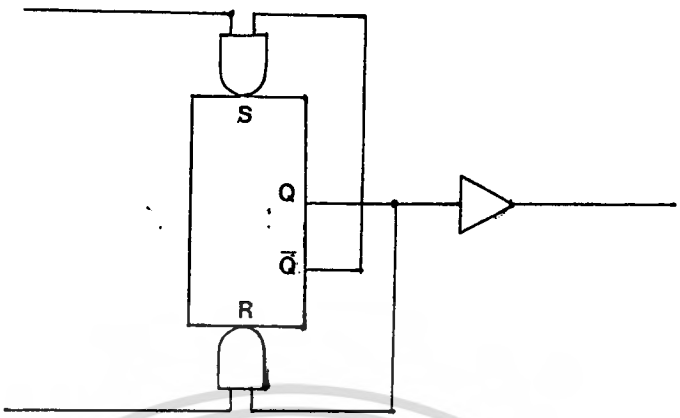
รูปที่ ๒.๑ ตัวเปรียบเทียบเฟส แบบ Exclusive OR

แบบ R-S ฟลิปฟลอป

จะใช้การต่อกันของ R-S ฟลิปฟลอป มาต่อกันในแบบไม่ทริก (trig) ตามสัญญาณของขา ขึ้นหรือลง เพราะว่าสัญญาณหนึ่ง ๆ จะสามารถมีขอบขาขึ้นหรือขอบขาลง ได้เพียงหนึ่งครั้งเท่านั้น ต่อคาบ การทำลักษณะนี้มีประโยชน์คือ ตัดการตรวจจับระดับสัญญาณออกไป ผลที่ออกมาจากตัวเปรียบเทียบ จะไม่ขึ้นกับค่า Duty Cycle ของสัญญาณขาเข้า การทำงานสามารถดูจากรูปทางด้านขาเข้า เมื่อสัญญาณที่มีของขาขึ้นอันแรก เข้ามา ถ้าสมมติให้ ขณะนั้นเอาต์พุต มีสถานะเป็น "๑" อยู่ จะทำให้สัญญาณเปลี่ยนแปลงระดับจาก "๑" ไป "๐" ไม่ว่าจะเป็นสัญญาณอ้างอิงหรือสัญญาณเปรียบเทียบ ถ้าตรวจพบว่า เป็นของขาขึ้นก็จะทำให้ ระดับสัญญาณที่ออกมาเปลี่ยนไปเป็นอีกสถานะหนึ่ง ระดับของสัญญาณหลังจากผ่านตัวกรองความถี่ต่ำผ่าน ก็จะได้ความสัมพันธ์

$$V_{out} = Vcc*(\phi_{sig}-\phi_{comp})/2\pi$$

ตัวเปรียบเทียบสัญญาณแบบนี้จะไม่จำกัด Duty Cycle ของสัญญาณขาเข้า และมีช่วงของเฟสแตกต่างอยู่ระหว่าง ๐-๒π เรเดียน

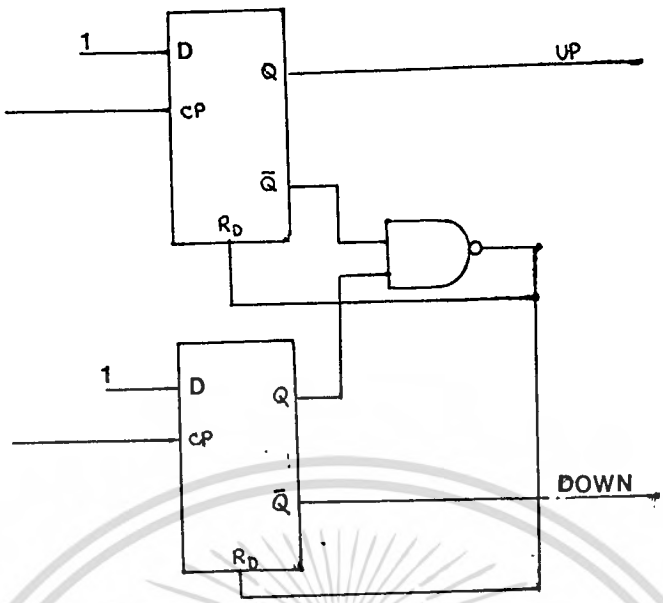


รูปที่ ๒.๒ ตัวเปรียบเทียบเฟสแบบ R-S ฟลิบฟลอป

แบบเทียบเฟสและความถี่

จะใช้สัญญาณของขาขึ้น(หรือขาลง) ในการตรวจสอบสัญญาณ ดังนั้นค่า Duty Cycle จะไม่มีผลต่อผลลัพธ์ ซึ่งจะแตกต่างกับสองแบบแรกที่กล่าวมา คือ ทางเอาต์พุตของตัวเปรียบเทียบจะมีสามสถานะ แทนที่จะเป็นแค่ "๐" กับ "๑" แต่กลับมีเพิ่มมาอีกอันหนึ่ง คือ แบบอิมพีแดนย์สูง (Hiyh Impedance) หรือเสมือนกับการเปิดวงจร การทำงานของตัวเทียบเฟสแบบนี้สามารถเขียนได้ในหลายแบบ ทั้งแบบแผนภูมิสเตท(State Diagram) หรือรูปของ D-ฟลิปฟลอป แต่ก็สามารถทำให้อยู่ในรูปเดียวกันได้(พิสูจน์ได้ว่าเป็นชนิดเดียวกัน) หลักการทำงานก็คือถ้ามีสัญญาณอ้างอิงเข้ามาขาขึ้นในขณะที่สัญญาณเปรียบเทียบเป็น "๐" อยู่ แสดงว่า สัญญาณอ้างอิงมีเฟสน่าสัญญาณเปรียบเทียบอยู่ ตัวเทียบก็จะให้ค่า "๑" ออกมา จนกว่าจะพบขอบขาขึ้น ของสัญญาณเปรียบเทียบ เมื่อมีขอบขาขึ้นมาจากสองสัญญาณ แสดงว่าตอนนี้สัญญาณขาเข้าทั้งคู่ มีสถานะเป็น "๑" ตัวเทียบเฟส ก็จะมีสภาวะเปิดวงจรเช่นกัน ถ้ามีสัญญาณเปรียบเทียบขอบขาขึ้นเข้ามาก่อน แสดงว่าสัญญาณเปรียบเทียบเป็นฝ่ายนำหรือ VCO อาจให้ความถี่หรือเฟสออกมามากเกินไป ตัวเทียบเฟสก็จะให้ "๐" ออกมาจนกว่าจะพบขอบขาขึ้นของสัญญาณอ้างอิง

ตัวเทียบเฟสแบบนี้ จะมีข้อดีคือ เมื่อลูฟอยู่ในสภาวะล็อคเฟสของสัญญาณอ้างอิง และสัญญาณเปรียบเทียบ จะเท่ากันพอดี และช่วงของการจับเฟส จะอยู่ในช่วง -๒π กับ +๒π ถ้าสัญญาณหลังจากผ่านตัวกรองความถี่ต่ำผ่านจะเป็น $V_{out} = Vcc*(\phi_{in}-\phi_{comp})/4\pi$



รูปที่ ๒.๓ ตัวเปรียบเทียบเฟสแบบเทียบเฟสและความถี่

บทที่ ๓

การอินเตอร์เฟส กับ คอมพิวเตอร์

ในปัจจุบัน คอมพิวเตอร์มีราคาถูกลงอย่างมาก มีการนำไปใช้งานกันอย่างกว้างขวาง ไม่ว่าจะเป็นในด้านที่เกี่ยวกับเอกสารหรืองานทั่ว ๆ ไป การใช้งานในด้านต่าง ๆ เหล่านี้ มักจะมีอุปกรณ์ที่ใช้ประกอบการทำงาน ซึ่งเป็นตัวเชื่อมต่อระหว่างผู้ใช้งาน และคอมพิวเตอร์ทำให้การใช้งานเป็นไปอย่างง่ายดายยิ่งขึ้น โดยมีรูปแบบการติดต่อเป็นมาตรฐานที่นิยมใช้กันอยู่ทั่วไป คือ มาตรฐานการสื่อสารแบบอนุกรม (RS-232) และมาตรฐานการสื่อสารแบบขนาน (GPIB)

การเชื่อมต่อ กับ IBM PC โดยอุปกรณ์ I/O ที่ต้องการติดต่อโดยตรงกับระบบบัสของ IBM PC ซึ่งการเชื่อมต่ออุปกรณ์โดยตรงกับระบบบัสของคอมพิวเตอร์ เป็นการเพิ่มขีดความสามารถในการติดต่อและเพิ่มประสิทธิภาพในการทำงาน

ฮาร์ดแวร์ของ IBM PC XT

AT สล็อต และ XT สล็อต

AT ใช้ ตัวประมวลผลกลาง(CPU) เบอร์ 80286 มีการประมวลผลภายในเป็น ๑๖ บิต และมีบัสข้อมูลขนาด ๑๖ บิต ในขณะที่ XT ใช้ CPUเบอร์ 8088 ซึ่งมีการประมวลผลภายในเป็นแบบ ๑๖ บิต แต่ข้อมูลมีเพียงแค่ ๘ บิต จะเห็นได้ว่าระบบบัสของ XT เป็นซับเซตของระบบบัสของ AT

บนเมนบอร์ด จะมีสล็อตอยู่ ๒ ชนิด คือ สล็อตสั้น และสล็อตยาว ซึ่งสล็อตยาวจะมีจำนวนขาสัญญาณบนสล็อตเหมือนกับ XT ส่วนขาสัญญาณที่อยู่บนสล็อตสั้น จะเป็นสัญญาณที่มีแต่เฉพาะบน AT ประกอบด้วย ข้อมูลดริ่งบน ๘ บิต แอดเดรสที่เพิ่มขึ้นมาอีก ๔ บิต และขาสัญญาณควบคุมที่เพิ่มขึ้นจาก XT

ดังนั้น การ์ดที่ใช้บน XT ทุกการ์ดสามารถนำมาใช้ได้บน AT โดยเสียบไว้บนสล็อตยาว ความหนาของแผ่นการ์ดรวมทั้งตัวอุปกรณ์จิงไม่ควรเกิน ๑๒.๗ มิลลิเมตร หรือ ๐.๕ นิ้ว

การ์ดขยายทำหน้าที่เป็นคอนเน็กเตอร์ตัวผู้ในตัว ขณะที่สล็อตเป็นคอนเน็กเตอร์ตัวเมีย เมื่อ

เสียบการ์ดขยายลงในสล็อตจะสัมผัส กับลายทองแดงบนแผ่นการ์ดทั้งสองด้าน ระยะห่างระหว่างลายทองแดงบริเวณคอนเน็กเตอร์ ให้มีระยะห่างเท่ากัน ๐.๑ นิ้ว

การควบคุมอุปกรณ์ I/O ที่ต่ออยู่กับ IBM PC จะกระทำผ่านพอร์ต โดยการอ้างถึงแอดเดรสของพอร์ตที่อุปกรณ์นั้นต่ออยู่โดยตรง ดังนั้น การที่จะใช้งานหรือควบคุมอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ต ในคอมพิวเตอร์ส่วนบุคคล พอร์ตและหน่วยความจำจะแยกจากกันโดยเด็ดขาด ถึงแม้ว่าการอ้างถึงจะใช้สัญญาณจากแอดเดรสบัสเหมือนกัน แต่สัญญาณที่ใช้ในการอีเนเบิ้ลในการอ่านและเขียนข้อมูลจะต่างกัน การจัดแอดเดรสของหน่วยความจำแสดงได้ในตารางที่ ๓.๑ และในตารางที่ ๓.๒ แสดงการจัดแอดเดรสของพอร์ต I/O

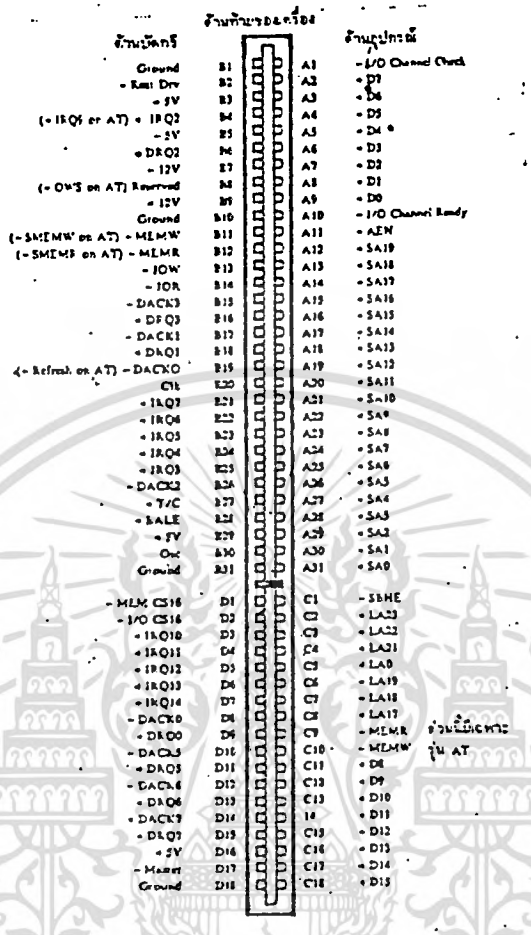
Block 0	00000-0FFFF	RAM to 64K
Block 1	10000-1FFFF	RAM to 128K
Block 2	20000-2FFFF	RAM to 192K
Block 3	30000-3FFFF	RAM to 256K
Block 4	40000-4FFFF	RAM to 320K
Block 5	50000-5FFFF	RAM to 384K
Block 6	60000-6FFFF	RAM to 448K
Block 7	70000-7FFFF	RAM to 512K
Block 8	80000-8FFFF	RAM to 576K
Block 9	90000-9FFFF	RAM to 640K
Block A	A0000-AFFFF	Extended video memory
Block B	B0000-BFFFF	Standard video memory
Block C	C0000-CFFFF	BIOS extension (eg EGA)
Block D	D0000-DFFFF	Other use
Block E	E0000-EFFFF	Other use
Block F	F0000-FFFFF	BIOS EPROM

ตารางที่ ๓.๑ การจัดแอดเดรสของหน่วยความจำบนแรม

การอ้างแอดเดรสของหน่วยความจำใช้แอดเดรสทั้งหมด ๒๐ เส้น คือ A0-A19 ในรุ่น XT ในรุ่น AT ใช้แอดเดรส ๒๔ เส้นคือ A0-A19 และ A20-A23 การอ้างแอดเดรสของพอร์ต ใช้แอดเดรสเพียง ๑๐ เส้น คือ A0-A9 ดังนั้นจำนวนพอร์ตสูงสุดที่สามารถอ้างได้ คือ ๑๐๒๔ พอร์ต และยังแบ่งออกเป็น ๒ กลุ่ม คือกลุ่มพอร์ตที่มีแอดเดรสอยู่ในช่วง 000H-0FFH จะใช้งานบนเมนบอร์ด สำหรับชิพซัพพอร์ตเท่านั้น และกลุ่มที่มีแอดเดรสอยู่ในช่วง 100H-3FFH จะใช้งานกับการ์ดขยายต่าง ๆ ที่เสียบในสล็อต จากตารางที่ ๓.๒ จะเห็นได้ว่าแอดเดรสของพอร์ต ถูกแบ่งออกเป็นช่วงย่อย ๆ ไว้ให้ใช้กับอุปกรณ์เฉพาะอย่าง ถ้าในระบบ ไม่ได้ใช้งานอุปกรณ์นั้น เราสามารถนำแอดเดรสของพอร์ตในช่วงนั้นมาใช้งานได้ แต่อย่างไรก็ตาม การเรียกใช้งานพอร์ตที่ไม่ได้ถูกกำหนดให้ใช้กับอุปกรณ์อื่น จะดีกว่า ทำให้อุปกรณ์ที่ใช้ผ่านพอร์ตนี้ สามารถใช้ได้กว้างขวางยิ่งขึ้น

หมายเลขพอร์ตฐานฐานสิบหก	ชื่ออุปกรณ์
000-01F	ดีเอ็มเอคอนโทรลเลอร์หมายเลข 1, 8237A-5
020-03F	อินเตอร์รัพต์คอนโทรลเลอร์หมายเล 1, 8259A ตัวหลัก
040-05F	ไทม์เมอร์ 8254-2
060-06F	8042 คีย์บอร์ด
070-07F	นาฬิกา และ NMI และซีมอส RAM
080-09F	DMA เพจรีจิสเตอร์
0A0-0BF	อินเตอร์รัพต์คอนโทรลเลอร์หมายเลข 2, 8259A
0C0-0DF	ดีเอ็มเอคอนโทรลเลอร์หมายเลข 2, 8237A-5
0F0	เกลียร์โปรเซสเซอร์คณิตศาสตร์
0F1	รีเซตโปรเซสเซอร์คณิตศาสตร์
0F8-0FF	โปรเซสเซอร์คณิตศาสตร์
1F0-1F8	ฮาร์ดดิสก์
200-207	เกมไอโอ
278-27F	พอร์ตเครื่องพิมพ์หมายเลข 2
2F8-2FF	พอร์ตอนุกรมหมายเลข 2
300-31F	โปรโตไทป์การ์ด
360-36F	สำรอง
378-37F	พอร์ตเครื่องพิมพ์หมายเลข 1
380-38F	SDLC, ไบซิงก์ 2
3A0-3AF	ไบซิงก์ 1
3B0-3BF	โมโนโครมและเครื่องพิมพ์
3C0-3CF	สำรอง
3D0-3DF	จอภาพสี
3F0-3F7	ควบคุมดิสเกตต์
3F8-3FF	พอร์ตอนุกรมหมายเลข 1

ตารางที่ ๓.๒ หมายเลขพอร์ตที่ใช้กับ IBM PC



รูปที่ ๓.๑ แสดงตำแหน่งของสัญญาณต่าง ๆ บนสล็อต

รายละเอียดของสัญญาณต่าง ๆ บนสล็อต

(I), (O) และ (I/O) หมายถึง ทิศทางของขาสัญญาณเมื่อเทียบกับเมนบอร์ด

โดยที่ (I) หมายถึง ขาสัญญาณอินพุต

(O) หมายถึง ขาสัญญาณเอาต์พุต

(I/O) หมายถึง ขาสัญญาณที่เป็นได้ทั้งอินพุตและเอาต์พุต

(*I/O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณเอาต์พุต แต่จะเป็นอินพุตในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้าจะหมายถึงขาสัญญาณที่แอคตีฟที่ลอจิก "๐" และขาสัญญาณที่ไม่มี หรือ มีเครื่องหมายบวกนำอยู่หน้าจะหมายถึง ขาสัญญาณที่แอคตีฟที่ลอจิก "๑" ขา

สัญญาณต่าง ๆ บนสล็อตของ XT และ AT สามารถแบ่งออกเป็นกลุ่ม ๆ ได้ดังนี้

เพาว์เวอร์ซัพพลาย

Ground	ต่อลู่กับกราวด์ของระบบเรกูเลเตอร์
+ ๕ V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC + ๕ โวลท์
- ๕ V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC - ๕ โวลท์
+ ๑๒ V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC + ๑๒ โวลท์
- ๑๒ V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC - ๑๒ โวลท์

แอดเดรสบัส และสัญญาณต่าง ๆ ที่เกี่ยวข้อง

SA0-SA19 (I)	เป็นแอดเดรสบิตที่ ๐ ถึง ๑๙ โดยที่ SA0 มีนัยสำคัญที่ต่ำสุด ขาสัญญาณนี้จะแอดดีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็น "๑" และจะถูกแลตช์ไว้ตอนขอบขาลงของขาสัญญาณ BALE แอดเดรสทั้ง ๒๐ บิตนี้
LA17-LA23 (*I/O)	(เฉพาะรุ่น AT) ขาสัญญาณนี้จะแอพดีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก "๑" แต่จะไม่มีการแลตช์ไว้ ตอนขอบขาลงของขาสัญญาณ BALE ถ้า I/O ไม่มีการอ้างแอดเดรสเกิด ๑ เมกกะไบต์ ขาสัญญาณนี้ก็ไม่จำเป็นต้องใช้ ถ้ามีอ้างแอดเดรสเกิน ต้องทำการแลตช์สัญญาณนี้ โดยใช้ขอบขาลงของขาสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR
AEN (O)	ขาสัญญาณนี้จะแอดดีฟเมื่อควบคุม DMA ได้ทำการควบคุมบัสต่างๆ ของระบบแล้ว ดังนั้น การอ้างพอร์ตของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ในการดีโค้ด เพื่อจะไม่ทำให้ เกิดการติดต่อระหว่างระบบกับอุปกรณ์ I/O ตัวอื่น ยกเว้นตัวที่กำลังทำขบวนการ DMA อยู่
BALE	ขาสัญญาณนี้ใช้ใน การแสดงการเริ่มต้นของขบวนการต่าง ๆ ที่มี

(O) การติดต่อกับหน่วยความจำโดยจะแอคตีฟเมื่อค่าแอดเดรสที่ CPU ต้องการติดต่อด้วย อยู่บนแอดเดรสบัสเรียบร้อยแล้ว ปกติขอบขาลงของสัญญาณ ทำให้เกิดการแลตช์สัญญาณ SA0-SA19 ถ้ามีการอ้างแอดเดรสเกิด ๑ เมกกะไบต์ใน AT จะใช้ขอบขาลงแลตช์สัญญาณ LA17-LA23 ด้วยเช่นกัน แต่สำหรับการ DMA สัญญาณนี้จะมีสถานะเป็น "๑" ตลอด

SBHE (*I/O) เป็นขา สัญญาณที่ใช้แสดงรับส่งข้อมูลในบิตที่ SD8-SD15

บัสข้อมูล

SD0-SD7 (I/O) บิตข้อมูล ๐ ถึง ๗ สำหรับรุ่น XT และสำหรับรุ่น AT คือ บิตข้อมูล ๐ ถึง ๑๕

สัญญาณอินเตอร์รัพต์

IRQ2-IRQ7 (I) สำหรับ AT ลำดับความสำคัญของสัญญาณ IRQ เป็นดังนี้ คือ 9, 10,11,12,14,15,3,4,5,6 และ7 โดย IRQ9 มีลำดับความสำคัญมากที่สุดและ IRQ7 มีลำดับความสำคัญน้อยที่สุด สำหรับ XT IRQ2 จะมีลำดับความสำคัญมากที่สุด รอง ๆ ลงไป คือ IRQ3,4,5,6,7

โดยปกติสัญญาณนี้จะมีสถานะเป็น "๐" เสมอถ้าต้องการอินเตอร์รัพต์ CPU ให้ส่งพัลส์ที่เป็นลอจิก "๑" ให้กับมัน การตรวจสอบสัญญาณอินเตอร์รัพต์จะใช้ขอบขาลงของสัญญาณนี้

I/O CH CK (I) เป็นขาสัญญาณที่บอกถึงความผิดพลาดในการรับส่งข้อมูล ซึ่งตรวจสอบจากพาริตี้บิต สัญญานี้ทำให้เกิดการอินเตอร์รัพต์แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิด Parity Error

สัญญาณที่ใช้ในขบวนการ DMA

DRQ1-DRQ3 (I)	เป็นขาสัญญาณใช้ใน DMA โดยที่ DRQ0 มีลำดับความสำคัญมากที่สุด และDRQ3 มีลำดับความสำคัญน้อยที่สุด บน XT DRQ0 ใช้สำหรับการรีเฟรช หน่วยความจำแบบไดนามิค จึงไม่มีขาสัญญาณ DRQ0 ต่อออกมาที่สล็อต การขอทำ DMA ทำได้โดยทำให้เขาสัญญาณนี้ มีสถานะเป็น "๑" แล้วรอจนกระทั่งได้รับตอบสนองการทำ DMA จาก CPU โดยการตรวจสอบสัญญาณ DAKC ที่ส่งออกมา
DACK0-3 (I)	เป็นสัญญาณตอบสนองการขอทำ DMA ถ้ามีการขอทำ DMA ผ่านทาง DRQ2 CPU รับรู้แล้ว จะทำให้สัญญาณ DACK2 แอคตีฟ บน XT สัญญาณ DACK0 ก็จะถูกต่อออกมาที่สล็อตด้วยเพื่อแสดงถึงขบวนการรีเฟรชไดนามิคแรม สามารถนำสัญญาณนี้ไปใช้ในการรีเฟรชหน่วยความจำแบบไดนามิคกับอุปกรณ์ I/O ได้
Refresh (O)	(เฉพาะรุ่น AT) ใช้แสดงขบวนการรีเฟรชหน่วยความจำ
Master (I)	(เฉพาะรุ่น AT) ขาสัญญาณนี้ใช้ร่วมกับ DMA Request ในการเข้าควบคุมระบบบัส ในขบวนการ DMA เวลาที่สัญญาณนี้แอคตีฟไม่ควรเกิน ๑๕ ไมโครวินาที มิฉะนั้นข้อมูลภายในหน่วยความจำจะสูญหายไปได้ เนื่องจากขาดสัญญาณรีเฟรชหน่วยความจำ
T/C (O)	เป็นขาสัญญาณที่บอกให้ทราบว่า จำนวนข้อมูลที่รับส่งในขบวนการ DMA นี้ครบแล้ว

สัญญาณที่ใช้สร้าง Wait States

I/O CH RDY (I)	ขาสัญญาณจะถูกแอคตีฟ เมื่ออุปกรณ์ I/O หรือหน่วยความจำที่ไม่สามารถทำงานได้ทันกับระบบ จะต้องเพิ่ม Wait States ให้สัญญาณนี้แอคตีฟในช่วงเวลาที่ I/O ได้รับสัญญาณจากการดีโค้ด

แอดเดรส, สัญญาณ MEMR, MEMW, IOR, IOW

OWS (I) (เฉพาะรุ่น AT) สัญญาณนี้จะบังคับไม่ให้เกิดการสร้าง Wait States ทำได้โดยการสร้างสัญญาณจากการดีโค้ดแอดเดรส และสัญญาณที่ใช้ในการอ่านหรือเขียนแอคตีฟ หลังจากสัญญาณอ่าน หรือเขียนไปแล้ว ๑ คล็อก

สัญญาณนาฬิกา

CLK (O) ขาสัญญาณนี้ จะมีความถี่ประมาณ ๔.๗๗ MHz หรือสูงกว่านี้ได้สำหรับโมเดลใหม่ และสำหรับ AT จะมีความถี่ประมาณ ๖ MHz

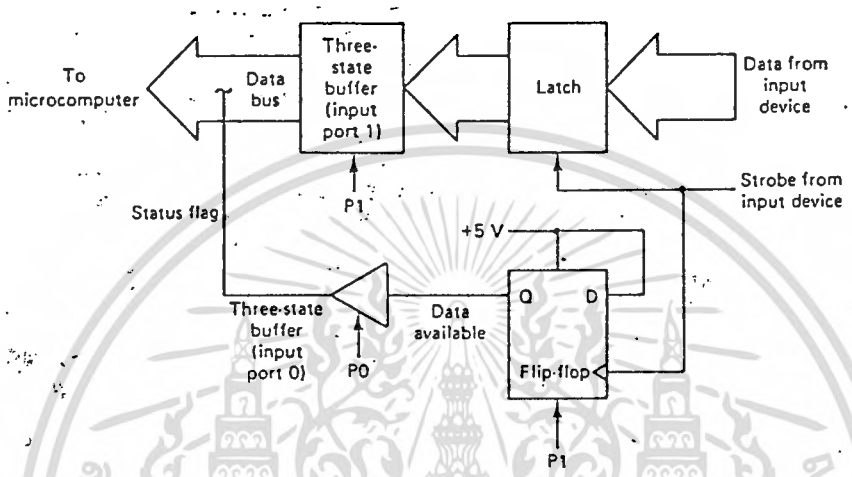
OSC (O) เป็นขาสัญญาณที่มีความถี่สูง ๑๔.๓๑๘๑๘ MHz และไม่ซิงโครนัสกับสัญญาณอื่น ๆ ในระบบ

การโปรแกรมรับส่งข้อมูล

ไมโครโปรเซสเซอร์ 8088 ใช้บัสข้อมูล ขนาด ๘ บิต ใช้แอดเดรสบัส ๒๐ เส้น อ้างหน่วยความจำได้ ๑ เมกะไบต์ การอ่านและเขียนข้อมูล จากหน่วยความจำ จะอ่านทีละ ๘ บิตแต่จะประมวลผลภายในทีละ ๑๖ บิต ทำงานที่ ๔.๗๗ เมกะเฮิรตซ์ โดยมี ไอซี สร้างสัญญาณนาฬิกาจากคริสตอล ๑๔.๓๑๘๑๘ เมกะเฮิรตซ์ โดยนำสัญญาณในการนี้มาหาร ๓ ให้ได้ ๔.๗๗ เพื่อส่งต่อให้ตัวประเมินผลกลาง การอ้างแอดเดรสใด ๆ ต้องใช้รีจิสเตอร์ ๒ ตัว ตัวหนึ่งบอกแอดเดรสในเซกเมนต์ อีกตัวบอกตำแหน่งเริ่มต้นของเซกเมนต์ จึงสามารถอ้างได้ครบ ๒๐ บิต

การเชื่อมต่อกับคอมพิวเตอร์มีทั้งฮาร์ดแวร์และซอฟท์แวร์ ในส่วนของซอฟท์แวร์จะต้องมีการเขียนโปรแกรม ซึ่งควบคุมการนำข้อมูลเข้าและออกจากคอมพิวเตอร์ โดยพอร์ตสามารถติดต่อกับตัวประเมิลผลกลาง โดยผ่านการอ้างแอดเดรส เรียกวิธีนี้ว่า Memory-Maped การนำเข้าและออกจะใช้คำสั่ง IN และ OUT ของ 8088 รูปที่ ๓.๒ แสดงการติดต่อพื้นฐาน โดยส่วน De-code Logic จะสร้างสัญญาณ Chip-select pulse จากแอดเดรสบัสและสัญญาณ IOR, IOW สัญญาณ Chip-select pulse สำหรับนำข้อมูลผ่านพอร์ตจะส่งไปที่อุปกรณ์รับข้อมูล ส่วนที่ใช้ส่ง

ข้อมูลจะออกทางอุปกรณ์ส่งข้อมูล โดยอุปกรณ์รับข้อมูล จะมีบัฟเฟอร์สามสถานะ เพื่อแยกข้อมูลกับบัสข้อมูล ส่วนอุปกรณ์ส่งข้อมูลจะแลทช์ข้อมูลไว้ดิพสวิทช์และดวงไฟเป็นตัวแสดงผลของการทดลองได้



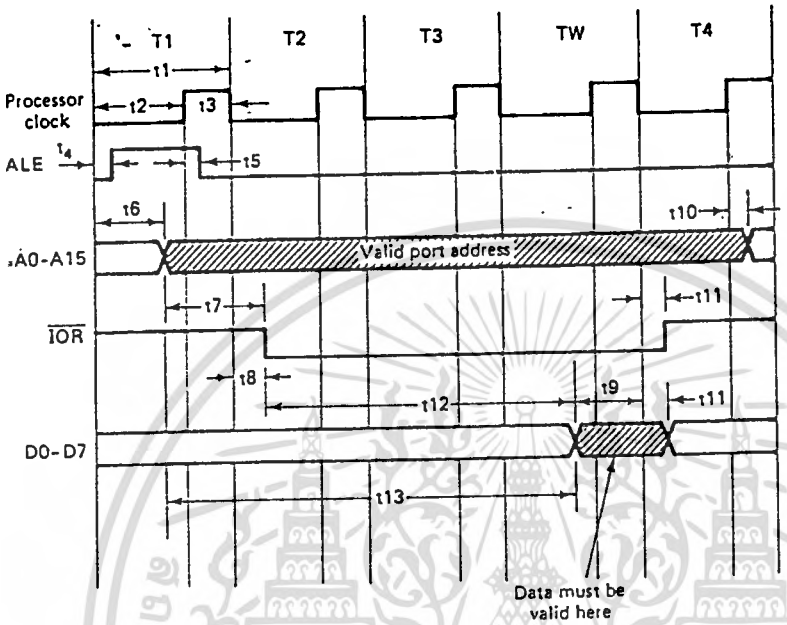
รูปที่ ๓.๒ Block Diagram of bask I/O operation

กุญแจสำคัญในการอินเตอร์เฟสคือ Timing Diagram ของคำสั่ง IN และ OUT เนื่องจาก ถ้ามี การส่งข้อมูลเข้าและออกเร็วกว่าพอร์ตจะรับได้ ข้อมูลบางส่วนจะหายไป รูปที่ ๓.๓ แสดงแผนภาพเวลาในการอ่านและเขียนบนบัสข้อมูล จากรูปจะเห็นว่าหนึ่งไซเคิลมีสี่คล็อค โดยคอมพิวเตอร์ส่วนบุคคลออกแบบให้มี TW(T-Wait) เพิ่มขึ้นมา ดังนั้นในไซเคิลของการรับส่งข้อมูลจะใช้เวลาอย่างน้อย ๕ ไซเคิลหรือประมาณ ๑.๐๕ ไมโครวินาที

การส่งข้อมูลเข้าและออก ถูกควบคุมโดยสัญญาณ IO CH RDY โดยขา A16-A19 ของแอดเดรสบัสจะไม่ทำงาน Timing Diagram จะมีการทำงานดังนี้

-สัญญาณอ่าน I/O จะเกิดเมื่อมีคำสั่ง IN ช่วง T1 สัญญาณ ALE จะแอคตีฟ ช่วง A0-A15 มีค่าตรงกับพอร์ตที่ต้องการ ในขอบขาลงของสัญญาณนี้ ช่วง T2 สัญญาณ IOR จะแอคตีฟ ทำให้รับข้อมูลผ่านบัสข้อมูลได้ และช่วง T4 CPU จะรับข้อมูลจากบัสข้อมูล ทำให้ IOW ไม่แอคตีฟ

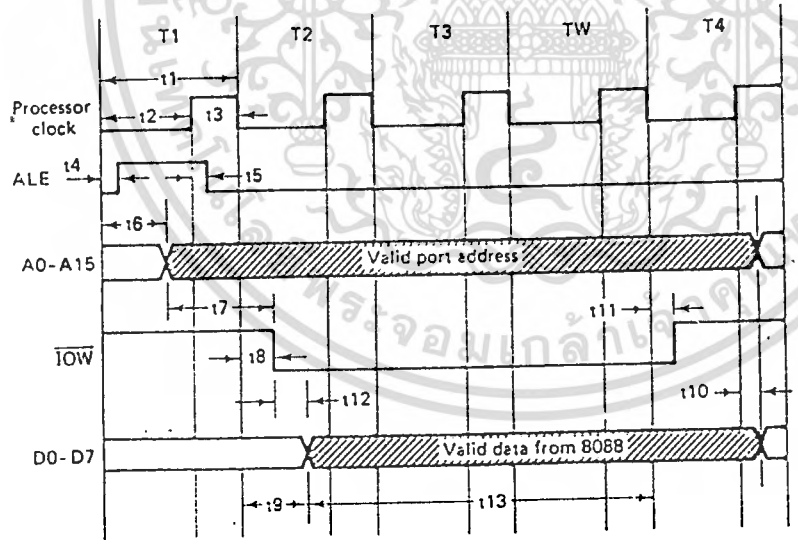
-สัญญาณเขียน I/O จะเกิดเมื่อมีคำสั่ง OUT เมื่อเกิดสัญญาณ IOW ช่วงใน T2 ทำให้ข้อมูลถูกส่งออกจากบัสข้อมูล และในช่วงต้นของ T4 จะไม่แอคตีฟ CPU จะปลดข้อมูลออกจากบัส



(a)

Symbol	Max.	Min.
t1	–	209.5
t2	–	124.5
t3	–	71.8
t4	15	–
t5	15	–
t6	128	16
t7	–	91.5
t8	35	10
t9	–	42
t10	35	10
t11	–	10
t12	–	551.5
t13	–	668

*All times are in nanoseconds



(b)

Symbol	Max.	Min.
t1	–	209.5
t2	–	124.5
t3	–	71.8
t4	15	–
t5	15	–
t6	128	16
t7	–	91.5
t8	35	10
t9	122	14
t10	–	10
t11	35	10
t12	112	–
t13	–	506.5

*All times are in nanoseconds

รูปที่ ๓.๓ I/O Port Timing

ทั้งหมดที่กล่าวมา เป็นตัวอย่างให้เห็นถึงเงื่อนไขในการเชื่อมต่อและข้อจำกัดต่าง ๆ ที่ต้องคำนึงถึง

บทที่ ๔

การสร้างสัญญาณเวลามาตรฐาน

" จะใช้คอมพิวเตอร์ส่วนบุคคล จับเวลาที่มีความละเอียดเป็นมิลลิวินาที "

" ต้องการใช้คอมพิวเตอร์ส่วนบุคคลเป็น DIGITAL OSCILLOSCOPE ที่มีการ SAMPLING ที่แน่นอน"

นี่เป็นตัวอย่างที่ต้องการใช้ประโยชน์จากคอมพิวเตอร์ส่วนบุคคล ให้มากที่สุดเท่าที่ระบบจะมีสนับสนุนให้ ปัญหาแรกคือ ระบบที่มีอยู่สามารถทำได้ดังปัญหาข้างบนหรือไม่ เริ่มที่ระบบ DOS(OPERATING SYSTEM) มีระบบเกี่ยวกับเวลาที่มีในการบริการคือ บริการหมายเลข 2DH สำหรับตั้งเวลาของระบบและบริการหมายเลข 2CH สำหรับอ่านเวลาของระบบ ในภาษาของการเขียนโปรแกรมเช่น TURBO PASCAL หรือ C หรือ BASIC ก็มีคำสั่งสำหรับการอ่านเวลาด้วย ดังนั้นถ้าอ่านเวลาเริ่มต้นเก็บเอาไว้ ถึงจุดสุดท้ายก็อ่านเวลาออกมานำเอาเวลาเริ่มต้นมาลบออก ก็จะได้เวลาที่แตกต่างของจุดสองจุดในเวลาแล้ว อย่ารีบด่วนสรุป

ถ้าดูให้ดีจะพบว่าดอสให้เพียงแต่นาฬิกาบอกเวลามาให้เท่านั้น ไม่ได้ให้นาฬิกาจับเวลา ดูบริการหมายเลข 2CH ของดอส จะให้ค่าที่ได้จากการบริการ คือ

CL เวลาเป็น นาที

CH เวลาเป็น ชั่วโมง

DL เวลาเป็น ๑/๑๐๐ ของวินาที

DH เวลาเป็น วินาที

จะเห็นได้ว่า ความละเอียดที่อ่านได้ แค่ ๑/๑๐๐ วินาทีเท่านั้น และ ๑/๑๐๐ ที่อ่านได้ ก็ใช้จะมีความถูกต้อง ถึง ๑/๑๐๐ วินาที ดังนั้นเรามีนาฬิกาที่มีความถูกต้องแค่เป็นวินาทีเท่านั้น ทำไมถึงเป็นเช่นนั้น มาดูระบบเวลาของคอมพิวเตอร์ส่วนบุคคลกัน

ฮาร์ดแวร์ของคอมพิวเตอร์ส่วนบุคคล มีตัวนับอยู่ตัวหนึ่งซึ่งส่งสัญญาณขัดจังหวะ ๑๘.๒ ครั้งต่อวินาที การขัดจังหวะทางฮาร์ดแวร์ที่เกิดจากตัวนับนี้ มีลำดับความสำคัญสูงสุดในบรรดาการขัดจังหวะทางฮาร์ดแวร์คือ เป็นการขัดจังหวะหมายเลข ๘ ของ CPU,BIOS จะมีพื้นที่ในหน่วยความจำจำนวน ๓ ไบต์ อยู่ที่ตำแหน่ง

0040: 6CH BYTE ที่มีค่าต่ำสุด

0040: 6DH BYTE ที่มีค่ารอง

0040: 6EH BYTE ที่มีค่าสูงสุด

บริการอ่านเวลาของดอสจึงเป็นแค่เพียงอ่านผลงานของไบออสที่เกิดขึ้น โดยการขัดจังหวะหมายเลข ๘ ที่บันทึกเอาไว้ในตำแหน่งดังกล่าว มาแปลงเป็นเวลาของระบบ ดังนั้นความละเอียดของเวลา มีความละเอียดมากกว่า ๑/๑๘.๒ วินาที ไม่ได้แน่นอน ดังนั้น ภาษาที่เรียกบริการของดอส ไม่ว่าจะเป็น PASCAL หรือ C หรือ BASIC จึงไม่สามารถที่จะจับเวลาได้ละเอียดเป็นมิลลิวินาทีแน่นอน

โครงสร้างของตัวจับเวลา (Timer Counter)

ไมโครคอมพิวเตอร์ในตระกูลของ IBM PC และ XT ใช้ IC 8253-5 เป็นตัวนับจับเวลา รุ่น AT ใช้ 8255-2 PS/2 รุ่น ๒๕ และ ๓๐ ก็ใช้ 8253-5 เหมือนกัน แต่ไม่ว่ารุ่นไหน ก็จะมีตัวนับจับเวลาอย่างน้อย ๓ ตัว คือ

- ตัวนับจับเวลาตัวที่ ๑ ใช้สร้างเวลาของระบบส่งสัญญาณการขัดจังหวะ ๑๘.๒ ครั้งใน ๑ วินาที
- ตัวนับจับเวลาตัวที่ ๒ ใช้รีเฟรสหน่วยความจำ DYNAMIC โดยเฉพาะ
- ตัวนับจับเวลาตัวที่ ๓ ใช้สำหรับควบคุมความถี่เสียงที่จะส่งออกที่ลำโพง

ตัวนับจับเวลาตัวที่ ๔ มีเฉพาะใน AT และ PS/2 รุ่น ๖๐ และ ๗๐ เท่านั้น เป็นสัญญาณให้เกิดการขัดจังหวะชนิด NONMASKABLE สำหรับการเฝ้าดู (WATCHDOG) เหตุการณ์

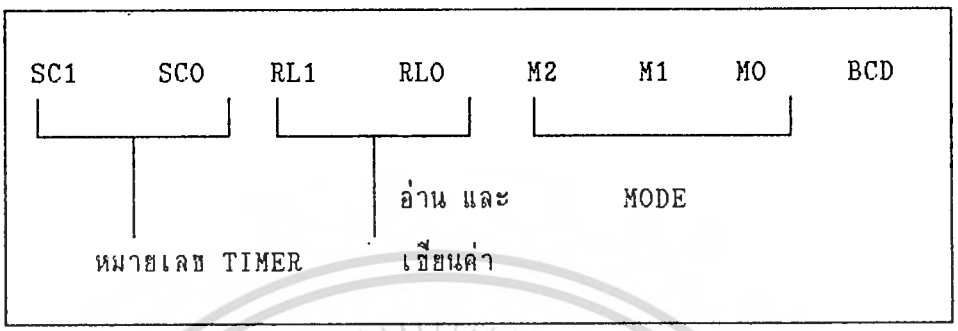
ตัวนับจับเวลาทั้งสามตัวที่กล่าวข้างบน พอจะนำมาใช้งานได้มี ๒ ตัว คือ ตัวที่ ๑ และตัวที่ ๓ ตัวนับจับเวลาทุกตัวที่สามารถโปรแกรมได้หลายโหมด หมายเลขพอร์ตของทั้ง ๓ ตัวนับจับเวลามีดังนี้

พอร์ต #40H สำหรับเขียนและอ่านค่าของ TIMER COUNTER ๑

พอร์ต #41H สำหรับเขียนและอ่านค่าของ TIMER COUNTER ๒

พอร์ต #42H สำหรับเขียนและอ่านค่าของ TIMER COUNTER ๓

พอร์ต #43H สำหรับโปรแกรม TIMER ทั้ง ๓ ตัว การตั้งค่าเริ่มต้น ทำได้โดย การเขียนคำสั่งไปที่ พอร์ต #43H ซึ่งมีรูปแบบของคำสั่งดังรูปที่ ๔.๑



รูปที่ ๔.๑ แสดงหน้าที่ของแต่ละบิทในพอร์ตเบอร์ 43H

การกำหนดหมายเลขของ TIMER โดย SCI และ SCO ดังรูปที่ ๔.๒

SC1	SCO	
๐	๐	ตัวนับจับเวลาตัวที่ ๑
๐	๑	ตัวนับจับเวลาตัวที่ ๒
๑	๐	ตัวนับจับเวลาตัวที่ ๓
๑	๑	ใช้ไม่ได้

รูปที่ ๔.๒ แสดงการเซ็ทบิทเพื่อเลือกตัวนับจับเวลา

การเขียนและอ่าน จะขอดูเฉพาะโหมด ๓ ดังนี้

ตัวนับจะเริ่มนับถอยหลัง หลังจากตัวนับถูกโหลดหรือเขียนด้วยค่าที่จะนับลงไป จนกระทั่งลดลงไปถึงศูนย์ สัญญาณที่ออกจากตัวนับยังคงค่าเนินต่อไปเรื่อย ๆ เอาท์พุต จะลดลงเป็น ๐ อีกครั้งเมื่อนับไปถึงครึ่งทาง หรือเมื่อมีการเขียนค่าลงในตัวนับใหม่ โดยที่การเขียนไบต์แรกจะหยุดการนับ การเขียนไบต์ที่สองจะเริ่มนับถอยหลังใหม่

ตัวนับตัวที่ ๑ ไบออสเริ่มต้นตั้งค่าให้ทำงานที่ MODE 0 และการอ่านเขียน เป็นแบบไบต์ที่มีค่าต่ำสุดก่อน โดยการเขียนไป พอร์ต #43H ด้วยค่า 30H

การขัดจังหวะทางฮาร์ดแวร์ IRQ0 ซึ่งอยู่ในตารางการขัดจังหวะหมายเลข ๘ จะเกิดขึ้น

ทุกครั้งที่ตัวนับจับเวลาตัวที่ ๑ นับถอยหลังไปจนถึงศูนย์ ไบออสปล่อยให้นับไปเรื่อย ๆ โดยไม่มี

การเขียนค่าใหม่ลงไป ฉะนั้นจึงเท่ากับว่าเป็นการนับ FFFFH หรือ ๖๕๕๓๕ ครั้ง จึงส่งสัญญาณไปขัดจังหวะหนึ่งครั้ง จะพบว่าการขัดจังหวะแต่ละครั้งห่างกันเท่ากับ ๖๕๕๓๕/(๑.๑๙๓๕*๑๐^{6})วินาที หรือ ๕๔.๙ มิลลิวินาทีหรือประมาณ ๑๘.๒ ครั้งต่อวินาที

การขัดจังหวะหมายเลข ๘ นั้น ควบคุมการหมุนของมอเตอร์บนดิสเกตต์ด้วย นอกเหนือจากเปลี่ยนแปลงค่าในหน่วยความจำที่ตำแหน่ง 0040:6CH,0040:6DH และ 0040:6EH ไบออสยอมให้ผู้ใช้ ใช้ประโยชน์จากการขัดจังหวะ โดยการขัดจังหวะหมายเลข ๘ จะเรียกการขัดจังหวะหมายเลข 1CH ต่อเมื่อทำงานในหน้าที่หมดแล้ว จะมองเห็นแล้วว่า เรามีตัวนับที่สามารถนับได้ละเอียดมากกว่าไมโครวินาที โดย ๑ พัลส์ของการนับเทียบเท่ากับ ๑/๑.๑๙๓ ไมโครวินาที

การนำเอาไมโครคอมพิวเตอร์ไปใช้งานทางด้านควบคุมหรือทางด้านงานวัด ต้องการความละเอียดในการสุ่มตัวอย่างสูง จำเป็นต้องมีช่วงเวลาของการสุ่มตัวอย่างที่ถูกต้องแม่นยำ เพื่อนำข้อมูลที่สุ่มมาวิเคราะห์ในเชิงเวลาได้ หากเราใช้คอมพิวเตอร์ทำหน้าที่สุ่มตัวอย่างและวิเคราะห์ด้วย

หนทางที่หนึ่งก็คือ การเขียนโปรแกรมมวนรอบรอเวลาให้ได้ตามที่ต้องการ วิธีได้ช่วงเวลาที่ไม่แน่นอน เพราะเราไม่ทราบว่าคำสั่งที่เราเขียนในโปรแกรมระดับสูงแต่ละคำสั่ง ใช้เวลาเท่าไร หรือแม้แต่ในภาษาแอสเซมบลี้เอง เนื่องจากความเร็วนาฬิกาของแต่ละเครื่อง อาจจะไม่เหมือนกัน ผลที่ได้จึงเป็นผลงานที่ขึ้นอยู่กับเครื่อง

หนทางที่สองก็คือ ใช้ประโยชน์จากเวลาที่อ่านได้จากตัวนับจับเวลาหรือ ผลงานของไบออสใช้ได้สำหรับ ช่วงเวลาของการสุ่มตัวอย่างไม่เร็วนัก เนื่องจาก เครื่องจะต้องเสียเวลาไปอ่านเปลี่ยนแปลงค่าและเปรียบเทียบก่อนจะตัดสินใจ โอกาสผิดพลาดมีมาก ถ้าช่วงเวลาชักตัวอย่างน้อยกว่าเสี้ยววินาที

หนทางที่สาม ใช้การขัดจังหวะตามเวลาซึ่งจะสะดวกไม่ต้องไปพะวงเรื่องความเร็วนาฬิกาของเครื่อง หนทางที่สามนั้น จะเป็นหนทางที่จะกล่าวโดยละเอียด

หนทางที่สี่ จะใช้ RTC (Real Time Clock) ซึ่งสามารถตั้งค่าโหมดให้ส่งการขัดจังหวะในลักษณะฮาร์ดแวร์อินเทอร์รัพ และตั้งค่า แต่วิธีนี้จะใช้ IBM PC XT ไม่ได้ แต่ให้ความเร็วและเวลาที่ละเอียดกว่าหนทางที่สาม

การขัดจังหวะหมายเลข ๘ และ การขัดจังหวะ ICH เกิดขึ้นประมาณ ๑๘.๒ ครั้งใน ๑ วินาที ถ้าช่วงเวลาของการสุ่มมากกว่า ๑/๑๘.๒ วินาที ใช้การขัดจังหวะ ๑CHได้ทันทีโดยไม่ต้อง

กำหนดได้ จะพบการนับถอยหลังของตัวนับจาก OFFFFH ลงจนเป็นศูนย์ ซึ่งใช้เวลา เท่ากับ ๑/๑๘.๒ วินาทีพอดี ถ้าเราจะลดเวลาของการเกิดการขัดจังหวะ ให้น้อยลงเราก็เปลี่ยนเลขที่จะนับถอยหลังลง การนับ ๑ ครั้ง ใช้เวลา ๑/๑.๑๘๓๑๘ ไมโครวินาที หรือ ๑๑๘๓.๑๘๓ พัลซ์ต่อ ๑ มิลลิวินาที ดังนั้น ถ้าต้องการนับเป็น ๑ มิลลิวินาที ก็โหลดค่า ๑๑๘๓ ให้กับตัวนับ

ถ้าเราเปลี่ยนช่วงเวลาการขัดจังหวะ เวลาของระบบจะผิดไป เพราะเวลาของระบบพึ่งการขัดจังหวะหมายเลข ๘ นอกจากนี้ ไบออสยังใช้ไปควบคุมการหมุนของมอเตอร์ของดิสเกตต์อีก ดังนั้นการเปลี่ยนแปลงการขัดจังหวะตามเวลา จะต้องระมัดระวังให้ดี ไม่เช่นนั้นการทำงานของเครื่องอาจจะหลงทางหายไป

การสร้างสัญญาณนาฬิกาอ้างอิง

จากการที่เราได้เลือกใช้ตัวนับจับเวลาของเครื่อง ซึ่งเป็นไอซีเบอร์ 8253 ลักษณะการนำมาใช้งานก็คือ ตั้งเวลาไอซี 8253 พอถึงค่าเวลาที่ตั้งไว้ก็จะให้ส่วนของโปรแกรมที่สร้างไว้ทำงาน เมื่อมาพิจารณาตัวนับจับเวลาที่ใช้ได้

ตัวนับจับเวลาตัวที่ ๓ สามารถตั้งให้เกิดค่าความถี่ต่างๆ แล้วส่งผ่านเกทไปออกลำโพงของเครื่อง ฮาเอาท์พุทจากตัวนี้จะมีวงจรโดยเฉพาะไม่สามารถเปลี่ยนแปลงได้ ตัวนับจับเวลาตัวนี้ จะนำมาใช้เฉพาะสำหรับการสร้างเสียงดนตรี เสียงจะออกมาได้ก็ต่อเมื่อตั้งค่าความถี่ให้กับตัวนับจับเวลาตัวที่ ๓ ผ่านทางพอร์ทเบอร์ 42H แต่ความถี่ดังกล่าวก็จะยังคงไม่ออกมาเป็นเสียง โดยตรงต้องสั่งให้วงจรเชื่อมต่อทำงานด้วย คล้ายเป็นสวิทต์ตัดต่อระหว่างลำโพงกับตัวนับจับเวลา ดังนั้นตัวนับจับเวลาตัวนี้ จึงไม่สามารถนำมาใช้ตามลักษณะที่ต้องการได้ เพราะไม่มีส่วนเชื่อมต่อมาถึงตัวโปรแกรมที่สร้างไว้ได้

ตัวนับจับเวลาตัวที่ ๑ เครื่องจะมีตัวนี้ เป็นหัวใจในการควบคุมเวลาต่างๆของเครื่อง หลังจากที่เราเปิดเครื่องขึ้นมา โปรแกรมในไบออสจะตั้งค่าไว้ให้นับถอยหลัง เมื่อการนับถอยหลังลงมาจนถึงศูนย์ แล้วจะส่งสัญญาณไปขัดจังหวะการทำงานของเครื่อง ให้มาทำโปรแกรมย่อยของการขัดจังหวะเบอร์ ๘ การส่งสัญญาณขัดจังหวะของตัวนับจับเวลามายังตัวซีพียู จะส่งผ่านมาทางไอซีควบคุมการขัดจังหวะเบอร์ 8259 ดังนั้นการขัดจังหวะนี้จึงเป็นการขัดจังหวะของฮาร์ดแวร์ และมีระดับความสำคัญของการขัดจังหวะสูงกว่าการขัดจังหวะของแป้นพิมพ์ การนับถอยหลังเมื่อลดลงเป็นศูนย์ เกิดการขัดจังหวะและจะเริ่มนับถอยหลังจากค่าที่ตั้งไว้ครั้งสุดท้าย ไม่ใช่ว่าหลังจากเกิดการขัดจังหวะแล้วจะต้องนับด้วย OFFFFh เสมอ โปรแกรมย่อยการขัดจังหวะของเครื่อง

จะไม่มีการตั้งค่านับถอยหลังใหม่ ตัวนับจับเวลาตัวนี้จึงสามารถนำมาใช้ได้

เมื่อพิจารณาแล้วพบว่าตัวนับจับเวลาตัวที่ ๑ มีความเหมาะสมเพราะมีวงจรที่ประกอบระบบการขัดจังหวะอยู่แล้ว และการตั้งนับถอยหลังก็ให้ช่วงการขัดจังหวะที่ถี่มากๆได้ แต่ข้อควรระวังก็มีอยู่เหมือนกัน เพราะตัวนับจับเวลาตัวนี้ไม่ได้ออกแบบมาให้รองรับกับการนำไปใช้ในงานทั่วไประบบของเครื่องจะเป็นผู้ใช้อยู่แล้ว ใช้ในการนับเวลาของเครื่องว่าเปิดเครื่องมานานเท่าใดแล้วและใช้กับการขับแผ่นจานแม่เหล็ก โดยจะเกี่ยวข้องกับการหมุนของมอเตอร์ ข้อที่สามารถเกิดขึ้นคือเมื่อแผ่นขับจานแม่เหล็กทำงานสิ้นสุดแล้ว จะมีการนับรอต่อไปค่าหนึ่งแล้วมอเตอร์จึงจะหยุดหมุนเป็นการทำงานของไบออสในโปรแกรมย่อยการขัดจังหวะ ถ้ามีการเปลี่ยนแปลงโปรแกรมย่อยการขัดจังหวะ โดยการไปเปลี่ยนที่ตารางเวคเตอร์ (Vector Table) ของการขัดจังหวะไม่ให้ผ่านไปถึงไบออส โดยไม่รอให้มอเตอร์หยุดเสียก่อน จะให้ผลคือมอเตอร์จะหมุนค้างไปจนกว่าไบออสจะได้ทำการนับถอยหลังเพื่อหยุดมอเตอร์ต่อไป

สัญญาณอ้างอิงที่ต้องการจะสร้างขึ้นมีลักษณะเหมือนรูปคลื่นสี่เหลี่ยมทั่วๆไป คือประกอบด้วยขอบขาขึ้น ขอบขาลง และระดับสัญญาณ "0", "1" ขอบขาขึ้นหรือลงของคาบใดๆจะมีเพียงครั้งเดียวเท่านั้น ในโครงงานนี้จะนำมาใช้แค่ขอบขาขึ้นเท่านั้น

รูปที่ ๔.๓ แสดงรูปคลื่นสี่เหลี่ยมทั่วไปเทียบกับรูปขอบขาขึ้นที่จะนำมาแทน

การสร้างสัญญาณการขัดจังหวะให้เกิดขึ้นตรงกับขอบขาขึ้น สามารถหามาจาก กำหนดคาบเวลาก่อน ค่านับถอยหลังค่าหนึ่งๆจะคิดเป็นเวลา ๑ / (๑.๑๙๓ * ๑๐^{6}) วินาที นำค่าของ ๑.๑๙๓*๑๐^{6} มาหารจะได้จำนวนที่จะนับนำไปใส่ลงในตัวนับจับเวลา แต่ในโครงงานต้องการ

หลังได้ จะถือว่าสัญญาณหมดคาบลงก็จะนำค่าคาบของการนับของช่องสัญญาณเดียวกันมาบวก เป็นค่ารอคอย จากนั้นจะหาว่าค่ารอคอยใดมีค่าน้อยที่สุด ซึ่งจะนำมาใช้เป็นค่าที่ให้แก่ตัวนับจับเวลาต่อไป

บทที่ ๕

การทดลองที่ ๑

ในการทดลองใช้เครื่องพีซี เป็นตัวเลียนแบบของเฟสดีเทคเตอร์โดยจะมีสัญญาณ ที่สร้างขึ้นภายในเครื่องเป็นตัวกำเนิดความถี่อ้างอิง ที่ป้อนให้แก่เฟสดีเทคเตอร์ส่วนสัญญาณเข้าอีกอันหนึ่งนั้น จะนำมาจากภายนอกรับผ่านเข้ามาทางพอร์ทของเครื่องเบอร์ 0300h แต่ละบิทจะหมายถึง 1 ช่องสัญญาณ ใช้โปรแกรมเป็นตัวประมวลผลแล้วส่งออกจากเครื่องสำหรับเฟสดีเทคเตอร์ที่เลียนแบบนี้ จะให้เอาท์พุทออกมา 3 สถานะคือ "0","1", และ อิมพีแดนซ์สูงเครื่องพีซีไม่สามารถให้เอาท์พุทในลักษณะนี้ได้ จึงให้แต่ละช่องสัญญาณต่อออกมา ช่องละ 2 บิทแทนโดยเป็นพอร์ทเบอร์ 0300h, 0301h ผ่านไปเข้าวงจรทำให้เป็น 3 สถานะ

การเลียนแบบนี้ได้แบ่งออกเป็น 2 อย่างคือ วิธีการของการเปิดตาราง และ วิธีการทางลอจิค

การเลียนแบบโดยวิธีการเปิดตาราง

ตารางที่จะนำมาเปิดนั้นมาจาก การแปลงวิธีการเปลี่ยนสถานะของของบอลงแบบ 3 สถานะ ส่วนของโปรแกรมย่อยของการเลียนแบบเฟสดีเทคเตอร์ มีส่วนการทำงานหลักเขียนด้วยภาษาแอสเซมบลี้ ของ 8086 แรกทีเดียวโปรแกรมย่อยส่วนนี้เขียนด้วยภาษาปาสคาล และภาษาซี แต่จะให้ความเร็วที่ต่ำกว่า หลักการของการเปิดตารางคือ ตารางจะแบ่งออกเป็น 2 ส่วน คือ ส่วนของสถานะ และส่วนของเอาท์พุท มีทั้งหมด 8 แถว และ 12 สถานะ เมื่อเริ่มต้นจะอยู่ที่สถานะใดสถานะหนึ่ง โปรแกรมจะมีอาเรย์ (ARRAY) เก็บค่าว่าสถานะใดอยู่ที่แถวใด ชื่อ RowOfState[0..11] ค่าที่อ่านได้จากอาเรย์นี้ จะได้ค่าแถวของสถานะนั้น เอาค่าที่ได้ประกอบกับค่า สัญญาณขาเข้าอีก 2 ขาเพื่อใช้เปิดตารางที่จะบอกว่าสถานะต่อไปคือสถานะใด StateTran[0..7][0..1][0..1] นำค่าไปหาจากตาราง RowOfState นำค่านี้ไปหาค่าเอาท์พุท จากตาราง _QComp[0..7], _QSig[0..7] วิธีการนี้จะต้องทำการเปิดตารางไปทีละบิตไปจนครบ 8 บิตเพราะว่าตารางไม่สามารถเปิดพร้อมกันได้ วิธีการนี้จะไม่จำเป็นต้องเก็บค่าของสัญญาณขาเข้า ในคาบที่แล้วเอาไว้แต่จะต้องเก็บค่าสถานะเอาไว้แทน

การทดสอบว่าโปรแกมย่อยนี้จะทำงานได้นั้น ทำได้โดยการให้ค่าแก่สัญญาณขาเข้าลงในตัวแปร _Sig เป็นสัญญาณอ้างอิง และ _Comp เป็นสัญญาณเปรียบเทียบ จากนั้นเรียกใช้

โปรแกรมย่อย PhaseDetect จะได้ค่าเอาท์พุทออกมาทางตัวแปร _QComp และ _QSig ค่าของ _QSig จะเป็น "1" เมื่อสัญญาณ _Sig นำหน้าขณะที่ _QComp เป็น "1" เมื่อสัญญาณ _Comp นำหน้า

รายละเอียดของโปรแกรมย่อย PhaseDetect

```
        locals  @@

_TEXT   segment byte public 'CODE'
_TEXT   ends

_DATA   segment word public 'DATA'
_DATA   ends

        assume  cs:_TEXT,ds:_DATA

_TEXT   segment byte public 'CODE'

LookUpTable     proc    near
        mov     bl,cl
        mov     ch,[_StateIndex+bx]
        shl     ch,1
        shl     ch,1

        mov     al,[_Sig]
        mov     ah,[_Comp]
        shr     ax,cl
        and     ax,0101h
        shl     al,1
        or      al,ah
```

```
or      ch,al

mov     bl,ch
mov     al,[StateTran+bx]
mov     bl,cl
mov     [_State+bx],al
mov     bl,al

mov     ch,[RowOfState+bx]
mov     bl,cl
mov     [_StateIndex+bx],ch
mov     bl,ch

mov     al,[QSigArray+bx]
mov     ah,[QCompArray+bx]
shl     ax,cl
mov     dx,ax

mov     bl,cl
mov     al,[MaskBit+bx]
not     al
mov     ah,al
and     al,[_QSig]
and     ah,[_QComp]
or      ax,dx

mov     [_QSig],al
mov     [_QComp],ah

ret
```

```
LookUpTable     endp

PhaseDetect     proc    far
        xor     cl,cl
        xor     bh,bh
@@1:
        call    LookUpTable
        inc     cl
        cmp     cl,8
        jnz     @@1
        ret
PhaseDetect     endp

_TEXT   ends

_DATA   segment word public 'DATA'

StateTran       label   byte
        db      0, 1, 3, 2
        db      4, 1, 7, 2
        db      4, 5, 7, 6
        db      8, 5,11, 6
        db      4, 1,11, 6
        db      0, 1, 7, 6
        db      8, 9,11,10
        db      4, 5,11,10
RowOfStatelabelbyte
        db      0,1,1,0,2,3,4,5,6,6,7,7
QSigArray       label   byte
        db      1,1,0,0,0,0,0,0
```

```
QCompArray      label   byte
        db      0,0,0,0,0,0,1,1
MaskBit         label   byte
        db      01h,02h,04h,08h,10h,20h,40h,80h
_State          label   byte            ; with rank of 0 .. 11
        db      8 dup (?)
_StateIndex     label   byte            ; with rank of 0 .. 7
        db      8 dup (?)

_DATA   ends

        extrn   _Sig:byte
        extrn   _Comp:byte
        extrn   _QSig:byte
        extrn   _QComp:byte
        public  PhaseDetect

        end
```

การเฉียนแบบโดยใช้วิธีการทางลอจิก

การทำงานของเฟสดีเทคเตอร์ที่กล่าวมาแล้วในบทต้นๆมีสิ่งสำคัญที่จำเป็นที่จะต้องหาคือขอบขาขึ้นของสัญญาณ (หรือขอบขาลง) ถ้าแทนสัญญาณลอจิคปัจจุบันด้วย X_n สัญญาณที่ได้จากการรับเข้ามาคาบที่แล้วคือ X_{n-1} ถ้ามีการเปลี่ยนแปลงสัญญาณจาก "0" เป็น "1" แสดงว่าเป็นขอบขาขึ้นและจะทำให้ค่าของ NOT(X_{n-1}) AND X_n เป็นจริง เมื่อแทน X ด้วยแต่ละบิทของ _Sig และ _Comp ก็จะสามารถหาขอบขาขึ้นของสัญญาณทั้งสองได้ จากหลักการของการตรวจจับเฟสแบบนี้เมื่อเกิดมีขอบขาขึ้นอันใดอันหนึ่งขึ้นมา สัญญาณเอาท์พุทของสัญญาณนั้น จะคงอยู่จนกว่าจะเกิดขอบขาขึ้นของอีกสัญญาณหนึ่ง หรืออาจดูได้อีกอย่างหนึ่งได้จากเมื่อสัญญาณเอาท์พุทอันแรกเป็น "1" จะคงอยู่จนกว่าสัญญาณเอาท์พุทอีกอันหนึ่งเป็น "1" พร้อมกันทั้งคู่ ขั้นตอนการกระทำทางลอจิกสามารถลำดับได้คร่าวๆคือ ตรวจสอบสัญญาณขาเข้าว่ามีอันไหนบ้างที่เกิดขอบขาขึ้นมา ทำให้สัญญาณเอาท์พุทที่ตรงกันเป็น "1" แล้วดูว่าถ้าเอาท์พุทเป็น "1" พร้อมกันทั้งคู่จะตั้งค่า

สัญญาณเอาท์พุทให้เป็น "0" ทั้งคู่ หรือจะเขียนเป็นขั้นตอนทางคณิตศาสตร์ได้ดังนี้

$$Q_{sig} := Q_{sig} \text{ OR } (\text{NOT}(Sig_{n-1}) \text{ AND } Sig_n)$$

$$Sig_{n-1} := Sig_n$$

$$Q_{comp} := Q_{comp} \text{ OR } (\text{NOT}(Comp_{n-1}) \text{ AND } Comp_n)$$

$$Comp_{n-1} := Comp_n$$

$$Reset := \text{NOT } (Q_{comp} \text{ AND } Q_{sig})$$

$$Q_{sig} := Q_{sig} \text{ AND } Reset$$

$$Q_{comp} := Q_{comp} \text{ AND } Reset$$

การใช้วิธีการแบบนี้มีข้อดีคือสามารถทำขั้นตอนที่กล่าวมาแล้วได้พร้อมๆกันทั้ง 8 บิทจะลดเวลาการเทียบเฟสลงได้มาก

รายละเอียดของโปรแกรม PhaseDetect ที่ใช้หลักสมการลอจิก

```
_TEXT   segment byte public 'CODE'
_TEXT   ends

_DATA   segment word public 'DATA'
_DATAends

        assume CS:_TEXT,DS:_DATA

PosEdge macro   Signal,Signaln1,QSignal,Dest
        mov     ah,[Signaln1]
        not     ah
        mov     al,[Signal]
        mov     [Signaln1],al
        and     al,ah           ; AL = 1 if Raising edge found
        mov     ah,[QSignal]    ; set output logic
```

```
        or      al,ah
        mov     Dest,al
endm
```

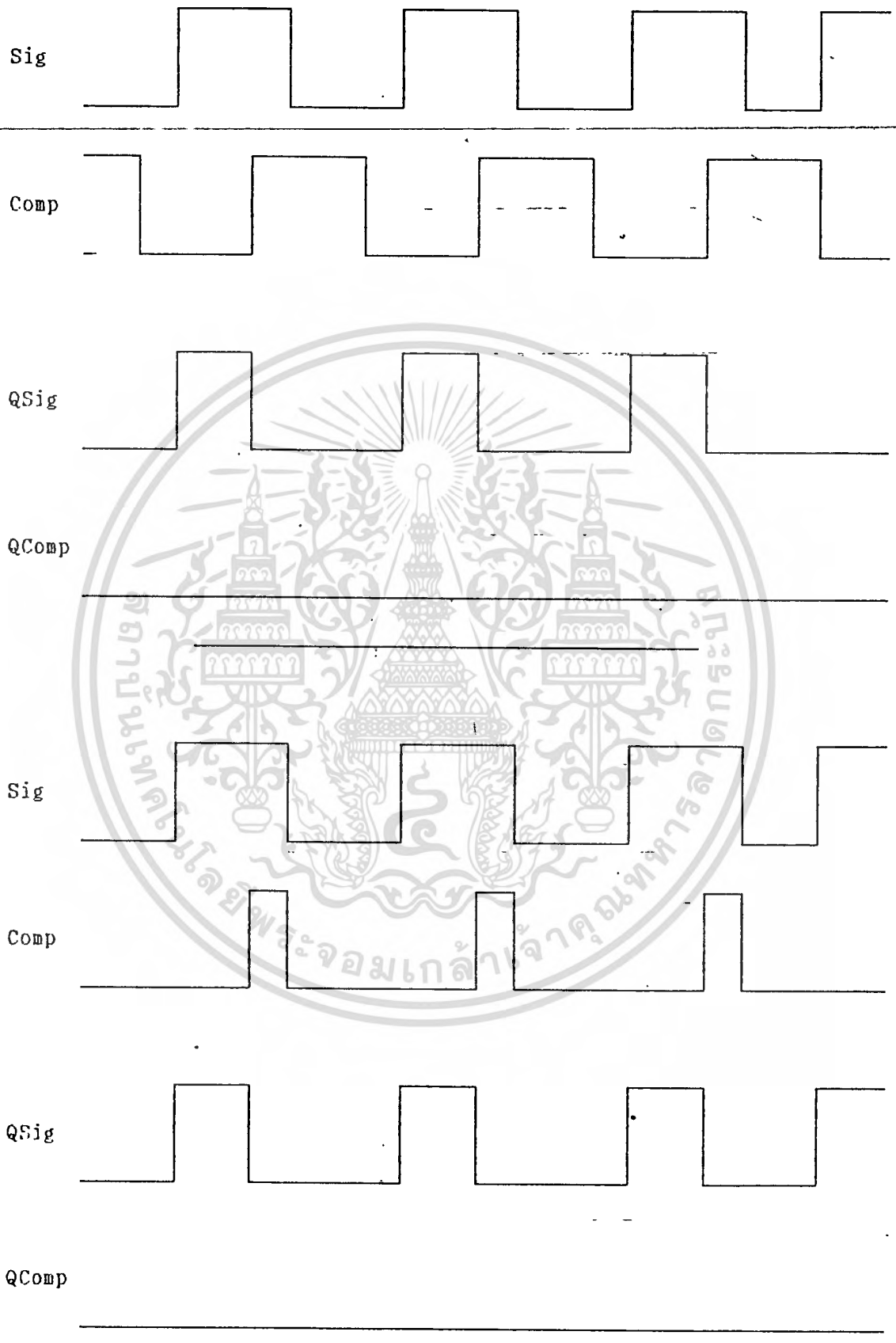
```
Valve   macro   Gate,source
        mov     al,bl
        and     al,source
        mov     [Gate],al
endm

_TEXT   segment byte public 'CODE'

PhaseDetect     proc    near
        rQSig   equ     cl
        rQComp  equ     ch

        PosEdge _Sig,_Sign1,_QSig,rQSig
        PosEdge _Comp,_Compn1,_QComp,rQComp
        mov     bl,rQSig
        and     bl,rQComp
        not     bl
        Valve   _QSig,rQSig
        Valve   _QComp,rQComp
ret
PhaseDetect     endp

_TEXT   ends

_DATA   segment word public 'DATA'
```

```
_Compn1         label   byte
        db      ?

_DATA   ends

        extrn   _QSig:byte
        extrn   _QComp:byte
        extrn   _Sig:byte
        extrn   _Comp:byte
        public  PhaseDetect
        extrn   _Sign1:byte

        end
```

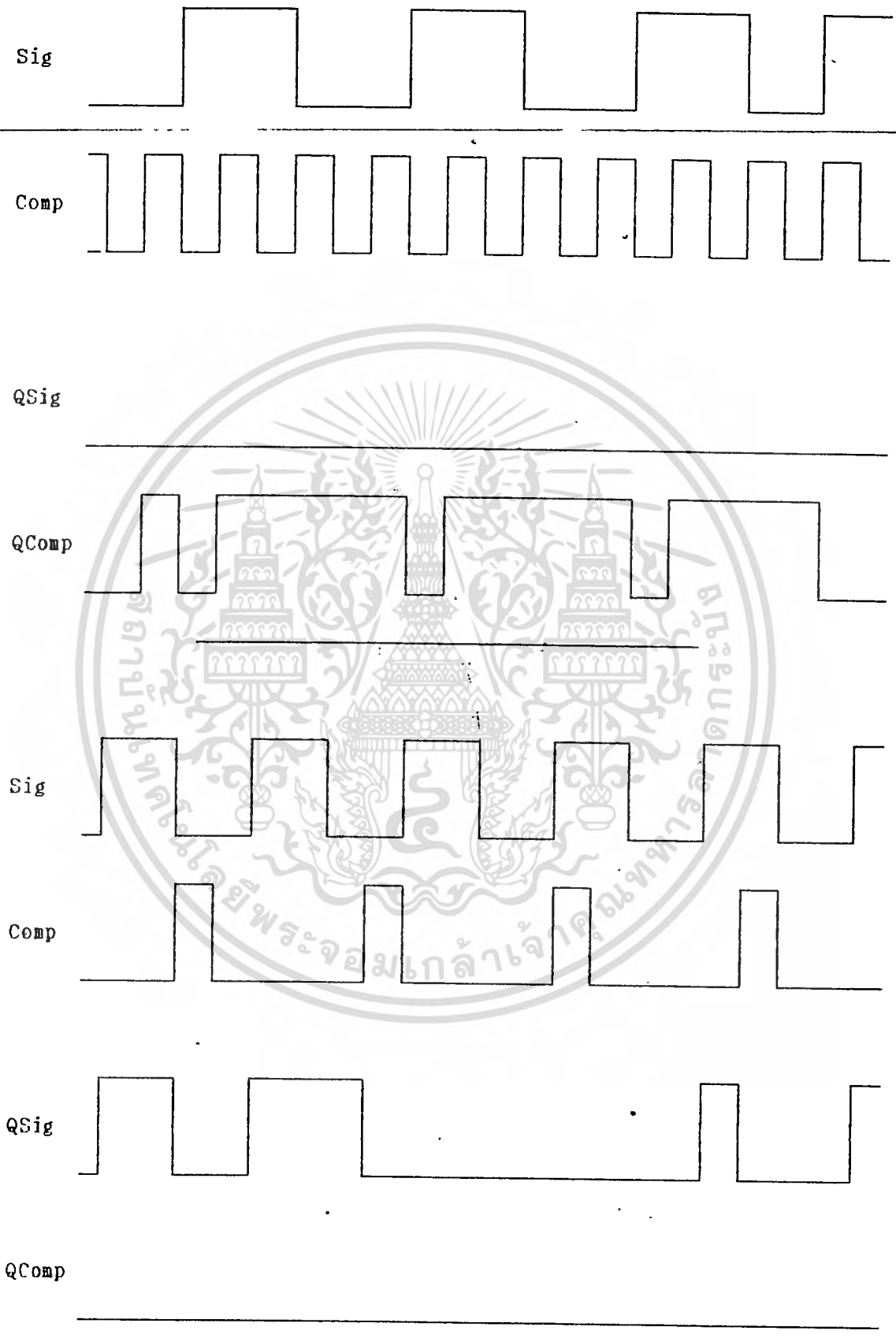
บทที่ ๖

ผลการทดลองที่ ๑

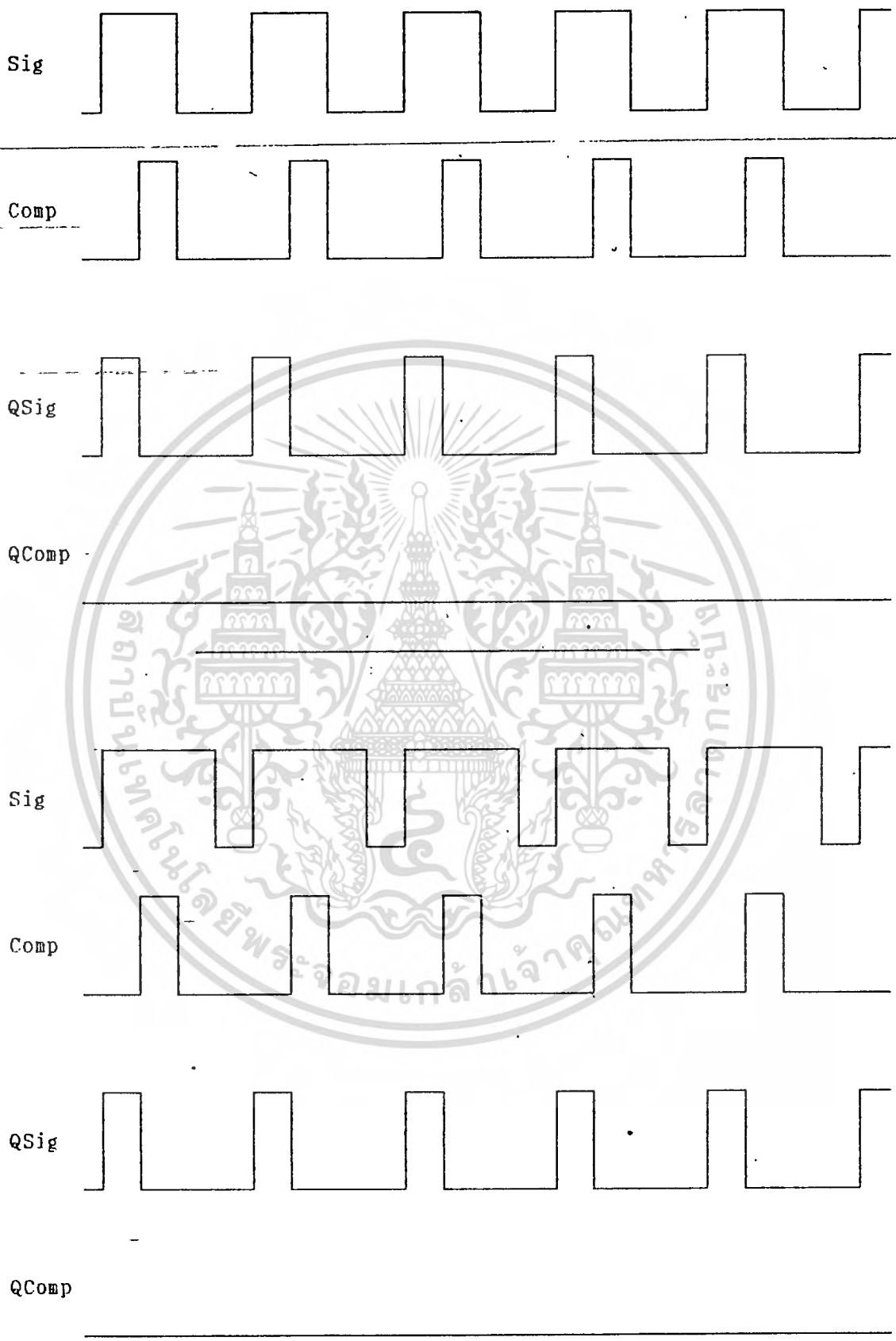
โปรแกรมย่อยทั้งสองโปรแกรมนี้การทดลองทำขึ้นเพื่อทดสอบความถูกต้องในด้านการทำงานของ จะเขียนโปรแกรมเพื่อทดสอบโดยการป้อนสัญญาณขาเข้าที่มีคาบเวลาเท่ากันแต่มี ค่าดิวตี้ไซเกิ้ล (Duty cycle) ไม่เท่ากัน และสัญญาณที่มีคาบเวลาไม่เท่ากัน โปรแกรมที่สร้างขึ้นเพื่อใช้ในการทดสอบนี้มีรายละเอียดอยู่ในภาคผนวก จะมีสัญญาณป้อนให้แก่โปรแกรมย่อยและผลที่ออกมาจะอยู่ในรูปตัวอักษรลักษณะของคลื่นสี่เหลี่ยม ดูได้โดยผ่านทางโปรแกรมประมวลผลคำภาษาไทยหรือผ่าน ไทยไดรเวอร์ (Thai driver) ก็ได้ การทดลองเพื่อทดสอบความถูกต้องทางลอจิกจะไม่สามารถบอกความเร็วของการทำงานได้ การทดสอบความเร็วทำต่อไปโดยรับสัญญาณจากภายนอกเข้ามา 2 เส้นและส่งสัญญาณที่ทำได้ออกทางพอร์ทเพื่อวัดจะเห็นว่ามีการเปลี่ยนแปลงระดับสัญญาณเมื่อความถี่ขาเข้า ใกล้เคียงกันเมื่อเข้าใกล้กันพอสมควรแล้วจะคงที่ความถี่ขาเข้าไว้ และใช้โปรแกรมย่อยสองอย่างเปรียบเทียบ เพื่อต้องการผลสรุปว่าวิธีใดทำได้เร็วกว่ากันก็ได้ออกมาว่าวิธีการของสมการลอจิกเร็วกว่า



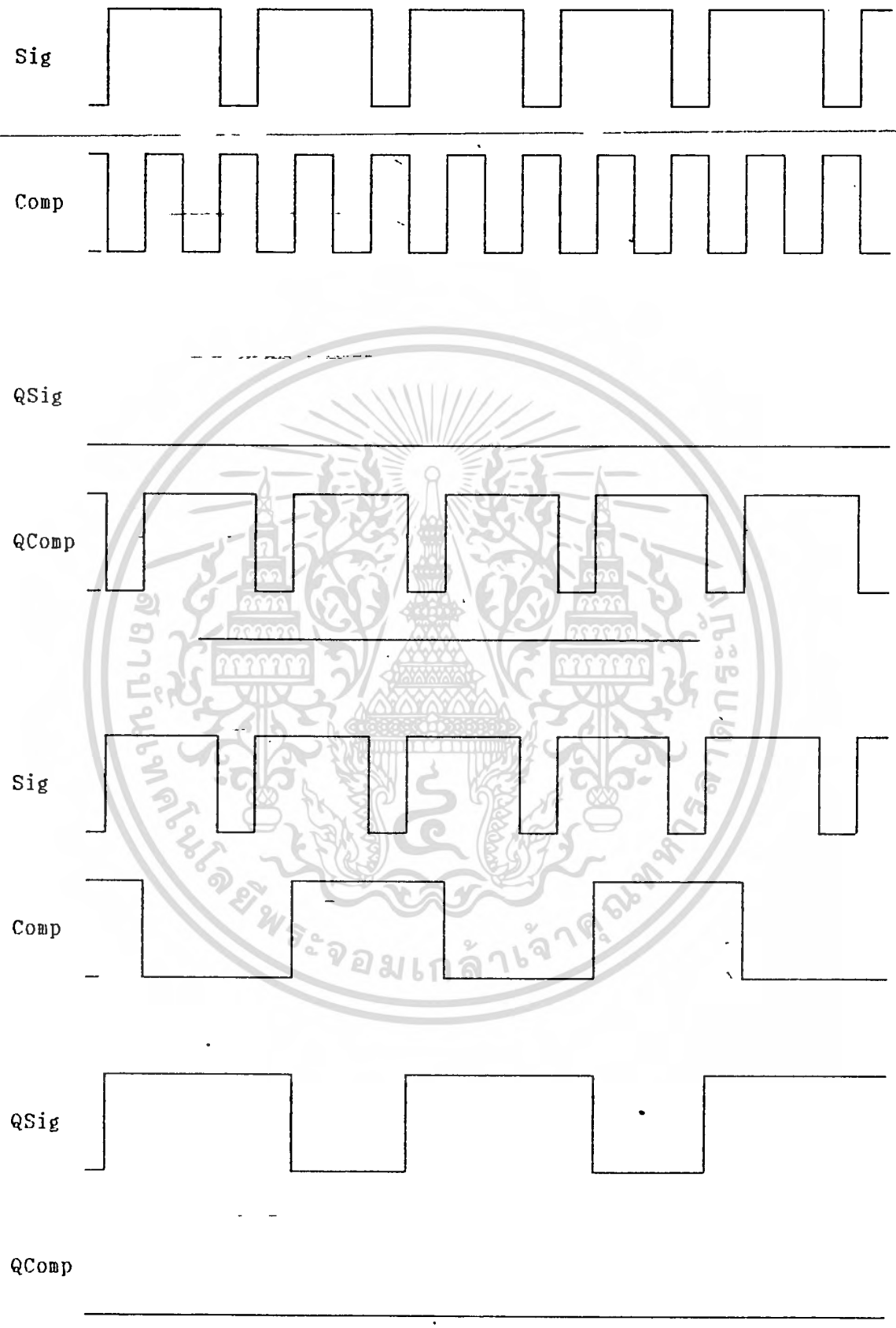
ผลการทดลองตัวเปรียบเทียบเฟสจากวิธีการเปิดตาราง เมื่อสัญญาณมีคาบเท่ากัน



ผลการทดลองตัวเปรียบเทียบเฟสจากวิธีการเปิดคาราง เมื่อสัญญาณมีคาบไม่เท่ากัน



ผลการทดลองตัวเปรียบเทียบเฟสจากวิธีการสมการลอจิก เมื่อสัญญาณมีค่าเท่ากัน



ผลการทดลองตัวเปรียบเทียบเฟสจากวิธีการสมการลอจิก เมื่อสัญญาณมีคาบไม่เท่ากัน

บทที่ ๗

การทดลองที่ ๒

การทดลองนี้เป็นการนำเอาโปรแกรมย่อยจากการทดลองที่แล้วมาใช้งาน การทดลองที่แล้วมี 2 โปรแกรมย่อยเราได้เลือกวิธีการของการใช้สมการลอจิคเพราะสามารถทำงานพร้อมกันไปได้ทั้ง 8 ปิทมาใช้ในส่วนของตัวเปรียบเทียบเฟส สัญญาณแรกเป็นสัญญาณอ้างอิงสร้างมาจากสัญญาณนาฬิกาของเครื่องใช้หลักการในบทที่ 4 สัญญาณที่ใช้เปรียบเทียบกับสัญญาณแรกรับเข้ามาจากพอร์ท มาจาก VCO ที่ต่อครบลูฟ จากเครื่องเปลี่ยนเป็นลอจิก 3 สถานะผ่านไปเข้าตัวกรองความถี่ต่ำผ่าน ต่อไปยัง VCO ป้อนกลับเข้ามาเป็นสัญญาณอ้างอิงของเครื่อง ผลที่ได้คือช่วงความถี่ของการจับสัญญาณ ๔๐๐-๘๓๓ เฮิรต์ โดยมีโปรแกรมอยู่สองโปรแกรม

๑. โปรแกรมสร้างสัญญาณนาฬิกา

```
        locals  @@
_TEXT   segment byte public 'CODE'
_TEXT   ends

_DATA   segment word public 'DATA'
_DATA   ends

Lisbon  struc
WaitCount       dw      ?
Period          dw      ?
MaskBit         db      ?
Next            db      ?
Active          db      ?
Filler          db      ?
Lisbon  ends
```

```
        assume  cs:_TEXT,ds:_DATA

_TEXT   segment byte public 'CODE'

NewTimer        proc    near
        mov     dx,0FFFFh
        xor     ch,ch
        mov     bh,ch
        mov     bl,[ListHeader]
        mov     cl,3
@@While1:
        shl     bl,cl
        mov     ax,Lisbon ptr [Schedule+bx].WaitCount
        or      ax,ax
        jz      short @@ExitWhile1

        mov     ax,[LastCount]
        cmp     ax,mincount
        ja      @@1

        add     ax,Lisbon ptr [Schedule+bx].Period
        or      ch,Lisbon ptr [Schedule+bx].MaskBit; Positive trig
@@1:
        cmp     ax,dx
        jnb     @@2
        mov     dx,ax
@@2:
        mov     word ptr [Schedule+bx].WaitCount,ax
```

```
        mov     bl,Lisbon ptr [Schedule+bx].Next
        jmp     @@while1
@@ExitWhile1:
;@@3:
        mov     ax,mincount
        mov     [LastCount],ax
        out     40h,al
        mov     al,ah
        out     40h,al

        mov     [_Sig],ch
        call    PhaseDetect
        xor     al,al
        mov     [_Sign1],al

        ret
NewTimer        endp

_TEXT   ends

_DATA   segment word public 'DATA'
_DATA   ends

        extrn   PhaseDetect:near
        extrn   _Sig:byte
        extrn   _Sign1:byte
        extrn   Schedule:Lisbon
        extrn   ListHeader:byte
```

```
        extrn   LastCount:word
        extrn   mincount:word
        extrn   maxcount:word
        public  NewTimer

        end
```

ข. โปรแกรมตั้งค่าสัญญาณนาฬิกา

```
uses Dos,Crt;
type
  PSchedule = ^TSchedule;
  TSchedule = record
    WaitCount : word;
    Period : word;
    MaskBit : byte;
    Next : byte;
    Active : byte;
    Filler : byte;
  end;

const
  ListHeader : byte = 8;
  Schedule : array [0..8] of TSchedule = (
    (WaitCount:0;Period:0;MaskBit:$01;Next:8;Active:0;Filler:0),
    (WaitCount:0;Period:0;MaskBit:$02;Next:8;Active:0;Filler:0),
    (WaitCount:0;Period:0;MaskBit:$04;Next:8;Active:0;Filler:0),
    (WaitCount:0;Period:0;MaskBit:$08;Next:8;Active:0;Filler:0),
```

```
        (WaitCount:0;Period:0;MaskBit:$10;Next:8;Active:0;Filler:0),
        (WaitCount:0;Period:0;MaskBit:$20;Next:8;Active:0;Filler:0),
        (WaitCount:0;Period:0;MaskBit:$40;Next:8;Active:0;Filler:0),
        (WaitCount:0;Period:0;MaskBit:$80;Next:8;Active:0;Filler:0),
        (WaitCount:0;Period:0;MaskBit:$00;Next:8;Active:0;Filler:0)
    );
    _Sig : byte = 0;
    _Sign1 : byte = 0;
    _Comp : byte = 0;
    _QComp : byte = 0;
    _QSig : byte = 0;
    LastCount : word = $FFFF;
    MinCount : word = $1000;
    MaxCount : word = $FFFF;
var
  SaveExitProc : pointer;
  FinishRestore : boolean;

procedure PhaseDetect; near; external;
{$L DFLIP.OBJ}
procedure NewTimer; near; external;
{$L CLOCK.OBJ}

procedure AdjustMaxCount;
var
  ix : byte;
  minperiod : word;
begin
```

```
  minperiod := $FFFF;
  ix := ListHeader;
  while ix <> 8 do
  begin
    if minperiod > Schedule[ix].period then
      minperiod := Schedule[ix].period;
    ix := Schedule[ix].next;
  end;
  minperiod := minperiod div 2;
  if minperiod < mincount then minperiod := mincount;

  maxcount := minperiod;
end;

procedure AddToSchedule(Pos : byte;Period : word);
var
  SchPosP : PSchedule;
begin
  SchPosP := @Schedule[Pos];
  if SchPosP^.active = 0 then
  begin
    SchPosP^.active := 1;
    SchPosP^.next := ListHeader;
    ListHeader := Pos;
  end;

  inline($FA {CLI});
  SchPosP^.period := Period;
```

```
    SchPosP^.waitcount := LastCount;
    inline($FB {STI});

    AdjustMaxCount;
  end;

  procedure DelFromSchedule(Pos : byte);
  var
    SchPosP : PSchedule;
    nextix : ^byte;
    ix : byte;
  begin
    Schedule[Pos].active := 0;
    if ListHeader = Pos then
    begin
      ListHeader := Schedule[Pos].next
    end else begin
      ix := ListHeader;
      repeat
        nextix := @Schedule[ix].next;
        if nextix^ = pos then nextix^ := Schedule[Pos].next;
        ix := nextix^;
      until ix = 8;
    end;

    AdjustMaxCount;
  end;
```

```
function Bit(bitno : byte;j: integer) : integer;
var
  f : word;
begin
  f := 1 SHL bitno;

  Bit := j AND f;
end;

var
  SaveInt08 : pointer;

procedure SysTimer; interrupt;
begin
  _Comp := Port[$0300];
  NewTimer;
  Port[$0300] := _QSig;
  Port[$0301] := _QComp;

  Port[$20] := $20;
end;

procedure RestoreSpeed; interrupt;
begin
  FinishRestore := TRUE;
  Port[$40] := $FF;
  Port[$40] := $FF;
```

```
  Port[$20] := $20;
end;

procedure WaitMotor;
var
  MotorState : byte absolute $0040:$003F;
begin
  repeat until (MotorState AND $0F) = 0;
end;

{$F+}
procedure RestoreClockExit;
begin
  ExitProc := SaveExitProc;

  FinishRestore := FALSE;
  SetIntVec($08,@RestoreSpeed);
  repeat until FinishRestore;
  SetIntVec($08,SaveInt08);
end;
{$F-}

var
  code : integer;
  newfreq : string;
  freq,period : word;
  x,y : byte;
```

```
begin
  if paramcount < 1 then halt;
  val(ParamStr(1),MinCount,Code);
  if Code <> 0 then halt;

  WaitMotor;

  GetIntVec($08,SaveInt08);
  SaveExitProc := ExitProc;
  ExitProc := @RestoreClockExit;
  SetIntVec($08,@SysTimer);

  repeat
   write('Index : ');
   readln(NewFreq);
   Val(NewFreq,freq,code);
   if (code = 0) AND (freq > 0) then
   begin
     period := trunc(1193000.0 / freq);
     if period > mincount then
     begin
       writeln('New period : ',period);
       AddToSchedule(0,period);
     end;
   end;
  until FALSE;

end.
```

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบคุณท่านอาจารย์ที่ปรึกษาคือ รองศาสตราจารย์ ดร.โยธิน เปรมปราณีรัชต์ ที่ได้กรุณาสละเวลาให้คำปรึกษาชี้แนะ ตลอดจนช่วยผลักดันให้วิทยานิพนธ์นี้ สำเร็จลุล่วงไปได้ด้วยดี นอกจากนี้ขอขอบคุณ อ. เกียรติวรรณ ทรงสัตย์ ที่ให้ความสะดวกในด้านอุปกรณ์และเครื่องมือทดลอง ข้าพเจ้าขอขอบคุณท่านที่มีส่วนสนับสนุนให้คำแนะนำ ดังมีรายนามดังนี้

๑. นายชัยศรี เอี่ยมอำไพ

๒. นายเจตนา สิงห์สินธ์

ผู้จัดทำ

หนังสืออ้างอิง

๑. "TURBO PASCAL (object-oriented programming guide", CA : Borland International Inc, ๑๙๘๙.

๒. Wills J. Tompkins, John G. Webster, "Interfacing Sensor to the IBM PC", New Jersey : Prentice-Hall International Editions, ๑๙๘๘.

๓. Ramakant A. Gayakwad, "Op-Amps and linear integrated circerts" New Jersey : Prentice-Hall Inc, ๑๙๘๘, P ๓๓๑-๓๔๓.

๔. รศ. โยธิน เปรมปราณีรัชต์, "Practical Design Phase-loched loop for Motor Speed Control", รายงานการวิจัย, สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง.

๕. ออมสิน พิบูลวรางกูร, "Phase Detector Simulation", วิทยานิพนธ์ ปริญญาตรี, ภาควิชาวิศวกรรมระบบควบคุม สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหารลาดกระบัง, ๒๕๒๙.

๖. ชัยยศ สุนทรวรรณ, สุรเดช พันธ์วัง, "เฟสล็อคลูฟสำหรับการควบคุมความเร็วของมอเตอร์กระแสตรง", วิทยานิพนธ์ปริญญาตรี, ภาควิชาวิศวกรรมระบบควบคุม สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง, ๒๕๒๕.

๗. บุญรอด มานิตย์โชติพิสิฐ, ปรีชัย ธารนิทัศน์, "Phase Locked Loop for DC motor control", วิทยานิพนธ์ปริญญาตรี, ภาควิชาวิศวกรรมระบบควบคุม สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง, ๒๕๓๒.

๘. รศ. โยธิน เปรมปราณีรัชต์, "ระบบเซอร์โวและอิเลคทรอนิกส์คอนโทรลมอเตอร์", ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง, ๒๕๓๓, หน้า ๓๘๙-๔๓๔.

๙. วารสารไมโครคอมพิวเตอร์ ๕๓, "OOP ความหวังของการเก็บของเก่ามาใช้ใหม่", หน้า ๒๓๘-๒๔๕, ตุลาคม ๒๕๓๒.

๑๐. ผศ. ครรชิต ไมตรี, "การวิเคราะห์และออกแบบวงจรชีเควนเชียล" ภาควิชาคอมพิวเตอร์ สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง, ๒๕๓๓.

๑๑. William C. Lindsey, Chak M. Chie ,"Phase-Locked Loop", New York : IEEE PRESS, ๑๙๘๖.

๑๒. J. Michael Jacob, "Industrial Control Electronics Application And Design", New Jersey : Prentice-Hall Inc., ๑๙๘๘.

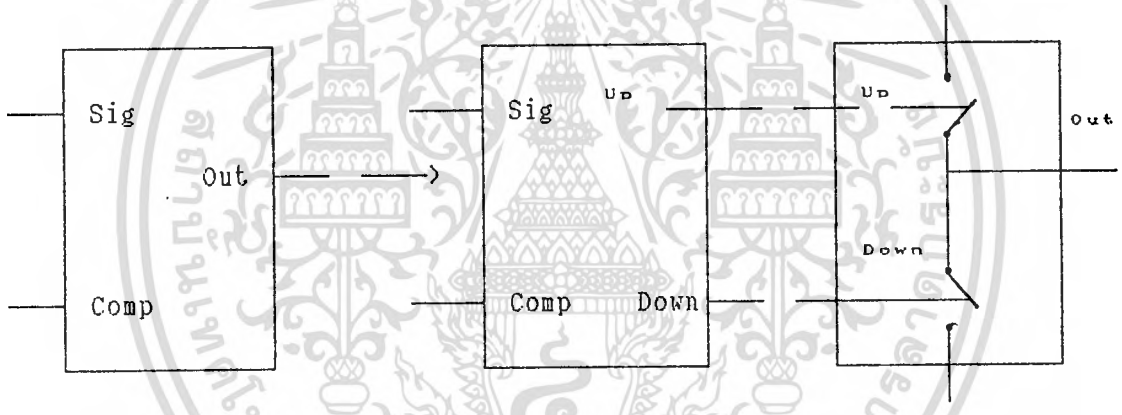
๑๓. Robert F. Coughlin, Robert S. Villanucoi,"Introductory Operational Amplifiers and Linear ICs. "(theory and experimentation), New Jersey: Prentice-Hall Inc, ๑๙๙๐ (P.๒๐๗ - ๒๒๙)

ภาคผนวก A

การแทนตัวเทียบเฟสและความถี่ในลักษณะต่าง ๆ

ตัวเทียบเฟส แบบที่สามารถใช้เปรียบเทียบทั้งเฟสและความถี่ เราสามารถพบได้ในหลายรูปแบบแตกต่างกันออกไป ตามแผ่นการออกแบบของคู่มือไอซีเบอร์ต่าง ๆ ซึ่งถ้าดูเพียงผิวเผินจะเห็นว่าของแต่ละเจ้า มีความแตกต่างกันทางด้านวิธีการ ซึ่งไม่น่าจะให้ผลในการเปรียบเทียบออกมาได้ตรงกัน สิ่งที่จะกล่าวกันต่อไป จะเกี่ยวกับการแสดงให้ดูว่ารูปแบบต่าง ๆ ที่พบได้นั้น ไม่ต่างกัน

ข้อสรุป สำคัญของตัวเทียบเฟสแบบนี้ คือ จะตรวจจับสัญญาณทางขอบขาขึ้น หรือขาลงของสัญญาณ ทางขอบขาขึ้น หรือขาลงของสัญญาณ และจะให้ผลออกมาเป็น สามสถานะ คือ "๐","๑" และเปิดวงจร



หรือมีบ้างก็จะแยกออกมาเป็นสองส่วน ส่วนหลังจะเป็นสวิทช์ควบคุมด้านขาควบคุม Up และ Down ถ้า Up = "๑" และ Down = "๐" จะทำให้ Out เป็น "๑" จะเขียนเป็นตารางได้ คือ

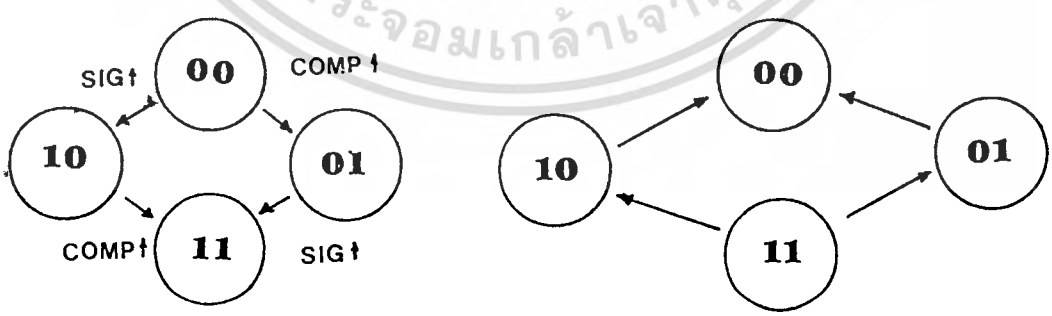
Up	Down	Out
0	0	High impedance
1	0	0
0	1	0
1	1	not allow

แต่ Up และ Down จะต้องไม่เป็น "๑" พร้อมกัน ตัวเทียบเฟสแทนที่จะส่งสัญญาณออกมาสามสถานะ ก็จะให้สัญญาณออกมาสองขาแทน โดยที่จะไม่ให้เป็น "๑" ทั้งคู่ สถานะที่เหลืออีกสามอัน ก็จะมีหน้าที่เหมือนกับตัวเทียบเฟสอันแรก

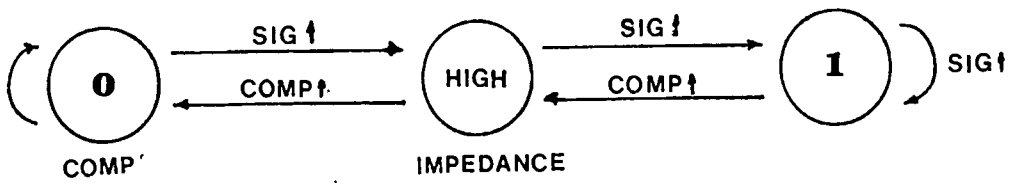
จะขอกล่าวถึง การทำงานของตัวเทียบเฟสแบบนี้อีกครั้งหนึ่ง สัญญาณที่เข้ามาจะมี สัญญาณอ้างอิง และสัญญาณเปรียบเทียบ ถ้ามองดูระบบโดยรวมของเฟสล็อคลูฟแล้ว จะเห็นว่าการควบคุมความถี่และเฟสให้ได้ตามต้องการ ถ้ามีสัญญาณอ้างอิงและลูฟอยู่ในสภาวะล็อค แสดงว่า สัญญาณนี้กลับมาเปรียบเทียบ มีเฟสและความถี่เท่ากับของสัญญาณอ้างอิงพอดี แต่ถ้าสัญญาณที่นำมาเปรียบเทียบ มีเฟสตามหลังอยู่ แสดงว่า ควรจะเพิ่มศักดาให้แก่ VCO และในทางตรงกันข้าม ถ้าสัญญาณที่ป้อนกลับมาเปรียบเทียบ มีเฟสทำหน้าสัญญาฯอ้างอิง ก็ควรจะลดศักดานี้ให้แก่ VCO ที่เป็นหลักคร่าว ๆ แล้ว ตัวเทียบเฟสจะทำอย่างไร ถ้าดูจากขอบสัญญาณขาขึ้นเป็นหลัก ตอนแรกจะดูว่าเราพบขอบขาขึ้นของสัญญาณใดก่อน ถ้าพบของตัวอ้างอิงก่อนแสดงว่าสัญญาณเปรียบเทียบตามหลังอยู่ ตามหน้าที่ของ VCO ที่ดีก็จะส่ง "๑" ไปที่ Out แต่ถ้าพบสัญญาณเปรียบเทียบก่อน ก็แสดงว่าสัญญาณเปรียบเทียบทำส่ง "๐" ไป สัญญาณ "๐" หรือ "๑" จะถูกส่งออกไป หลังจากขอบขาขึ้นของสัญญาณแรกผ่านเข้ามาและจะสิ้นสุดเมื่อ มีขอบขาขึ้นของอีกสัญญาณหนึ่งเข้ามา (ไม่ใช่ขอบขาขึ้นของสัญญาณเดียวกัน)

การแทนด้วยแผนภูมิเสตท

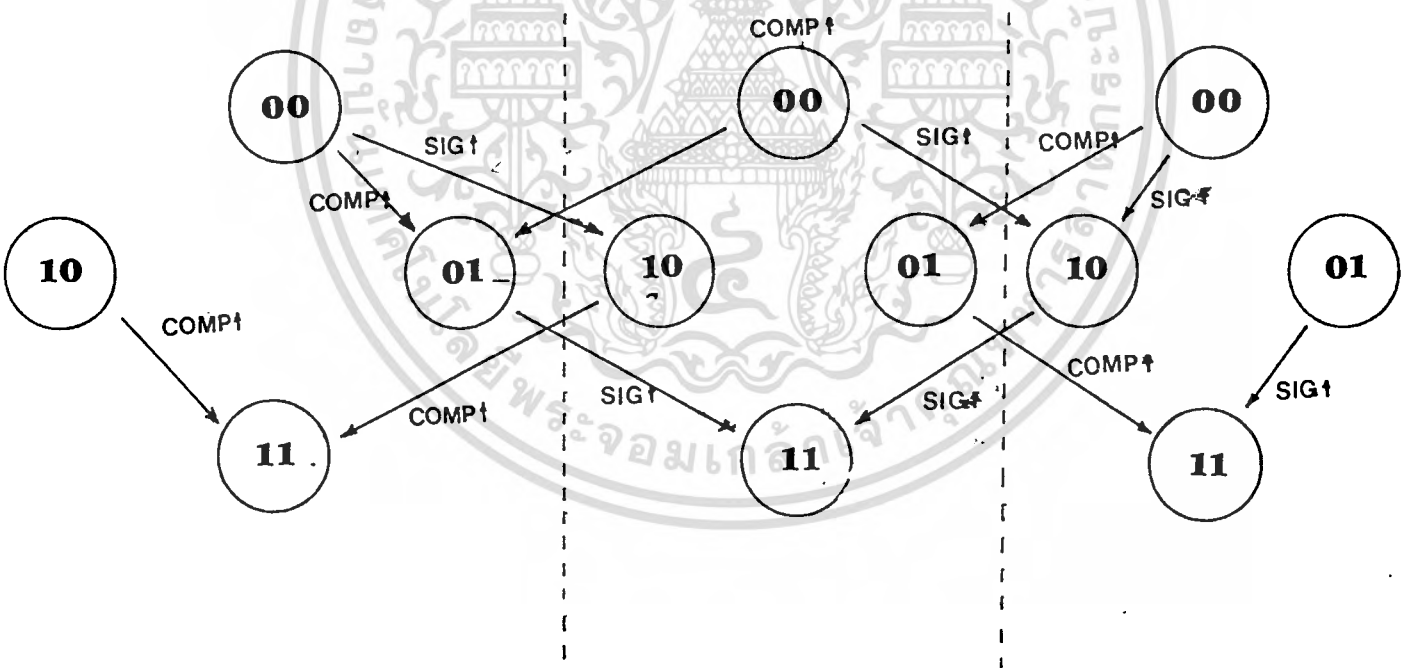
ถ้ากำหนดให้มีสถานะหลัก ๆ ขึ้นมา แทนผลที่ออกมาจากตัวเทียบเฟส จะได้สามสถานะหลัก ๆ และมีการเปลี่ยนสถานะเนื่องด้วยสัญญาณขาเข้าดังนี้



จากแผนภูมิดังกล่าว จะเห็นว่ายกมาเพียง เมื่อมีการเปลี่ยนแปลงสัญญาณขาเข้า สัญญาณขาออกจะให้ค่าอะไร แต่ที่กล่าวมาเราไม่สามารถบอกได้ว่า สัญญาณมีการเปลี่ยนแปลงเป็นขอบขาขึ้นได้โดยการวัดเพียงครั้งเดียว ในการวัดสิ่งนี้จะได้ออกมาก็คือ ค่าของ Sig และ Comp จากค่าสัญญาณขาเข้า ทั้งสองนี้เอง เรามาทำให้เป็นสเตทที่ย่อยลงไปอีกได้คือ



สังเกตได้ว่า ก้าวาง 00,01,11,10 ในลักษณะดังกล่าวแล้ว สัญญาณจากบนลงล่างจะมีโอกาสทำให้เกิดการเปลี่ยนเสตท แต่ลูกศรจากล่างขึ้นบน จะไม่ทำให้เกิดการเปลี่ยนเสตทเลย เพราะไม่มีขอบขาขึ้นของ Sig และ Comp การเขียนในรูปของสเตทไดอะแกรม จะมีข้อจำกัดว่าจะโยงเฉพาะสัญญาณที่เปลี่ยนแปลงเพียงอันเดียว ไม่สามารถแสดงสัญญาณที่เกิด จากการเปลี่ยนแปลงทั้งสองอันได้



จะมีสถานะทั้งหมดรวม 12 สถานะแบ่งเป็นกลุ่มตามค่าของเอาท์พุท ได้ 3 กลุ่มใหญ่ๆ และภายในกลุ่มนี้จะจัดกลุ่มย่อยโดยที่ถ้ามีการเชื่อมโยงถึงกันในกลุ่มเอาท์พุทเดียวกัน จะถือว่าอยู่ในกลุ่มย่อยเดียวกัน จากตัวเลขที่กำกับไว้จะมีกลุ่มดังนี้ (1,4), (2,3), (5), (6), (7), (8), (9,10), (11,12)

เมื่อนำมาเขียนเป็นตารางมี 8 แถวตามจำนวนกลุ่มย่อยโดยค่าสถานะที่เป็นเลขกำกับในรูปจะอยู่ในวงเล็บในตาราง ภายในแถวเดียวกันจะแบ่งออกเป็น 4 ช่องตามสัญญาณเข้า 2 ขา ซึ่ง 4 ช่องนี้จะมีอย่างน้อย 1 ช่องที่เป็นตัวสถานะ (มีวงเล็บ) ส่วนช่องที่เหลือถ้าจากแผนภูมิสามารถเชื่อมโยงไปยังสถานะอื่นได้ก็จะใส่ตัวเลขนั้นไว้ โดยมีเส้นใต้กำกับ ตารางจะมีช่องทั้งหมด 8 * 4 = 32 ช่อง เป็นตัวสถานะเสีย 12 ช่อง และมีการเชื่อมโยงตามรูปอีก 8 ช่อง จะเหลืออีก 12 ช่องที่ยังว่างอยู่ ให้ถือว่าเป็นการเปลี่ยนสถานะภายในกลุ่มเอาท์พุทเดียวกัน จะนำเอาค่าของสถานะที่อยู่ภายในกลุ่มเอาท์พุทเดียวกันที่มีค่า สัญญาณขาเข้าเหมือนกันมาใส่ไว้ตั้งแสดงไว้ตังตาราง

สัญญาณขาเข้า (R V)				สัญญาณขาออก
00	01	11	10	
(1)	6	7	(4)	1
1	(2)	(3)	4	1
(5)	10	7	4	อิมพีแดนซ์สูง
5	(6)	3	8	อิมพีแดนซ์สูง
5	6	(7)	8	อิมพีแดนซ์สูง
5	6	11	(8)	อิมพีแดนซ์สูง
(9)	(10)	7	8	0
9	10	(11)	(12)	0

สำหรับไอซีเบอร์ 4044 จะใช้ขอบขาลงในการเปลี่ยนสถานะแทนที่จะเป็นขอบขาขึ้นดังที่ได้กล่าวมาแล้วรูปแบบของแผนภูมิสถานะจะต่างออกไป แต่เป็นตัวตรวจจับเฟสที่มีการทำงานเหมือนกัน

ภาคผนวก B

รายละเอียดโปรแกรมที่ใช้ในการทดสอบเฟสดีเทคเตอร์

```
type
  BiInput = array [FALSE..TRUE,FALSE..TRUE] of byte;
  SigCurve = object
    tilen : byte;
    hilevel,midlevel,lolevel : string;
    oldlevel : boolean;

    constructor init(level : boolean;ch : string);
    procedure instate(inlevel : boolean); virtual;
    function getstate : boolean;
    procedure print; virtual;
  end;
const
  DownCon : BiInput = (($85,$83),($82,$20));
  UpCon   : BiInput = (($20,$80),($81,$85));
  DownInd : array [FALSE..TRUE] of char = (#$85,#32);
  UpInd   : array [FALSE..TRUE] of char = (#32,#$85);
  InputR : string = 'XXX.XXX.XXX.XXX.XXX.XXX.X!';
  InputV : string = '.XXXX....XXXX....XXXX....!';
  _Sig : byte = 0;
  _Comp : byte = 0;
  _Sign1 : byte = 0;
  _Compn1 : byte = 0;
  _QComp : byte = 0;
  _QSig : byte = 0;
```

```
var
  row,state : byte;
  i : integer;
  R,V,U,D : SigCurve;

procedure PhaseDetect; near; external;
{$L DFLIP.OBJ}
{!$L SGRAM.OBJ}

function space(n : byte):string;
var
  s : string;
begin
  fillchar(s[1],n,' ');
  s[0] := char(n);

  space := s;
end;

constructor SigCurve.Init(level : boolean;ch : string);
begin
  hilevel := space(length(ch))+UpInd  [level];
  lolevel := space(length(ch))+DownInd[level];
  midlevel := ch+' ';
  tilen := length(ch);
  oldlevel := level;
end;
```

```
procedure SigCurve.instate(inlevel : boolean);
begin
  hilevel := hilevel + char(UpCon  [oldlevel,inlevel])
                     + UpInd  [inlevel] + UpInd  [inlevel];
  lolevel := lolevel + char(DownCon[oldlevel,inlevel])
                     + DownInd[inlevel] + DownInd[inlevel];
  if oldlevel <> inlevel then midlevel := midlevel + #132'  '
                         else midlevel := midlevel + '    ';
  oldlevel := inlevel;
end;

function SigCurve.getstate : boolean;
begin
  getstate := oldlevel;
end;

procedure SigCurve.print;
var
  slo : string;

function cuthead(s : string;l : byte) : string;
var
  bcut : byte;
begin
  if length(s) > l then
  begin
    bcut := length(s) - l;
    Delete(s,tilen+1,bcut);
  end;
```

```
  cuthead := s;
end;
```

```
begin
  writeln(cuthead(hilevel,70));
  writeln(cuthead(midlevel,70));
  slo := cuthead(lolevel,70);
  if slo[tilen+1] = ' ' then slo[tilen+1] := #133;
  writeln(slo);
end;

begin
  R.init(InputR[1]='X','Sig   ');
  V.init(InputV[1]='X','Comp  ');
  i := 1;
  U.init(_QSig=1,'QSig  ');
  D.init(_QComp=1,'QComp ');
  while InputR[i] <> '!' do
  begin
    R.instate(InputR[i]='X');
    V.instate(InputV[i]='X');
    _Sig := Ord(R.getstate);
    _Comp := Ord(V.getstate);
    PhaseDetect;
    U.instate((_QSig AND 1) = 1);
    D.instate((_QComp AND 1) = 1);
    inc(i);
  end;
```

```
    R.print;
    V.print;
    writeln;
    U.print;
    D.print;
    writeln;
end.
```

SCL4046B
SCL4446B

CMOS PHASE-LOCKED LOOPS

FEATURES

- Very low power consumption — 70 μW (typ) @ f_o = 10kHz, 5Vdc
- Operating frequency range (no offset) — Up to 3MHz (typ) @ 10Vdc (SCL4046B) Up to 4MHz (typ) @ 10Vdc (SCL4446B)
- Low frequency drift — 0.04%/°C (typ) @ 10Vdc
- Choice of two phase comparators:
  1. Exclusive-OR network
  2. Edge-controlled memory network with phase-pulse output for lock indication
- VCO Inhibit control for ON-OFF keying and ultra-low standby power consumption
- High VCO linearity 1% (typ)
- Source-follower output of VCO control input (Demodulator Output)
- Zener Diode to assist Supply Regulation
- Balanced Output Drive Current Specifications

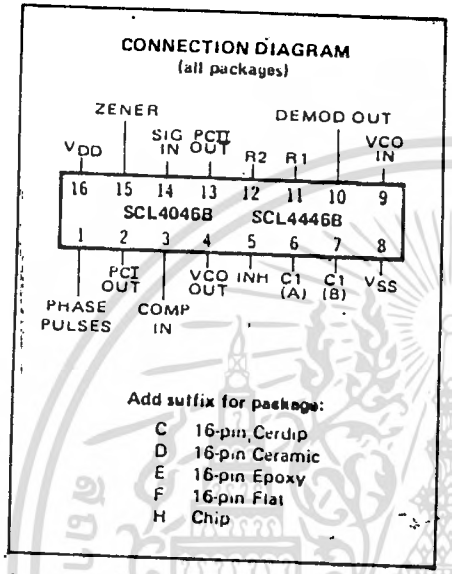
APPLICATIONS

- FM demodulator and modulator
- Frequency synthesis and multiplication
- Frequency discriminator
- Data synchronization
- Voltage-to-frequency conversion
- Tone decoding
- FSK-Modems
- Signal conditioning

DESCRIPTION

The SCL4046B and SCL4446B phase-locked loops contain two phase comparators, a voltage-controlled oscillator (VCO), source follower, and zener diode. The comparators have two common inputs. The Signal input can be used directly coupled to large voltage signals, or indirectly coupled (with a series capacitor) to small voltage signals. The self-bias circuit adjusts small voltage signals in the linear region of the amplifier. Phase comparator I (an exclusive-OR gate) provides a digital error signal PCI_{out}, and maintains 90° phase shift at the center frequency between Signal and Comparator inputs (both at 50% duty cycle). Phase comparator II (with leading edge sensing logic) provides digital error signals $PCII_{out}$ and Phase Pulses, and maintains a 0° phase shift between input signals (duty cycle is immaterial). The linear VCO produces an output signal VCO_{out} whose frequency is determined by the voltage of input VCO_{in} and the capacitor and resistors connected to pins $C1_A$, $C1_B$, R1, and R2. The source follower output, Demod Out, with an external resistor is used where the VCO_{in} signal is needed but no loading can be tolerated. The inhibit input Inh, when high, disables the VCO and source follower to minimize standby power consumption. The zener diode can be used to assist in power supply regulation.

CONNECTION DIAGRAM
(all packages)



Add suffix for package:

C	16-pin Cerdip
D	16-pin Ceramic
E	16-pin Epoxy
F	16-pin Flat
H	Chip

RECOMMENDED OPERATING CONDITIONS

For maximum reliability:

DC Supply Voltage	$V_{DD} - V_{SS}$	3 to 15	Vdc
Operating Temperature	T_A		
C, D, F, H Device		-55 to +125	°C
E Device		-40 to +85	°C

BLOCK DIAGRAM

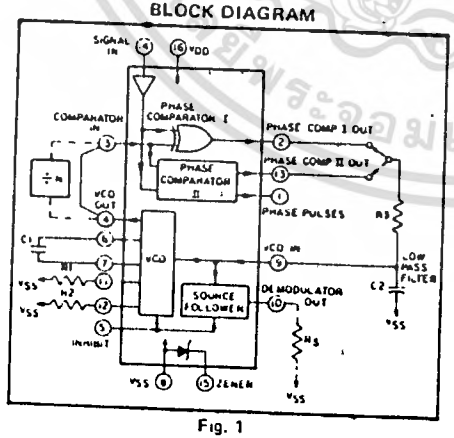


Fig. 1

VCO SECTION

The VCO requires one external capacitor (C1) and one to two external resistors (R1 or R1 and R2). Resistor R1 and capacitor C1 determine the frequency range of the VCO and resistor R2 enables the VCO to have a frequency offset if required. The high input impedance ($10^{12}\Omega$) of the VCO simplifies the design of low-pass filters by permitting the designer a wide choice of resistor-to-capacitor ratios. In order not to load the low-pass filter, a source-follower output of the VCO input voltage is provided at terminal 10 (DEMODULATOR OUTPUT). If this terminal is used, a load resistor (R_S) of 50kΩ or more should be connected from this terminal to V_{SS}. If unused, this terminal should be left open. The VCO can be connected directly or through frequency dividers to the comparator input of the phase comparators. A full CMOS logic swing is available at the output of the VCO. A logic 0 on the INHIBIT input "enables" the VCO and the source follower, while a logic 1 "turns off" both to minimize stand-by power consumption.

PHASE COMPARATORS

The phase-comparator signal input (terminal 14) can be direct-coupled provided the signal swing is within CMOS logic levels [logic "0" ≤ 30% (V_{DD}-V_{SS}), logic "1" ≥ 70% (V_{DD}-V_{SS})]. For smaller swings the signal must be capacitively coupled to the self-biasing amplifier at the signal input.

Phase comparator I is an exclusive-OR network; it operates analogously to an over-driven balanced mixer. To maximize the lock range, the signal and comparator-input frequencies must have a 50% duty cycle. With no signal or noise on the signal input, this phase comparator has an average output voltage equal to $V_{DD}/2$. The low-pass filter connected to the output of phase comparator I supplies the averaged voltage to the VCO input, and causes the VCO to oscillate at the center frequency (f_o).

The frequency range of input signals on which the PLL will lock, if it was initially out of lock, is defined as the frequency capture range ($2f_c$).

The frequency range of input signals on which the loop will stay locked if it was initially in lock is defined as the frequency lock range ($2f_L$). The capture range can not exceed the lock range.

With phase comparator I, the range of frequencies over which the PLL can acquire lock (capture range) is dependent on the low-pass-filter characteristics, and can be made as large as the lock range. Phase-comparator I enables a PLL system to remain in lock in spite of high amounts of noise in the input signal.

One characteristic of this type of phase comparator is that it may lock onto input frequencies that are close to harmonics of the VCO center-frequency. A second characteristic is that the phase angle between the signal and the comparator input varies between 0° and 180°, and is 90° at the center frequency. Figure 2 shows the (typical) triangular phase-to-output response characteristic of phase-comparator I. Typical waveforms for a CMOS phase-locked-loop employing phase comparator I in locked condition is shown in Figure 3.

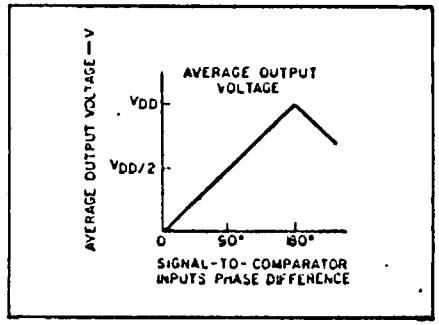


Fig. 2 — Phase-comparator I characteristics at low-pass filter output.

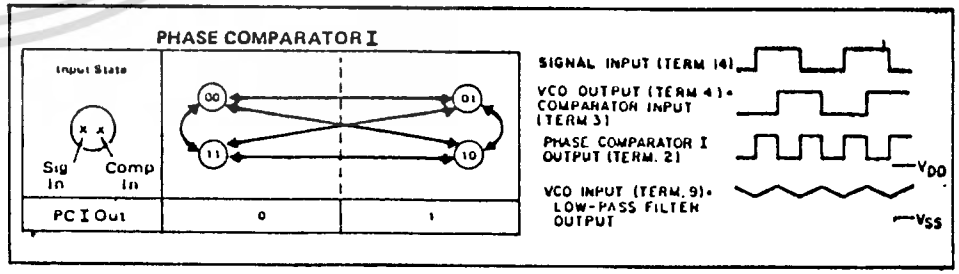


Fig. 3 — Typical waveforms employing phase comparator I in locked condition

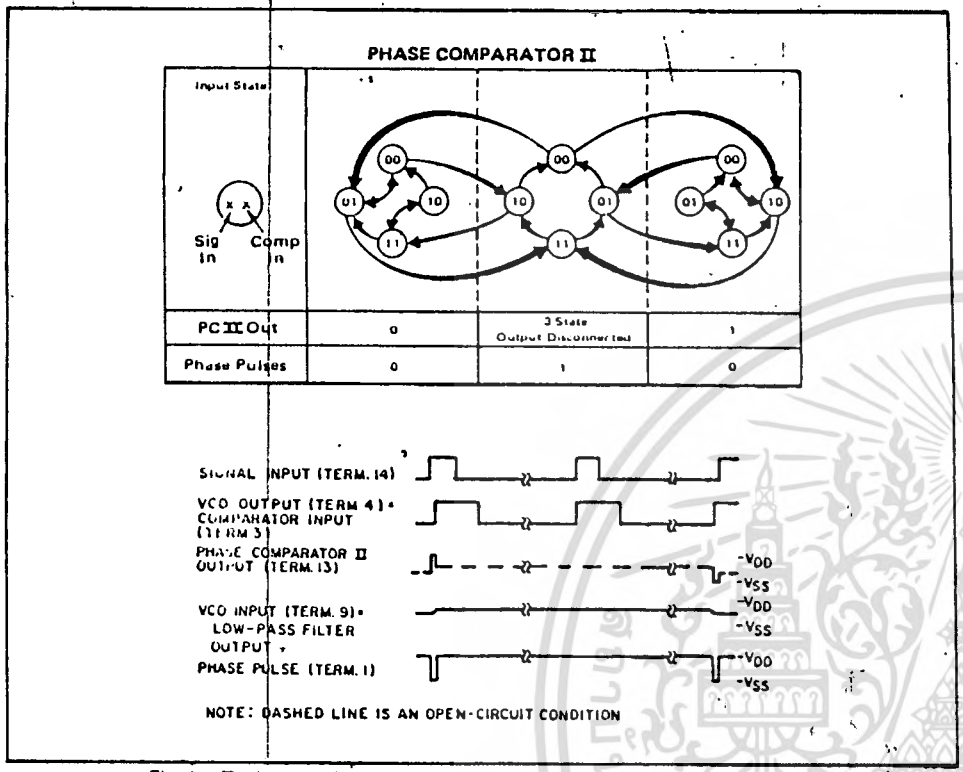


Fig. 4 — Typical waveforms employing phase comparator II in locked condition.

Phase-comparator II is an edge-controlled digital memory network. It consists of several flip-flop stages, control gating, and a three state output circuit comprising p- and n-type drivers having a common output node. When the p-MOS or n-MOS drivers are ON, they pull the output up to V_{DD} or down to V_{SS}, respectively. This type of phase comparator acts only on the positive edges of the signal and comparator inputs. The duty cycles of the signal and comparator inputs are not important since positive transitions control the PLL system utilizing this type of comparator. If the signal lags the comparator input in phase, the n-type output driver is maintained ON for a time corresponding to the phase difference. If the comparator input lags the signal in phase, the p-type output driver is maintained ON for a time corresponding to the phase difference. Subsequently, the capacitor voltage of the low-pass filter connected to this phase comparator is adjusted until the signal and comparator inputs are equal in both phase and frequency. At this stable point, both p- and n-type output drivers remain OFF. Thus, the phase comparator output becomes an open circuit and holds the voltage on the capacitor of the low-pass filter constant. Moreover, the signal at the "phase pulses" output is a high level which can be used for indicating a locked condition. Thus, for phase comparator II, no phase difference exists between signal and comparator input over the full VCO frequency range. Moreover, the power dissipation due to the low-pass filter is reduced when this type of phase comparator is used because both the p- and n-type output drivers are OFF for most of the signal input cycle.

It should be noted that the PLL lock range for this type of phase comparator is equal to the capture range, independent of the low-pass filter. With no signal present at the signal input, the VCO is adjusted to its lowest frequency for phase comparator II. Figure 4 shows typical waveforms for a CMOS PLL employing phase comparator II in a locked condition.

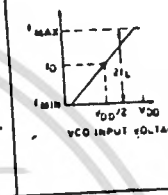
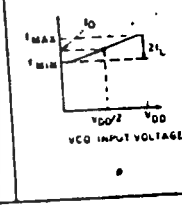
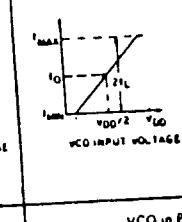
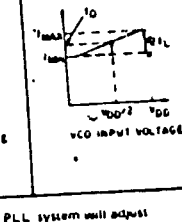
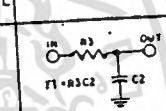
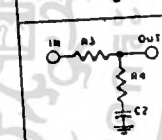
DESIGN INFORMATION

This information is a guide for approximating the values of external components for the SCL4046B and SCL4446B in a Phase-Locked Loop system. The selected external components must be within the following ranges:

$R1, R2 \geq 2k\Omega, R_S \geq 10k\Omega$

$C1 \geq 15pF$

In addition to the given design information refer to Figure 5 for R1, R2, and C1 component selections.

CHARACTERISTICS	USING PHASE COMPARATOR I		USING PHASE COMPARATOR II	
	VCO WITHOUT OFFSET $R_2 = \infty$	VCO WITH OFFSET	VCO WITHOUT OFFSET $R_2 = \infty$	VCO WITH OFFSET
VCO Frequency				
For No Signal Input	VCO in PLL system will adjust to center frequency, f_o		VCO in PLL system will adjust to lowest operating frequency, f_{min}	
Frequency Lock Range, $2f_L$	$2f_L$ = full VCO frequency range $2f_L = f_{max} - f_{min}$			
Frequency Capture Range, $2f_C$	 $f1 = R3C2$	$2f_C \approx \frac{1}{\pi}\sqrt{\frac{2\pi f_L}{f1}}$	$f_C = f_L$	
Loop Filter Component Selection		For $2f_C$ see Ref.		
Phase Angle between Signal and Comparator	90° at center frequency (f_o), approximating C° and 180° at ends of lock range ($2f_L$)		Always 0° in lock	
Locks on Harmonics of Center Frequency	Yes		No	
Signal Input Noise Rejection	High		Low	
VCO Component Selection	– Given: f_o – Use f_o with Fig.5a to determine R1 and C1.	– Given: f_o and f_L – Calculate f_{min} from the equation $f_{min} = f_o - f_L$ – Use f_{min} with Fig. 5b to determine R2 and C1 – Calculate $\frac{f_{max}}{f_{min}}$ from the equation $\frac{f_{max}}{f_{min}} = \frac{f_o + f_L}{f_o - f_L}$ – Use $\frac{f_{max}}{f_{min}}$ with Fig.5c to determine ratio R2/R1 to obtain R1	– Given: f_{max} – Calculate f_o from the equation $f_o = \frac{f_{max}}{2}$ – Use f_o with Fig.5a to determine R1 and C1	– Given f_{min} & f_{max} – Use f_{min} with Fig.5b to determine R2 and C1 – Calculate $\frac{f_{max}}{f_{min}}$ – Use $\frac{f_{max}}{f_{min}}$ with Fig.5c to determine ratio R2/R1 to obtain R1

REF. G. S. Moschytz, "Miniaturized RC Filters Using Phase-Locked Loop", BSTJ, May, 1965.

ELECTRICAL CHARACTERISTICS [1,4]

PARAMETER		V_{DD} (Vdc)	CONDITIONS	T_{LOW} [2] Min.	T_{LOW} [2] Max.	+25°C Min.	+25°C Typ.	+25°C Max.	T_{HIGH} [2] Min.	T_{HIGH} [2] Max.	Units
QUIESCENT DEVICE CURRENT	I_{DD}	5	Inhibit = V_{DD}	–	5	–	0.05	5	–	150	μAdc
		10	Signal Input =	–	10	–	0.01	10	–	300	
		15	V_{DD}	–	20	–	0.2	20	–	600	
TOTAL POWER DISSIPATION	P_D		Inh V_{SS}, $VCO_{IN} = \frac{V_{DD}}{2}$, f_o = 10kHz, C_L = 15pF, R1 = 1MΩ, R2 = R_S = ∞								
		5		–	–	–	0.07	–	–	–	mW
		10		–	–	–	0.6	–	–	–	
		15		–	–	–	2.4	–	–	–	

NOTES:
1. Remaining Static Electrical Characteristics are listed under "SCL4000B Series Family Specifications".
2. T_{Low} = -55°C for C, D, F, H device.
   = -40°C for E device.
   T_{High} = +125°C for C, D, F, H device.
   = + 85°C for E device.
3. VCO output (pin 4) and Phase Comparator Outputs (pins 2 and 13) have been designed for balanced output drive current specifications. Consult Family Specifications.

PARAMETER		CONDITIONS		V_{DD}	25°C Min.	25°C Typ.	25°C Max.	UNIT
VCO SECTION								
MAXIMUM OPERATING FREQUENCY	f_{max}		R1 C1					
SCL4046B		R2 = ∞, VCO_{IN} = V_{DD}	10k 50pF	5	0.5	0.8	–	MHz
				10	1.0	1.5	–	
				15	1.3	1.9	–	
			5k 50pF	5	0.6	1.0	–	MHz
				10	1.4	2.1	–	
				15	1.8	2.7	–	
			2k 50pF	5	–	1.3	–	MHz
				10	–	2.9	–	
				15	–	3.8	–	
SCL4446B		R2 = ∞, VCO_{IN} = V_{DD}	R1 C1					
			10k 50pF	5	0.7	1.0	–	MHz
				10	1.3	2.0	–	
				15	1.9	2.8	–	
			5k 50pF	5	0.9	1.3	–	MHz
				10	1.9	2.9	–	
				15	2.6	3.9	–	
			2k 50pF	5	–	1.8	–	MHz
				10	–	3.9	–	
				15	–	5.4	–	
LINEARITY		R2 = ∞, VCO_{IN} = 2.5±0.3V, R1 ≥ 10kΩ		5		1	–	%
		VCO_{IN} = 5.0±2.5V, R1 ≥ 400kΩ		10	–	1	–	
		VCO_{IN} = 7.5±5.0V, R1 ≥ 1MΩ		15	–	1	–	

ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		CONDITIONS	V_{DD}	+25°C Min.	+25°C Typ.	+25°C Max.	UNIT
VCO SECTION (Continued)							
TEMPERATURE-FREQUENCY STABILITY							
No Offset		R2 = ∞	5	–	0.12-0.24	–	%/°C
			10	–	0.04-0.08	–	
			15	–	0.015-0.03	–	
With Offset		R2 ≤ 10X R1	5	–	0.06-0.12	–	%/°C
			10	–	0.05-0.1	–	
			15	–	0.03-0.06	–	
INPUT RESISTANCE (VCO_{IN})	R_{IN}		5, 10, 15	–	10^6	–	MΩ
OUTPUT DUTY CYCLE		All valid input combinations and voltages		–	50	–	%
OUTPUT TRANSITION TIME	t_{TLH}, t_{THL}	C_L = 50pF	5	–	100	200	ns
			10	–	50	100	
			15	–	40	80	
PHASE COMPARATORS							
INPUT RESISTANCE							
Signal Input	R_{IN}		5	1	3	–	MΩ
			10	0.2	0.7	–	
			15	0.1	0.3	–	
Comparator Input	R_{IN}		5, 10, 15	–	10^6	–	MΩ
AC-COUPLED INPUT SENSITIVITY	V_{IN}						
Signal Input			5	–	200	400	mV
			10	–	400	800	
			15	–	700	1400	
OUTPUT TRANSITION TIME							
PCI, PCII Outputs	t_{TLH}, t_{THL}	C_L = 50pF	5	–	100	200	ns
			10	–	50	100	
			15	–	40	80	
Phase Pulses Output	t_{TLH}, t_{THL}		5	–	130	260	ns
			10	–	65	130	
			15	–	50	100	
DEMODULATOR OUTPUT							
OFFSET VOLTAGE	VCO_{IN}-V_{DEM}	R_S ≥ 50kΩ	5	–	1.4	2.2	Vdc
			10	–	1.6	2.2	
			15	–	1.8	2.2	
LINEARITY		R_S ≥ 50kΩ					
		VCO_{IN} = 2.5±0.3V	5	–	0.1	–	%
		VCO_{IN} = 5.0±2.5V	10	–	0.6	–	
		VCO_{IN} = 7.5±5.0V	15	–	0.8	–	
ZENER DIODE							
ZENER VOLTAGE	V_Z	I_Z = 50μA	–	6.3	7.0	7.7	V
DYNAMIC RESISTANCE	R_Z	I_Z = 1mA	–	–	100	–	Ω

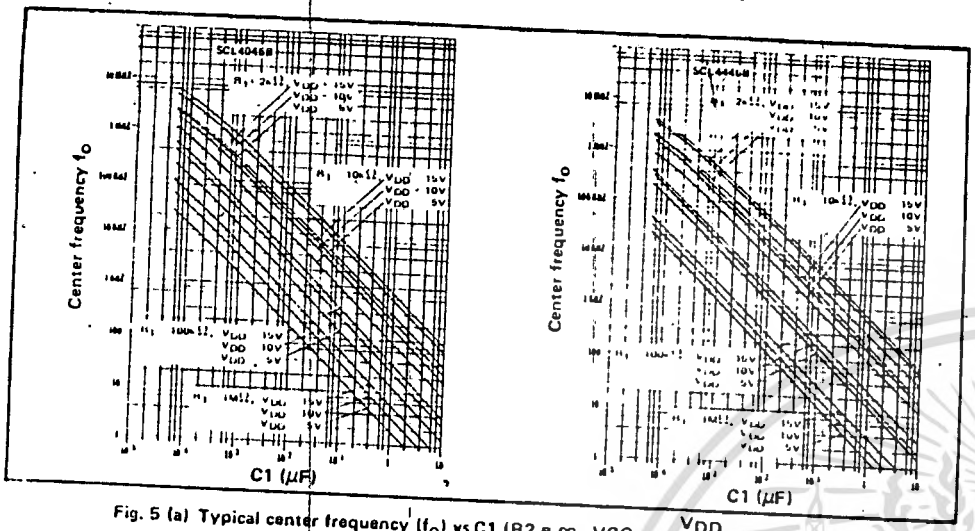


Fig. 5 (a) Typical center frequency (f_o) vs C1 ($R2 = \infty$, $VCO_{IN} = \frac{V_{DD}}{2}$, $T_A = 25°C$)

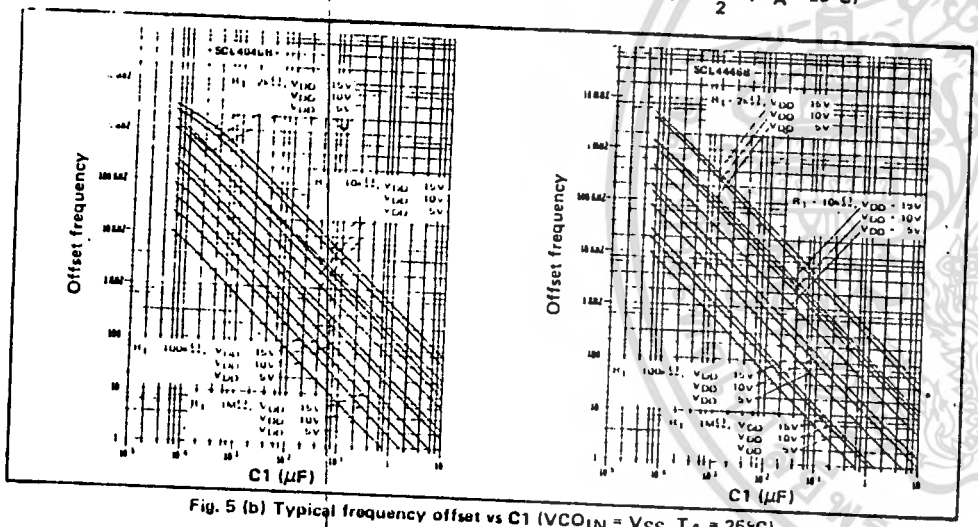


Fig. 5 (b) Typical frequency offset vs C1 ($VCO_{IN} = V_{SS}$, $T_A = 25°C$)

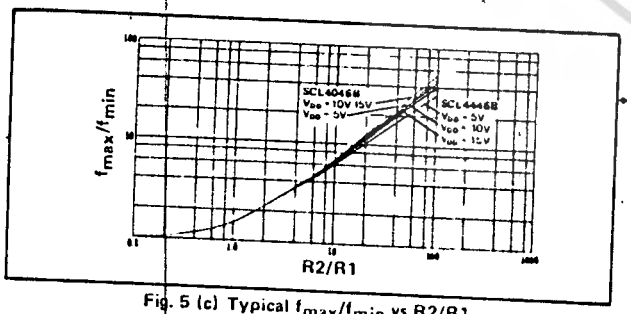


Fig. 5 (c) Typical f_{max}/f_{min} vs R2/R1

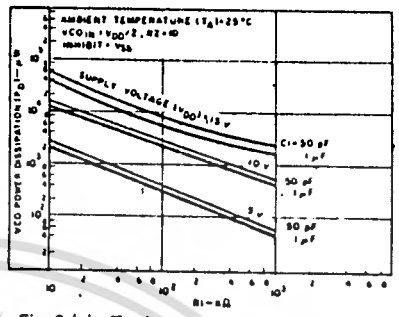


Fig. 6 (a) - Typical VCO power dissipation at center frequency vs R1.

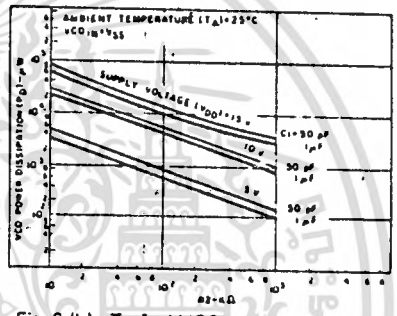


Fig. 6 (b) - Typical VCO power dissipation at f_{min} vs R2.

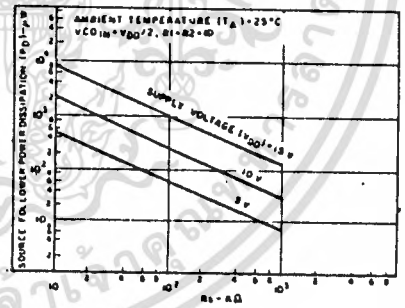


Fig. 6 (c) - Typical source follower power dissipation vs R_S.

NOTE: To obtain approximate total power dissipation of PLL system for no-signal input

P_D (Total) = P_D (f_o) + P_D (f_{MIN}) + P_D (R_S) – Phase Comparator I

P_D (Total) = P_D (f_{MIN}) – Phase Comparator II

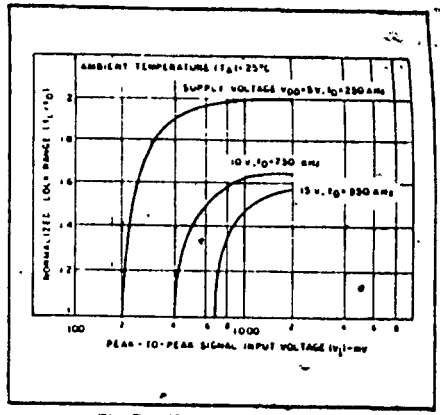


Fig. 7 – Typical lock range vs signal input amplitude

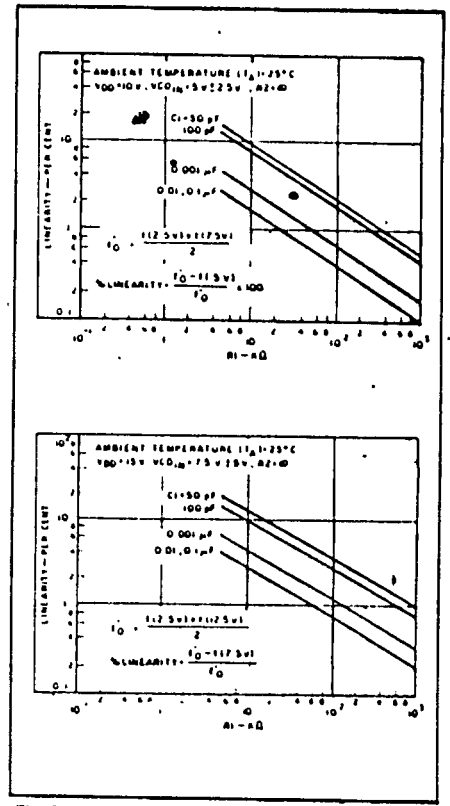


Fig. 8(a, b) – Typical VCO linearity vs R1 and C1