อัลตร้าโซนิค CT คอนโทลเลอร์

ULTRASONIC CT CONTROLLER

โดย

นายชัชวาลย์ สุวรรณสิงห์

นายวิทูร เกี้ยวม่าน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

อัลตร้าโซนิค CT คอนโทลเลอร์

ULTRASONIC CT CONTROLLER

โดย

นายชัชวาลย์ สุวรรณสิงห์ รหัสประจำตัว 43515909

นายวิทูร เกี้ยวม่าน รหัสประจำตัว 43515930

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

อัลตร้าโซนิค CT คอนโทลเลอร์

ULTRASONIC CT CONTROLLER

นาย ชัชวาลย์ สุวรรณสิงห์ 43515909

นาย วิทูร เกี้ยวม่าน 43515930

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร. มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

ปริญญานิพนธ์ ปีการศึกษา 2545

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง อัลตร้าโซนิค CT คอนโทลเลอร์

ผู้จัดทำ

1. นาย ชัชวาลย์ สุวรรณสิงห์
2. นาย วิทูร เกี้ยวม่าน

.....อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรชิลป์)

อัลตร้าโซนิค CT คอนโทลเลอร์

นาย ชัชวาลย์ สุวรณสิงห์ 43515909
นาย วิทูร เกี้ยวม่าน 43515930
รศ.ดร. มนัส สังวรศิลป์
ปีการศึกษา 2545

บทคัดย่อ

วัตถุประสงค์ของโครงงานนี้เพื่อสร้างชุดควบคุมอัลตร้าโซนิค CT โดยจะสั่งงานผ่านคอมพิวเตอร์โดยใช้ มาตรฐาน IEEE 488 (GPIB) ชุดควบคุมจะประกอบไปด้วย ไมโครคอลโทรลเลอร์ (MCS-51) ซึ่งจะทำหน้าที่ควบคุมการทำงานของมอเตอร์ที่ใช้ในการควบคุม หัวรับ และ หัวส่งของอัลตร้าโซนิค จากนั้นนำสัญญาณที่วัดได้มาเก็บ และ แสดงผลบนคอมพิวเตอร์เพื่อใช้ในการสร้างภาพตัดขวางต่อไป

จากเครื่องต้นแบบชุดนี้ สามารถทำได้ความถูกต้องของสัญญาณที่ทำการวัดและตำแหน่งของหัวรับและหัวส่งถูกต้องสูง

ULTRASONIC CT CONTROLLER

Mr.Chatchawan Suwanasing 43515909

Mr.Witoon Kiewman 43515930

Assoc.Prof.Dr.Manas Sangwarasilp(Advisor)

Year 2002

ABSTRACT

This paper describes the constructions of Ultrasonic CT controller. A computer is used to control the system via IEEE 488 : The General purpose Interface Bus (GPIB) strandard. It utilizes the microcontroller (MCS-51) to control two motors for a transceiver and a receiver ultrasonic head, and to interface with the computer. The measured signal are stored and displayed on the compute using the particular software developed for this system.

From this project, the high precision of the measured signal and the position of the sensor heads can, therefore, be obtained.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

ในปัจจุบันไมโครคอนโทลเลอร์ MCS-51 ได้รับความนิยมอย่างมากในการนำไปศึกษาและนำไปใช้งานกันอย่างแพร่หลาย จนอาจกล่าวได้ว่าได้รับความนิยมมากที่สุด สินค้าและผลิตภัณฑ์จำนวนมากจะใช้ไมโครคอนโทลเลอร์ตระกลูนี้ในการควบคุมการทำงาน เพราะเป็นอุปกรณ์ที่มีความนิยม หาซื้อได้ง่ายและราคาไม่แพง

ด้วยเหตุนี้ทางผู้จัดทำจึงได้นำไมโครคอนโทลเลอร์ตระกลูนี้มาทำการควบคุมการทำงานของมอเตอร์และสร้างความถี่ในย่านอัลตร้าโซนิค เพื่อทำการสแกนวัตถุ โดยสัญญาณที่ได้นี้นักศึกษาในระดับปริญญาโทจะนำไปวิเคราะห์และวิจัยเป็นผลงานต่อไป

1.2 ขอบเขต

1.2.1 ทำการออกแบบและสร้างวงจรกำเนิดอัลตร้าโซนิค

1.2.2 สร้างชุดควบคุมมอเตอร์เครื่องสแกนอัลตร้าโซนิคโดยไมโครคอนโทลเลอร์ MCS-51

1.2.3 ใช้โปรแกรม Delphi สร้างโปรแกรมควบคุมการทำงานของเครื่องสแกนอัลตร้าโซนิคผ่านทางพอร์ตอนุกรม

1.2.4 สร้างวงจรจ่ายไฟสูงสำหรับการส่งคลื่นอัลตร้าโซนิค

บทที่ 2
การใช้งาน MCS –51

2.1 จัดขาของไมโครคอนโทรลเลอร์ 8051

V_{cc} : สำหรับแหล่งจ่ายไฟฟ้า (+5v.)

V_{ss} : สำหรับต่อกราวด์

P0 : เป็นขาพอร์ต 0 ของ 8051 ที่มีขนาด 8 บิตชนิดสองทิศทาง ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังบิตนั้น โดยแต่ละบิตเมื่อเป็นเอาต์พุตจะสามารถต่อพ่วงกับอุปกรณ์ TTL แบบ LS ได้ 8 ตัว และยังเป็นขาให้สัญญาณ Multiplex ระหว่างสัญญาณข้อมูลกับสัญญาณ Address 8 บิตแรก ในกรณีที่ใช้หน่วยความจำภายนอก

P1 : เป็นขาพอร์ต 1 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังบิตนั้น และสามารถต่อพ่วงกับอุปกรณ์ LS TTL ได้ 4 ตัว

P2 : เป็นขาพอร์ต 2 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับพอร์ต 1 นอกจากนี้พอร์ต2นี้ยังทำหน้าที่ให้สัญญาณ Address 8 บิตบน ในกรณีที่ใช้หน่วยความจำภายนอก ในกรณีอ้าง Address หน่วยความจำขนาด 16 บิต ดังนั้นขณะที่ใช้หน่วยความจำภายนอก จะต้องไม่มีการเขียนข้อมูลใด ๆ ไปที่พอร์ต 2 จะทำให้เกิดความผิดพลาดการทำงานได้

P3 : เป็นขาพอร์ต 3 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับขาพอร์ต 1 และพอร์ต 2 แต่พอร์ต 3 นี้จะมีหน้าที่พิเศษดังตารางที่ 2.1

ขาพอร์ต	หน้าที่พิเศษ
P3.0	R x D (สำหรับรับข้อมูลแบบอนุกรม)
P3.1	T x D (สำหรับส่งข้อมูลแบบอนุกรม)
P3.2	INT0 (ขาอินเตอร์รัพท์ภายนอก 0)
P3.3	INT1 (ขาอินเตอร์รัพท์ภายนอก 1)
P3.4	T0 (ขาอินพุตของ Timer 0)
P3.5	T1 (ขาอินพุตของ Timer 1)
P3.6	WR (สำหรับสัญญาณเขียนหน่วยความจำข้อมูลภายนอก)
P3.7	RD (สำหรับสัญญาณอ่านหน่วยความจำข้อมูลภายนอก)

ตารางที่ 2.1 หน้าที่พิเศษของขาต่าง ๆ ของ PORT 3

ดังนั้น เมื่อมีการใช้สัญญาณดังกล่าว จึงไม่ควรเขียนข้อมูลไปที่พอร์ต 3 จะทำให้การทำงานของ 8051 ผิดพลาดได้

RST : เป็นขาสำหรับรีเซตการทำงานของ 8051 โดยการให้ลอจิกหนึ่งเป็นเวลาอย่างน้อย 2 ช่วง Machine Cycle

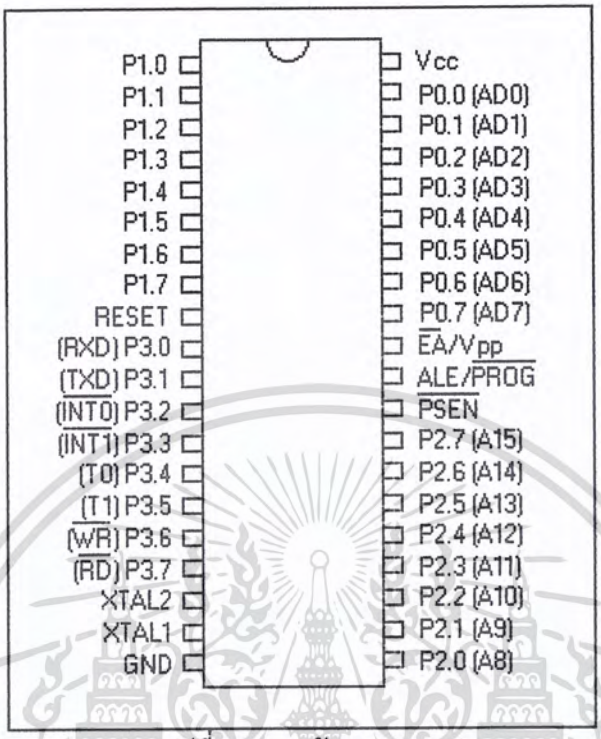
ALE : เป็นขาที่ใช้ในการควบคุมการแลตช์ของขา พอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก

PSEN : เป็นขาสัญญาณเพื่อร้องขอติดต่อหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านหน่วยความจำโปรแกรมภายนอก

EA : เป็นขาใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในไมโครคอนโทรลเลอร์ โดยที่ให้ลอจิก 0 จะอ่านหน่วยความจำโปรแกรมภายนอก และลอจิก 1 จะอ่านหน่วยความจำโปรแกรมภายใน

XTAL1 : ขาเข้าของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051

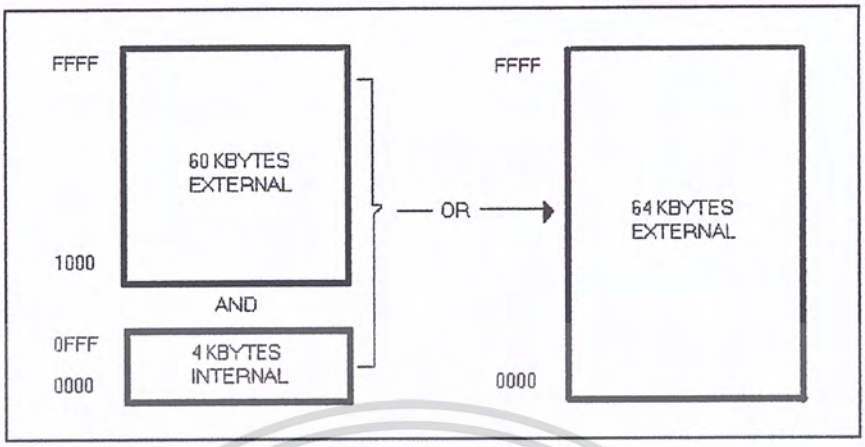
XTAL2 : ขาออกของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051



รูปที่ 2.1 การจัดขาของ 8051

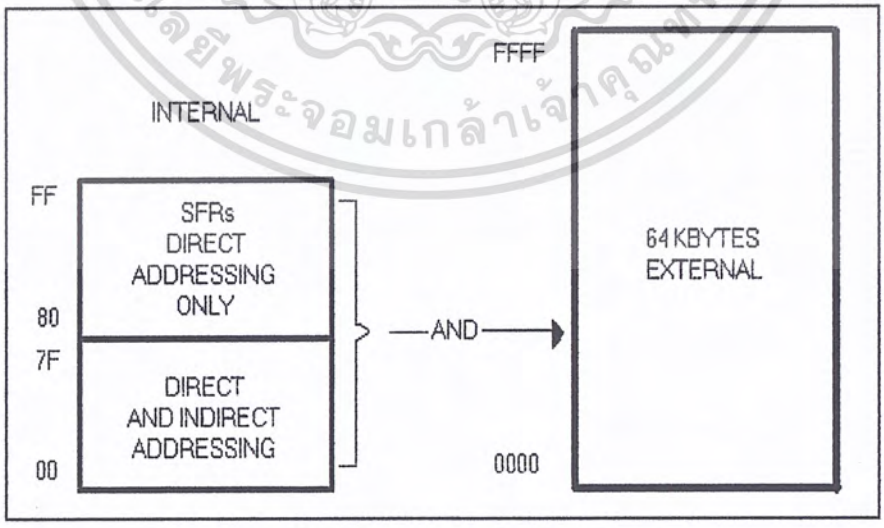
2.2 โครงสร้างหน่วยความจำของ 8051

ดังที่กล่าวมาแล้ว 8051 จะแบ่งหน่วยความจำออกเป็นสองส่วน ได้แก่ หน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับเก็บข้อมูล โดยมีขนาดของแต่ละส่วนเท่ากับ 64 กิโลไบต์ ในส่วนของหน่วยความจำโปรแกรมจะเป็นส่วนหน่วยความจำสำหรับอ่านอย่างเดียว โดยที่ 8051 จะใช้สัญญาณ PSEN ในการอ่านเท่านั้น แต่หน่วยความจำข้อมูลของ 8051 จะสามารถอ่านและเขียนได้โดยใช้สัญญาณ RD และ WR ตามลำดับ แต่อย่างไรก็ตาม ผู้ใช้สามารถรวมหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลเข้าด้วยกันได้ โดยนำสัญญาณ RD และ PSEN มาต่อเข้าวงจรแอนเกท สำหรับสร้างสัญญาณในการอ่านหน่วยความจำ นอกจากนี้หน่วยความจำโปรแกรมยังแบ่งออกเป็นภายนอกและภายในของ 8051 ดังแสดงในรูปที่ 2.2 รูปที่ 2.3 โดยรูปที่ 2.2 แสดงหน่วยความจำโปรแกรมในกรณีที่เลือกใช้หน่วยความจำภายนอกและภายใน ในด้านซ้ายมือเป็นส่วนหนึ่งของหน่วยความจำโปรแกรมภายในที่มีขนาด 4 กิโลไบต์ของ 8051 ส่วนที่เหลือจะเป็นหน่วยความจำภายนอก ส่วนด้านขวามือแสดงหน่วยความจำโปรแกรมเมื่อเลือกให้ติดต่อหน่วยความจำภายนอกทั้งหมด



รูปที่ 2.2 แสดงหน่วยความจำโปรแกรมของ 8051

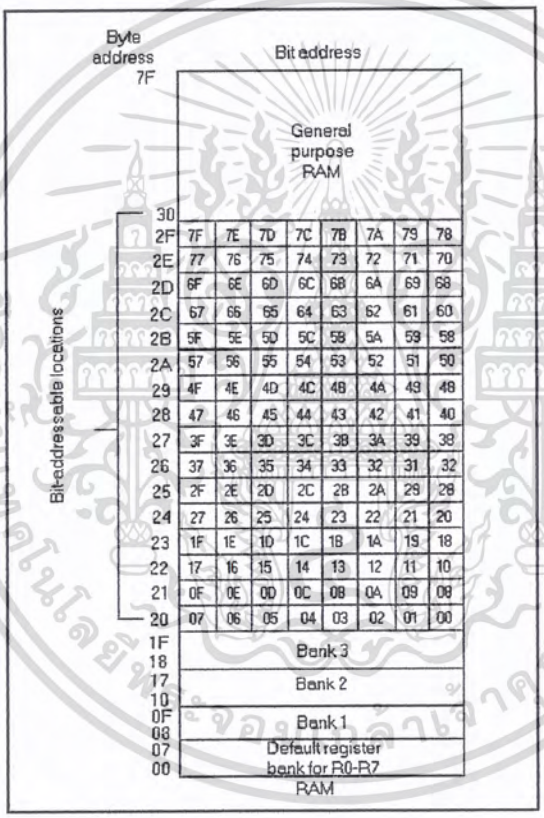
สำหรับหน่วยความจำข้อมูลของ 8051 สามารถแบ่งออกเป็นภายนอกและภายใน โดยหน่วยความจำภายนอกแสดงไว้ด้านขวามือของรูปที่ 2.3 ซึ่งมีขนาด 64 กิโลไบต์ ส่วนหน่วยความจำข้อมูลภายในแสดงไว้ด้านซ้ายของรูปที่ 2.3 โดยหน่วยความจำภายในของ 8051 แบ่งออกเป็นสองส่วน ได้แก่ ส่วนของหน่วยความจำข้อมูลที่สามารถอ้างอิงแบบ Direct และ Indirect ซึ่งมีขนาด 128 ไบต์ กับหน่วยความจำที่อ้างอิงได้เฉพาะแบบ Direct หรือในส่วนนี้จะเรียกอีกแบบหนึ่งว่า SFR (Special Function Register) โดยจะแบ่งกล่าวได้ดังนี้



รูปที่ 2.3 แสดงหน่วยความจำข้อมูลของ 8051

ในส่วนของหน่วยความจำข้อมูลภายในที่อ้างอิงแบบ direct และ Indirect นั้นจะสามารถแบ่งออกได้ 3 ส่วน ดังแสดงในรูปที่ 2.4 โดยมีรายละเอียดดังนี้

- ส่วนที่ 1 เรียกว่า Register Banks 0-3 ซึ่งอยู่ที่ตำแหน่งความจำข้อมูลภายใน ตั้งแต่ 00H ถึง 1FH จำนวน 32 ไบต์ โดยจะแบ่งออกเป็นชุด ชุดละ 8 ไบต์จำนวน 4 ชุด ซึ่งแต่ละชุดจะมีชื่อเรียกเป็น R0 ถึง R7 จะเป็น Register ที่ใช้งาน โดยเมื่อ 8051 ถูกรีเซต Register Bank 0 จะถูกเลือกใช้
- ส่วนที่ 2 เรียกว่า Bit Addressable Area ซึ่งมีขนาด 16 ไบต์ที่ตำแหน่งหน่วยความจำข้อมูล 20H ถึง 2FH ในส่วนนี้สามารถที่จะอ้างอิงข้อมูลได้เป็นระดับบิตถึง 128 บิต โดยการอ้างอิงตำแหน่งโดยตรงในลักษณะบิต ตั้งแต่ตำแหน่ง 00H ถึง 7FH



รูปที่ 2.4 แสดงหน่วยความจำข้อมูลภายใน

- ส่วนที่ 3 เรียกว่า Scratch Pad Area จะอยู่ที่ตำแหน่งตั้งแต่ 30H ถึง 7FH ซึ่งเป็นบริเวณหน่วยความจำข้อมูลภายในเอนกประสงค์ที่ผู้ใช้สามารถใช้ได้โดยตรง นอกจากนี้ยังสามารถใช้หน่วยความจำข้อมูลบริเวณนี้สำหรับการเก็บข้อมูลแบบ Stack ได้ด้วย

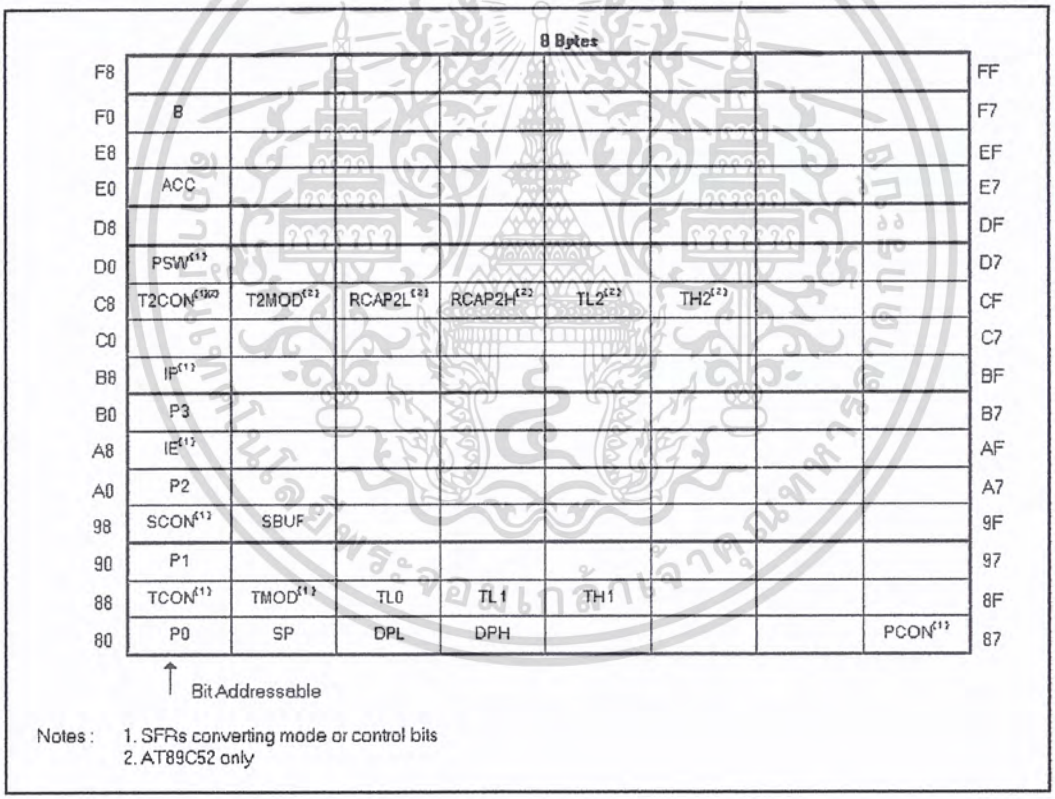
ในส่วนของหน่วยความจำข้อมูลภายในที่ใช้อ้างอิงแบบ Direct เพียงอย่างเดียวหรือที่เรียกว่า SFR ซึ่งเป็นส่วนสำหรับเก็บหรือกำหนดการทำงานภายในของ 8051 ดังแสดงในรูปที่ 2.5

ในส่วนของบริเวณนี้จะมีขนาด 128 ไบต์แต่ในการใช้งานนั้นใช้ได้เฉพาะตำแหน่งซึ่งแสดงไว้ในรูปที่ 2.5เท่านั้น หากผู้ใช้อ้างตำแหน่งนอกเหนือจากนี้นั้นจะได้ข้อมูลที่คาดเดาไม่ได้ โดยแต่ละตำแหน่งจะมีหน้าที่ดังนี้

ACC : เป็น Accumulator ซึ่งเป็นรีจีสเตอร์สำหรับการประมวลผลทางคณิตศาสตร์และลอจิก โดยผู้ใช้สามารถอ้างอิงได้ในรูปแบบของไบต์หรือระดับบิตได้

B : เป็นรีจีสเตอร์พิเศษสำหรับใช้กับคำสั่งในการคูณหรือหาร นอกจากนี้ยังใช้เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลได้

PSW : เป็นรีจีสเตอร์ Program Status Word หรือแฟลกจะแสดงสถานะการทำงานของ 8051 สำหรับการตรวจสอบซึ่งจะอธิบายรายละเอียดในภายหลัง



รูปที่ 2.5 แสดงรายละเอียดของ Special Function Register

SP : เป็นรีจีสเตอร์สำหรับชี้หน่วยความจำข้อมูลภายในสำหรับการเก็บแบบ Stack

DPTR : เป็นรีจีสเตอร์ขนาด 16 บิต โดยแบ่งเป็น 8 บิตบนและ 8 บิตล่าง ให้สำหรับชี้ตำแหน่งของหน่วยความจำข้อมูลภายนอกหรือสำหรับการอ่านตารางข้อมูลของหน่วยความจำโปรแกรม

P0 : เป็นรีจีสเตอร์สำหรับพอร์ต 0 ของ 8051

P1 : เป็นรีจีสเตอร์สำหรับพอร์ต 1 ของ 8051

P2 : เป็นรีจีสเตอร์สำหรับพอร์ต 2 ของ 8051

P3 : เป็นรีจีสเตอร์สำหรับพอร์ต 3 ของ 8051

IP : เป็นรีจีสเตอร์สำหรับกำหนดลำดับความสำคัญของการอินเตอร์รัพท์ของ 8051

IE : เป็นรีจีสเตอร์สำหรับกำหนดการรับหรือไม่รับการอินเตอร์รัพท์ของ 8051

TMOD : เป็นรีจีสเตอร์สำหรับควบคุมหน้าที่ของ Timer/Counter ของ 8051

TCON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter ของ 8051

T2CON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter 2 ของ 8052

TH0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตบน

TL0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตล่าง

TH1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตบน

TL1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตล่าง

TH2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตบนของ 8052

TL2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตล่างของ 8052

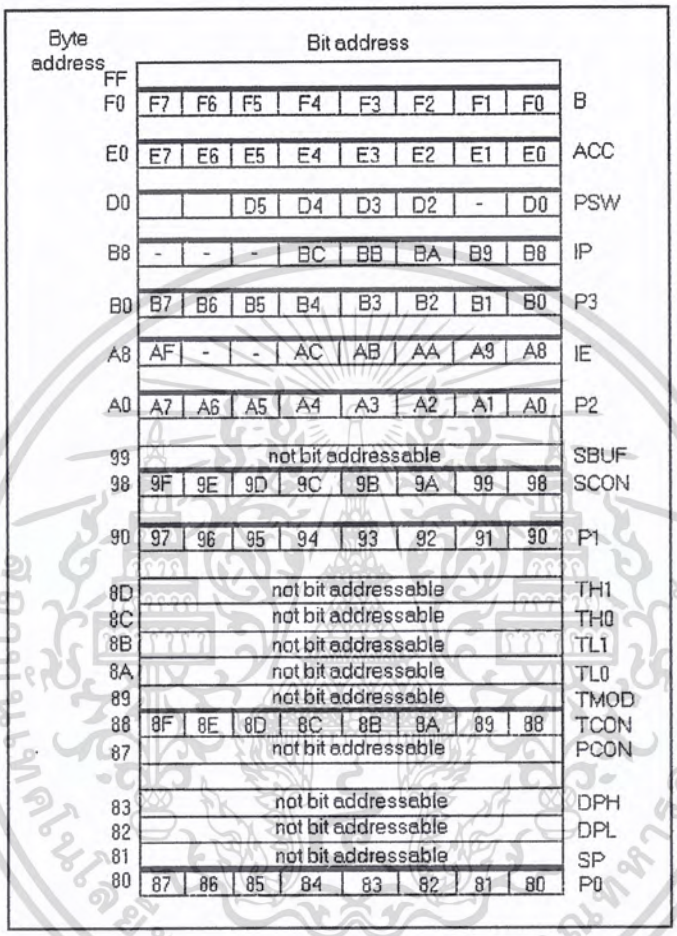
RCAP2H : เป็น Capture Register ของ Timer/Counter 2 8บิตบนของ 8052

SCON : เป็นรีจีสเตอร์สำหรับควบคุมการรับส่งข้อมูลแบบอนุกรมของ 8051

SBUF : เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลที่ได้จากการรับส่งข้อมูลแบบอนุกรมของ 8051

PCON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ 8051 ด้านเกี่ยวกับการใช้กำลังไฟฟ้า

ในส่วนของรีจีสเตอร์ SFR นี้สามารถที่จะอ้างอิงในระดับบิตได้โดยตำแหน่งการอ้างอิงระดับบิตแสดงไว้ในตารางต่อไปนี้



รูปที่ 2.6 แสดงตำแหน่งการอ้างอิงระดับบิตของรีจิสเตอร์ SFR

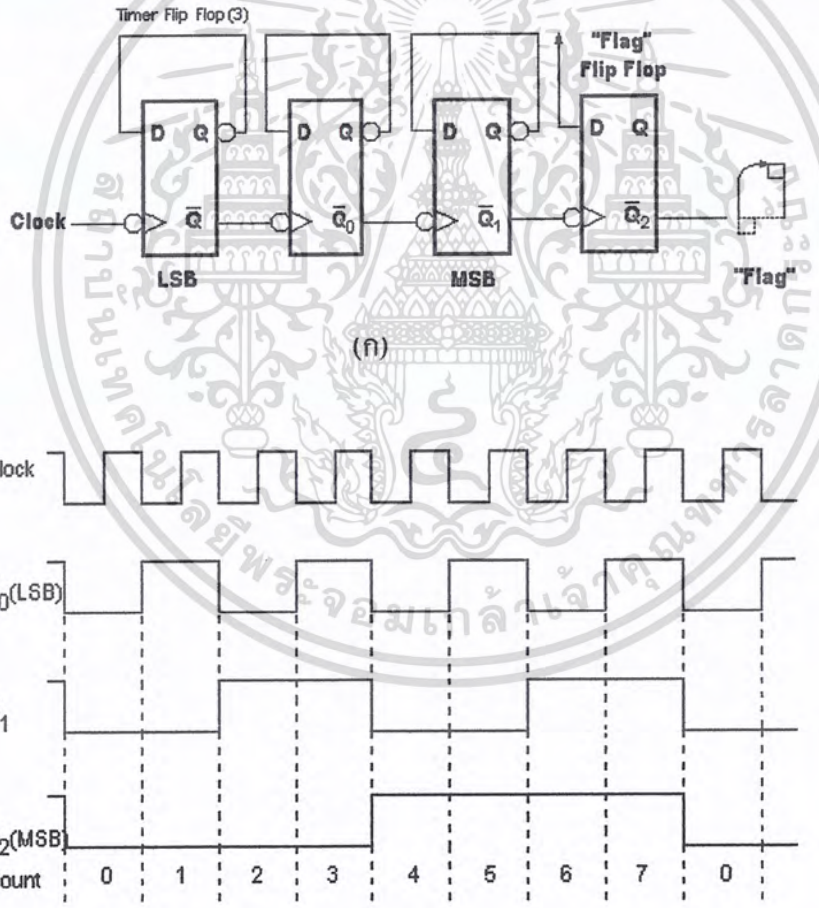
2.3 TIMER

ตัว Timer อาจพิจารณาได้ง่าย ๆ ว่าเป็นตัวฟลิปฟลอปมาต่อเรียงกันโดยมี Clock เป็นอินพุต สำหรับเอาต์พุตที่ออกมาจากฟลิปฟลอปแต่ละตัวจะถูกหารด้วย 2 พิจารณาการต่อฟลิปฟลอปตามรูปที่ 2.7 ถ้าใส่ Clock เข้าไปในฟลิปฟลอปตัวแรก ความถี่ของ Clock ที่ออกจากเอาต์พุตตัวแรกจะถูกหารด้วย 2 และเอาต์พุตนี้จะต่อกับฟลิปฟลอปตัวที่สอง และสัญญาณที่ออกมาจะถูกหารด้วย 2 อีก ดังนั้น ถ้ามีฟลิปฟลอปต่ออยู่ n Stages จะหารสัญญาณนาฬิกาได้ 2^n ถ้าให้เอาต์พุต Stage สุดท้ายของ Timer เป็น Overflow Flip-Flop หรือ Flag และจะให้เอาต์พุตออกมาเมื่อการนับเป็น Overflow เช่น ถ้าเป็นตัวนับ

แบบ 16 บิต (มีฟลิปฟลอปต่ออยู่ 16 ตัว) วงจรจะนับตั้งแต่ 0000H ถึง FFFFH เมื่อฟลิปฟลอปเปลี่ยนจาก FFFFH เป็น 0000H จะให้บิต Overflow ออกมา

พิจารณารูป 2.7(ก) เป็น 3-bit Timer โดยฟลิปฟลอปแต่ละตัวจะนำขา Q มาต่อกับ D ซึ่งอาจเรียกว่าเป็นการใช้ฟลิปฟลอปแบบ Divide-by-two Mode โดยความถี่ของสัญญาณที่ได้จากฟลิปฟลอปแต่ละตัวจะมีค่าหารสองจากสัญญาณนาฬิกาที่เข้ามา เมื่อนับไปถึงค่า 111 (หรือ $Q_2 = 1, Q_1 = 1, Q_0 = 1$) และเปลี่ยนกลับมาเป็น 000 จะให้บิต Flag ออกมา ดังแสดงในรูปที่ 2.7(ข)

ใน MCS – 51 จะมีตัวจับเวลาอยู่ภายในชิพ ถ้าเป็นเบอร์ 8051 หรือ 8031 จะมี 2 ตัว คือ Timer 0 และ Timer 1 แต่ถ้าเป็นเบอร์ 8052 จะมีเพิ่มอีกหนึ่งตัวคือ Timer 2 รีจีสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการใช้ Timer แสดงได้ดังตารางที่ 2.2 ซึ่งจะเห็นว่ามีรีจีสเตอร์บางตัวสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย นอกจากนี้ตัว Timer สามารถใช้เป็นตัวนับ (Counter) ได้อีกด้วย โดยการโปรแกรมในรีจีสเตอร์ TMOD



(ข)

รูปที่ 2.7 รีจีสเตอร์ที่ใช้เป็น Timer

รีจีสเตอร์	หน้าที่	ตำแหน่ง	สามารถอ้างอิงตำแหน่งบิต
TCON	Control	88H	Yes
TMOD	Mode	89H	No
TL0	Timer 0 Low-byte	8AH	No
TL1	Timer 1 Low-byte	8BH	No
TH0	Timer 0 High-byte	8CH	No
TH1	Timer 1 High-byte	8DH	No
T2CON*	Timer 2 Control	C8H	Yes
RCAP2L*	Timer 2 Low-byte Capture	CAH	No
RCAP2H*	Timer 2 High-byte Capture	CBH	No
TL2*	Timer 2 Low-byte	CCH	No
TH2*	Timer 2 High-byte	CDH	No

* มีในเบอร์ 8032 / 8052

ตารางที่ 2.2 รีจีสเตอร์ที่ใช้เป็น Timer

2.3.1 Timer Mode Register (TMOD)

ตัวรีจีสเตอร์ TMOD เป็นรีจีสเตอร์ควบคุม Timer จะแบ่งออกเป็น 2 กลุ่ม กลุ่มละ 4 บิต โดย 4 บิตบนจะเป็นการควบคุม Timer 1 ส่วน 4 บิตล่างจะเป็นการควบคุม Timer 0 ความหมายของแต่ละบิตดูในตารางที่ 2.3 ซึ่งตัวรีจีสเตอร์นี้เป็นตัวเลือกการทำงานว่าจะให้ตัว Time/Counter ทำงานในโหมดใดและเป็น Timer หรือ Counter รีจีสเตอร์ TCON ไม่สามารถจะโปรแกรมเข้าไปในระดับบิตได้ (Not Bit-Addressable) ซึ่งการใช้งานมักจะโปรแกรมเข้าไปครั้งเดียวในตำแหน่งเริ่มต้นของโปรแกรม

บิต	ชื่อ	Timer	ความหมาย
7	GATE	1	Gate bit ถ้าบิตนี้เซตวงจรจะทำงาน เมื่อ INT1 เป็น High
A	C/T	1	เป็นบิตเลือก Counter / Timer 1 = ใช้เป็น Counter 0 = ใช้เป็น Timer
5	M1	1	Mode bit 1 (ดูตาราง 5-3)
4	M0	1	Mode bit 0 (ดูตาราง 5-3)
3	GATE	0	บิต Gate ของ Timer 0
2	C/T	0	บิตเลือก Counter / Timer ของ Timer 0
1	M1	0	Timer 0 M1 bit
0	M0	0	Timer 0 M0 bit

ตารางที่ 2.3 รีจิสเตอร์ TMOD (Timer Mode)

M1	M0	Mode	ความหมาย
0	0	0	ใช้เป็น Timer แบบ 13-bit (8048 Mode)
0	1	1	ใช้เป็น Timer แบบ 16-bit
1	0	2	ใช้เป็น Timer แบบ 8-bit Auto-reload Mode
1	1	3	Split Timer Mode : แยก Timer 0 ออกเป็น Timer 8 บิตสองตัวคือ TL0 และ TH0 โดยไม่ใช้ Timer 1

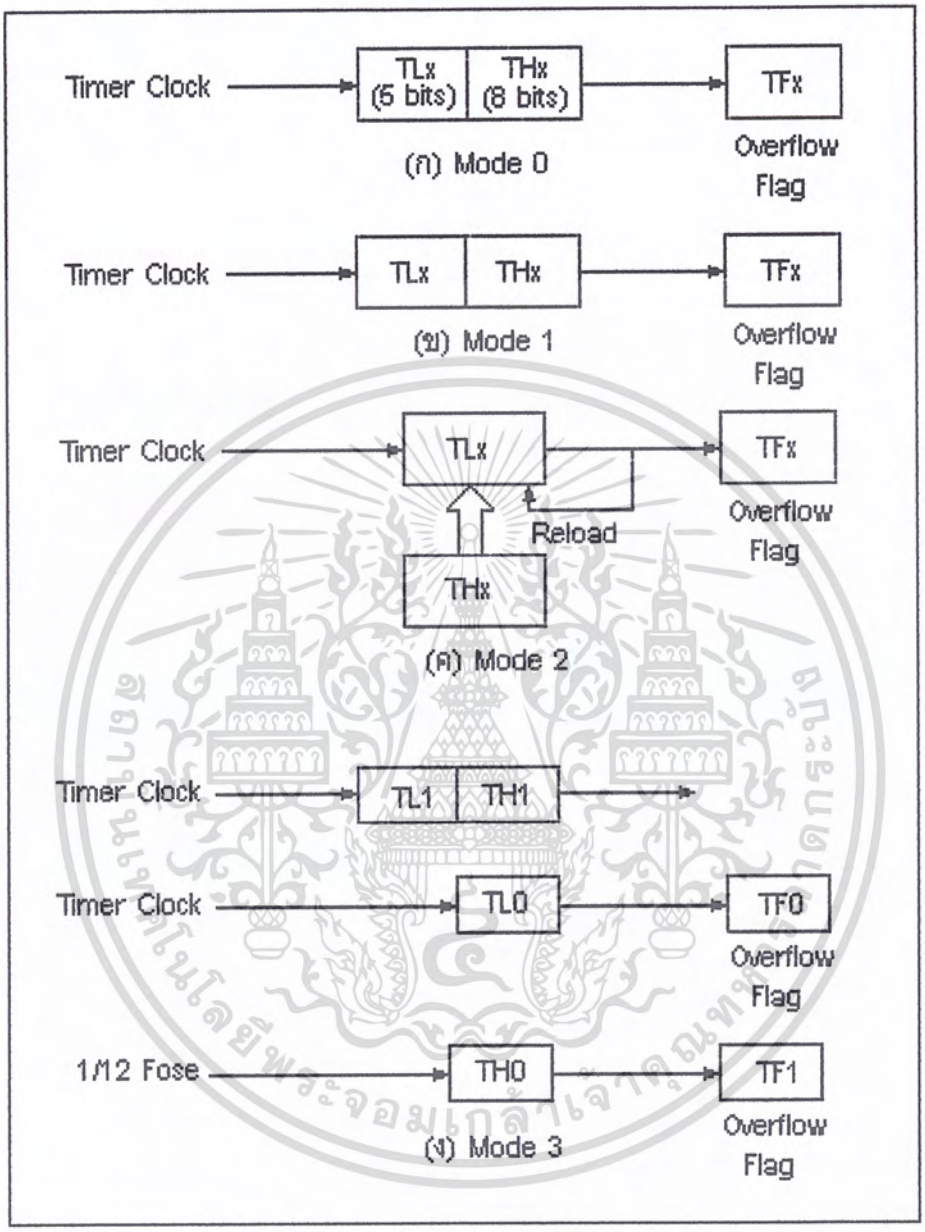
ตารางที่ 2.4 การใช้ Timer โหมดต่าง ๆ

2.3.2 Timer Control Register (TCON)

รีจีสเตอร์ TCON เป็นรีจีสเตอร์ที่บอกสถานะและควบคุมบิต Timer 0 และ Timer 1 ซึ่งดูได้จากตารางที่ 2.5 รีจีสเตอร์นี้สามารถเข้าถึงข้อมูลระดับบิตได้

บิต	ชื่อ	ตำแหน่งบิต	ความหมาย
TCON.7	TF1	8FH	บิตแฟลกแสดงการโอเวอร์โฟลว์ของ Timer 1 จะ Set โดย Hardware และ Clear โดย Software
TCON.6	TR1	8EH	บิตควบคุมการปิด-เปิด Timer 1 Set และ Clear โดย Software
TCON.5	TF0	8DH	แฟลกแสดงการโอเวอร์โฟลว์ของ Timer 0
TCON.4	TR0	8CH	บิตควบคุมการปิด-เปิด Timer 0
TCON.3	IE1	8BH	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT1 จะ Set โดย Hardware และสามารถ Clear ได้ด้วย Software
TCON.2	IT1	8AH	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT1 สามารถ Set และ Clear ได้ด้วย Software
TCON.1	IE0	89H	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT0
TCON.0	IT0	88H	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT0

ตารางที่ 2.5 แสดงความหมายแต่ละบิตของรีจีสเตอร์ TCON (Timer Control)



รูปที่ 2.8 การทำงานของ Timer ในโหมดต่าง

2.3.3 Timer Mode And Overflow Flag

เมื่อใช้ Timer 0 และ Timer 1 จะต้องใช้รีจีสเตอร์คู่ TLx และ THx โดยค่า x จะเป็นตัวบอกว่าเป็น Timer 0 หรือ Timer 1 การใช้ Timer สามารถใช้งานได้หลายโหมด ดังแสดงในรูปที่ 5.2 ซึ่งเราสามารถเซตค่าโหมดการทำงานได้ โดยการโปรแกรมในรีจีสเตอร์ TMOD

13-Bit Timer Mode (Mode 0)

การทำงานในโหมด 0 นี้จะเป็นการใช้ Timer แบบ 13 บิต ดังแสดงในรูป 2.8(ก) ซึ่งจะใช้ 5 บิต ล่างของ TLx โดยไม่สนใจ 3 บิตที่เหลือ และ 8 บิต ของ THx การทำงานในโหมดนี้ เมื่อบิตของ TLx นับไปจนเป็น "1" ทุกบิตจะส่ง Clock 1 ลูกให้ หนึ่งลูกให้ THx นับต่อและเมื่อนับเป็น "1" ทุกบิต และเปลี่ยนกลับเป็น "0" จะเกิด Overflow Flag เกิดขึ้น

16-Bit Timer Mode (Mode 1)

การทำงานในโหมดนี้จะเหมือนกับการทำงานในโหมด 0 แต่เป็น Timer แบบ 16 บิต ซึ่งการนับจะเริ่มตั้งแต่ 0000H,0001H,0002H ไปเรื่อย ๆ และจะเกิด Overlflow ขึ้น เมื่อมีการเปลี่ยนจาก FFFFH เป็น 0000H ดังรูปที่ 2.8(ข) ซึ่งเป็นการเซต Overflow Flag และค่านี้จะเกิดขึ้นในบิต TFx ของรีจีสเตอร์ TCON ซึ่งสามารถอ่านและเขียนด้วยโปรแกรม

การใช้ตัว Timer นี้ค่าของบิตสูงสุด (MSB) คือค่าบิต 7 ของ THx ส่วนบิตต่ำสุด (LSB) คือบิต 0 ของ TLx บิต LSB จะเป็น Toggles เมื่อมีสัญญาณอินพุตเข้ามา ถูกหารด้วย 2 ดังนั้นจะพบว่าบิต MSB จะ Toggles ด้วยค่าความถี่ของสัญญาณอินพุตหารด้วย 65,536 (2^{16}) และค่า Timer รีจีสเตอร์นี้ (TLx/THx) สามารถอ่านและเขียนได้ด้วยการโปรแกรม ดังนั้นสามารถนำไปประยุกต์ใช้งานได้ตามต้องการ

8-Bit Auto – Reload Mode (Mode 2)

การทำงานในโหมด 2 เรียกอีกอย่างหนึ่งว่า 8-bit Auto – reload Mode โดยใช้ Timer ไบต์ต่ำ (TLx) เป็น Timer แบบ 8 บิต เมื่อไบต์ต่ำเกิด Overflows หรือเกิดการเปลี่ยนแปลงจาก FFH เป็น 00H จะมีการโหลดค่าที่เก็บไว้ในไบต์สูง (THx) ไปเก็บไว้ในไบต์ต่ำ (TLx) ซึ่งจะเป็นค่าเริ่มต้นของการนับครั้งต่อไป นิยมใช้สร้างเป็นฐานเวลาที่สามารถโปรแกรมได้ การทำงานในโหมดนี้แสดงดังรูปที่ 2.8(ค)

Split Timer Mode (Mode 3)

การทำงานในโหมด 3 นี้ ตัว Timer 1 จะไม่ทำงาน ตัว Timer 0 จะแยกเป็น 2 ตัว ตัวละ 8 บิต คือ TL0 และ TH0 เมื่อ Timer เกิด Overflows จะมีการเซตบิต TF0 และ TF1 ดังแสดงในรูปที่ 2.8(ง)

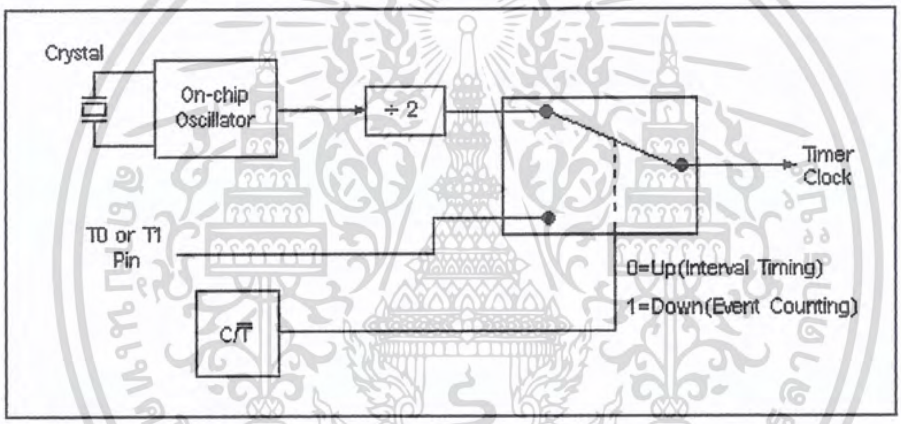
การทำงานในโหมด 3 นี้ Timer 1 จะไม่ถูกใช้งานแต่เราสามารถสวิตช์ให้ Timer 1 ไปทำงานในโหมดอื่นได้ แต่การทำงานของ Timer 1 จะไม่มีการอินเทอร์รัพท์เกิดขึ้น เพราะบิต TF1 ถูกใช้ในการนับของ TH0 ในการทำงานของโหมด 3 ไปแล้ว เราอาจมองว่าถ้าให้ Timer ทำงานในโหมด 3 ทำให้เรามี Timer เพิ่มขึ้น คือ TH0 และ TL0 ใน Timer 0 โหมด 3 และโปรแกรมให้ Timer 1 ไปทำงานในโหมดอื่น ๆ

2.3.4 Clocking Source

ในรูปที่ 2.8 ไม่ได้แสดงว่า Timer Clock นำมาจากที่ใดซึ่งการใช้ Timer นี้สามารถใช้ได้ 2 หน้าที่ คือเป็นตัวจับเวลา (Timer) และเป็นตัวนับ (Counter) ซึ่งสามารถโปรแกรมได้โดยการเซตหรือรีเซต บิต C / Tในรีจีสเตอร์ TMOD

การใช้เป็นตัวจับเวลา (Timer)

ถ้าบิต C / Tใน TMOD เป็นลอจิก "0" จะเป็นการเลือกให้ Timer นำ Clock มาจากวงจร Oscillator ในชิพ ซึ่งสัญญาณนาฬิกาจะเข้ามาทุก ๆ Machine Cycle หรืออาจกล่าวได้ว่าค่าใน THx และ TLx จะมีค่าเพิ่มขึ้นด้วยอัตราการนับแต่ละครั้งใช้เวลาเท่ากับ 1/12 ของความถี่ของสัญญาณนาฬิกาที่ใช้บนชิพ ดังแสดงในรูปที่ 2.9 ถ้า MCS – 51 ใช้สัญญาณนาฬิกา 12 MHz การนับจะมีความถี่เท่ากับ 1 MHz



รูปที่ 2.9 ความถี่ของสัญญาณนาฬิกาที่เข้าขา Timer

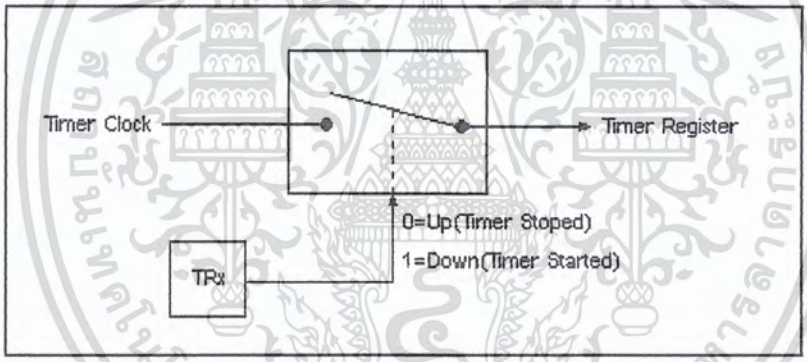
การใช้เป็นตัวนับ (Counter)

ถ้าบิต C / T เป็น "1" ตัว Timer จะนำ Clock มาจากภายนอกโดยใช้ขา P3.4 หรือ T0 เป็นขา Input Clock ให้กับ Timer 0 และใช้ขา P3.5 หรือ T1 เป็น Input Clock ให้กับ Timer 1 ดังรูปที่ 2.9 หรืออาจมองว่า ถ้าจะให้นับอะไรสัญญาณที่จะนับให้ต่อกับขา T0 และ T1 ในการใช้เป็น Counter สัญญาณที่เข้ามามีการเปลี่ยนแปลงจาก "1" เป็น "0" จะทำให้วงจรนับ TLx มีค่าเพิ่มขึ้น 1 ภายใน MCS – 51 นี้จะตรวจสอบขาอินพุต T0 และ T1 ในช่วงเวลาเฟส 2 ของ State 5 (S5P2) ถ้าพบว่ามีค่าเป็น "1" ต่อมาในอีกหนึ่ง Machine Cycle ที่เฟส 2 ของ State 5 (S5P2) ลอจิกอินพุตเปลี่ยนเป็น "0" จะทำให้ค่าใน Timer เพิ่มขึ้น 1 ดังนั้น จะเห็นได้ว่าการนับ 1 ครั้งจะต้องใช้เวลา 2 Machine Cycles ดังนั้นความถี่สูงสุดที่จะให้ Timer ทำงานเป็น Counter นับได้ จะมีค่ามากที่สุด 500 kHz ถ้า MCS – 51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz

2.3.5 การเริ่ม , หยุด และการควบคุม Timer

ในรูปที่ 2.8 จะแสดงลักษณะของ Timer Registers ซึ่งจะเห็นว่าประกอบด้วย TLx และ THx และเมื่อเกิด Overflow จะเกิดเอาต์พุตที่บิต TFx สำหรับสัญญาณนาฬิกาที่จะเข้าไปใน Time จะมาจาก 2 ส่วนดังแสดงในรูปที่ 2.9 ต่อไปจะกล่าวถึงว่าเราจะควบคุมให้เริ่ม , หยุดตัว Timer ได้อย่างไร

วิธีเริ่มและหยุดตัว Timers สามารถควบคุมได้ที่บิต TRx ในรีจีสเตอร์ TCON โดยปกติแล้ว TRx จะเคลียร์หลังจากที่ระบบถูกรีเซต ซึ่งจะเป็นการให้ Timer ไม่นับและ TRx นี้จะเซตได้จากชุดคำสั่ง หรือการโปรแกรม พิจารณารูปที่ 2.10



รูปที่ 2.10 การใช้บิตควบคุม TR

ตัวบิต TRx จะเป็นส่วนที่สามารถเข้าถึงข้อมูลในระดับบิตได้ (Bit Addressable)ในรีจีสเตอร์ TCON ถ้าจะให้ TIMER 0 เริ่มทำงานจะเขียนคำสั่งได้ดังนี้

SETB TR0

ถ้าจะหยุดทำงานเขียนคำสั่งได้ดังนี้

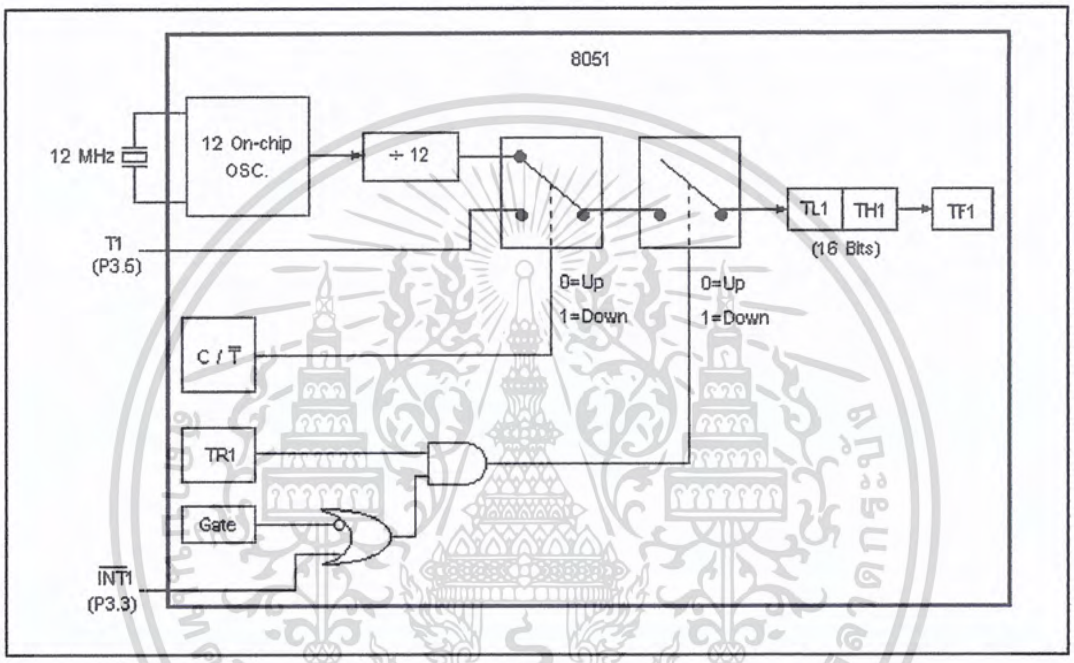
CLR TR0

ในการเขียนโปรแกรมภาษาแอสแซมบลี สามารถใช้สัญลักษณ์ TR0 ในคำสั่ง SETB TR0 เลยได้ เพราะตัวแอสแซมบลีจะตีความ TR0 เป็น Bit Address ตำแหน่ง 8CH

วิธีควบคุม Timer สามารถควบคุมได้ที่บิต GATE ใน TMOD และขาอินเตอร์รัพท์จากภายนอก INTx ถ้า INT0 เป็นลอจิก “0” และโปรแกรมให้ Timer 0 ทำงานในโหมด 2 เมื่อ TL0/TH0 = 0000H, GATE = 1 และ TR0 = 1 เมื่อ INT0 ขึ้นเป็นลอจิก “1” ตัว Timer จะ “Gate On” และจะให้สัญญาณ

นาฬิกาความถี่ 1 MHz เมื่อ INT0 ลงเป็น "0" ตัว Timer " Gate Off " สัญญาณที่ได้จะมีความกว้างของสัญญาณนาฬิกา 1 μS ส่งเข้าไปใน TL0/TH0

รูปที่ 2.11 จะเป็นระบบที่สมบูรณ์ของ Timer 1 เมื่อทำงานในโหมด 1 ซึ่งเป็น 16-bit Timer โดยใช้รีจีสเตอร์ TL1 / TH1 และ Overflow Flag TF1 ในรูปจะเห็นถึงการควบคุมแหล่งกำเนิด Clock การเริ่มทำงาน และการหยุดทำงาน



รูปที่ 2.11 ระบบทั้งหมดของ Timer 1

2.3.6 Intializing And Accessing Timer Register

การใช้งาน Timer เริ่มแรกจะต้องโปรแกรมเพื่อเลือกโหมดการทำงานของ Timer ก่อนเมื่อเริ่มใช้งานก็โปรแกรมให้ เริ่มทำงาน, หยุดทำงาน, อ่าน และ เคลียร์ค่า Flag Bits อ่านค่า Timer Registers ตามลำดับ เพื่อนำไปประยุกต์การใช้งานต่อไป

TMOD คือ รีจีสเตอร์ที่ต้องโปรแกรม โดยเซตโหมดการทำงานก่อน ตัวอย่างเช่น ถ้าให้ Timer 1 เป็น 16-bits Timer (โหมด 1) นับสัญญาณนาฬิกาบนชิพ สามารถเขียนคำสั่งได้ดังนี้

MOV TMOD, #00010000B

ผลที่ได้จากคำสั่งข้างบนคือ เซตบิต M1 = 0 และ M0 = 1 ซึ่งเป็นการเลือกโหมด 1 และให้ C / T= 0 และ GATE = 0 ซึ่งเป็นการใช้สัญญาณนาฬิกาจากภายในหรือใช้เป็น Timer และตัว Timer นี้จะยังไม่ทำงาน ถ้าบิตควบคุม TR1 ยังไม่ได้เซต

ถ้าให้ Timer นี้นับขึ้นโดยใช้รีจีสเตอร์ TL1 / TH1 และจะเซตบิต Overflow Flag เมื่อรีจีสเตอร์เปลี่ยนจาก FFFFH เป็น 0000H โดยให้นับเวลาไป 100 μS หรือให้ TL1 / TH1 นับสัญญาณนาฬิกาได้ 100 ลูก ดังนั้นค่าเริ่มต้นของ TL1 / TH1 จะไม่เริ่มที่ 0000H จะต้องเริ่มที่ FFFFH ลบด้วย 100 ลูก หรือ FF9CH เพื่อให้นับไปถึง FFFFH และเปลี่ยนเป็น 0000H ได้สัญญาณนาฬิกา 100 ลูกพอดี สามารถเขียนคำสั่งได้ดังนี้

```
MOV TL1 , # 9CH
MOV TH1 , # 0FFH
```

ถ้าให้ Timer เริ่มทำงานก็ให้บิตควบคุมดังนี้

```
SETB TR1
```

จากนั้นบิต Overflow Flag จะส่งออกมาหลังเวลาผ่านไป 100 μS ซึ่งเราสามารถเขียนโปรแกรมเป็นโปรแกรมวนลูป 100 μS ได้ โดยตรวจสอบบิต TF1 ว่าถูกเซตหรือไม่ ถ้าไม่เซตก็ให้วนลูปต่อไปดังนี้

```
CLR TR1
CLR TF1
```

การใช้แบบ Readind a Timer "On the Fly"

การใช้งานแบบประยุกต์บางงานจะต้องอ่านค่าจาก Timer Register เนื่องจากตัว Timer Register มีขนาด 2 ไบต์ ถ้าหากไบต์ต่ำเกิด Overflow จะทดเข้าไบต์สูง ถ้าหากเขียนโปรแกรมให้อ่านค่าจากไบต์ต่ำก่อน แล้วจึงอ่านไบต์สูงข้อมูลที่ได้อาจเกิดข้อผิดพลาดได้เนื่องจากไบต์ต่ำมีการเปลี่ยนแปลงเร็วกว่าไบต์สูง การอ่านข้อมูลควรอ่านจากไบต์สูงก่อน แล้วจึงกลับมาอ่านไบต์ต่ำ จากนั้นอ่านข้อมูลไบต์สูงอีกครั้ง ถ้าค่าไบต์สูงที่อ่านได้ไม่มีการเปลี่ยนแปลงให้ใช้ค่านั้นได้เลย แต่ถ้ามีการเปลี่ยนแปลงให้อ่านอีกครั้ง ถ้าต้องการอ่านข้อมูลจาก TL1 / TH1 เข้าในรีจีสเตอร์ R6 / R7 อาจเขียนโปรแกรมได้ดังนี้

```
AGAIN :  MOV A, TH1
         MOV R6, TL1
         CJNE A, TH1, AGAIN
         MOV R7, A
```

2.3.7 Short Intervals And Long Intervals

ถ้า MCS – 51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz ถ้าให้ Timers ใช้วงจร Oscillator บนชิพ สัญญาณนาฬิกาจะถูกหารด้วย 12 และ Timer จะทำงานด้วยความถี่ 1 MHz ถ้าต้องการใช้โปรแกรมสร้างสัญญาณนาฬิกาออกมาอาจทำได้โดยง่าย ซึ่งพิจารณาจากการทำงานชุดคำสั่งต่าง ๆ ของ MCS – 51

ใน 1 Machine Cycle จะใช้เวลา 1μS ในตารางที่ 2.6 จะแสดงความกว้างของสัญญาณที่สร้างขึ้นจาก MCS – 51 ที่ทำงานด้วย Crystal ความถี่ 12 MHz

Maximum Interval in Microseconps	Technique
≈10	Software Tuing
256	8 – bit Timer with Auto-reload
65536	8 – bit Timer
No Limit	16 – bit Timer Plus Software Loops

ตารางที่ 2.6 ค่าสูงสุดของการใช้ Timer โหมดต่าง ๆ

2.4 การอินเตอร์รัพท์

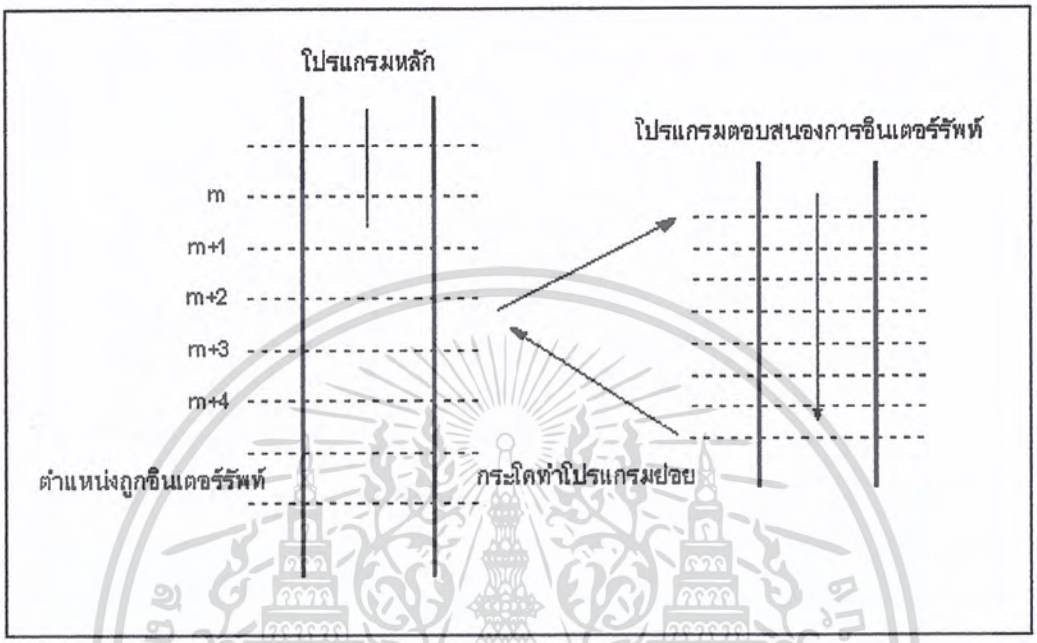
การทำงานของระบบคอมพิวเตอร์ โดยทั่วไปมักมีอุปกรณ์ภายนอกต่อร่วมอยู่ ถ้าคอมพิวเตอร์ต้องการทำงานกับอุปกรณ์ภายนอกจะต้องคอยตรวจสอบอุปกรณ์เหล่านั้นเสมอ ตัวอย่างเช่น ถ้าหากให้คอมพิวเตอร์พอร์ทหนึ่งต่ออยู่กับหลอด LED 7 ส่วน อีกพอร์ททหนึ่งต่อกับสวิตช์ ถ้าระบบของเราทำงานเป็นนาฬิกาเดินไปให้คอยตรวจสอบสวิตช์ด้วยว่ามีการกดหรือยัง การทำงานแบบนี้เรียกว่า Polling Method คือตัวไมโครโปรเซสเซอร์จะต้องคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลาว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ ถ้ามีอุปกรณ์ภายนอกหลายตัวระบบต้องตรวจสอบอุปกรณ์ภายนอกหลายตัว ทำให้เสียเวลาในการทำงานหลักไป การทำงานอีกแบบหนึ่งจะให้ CPU ทำงานหลัก ถ้ามีการกดสวิตช์เมื่อไรให้นาฬิกาหยุดเดินทันที การทำงานในลักษณะนี้ CPU ไม่ต้องเสียเวลาในการตรวจอุปกรณ์ภายนอก ถ้าอุปกรณ์ภายนอกต้องการติดต่อกับ CPU อุปกรณ์ภายนอกจะส่งสัญญาณมาบอก CPUเอง ระบบนี้เรียกว่า การอินเทอร์รัพท์ (Interrupt)

2.4.1 ขบวนการเกิดอินเตอร์รัพท์

ถ้าหากคอมพิวเตอร์กำลังทำงานโปรแกรมหลักอยู่เมื่อมีการอินเทอร์รัพท์เข้ามาคอมพิวเตอร์จะละทิ้งโปรแกรมหลัก แต่ไปทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์ (Interrupt Service Routine) เมื่อทำโปรแกรมตอบสนองอินเตอร์รัพท์เสร็จ คอมพิวเตอร์จะกลับมาทำโปรแกรมเดิม พิจารณารูปที่ 2.12

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m, m+1, m+2 ไปเรื่อย ๆ โดย PC จะชี้ที่ตำแหน่งที่จะอ่านค่าคำสั่งถัดมา เมื่อโปรแกรมทำงานมาถึงตำแหน่งที่ m+3 แล้วเกิดการอินเตอร์รัพท์ขึ้น (ขณะนั้น PC อยู่ที่ m+4) โปรแกรมจะต้องทำงานโปรแกรมตอบ

สนองการอินเตอร์รัพท์ โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเตอร์รัพท์ จากนั้นจะเก็บค่า PC เดิมลงในหน่วยความจำสแตค เมื่อคอมพิวเตอร์ทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์เสร็จสิ้นลง จะคืนค่าใน สแตค (m+4) ให้กับ PC ทำโปรแกรมหลักต่อไป



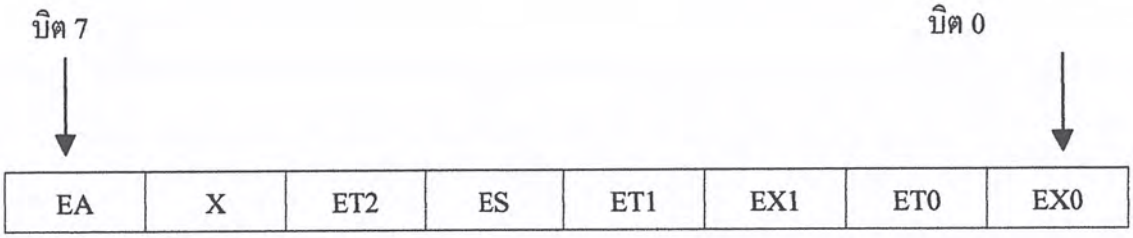
รูปที่ 2.12 ขั้นตอนการทำงานของโปรแกรมเมื่อถูกอินเตอร์รัพท์

2.4.2 สัญญาณอินเตอร์รัพท์

แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ที่ใช้กับ MCS – 51 มีสองชนิดคือ อินเตอร์รัพท์ภายในและภายนอก โดยอินเทอร์รัพท์ภายในจะเกิดขึ้นจากภายในตัว MCS – 51 เอง ได้แก่สัญญาณจาก ไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1 (TF1) และพอร์ทอนุกรม สำหรับอินเตอร์รัพท์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทางขา INT0 และ INT1 เมื่อมีสัญญาณอินเตอร์รัพท์จากแหล่งต่าง ๆ เข้ามา เราสามารถโปรแกรมได้ว่าจะให้ MCS – 51 ยอมให้มีการอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมไปที่ รีจีสเตอร์ IE (Interrupt Enable) และถ้ามีสัญญาณอินเตอร์รัพท์มากจากแหล่งต่าง ๆ หลายแหล่งพร้อมกันเราสามารถจัดลำดับได้ว่า จะให้อินเตอร์รัพท์ใดเกิดก่อน โดยการโปรแกรมไปที่ อินเตอร์รัพท์ไพออริตี้ IP (Interrupt Priority) รีจีสเตอร์ทั้งสองตัวมีรายละเอียดดังนี้

Interrupt Enables

เป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเตอร์รัพท์จากแหล่งต่าง ๆ จะทำอินเตอร์รัพท์เหล่านั้นหรือไม่ โดยรายละเอียดของบิตต่าง ๆ มีดังตารางที่ 2.7



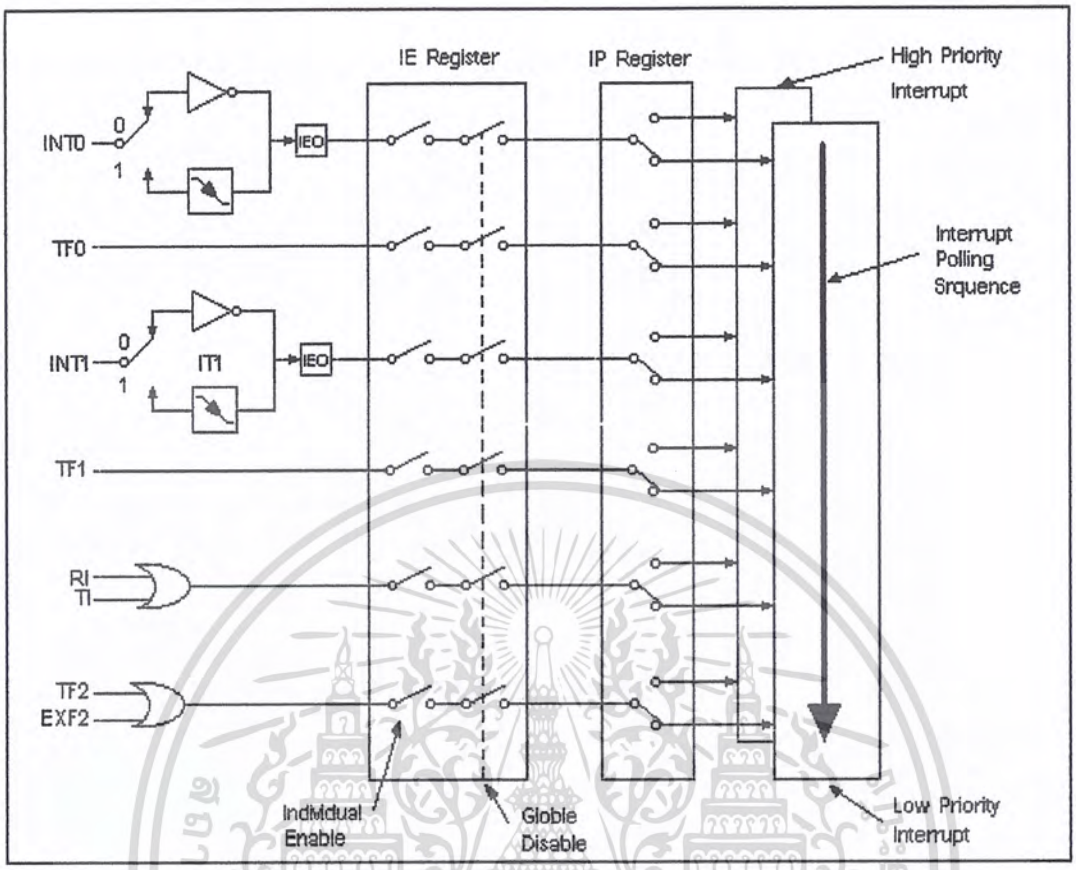
บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFH	ถ้าเซตยอมให้มีการอินเตอร์รัพท์
IE.6	-	AEH	ไม่ใช้งาน
IE.5	ET2	ADH	Enable อินเตอร์รัพท์จาก Timer 2 (ใช้กับ 8052)
IE.4	ES	ACH	Enable อินเตอร์รัพท์จากพอร์ทอนุกรม
IE.3	ET1	ABH	Enable อินเตอร์รัพท์จาก Timer 1
IE.2	EX1	AAH	Enable อินเตอร์รัพท์จาก INT1
IE.1	ET0	A9H	Enable อินเตอร์รัพท์จาก Timer 0
IE.0	EX0	A8H	Enable อินเตอร์รัพท์จาก INT0

ตารางที่ 2.7 บิตต่าง ๆ ของรีจิสเตอร์ IE

Interrupt Priority

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ใช้ในการจัดลำดับความสำคัญของการอินเทอร์รัพท์ซึ่งสามารถจัดได้สองลำดับ ถ้าเป็น "1" หมายความว่ามีความสำคัญสูงสุด ถ้าเป็น "0" หมายความว่ามีลำดับความสำคัญต่ำสุด ความหมายของบิตต่าง ๆ แสดงได้ดังตารางที่ 2.8 ถ้าหากกำหนดให้มีความสำคัญเป็น "1" เหมือนกันหมด MCS – 51 จะจัดลำดับความสำคัญใหม่ดังนี้

ลำดับ	อินเตอร์รัพท์
1 (สูงสุด)	IE0
2	TF0
3	IE1
4	TF1
5 (ต่ำสุด)	Serial Port



รูปที่ 2.13 รีจิสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการอินเตอร์รัพท์

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	-	ไม่ใช้งาน
IP.6	-	-	ไม่ใช้งาน
IP.5	PT2	0BDH	ใช้กับ Timer 2 (8052)
IP.4	PS	0BCH	ใช้กับพอร์ทอนุกรม
IP.3	PT1	0BBH	ใช้กับ Timer 1
IP.2	PX1	0BAH	ใช้กับอินเทอร์รัพท์จาก INT1
IP.1	PT0	0B9H	ใช้กับ Timer 0
IP.0	PX0	0B8H	ใช้กับอินเทอร์รัพท์จาก INT0

ตารางที่ 2.8 บิตและหน้าที่ต่าง ๆ ของรีจีสเตอร์ IP

จากรูปที่ 2.13 แสดงการอินเตอร์รัพท์จากแหล่งต่าง ๆ ที่มีผลกับ MCS – 51 ถ้าเป็นเบอร์ 8051 8031 จะถูกอินเตอร์รัพท์ได้ 5 แหล่ง ถ้าเป็นเบอร์ 8052,8032 จะถูกอินเตอร์รัพท์ได้ 6 แหล่ง โดยเพิ่มอิน

เตอร์รัพท์จาก Timer 2 ในรูปที่ 2.13 จะแสดงให้เห็นว่า ถ้า MCS – 51 จะถูกอินเตอร์รัพท์ได้จะต้องเซต ค่า Global Enable ในรีจีสเตอร์ IE นอกจากนี้ยังกำหนดได้ว่าจะให้อินเตอร์รัพท์ใดเกิดได้ โดยการเซตค่า Interrupt Enable ของอินเตอร์รัพท์จากแหล่งต่าง ๆ ในรีจีสเตอร์ IE จากรูปยังแสดงให้เห็นอีกว่าเมื่อมีการอินเทอร์รัพท์เข้ามาจะมีผลต่อแฟลกใด เช่นถ้า INT0 เป็น "1" บิต IE0 จะเป็น "1" หมายความว่าถูกอินเตอร์รัพท์ โดยแฟลกต่าง ๆ ที่มีผลจากการถูกอินเตอร์รัพท์แสดงได้ดังตารางที่ 2.9

อินเตอร์รัพท์	แฟลก	ประกอบอยู่ในรีจีสเตอร์
External 0	IE0	TCON.1
External 1	IE1	TCON.3
Timer 1	TF1	TCON.7
Timer 0	TF0	TCON.5
Serial port	T1	SCON.1
Serial port	RI	SCON.0
Timer 2	TF2	T2CON.7 (8052)
Timer 2	EXF2	T2CON.6 (8052)

ตารางที่ 2.9 แฟลกที่จะทำงานเมื่อถูกอินเตอร์รัพท์

จากตารางจะเห็นว่า ถ้ามีการอินเตอร์รัพท์จากภายนอกเข้ามา ตัวที่จะอินเตอร์รัพท์ MCS – 51 คือ บิตแฟลก IE0 ซึ่งอยู่ในรีจีสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมดแล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก TI ถ้ารับข้อมูลหมดแล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก RI ซึ่งอยู่ในรีจีสเตอร์ SCON และถ้าใช้ Timer 0 ในการนับเมื่อเกิด Overflow สามารถอินเตอร์รัพท์ MCS – 51 ได้ทางบิต TF0

2.4.3 การทำงานของระบบหลังถูกอินเตอร์รัพท์

เมื่อ MCS – 51 ถูกอินเทอร์รัพท์จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์ โดยตำแหน่งที่จะกระโดดไปเรียกว่า อินเตอร์รัพท์เวกเตอร์ (Interrupt Vectors) เมื่อทำโปรแกรมตอบสนองการอินเตอร์รัพท์เรียบร้อยแล้ว MCS – 51 จะกระโดดมาทำงานยังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า PC ลงหน่วยความจำสแตคซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจีสเตอร์ SP เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัพท์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตคให้ PC ตามเดิม ค่าอินเทอร์รัพท์เวกเตอร์ของ MCS – 51 แสดงได้ดังตารางที่ 2.10

อินเทอร์รัพท์	อินเทอร์รัพท์เวกเตอร์
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

ตารางที่ 2.10 อินเตอร์รัพท์เวกเตอร์ของอินเตอร์รัพท์ต่าง ๆ

จากตารางจะเห็นว่าถ้าระบบถูกอินเตอร์รัพท์จากภายนอกทาง INT0 ตัว MCS – 51 จะกระโดดไปทำงานที่ตำแหน่ง 0003H ถ้าระบบถูกอินเตอร์รัพท์จาก Timer 0 จะกระโดดไปทำงานตำแหน่ง 000BH

2.4.4 การออกแบบโปรแกรมอินเตอร์รัพท์

ในการเขียนโปรแกรมหลัก (Main Program) จะต้องกำหนดค่าว่าจะให้ MCS – 51 ถูกอินเตอร์รัพท์ด้วยอะไร และจะให้ MCS – 51 ถูกอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมค่าต่าง ๆ ใน IE รีจีสเตอร์ ถ้ามีการอินเทอร์รัพท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจีสเตอร์ IP ดังนั้นในโปรแกรมหลักจะต้องมีการโปรแกรมต่อไปนี้

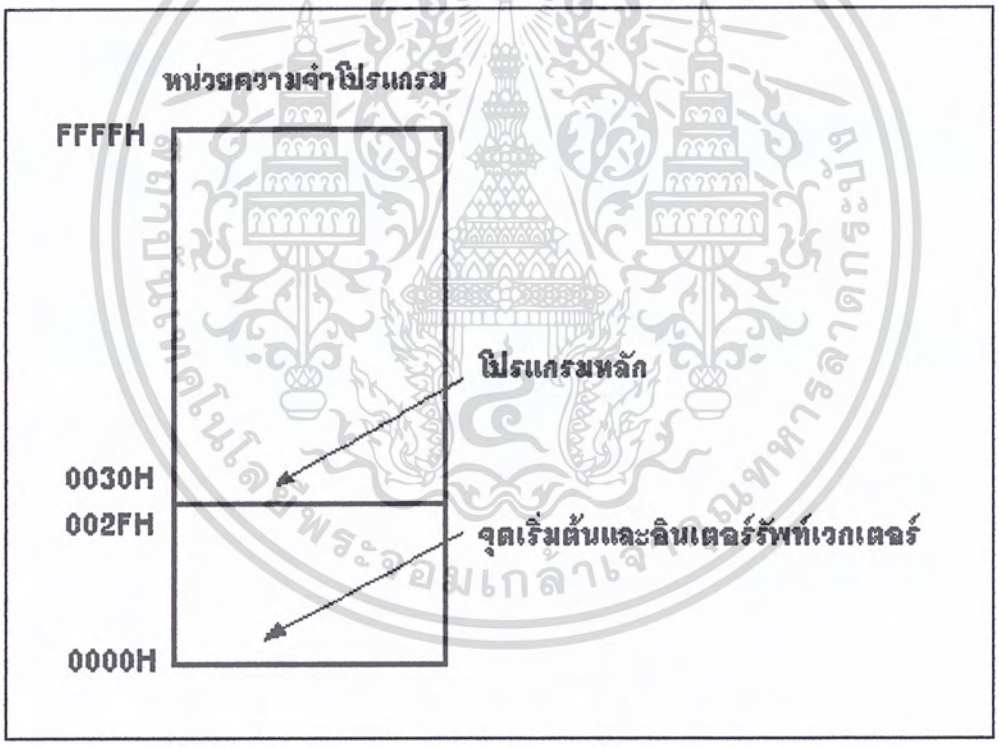
1. โปรแกรมค่าในรีจีสเตอร์ IE
2. โปรแกรมค่าในรีจีสเตอร์ IP

สำหรับโปรแกรมตอบสนองการอินเทอร์รัพท์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่ง แต่จะต้องจบโปรแกรมย่อยด้วยค่า RETI (Return From Interrupt)

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าถ้ากด Reset หรือให้ระบบเริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง 0000H และจะเห็นว่า ตำแหน่งที่เก็บโปรแกรมหลักมีโอกาส อย่างมากที่จะทับกับหน่วยความจำโปรแกรมที่เก็บค่าอินเทอร์รัพท์เวกเตอร์ที่ตำแหน่ง 0003H ถ้าโปรแกรมยาวมากอาจจะไปทับตำแหน่ง 000BH ได้ซึ่งเป็นตำแหน่งของอินเทอร์รัพท์เวกเตอร์ของ Timer 0 ดังนั้นในการเขียน

โปรแกรมหลัก ภายใน 3 ตำแหน่งแรก คือ 0000H,0001H,0002H จะต้องกระโดดไปที่อื่นก่อนเพื่อให้ข้ามอินเทอร์รัพท์เวกเตอร์ไป ซึ่งอาจเขียนโปรแกรมได้ดังนี้

```
        ORG   0000H          ; เริ่มต้นโปรแกรม
        LJMP  MAIN           ; กระโดดไปโปรแกรมหลัก
        .....                ; เพื่อหนีอินเทอร์รัพท์เวกเตอร์
        .....
        ORG   0030H          ; ตำแหน่งเริ่มต้นของโปรแกรม
MAIN :  .....                ; เริ่มต้นโปรแกรมหลัก
        .....
```



รูปที่ 2.14 การจัดตำแหน่งโปรแกรมในหน่วยความจำ

จากตัวอย่างโปรแกรมจะเห็นว่า เมื่อเริ่มต้นโปรแกรมหรือระบบถูกรีเซต ระบบจะทำงานตำแหน่งแรก คือคำสั่งกระโดดไปโปรแกรมหลัก ซึ่งอยู่ต่อจากโปรแกรมตอบสนองการอินเตอร์รัพท์ที่อยู่ที่ตำแหน่ง 0030H

โปรแกรมตอบสนองการอินเตอร์รัพท์แบบสั้น

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเตอร์รัพท์แต่ละแหล่งจะห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเตอร์รัพท์จากแหล่งต่าง ๆ หลาย ๆ แหล่งและโปรแกรมตอบสนองการอินเตอร์รัพท์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์ จะทำให้โปรแกรมไปทับกับตำแหน่งของโปรแกรมตอบสนองการอินเทอร์รัพท์ของอินเทอร์รัพท์ถัดไป แต่ถ้าโปรแกรมตอบสนองการอินเทอร์รัพท์ไม่ยาวมากเกินไปเราสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```
        ORG     0000H
        LJMP    MAIN  ; กระโดดไปโปรแกรมหลัก
        ORG     000BH ; ตำแหน่งเริ่มต้นของอินเตอร์รัพท์ Timer 0
TOISR :.....
        .....
        RETI          ; กลับโปรแกรมหลัก
MAIN : .....          ; โปรแกรมหลัก
        .....
```

จากตัวอย่างโปรแกรมจะใช้อินเตอร์รัพท์จาก Timer 0 เมื่อระบบเริ่มทำงานจะทำตำแหน่ง 0000H โดยกระโดดไปโปรแกรมหลักซึ่งอยู่ที่ตำแหน่งต่อจากโปรแกรมตอบสนองการอินเทอร์รัพท์เมื่อมีการอินเทอร์รัพท์ Timer 0 ระบบจะทำโปรแกรมตำแหน่งที่ 000BH ซึ่งเป็นอินเตอร์รัพท์เวกเตอร์ของ Timer 0 โดยโปรแกรมตอบสนองการอินเตอร์รัพท์จะจบด้วยคำสั่ง RETI เพื่อกลับสู่โปรแกรมหลักต่อไป

โปรแกรมตอบสนองการอินเตอร์รัพท์ขนาดใหญ่

ในกรณีที่มีการอินเตอร์รัพท์จากหลายแหล่ง และโปรแกรมตอบสนองการอินเตอร์รัพท์แต่ละโปรแกรมยาวเกิน 8 ไบต์ เราไม่สามารถเขียนโปรแกรมตอบสนองการอินเทอร์รัพท์ไว้ที่ตำแหน่งของอินเทอร์รัพท์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเทอร์รัพท์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเทอร์รัพท์ที่เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่าง ต่อไปนี้

```
        ORG    0000H  ; เริ่มโปรแกรมของระบบ
```

```
        LJMP   MAIN  ; กระโดดไปโปรแกรมหลัก
        ORG    000BH ; ตำแหน่งของอินเตอร์รัพท์ Timer 0
        LJMP   LED1  ; กระโดดไปโปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1
        ORG    0030H ; ตำแหน่งหลังอินเตอร์รัพท์เวกเตอร์
MAIN : .....         ; โปรแกรมหลัก
        .....
LED1 : .....         ; โปรแกรมตอบสนองการอินเตอร์รัพท์ Timer 1
        .....
        RETI         ; กลับสู่โปรแกรมหลัก
```

จากโปรแกรมจะเห็นว่า เมื่อระบบทำงาน จะต้องทำที่ตำแหน่ง 0000H โดยกระโดดไปทำโปรแกรมหลักที่ตำแหน่งต่อจาก 0030H เพราะตำแหน่งดังกล่าวข้ามอินเตอร์รัพท์เวกเตอร์จากแหล่งต่าง ๆ ไปแล้ว เมื่อมีการอินเตอร์รัพท์จาก Timer 0 โปรแกรมจะต้องทำงานที่ตำแหน่ง 000BH แต่โปรแกรมตอบสนองการอินเตอร์รัพท์ยาวมาก ที่ตำแหน่ง 000BH จึงให้ทำโปรแกรมกระโดด โดยกระโดดไปที่โปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1 ซึ่งอยู่ท้ายโปรแกรม เมื่อจบโปรแกรมจะจบด้วยคำสั่ง RETI เพื่อกลับไปโปรแกรมหลักต่อไป

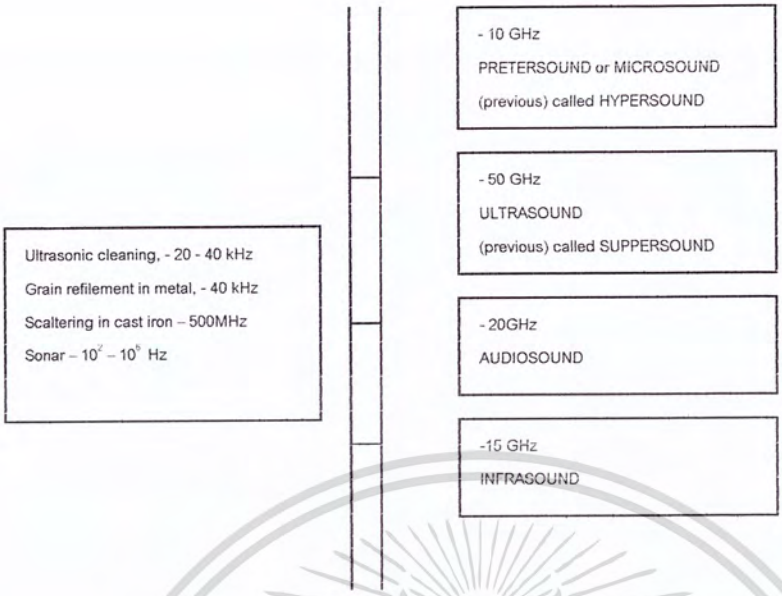
2.5 ทฤษฎีของคลื่นอัลตร้าโซนิค

คุณสมบัติและธรรมชาติของเคลื่อนอัลตร้าโซนิค

หลักการสะท้อนกลับของคลื่นเสียงคือ ฟิลด์ของพลังงานจะถูกส่งออกมาจากตัวส่งถ้าไปกระทบกับวัตถุพลังงานบางส่วนจะสะท้อนกลับไปที่ตัวส่งด้วย แต่การใช้วัตถุเป็นตัวรับที่เหมาะสม (Suitable) จะสามารถรับการสะท้อนกลับของสัญญาณ (Signal) เวลาที่ใช้ในการเดินทางในอากาศ และสามารถที่จะคำนวณระยะทางได้ เพราะฉะนั้นการศึกษาถึงธรรมชาติของคลื่นจะสามารถทำให้เข้าใจในการนำไปประยุกต์ใช้ได้อย่างถูกต้อง

คลื่นอัลตร้าโซนิค

คลื่นอัลตร้าโซนิคคือ คลื่นเสียงที่มีความถี่สูงเกินที่หูมนุษย์จะได้ยิน โดยทั่วไปแล้วหูมนุษย์จะสามารถได้ยินคลื่นความถี่ประมาณ 20 Hz ถึง 20kHzดังนั้นแล้วคลื่นอัลตร้าโซนิคจึงหมายถึงแอมปลิจูด (Amplitude) สูงๆ จะเรียกว่า “ไฮเปอร์ซาวด์”

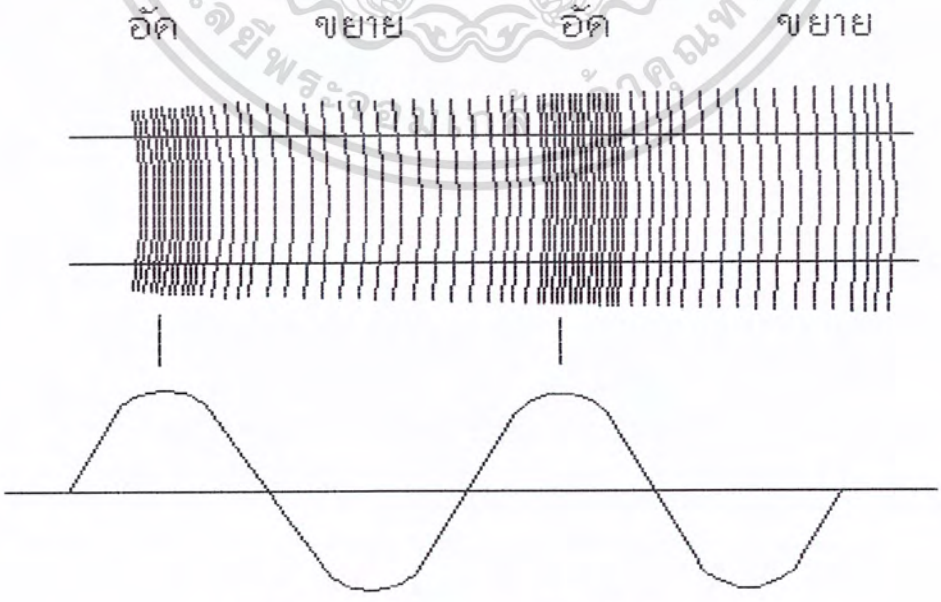


รูปที่ 2.15 แสดงช่วงความถี่ต่างๆ ที่ถูกนำไปใช้งาน

ชนิดของคลื่นอัลตร้าโซนิค

คลื่นอัลตร้าโซนิคที่ทางผ่านตัวกลางต่างๆ มีหลายชนิดด้วยกันแต่ละชนิดแตกต่างกันตามการเคลื่อนของอนุภาคในตัวกลางนั้น

1. คลื่นตามยาว (Longtudinal Wave) คือ คลื่นอนุภาคตัวกลางมีการเคลื่อนที่ไปในทิศทางการเคลื่อนที่ของคลื่น



รูปที่ 2.16 แสดงลักษณะการเกิดคลื่นตามยาว

จากรูปจะเห็นว่าจะมีส่วนของคลื่นอัด (Compression) ซึ่งก็คือคลื่นช่วงของอนุภาคของตัวกลางที่มีความชันสูง และคลื่นขยาย (reaction) คือคลื่นที่อนุภาคของตัวกลางที่มีความดันต่ำและเมื่อนำค่าของความดัน ที่เปลี่ยนแปลงตามระยะทางมาเขียนกราฟจะได้รูปซายน์ (Sine Wave) โดยยอดคลื่นจะตรงกับส่วนอัดและท้องคลื่นจะตรงกับส่วนขยาย ระยะทางระหว่างส่วนอัดและส่วนขยายถึงส่วนขยายคือ 1 ความยาวคลื่นและมีคาบเวลาเป็นคลื่น (T) ซึ่งเท่ากับ 1/f โดยจุดที่เป็นแกนนั้นมีความดัน 1 บรรยากาศ

2. คลื่นตามขวาง (Transerse Wave) คือ คลื่นที่ๆ จุดบนคลื่นมีการเคลื่อนที่ไปในทิศทางตั้งฉากกับทิศทางการเคลื่อนที่ คลื่นชนิดนี้จะเดินทางผ่านตัวกลาง คลื่นชนิดนี้ไม่สามารถเดินทางผ่านตัวกลางที่เป็นของแข็งหรือก๊าซได้ คลื่นตามขวางมีลักษณะเสมือนการเกิดขั้วบวกขั้วลบซึ่งเหตุผลที่ว่าการเปลี่ยนตำแหน่งของอนุภาคในทิศทางเดียวเช่น ในระนาบตั้งฉากกับทิศทางการเคลื่อนที่ไปจากต้นกำเนิดของเคลื่อนตามขวางเป็นพื้นหน้าเรียบ ของระนาบที่เกิดจากการเปลี่ยนแปลงอนุภาคันเนื่องมาจากการแกว่ง ความหนาของตัวกลางความเร็วของคลื่นชนิดนี้จะน้อยกว่าความเร็วของคลื่นชนิดตามยาวในขณะที่เดินทางผ่านตัวกลางชนิดเดียวกัน ดั่งนั้นที่ความถี่เดียวกันความยาวของคลื่นตามขวางจะน้อยกว่าคลื่นตามยาวเสมอ

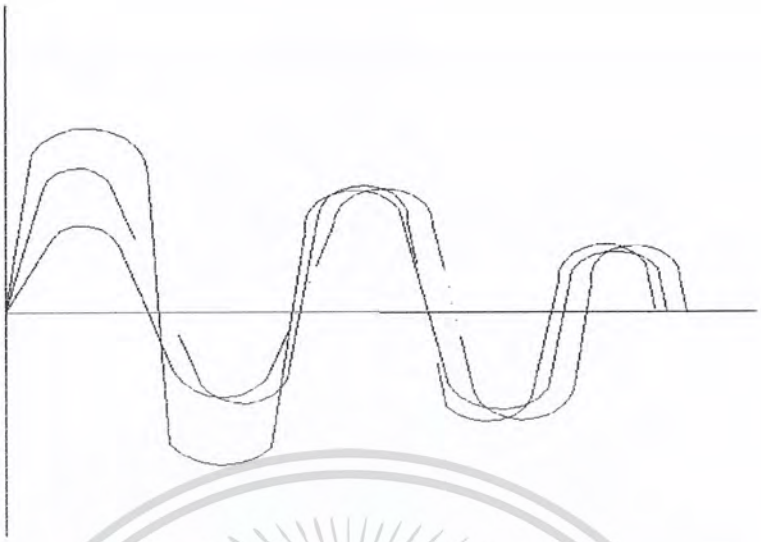
3. คลื่นผิวหน้า (Surface Wave or Reyleigh) คือ คลื่นชนิดหนึ่งซึ่งคล้ายกับคลื่นตามขวางจะต่อกันตรงที่เคลื่อนที่ว่าการเปลี่ยนตำแหน่งของอนุภาคไม่เพียงในทิศทางตั้งฉากกับทิศทางที่เคลื่อนที่เพียงอย่างเดียวแต่มีการเปลี่ยนแปลงในทิศทางเดียวกับทิศทางการเคลื่อนที่ไปตามระนาบในแนวนอน ด้วยเหตุนี้คลื่นจึงเดินทางผ่านไปเฉพาะบนผิวของตัวกลางเท่านั้น

คุณสมบัติที่สำคัญของคลื่นอัลตร้าโซนิค

คุณสมบัติโดยทั่วๆ ไปของคลื่นเสียงจะแสดงได 4 แบบคือ

1. การสอดแทรกของเสียง (Interference) คือ การสอดแทรกของเสียงเกิดจากการรวมตัวของคลื่น 2 คลื่นขึ้นไป ขณะเมื่อพบกันในตัวกลาง (Medium) เดียวกันซึ่งทำให้เกิดผลได้หลายลักษณะคือ

การบิสต์ (beats) ของคลื่นเสียงเป็นปรากฏการณ์ที่เกิดจากการรวมคลื่นที่ความต่างกันหรือต่างเฟสกันที่ไปในตัวกลางเดียวกันแล้วรวมเป็นคลื่นใหม่ซึ่งทำให้แอมปลิจูเปลี่ยนไปดังแสดงในรูป จากรูปเป็นการแสดงบีสต์ขิงคลื่นเสียง 2 คลื่นซึ่งคลื่นทีมีแอมปลิจูดเปลี่ยนไปซึ่งประโยชน์ของการบีสต์ของคลื่นนั้นจะนำไปใช้ในการเปรียบเทียบความถี่ของคลื่นให้แสดงผลออกมาในักษณะแอมปลิจูดที่ต่างกัน



รูปที่ 2.17 แสดงการบีสต์ของคลื่นเสียง 2 คลื่นเสียง

การเกิดคลื่นนิ่ง (Standing Wave) เกิดจากการแทรกสอดของคลื่นที่แอมปลิจูดเท่ากันและความถี่เท่ากันแต่มีทิศทางตรงกันข้ามกันหรือมีเฟสตรงกันข้ามซึ่งจะทำให้เกิดคลื่นนิ่งดังรูป เสียงจะมีความแข็งสูงสุดที่ตำแหน่ง ค. และเบาสุดที่ตำแหน่ง ข. ระยะทางระหว่างขั้วทั้ง 2 เท่ากับ $d = \lambda / 2$ M หรือ $L = n(\lambda / 2)$; n = 1,2,3…

2. การเลี้ยวเบนของคลื่นเสียง (Diffraction) คือ คลื่นเสียงเบนอ้อมสิ่งขีดขวางที่ลักษณะเป็นมุมหรือช่องแคบซึ่งปรากฏการณ์เช่นนี้พบในชีวิตประจำวันอยู่ตลอดเวลา เช่น ในกรณีที่ได้ยินเสียงแตรรถที่อยู่คนละมุมของตึก หรือการที่ได้ยินเสียงลอดผ่านช่องเล็กๆ จากอีกห้องหนึ่งดังแสดงในรูป

3. การสะท้อนของคลื่นเสียง (Reflection) คือ คลื่นเสียงสามารถสะท้อนได้เมื่อตกกระทบตัวกลางโดยที่มุมสะท้อนและจะทำให้เกิดเสียงก้อง (Echo) ซึ่งเสียงก้องที่สะท้อนกลับมาในเวลาที่มากกว่า 50 ms จะทำให้เราได้ยินเสียงนี้เป็นครั้งที่ 2

4. การหักเหของคลื่นเสียง (Refraction) คือ คลื่นเสียงเมื่อเดินทางผ่านตัวกลางที่มีความหนาแน่นต่างกันจะเกิดการหักเหของคลื่น ซึ่งทำให้ความเร็วของคลื่นเสียงเปลี่ยนไป โดยที่ความถี่ยังคงอยู่

การเกิดคลื่นอัลตร้าโซนิค

อัลตร้าโซนิคเป็นคลื่นที่เกิดจากการเปลี่ยนแปลงพลังงานรูปอื่นให้มาเป็นพลังงานกลโดยการสั่นไปมา หรือเกิดจากการเปลี่ยนแปลงพลังงานไฟฟ้ากับพลังงานกล ให้เกิดย่านอัลตร้าโซนิคกระจายออกไปในอากาศดังนั้นจึงถือได้ว่า คลื่นทีเกิดขึ้นเป็นกล (Mechanical Wave) อัลตร้าโซนิคสามารถสร้างได้โดยทรานดิ้วเซอร์ ซึ่งเป็นอุปกรณ์ที่ใช้เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกลหลักกการมีหลายวิธีดังนี้

1. แบบเปียโซอิเลคทริค (Piezo-Electric Transducer)

2. แบบแมกนิโตสตริคทีฟ (Magnetrostictive Transducer) ซึ่งเปลี่ยนแปลงไปมาระหว่าง พลังงานไฟฟ้าในขดลวดกับตำแหน่งความยาวของแกนเหล็กที่สวมขดลวดนั้น

3. แบบอิเลคทริคโทรสติคทีฟ (Electrostrictive Transducer) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้ากับพลังงานกล

สำหรับเปียโซอิเลคทริคเป็นแบบที่นิยมใช้เพราะมีราคาถูก และหาซื้อได้ง่าย

ความถี่และความยาวคลื่น (Frequency and Wave Length)

ความถี่ คือ จำนวนของการออสซิเลทที่สมบรูณ์จากแหล่งกำเนิดคลื่นภายในเวลา 1 ms คลื่นที่ถูกส่งจากแหล่งกำเนิดจะเดินทางด้วยความถี่เดียวกัน

ความยาวคลื่น คือ ระยะทางที่คลื่นเดินทางระหว่างการสั่นที่สมบรูณ์หรือการเดินทางครบหนึ่งรอบ (1 Cycle) สามารถกล่าวได้ว่า ความยาวคลื่นเป็นระยะทางระหว่างการอัดอย่างต่อเนื่อง (Successive Compression) หรือการเบาบางของอากาศ (Rare-Fractions)

การอัด คือ การที่บริเวณนั้นมีความหนาแน่นของโมเลกุลและแรงดันบริเวณรอบๆ ส่วนการเบาบางเป็นบริเวณพาหะ ที่เกิดจากการลดความหนาแน่นของโมเลกุลและแรงดันสัมพันธ์กับแรงดันของบรรยากาศปกติ

ความถี่และความยาวคลื่นมีความสัมพันธ์กันตามสมการข้างล่าง

$$C : f\lambda$$

C : ความเร็วของการเดินทาง

F : ความถี่ (Hz)

λ : ความยาวคลื่น (m)

ความเร็วของคลื่นอัลตร้าโซนิค

ความเร็วของคลื่นอัลตร้าโซนิคในอากาศที่อุณหภูมิปกติสัมพันธ์ใช้จะเป็นสมการ

$$V = 331.45 + 0.607t \quad (m/s)$$

V : ความเร็วของคลื่นในตัวกลางอากาศ

t : อุณหภูมิของคลื่นในอากาศ

GAS	Velocity (m/s)
AIR (DRY 1 องศาเซลเซียส)	331.45
ARGON	319
CARBON MONOXIDE	338
CARBON DIOXIDE	259
HELIUM	965
HYDROGEN METHANE	1284

ตารางที่2.13 แสดงความสัมพันธ์ความเร็วของคลื่นในก๊าซต่างๆ

ปริมาณพลังงานของคลื่นอัลตร้าโซนิค

ปริมาณพลังงานของอัลตร้าโซนิค จะถูกวัดในรูปความเข้มของคลื่นอัลตร้าโซนิคจะมีหน่วยเป็น วัตต์ต่อตารางเซนเมตร (W/cm^2) เป็นการไหลของพลังงานผ่านพื้นที่ 1 ตารางเซนติเมตรซึ่งตั้งฉากกับทิศทางการเดินทางของคลื่นใน 1 วินาที

การลดทอนของคลื่นอัลตร้าโซนิค

เมื่อคลื่นเดินทางผ่านตัวกลางลำคลื่น (Bleam) ของคลื่นอัลตร้าโซนิคจะสูญเสียความเข้มซึ่งเกิดจากการลู่ออกของลำอัลตร้สาโซนิค หรือเกิดจากการกระจายพลังงานของคลื่นออกจากลำคลื่นเนื่องจากความไม่ต่อเนื่องของตัวกลาง และอาจจะเกิดการดูดซับพลังงานส่วนหนึ่งของคลื่นโดยตัวกลางที่คลื่นเคลื่อนที่ผ่านพลังงานที่ดูดซับนี้จะเปลี่ยนเป็นพลังงานความร้อน การดูดซับพลังงานความร้อนนี้ขึ้นอยู่กับลักษณะของวัตถุ ความยืดหยุ่นและความหนาแน่น รวมทั้งความถี่ที่ใช้ ยิ่งความถี่สูงพลังงานยิ่งดูดซับได้มาก

อัลตร้าโซนิคทรานสดิวเซอร์

ปรากฏการณ์เปียโซอิเลคทริค เปียโซอิเลคทริคเป็นปรากฏการณ์ธรรมชาติอย่างหนึ่งซึ่งทำให้พลังงานเปลี่ยนแปลงจากรูปหนึ่งไปเป็นอีกรูปหนึ่งได้ กล่าวคือถ้าป้อนแรงกลให้แก่ Solid Cystalline Dielectric ดังในรูป ก็จะเกิดความเค้น (Stress) ภายในคริสตอล และทำให้ผลึกของคริสตอลผิดรูปไป เช่น พวกควอทซ์ (Quartz) ผลก็คือประจุเปลี่ยนไปการผิดรูปผิดร่างของเลททิส เป็นผลให้ความสัมพันธ์ระหว่างการแทนที่ (Displancement) ของประจุบวกและลบ ในแลททิสเปลี่ยนไปการแทนที่ของประจุภายในจะเท่ากับประจุภายนอกของขั้วที่ตรงข้ามของคริสตอล เรียกว่า ผลของเปียโซอิเลคทริค (Piezo-Electric Effect)

การวัดประจุทำได้โดยการต่ออิเลคโทรด (Electrod) เข้าที่ผิวด้านนอกแล้ววัดความต่างศักดิ์ระหว่างขั้วทั้งสอง ขนาด (Magnitude) และการมีขั้ว (Poarity) ของประจุบนผิวที่ถูกเหนี่ยวนำ (Induced Surface Charge) เป็นสัดส่วนโดยตรงกับขนาดและทิศทางของแรง (Force) ที่มากระทำ

Cc : คาปาซิเตอร์ของชิ้นส่วนเปียโซอิเลคทริค

Rc : Leakage Resistance ของเปียโซอิเลคทริค ปกตจะมีค่าสูงประมาณ 10-10

ความต้านทานระหว่าเทอร์มินอล โดยทั่วไปจะเป็นความต้านทานทางโหลด RL

ที่ความถี่ปานกลางและความถี่สูงโวลท์เตจ E_0 ที่คร่อมโหลดหาโดย Cc และ Cz ค่าเหล่านี้มาได้โดยการแบ่งโวลท์เตจ E_0 ได้จากสมการ

$$E = Ecc/(Cc+Cl)$$

โดยขึ้นกับความถี่ถ้าเอาท์พุทโวลท์เตจมีค่ามาก สามารถทำให้ลดลงโดยการเพิ่มค่า Cl เช่น การนำค่า Cc ต่อขนานเข้าไป ที่ความถี่ต่ำโวลท์เตจ E_0 คร่อมโหลดหาได้โดยค่า รีแอคแตนซ์ของ Cc และ อิมพีแดนซ์ของ Cl และ RL ที่ต่อขนานกัน โวลท์เตจ E_0 ขึ้นกับความถี่และลดลงถ้าความถี่ลดลง

วัสดุเปียโซอิเลคทริค วัสดุเปียโซอิเลคทริคที่ใช้กันอย่างกว้างขวาง เช่น Quartz, Tourmoaline Sulphate, Barium Tilanate และ Zirconnate Titanate (TZI) โดยทั่วไปแล้วพวกควอทซ์ และ คริสตอลที่เป็นเปียโซอิเลคทริคธรรมชาติมันจะมีขั้วของมันเองตามธรรมชาติ แต่พวกวัสดุเปียโซอิเลคทริคที่สังเคราะห์ขึ้นมา เช่น แบเรียม ไททาเนท เซรามิค จะต้องนำมาทำการอบคริสตอลภายในแรงดัน และวางวัสดุที่ได้นี้ในสนามไฟฟ้าที่มีแรงดันขง D.C. มาก หลังจากชิ้นส่วนนี้ถูกนำไปในสนามไฟฟ้าแล้วคริสตอลนี้จะมีขั้วตามแนวของทิศทางของสนาม และประพฤติตัวตามคุณสมบัติของเปียโซอิเลคทริค สำหรับชิ้นส่วนที่ทำจากวัสดุสังเคราะห์นี้ไม่มีข้อจำกัดทางขนาด โดยโครงสร้างของคริสตอลและยังสามารถทำให้มีรูปร่างและขนาดต่างๆ และทิศทางขั้วก็จะถูกสร้างขึ้นระหว่างขั้นตอนการผลิต

ชนิดของเปียโซอิเลคทริคทรานสดิวเซอร์

เปียโซอิเลคทริคทรานดิวสเซอร์สามารถแบ่งออกได้เป็น 2 ชนิด คือ

1. แบบ Generation – action Transducer ใช้ตัวรับโดยแรงดันไฟฟ้าที่เกิดขึ้นจะหาได้จากแรงดันและความถี่ที่มากระทำต่อเปียโซอิเลคทริค
2. แบบ Motor – action Transducer ใชเป็นตัวส่งโดยการเปลี่ยนแปลงของรูปร่างทำให้เกิดคลื่นอัลตร้าโซนิค จะขึ้นอยู่กับขนาดความสูงและความถี่ของแรงดันไฟฟ้าที่ป้อนให้ในทั้งสองกรณี ค่าแรงดันไฟฟ้าที่เกิดขึ้นจะขึ้นกับขนาดของวัสดุ

การทำงานของทรานดิวสเซอร์ตัวส่งตัวรับ

เมื่อเซรามิคได้รับสัญญาณแรงดันตกคร่อมจะทำให้ชิ้นสารเซรามิคโก่งงอ ทำให้เกิดการอัดอากาศโดยรอบเกิดเป็นคลื่นขึ้นมา ดังนั้นถ้าป้อนสัญญาณเป็นห้วงๆ (Electrically Pulse) จากการออส-ซิเลท ก็จะทำให้ชิ้นสารโก่งงอมากน้อยหรือทิศทางใดตามขนาดและทิศทางการเปลี่ยนแปลงขนาดของสัญญาณไฟฟ้าจากการออสซิเลทนั้นออกไป โดยทั่วไปกำลังเอาท์พุทที่ออกมาจะตกลงประมาณ 10% ของกำลังไฟฟ้าที่ป้อนให้แต่เอาท์พุทจะสูงที่ค่านี้ โดยประมาณก็ต่อเมื่อความถี่ของสัญญาณออสซิเลทที่ป้อนเข้าชิ้นสารเซรามิคตรงกับความถี่รีโซแนนซ์ที่เป็นความถี่ทางกลตามธรรมชาติของชิ้นสารเซรามิคนั้นๆ ส่วนที่ความถี่อื่นๆ กำลังเอาท์พุทจะลดลงกว่านี้ ส่วนความถี่รีโซแนนซ์ของชิ้นสารเซรามิคเข้ามาจะทำให้ชิ้นสารโก่งตัวไปมากและเกิดสัญญาณแรงดันไฟฟ้าที่มีขนาดเล็กขึ้นคร่อมขั้วทั้งสองของตัวมัน

ข้อควรรู้ในการใช้งานตัวรับ-ตัวส่งทรานดิวสเซอร์

1. ไม่ควรให้ตัวทรานดิวสเซอร์ได้รับการกระทบกระเทือน หรือตกจากที่สูงเพื่อป้องกันโครงสร้างมิให้เสียหาย

2. ทรานดิวสซอร์ที่ขายโดยทั่วไปจะทนแรงดันตกคร่อมตัวมันไม่เกิน 20 V_{rms} ดังนั้นขนาดของสัญญาณที่จะป้อนให้กับตัวทรานดิวสเซอร์ก็ควรอยู่ในขีดจำกัดนี้

3. ความถี่รีโซแนนซ์ (คือความถี่ที่ตัวมันสามารถทำงานอย่างมีประสิทธิภาพที่สุดของทรานดิวสเซอร์) 40kHz (Bandwidth) ที่มีขายโดยทั่วไปจะผิดพลาดไม่เกิน +/- 1kHz และมีแถบความถี่ประมาณ 4.5 kHz (Bandwidth) สำหรับตัวส่ง และมีความถี่ประมาณ 5.0 kHz สำหรับตัวรับ จะเห็นได้ว่าแถบความถี่ของตัวรับจะกว้างกว่าตัวส่งเล็กน้อย เพื่อให้แน่ใจว่าตัวรับสามารถรับความถี่ทั้งหมดที่ออกจากตัวส่งได้

4. อุณหภูมิที่ใช้งานของตัวทรานดิวสเซอร์ควรอยู่ภายในช่วง -20 องศาเซลเซียล ถึง 60 องศาเซลเซียล

5. ตัวส่งและตัวรับจะมีทิศทางคล้ายคลึงกันมาก กล่าวคือ ที่ตำแหน่งบนจากแนวแกนของตัวส่งไปประมาณ 30 องศา ความแรงของคลื่นเสียงที่ถูกส่งออกไปจะลดลงจากแนวแกนประมาณ 30 องศา ความไวหรือขนาดแรงดันที่ออกมาจะลดลงจึงควรจะทำให้ทั้งตัวรับและตัวส่งอยู่ในแนวที่พุ่งตรงกันมากที่สุดอย่างไรก็ตาม ในกรณีที่อยู่ในห้องจะเกิดการเบี่ยงเบนจากกันได้มาก เพราะคลื่นเสียงอัตร้าโซนิคจะสามารถจะสะท้อนกับกำแพง และวัตถุที่อยู่ภายในห้องทำให้คลื่นเสียงเข้าไปหาตัวรับได้หลายทาง

อัลตร้าโซนิคนี้สามารถทำให้เป็นลำแคบได้โดยที่ใช้เลนส์ที่เรียกว่า Planoconc-avelenrs วางข้างหน้าของทรานดิวส์เซอร์เพื่อทำให้ลำแสงแคบ Beam นี้จะทำให้ near field สั้นลงและ far field

กระจายกว้างขึ้น การทำให้เป็นลำคลื่นเหมาะสำหรับการทรานดิวสเซอร์ที่สร้างความถี่สูงที่เหมาะสำหรับที่ใช้ในการแพทย์ (Ultrasound) ที่มีความถี่ตั้งแต่ 2-5 MHz เพื่อใช้ในการตรวจเนื้อเยื่อ

6. ในกรณีที่ใช้งานตัวรับจะต้องมีความต้านทานต่อขนานกับตัวรับเพื่อทำหน้าที่เป็นโหลด ตามปกติแล้วตัวต้านทานนี้ควรมีค่าอยู่ระหว่าง 10 กิโลโอห์ม ถึง 100 กิโลโอห์ม จากการทดสอบพบว่าถ้าเปลี่ยนจาก 100 กิโลโอห์ม มาเป็น 10 กิโลโอห์ม ความไวจะลดลงประมาณ 10-12 dB แต่ความถี่จะกว้างขึ้น ถ้าใช้ความต้านทานต่ำลงไปอีกความถี่รีโซแนนซ์ (ความถี่กลาง) จะลดลงไปจากที่ระบุไว้ ถ้าต้องการใช้งานมีสัญญาณรบกวนมากควรใช้โหลที่มีความต้านทานสูงสักหน่อย เพื่อให้ตัวส่งมีความไวสูงและมีความถี่แคบ

7. ตามปกติแล้วสามารถนำเอาตัวส่งและตัวรับ มาใช้งานแทนกันได้ในการใช้งานส่วนใหย่ตัวส่งและตัวรับรุ่นใดก็สามารถใช้งานแทนกันได้ในงานส่วนใหญ่ ขอเพียงแต่ให้มีความถี่รีโซแนนซ์เดียวกนเท่านั้นเอง อย่างไรก็ตามในบางกรณีอาจต้องมีการเปลี่ยนแปลงค่าความต้านทานสมมูลยีทางไฟฟ้าทางด้านไฟฟ้าสลับเพื่อ ให้เกิดลักษณะผลตอบสนองทางความถี่สอดคล้องกับความถี่เดิม

8. ประโยชน์การใช้งานคลื่นอัลตร้าโซนิค คลื่นัลตร้าโซนิคเป็นคลื่นที่มีทิศทางทำให้สามารถเล็งเห็นคลื่นไปตามเป้าหมายที่ต้องการเจาะจงได้ยิ่งคลื่นมีความถี่สูงขึ้น ความยาวคลื่นก็ยิ่งสั้นลง ถ้าความยาวคลื่นยาวกว่าช่องเปิด (ที่ทำให้เสียงออกมา) ของตัวที่ทำให้เกิดเสียงความถี่นี้ เช่นคลื่นความยาว 300Hz ในอากาศจะมีความยาวคลื่นประมาณ 1 เมตร เศษๆ ซึ่งจะยาวกว่าช่องเปิดที่ให้คลื่นเสียงออกมาจากตัวกำเนิดเสียง โดยทั่วไปมากมายคลื่นจะหักเหที่ด้านนอก ของตัวกำเนิดเสียงที่ทำให้เกิดการกระจายทิศทางของคลื่น แต่ถ้าความถี่สูงขึ้นมาอยู่ในย่านอัลตร้าโซนิคอย่างเช่น 40 kHz จะมีความยาวคลื่นในอากาศเพียง 8 มิลลิเมตร เท่านั้นซึ่งเล็กกว่าตัวที่ทำให้เกิดคลื่นเสียง ความถี่นี้มากคลื่นเสียงจะไม่มีการเลี้ยวเบนที่ขอบ ซึ่งพุ่งออกมาเป็นลักษณะลำแคบๆ หรือที่เรียกว่า มีทิศทางนั้นเอง การมีทิศทางของคลื่นเสียงอัลตร้าโซนิคทำให้เราสามารถนำเอาไปใช้งานได้หลายอย่าง เช่น คลื่นวัดความหนาของวัตถุ โดยสังเกตระยะที่คลื่นสะท้อนกลับมา เครื่องวัดความลึกทำแผนที่ใต้ท้องทะเล ส่วนการใช้งานด้านการแพทย์อาจใช้ความถี่ช่วง 1 MHz ถึง 10 MHz และมีความถี่เป็น GHz ก็มีใช้งานกัน

2.6 ทฤษฎีการสร้างภาพของ CT

สัมประสิทธิ์การลดทอนของรังสีเอ็กซ์

เนื่องจากการสร้างภาพที่จะกล่าวต่อไปนี้เกี่ยวข้องกับการพิจารณาการจำแนกของสัมประสิทธิ์การลดทอนของรังสีเอกซ์ ดังนั้นตอนแรกนี้ควรจะพิจารณาถึงเทอมสัมประสิทธิ์การลดลงของรังสีเอกซ์เป็นการเริ่มต้น

รังสีเอกซ์เป็นคลื่นแม่เหล็กไฟฟ้าความถี่สูงและเป็นพวกไอออนไนซิงเรดิเอชั่น (ionzing radiation) เมื่อเดินทางผ่านเข้าในตัวกลางใดก็ตามจะเกิดอันตรกิริยา (interaction) กับตัวกลางนั้น เช่น ปรากฏการณ์ โฟโตอิเล็กทรอนิกส์ (photoelectric effect) ปรากฏการณ์คอมตัน (Compton effect) หรือการผลิตสารคู่ (pair production) เป็นต้น ปรากฏการณ์ต่างๆ เหล่านี้มีผลทำให้รังสีเอ็กซ์ที่เดินทางผ่านตัวกลางนั้นๆ ออกมาแล้วมีความเข้มลดลง

สมมุติตัวกลางหรือวัตถุที่ประกอบด้วยเนื้อเดียวกันตลอด มีความหนา x ฉายรังสีเอ็กซ์ที่มีความเข้ม Io ผ่านเข้าไปในตัวกลางนี้ เมื่อรังสีเอ็กซ์ทะลุออกมาปรากฏว่ามีความเข้มลดลงเป็น I ดังรูป ถ้ารังสีเอ็กซ์มีพลังงานค่าเดียว (monochromatic X-ray) กรณีนี้สามารถแสดงความสัมพันธ์ระหว่าง Io, I และ x ได้ดังนี้

$$I = Io\, e^{-ux}$$

เมื่อ u คือ สัมประสิทธิ์การลดลงของรังสีเอ็กซ์ จากสมการ จะเห็นว่ารังสีเอกซ์ที่เดินทางผ่านตัวกลางออกมาแล้วจะมีความเข้มข้นลดลงโดยขึ้นกับความหนาและคุณสมบัติของตัวกลางนั้น ซึ่งแสดงการลดทอนลงในเทอม e^{-ux}

กรณีตัวกลางระกอบด้วยสารสองชนิดหนา x1 และ x2 ตามลำดับ และมีค่าสัมประสิทธิ์การลดทอนลงของรังสีเอ็กซ์ของตัวกลางทั้งสองเป็น u1 และ u2 ตามลำดับจะได้

$$I = Io\, e^{-u1x1 - u2x2}$$

ปกติร่างกายมนุษย์เป็นตัวกลางที่มีสารชนิดประกอบกัน ในการคิดการดูดกลืนรังสีเอ็กซ์จำเป็นต้องแบ่งตัวกลางออกเป็นแถบเล็กๆ ขนาดความหนา dx เท่ากันจำนวนมาก ตามความหนาของแถบเล็กๆ นั้นเล็กจนกระทั่งสามารถคิดได้ว่าในแถบเล็กๆ นั้นประกอบด้วยเนื้อสารเนื้อเดียวตลอด กรณีนี้ ความสัมพันธ์ระหว่าง I, Io, u และความหนาของตัวกลางจนเป็นไปตามสมการ

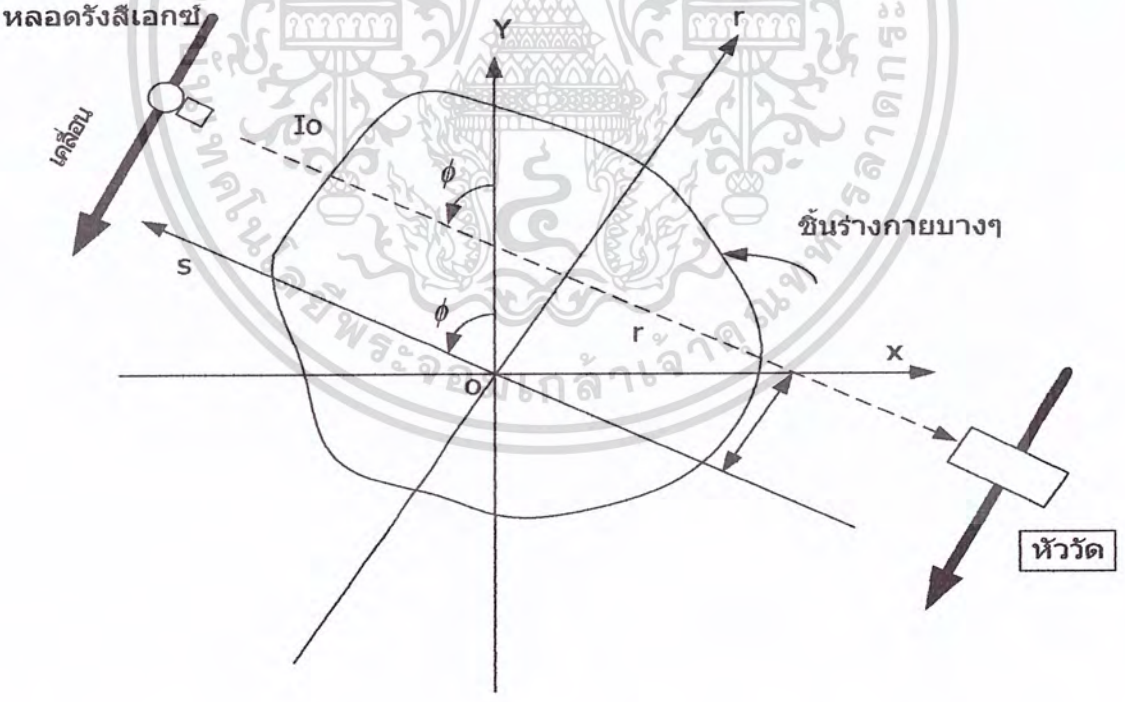
$$I = Io \exp\left[\int -u dx\right]$$

นิยามของเรย์ซัม (Ray-sum)

สมการข้างต้นเป็นสมการความเข้มของรังสีเอ็กซ์ที่ผ่านตัวกลางซึ่งประกอบด้วยสารหลายชนิด เทอมที่แสดงการลดของรังสีเอ็กซ์เปลี่ยนจากการรวมกันธรรมดาเช่นสมการ มาเป็นการอินทิเกรตตามเส้นทางลำรังสีเอ็กซ์ (line integral) และสมการนี้จะยังคงเป็นจริงเฉพาะเมื่อรังสีเอกซ์มีพลังงานค่าเดียวเท่านั้น ในทฤษฎีค่าเดียวเท่านั้น ในทฤษฎีการคำนวณสร้างภาพสมการนี้ จะมีบทบาทสำคัญ ดังนั้นเพื่อความสะดวกจะขอกำหนดโคออร์ดิเนต (coordinate) ขึ้นดังรูป

สมมุติหั่นร่างการเป็นชิ้นบางๆ แล้วยกมาแปะไว้กับกระดาษดังรูป ทุกจุดบนระนาบนี้อธิบายด้วยโคออร์ดิเนต (x, y) รังสีเอ็กซ์ลำแคบพุ่งออกจากหลอดทำมุม ϕ กับแกน Y เพื่อความสะดวกรังสีเอ็กซ์ลำแคบจะอธิบายด้วยโคออร์ดิเต (r, s) และลำแสงเอ็กซ์แต่ละลำอธิบายด้วยโคออร์ดิเนต (r, ϕ) ในระบบโคออร์ดิเนตที่กำลังพิจารณานี้เฉพาะรังสีเอ็กซ์ลำแคบหนึ่งๆ สมการที่อธิบายความเข้มที่ทะลุผ่านออกมาจะคล้ายกับสมการ โดยจะเป็น

$$I = Io \exp\left[-\int_{r,\phi} u(x,y)ds\right]$$



รูปที่ 2.18 แสดงเรขาคณิตของรังสีเอ็กซ์ที่เดินทางผ่านร่างกายมนุษย์ชิ้นบางๆ ทุกๆ จุดบนระนาบอธิบายด้วย(x,y)

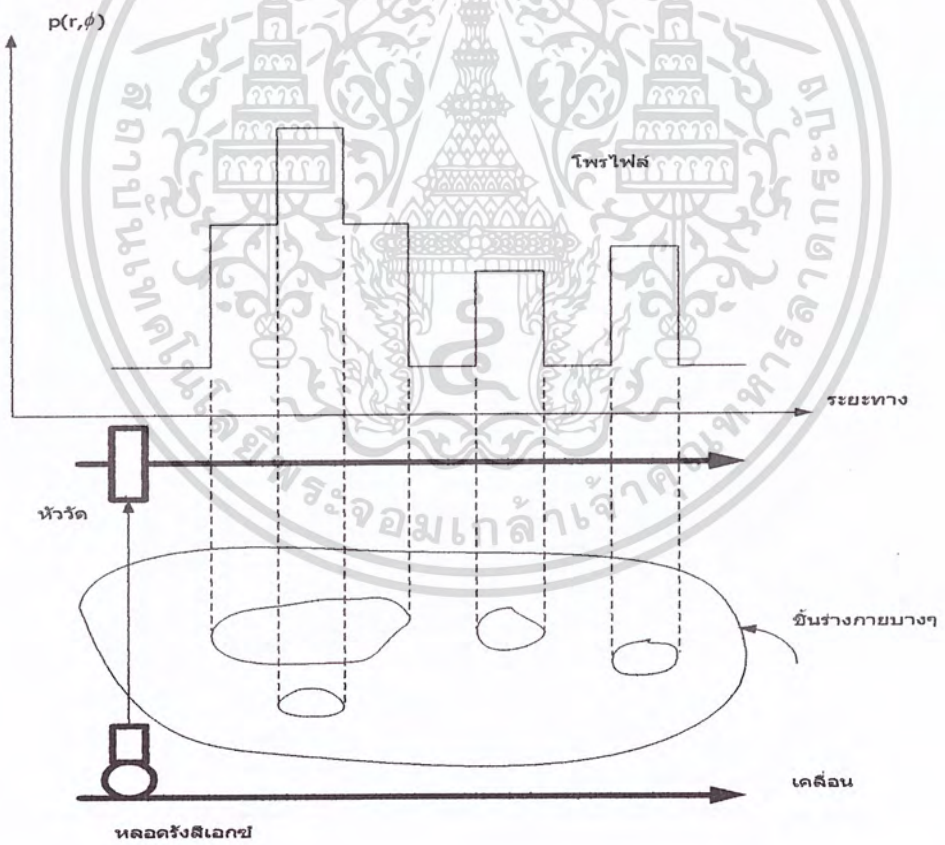
ลำรังสีเอ็กซ์เส้นประอธิบายด้วยมุม ϕ วัดเทียบกับแกน y และระยะ r วัดเทียบกับจุดกำเนิด

โดยที่เทอมในวงเล็บหมายถึง การอินทิเกรตตามเส้นทางที่ถูกกำหนดด้วยโคออร์ดิเนต (r, s) สำหรับมุม ϕ ใดๆ มุมหนึ่งเท่านั้น สมการ (2-4) สมการเขียนอีกรูปหนึ่งได้ดังนี้

P(r, ϕ) เป็นเทอมที่กำหนดขึ้นมีชื่อเรียกว่า เรย์ซัม หรือ เรย์โพเจคชั่น (ray - projection) นิยามว่าเป็นการอินทิเกรต u(x, y) ตามเส้นทางของรังสีเอ็กซ์ลำแคบที่มีโคออร์ดิเนต (r, ϕ) เป็นค่าที่สามารถวัดได้จากการทดลอง

ที่มุม ϕ ใดๆ การเคลื่อนที่ตัดในแนวเส้นตรง (translation) หมายถึง การเปลี่ยนค่า r จะได้เซท (set) ที่สมบรูณ์ของเรย์ซัม สำหรับมุม ϕ นั้น เซทดังกล่าวเรียกว่า โพร์ไฟล์ (profile) โปรเจกชัน (projection) ดูรูป ประกอบความสัมพันธ์ระหว่างโคออร์ดิเนต (x, y) และ (r, ϕ) คือ

$$R = x \cos \phi + y \sin \phi$$



รูปที่ 2.19 การเคลื่อนที่ตัดในแนวเส้นตรงของรังสีเอกซ์ลำแคบตัดผ่านในระนาบของร่ากายมุม ϕ ใดๆ หัววัดบันทึกข้อมูลไว้ 1 โพไฟล์

ค่าสัมประสิทธิ์การลดลงของรังสีเอ็กซ์หรือ u(x, y) เป็นค่าคงที่ของตัวกลางหนึ่งๆ ตรงจุด (x, y) ในระนาบที่สนใจถ้า u(x, y) ที่ค่าสูงแสดงว่าตัวกลางนั้นมีค่าความหนาแน่นมาก และสามารถดูดกลืนรังสีเอ็กซ์ได้ดีในทำนองกลับกันถ้า u(x, y) มีค่าต่ำ แสดงว่าตัวกลางนั้นมีความหนาแน่นต่ำและดูดกลืนรังสีเอกซ์ได้น้อยดังนั้นในระนาบใดๆ ของร่างกายหรือตัวกลางใดๆ ถ้าสามารถคำนวณค่าสัมประสิทธิ์การทดลองของรังสีเอ็กซ์บนทุกๆ จุดในระนาบนั้นๆได้ ก็จะสามารถสร้างภาพด้วยการใช้ค่า u(x, y) ที่คำนวณได้มาเรียงตามตำแหน่งที่สอดคล้องกับตำแหน่งบนระนาบจริงๆ ทฤษฎีการสร้างภาพจึงได้มุ่งหาคำตอบของสมการ นั้นคือค่า u(x, y) หรือการกระจายของสัมประสิทธิ์การลดลงของรังสีเอ็กซ์

แบ็กโพเจกชั่น (Back – projection)

ทฤษฎีการสร้างภาพอันดับแรกที่จะกล่าวถึงเรียกว่า แบ็กโพรเจกชัน ซึ่งเป็นวิธีที่ง่ายที่สุด ความยุงยาก ในทางคณิตศาสตร์มีน้อย คูห์ล (kuhl) และเอดเวิร์ดส์ (Edwards) เป็นสองคนแรกที่นำวิธีนี้ไปสร้างภาพของระนาบในตัวผู้ป่วยได้สำเร็จ

ได้กล่าวแล้วว่าการที่รังสีเอกซ์ลำแคบเคลื่อนที่ตัดในแนวเส้นตรงไปในระนาบของร่างกายหรือตัวกลางใดๆ 1 ครั้ง โพรไฟล์ ซึ่งประกอบด้วยเรย์ซัม หรือ p (r, ϕ) คำนวณหนึ่งเช่นในรูปที่แสดงมา ถ้าบิดแนวรังสีเอ็กซ์ไปจากแนวเดิมทีละ 1 องศา จนครบ 180 องศา โดยที่แต่ละองศาให้รังสีเอกซ์ลำแคบเคลื่อนที่ตัดในแนวเส้นตรงผลลัพธ์จะได้ 180 โพรไฟล์ ซึ่งเป็นข้อมูลสำหรับนำไปคำนวณสร้างภาพ

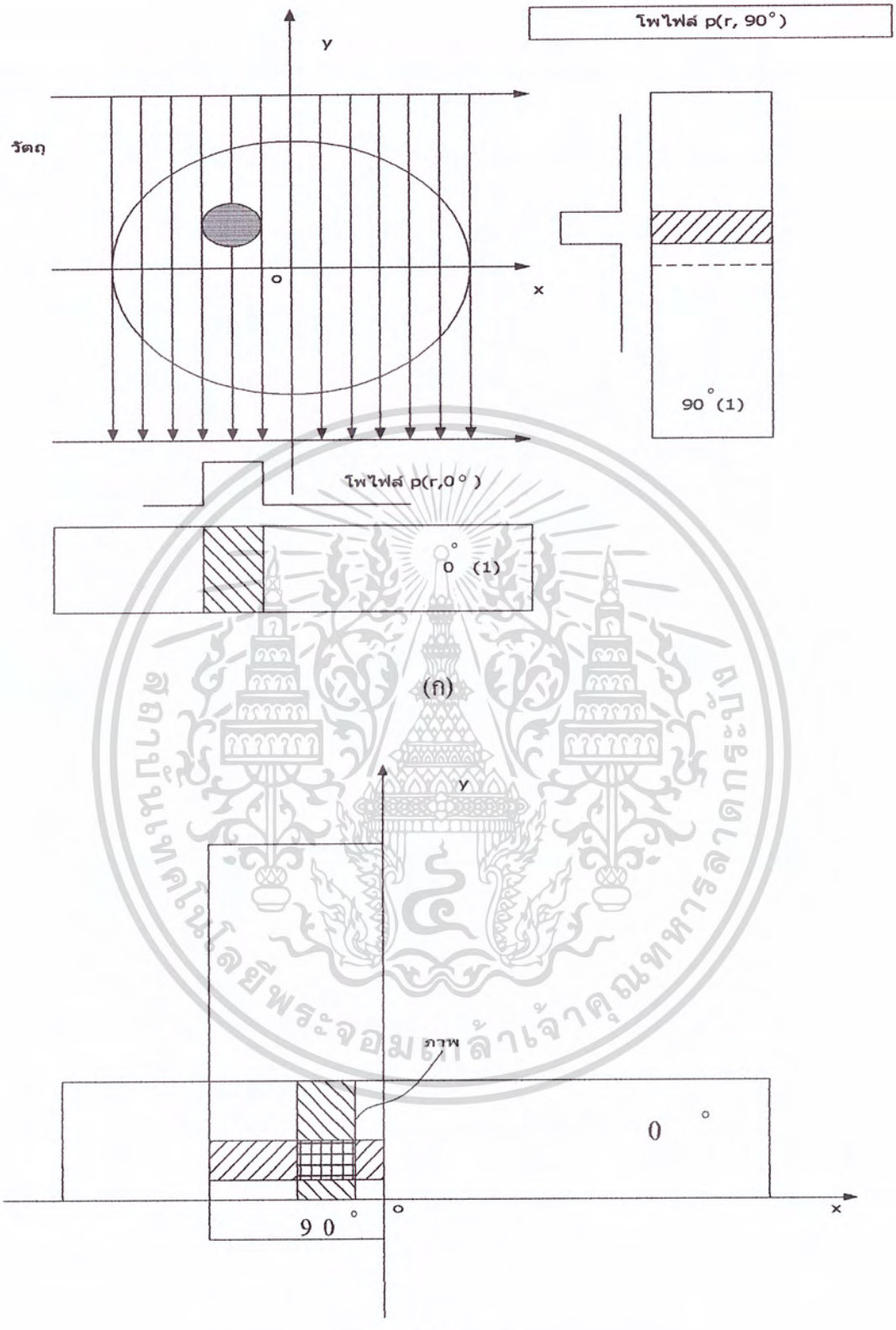
การคำนวณสร้างภาพตามวิธีการนี้ไม่ได้คำนวณการจำแนกของสัมประสิทธิ์การลดลงในระนาบที่สนใจอย่างที่กล่าวไว้ในตอนต้น แต่เป็นการคำนวณการจำแนกของ $\overline{u}$ แทนโดยที่

$$\overline{u}(x, y) = \sum_{j=1}^{m} p(rj, \phi j)\Delta\phi$$

เมื่อ

$$rj = x \cos \phi j + y \sin \phi j$$

ในสมการ m คือ จำนวณโพไฟล์ทั้งหมดหมด ϕj คือ มุมของโพรไฟล์ ที่ j , $\Delta\phi$ เรียกว่าระยะห่างเชิงมุม (angular distance) ระหว่างโพรไฟล์มีค่าเท่ากับ π /m สัญลักษณ์ $\overline{u}$ (x, y) ไม่ใช่ค่าสัมประสิทธิ์การลดลงของตัวกลางจริงๆ แต่การพิจารณาจำแนกของมันบนระนาบ XY ที่กำหนดขึ้นจะสามารถจำลองภาพขึ้นได้เหมือนจริงมาก



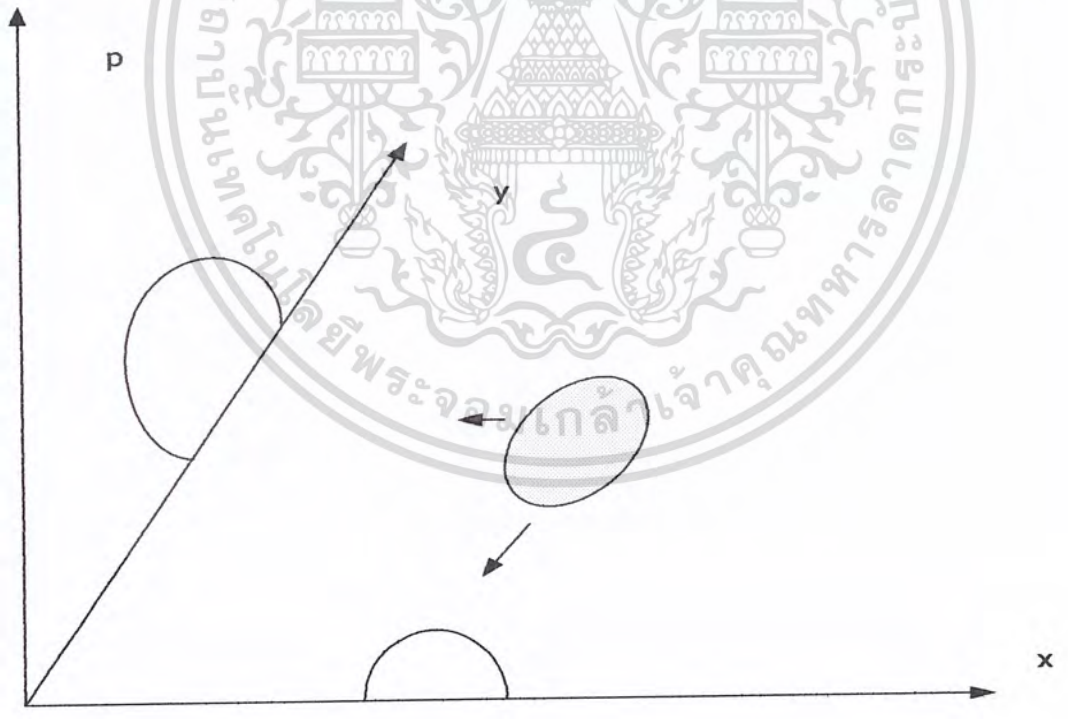
รูปที่ 2.20 แสดง การเกิดภาพตามวิธีการแบ็กโพรเจกชัน

ก.) โพรไฟล์ของวัตถุที่บันทึกได้ในทิศทางคือ 0° และ 90° วัดเทียบกับแกน y

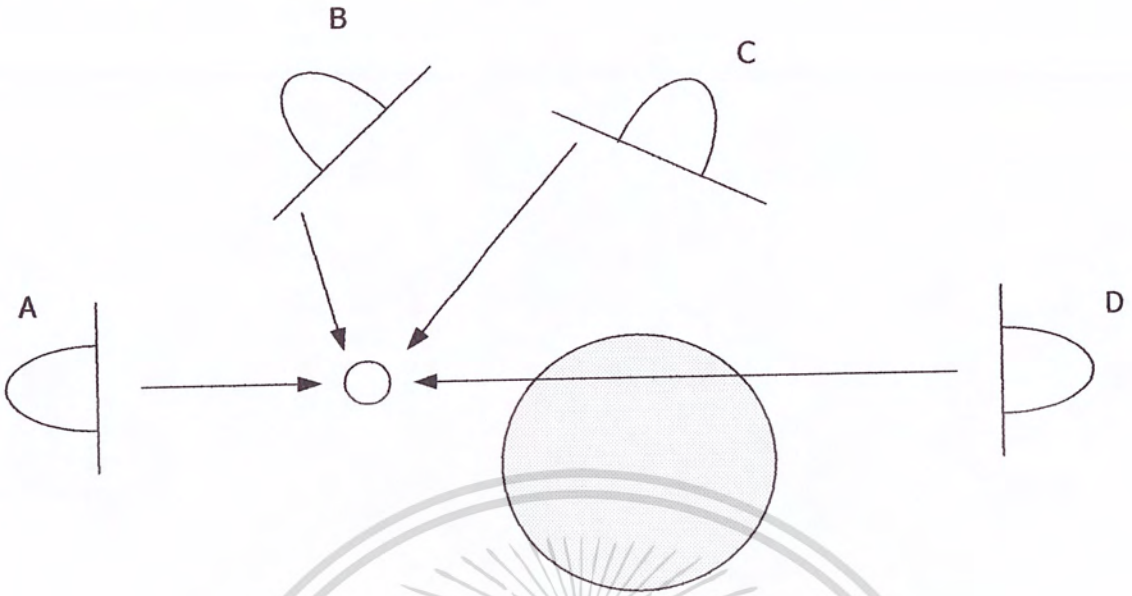
ข.) โพรไฟล์ในรูป ก. นำมาซ้อนกันอย่างเหมาะสมเพื่อสร้างภาพ

เพื่อความเข้าใจที่ชัดเจนขึ้นจะพิจารณารูป สมมุติว่าการเคลื่อนที่ตัดในแนวเส้นตรงครั้งแรกมุม $\phi = 0^\circ$ และ 90° จะได้โพรไฟล์ 1 ชุด หรือได้ $p(r, 0^\circ)$ การเคลื่อนทั้งสองมาซ้อนทับกันอย่างเหมาะดังรูป ข. จะเห็นว่า ตรงตำแหน่งที่สอดคล้องกับตำแหน่งของวัตถุ $p(r, 0^\circ)$ และ $p(r, 90^\circ)$ ที่มีค่าสูงมากจะรวมกันหรือเสริมกันทำให้เห็นเด่นเป็นภาพของวัตถุขึ้น แนวความคิดแบบนี้จึงเกิดเป็นสมการ ซึ่งนำเอา $p(r, \phi)$ ที่มี r และ ϕ ที่เหมาะสมมารวมกันตรงตำแหน่ง (x, y) ใดๆ ในระนาบ XY ที่จะสร้างภาพ

แบ็กโพรเจกชันไม่ใช่วิธีการที่ดีในการนำข้อมูลที่หัววัดรังสีเอ็กบันทึกไว้มาสร้างภาพ เพราะแต่ละเรย์ซัมไม่ได้นำไปรวมกันเฉพาะตรงจุดที่มีความหนาแน่นสูงเท่านั้น แต่จะนำไปรวมกันทุกๆ จุดตามเส้นทางของลำรังสีเอ็กซ์ เหตุนี้เองจึงทำให้ภาพที่สร้างขึ้นไม่คมชัดเท่าที่ควร โดยตรงขอบภาพจะเห็นเป็นแฉกรูปดาวเรียกว่า อทิแฟคต์รูปดาว (star artifact) ตามรูด้านล่าง จะเห็นได้ชัดเจนว่า จุดที่อยู่นองภาพของวัตถุจะมีความหนาแน่นสูงซึ่งได้รับอิทธิพลโดยตรงจากโพรไฟล์ A และ D อย่างไรก็ตามแม้วิธีนี้จะมีข้อบกพร่องอยู่บ้างแต่ก็เป็นพื้นฐานของวิธีการสร้างภาพแบบที่ต้องการใช้คณิตศาสตร์ชั้นสูงเข้าช่วย ซึ่งจะได้กล่าวต่อไป



(ก)



(ข)

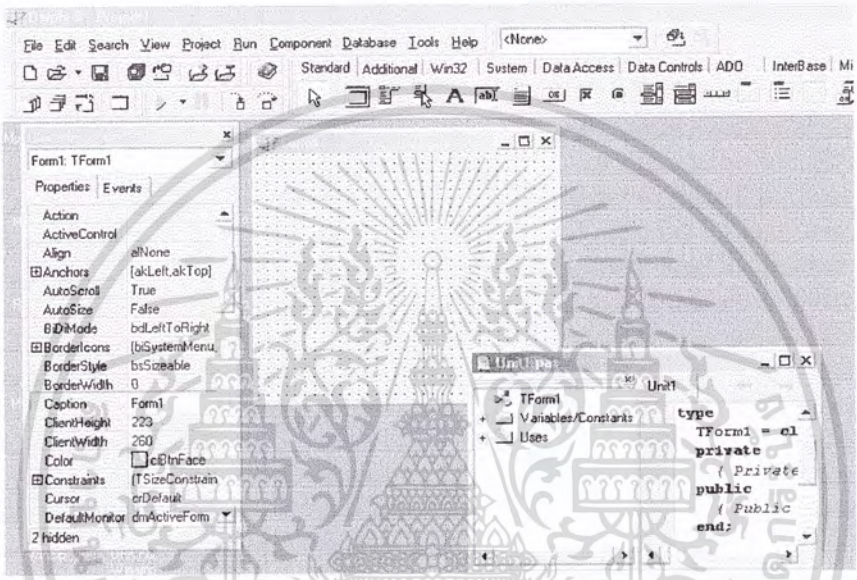
รูปที่ 2.21 ก.) โพรไฟล์ของวัตถุรูปวงกลมในสองทิศทาง

ข.) โพรไฟล์ และ มีส่วนทำให้ความหนาแน่นของจุดที่อยู่นอกภาพของวัตถุมีค่าสูงขึ้น เป็นสาเหตุทำให้เกิดความไม่คมชัด

บทที่ 3
การใช้งานโปรแกรม Delphi

3.1 การใช้งานโปรแกรม Delphi

เมื่อเรียกใช้โปรแกรม Delphi จะปรากฏจอภาพดังรูปที่ 3.1 ซึ่งประกอบด้วยหน้าต่าง 4 หน้าต่างดังนี้



รูปที่ 3.1 หน้าต่างของ Delphi

1. หน้าต่างหลัก (Main Window)

เป็นหน้าต่างแรกที่ปรากฏเมื่อเรียกใช้โปรแกรม Delphi ซึ่งประกอบไปด้วย 3 ส่วนคือ Manubar , Speedbar และ Component Palette

2. หน้าต่าง Object Inspector

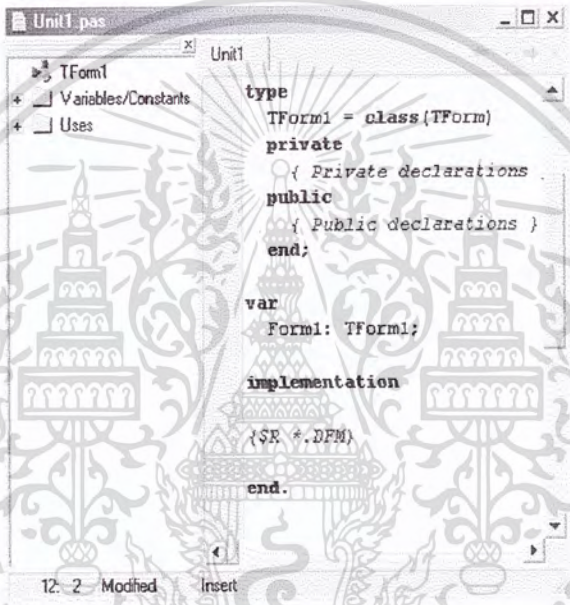
Object Inspector เป็ที่สำหรับกำหนดคุณสมบัติและโปรซีเดอร์ที่ควบคุม Event ของ Component ในฟอร์มด้านบนของ Object Inspector จะเป็นคอมโบบ๊อกที่แสดงชื่อของบคอมโพเนนต์และชนิดของคอมโพเนนต์ เช่นถ้าเราเลือกคอมโพเนนต์ “From1: Tfrom1”โดยที่ From1 เป็นชื่อของฟอร์มที่เราเลือกและ TFrom เป็นชื่อชนิดของคอมโพเนนต์ (Delphi จะใส่ T ไว้ที่หน้าชื่อชนิดของคอมโพเนนต์ที่เรียกใช้กันทั่วไป เช่น Tfrom , Tbutton ซึ่งจะแทนชนิดของคอมโพเนนต์นั้นๆ)

3. หน้าต่าง From Designer

From Designer ใช้สำหรับออกแบบส่วนที่ติดต่อกับผู้ใช้ซึ่งเราสามารถนำเอาคอมโพเนนต์ต่างๆจาก Component Palette มาวางลงบน From Designer โดยที่เราสามารถแก้ไขขนาดย้ายคอมโพเนนต์ไปมา รวมทั้งเพิ่มหรือลบคอมโพเนนต์ออกจากฟอร์มได้ โดยเราสามารถเปรียบ From Designer ได้กับกระดาษวาดเขียนที่เราสามารถเขียนหรือลบส่วนที่เราวาดได้

4. หน้าต่าง Code Editor

Code Editor เป็น Text Editor สำหรับการเขียนโปรแกรมใน Delphi ซึ่งประกอบไปด้วยคุณสมบัติต่างๆที่อำนวยความสะดวกในการเขียนโปรแกรม Delphi ดังนี้



รูปที่ 3.2 หน้าต่าง Code Editor

ส่วนล่างของ Code Editor จะแสดงสถานะการทำงานซึ่งประกอบด้วย

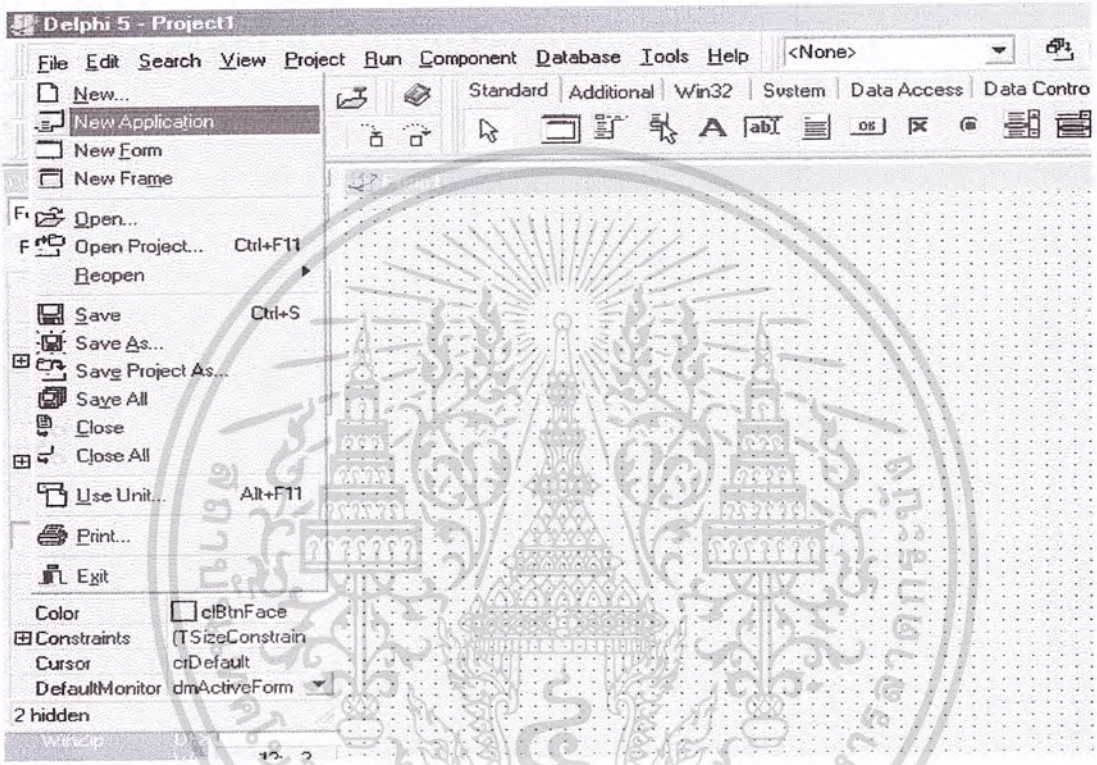
- ตำแหน่งของเคอร์เซอร์บอกว่าตอนนี้เคอร์เซอร์อยู่ที่บรรทัดไหนและคอลัมน์ใด
- สถานะการแก้ไขเป็นตัวบอกว่าไฟล์นั้นได้มีการแก้ไขแล้วหรือยังถ้ามีการแก้ไขแล้วจะปรากฏคำว่า Modified
- สถานะการพิมพ์ เป็นตัวบอกสถานะการพิมพ์ว่าเป็นการพิมพ์แทรก (Insert) หรือพิมพ์ทับ (Overwrite)

3.2 การทำงานและการจัดการกับโปรเจ็ค

โปรเจ็ค (Project) คือไฟล์ที่ใช้สำหรับเก็บรวบรวม ฟอร์ม , Unit ต่างๆเข้าไว้ด้วยกันเพื่อใช้ในการสร้างแอปพลิเคชั่น เมื่อเราทำการคอมไพล์โปรเจ็ค เราก็จะได้แอปพลิเคชั่น 1 ตัว

การสร้างแอปพลิเคชั่นใหม่ (New Application)

เมื่อเราทำการสร้างแอปพลิเคชั่นขึ้นมาใหม่ Delphi จะทำการสร้างโปรเจ็คใหม่ให้ 1 โปรเจ็ค ซึ่งประกอบด้วยฟอร์มพื้นฐาน 1 ฟอร์มและ Unit ที่ใช้สำหรับฟอร์มนั้น 1 Unit โดยเราสามารถสร้างแอปพลิเคชั่นใหม่ได้โดยเลือกที่เมนู File>New Application ดังรูปที่ 4.3



รูปที่ 3.3 การสร้างแอปพลิเคชั่นใหม่

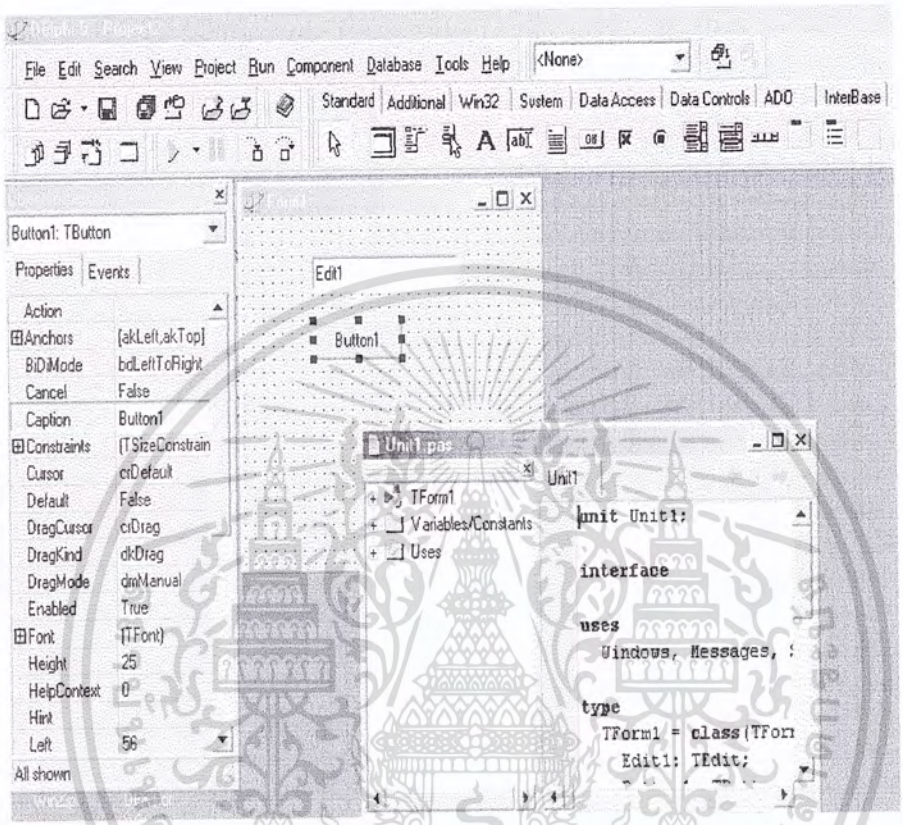
การสร้างแอปพลิเคชั่นแรกใน Delphi

เราจะลองสร้างแอปพลิเคชั่นแรกใน Delphi ซึ่งแอปพลิเคชั่นนี้จะสามารถรับข้อมูลและโต้ตอบกับผู้ใช้งานได้ คือเมื่อเราพิมพ์ข้อความลงไปในช่อง Edit แล้วคลิกที่ปุ่ม 

โดยมีขั้นตอนการสร้างดังต่อไปนี้

1. เลือกคำสั่ง File>New Application
2. คลิกเม้าท์ที่ 
3. คลิกบนฟอร์มเพื่อสร้างออปเจ็ค Edit
4. คลิกเม้าท์ที่

5. คลิกบนฟอร์มเพื่อสร้างออปเจ็ค Button จากนั้น Double Click บนออปเจ็ค Button 1



รูปที่ 3.4 การสร้างแอปพลิเคชั่นแรกใน Delphi

6. Code Editor จะปรากฏขึ้นมาด้านหน้า
7. พิมพ์คำสั่ง ShowMassage (Edit1.Text)
8. เลือกเมนูRun>Run เพื่อรันโปรเจ็ค
9. จากแอปพลิเคชั่นที่สร้างขึ้น เมื่อใส่ข้อความในช่อง Edit1 แล้วคลิกที่ปุ่ม Button1 จะปรากฏไดอะล็อกแสดงข้อความที่เราใส่เข้าไป

การบันทึกโปรเจ็ค

การบันทึกข้อมูลใน Delphi สามารถทำได้ 4 ลักษณะ ได้แก่

1. Save

เป็นการสั่งให้บันทึกข้อมูล โดยถ้าเป็นการบันทึกในครั้งแรก Delphi จะมีไดอะล็อกขึ้นมาถามว่าต้องการบันทึกข้อมูลด้วยไฟล์ชื่ออะไรและสำหรับกรณีที่เราต้องการบันทึกข้อมูลที่เปิดขึ้นมาแก้ไข

จะเป็นการใช้ชื่อไฟล์เดิมในการบันทึกข้อมูล เราสามารถทำการสั่งให้บันทึกข้อมูลด้วยวิธีการนี้โดยให้คลิกที่เมนู File>Save

2. Save As

จะเป็นการบันทึกไฟล์ที่ถูกแก้ไขด้วยชื่อไฟล์ใหม่ โดยที่จะมีหน้าต่างไดอะล็อกให้เราใส่ชไฟล์ที่เราจะต้องการจะเก็บข้อมูล โดยมีขั้นตอนดังนี้

- คลิกที่เมนู File>Save As
- ใส่ชื่อไฟล์ที่ต้องการบันทึกพร้อมทั้งระบุโฟลเดอร์ที่จะจัดเก็บไฟล์
- คลิกที่ปุ่ม Save
- เมื่อทำคำสั่งเสร็จแล้วจะปรากฏไฟล์ใหม่ดังที่เรากำหนด

3. Save Project As

จะเป็นการบันทึกเฉพาะไฟล์โปรเจ็คด้วยชื่อใหม่ ซึ่งไฟล์ที่ได้จะมีนามสกุลเป็น .DPR โดยมีขั้นตอนการบันทึกดังต่อไปนี้

- คลิกที่ปุ่ม File>Save Project As
- ใส่ชื่อไฟล์โปรเจ็คที่ต้องการบันทึกพร้อมทั้งระบุโฟลเดอร์
- คลิกที่ปุ่ม Save
- จะปรากฏไฟล์ใหม่เมื่อทำคำสั่งเสร็จแล้ว

4. Save All

เป็นการบันทึกข้อมูลทั้งหมดที่ถูกแก้ไขในโปรเจ็คที่กำลังทำงานอยู่ในขณะนั้น จะมีขั้นตอนเหมือนกับการใช้คำสั่ง Save กับทุกไฟล์แล้วตามด้วยคำสั่ง Save Project As โดยถ้าไฟล์ใดมีการตั้งชื่อไฟล์แล้วจะใช้ชื่อไฟล์เดิม แต่ถ้ายังไม่มีการตั้งชื่อไฟล์ก็จะปรากฏไดอะล็อกขึ้นมาถามว่าจะใช้ชื่ออะไร โดย Delphi จะถามชื่อของ Unit และชื่อของโปรเจ็คตามลำดับ ชื่อ Unit คือชื่อของไฟล์ .PAS ที่เก็บโปรแกรมที่เราเขียนขึ้นและชื่อโปรเจ็คคือชื่อไฟล์ .DPR ซึ่งเป็นตัวเก็บว่าในโปรเจ็คของเราประกอบด้วยฟอร์มและ Unit อะไรบ้าง

ชื่อของ Unit และชื่อของโปรเจ็คจะต้องเป็นชื่อที่ไม่ซ้ำกัน Delphi ไม่อนุญาตให้ใช้ชื่อ Unit กับชื่อโปรเจ็คเป็นชื่อเดียวกัน

บทที่ 4
หลักการทำงาน

หลักการทำงานโดยรวมของวงจรโปรเจ็คนี้จะแยกเป็นส่วนของภาคต่างๆ ได้แก่ ภาคจ่ายไฟ (แหล่งจ่ายไฟสูง และแหล่งจ่ายไฟขับมอเตอร์) , วงจรไดรฟ์ควบคุมการสแกน , วงจรคอลโทรลด้วย MCS-51 , และส่วนของโปรแกรมสั่งงานผ่านพอร์ตอนุกรมของคอมพิวเตอร์ซึ่งจะได้กล่าวในบทต่อไป

4.1 ภาคจ่ายไฟ

ในที่นี้ได้ทำการแยกภาคจ่ายไฟออกเป็น 2 ชุด คือ แหล่งจ่ายไฟสูงสำหรับวงจรไดร์ฟอัตร้าโซนิค และ แหล่งจ่ายไฟเลี้ยงวงจรที่มีแรงดันต่ำ 5V และ 12V สำหรับวงจรทั่วไปและวงจรไดร์ฟมอเตอร์

4.1.1 แหล่งจ่ายไฟแรงดันต่ำ

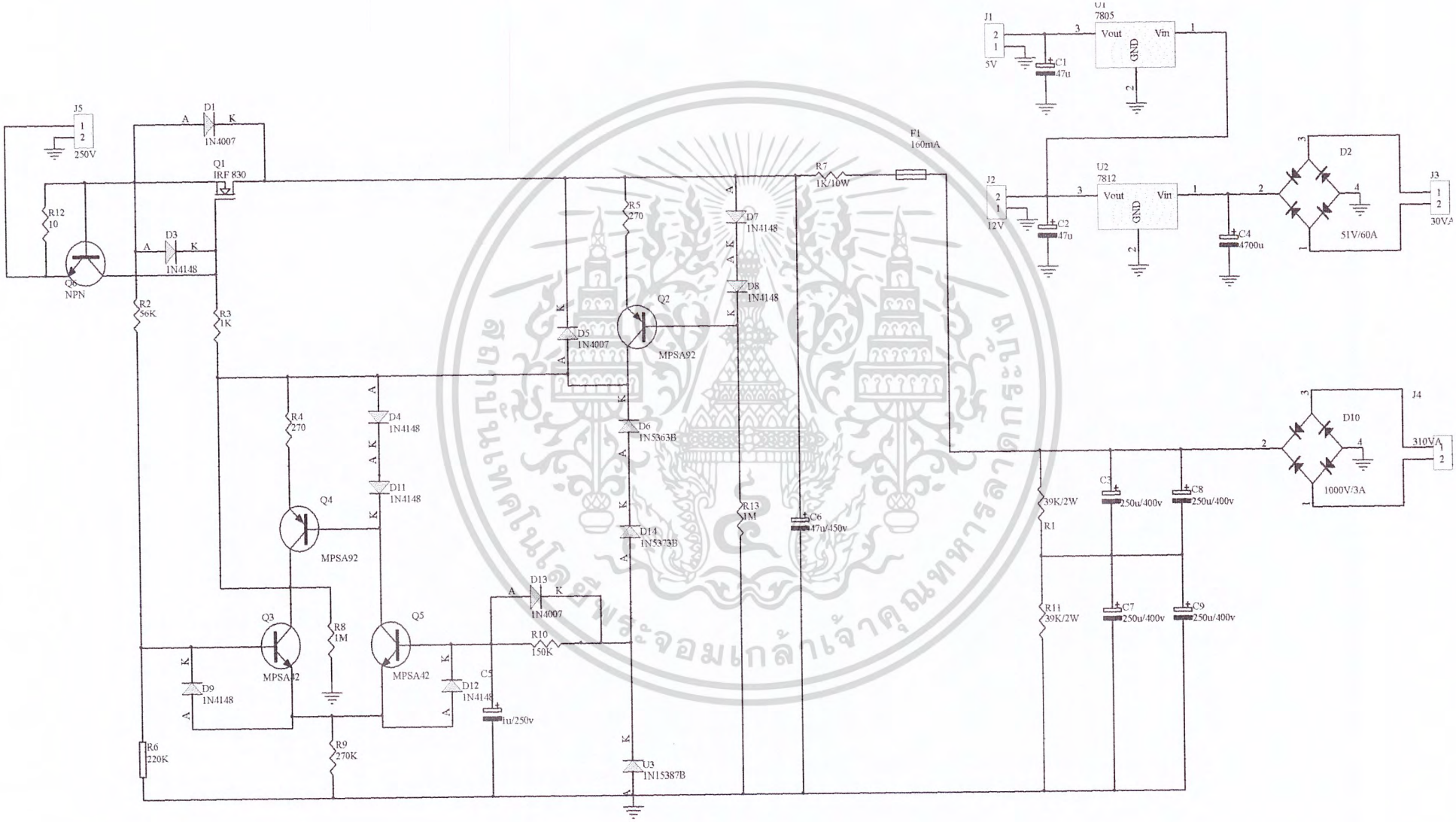
วงจรที่ใช้เป็นไดโอดเรียงกระแสแบบบริดจ์แล้วผ่าน IC เร็กกลูเลตเบอร์ 7812 และ 7805 ซึ่งจะให้แรงดัน 12V และ 5V ตามลำดับ ซึ่งแรงดันชุดนี้มีไว้สำหรับ วงจรไดร์ฟมอเตอร์ และวงจรไมโครคอนโทรลเลอร์ MCS-51

4.1.2 แหล่งจ่ายไฟสูง

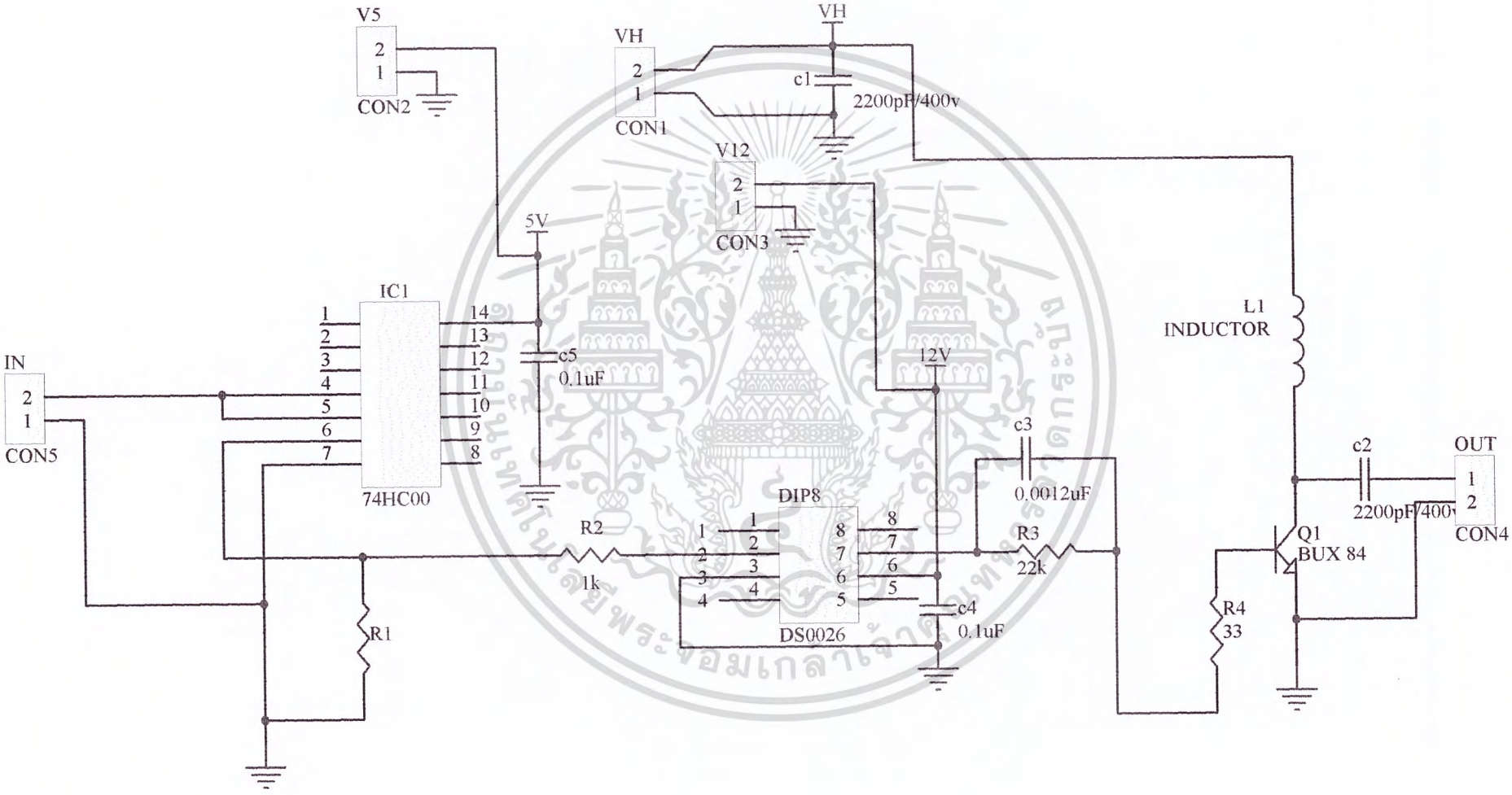
การส่งคลื่นอัลตร้าโซนิคจำเป็นต้องดันซึ่งมีค่ามาก ภาคจ่ายไฟนี้จะทำการผลิตแรงดัน 250V และยังมีแรงดัน 5V และ 12V สำหรับอุปกรณ์บางตัว หลังการทำงานของภาคนี้จะให้ไฟสูงจากวงจรเร็กกลูเลตผ่านวงจรฟิลเตอร์เพื่อลดการกระเพื่อมของแรงดัน จากนั้นวงรจดิฟแอมป์จะทำหน้าที่ควบคุมแรงดันที่เอาท์พุทให้มีความคงที่

4.2 วงจรไดร์ฟมอเตอร์

วงจรไดร์มอเตอร์จะมีด้วยกัน 2 ชุดเพื่อควบคุมควบคุมมอเตอร์ 2 ตัว โดยตัวแรกจะทำการเลื่อนให้หัวส่งหัวรับคลื่นอัลตร้าโซนิคเคลื่อนไปตามแนวนอน และอีกตัวจะทำการเลื่อนหัวรับหัวส่งอัลตร้าโซนิคในแนวตั้ง ในรูปเราจะใช้ IC เบอร์ L298 แบบ Duall full-bridge ซึ่งแสดงตารางการทำงานสำหรับควบคุมการหมุนของมอเตอร์ตามตารางที่ 4.1



วงจรภาคจ่ายไฟสูง



วงจรไดร์ฟอัลตร้าโซนิค

Input		Function
Ven = H	C = H; D = L;	Turn Right
	C = L; D = H;	Turn Left
	C = D	Fast Motor Stop
Ven = L	C = X ; D = C	Free Running Motor Stop

L = Low H = High X = Don't Care

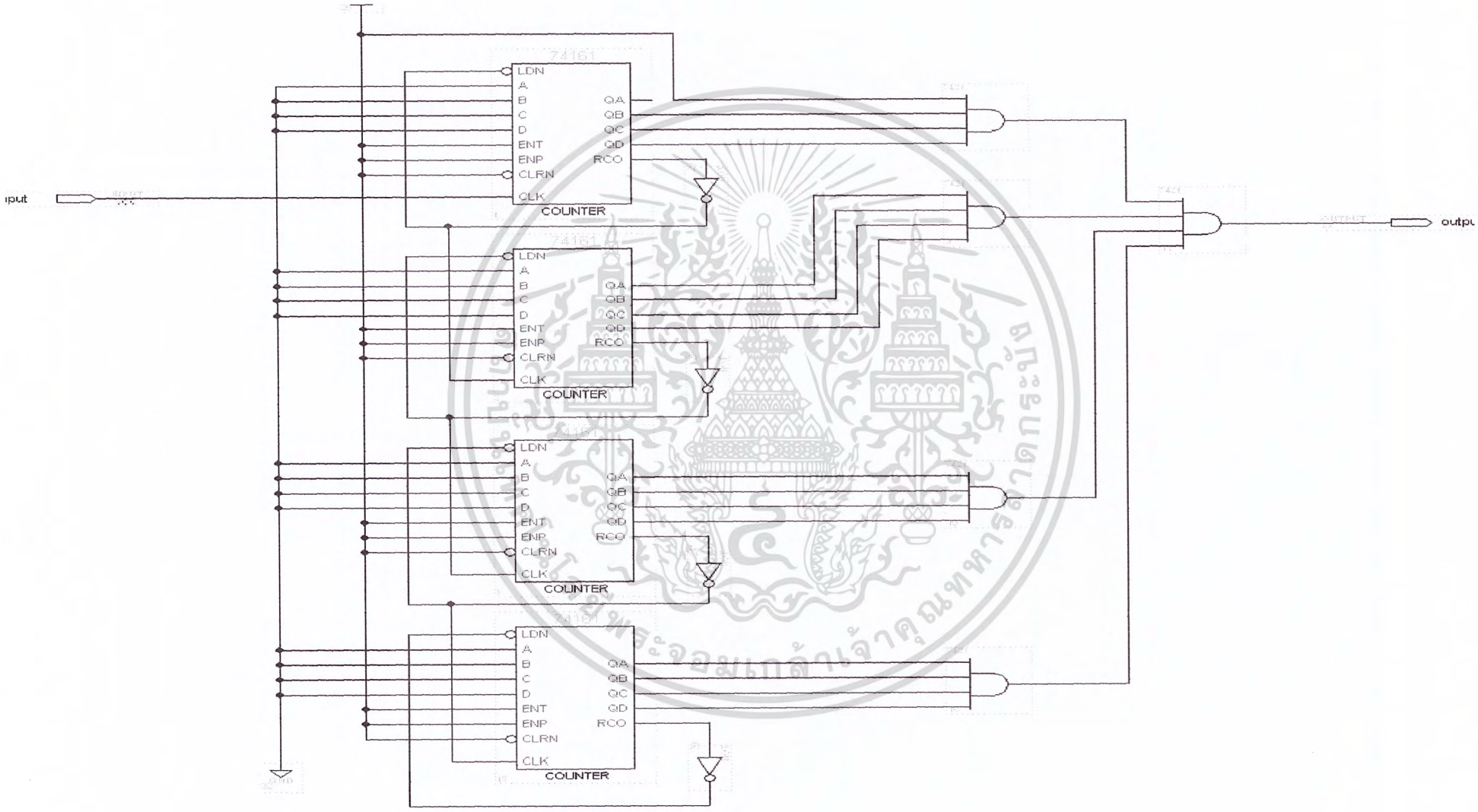
ตารางที่ 4.1 ควบคุมการการทำงานของ IC L298

4.3 การทำงานของไมโครคอลโทรลเลอร์ MCS-51

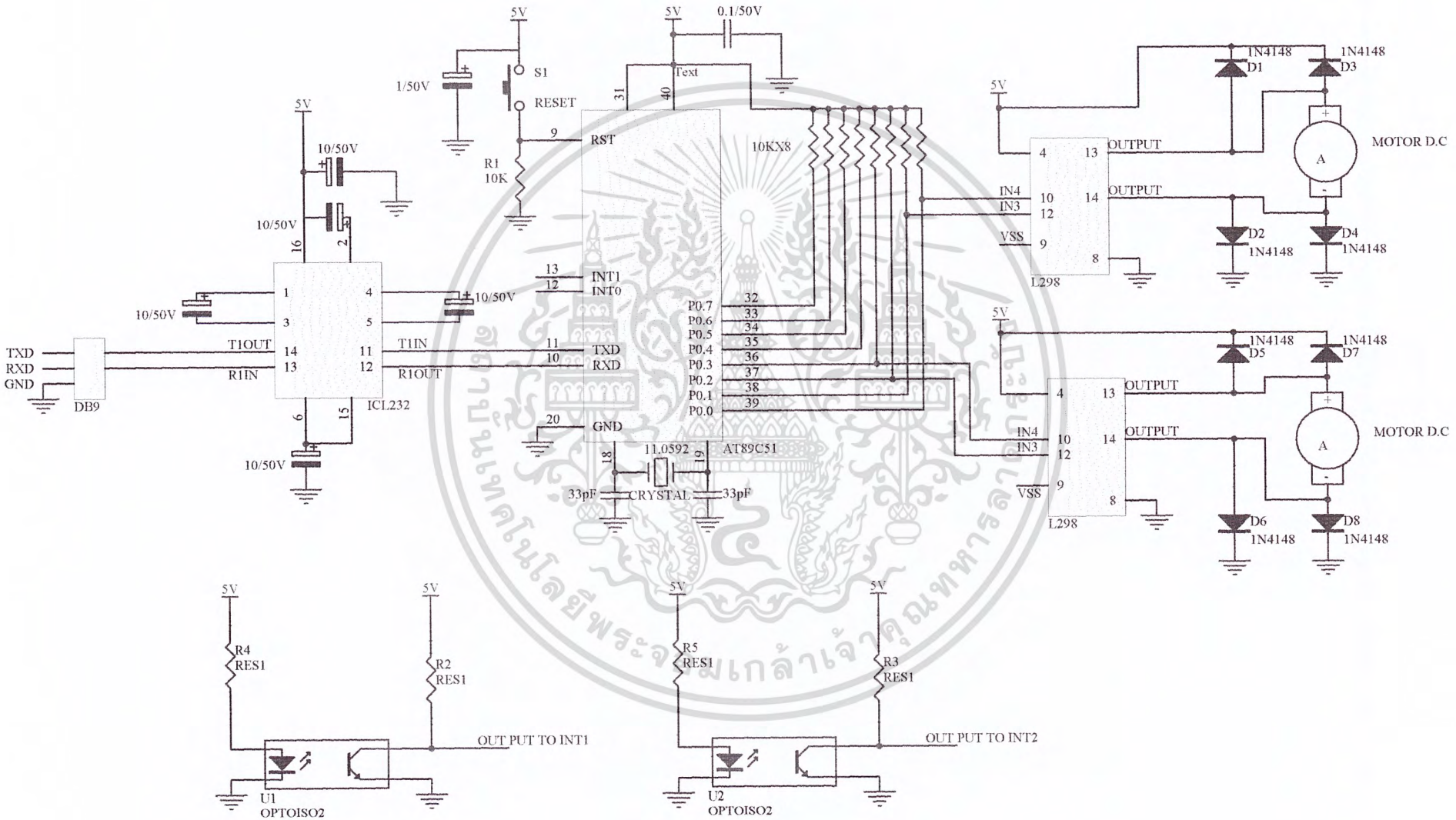
ในโปรเจ็คชุดนี้จะมีส่วนประกอบด้วยกัน 2 ส่วนหลักอันได้แก่ ส่วนควบคุมมอเตอร์สแกนและส่วนของการส่งคลื่นอัลตร้าโซนิค แต่จะใช้ไมโครคอนโทลเลอร์ MCS-51 ตัวเดียวกันเป็นตัวควบคุมการทำงาน ในการทำงานของส่วนควบคุมมอเตอร์สแกนนั้นจะประกอบด้วยมอเตอร์ 2 ตัวสำหรับการเลื่อนหัวส่ง-หัวรับคลื่นอัลตร้าโซนิคทางด้านแนวนอนและแนวตั้ง ในตอนแรกเราจะทำการกำหนดค่าขนาดของวัตถุลงในส่วนรับค่าที่คอมพิวเตอร์ จากนั้น MCS-51 จะรับคำสั่งที่ได้ทำการขับมอเตอร์ทางด้านแนวนอนหมุนทุกครั้งที่มอเตอร์หมุนเป็นระยะทางที่ป้อนจากคอมพิวเตอร์ จะส่งสัญญาณส่งไปให้ MCS-51 สั่งให้มอเตอร์หยุดทำงาน จากนั้น MCS-51 ก็จะทำการสร้างพัลส์เพื่อส่งไปยังวงจรไดรฟ์คลื่นอัลตร้าโซนิค ซึ่งจะส่งออกไปเพียงลูกเดียวเท่านั้น เราก็จะได้สัญญาณออกมา 1 จุด จากนั้น MCS-51 ก็จะส่งข้อมูลมาบอกว่าทำเสร็จแล้ว คอมพิวเตอร์ก็จะส่งข้อมูลมาให้ MCS -51 ทำงานต่อ มอเตอร์ก็จะทำการหมุนต่อไปเป็นเช่นนี้เรื่อยๆจนการสแกนวัตถุในแนวนอนเป็นไปตามระยะทางที่เรากำหนดไว้ในตอนแรก การสแกนนี้จะทำตั้งแต่ด้านบนลงมาจนถึงฐานของวัตถุ ในส่วนของสัญญาณที่ได้นักศึกษาในระดับปริญญาโทจะใช้ซอฟแวร์ที่เขียนขึ้นเพื่อสร้างรูปแบบของวัตถุออกมา

4.4 การทำงานของ Delphi

จากรูปที่ 4.4 แสดงหน้าตาของโปรแกรมที่ใช้ควบคุมการทำงานของวงจร ซึ่งจะทำการป้อนค่าระยะห่างของแต่ละ Step ที่ช่อง Step ซึ่งจะป้อนค่าได้ตั้งแต่ 0-99 และทำการป้อนค่าของจำนวนครั้งที่จะทำการวัดที่ช่อง Length จะเห็นได้แล้วว่าถ้าไม่มีการ Connect ปุ่มต่างๆ ก็ไม่สามารถที่จะกดได้ เมื่อเราป้อนค่าที่ต้องการแล้วเลือกสั่งให้ Motor จะหมุนไปทิศทางใดก็ได้ที่มีไว้แล้วกดปุ่ม Send Computer จะส่งค่าที่กำหนดเป็นมาตรที่ Computer และ MCS -51 รู้กันเมื่อส่งเสร็จแล้ว Computer ก็จะรอเช็คว่าค่าที่ได้จาก MCS -51 ที่ MCS -51 ตอบกลับมาตรงกับค่าที่กำหนดไว้หรือเปล่าโดยจะมีค่าของปุ่มแต่ละปุ่มดังนี้



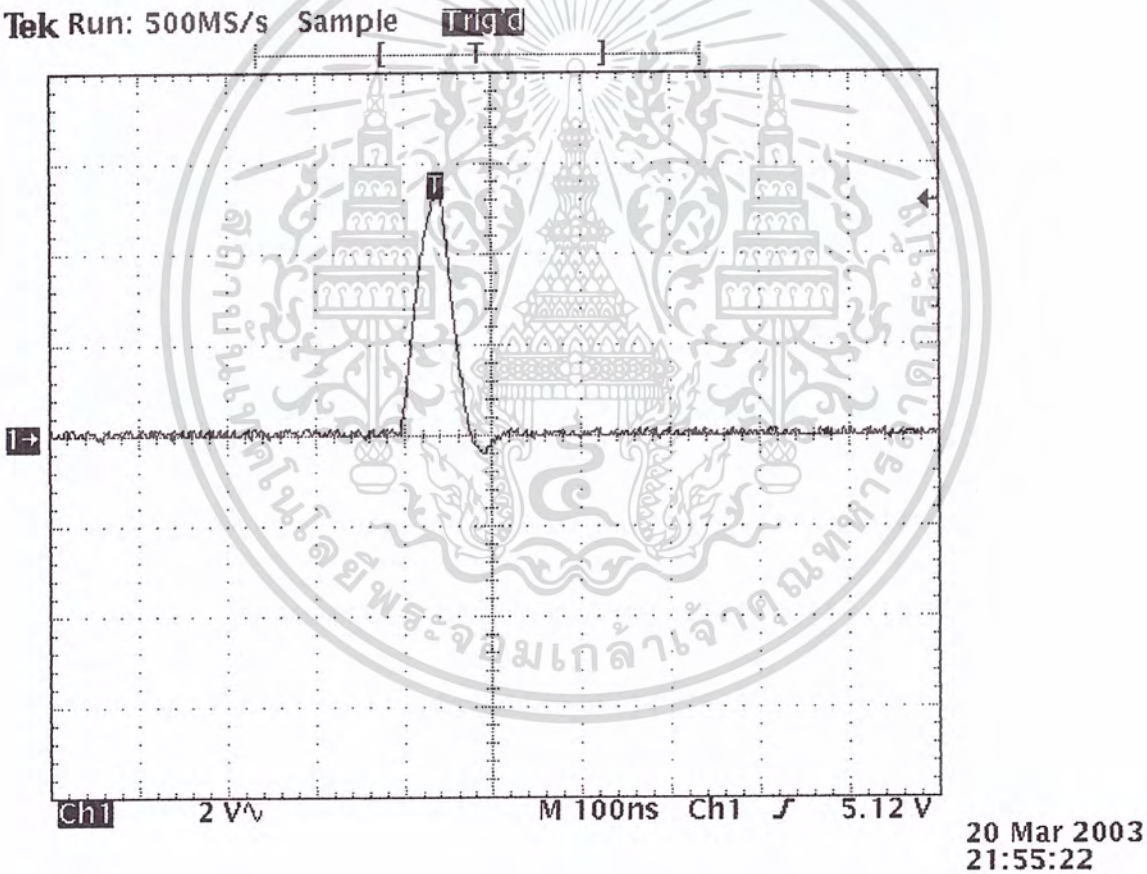
วงจรสร้างพัลส์ด้วยเคาเตอร์



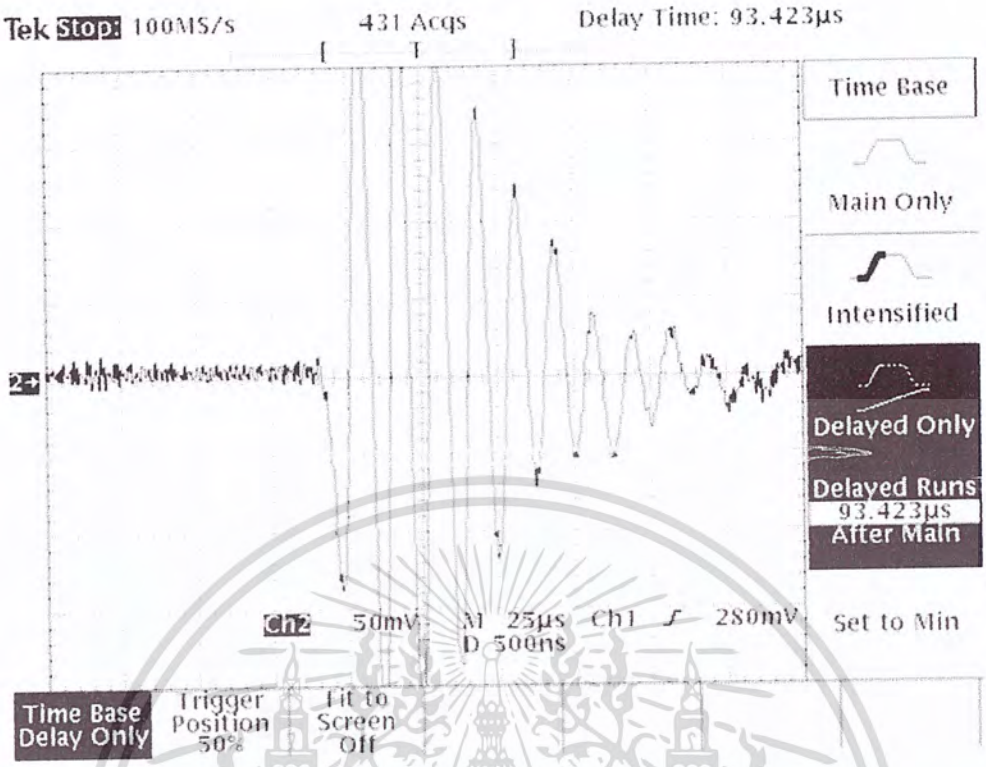
วงจรควบคุมด้วย MCS-51

บทที่ 5
สรุปผลการทดลอง

ในบทนี้จะกล่าวถึงผลสัญญาณที่ได้จากการสแกนของหัวรับ – หัวส่ง ของอัลตร้าโซนิคที่มุมต่างๆ ได้แก่ มุม 0 องศา, มุม 90 องศา และ 170 องศา ดังแสดงรูปที่ 5.2, 5.3, 5.4 ตามลำดับส่วนรูปที่ 5.5 เป็นสัญญาณภาพแบบ 3 มิติที่นำเอาสัญญาณจากการสแกนที่มุม 0 องศา – 170 องศา นำมารวมกันโดยใช้โปรแกรม MATLAB ประมวลผลของสัญญาณ และรูปที่ 5.6 เป็นภาพตัดขวางของวัตถุใน 1 ระนาบที่ได้จากการประมวลโดย MATHLAB ส่งรูปที่ 5.1 เป็นพัลส์ที่ได้จากวงจร counter เพื่อสัญญาณอินพุทให้กับวงจรไดรฟ์คลื่นอัลตร้าโซนิค



รูปที่ 5.1 แสดงสัญญาณพัลส์ที่ได้จากวงจรเคาร์เตอร์



รูปที่ 5.2 แสดงสัญญาณที่หัวรับของอัลตร้าโซนิคที่ทำการวัด

ผลที่ได้รับจากการดำเนินงานเป็นไปตามวัตถุประสงค์ คือสามารถนำไมโครคอนโทรลเลอร์ MCS – 51 มาควบคุมการทำงานของมอเตอร์ และสร้างคลื่นอัลตร้าโซนิคความถี่สูงให้กับหัวส่งอัลตร้าโซนิค อีกทั้งนำสัญญาณที่ได้จากหัวรับผ่านสโคป เข้าสู่คอมพิวเตอร์โดยใช้มาตราฐาน GPIB เพื่อเก็บสัญญาณที่ได้ เพื่อนำไปประมวลภาพในรูปของระนาบวัตถุได้

ปัญหาที่พบจากการทำงาน เนื่องจากคลื่นอัลตร้าโซนิคเป็นคลื่นที่มีความถี่สูงจึงมีปัญหาของการรบกวนของสัญญาณ noise ต่างๆ และในการสแกนวัตถุในแต่ระนาบจะใช้เวลานานมาก เพราะการเลื่อนของหัวรับ – หัวส่ง อัลตร้าโซนิค ในน้ำต้องเคลื่อนที่อย่างช้าๆ เพื่อให้การกระเพื่อมของน้ำมีน้อยที่สุด เพื่อป้องกันความผิดพลาดของสัญญาณที่ส่งออกจากหัวส่งสู่หัวรับ

ผนวก

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, TeEngine, Series, TeeProcs, Chart, ExtCtrls, aGPIB, CPDrv;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Chart1: TChart;
    Series1: TLineSeries;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    CPDrv: TCommPortDriver;
    aGPIB1: TaGPIB;
    Button7: TButton;
    SaveDialog1: TSaveDialog;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure CPDrvReceiveData(Sender: TObject; DataPtr: Pointer;
      DataSize: Cardinal);
    procedure Button5Click(Sender: TObject);
```

```
  procedure Button6Click(Sender: TObject);
  procedure Button8Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;
 WriteBuff:Packed array[0..35] of char;
 ReadBuff:Packed array[0..1010] of byte;

implementation

uses Unit2;

{SR *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
// setport
 CPDrv.Port := pnCOM2;
 // Do nothing if already connected
    if cpDrv.Connected then
     exit;
 // Try connecting
    if not cpDrv.Connect then
    begin
     Application.MessageBox( 'Could not connect to serial port.'#13#10+
                  'Please, check settings and try again.',
                  'Error',
                   MB_OK or MB_ICONERROR );
     exit;
    end;
```

```
 auto := false;

end;

procedure TForm1.Button1Click(Sender: TObject);
var  i:Byte;
begin
  CPDrv.SendByte(1);
  i := strtoint(Edit1.text);
  CPDrv.SendByte(i);
  Button1.Enabled := False;
  Button2.Enabled := False;
  Button3.Enabled := False;
  Button4.Enabled := False;
  Button7.Enabled := True;
end;

procedure TForm1.Button2Click(Sender: TObject);
var  i:Byte;
begin
  CPDrv.SendByte(2);
  i := strtoint(Edit1.text);
  CPDrv.SendByte(i);
  Button1.Enabled := False;
  Button2.Enabled := False;
  Button3.Enabled := False;
  Button4.Enabled := False;
  Button7.Enabled := True;
end;

procedure TForm1.Button3Click(Sender: TObject);
var  i:Byte;
begin
  CPDrv.SendByte(3);
  i := strtoint(Edit1.text);
```

```
  CPDrv.SendByte(i);
  Button1.Enabled := False;
  Button2.Enabled := False;
  Button3.Enabled := False;
  Button4.Enabled := False;
  Button7.Enabled := True;
end;

procedure TForm1.Button4Click(Sender: TObject);
var  i:Byte;
begin
  CPDrv.SendByte(4);
  i := strtoint(Edit1.text);
  CPDrv.SendByte(i);
  Button1.Enabled := False;
  Button2.Enabled := False;
  Button3.Enabled := False;
  Button4.Enabled := False;
  Button7.Enabled := True;
end;

procedure TForm1.Button7Click(Sender: TObject);
var  i:Byte;
begin
  CPDrv.SendByte(0);
  CPDrv.SendByte(0);
  Button1.Enabled := True;
  Button2.Enabled := True;
  Button3.Enabled := True;
  Button4.Enabled := True;
  Button8.Enabled := True;
  Button7.Enabled := False;
  Form2.Timer1.Enabled := False;
end;
```

```
procedure TForm1.CPDrvReceiveData(Sender: TObject; DataPtr: Pointer;
 DataSize: Cardinal);
var s: string;
   temp: byte;
begin
 if auto then
 begin
  Form2.Timer2.Enabled := true;
 end
 else
 begin
 // Convert incoming data into a string
 s := StringOfChar( ' ', DataSize );
 move( DataPtr^, pchar(s)^, DataSize );
 // Exit if s is empty. This usually occurs when one or more NULL characters
 // (chr(0)) are received.
 while pos( #0, s ) > 0 do
  delete( s, pos( #0, s ), 1 );
 if s = '' then
  exit;
 temp := byte(s[1]);
 if temp = 170 then
  begin
   Button1.Enabled := True;
   Button2.Enabled := True;
   Button3.Enabled := True;
   Button4.Enabled := True;
   Button7.Enabled := False;
  end;
 end;
end;

procedure TForm1.Button5Click(Sender: TObject);
var i : integer;
begin
```

```
Button5.Enabled := false;

aGPIB1.LoadInitial;
aGPIB1.SendIFC(0);
aGPIB1.DevClear(0,1);

writebuff:='data:source ch2;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
writebuff:='data:encdg rpbinary;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
writebuff:='data:width 1;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
writebuff:='data:start 1;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
writebuff:='data:stop 1000;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
writebuff:='curve?;';
aGPIB1.send(0,1,writebuff,length(writebuff),1);
// sleep(1000);
aGPIB1.receive(0,1,readbuff,length(readbuff),1);

Series1.Clear;
for i := 0+6 to 999+6 do
  Series1.AddXY(i,readbuff[i],' ',clred);

Button5.Enabled := True;
Button6.Enabled := True;
end;

procedure TForm1.Button6Click(Sender: TObject);
var fn : file of byte;
   i : integer;
begin
 if savedialog1.Execute then
  begin
```

```
    assignfile(fn,Savedialog1.FileName);
    rewrite(fn);
    for i := 0+6 to 999+6 do
      write(fn,readbuff[i]);
    closefile(fn);
   end;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  form2.ShowModal;
end;

end.
```

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    ComboBox1: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    SaveDialog1: TSaveDialog;
    Timer1: TTimer;
    Timer2: TTimer;
    procedure BitBtn1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;
  count,n,iii:integer;
  fn : file of byte;
  auto : bool;
```

```
implementation

uses Unit1;

{$R *.DFM}

procedure TForm2.BitBtn1Click(Sender: TObject);
begin
  assignfile(fn,'c:\ResultUltrasonic\Prj.dat');
  rewrite(fn);

  count :=  strtoint(Edit2.Text);
  n :=  strtoint(Edit1.Text);

  Timer1.Enabled := true;
  iii := 0;

  Form1.Button1.Enabled := False;
  Form1.Button2.Enabled := False;
  Form1.Button3.Enabled := False;
  Form1.Button4.Enabled := False;
  Form1.Button8.Enabled := False;
  Form1.Button7.Enabled := True;

end;

procedure TForm2.Timer1Timer(Sender: TObject);
var i,ii:integer;
begin
  if iii < count then
    begin
      iii := iii + 1;
      // control
      if combobox1.ItemIndex = 0 then
         form1.CPDrv.SendByte(1)
```

```
    else
       form1.CPDrv.SendByte(2);
    form1.CPDrv.SendByte(n);
    auto := true;

    Timer1.Enabled := false;

//    sleep(5000);
   end
  else
   begin
    closefile(fn);
    Timer1.Enabled := False;
    auto := false;
    Form1.Button1.Enabled := True;
    Form1.Button2.Enabled := True;
    Form1.Button3.Enabled := True;
    Form1.Button4.Enabled := True;
    Form1.Button8.Enabled := True;
    Form1.Button7.Enabled := False;
   end;
end;

procedure TForm2.Timer2Timer(Sender: TObject);
var ii: integer;
begin
    // read
    form1.aGPIB1.LoadInitial;
    form1.aGPIB1.SendIFC(0);
    form1.aGPIB1.DevClear(0,1);

    writebuff:='data:source ch2';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
    writebuff:='data:encdg rpbinary';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
```

```
    writebuff:='data:width 1';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
    writebuff:='data:start 1';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
    writebuff:='data:stop 1000';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
    writebuff:='curve?';
    form1.aGPIB1.send(0,1,writebuff,length(writebuff),1);
    form1.aGPIB1.receive(0,1,readbuff,length(readbuff),1);

    form1.Series1.Clear;
    for ii := 0+6 to 999+6 do
      form1.Series1.AddXY(ii,readbuff[ii],'',clred);

    //save
    for ii := 0+6 to 999+6 do
      write(fn,readbuff[ii]);
    auto := false;
    Timer2.Enabled := false;
    Timer1.Enabled := true;
end;

end.
```

```
        ORG     0000H                   ; Reset Vector
MAIN:   MOV     P0,#0FFH
        MOV     P1,#0FFH
        CALL    DELAY_500ms
        MOV     P0,#00H
        MOV     P1,#00H
        CALL    DELAY_500ms
        MOV     P0,#01H
        MOV     P1,#0FFH
        CALL    DELAY_500ms
        MOV     P0,#02H
        MOV     P1,#00H
        CALL    DELAY_500ms
        MOV     P0,#03H
        MOV     P1,#0FFH
        CALL    DELAY_500ms
        MOV     P0,#04H
        MOV     P1,#00H
        CALL    DELAY_500ms
        MOV     P0,#01H
        MOV     P1,#0FFH
        CALL    DELAY_500ms
        MOV     P0,#02H
        MOV     P1,#00H
        CALL    DELAY_500ms
        MOV     P0,#03H
        MOV     P1,#0FFH
        CALL    DELAY_500ms
        MOV     P0,#00H
        MOV     P1,#00H
        MOV     P3,#0FFH
        MOV     R1,#00H
        MOV     R2,#00H
        MOV     SCON,#051H
        MOV     PCON,#00H
        MOV     TMOD,#020H
        MOV     TH1,#0FDH
        SETB    TR1
        CLR     RI
        LJMP    CHK_RX
```

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

CHK_RX_INT0:    JB      P3.2,CHK_RX_INT1
                LCALL   DELAY_100ms
                JNB     P3.2,$
                CJNE    R2,#00H,INTERRUP1
                MOV     R1,#00H
                LJMP    CHK_RX

INTERRUP1:      DEC     R2
                MOV     P1,R2
                LCALL   DELAY_10ms
                CJNE    R2,#00H,XX
                MOV     P0,#00H
                MOV     R1,#00H
                MOV     SBUF,#0AAH
                JNB     TI,$
                CLR     TI
XX:             LJMP    CHK_RX

CHK_RX_INT1:    JB      P3.3,CHK_RX
                LCALL   DELAY_100ms
                JNB     P3.3,$
                CJNE    R2,#00H,INTER1
                MOV     R1,#00H
                LJMP    CHK_RX

INTER1:         DEC     R2
                MOV     P1,R2
                LCALL   DELAY_10ms
                CJNE    R2,#00H,CHK_RX
                MOV     P0,#00H
                MOV     R1,#00H
                MOV     SBUF,#0AAH
                JNB     TI,$
                CLR     TI
                LJMP    CHK_RX

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

```
CHK_RX_INT0_1:  JB      P3.2,CHK_RX_INT1_1
                LCALL   DELAY_100ms
                JNB     P3.2,$
                CJNE    R2,#00H,INTERRUP1_1
                MOV     R1,#00H
                LJMP    CHK_RX

INTERRUP1_1:    DEC     R2
                MOV     P1,R2
                LCALL   DELAY_10ms
                CJNE    R2,#00H,CHK_RX_1
                MOV     P0,#00H
                MOV     R1,#00H
                MOV     SBUF,#0AAH
                JNB     TI,$
                CLR     TI
                LJMP    CHK_RX_1

CHK_RX_INT1_1:  JB      P3.3,CHK_RX_1
                LCALL   DELAY_100ms
                JNB     P3.3,$
                CJNE    R2,#00H,INTER1_1
                MOV     R1,#00H
                LJMP    CHK_RX_1

INTER1_1:       DEC     R2
                MOV     P1,R2
                LCALL   DELAY_10ms
                CJNE    R2,#00H,CHK_RX_1
                MOV     P0,#00H
                MOV     R1,#00H
                MOV     SBUF,#0AAH
                JNB     TI,$
                CLR     TI
                LJMP    CHK_RX_1

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

```
;*****************************************************************************************************
*******
TEMP:           LJMP        CHK_RX_INT0
;*****************************************************************************************************
*******

CHK_RX:         JNB         RI,TEMP                 ;รอรับค่าจาก computer
                MOV         R1,SBUF
                CLR         RI

CHK_RX_1:       JNB         RI,CHK_RX_INT0_1        ;รับค่า step
                MOV         R2,SBUF
                CLR         RI

CHK_CASE:       CJNE        R1,#00H,CASE1           ;เช็กว่าเท่ากับหยุดมอเตอร์หรือ
เปล่า
                MOV         P0,#00H
                MOV         P1,R2
                LJMP        CHK_RX

CASE1:          CJNE        R1,#01H,CASE2           ;เปรียบเทียบค่าคำสั่งว่า หมุนจาก
ซ้ายไปขวา ตัวที่ 1
                MOV         P0,#01H
                MOV         P1,R2
                LCALL       DELAY_500MS
                LJMP        CHK_RX

CASE2:          CJNE        R1,#02H,CASE3           ;เปรียบเทียบค่าคำสั่งว่า หมุนจาก
ขวาไปซ้าย ตัวที่ 1
                MOV         P0,#02H
                MOV         P1,R2
                LCALL       DELAY_500MS
                LJMP        CHK_RX

CASE3:          CJNE        R1,#03H,CASE4           ;เปรียบเทียบค่าคำสั่งว่า หมุนจาก
บนลงล่าง ตัวที่ 2
                MOV         P0,#04H
                MOV         P1,R2
                LCALL       DELAY_500MS
                LJMP        CHK_RX
```

```
CASE4:      CJNE    R1,#04H,CASE5          ;เปรียบเทียบค่าคำสั่งว่า หมุนจาก
ล่างขึ้นบน ตัวที่ 2
            MOV     P0,#08H
            MOV     P1,R2
            LCALL   DELAY_500MS
            LJMP    CHK_RX

CASE5:      MOV     P0,#00H
            MOV     R1,#00H
            MOV     R2,#00H
            MOV     P1,R2
            LCALL   DELAY_500MS
            LJMP    CHK_RX

;-------------------------------------------------------------
;           DELAY TIME
;-------------------------------------------------------------
DELAY_1ms:      MOV   31H,#0E6H
DELAY_1ms_1:    NOP
                NOP
                DJNZ  31H,DELAY_1ms_1
                RET

DELAY_5ms:      MOV   36H,#05
DELAY_5ms_1:    LCALL DELAY_1ms
                DJNZ  36H,DELAY_5ms_1
                RET

DELAY_10ms:     MOV   35H,#010
DELAY_10ms_1:   LCALL DELAY_1ms
                DJNZ  35H,DELAY_10ms_1
                RET

DELAY_100ms:    MOV   32H,#100
DELAY_100ms_1:  LCALL DELAY_1ms
                DJNZ  32H,DELAY_100ms_1
                RET

DELAY_500ms:    MOV   33H,#05
DELAY_500ms_1:  LCALL DELAY_100ms
                DJNZ  33H,DELAY_500ms_1
```

```
                RET

DELAY_1s:       MOV     34H,#10
DELAY_1s_1:     LCALL   DELAY_100ms
                DJNZ    34H,DELAY_1s_1
                RET

                END
```

บรรณานุกรม

1. ธีรวัฒน์ ประกอบผล, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์", สมาคมส่งเสริมเทคโนโลยี (ไทยญี่ปุ่น)
2. รศ. สมยศ จุณณะปิยะ, "การประยุกต์การใช้งานไมโครคอนโทรลเลอร์", สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ
3. กนก กุสุมาลย์กูล, ไกรวุฒิ มั่นเสถียรสิน, "คู่มือการเขียนโปรแกรม Delphi 4", บริษัท ซัคเซส มีเดีย จำกัด กรุงเทพฯ