ฟังก์ชันเจเนอเรเตอร์ชนิดโปรแกรมได้

A PROGRAMMABLE FUNCTION GENERATOR DESIGN

โดย

นางสาวกฤติยา วัฒนศิริสุข

นางสาวธนัญญา ใจภักดีมั่น

นายวรินทร์ ธีระพงษ์พันธุ์

วัน เดือน ปี 14 ต.ค. 2540

เลขทะเบียน 037219

เลขเรียก T38318 ก 2538

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

037219

ฟังก์ชันเจเนอเรเตอร์ชนิดโปรแกรมได้

A PROGRAMMABLE FUNCTION GENERATOR DESIGN

โดย

นางสาวกฤติยา วัฒนศิริสุข 35104010

นางสาวธนัญญา ใจภักดีมั่น 35104170

นายวรินทร์ ธีระพงษ์พันธุ์ 35104369

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ฟังก์ชันเจเนอเรเตอร์ซนิดโปรแกรมได้

A PROGRAMMABLE FUNCTION GENERATOR DESIGN

ผู้จัดทำ

1. นางสาวกฤติยา วัฒนศิริสุข 35104010
2. นางสาวธนัญญา ใจภักดีมั่น 35104170
3. นายวรินทร์ ธีระพงษ์พันธุ์ 35104369

อาจารย์ที่ปรึกษา

(รศ.ดร.กอบชัย เดชหาญ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชันเจเนอเรเตอร์ชนิดโปรแกรมได้

A PROGRAMMABLE FUNCTION GENERATOR DESIGN

โดย นางสาวกฤติยา วัฒนศิริสุข 35104010

นางสาวธนัญญา ใจภักดีมั่น 35104170

นายวรินทร์ ธีระพงษ์พันธุ์ 35104369

อาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ

บทคัดย่อ

เครื่องสร้างสัญญาณมีความสำคัญมากในการใช้งานด้านทดสอบเครื่องมือวัดต่างๆ โครงงานนี้เสนอการออกแบบและสร้าง ฟังก์ชันเจเนอเรเตอร์ ที่โปรแกรมเลือกค่าสัญญาณ รูปซายน์ รูปสี่เหลี่ยม รูปสามเหลี่ยม และรูปฟันเลื่อยได้ โดยเลือกค่าความถี่จากย่าน 10Hz ถึง 100 kHz พร้อมกันนั้นก็เลือกขนาดของสัญญาณได้ตั้งแต่ 1 V_{p-p} ถึง 10 V_{p-p} ในการออกแบบนี้ ต้องการสัญญาณซายน์ที่มีค่าผิดเพี้ยนต่ำมากๆ เพื่อนำมาใช้งานได้ดี

ABSTRACT

The function generator is an important equipment for testing, experiment. This project proposes a design and consturction of programmable function generator. It can produce sine wave, square wave, triangle wave and saw tooth wave. The selecting frequency range is 10 Hz to 100 kHz and the ampletude variation is between 1 V_{p-p} to 10 V_{p-p}. This project would like to have very low distortion.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในงานด้านอิเลคทรอนิคส์ สัญญาณนับว่ามีบทบาทสำคัญอย่างมาก การนำสัญญาณที่มีความถูกต้อง และแม่นยำไปใช้ในงานต่างๆ ย่อมเป็นที่ต้องการของทุกคน สมมติว่าเราต้องการผลิตสัญญาณความถี่ 1.536 kHz โดยใช้เครื่องกำเนิดสัญญาณแบบอนาล็อก เวลาที่จะใช้งานต้องบิดปุ่มปรับความถี่ ซึ่งอาจไม่แน่ใจได้ว่าความถี่ที่ปรับนั้นเป็นความถี่ที่ต้องการ และแน่นอนเที่ยงตรง เมื่อประสบปัญหาเช่นนี้ คงต้องการเครื่องกำเนิดสัญญาณที่สามารถกำหนด ค่าความถี่ และค่าแอมปลิจูดที่คงที่แน่นอนได้

โครงงานนี้ ได้เสนอเรื่องการกำเนิดคลื่นสัญญาณชนิดต่างๆ โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม คือสามารถกำหนดค่าความถี่ และแอมปลิจูดได้ โดยโปรแกรมจาก ANT-32 ผ่านทางคีย์บอร์ด ทำให้มีความสะดวกในการใช้งาน และให้ความเที่ยงตรงสูง มีความสามารถในการกำเนิดคลื่นสัญญาณได้ 4 รูปแบบ

1. สัญญาณรูปคลื่นซายน์ (sine wave)
2. สัญญาณรูปคลื่นสี่เหลี่ยม (square wave)
3. สัญญาณรูปคลื่นสามเหลี่ยม (triangle wave)
4. สัญญาณรูปคลื่นฟันเลื่อย (sawtooth wave)

โดยมีความสามารถกำเนิดสัญญาณได้ที่ความถี่ตั้งแต่ 10Hz ถึง 100kHz และแอมปลิจูดตั้งแต่ $1V_{p-p}$ ถึง $10V_{p-p}$ มีการแสดงผลการเลือกชนิดของสัญญาณ, ความถี่ และแอมปลิจูด ผ่านออกทางจอแอลซีดี 16 หลัก 2แถว

เอาท์พุทที่ได้ มี 2 ช่องสัญญาณ

1. เอาท์พุตอิมพีแดนซ์ 600 โอห์ม :- ใช้ทดสอบสัญญาณในย่านความถี่เสียง และทดสอบสัญญาณในระบบโทรศัพท์

2. เอาท์พุตอิมพีแดนซ์ 50 โอห์ม :- ใช้กำเนิดสัญญาณรูปต่างๆ สำหรับอุปกรณ์ในห้องทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎี และหลักการ

2.1 การกำเนิดสัญญาณ

เครื่องกำเนิดสัญญาณหลายแบบ (Function Generator) จะเป็นเครื่องกำเนิดความถี่ที่มีย่านกำเนิดกว้าง ตั้งแต่ความถี่เสียงไปจนถึงความถี่วิทยุ สามารถผลิตรูปคลื่นขึ้นมาได้หลายอย่าง คือ คลื่นซายน์ คลื่นสี่เหลี่ยมจัตุรัส คลื่นสามเหลี่ยม คลื่นลาดเอียง และคลื่นพัลส์ โดยทั่วไปเจเนอเรเตอร์จะกำเนิดสัญญาณขึ้นมา 4ชนิด คือ

1 รูปคลื่นซายน์ (sine wave)

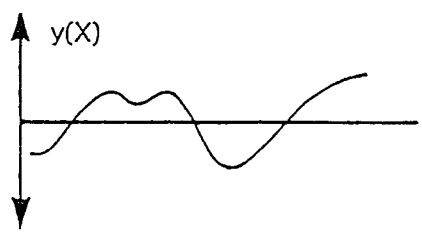
2 รูปคลื่นสี่เหลี่ยม (square wave)

3 รูปคลื่นสามเหลี่ยม (triangle wave)

4 รูปคลื่นฟันเลื่อย (sawtooth)

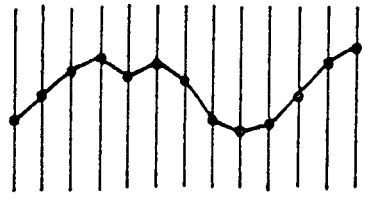
2.1.1 หลักการสร้างสัญญาณ

เริ่มจากการนำค่าข้อมูลเก็บลงใน EPROM โดยเขียนโปรแกรมเพื่อเก็บค่าข้อมูลภายใน 1 คาบของสัญญาณรูปต่างๆ อาศัยการแซมปลิ้งสัญญาณรูปคลื่นที่มีความต่อเนื่อง (analog) ให้เป็นสัญญาณ แบบไม่ต่อเนื่อง (discrete) เพื่อเก็บระดับของสัญญาณเป็นค่าแรงดันในแต่ละครั้งที่แซมปลิ้งได้ โดยระยะห่างระหว่างการแซมปลิ้งแต่ละครั้งคงที่ ดังอธิบายตามกระบวนการ Digital Signal Processing (DSP) ในรูปที่ 2.1 แสดงสัญญาณที่เป็นฟังก์ชั่นต่อเนื่อง y(X) แล้วนำมาทำการแซมปลิ้งตามแนวนอน (horizontal axis sampling) ตามรูป 2.2 เพื่อเก็บค่า Y(I) ซึ่ง I = 1, 2, 3, ... เป็นจำนวนครั้งที่เป็นจำนวนเต็มในแต่ละครั้งที่แซมปลิ้ง (index integer) ผลที่ได้ออกมาดังแสดงในรูป 2.3

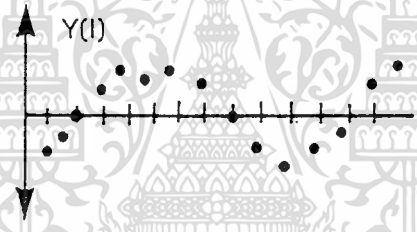


รูปที่ 2.1 สัญญาณที่มีความต่อเนื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 การแซมปลิ้งตามแนวนอน

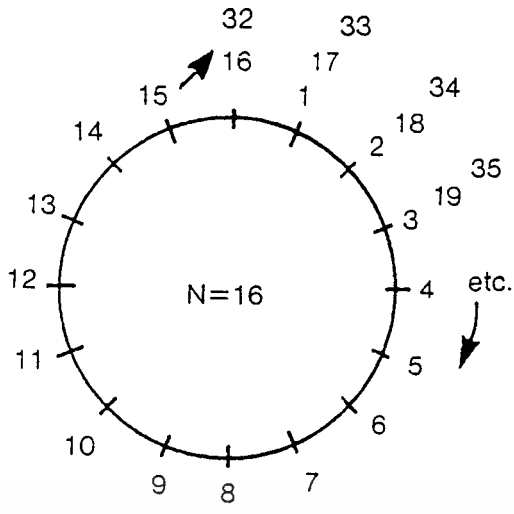


รูปที่ 2.3 สัญญาณไม่ต่อเนื่องที่ได้

ด้วยวิธีนี้เราสามารถเก็บค่ารูปคลื่นต่อเนื่องชนิดใดก็ได้ และข้อมูลในแกนตั้ง และแกนนอนจะเป็นหน่วยใดก็ได้ เช่น แรงดันต่อเวลา, แอมปลิจูดต่อความถี่ หรือค่าออฟเซ็ตต่ออุณหภูมิ ในที่นี้เราเลือกเก็บเป็นค่าแรงดันในแกนตั้ง จากการแซมปลิ้ง ณ เวลาต่างๆในแกนนอน เช่น สัญญาณซายน์อาศัยหลักการของวงกลมหนึ่งหน่วยในการเก็บค่าสัญญาณ เมื่อ R คือรัศมีหมุนไปตามเส้นรอบวงเป็นมุม P ดังแสดงในรูปที่ 2.8 เมื่อลากเส้นจากปลายรัศมี R ขนานกับแกน X ไปตัดกับแกน Y ค่า S ที่ได้ในแกน Y ซึ่งสัมพันธ์กับรัศมี R และมุม P คือ $S = R\sin P$

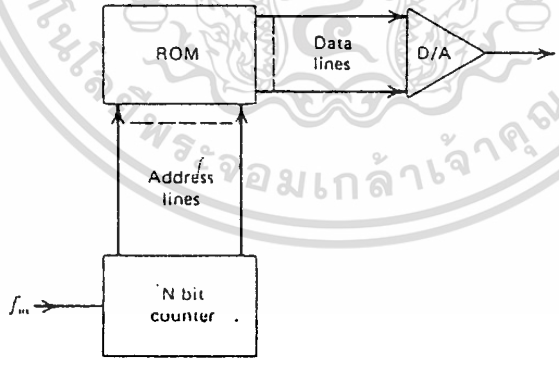
ถ้าหากการหมุนของรัศมี R เป็นสเต็ปแบบไม่ต่อเนื่องตามจังหวะการแซมปลิ้ง เป็นจำนวน N ครั้ง ดังในรูป 2.4 จะได้ค่า S จำนวน N ค่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.4 การแซมปลิ้งสัญญาณภายในวงกลมหนึ่งรอบ

สำหรับการเข้ารหัสข้อมูลเพื่อเก็บในแต่ละแอดเดรสของ ROM จะมีความละเอียดมาก หรือน้อยขึ้นอยู่กับจำนวนบิต (length code word) เช่น เก็บเป็นข้อมูลที่มีความละเอียด 8บิต ก็จะได้จำนวนข้อมูลที่สามารถเข้ารหัสได้ 256 ค่า (2^8) เมื่อแซมปลิ้งได้ค่าระดับสัญญาณใกล้เคียงค่าใด ก็จะเก็บค่านั้นไว้ใน ROM เพื่อรอการเรียกออกมาเป็นเอาท์พุตของสัญญาณ ซึ่งภายใน 1 คาบจะเก็บค่าสัญญาณได้ละเอียด มากน้อยเพียงใด ขึ้นอยู่กับขนาดของ ROM ซึ่งค่าแอดเดรสมีได้ตั้งแต่ 2-N บิต โดยจะใช้วงจรนับเป็นตัวกำหนดค่าแอดเดรส ในการเรียกข้อมูลออกมา



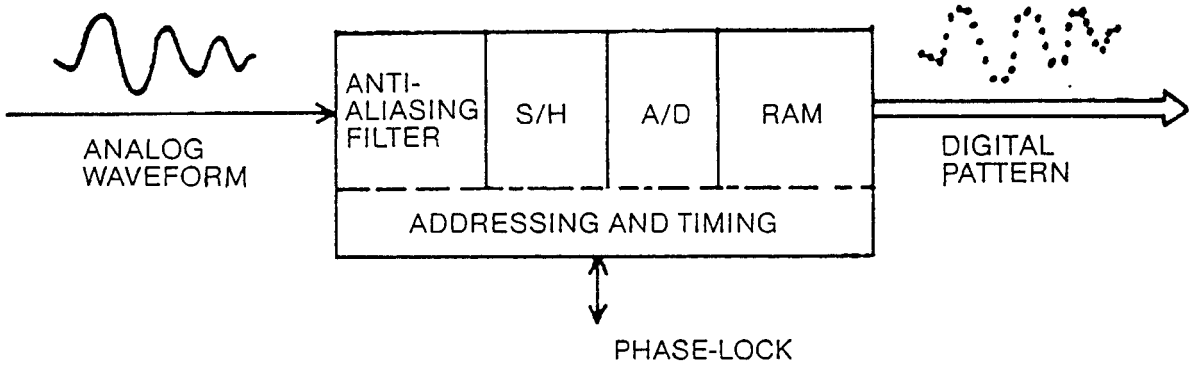
รูปที่ 2.5 บล็อกไดอะแกรมของวงจรกำเนิดสัญญาณ

จากรูปที่ 2.5 เป็นวงจรกำเนิดสัญญาณเอาท์พุตที่มีความถี่ f_0 ซึ่งในทางปฏิบัติการออกแบบวงจรจะขึ้นอยู่กับความถี่สัญญาณนาฬิกา (f_{in}) และค่าความละเอียดของสัญญาณ (N) เป็นไปตามสมการ

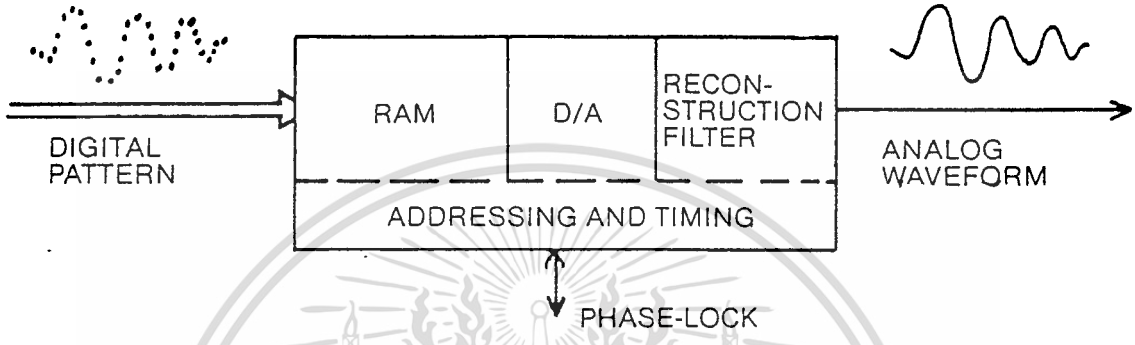
$$f_{in} = 2^N f_0$$

ค่าความผิดเพี้ยนของสัญญาณเอาท์พุตที่ได้ สามารถลดลงได้โดยใช้วงจรกรองสัญญาณความถี่ต่ำผ่าน (LPF) เพื่อกำจัดความถี่สูงซึ่งมาจากการเปลี่ยนระดับโวลท์เตจระหว่างแอดเดรสที่ติดกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

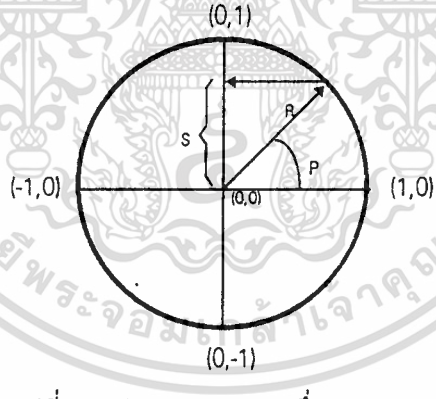


รูปที่ 2.6 แสดงบล็อกไดอะแกรมในการแปลงรูปสัญญาณอนาล็อกเป็นดิจิตอลเพื่อนำไปเก็บใน RAM



รูปที่ 2.7 บล็อกแสดงการเรียกสัญญาณดิจิตอลจาก RAM มาแปลงเป็นเอาท์พุตในรูปสัญญาณอนาล็อก

2.1.2 รูปคลื่นซายน์ (Sine Wave)



รูปที่ 2.8 ลักษณะวงกลมหนึ่งหน่วย

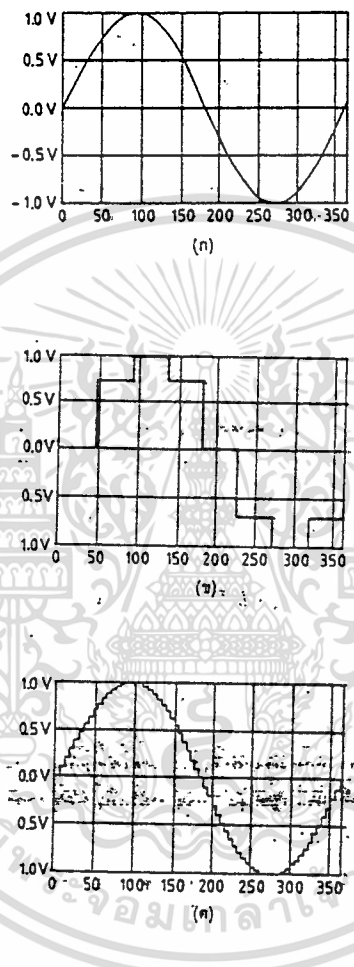
จากรูป 2.8 วงกลมหนึ่งหน่วยมีรัศมี R หมุนไปตามเส้นรอบวงของวงกลมจะทำให้เกิดมุม P ซึ่งผลจากการหมุนนี้จะทำให้เกิดการเปลี่ยนแปลงของแอมปลิจูด ซึ่งถ้าหากลากเส้นจากปลายรัศมี R ไปตามแนวนอนจนตัดกับแกน Y ก็จะได้ความยาว S ที่ เปลี่ยนแปลงตามการหมุนของรัศมี R มีค่าอยู่ระหว่าง 1 และ -1 และมุม P ก็จะเกิดการเปลี่ยนแปลงจาก 0 ถึง 360 องศา ความยาวของ S ที่เปลี่ยนแปลงนี้จะสัมพันธ์กับมุม P ในรูปของฟังก์ชันซายน์ [S=Rsin (P)] แสดงดังรูป 2.9 ก

ในหนึ่งรอบของวงกลมถ้าหมุนรัศมี R ไปเป็นสเต็ปไม่ต่อเนื่อง จำนวน 8 ครั้ง ก็จะได้ค่าของ S มา 8 ค่า ดังในรูป 2.9 ข ถ้าหากกำหนดให้จำนวนสเต็ปในการหมุนรอบวงกลมหนึ่งรอบให้มีจำนวนเพิ่มมากขึ้น รูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลื่นสัญญาณซายน์ที่ได้ก็จะมีความละเอียดมากขึ้นดังรูป 2.9 ค ได้กำหนดจำนวนสเต็ปในการหมุนรอบวงกลมหนึ่งรอบไว้เท่ากับ 64 จะสังเกตเห็นความเป็นซายน์มากขึ้น

นั่นคือ ในหนึ่งช่วงคาบเวลา ค่าของสัญญาณจะอยู่ในแอดเดรสช่วง 000h~7FFh โดยมีค่าตั้งแต่ 00h~FFh ซึ่งคำนวณได้จากสมการ $127.5 \sin(2\pi P/2048 - \pi/2)$ โดยที่ P มีค่าระหว่าง 0~2047 (000h~7FFh)



รูปที่ 2.9 แสดงรูปคลื่นสัญญาณซายน์

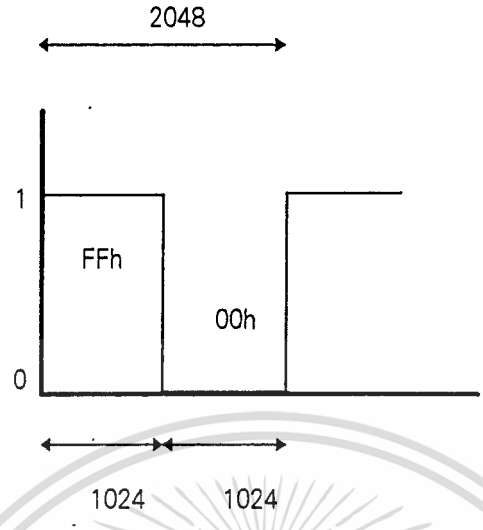
ก) ที่เกิดจากการหมุนของรัศมี R

ข) เมื่อถูกแบ่งด้วยวิธีทางดิจิตอลเป็น 8 สเต็ป

ค) เมื่อถูกแบ่งเป็น 64 สเต็ป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

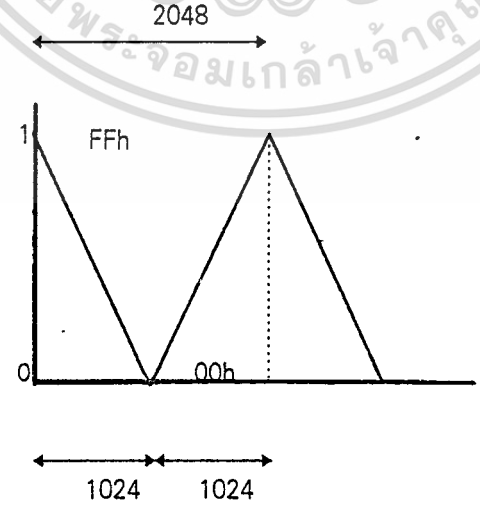
2.1.3 รูปคลื่นสี่เหลี่ยม (Square Wave)



รูปที่ 2.10 สัญญาณรูปคลื่นสี่เหลี่ยม

จากรูป 2.10 ในหนึ่งคาบเวลาสัญญาณจะมีเพียง 2 ค่า คือ ช่วงครึ่งแรกของคาบเวลาจะมีค่าเป็น "1" และครึ่งหลังของคาบเวลาจะมีค่าเป็น "0" ดังนั้นค่าที่เก็บในแอดเดรสช่วง 000h~3FFh จะมีค่าเป็น FFh และช่วง 400h~7FFh จะมีค่าเป็น 00h

2.1.4 รูปคลื่นสามเหลี่ยม (Triangle Wave)

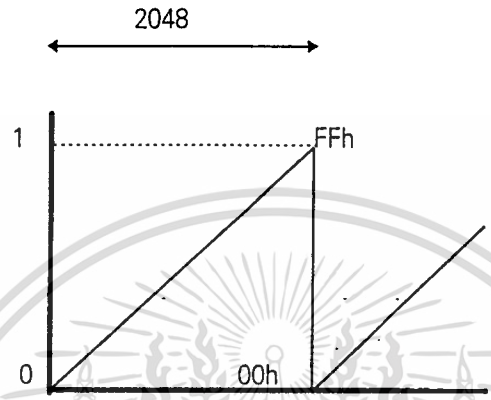


รูปที่ 2.11 สัญญาณรูปคลื่นสามเหลี่ยม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูป 2.11 ในช่วงครึ่งคาบเวลาแรกสัญญาณจะมีค่าลดลงอย่างเป็นลิเนียร์ จาก 1 ไปจนถึง 0 และช่วงครึ่งคาบเวลาหลังสัญญาณจะมีค่าเพิ่มขึ้นอย่างเป็นลิเนียร์จาก 0 ไปจนถึง 1 ซึ่งจะได้ว่าในช่วงครึ่งคาบเวลาแรกค่าของสัญญาณจะอยู่ในแอดเดรสช่วง 000h~3FFh และครึ่งคาบเวลาหลังค่าของสัญญาณจะอยู่ในแอดเดรสช่วง 400h~7FFh โดยมีค่าสูงสุดเป็น FFh และค่าต่ำสุดเป็น 00h

2.1.5 รูปคลื่นฟันเลื่อย (Sawtooth Wave)



รูปที่ 2.12 สัญญาณรูปคลื่นฟันเลื่อย

จากรูป 2.12 ในหนึ่งคาบเวลาสัญญาณจะมีค่าเพิ่มขึ้นจาก 0 ไปจนถึง 1 อย่างเป็นลิเนียร์ ดังนั้นในแอดเดรสที่ตำแหน่ง 000h จะมีค่า 00h และค่าจะเพิ่มขึ้นเรื่อยๆไปจนถึงตำแหน่ง 7FFh ซึ่งมีค่า FFh

2.2 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

2.2.1 โครงสร้าง และพื้นฐาน

ไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์ มีลักษณะที่แตกต่างกัน คือ ไมโครโปรเซสเซอร์จะประกอบด้วยหน่วยควบคุม (Control Unit) และหน่วยคำนวณทางคณิตศาสตร์ และลอจิก (ALU) ส่วนไมโครคอนโทรลเลอร์จะประกอบด้วยหน่วยควบคุม, หน่วยควบคุมทางคณิตศาสตร์และลอจิก, หน่วยความจำ (memory), อินพุท/เอาท์พุทพอร์ต (I/O Port) ซึ่งอาจกล่าวได้ว่า ไมโครโปรเซสเซอร์ก็คือส่วนหนึ่งของไมโครคอนโทรลเลอร์นั่นเอง แต่เมื่อเปรียบเทียบกันแล้ว หน่วยการทำงานภายในไมโครคอนโทรลเลอร์นั้นสามารถใช้งานได้อย่างค่อนข้างจำกัดกว่า

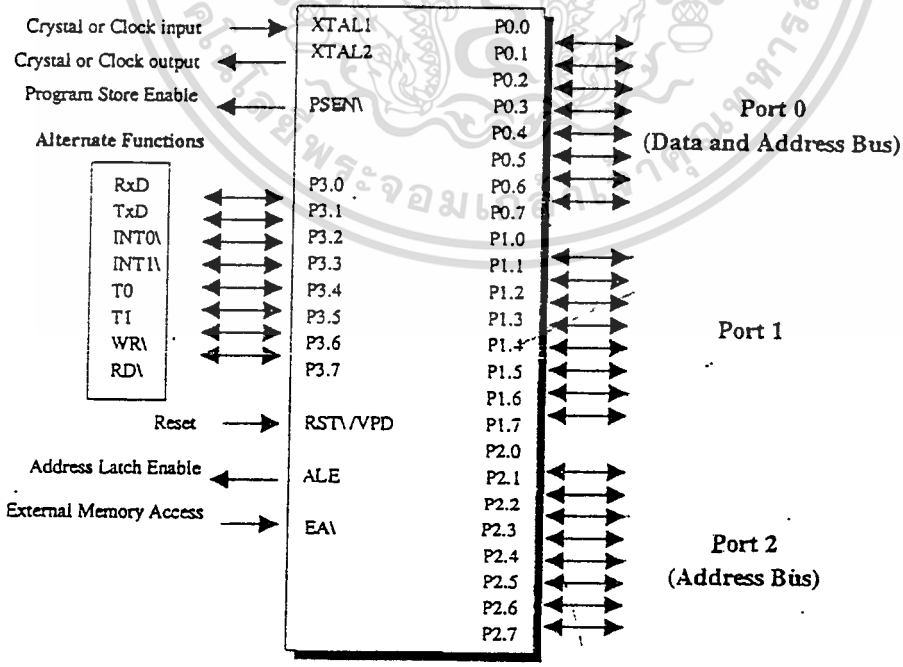
ลักษณะงานที่เหมาะสมกับการนำไมโครคอนโทรลเลอร์ไปใช้งาน มักเป็นงานประยุกต์ที่เกี่ยวข้องกับการควบคุม หรือการจัดการสัญญาณอินพุท/เอาท์พุทของวงจรดิจิตอล และวงจรอิเลคทรอนิคส์ต่างๆ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วย ไมโครคอนโทรลเลอร์หลายรุ่น ซึ่งมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาด หรือจำนวน หน่วยทำงานภายในที่แตกต่างกันออกไป ดังแสดงให้เห็นในตารางที่ 2.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่2.1 ตารางแสดงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของบริษัทอินเทล

EMBEDDED CONTROLLERS										
Feature	**8051AH**	**8031AH**	**8751H**	**80C51BH**	**80C31BH**	**87C51**	**8052AH**	**8032AH**	**8752**	**8044H**
Program Memory (Bytes)	4K	-	4K	4K	-	4K	8K	-	8K	4K
RAM Memory (Bytes)	128	128	128	128	128	128	256	256	256	192
Program Memory Expansion (Off Chip) (Bytes)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
Data Memory Expansion (Off Chip) (Bytes)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
Max Clock Frequency (MHz)	12	12	12	16	16	16	16	12	12	12
Typical Instruction Time (uS)	1	1	1	0.75	0.75	0.75	1	1	1	1
16-Bit Timer / Counters	2	2	2	2	2	2	3	3	3	2
Serial Communications	Synchronous Mode, Asynchronous Mode, 9 or 10 - Bit Programmable									HDLC/SDLC
No. of I/O Lines	32	16	32	32	16	32	32	16	32	32
Interrupt Sources	5	5	5	5	5	5	6	6	6	5
(Two Priority Levels) Power Requirements 125	125	250	24	24	29	175	175	175	200	
(ICC Max. mA) Programmable Power Modes	-	-	-	4.0 mA	4.0 mA	4.0 mA	-	-	-	-
Idle Power Down				50 uA	51 uA	52 uA				30 mA

ปกติไมโครคอนโทรลเลอร์ตระกูลนี้มักมีรูปร่างของไอซี ดังรูปที่ 2.13



รูปที่ 2.13 การกำหนดหน้าที่ขาสัญญาณของไอซี 8051

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 หน่วยความจำโปรแกรมของ MCS-51

หน่วยความจำโปรแกรมเป็นบริเวณหน่วยความจำสำหรับเก็บข้อมูล และคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบ ข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหาย โครงสร้างของหน่วยความจำโปรแกรมมีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไอซีหน่วยความจำประเภทต่างๆ เช่น หน่วยความจำแบบ ROM(Read Only Memory) หรือ EPROM(Erasable Programmable Read Only Memory) เป็นต้น

MCS-51 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (ซึ่งเป็นหน่วยความจำ ROM หรือ EPROM) ที่อยู่ภายในตัวไอซีไมโครคอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ

1. หน่วยความจำโปรแกรมภายใน

ไมโครคอนโทรลเลอร์เบอร์ต่างๆที่อยู่ในตระกูล MCS-51 นี้มีขนาดหน่วยความจำโปรแกรมภายในแตกต่างกันออกไป เพื่อความเหมาะสมกับการนำไปใช้งานในลักษณะต่างๆกัน

2. หน่วยความจำโปรแกรมภายนอก

เป็นการใช้หน่วยความจำ EPROM (หรือ ROM) เชื่อมต่อเข้ากับระบบของ MCS-51 ซึ่งอาจเกิดจากการที่หน่วยความจำภายในมีขนาดความจุไม่เพียงพอกับการเก็บโปรแกรม หรือเกิดจากการที่ตัวไมโครคอนโทรลเลอร์เองไม่มีหน่วยความจำภายใน (เช่น 8031 และ8032) ไมโครคอนโทรลเลอร์เบอร์ต่างๆของตระกูล MCS-51 นี้ สามารถขยายให้ใช้งานหน่วยความจำภายนอกได้ทั้งสิ้น ในกรณีที่มีหน่วยความจำภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งหน่วยความจำภายใน และภายนอกนั้น จะต้องทำการพิจารณาระดับลอจิกของสัญญาณ EA ในขณะนั้นด้วย

การเชื่อมต่อกับหน่วยความจำ EPROM จะเกี่ยวข้องกับสัญญาณ 4 กลุ่มด้วยกัน คือ

- ขาสัญญาณสำหรับจ่ายไฟให้กับไอซี (Power)
- บัสแอดเดรส
- บัสข้อมูล
- บัสสัญญาณควบคุมการส่งออกข้อมูล

สำหรับการพิจารณาจำนวนเส้นของสัญญาณแอดเดรสที่ใช้กับ EPROM ตัวหนึ่งๆนั้น ตัวอย่างเช่น EPROM ขนาด 4096x8 จะนำตัวเลขค่าแรกซึ่งบอกถึงความจุในการเก็บข้อมูล มาคำนวณ ดังนี้

$2^x = 4096$

$x = \log(4096)/\log(2)$

$x = 12$

ดังนั้นจำนวนเส้นสัญญาณแอดเดรสที่ใช้กับ EPROM จึงมีทั้งหมด 12 เส้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณที่มาเชื่อมต่อเข้ากับกลุ่มสัญญาณแอดเดรสของ EPROM มักเป็นกลุ่มของสัญญาณเอาท์พุทที่มาจากพอร์ตของไมโครคอนโทรลเลอร์(โดยปกติจะเป็นบัสแอดเดรสของระบบ) เพื่อทำการกำหนดตำแหน่งข้อมูลที่ต้องการ ข้อมูลภายในบัสแอดเดรสนี้จะถูกนำมาทำการถอดรหัสเพื่อหาตำแหน่งที่ต้องการภายใน EPROM ส่วนบัสข้อมูลมีหน้าที่สำหรับการเอาท์พุทข้อมูลที่เก็บอยู่ในตำแหน่งภายในหน่วยความจำ ซึ่งถูกระบุมาจากบัสแอดเดรส

สำหรับสัญญาณควบคุมการส่งออก ได้แก่

- สัญญาณ CS (Chip Select) :- มีหน้าที่เลือกให้ EPROM ทำงาน

- สัญญาณ OE (Output Enable) :- จะทำให้มีการไปนำข้อมูลภายในตำแหน่งที่ถูกระบุตามข้อมูลของบัสแอดเดรส และทำการส่งออกมายังบัสข้อมูลของ EPROM สัญญาณควบคุมนี้มักต่อเข้ากับขาสัญญาณ RD ของไมโครคอนโทรลเลอร์

สัญญาณต่างๆของ MCS-51 ที่นำมาใช้ในการติดต่อกับหน่วยความจำภายนอก

- สัญญาณ EA(External Access) :- ใช้ในการเลือกว่าจะอ่านข้อมูลจากหน่วยความจำภายนอก หรือภายใน

- พอร์ต0 (P0.0-P0.7) :- ในช่วงเวลาเริ่มต้นในการติดต่อกับหน่วยความจำภายนอกพอร์ต0 จะเป็นค่าของแอดเดรสไบต์ต่ำ และเวลาต่อมาจึงจะเป็นบัสข้อมูล การส่งค่าแอดเดรสไบต์ต่ำจะอยู่ในช่วงเวลาขอบขาลงของสัญญาณ ALE และจะคงอยู่จนสัญญาณ PSEN เปลี่ยนเป็นลอจิกต่ำ

- สัญญาณ ALE :- ใช้ทำให้ไอซีแลตช์ ค้างค่าสัญญาณแอดเดรสไว้

- สัญญาณ PSEN :- ใช้ในการเลือกให้ EPROM ทำงาน และอ่านค่าข้อมูลกลับมา

- พอร์ต2 (P2.0-P2.3) :- เป็นข้อมูลแอดเดรสไบต์สูงของหน่วยความจำ ซึ่งค่าแอดเดรสที่พอร์ตนี้จะค้างอยู่ตลอดช่วงรอบเวลาของการติดต่อกับหน่วยความจำโปรแกรม

2.2.3 หน่วยความจำข้อมูลของ MCS-51

หน่วยความจำข้อมูลมีหน้าที่สำหรับเก็บข้อมูล หรือตัวแปรที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว ซึ่งหน่วยความจำข้อมูลจัดเป็นหน่วยความจำ RAM แบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟให้กับระบบ ก็จะทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำสูญหายไป พื้นที่หน่วยความจำข้อมูลของ MCS-51 มีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกได้เป็น 2 ลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำ คือ หน่วยความจำภายใน ซึ่งเป็น RAM ที่อยู่ภายในตัวไอซีไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลภายนอก ซึ่งเป็นการใช้หน่วยความจำ RAM มาเพิ่มเข้าไปในวงจร

1. หน่วยความจำข้อมูลภายใน

มีจำนวนทั้งหมด 256 ไบต์ โดยจำแนกได้เป็น 2 ลักษณะคือ พื้นที่เฉพาะสำหรับซีพียูใช้งาน (เรียกรีจิสเตอร์) และพื้นที่สำหรับโปรแกรมใช้งานที่ผู้ใช้สร้างขึ้นมา

1.1) หน่วยความจำ 128 ไบต์แรก

มีตำแหน่งแอดเดรสอยู่ในช่วง 00H-7FH ซึ่งแยกเป็น 3 ส่วนการใช้งาน

เอกสารนี้ได้รับการคุ้มครองลิขสิทธิ์ ห้ามมิให้ทำซ้ำ ดัดแปลง หรือนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- แอดเดรส 00H-1FH : จำนวน 32 ไบต์ แบ่งเป็น 4กลุ่ม(เรียก แบงค์) แต่ละกลุ่มใช้งานในฐานะของรีจิสเตอร์ใช้งานทั่วไป เรียก รีจิสเตอร์ R0-R7 การสวิทช์เลือกแต่ละกลุ่มกำหนดจากค่าบิตภายในรีจิสเตอร์ PSW แต่ปกติมักใช้งานในแบงก์0เท่านั้น แบงก์อื่นๆที่เหลือสามารถใช้งานในลักษณะหน่วยความจำข้อมูลภายในปกติ ด้วยการอ้างถึงแอดเดรสนั้นๆโดยตรง

- แอดเดรส 20H-2FH : จำนวน 16ไบต์ เป็นพื้นที่ที่ผู้ใช้สามารถอ้างถึงหน่วยความจำได้ทั้งลักษณะของ ไบต์ข้อมูล เช่นปกติ หรืออาจเป็น บิตข้อมูล ได้โดยตรง ซึ่งถือว่าเป็นการใช้งานอย่างมีประสิทธิภาพมาก

- แอดเดรส 30H-7FH : เป็นบริเวณที่สามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถอ้างถึงได้ในลักษณะของไบต์ข้อมูลตามปกติเท่านั้น

1.2) หน่วยความจำ 128ไบต์ถัดมา

ตั้งแต่แอดเดรส 80H-FFH เป็นหน่วยความจำที่มีการใช้งานเฉพาะจาก MCS-51 เท่านั้น โดยจะนำมาใช้เป็นตำแหน่งของรีจิสเตอร์หน้าที่พิเศษ (Special Function Register : SFR) จำนวน 20 ตำแหน่ง SFR เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่ และการทำงานของอุปกรณ์ หรือพอร์ตของ MCS-51

2. หน่วยความจำข้อมูลภายนอก

การนำหน่วยความจำข้อมูลภายนอกมาใช้กรณีที่มีความต้องการหน่วยความจำสำหรับเก็บข้อมูลชั่วคราว หรือตัวแปรของโปรแกรมมีมากเกินกว่าขนาดหน่วยความจำข้อมูลภายใน(ซึ่งมีขนาดเพียง 128 หรือ 256 ไบต์) จะเก็บได้ ในการนำหน่วยความจำข้อมูลภายนอกมาใช้จะมีผลทำให้พอร์ตอินพุท/เอาท์พุทข้อมูลของ MCS-51 ถูกนำไปใช้เพื่อติดต่อกับหน่วยความจำเหล่านี้แทน ดังนั้นต้องมีการใช้วงจรประกอบอื่นๆเข้ามาชดเชย

ลักษณะของหน่วยความจำ ROM เหมือน EPROM มาก และยังใช้การติดต่อที่เหมือนกันด้วย คือ มีการใช้ บัสแอดเดรส, บัสข้อมูล, กลุ่มสัญญาณควบคุม แต่มีการเพิ่มสัญญาณ WR ขึ้นมา เพื่อใช้ในการนำข้อมูลจากบัสข้อมูลเก็บลงไปยังหน่วยความจำตามตำแหน่งที่ระบุแอดเดรสมา

ในการเชื่อมต่อ MCS-51 เข้ากับ RAM จะใช้วิธีการเดียวกับการเชื่อมต่อกับ EPROM โดยมีการใช้ไอซีแลตช์ในการค้างค่าแอดเดรสให้กับอินพุตของหน่วยความจำ RAM

MCS-51 มีการติดต่อกับหน่วยความจำข้อมูลทั้งภายในไอซี และภายนอก จึงได้มีการแยกชุดคำสั่งที่ติดต่อกับหน่วยความจำทั้งสอง ดังแสดงในตารางที่ 2.2 และตารางที่ 2.3

ตารางที่ 2.2 ชุดคำสั่งในการโอนย้ายข้อมูลระหว่างหน่วยความจำข้อมูลภายใน

รูปแบบคำสั่ง	การทำงาน	เวลาการประมวลผลคำสั่ง (machine cycle)
MOV A, <src>.	A = ,src>	1
MOV <dest>, A	<dest> = A	1
MOV <dest>, <src>	<dest> = <src>	2
MOV DPTR, #data16	DPTR = ค่าคงที่ 16 บิต	2
PUSH <src>	INC SP; MOV "@SP", <src>	2
POP <dest>	MOV <dest>, "@SP"; DEC SP	2
XCH A,<byte>	ACC and <byte> exchange data	1
XCHD A, @Ri	ACC and @Ri exchange low nibbles	1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.3 ชุดคำสั่งในการโอนย้ายข้อมูลระหว่างหน่วยความจำข้อมูลภายนอก

จำนวนบิตแอดเดรส	รูปแบบคำสั่ง	การทำงาน	เวลาการประมวลผลคำสั่ง (machine cycle)
8 บิต	MOVX A,@Ri	Read External RAM @Ri	2
8 บิต	MOVX @Ri, A	Write External RAM @Ri	2
16 บิต	MOVX A,@DPTR	Read External RAM @DPTR	2
16 บิต	MOVX @DPTR,A	Write External RAM @DPTR	2

กรณีคำสั่งติดต่อกับหน่วยความจำข้อมูลภายนอกนั้น ต้องทำการอ้างแอดเดรสโดยอ้อมผ่านทางรีจิสเตอร์ R0, R1, DPTR ซึ่งทำหน้าที่เก็บแอดเดรสหน่วยความจำที่ต้องการไว้แทน โดย R0, R1 จะทำได้ในช่วงแอดเดรสไม่เกิน 256 ไบต์ เนื่องจากเป็นรีจิสเตอร์ 8 บิต หากใช้ผ่านรีจิสเตอร์ DPTR จะทำได้ในขอบเขตที่กว้างมากเต็มพื้นที่ที่ MCS-51 สามารถอ้างได้ สำหรับตัวอักษร X ที่เพิ่มเติมท้ายคำสั่งนั้นแสดงให้ทราบว่าต้องการจะติดต่อกับหน่วยความจำข้อมูลภายนอกเท่านั้น

2.2.4 พอร์ตอินพุต/เอาท์พุตของ MCS-51

พอร์ตหมายถึงแอดเดรสหนึ่งที่ได้รับการกำหนดไว้เพื่อการโอนย้ายข้อมูลระหว่างไมโครคอนโทรลเลอร์ กับอุปกรณ์ภายนอก การนำเข้าข้อมูลจากอุปกรณ์ภายนอก เรียก พอร์ตอินพุต ส่วนการส่งออกข้อมูล เรียก พอร์ตเอาท์พุต ซึ่งสามารถแยกประเภทพอร์ตได้เป็น 2 ลักษณะ คือ พอร์ตแบบขนาน ซึ่งทำการส่งจำนวนบิตข้อมูลทั้งหมดออกมา หรือเข้าไปพร้อมกันในคราวเดียว และพอร์ตแบบอนุกรม ซึ่งทำการโอนย้ายข้อมูลทีละบิตๆจนครบจำนวน

MCS-51 จะหลักการ Memory mapped system คือ การอ้างถึงหน่วยความจำ เนื่องจากรีจิสเตอร์หรืออุปกรณ์ต่างๆภายในระบบจะเป็นการติดต่อกับหน่วยความจำตำแหน่งหนึ่งเท่านั้น ดังนั้นในการนำเข้าหรือส่งออกข้อมูลกับพอร์ต จึงใช้คำสั่งการอ่านค่าจากหน่วยความจำ หรือคำสั่งการเขียนค่าข้อมูลไปยังตำแหน่งหน่วยความจำแทน ดังนั้นชุดคำสั่งของ MCS-51 จะไม่มีคำสั่งที่เกี่ยวข้องกับการทำงานพอร์ต เช่น คำสั่ง IN (นำเข้าข้อมูลจากพอร์ต) หรือ OUT (ส่งข้อมูลออกจากพอร์ต)

MCS-51 ยังมีชุดคำสั่งที่จัดการข้อมูลแบบบิตได้โดยตรง ดังนั้นสามารถใช้คำสั่งเหล่านี้จัดการพอร์ตอินพุต/เอาท์พุตแบบเส้นสัญญาณเดี่ยวได้ โดยใช้คำสั่ง SETB เพื่อกำหนดค่าเป็น1 หรือ CLR เพื่อทำให้บิตมีค่าเป็น0 ส่วนการตรวจสอบบิตของแต่ละพอร์ตอินพุต/เอาท์พุตสามารถใช้ชุดคำสั่งในการทดสอบได้โดยตรง ไม่ต้องมีการย้ายค่าของพอร์ตไปยังรีจิสเตอร์ก่อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4 คำสั่งการส่งข้อมูลออกทางพอร์ต

คำสั่ง	ความหมาย
MOV P0, A	นำค่าภายในรีจิสเตอร์ A ส่งออกไปยังพอร์ต 0
MOV P1, #00000101B	ส่งค่าข้อมูลออกไปยังพอร์ต 1 โดยตรง
MOV P1, #05H	ส่งค่าข้อมูลที่มีค่าเช่นเดียวกับบรรทัดที่ผ่านมา เพียงแต่กำหนดค่าข้อมูลเป็นเลขฐานสิบหก

ตารางที่ 2.5 คำสั่งการรับข้อมูลเข้ามาทางพอร์ต

คำสั่ง	ความหมาย
MOV A, P1	อ่านข้อมูลเข้ามาพอร์ต 1 และนำไปเก็บไว้ใน รีจิสเตอร์ A
MOV R0, P2	อ่านข้อมูลเข้ามาจากพอร์ต 2 และนำไปเก็บไว้ใน รีจิสเตอร์ R0

2.2.5 การใช้งาน 8255 กับ MCS-51

การขยายเพิ่มเติมพอร์ตอินพุท/เอาท์พุตของ MCS-51 นอกจากใช้ไอซีประเภทแลตช์ และบัฟเฟอร์ประกอบเข้ากับบัสของระบบแล้ว ยังสามารถใช้วงจรLSI เบอร์ 8255 มาทำหน้าที่เป็นพอร์ตอินพุต/เอาท์พุต ตามการโปรแกรมด้วยซอฟท์แวร์

ไอซีเบอร์ 8255 ได้รับการออกแบบมาเพื่อทำหน้าที่เป็นพอร์ตสำหรับการรับส่งข้อมูลแบบขนานระหว่างอุปกรณ์ภายนอกกับไมโครคอนโทรลเลอร์ ซึ่ง 8255นี้ สามารถเปลี่ยนแปลงลักษณะการทำงานของพอร์ตให้เป็นการิอนพุต หรือเอาท์พุตได้โดยการส่งข้อมูลควบคุมจากไมโครคอนโทรลเลอร์ก่อนที่จะเริ่มต้นใช้งาน คือ สามารถโปรแกรมการทำงานได้

จากรูปที่ 2.14 แสดงบล็อกภายในของ 8255 ทั้ง PA, PB, PC เป็นพอร์ตขนานแบบ 8 บิต 8255 ได้จัดกลุ่มพอร์ตเป็น 2 กลุ่ม คือ GROUP A ซึ่ง มีพอร์ต A ทุกบิต (PA0-PA7) และ พอร์ต C 4 บิตบน (PC4-PC7) กับ GROUP B ซึ่งมี พอร์ต B ทุกบิต (PB0-PB7) และพอร์ต C 4บิตล่าง (PC0-PC3) ส่วนหน้าที่ของขาสัญญาณต่างๆใน 8255 มีดังนี้

D0-D7 : กลุ่มเส้นสัญญาณข้อมูลของ 8255 เมื่อมีการเขียน หรืออ่าน

CS : สัญญาณเลือกอุปกรณ์ เมื่อขาสัญญาณนี้เป็นลอจิกต่ำ ซีพียูสามารถเขียน หรืออ่านข้อมูลจาก 8255ได้

RD : สัญญาณบอกสถานะต้องการอ่านข้อมูลจากรีจิสเตอร์ของ 8255

WR : สัญญาณบอกสถานะต้องการเขียนข้อมูลให้กับรีจิสเตอร์ของ 8255

A0-A1 : สัญญาณระบุตำแหน่งรีจิสเตอร์ภายใน 8255 ที่ต้องการ

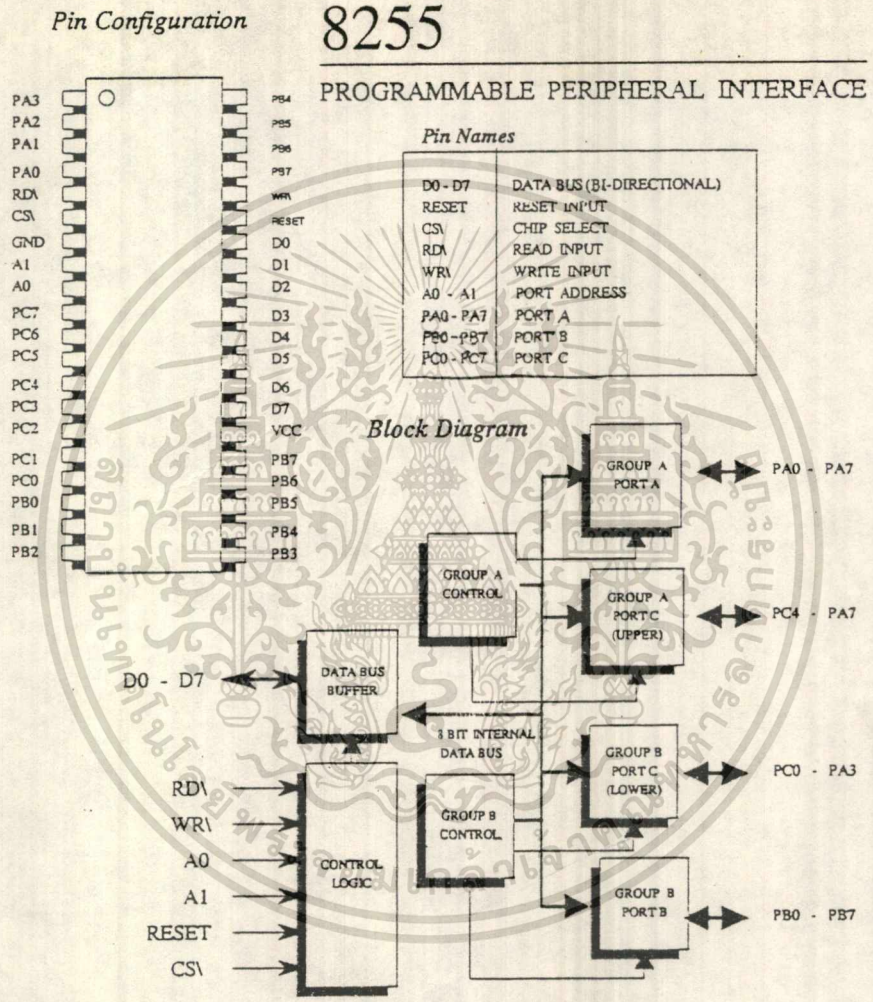
เอกสารนี้ [illegible] ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RESET : สัญญาณการรีเซตวงจรทำงานภายใน 8255 เพื่อเริ่มต้นใหม่

PA0-PA7 : กลุ่มสัญญาณ 8 เส้น เมื่อทำการติดต่อกับพอร์ต A ของ 8255

PB0-PB7 : กลุ่มสัญญาณ 8 เส้น เมื่อทำการติดต่อกับพอร์ต B ของ 8255

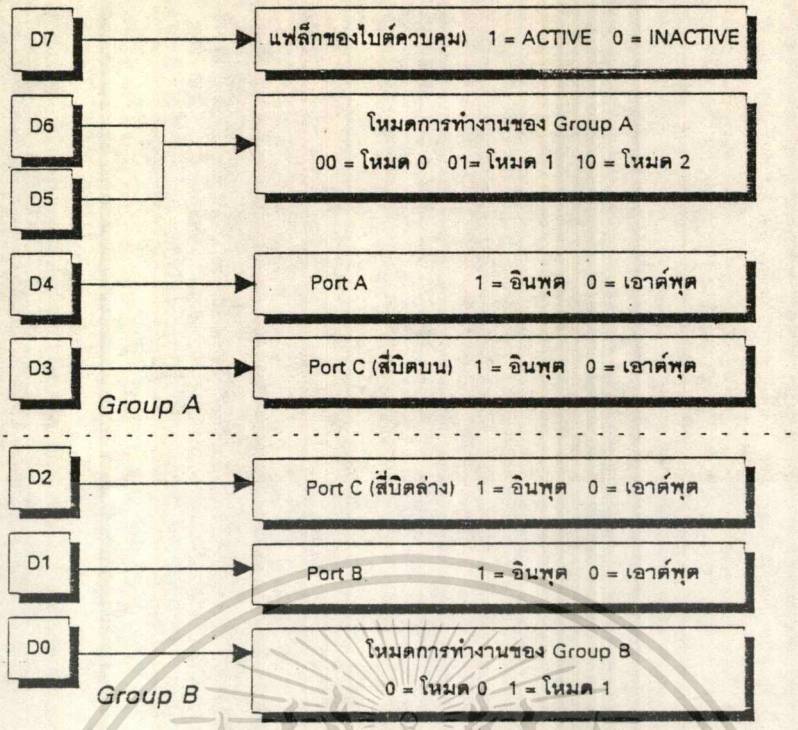
PC0-PC7 : กลุ่มสัญญาณ 8 เส้น เมื่อทำการติดต่อกับพอร์ต C ของ 8255



รูปที่ 2.14 แผนภาพแบบบล็อกภายใน และขาสัญญาณของไอซี 8255

การกำหนดให้พอร์ตทั้ง 3 ของ 8255 ทำงานในลักษณะต่างๆกัน นั้นเรียก โหมดการทำงาน (MODE) จะเริ่มต้น ด้ วยการส่งไบต์ข้อมูลควบคุม (Control Word) ให้กับรีจิสเตอร์ควบคุมการทำงาน โดยแต่ละบิตของข้อมูลจะมีความหมายที่ระบุถึงความต้องการต่างๆ ดังแสดงในรูปที่ 2.15 ซึ่งจะเห็นว่าเราสามารถกำหนดให้พอร์ตใดเป็นอินพุต หรือเอาท์พุตก็ได้ ส่วนการกำหนดให้ไบต์ควบคุมมีผลนั้น บิต D7 ต้องเป็น 1 เสมอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 ความหมายของบิตภายในของไบต์ข้อมูลควบคุม

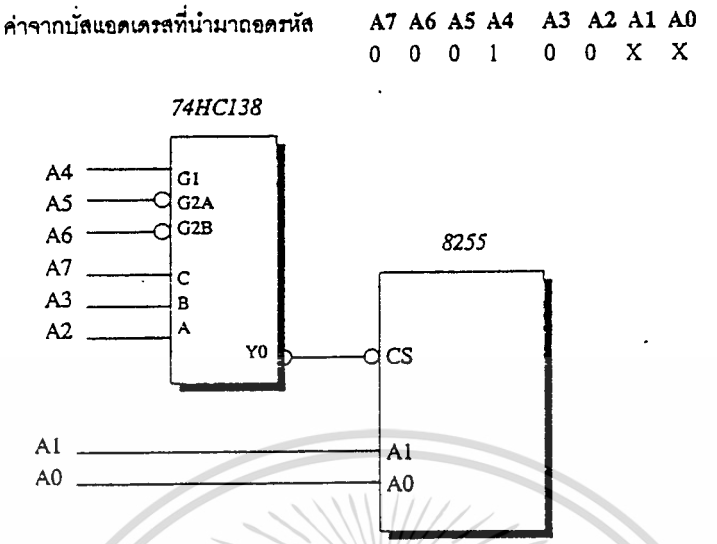
ส่วนการเชื่อมต่อ 8255 กับ 8051 นั้น เมื่อพิจารณาจากแผนภาพ 8255 จะเห็นว่า มีขาสัญญาณแอดเดรส 2 เส้น คือ A0 และ A1 ทำให้สามารถอ้างตำแหน่งได้ 2^2 เท่ากับ 4 ตำแหน่ง ซึ่งแต่ละตำแหน่งจะมีความหมายถึงการระบุรีจิสเตอร์ หรือพอร์ตภายใน 8255 ดังตารางที่2.6

ตารางที่ 2.6 การระบุแอดเดรส A0-A1 ของพอร์ตต่างๆภายใน 8255

A1	A0	ชื่อของรีจิสเตอร์
0	0	พอร์ต A
0	1	พอร์ต B
1	0	พอร์ต C
1	1	รีจิสเตอร์ควบคุม

ซึ่งเมื่อนำค่า A0, A1 นี้มาพิจารณาร่วมกับขาสัญญาณ RD และ WR จะเป็นการอ่าน หรือเขียนข้อมูลทางขาสัญญาณ D0-D7 ให้กับพอร์ตนั้น โดยทั่วไปมักกำหนดให้แอดเดรสของ 8255 ทั้ง 4 ตำแหน่งนี้อยู่ในช่วงใดช่วงหนึ่งของระบบ เช่น 10h, 11h, 12h, 13h โดยขาสัญญาณที่นอกเหนือไปจากA0, A1 จะนำมาถอดรหัสแอดเดรส เพื่อสร้างสัญญาณเลือกอุปกรณ์ (CS) ในช่วงที่แอดเดรสต้องการ ดังแสดงในรูปที่ 2.16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.16 แผนภาพแสดงการสร้างสัญญาณเลือกอุปกรณ์ (CS) ให้กับ 8255 โดยการถอดรหัสจากบัสแอดเดรส A2-A7

เมื่อค่าในบัสแอดเดรส A2-A7 มีค่าเท่ากับ 0000100xx (xx ใช้ระบุพอร์ตภายใน 8255) ก็คือ CS จะเป็นสภาวะลอจิกต่ำ จากวงจรพอร์ต 8255 จะมีค่าดังตารางที่ 2.7

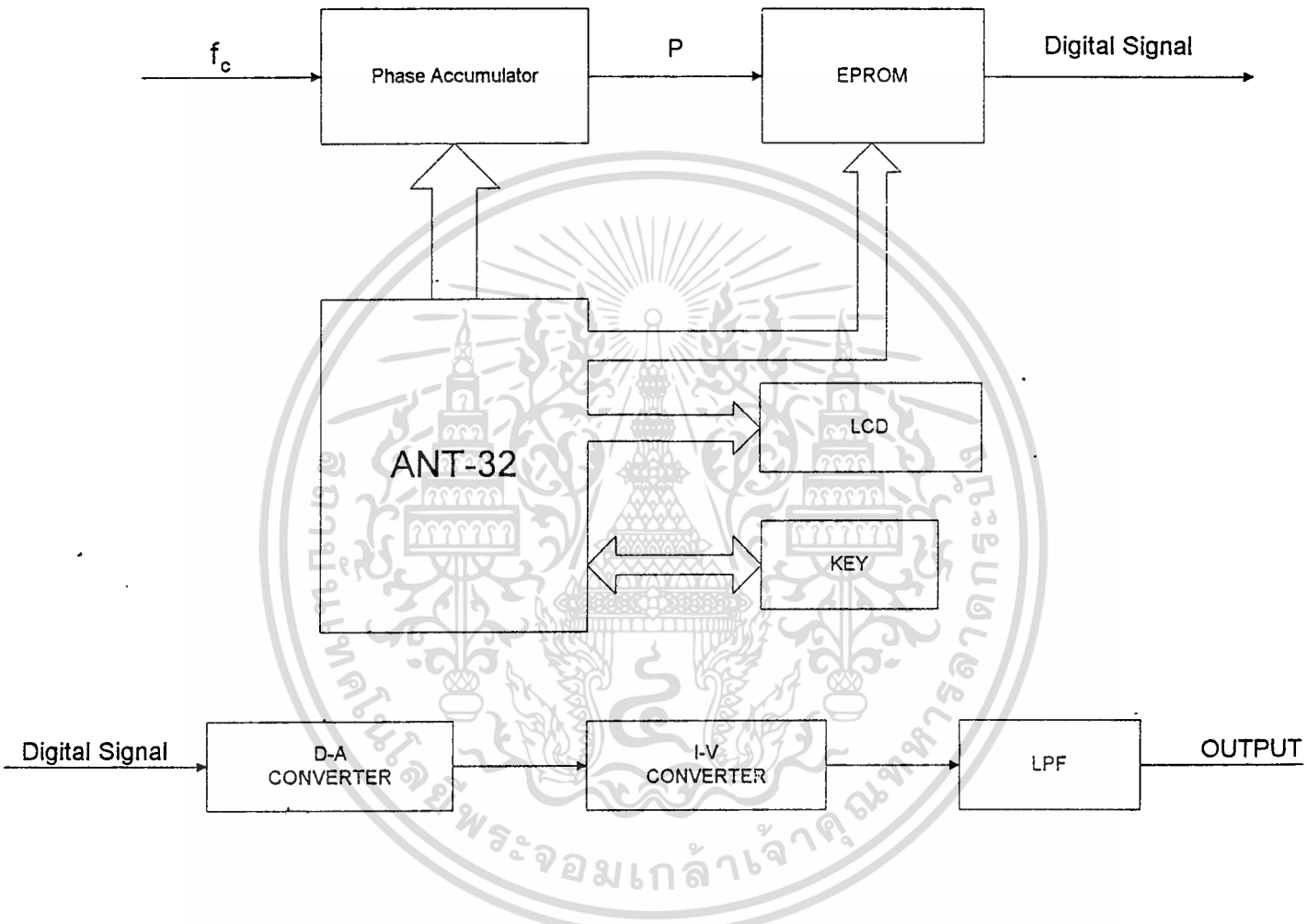
ตารางที่ 2.7 ตำแหน่งแอดเดรสของพอร์ตต่างๆใน 8255

ตำแหน่งแอดเดรส	ความหมาย
10h	พอร์ต A
11h	พอร์ต B
12h	พอร์ต C
13h	รีจิสเตอร์ควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การคำนวณและการสร้าง

3.1 การออกแบบ

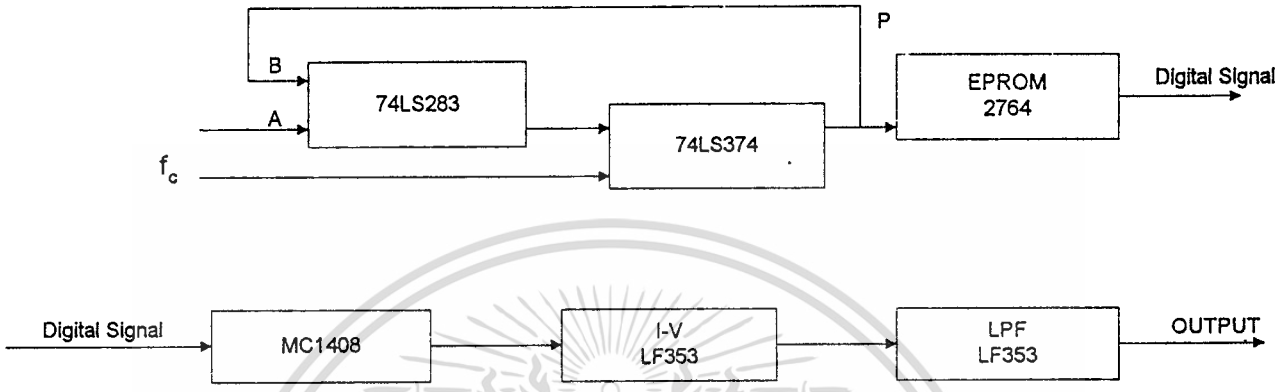


รูปที่ 3.1 บล็อกไดอะแกรมของวงจรกำเนิดสัญญาณแบบดิจิตอล

จากรูปที่ 3.1 บล๊อกกลุ่มบนแสดงเฟสแอกคิวมูเลเตอร์ซึ่งทำหน้าที่คำนวณหาค่าความถี่ ที่กำหนดจากผู้ใช้ผ่านคีย์กดซึ่งควบคุมผ่าน ANT-32 และแสดงข้อมูลการเลือกคีย์ผ่านจอ LCD จากนั้นผ่านมาสู่อีพรอมซึ่งบล๊อกนี้จะรับค่าที่คำนวณได้จากเฟสแอกคิวมูเลเตอร์ และค่าจากการเลือกรูปสัญญาณ มาเปลี่ยนเป็นสัญญาณดิจิตอล โดยใช้ข้อมูลของสัญญาณรูปต่างๆ ที่ได้จากโปรแกรมคำนวณค่าที่ได้เก็บไว้ในตัวอีพรอมแล้ว จากนั้นสัญญาณดิจิตอลที่ได้ถูกส่งมายังบล๊อกกลุ่มล่าง เพื่อแปลงเป็นสัญญาณอนาล๊อกในส่วนของดีทูเอคอนเวอร์เตอร์ ค่าที่ได้ยังคงอยู่ในรูปของกระแส จึงต้องผ่านวงจรแปลงให้อยู่ในรูปของแรงดันไฟฟ้าเสียก่อนแล้วจึงส่งผ่านไปยังวงจรกรองความถี่ต่ำเพื่อให้รูปสัญญาณสมบูรณ์ขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความถี่ของสัญญาณเอาท์พุทที่ได้จะขึ้นอยู่กับสัญญาณนาฬิกา (fc) และจำนวนบิทจากเฟสแอกคิวมูเลเตอร์ ค่าความละเอียดของความถี่เท่ากับ fc/2^{N} และถ้าค่าไบนารี่บิทจาก ANT-32 (M) ความถี่เอาท์พุทจะหาได้จากสูตร 2^{M} x fc/2^{N} ในการออกแบบ



รูปที่ 3.2 แสดงชิพเบอร์ต่างๆ ที่ใช้ในบล๊อกไดอะแกรม 3.1

3.2 การประยุกต์ใช้งาน ANT-32 กับโครงงาน

ในโครงงานนี้มีการนำ ANT-32 มาประยุกต์ใช้งาน เพื่อให้โครงงานมีประสิทธิภาพยิ่งขึ้นจึงต้องทำความเข้าใจกับบอร์ด ANT-32 ก่อน

3.2.1 คุณสมบัติของบอร์ด ANT-32

ANT-32 มีส่วนประกอบต่างๆ ดังภาพแสดงบอร์ด,ตำแหน่งจัมพ์เปอร์ และพอร์ทต่างๆ ดังรูป 3.3 ทั้งนี้ชิพในบางตำแหน่งที่ไม่ได้ถูกนำมาใช้ในโครงงาน

- เป็นบอร์ดคอนโทรลใช้กับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 (8031/8032) ซึ่งใช้ CPU เบอร์ 80C32 ทำงานที่ความถี่สัญญาณนาฬิกา 11.0592 Mhz
- ใช้งานหน่วยความจำบนบอร์ดได้ 3 ตำแหน่งด้วยกัน คือ

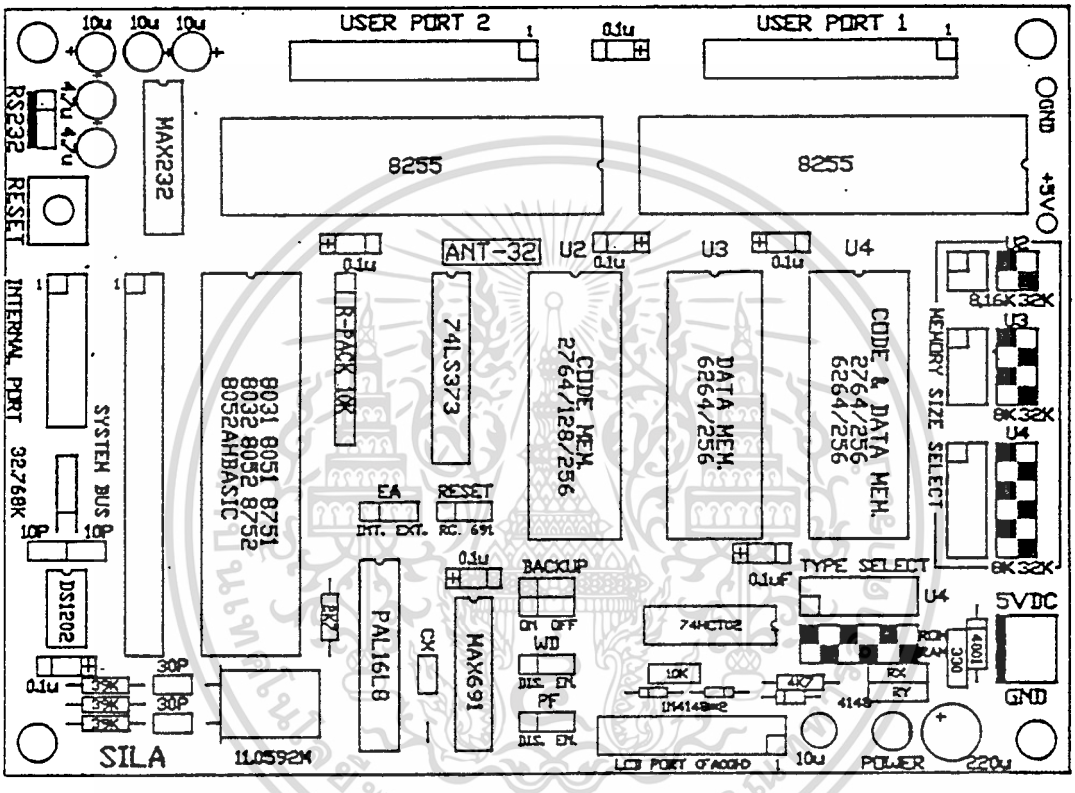
U2 เป็นหน่วยความจำโปรแกรม (Program Memory) ใช้กับอีพรอมขนาด 8-32 กิโลไบท์ เบอร์ 2764,27128 หรือ 27256

U3 เป็นหน่วยความจำข้อมูล (Data Memory) ใช้กับแรมขนาด 8 กิโลไบท์ เบอร์ 6264 หรือ 32 กิโลไบท์ เบอร์ 62256

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
037219

U4 เป็นหน่วยความจำโปรแกรมและข้อมูล ใช้กับอีพรอม หรือแรม เบอร์ 2764, 27256, 6264 หรือ 62256

- พอร์ท I/O เบอร์ 8255 จำนวน 2 ตัว (48 บิท) สำหรับต่อเพื่อใช้งานกับอุปกรณ์ภายนอก- พอร์ท LCD สำหรับการต่อใช้งานกับ LCD แบบ DOT MATRIX
- วงจร SERIAL INTERFACE DRIVER RS232 เบอร์ MAX232 ต่อเข้ากับเครื่องไมโครคอมพิวเตอร์
- วงจร RTC (Real Time Clock) เบอร์ DS1202



รูปที่ 3.3 แสดงบอร์ด และตำแหน่งจัมพ์เปอร์

3.2.2 การประยุกต์ใช้งาน ANT-32 กับโครงงาน

จากคุณสมบัติของ ANT-32 ทำให้ทราบว่าใช้ไมโครคอนโทรลเลอร์เบอร์ 8032 เป็นซีพียู ซึ่ง 8032 ไม่มีหน่วยความจำโปรแกรม (program memory) อยู่ภายใน ดังนั้นในการใช้งาน เราต้องใช้หน่วยความจำโปรแกรมภายนอกเสมอ ในโครงงานเราเลือก EPROM เบอร์ 2764 ซึ่งมีขนาด 8K x 8 มาใช้เก็บโปรแกรมควบคุมการทำงานของอุปกรณ์ภายนอกที่นำมาเชื่อมต่อกับ ANT-32

เนื่องจากระบบบัสแอดเดรส และบัสข้อมูลของ 8032 เป็นลักษณะการมัลติเพล็กซ์จากพอร์ตเดียวกัน คือในช่วงเริ่มต้น เส้นสัญญาณเหล่านี้ของพอร์ตจะใช้ในการส่งค่าแอดเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงเปลี่ยนไปเป็นสภาวะอิมพีแดนซ์สูง เพื่อใช้งานในฐานะของบัสข้อมูล แต่ EPROM นั้นไม่ใช้การมัลติเพล็กซ์ มันจะมีขาสัญญาณบัสแอดเดรส และบัสข้อมูลแยกจากกัน ดังนั้นในการเชื่อมต่อ

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

EPROM เพื่อทำหน้าที่เป็นหน่วยความจำโปรแกรมภายนอกกับ 8032 นั้น ต้องวงจรแลตช์เพิ่มเติม เพื่อทำการค้างค่าแอดเดรสที่ส่งมาจาก 8032 ในช่วงเริ่มต้น ให้กับขาสัญญาณของ EPROM โดยในบอร์ด ANT-32 .ใช้ไอซีเบอร์ 74HC7373 เป็นวงจรแลตช์

แต่ 8032 มีหน่วยความจำข้อมูลภายในตัวมันเองจำนวน 256 ไบต์ ซึ่งก็ยังไม่เพียงพอในการเก็บข้อมูลที่ใช้ในการคำนวณ เราจึงใช้ RAM เบอร์ 6264 ซึ่งมีขนาด 8K x 8 มาช่วยในการเก็บข้อมูล

การใช้งาน ANT-32 จะมีการกำหนดหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลภายนอกให้เริ่มต้นที่แอดเดรส 0000H เสมอ เพราะเมื่อมีการรีเซต หรือเริ่มต้นการจ่ายไฟฟ้าให้กับระบบ 8032จะได้เริ่มต้นทำงานตามคำสั่งที่แอดเดรสนี้ทันที

ในการที่เรานำเอา RAM 6264 มาใช้เป็นหน่วยความจำข้อมูลภายนอกจะทำให้มีการนำพอร์ตอินพุต/เอาท์พุตข้อมูลของ 8032 ไปใช้ติดต่อกับหน่วยความจำข้อมูลภายนอก ดังนั้นเพื่อเป็นการเพิ่มเติมพอร์ตอินพุต/เอาท์พุต ในบอร์ด ANT-32 จึงมีการนำไอซีเบอร์ 8255 มาใช้ ซึ่ง 8255 นี้สามารถทำหน้าที่เป็นได้ทั้งพอร์ตอินพุต/เอาท์พุต ตามแต่เราจะโปรแกรม (Program Peripheral Interface) และใช้เป็นส่วนที่เชื่อมต่อกับอุปกรณ์ภายนอกโดยตรง

สำหรับบอร์ด ANT-32 เราใช้ 8255 จำนวน 2ตัว เพราะฉะนั้นจะมีพอร์ตอินพุต/เอาท์พุตจำนวน 24x2 = 48 บิต โดยแบ่งเป็น USER PORT1 และ 2 ซึ่งมีตำแหน่งแอดเดรส และการทำงาน ดังนี้

USER PORT 1	แอดเดรส F800H + 8255 offset addr	= actual addr
พอร์ท A	แอดเดรส F800H + 00H	= F800H
พอร์ท B	แอดเดรส F800H + 01H	= F801H
พอร์ท C	แอดเดรส F800H + 02H	= F802H
โหมดพอร์ท	แอดเดรส F800H + 03H	= F803H
USER PORT 2	แอดเดรส FC00H + 8255 offset addr	= actual addr
พอร์ท A	แอดเดรส FC00H + 00H	= FC00H
พอร์ท B	แอดเดรส FC00H + 01H	= FC01H
พอร์ท C	แอดเดรส FC00H + 02H	= FC02H
โหมดพอร์ท	แอดเดรส FC00H + 03H	= FC03H

ทุกพอร์ตใน USER PORT2 ใช้เป็นเอาท์พุตพอร์ต โดยข้อมูลที่ออกทางพอร์ต PC4 และ PC5 จะไปเป็นแอดเดรสแอดเดรสให้กับ EPROM (A11-A12) ในการเลือกชนิดของสัญญาณ ดังนี้

แอดเดรส 00 เลือกสัญญาณรูปคลื่นซายน์

แอดเดรส 10 เลือกสัญญาณรูปคลื่นสี่เหลี่ยม

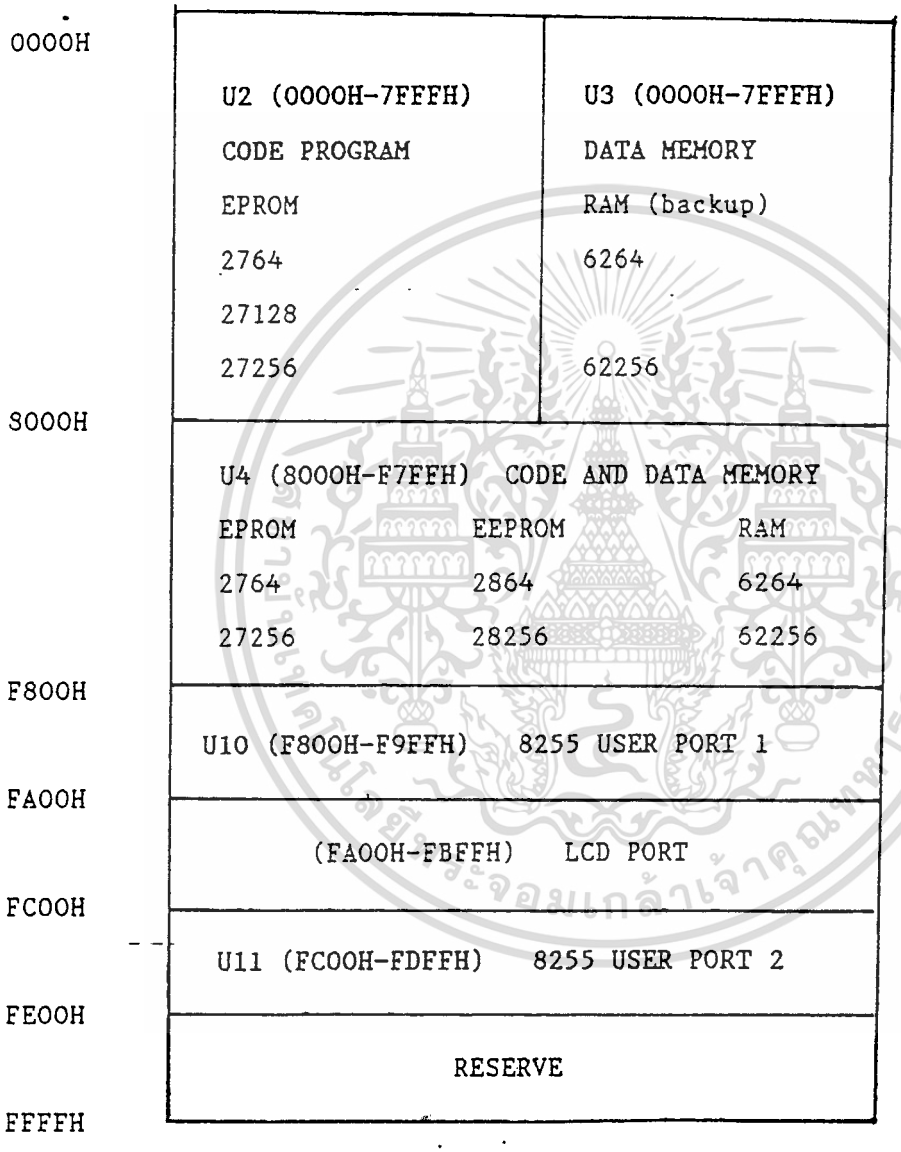
แอดเดรส 20 เลือกสัญญาณรูปคลื่นสามเหลี่ยม

แอดเดรส 30 เลือกสัญญาณรูปคลื่นฟันเลื่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลที่ออกทางพอร์ต PA0-PA7, PB0-PB7 และ PC0-PC2 จะไปเป็นอินพุตให้วงจรฟูลแอดเดอร์ 4 บิต จำนวน 5ตัว (โดยตัวที่ 5 ใช้แค่ 3 บิต) และข้อมูลที่ออกจากส่วนนี้จะเข้าแลตช์ แล้วไปเป็นแอดเดรสให้กับ EPROM 2716 เพื่อเรียกข้อมูลประจำแต่ละค่าแอดเดรสออกมา (เป็นข้อมูลที่ได้โปรแกรมไว้ในลักษณะของสัญญาณดิจิตอง ที่เป็นตัวแทนของขนาดแอมปลิจูดของสัญญาณ)

ตารางที่ 3.1 แสดงเมโมรีแมปของ ANT-32



บอร์ด ANT-32 จะมีพอร์ท LCD ให้สำหรับการต่อใช้งานเข้ากับ LCD MODULE แบบ DOT MATRIX ได้ทันที ซึ่งจะใช้ขาสัญญาณทั้งหมด 14 ขา และสำหรับการใช้งานพอร์ท LCD นั้นจะมีการจัดวงจรในแบบเมมโมรีแม็ป (Memory Map) ซึ่งช่วยให้การเขียนโปรแกรมทำได้ง่าย โดยตำแหน่งต่างๆ สรุปได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แอดเดรส	ลักษณะของพอร์ทที่ติดต่อ
FA00H	สำหรับเขียนคำสั่ง (RS=0 R/W=0)
FA01H	สำหรับอ่านค่า BUSY (RS=0 R/W=1)
FA02H	สำหรับเขียนข้อมูล (RS=1 R/W=0)
FA03H	สำหรับอ่านข้อมูล (RS=1 R/W=1)

การอ่านค่า LCD แบบ DOT MATRIX นี้สามารถเลือกรุ่นใดก็ได้ โดยมีจำนวนตัวอักษรต่อบรรทัดและจำนวนบรรทัดตามที่ต้องการ เพราะสายสัญญาณที่ใช้จะใช้แบบเดียวกันหมด แตกต่างกันที่โปรแกรมเท่านั้น

3.3 การใช้งาน LCD MODULE

โครงงานนี้ได้นำ LCD MODULE (LCM) ในการแสดงผลเนื่องจากสามารถแสดงผลเป็นตัวอักษรและตัวเลขได้ทั้งยังกินกระแสไฟต่ำ

3.3.1 การประยุกต์ใช้ LCM กับ MCS-51

ปัจจุบัน LCD ส่วนใหญ่จะประกอบเป็นโมดูลเพื่อสะดวกในการใช้งาน

LCM มีส่วนประกอบสำคัญดังนี้

1. DOT MATRIX LCD : เป็นส่วนที่ทำหน้าที่แสดงผล โดยใช้หลักการหักเหของแสงผ่านผลึกโดยจะประกอบไปด้วยจุด (pixel) จำนวนมากที่สามารถบังคับให้ติดหรือดับได้ทุกจุด

2. DRIVER : เป็นวงจรที่ขับ LCD ส่วนใหญ่จะใช้ชิพเบอร์ HD44110H

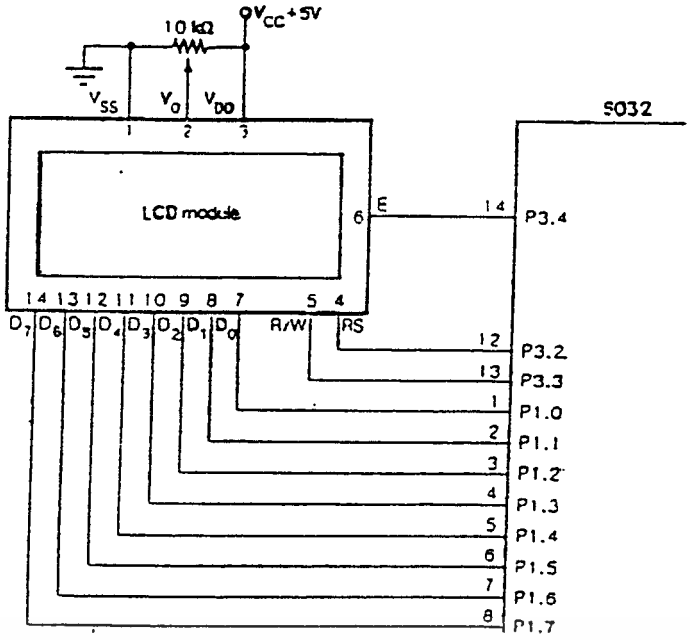
3. คอนโทรลเลอร์ : เป็นส่วนที่ใช้ควบคุมการทำงานทั้งหมดของ LCM โดยจะรับข้อมูลจากภายนอกมาจัดการให้ LCD แสดงผลในรูปแบบต่างๆ ซึ่งส่วนใหญ่ใช้ชิพเบอร์ HD44780

สำหรับโครงงานนี้ เราสามารถต่อ LCD เข้ากับ LCD PORT ใน ANT-32 ได้ โดยโครงสร้างภายในคล้ายการอินเตอร์เฟสกับ 8032 ดังรูปที่ 3.4

ในการเริ่มต้นใช้งาน LCM เพื่อให้ทำงานได้ถูกต้องจำเป็นต้องมีการรีเซทเสียก่อนเช่นเดียวกับชิพไมโครโปรเซสเซอร์ การรีเซทชิพคอนโทรลเลอร์ HD44780 ซึ่งทำได้ 2 วิธีดังนี้

1. เริ่มต้นใช้งานโดยวงจรที่ทำหน้าที่รีเซทภายในชิพคอนโทรลเลอร์ HD44780 สามารถรีเซทวงจรภายในได้เองในขณะเริ่มต้นใช้งาน ในขณะทำตามคำสั่งเมื่อมีการรีเซท สัญญาณ BF จะอยู่ในสถานะ Busy จนกระทั่งกระบวนการรีเซทภายในเสร็จสิ้น นั่นคือสัญญาณ BF จะมีค่าเป็น 1 เป็นเวลา 10 มิลลิวินาที ภายหลังจากที่ไฟเลี้ยงชิพมีค่าเป็น 4.5 โวลท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.4 ตัวอย่างการอินเตอร์เฟสระหว่าง MCS-51 กับ LCM

คำสั่งที่ทำงานในช่วง Busy หรือในช่วงรีเซทมีดังต่อไปนี้ (อ้างอิงจากตารางคำสั่ง)

- Clear Display

- Function Set

DL = 1 : Interface Data Length 8 บิท

N = 0 : Display Line เป็น 1

F = 0 : Character Font 5X7 dots

- Display ON/OFF Control

D = 0 : Display OFF

C = 0 : Cursor OFF

B = 0 : Blink OFF

- Entry Mode Set

I/D = 1 : Increment (+1)

S = 0 : No Shift

- หลังจากช่วงนี้ผ่านไปแล้วสามารถเริ่มต้นใช้งานได้

หาก Rise Time ของแหล่งจ่ายมีค่าอยู่นอกช่วง 0.1-10 มิลลิวินาที หรือเมื่อไฟเลี้ยงที่จ่ายให้ LCM มีค่าตกลงมาต่ำกว่า 0.2 โวล์ท น้อยกว่า 1 มิลลิวินาที ทำให้วงจรที่ทำหน้าที่รีเซทโดยอัตโนมัติภายในชิพไม่สามารถทำงานได้ถูกต้อง

กรณีที่การรีเซทของวงจรภายในชิพทำงานผิดพลาดเนื่องมาจากสาเหตุใดก็ตาม จำเป็นที่วงจรภายนอกต้องทำการรีเซท LCM เองเสียก่อนโดยการส่งคำสั่งไปควบคุม LCM เองเพื่อให้การใช้งานไม่เกิดความผิดพลาด

2. การเริ่มต้นใช้งานโดยการส่งคำสั่งไปควบคุม LCM เองโดยวงจรภายนอก เพื่อป้องกันไม่ให้เกิดการรีเซตผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Instruction	Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Clear display	0	0	0	0	0	0	0	0	0	1	Clear or display and returns the cursor to the home position (address 0)
Return home	0	0	0	0	0	0	0	0	1	0	Returns the cursor to the home position (address0) . Also returns the display begin shifted to the original position DDRAM remain unchanged
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and not to shift the display these operatoins are performed during data write and read
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D) cursor ON/OFF (C), and block of cursor position character (B)
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	0	0	Moves the cursor and shifts the display without changing DDRAM contents
Function set	0	0	0	0	1	DL	N	F	0	0	Sets interface data length (DL) number of display lines (L) and character font (F)
Set CGRAM address	0	0	0	1	A_{CG}						Sets the CGRAM address, CGRAM data is sent and recieved after this setting
Set DDRAM address	0	0	1	A_{DD}							Sets the DDRAM address, DDRAM data is sent and recieved after these setting
Read Busy Flag and adderss	0	1	BF	AC							Reads Busy Flag (BF) for internal operations is being performed and reads address counter contents
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM
Read data to CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM

I/D = 1 Increment (+1), = 0 Decrement (-1)

S = 1 Accumulator display shift

S/C = 1 Display shift, = 0 Cursor move

R/L = 1 Shift to the right, = 0 Shift to the left

DL = 1 8 bits, = 0 4 bits

N = 1 2 lines, = 0 1 line

F = 1 5 x10 dots, = 0 5 x 7 dots

BF = 1 Unready operating

BF = 0 Can accept instruction

DDRAM Display data RAM

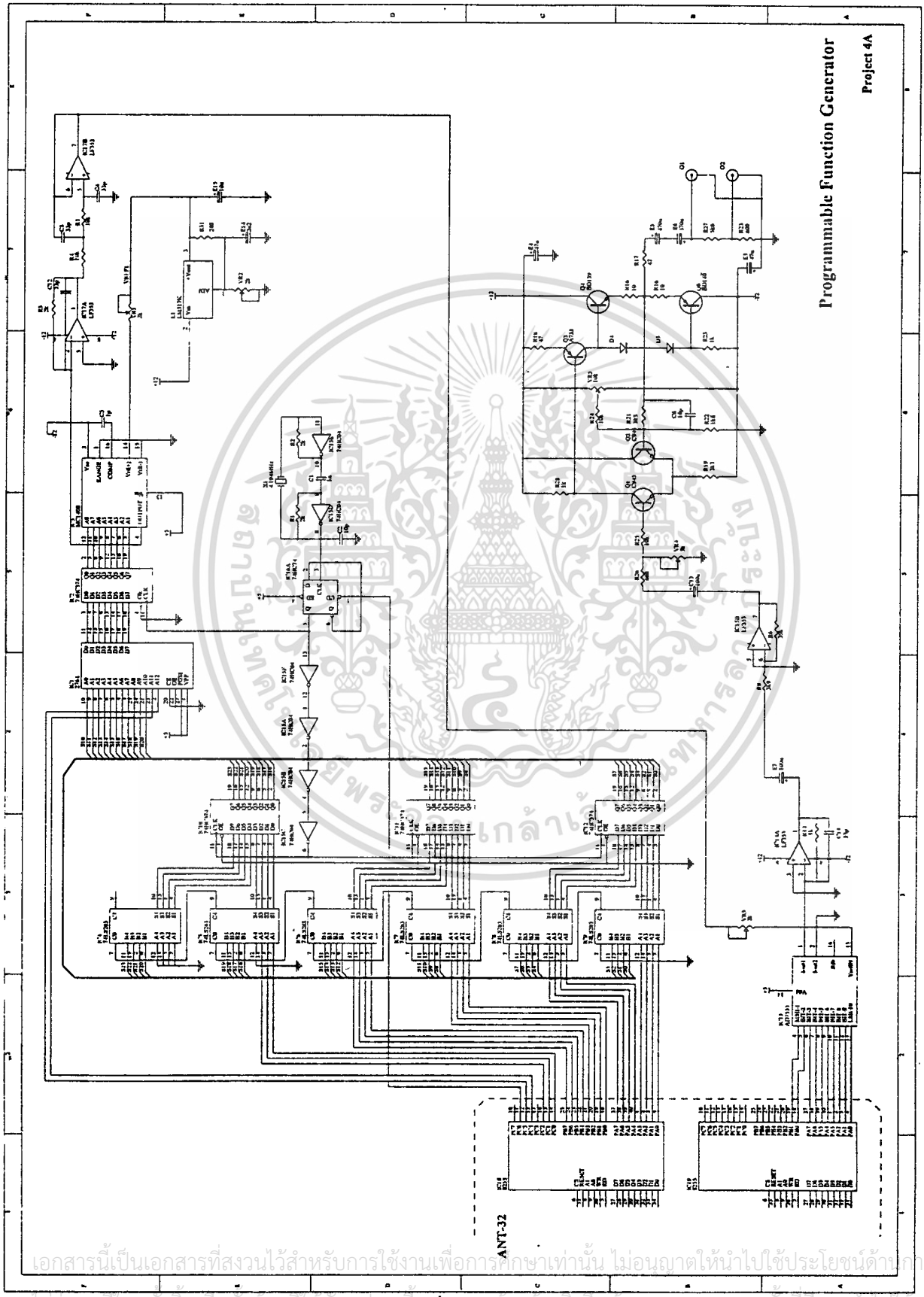
CGRAM Character generator RAM

A_{CG} CGRAM address

A_{DD} DDRAM address Corresponds to cursor address

AC Address counter user for both of DD and CGRAM address

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 การทำงานของวงจร

จากวงจรที่แสดง เริ่มจากส่วนของเฟสแอกคิวมูเลเตอร์ เอาท์พุทจากแลตช์ 74LS374 จะถูกป้อนกลับมาเป็นอินพุทของฟูลแอดเดอร์ 4 บิท 74LS283 โดยนำมาบวกกับค่าที่ตั้งไว้ จากการควบคุมของเอาท์พุทพอร์ท 8255(port2) จาก ANT-32 ผลบวกข้อมูลที่ได้ถูกนำไปเก็บไว้ในแลตช์อีก แล้วเอาท์พุทจากแลตช์จะวนไปเป็นอินพุทของฟูลแอดเดอร์เป็นรอบตามจังหวะของสัญญาณนาฬิกา 2.097 MHz (f_c) ซึ่งเป็นสัญญาณที่ได้จากการหารความถี่คริสตอล 4.197 MH z โดยผ่าน D flip-flop 74HC74

นอกจากนี้เอาท์พุทจากแลตช์ยังถูกส่งไปเป็นอินพุทเพื่อกำหนดแอดเดรสให้แก่ EPROM 2764 เพื่อเรียกข้อมูลที่เก็บในแอดเดรสนั้นออกมา สำหรับข้อมูลที่เก็บใน EPROM 2764 มีขนาด 8 กิโลไบท์ (2^{13}) โดยแบ่งแอดเดรสออกเป็น 4 ส่วนเพื่อเก็บข้อมูลที่เป็นสัญญาณซายน์ ,สี่เหลี่ยม ,สามเหลี่ยม และฟันเลื่อย โดยแต่ละรูปคลื่นมีแอดเดรสเก็บ 2 กิโลไบท์ (2^{11}) และมีสัญญาณเลือกรูปคลื่น 2 เส้นที่ส่งมาจากพอร์ท C ของ 8255(port2) PC5-PC4 ใน ANT-32 เพื่อกำหนดช่วงของแอดเดรสตามรูปคลื่นที่ต้องการ ข้อมูลเอาท์พุทจาก EPROM ที่ออกมาเป็นรอบตามจังหวะสัญญาณนาฬิกา f_c เป็นข้อมูลทางดิจิตอลจำนวน 8 บิท มีค่าระหว่าง 0-255 จะถูกส่งไปเก็บในแลตช์อีกครั้งเพื่อรอการส่งออกตามจังหวะสัญญาณนาฬิกา f_c แล้วจึงเข้าสู่ MC 1408 ดีทูเอคอนเวอร์เตอร์

MC 1408 จะแปลงสัญญาณทางดิจิตอลเป็นอนาลอกโดยมี V_{ref} จาก LM317K โวลเตจเรกูเลเตอร์มาควบคุม และเนื่องจากสัญญาณที่ได้จาก MC1408 ยังอยู่ในรูปกระแส จึงต้องผ่าน LF353 โดยใน LF353 ส่วนแรกจะแปลงสัญญาณกระแสไปอยู่ในรูปของแรงดัน และยังกรองความถี่ต่ำผ่านด้วย ซึ่งค่าแรงดันเอาท์พุต (V_o) คำนวณได้จากสมการ

$$V_o = (V_{ref} / R_1)(R_3)\left[A_1/2 + A_2/4 + A_3/8 + A_4/16 + A_5/32 + A_6/64 + A_7/128 + A_8/256 \right]$$

โดย A_0 - A_7 คือ ค่าข้อมูลในแต่ละบิต

ส่วน LF353 ส่วนหลังทำหน้าที่กรองความถี่หลัก ทำให้รูปสัญญาณที่ได้ราบเรียบยิ่งขึ้น

สำหรับสัญญาณนาฬิกาที่จ่ายให้วงจรในส่วนของเฟสแอกคิวมูเลเตอร์ต้องนำไปผ่านอินเวอร์เตอร์เพื่อให้แน่ใจว่าวงจรในส่วนเฟสแอกคิวมูเลเตอร์ทำงานช้ากว่าแลตช์ในส่วนของ EPROM

การคำนวณค่าความถี่สูงสุดที่สามารถสร้าง ได้จากค่าอินพุทของเฟสแอกคิวมูเลเตอร์จาก ANT-32 ซึ่งมี 19 บิท และค่าข้อมูล 23 บิทของเฟสแอกคิวมูเลเตอร์ คือ $2^{19} \times f_c/2^{23}$ = 131.072 kHz ประมาณ 100 KHz

สัญญาณแรงดันเอาท์พุทที่ได้จาก LF353 จึงขึ้นอยู่กับค่าความถี่ที่ผู้ใช้กำหนด และจะถูกควบคุมค่าแอมปลิจูด (1-10 V_{p-p}) โดย AD7533 ซึ่งกำหนดโดยผู้ใช้ผ่านทางพอร์ท A และ B ของ 8255(1) ในANT-32 ส่วนพอร์ท C จะต่อกับคีย์กดเพื่อตรวจการกดคีย์เลือกของผู้ใช้ จากนั้นเอาท์พุทที่ถูกควบคุมทั้งความถี่และแอมปลิจูดแล้วจะผ่าน LF353 อีกครั้งเพื่อกรองความถี่ แล้วจึงผ่านเข้าสู่วงจรส่วนสุดท้ายซึ่งเป็นภาคขยายสัญญาณ โดยประกอบด้วยวงจรขยายผลต่างของสัญญาณอินพุทซึ่งมีทรานซิสเตอร์ C945 2 ตัว แล้วส่งสัญญาณไปยังทรานซิสเตอร์ A733 ทำหน้าที่ไบอัสกระแสและขยายสัญญาณก่อนส่งไปให้ทรานซิสเตอร์ BD139 และ BD140 ขยายสัญญาณทั้งซีกบวกและซีกลบโดยจะผลัดกันทำงานเพื่อให้สัญญาณที่ออกมามีความถี่คงที่ตลอด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับข้อมูลรูปซายน์ที่เก็บใน EPROM ได้จากการใช้โปรแกรมภาษาซีคำนวณค่าจากสมการ 127.5 [SIN (2πP/2048 - π/2)] โดย P คือค่าแอดเดรสอยู่ในช่วงของสัญญาณซายน์

สัญญาณสี่เหลี่ยม, สามเหลี่ยม และฟันเลื่อยได้จากการใช้โปรแกรมภาษาซีเพื่อกำหนดค่าในแต่ละแอดเดรสตามลักษณะของสัญญาณแต่ละชนิดใน 1คาบเวลา

โดยโปรแกรมภาษาซีที่ใช้คำนวณค่าข้อมูลสัญญาณชนิดต่างๆ ที่นำไปเก็บใน EPROM คือ

```
/*      Sine     */

#include <stdio.h>
#include <math.h>

main()
{
  double p=0;
  double s=0;
  int S;
  double pi=3.14592654;
  int addr=0;
  int bytes=2048;

  while (addr < bytes)
     {
     if ( addr % 16==0)
       printf("\n%4x ", addr);
     p = 2.0*pi* ( (double) addr)/( (double) bytes);
     s = 256* (1.0+ sin (p-pi/2.0));
     S = ( (int) s );                    /* round */
     if ( S - s >= 0.5 )                 /*       */
        s++;                             /*       */
     printf("  %2x",S);
     addr++;
     }
  }

/*      Square     */

#include <stdio.h>

main()
{
  int Q=255;
  int addr=2048;
  int bytes=4096;

  while (addr < bytes)
     {
     if ( addr % 16==0)
       printf("\n%4x ", addr);
     printf("  %2x",Q);
     if ( addr >= 3072 )
       Q = 0;
     addr++;
     }
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*      Triangle     */

#include <stdio.h>

main()
{
  int T=0;
  int addr=4096;
  int bytes=6144;

  while (addr < bytes)
     {
     if ( addr % 16==0)
       printf("\n%4x ", addr);
     printf("  %2x",T);
     if ( (addr+1) % 4 ==0 )
       { if ( (addr+1)<5120 )
        T++;
       else {if ( (addr+1)>5123 ) T--;}
       }
      addr++;
     }
  }


/*      Saw-tooth     */

#include <stdio.h>

main()
{
  int W=0;
  int addr=6144;
  int bytes=8192;

  while (addr < bytes)
     {
     if ( addr % 16==0)
       printf("\n%4x ", addr);
     printf("  %2x",W);
     if ( (addr+1) % 8 == 0 )
       W++;
     addr++;
     }
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การทดลอง และผลการทดลอง

4.1 การทดลอง

คู่มือการใช้งานของเครื่อง

1. ปุ่ม POWER ON/OFF จะทำหน้าที่ปิดเปิดเครื่อง

2. จอแอลซีดี ทำหน้าที่สดงผลว่าสัญญาณที่เลือกนั้นเป็นสัญญาณอะไร

3. ปุ่มเลือกความถี่ (Freq) ทำหน้าที่เพื่อเลือกความถี่ของสัญญาณตั้งแต่ 10Hz - 100kHz เช่นเมื่อเราต้องการความถี่ 20 Hz นั้น กดไปที่ปุ่ม Freq และกดตัวเลข 20 แล้วกดหน่วยของความถี่ Hz

4. ปุ่มเลือกโวลท์ (Amp) ทำหน้าที่เพื่อเลือกโวลท์ ตั้งแต่ $1V_{p-p}$ - $10V_{p-p}$ เช่น เมื่อต้องการระดับสัญญาณ $5V_{p-p}$ ก็ให้กดที่ปุ่ม Amp และกดตัวลข 5 แล้วกดหน่วยโวลท์

5. ปุ่มเลือกรูปแบบสัญญาณ (Waveform) ทำหน้าที่เลือกสัญญาณที่ต้องการ เช่น ต้องการเลือสัญญาณซายน์ ให้กดปุ่ม waveform และไปกดหมายเลข 1 ก็จะได้สัญญาณซายน์ ถ้าต้องการสัญญาณสี่เหลี่ยมกดหมายเลข 2 ถ้าต้องการสัญญาณสามเหลี่ยมกดหมายเลข3 ถ้าต้องการสัญญาณฟันเลื่อยกหมายเลข 4

6. ปุ่ม BP ทำหน้าที่ลบออกทีละตัว

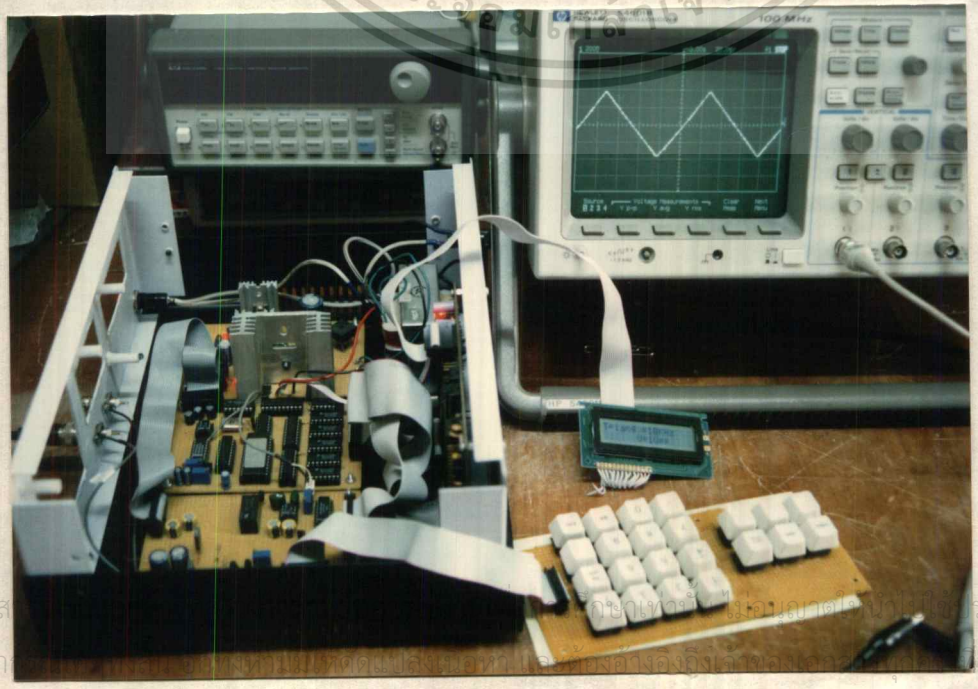
7. ปุ่ม ESC ทำหน้าที่แก้ไขเมื่อเวลากดผิด

8. ปุ่มจุด จะทำหน้าที่เป็นการกำหนดทศนิยม

9. ปุ่ม kHz, V จะเป็นปุ่มเลือกหน่วยของความถี่ หรือหน่วยของโวลท์

10. ปุ่ม Hz, mV จะเป็นปุ่มเลือกหน่วยของความถี่ หรือโวลท์

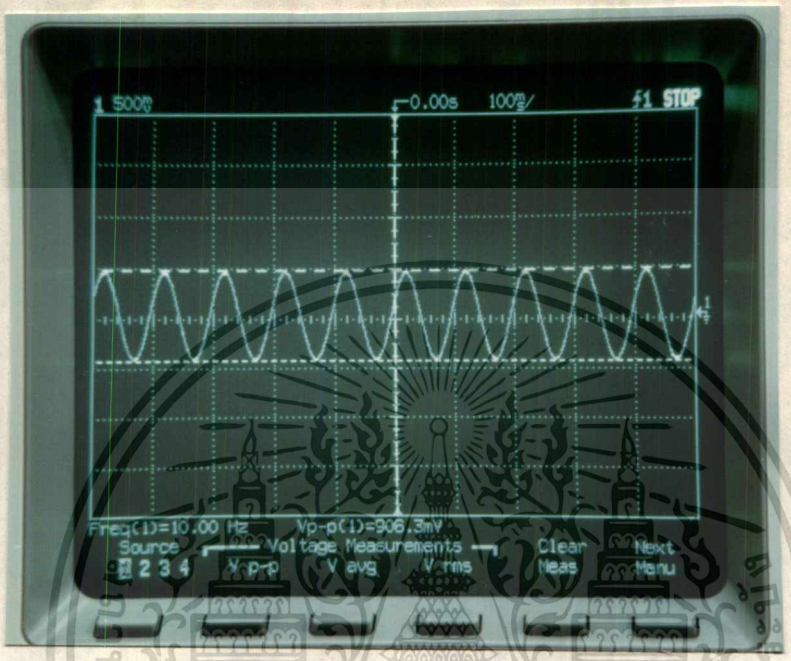
รูปแสดงเครื่องฟังก์ชันเจเนอเรเตอร์ และคีย์กด



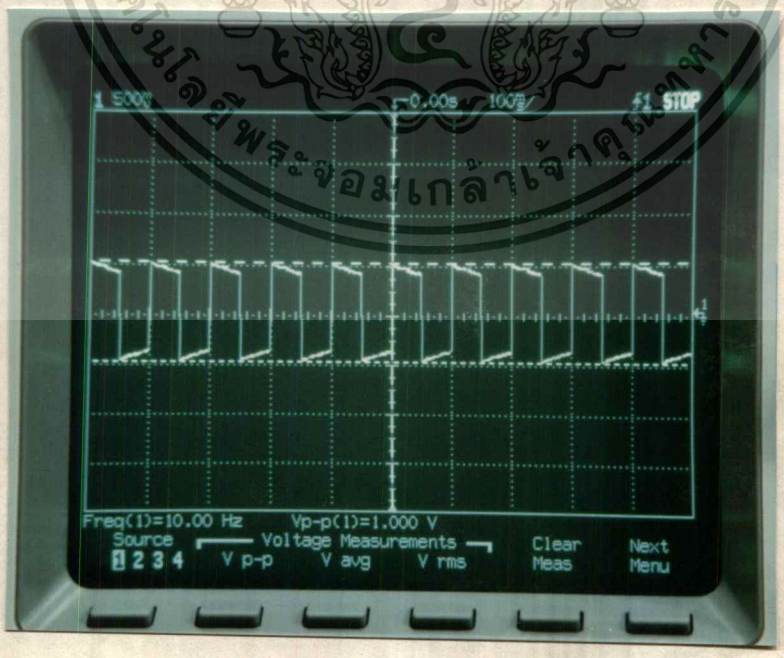
เอกสาร ... ไม่ว่า ... ไม่การนำไปใช้

4.2 วิธีการทดลอง และผลการทดลอง

1.นำเครื่องฟังก์ชั่นเจเนอเรเตอร์ต่อเข้ากับเครื่องสโคป เพื่อแสดงผลที่ได้จากการทดลองเลือกความถี่และแอมปลิจูดค่าต่างๆในช่วงขอบเขตที่เครื่อง สามารถสร้างได้ คือ ที่ระดับสัญญาณ $1V_{p-p}$ ความถี่ 10Hz, ระดับสัญญาณ $5V_{p-p}$ ความถี่ 1kHz, ระดับสัญญาณ $10V_{p-p}$ ความถี่ 100kHz

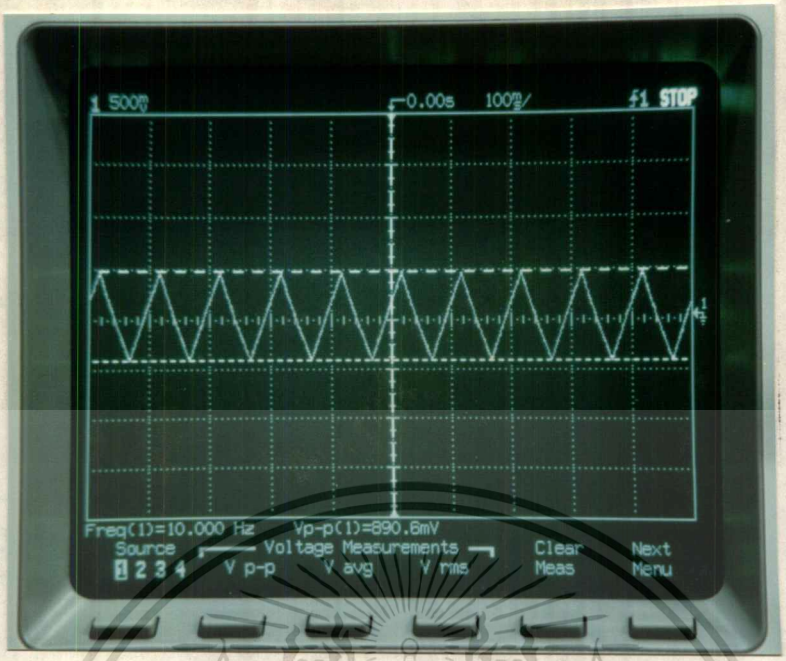


รูปที่ 4.1 สัญญาณซายน์ที่ความถี่ 10Hz แอมปลิจูด $1V_{p-p}$

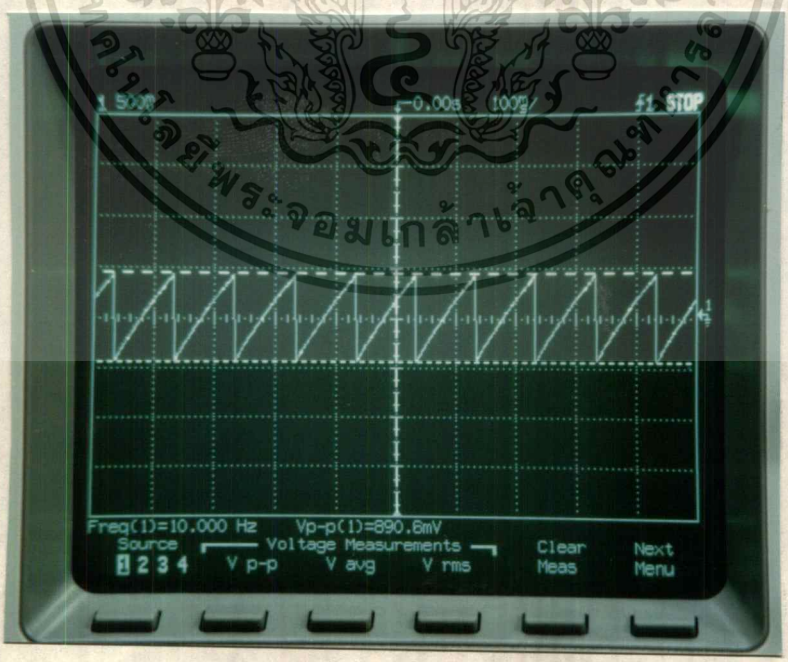


รูปที่ 4.2 สัญญาณสี่เหลี่ยม ที่ความถี่ 10Hz แอมปลิจูด $1V_{p-p}$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

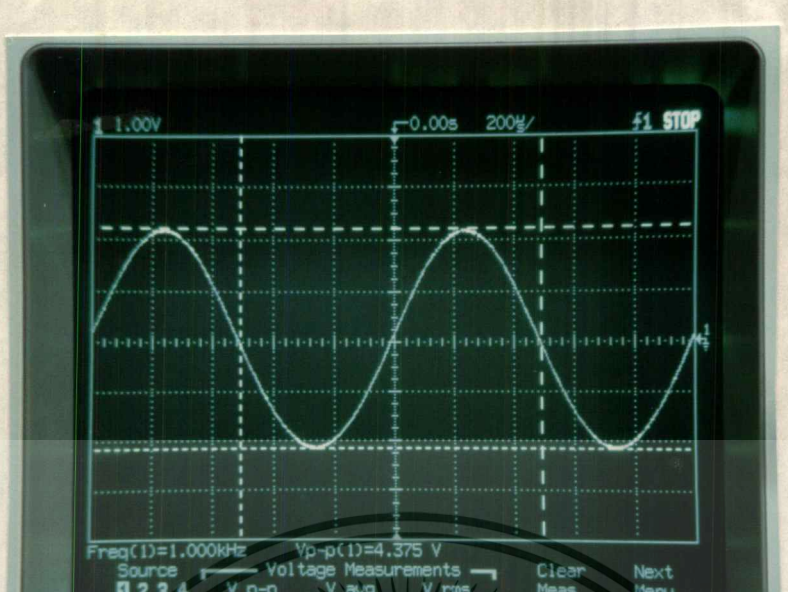


รูปที่ 4.3 สัญญาณสามเหลี่ยม ที่ความถี่ 10Hz แอมปลิจูด $1V_{p-p}$



รูปที่ 4.4 สัญญาณฟันเลื่อย ที่ความถี่ 10Hz แอมปลิจูด $1V_{p-p}$

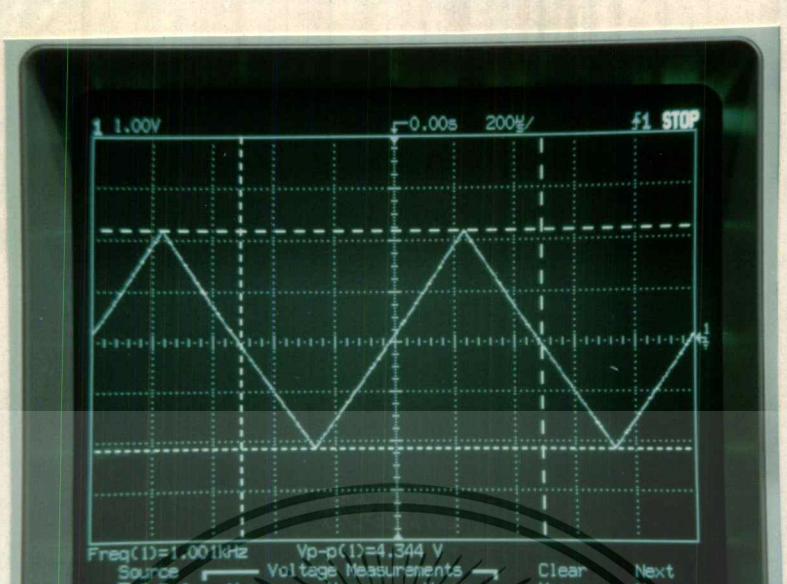
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.5 สัญญาณซายน์ ที่ความถี่ 1kHz แอมปลิจูด 5 V_{p-p}

รูปที่ 4.6 สัญญาณสี่เหลี่ยม ที่ความถี่ 1kHz แอมปลิจูด 5 V_{p-p}

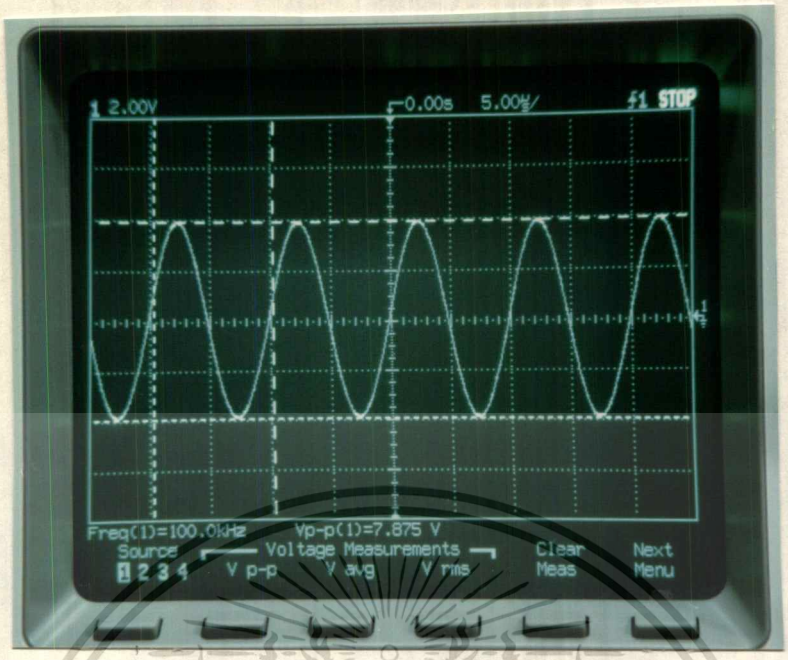
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



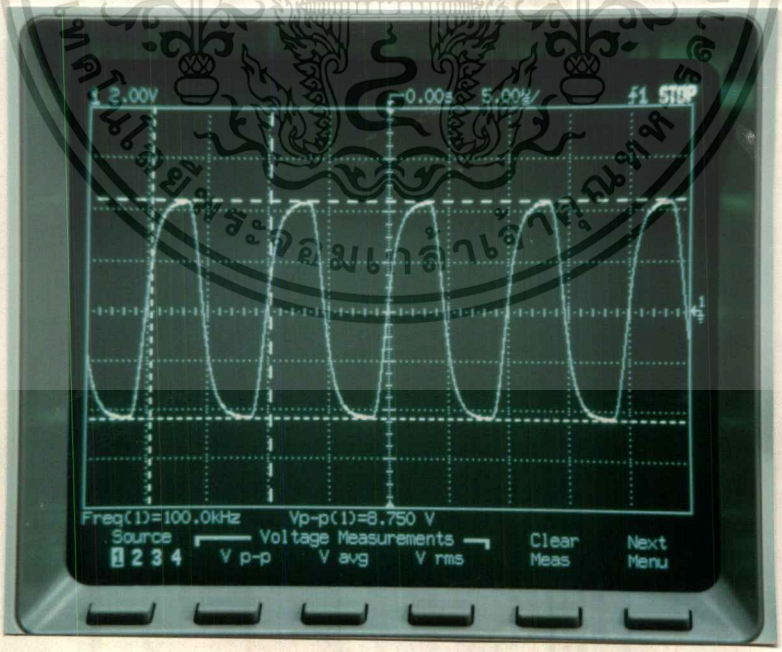
รูปที่ 4.7 สัญญาณสามเหลี่ยม ที่ความถี่ 1kHz แอมปลิจูด 5 V_{p-p}

รูปที่ 4.8 สัญญาณฟันเลื่อย ที่ความถี่ 1kHz แอมปลิจูด 5 V_{p-p}

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

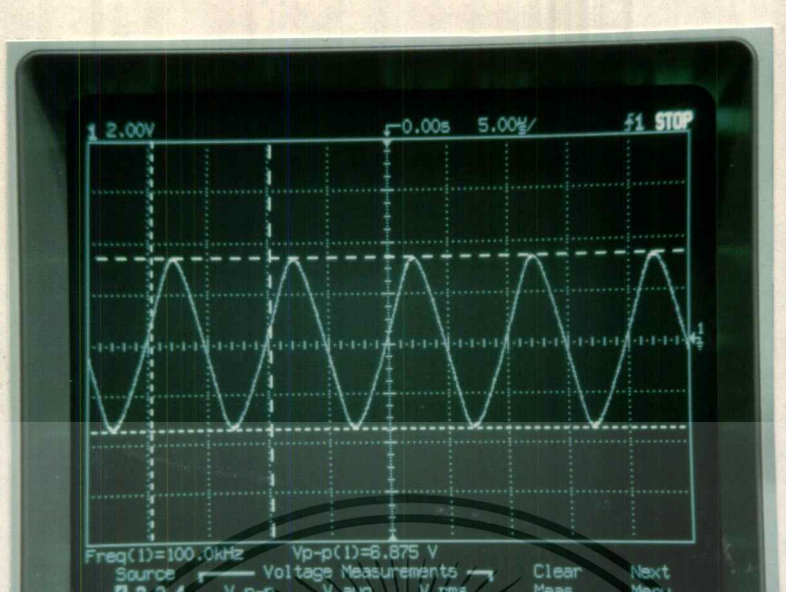


รูปที่ 4.9 สัญญาณซายน์ ที่ความถี่ 100 kHz แอมปลิจูด 10 V_{p-p}

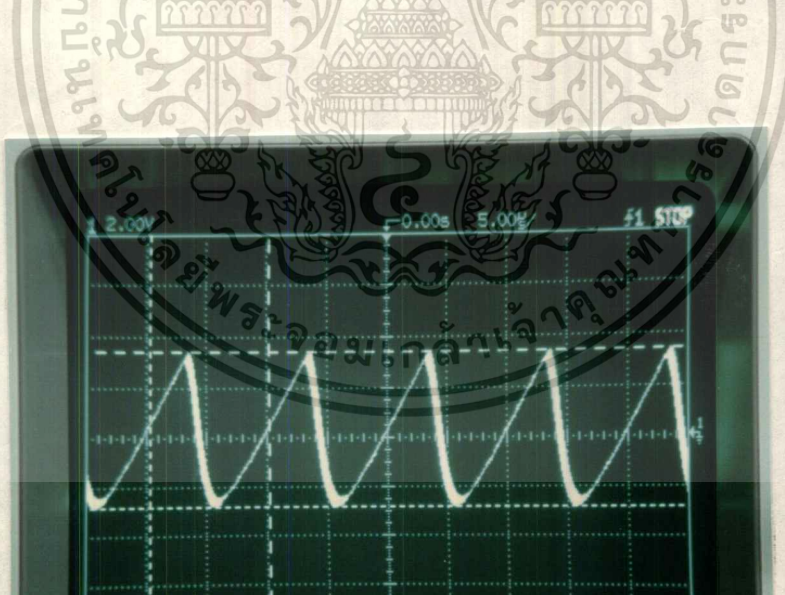


รูปที่ 4.10 สัญญาณสี่เหลี่ยม ที่ความถี่ 100 kHz แอมปลิจูด 10 V_{p-p}

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.11 สัญญาณสามเหลี่ยม ที่ความถี่ 100 kHz แอมปลิจูด 10 V_{p-p}



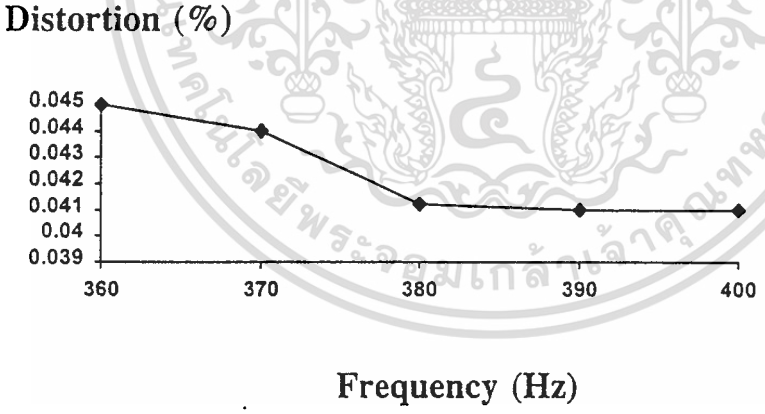
รูปที่ 4.12 สัญญาณฟันเลื่อย ที่ความถี่ 100 kHz แอมปลิจูด 10 V_{p-p}

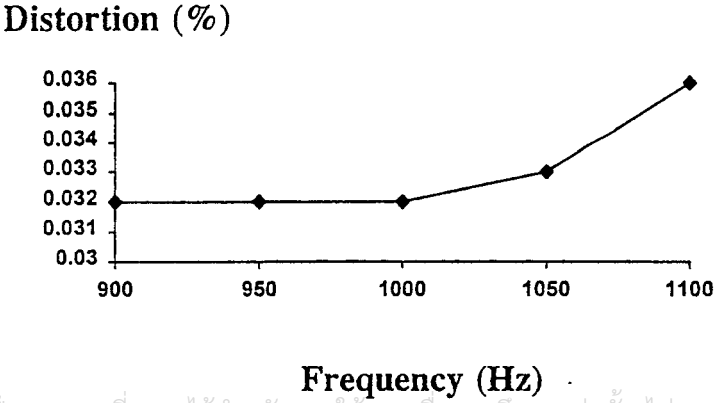
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. วัดค่าความเพี้ยนของสัญญาณซายน์ โดยการวัดเปอร์เซนต์ดิสทอร์ชั่น ต่อเครื่องฟังก์ชั่นเจเนอเรเตอร์เข้ากับ ดิสทอร์ชั่นมิเตอร์ (distortion meter) และทำการวัดค่าดิสทอร์ชั่น ที่ระดับสัญญาณ $1V_{p-p}$ โดยที่ดิสทอร์ชั่นมิเตอร์ที่ใช้ สามารถวัดค่าได้ในช่วงความถี่ 2ช่วง คือ ที่ 360-400 Hz และ 900Hz-1.1kHz แสดงผลการวัดดังตารางที่ 4.1

ตารางที่ 4.1 แสดงเปอร์เซนต์ดิสทอร์ชั่นของสัญญาณซายน์ที่ความถี่ต่างๆ

ความถี่ (Hz)	เปอร์เซ็นต์ดิสทอร์ชั่น
360	0.045
380	0.041
400	0.041
:	:
900	0.032
1K	0.032
1.1K	0.036





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. วัดค่าความเพี้ยนของสัญญาณสี่เหลี่ยม, สัญญาณสามเหลี่ยม และสัญญาณฟันเลื่อย โดยการวัดค่า rise/fall time และ duty cycle (%) แสดงผลการวัดดังตารางที่ 4.2

ตารางที่ 4.2 แสดงคุณสมบัติของสัญญาณทั้งสามชนิด

ชนิดของสัญญาณ	rise/fall time (μs)	duty cycle (%)
สัญญาณสี่เหลี่ยม	1.1	50.2
สัญญาณสามเหลี่ยม	-	50.3
สัญญาณฟันเลื่อย	1.2	-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทวิจารณ์ และสรุป

5.1 วิจารณ์ผลการทดลอง

จากผลการทดลองที่ได้ จากรูปถ่าย จะเห็นว่าที่ความถี่ต่ำๆ รูปสัญญาณจะไม่ค่อยผิดเพี้ยนนัก แต่ที่ความถี่สูงขึ้นจะเห็นว่ารูปสัญญาณจะเพี้ยนมากขึ้น และจากการวัดเปอร์เซ็นต์ดิสทอร์ชั่นของสัญญาณรูปซายน์เฉพาะในช่วงความถี่ที่ดิสทอร์ชั่นมิเตอร์สามารถวัดได้ (คือในช่วงความถี่ 360-400 Hz และ 900Hz-1.1kHz) ซึ่งอยู่ในย่านความถี่ต่ำ จะเห็นว่าเปอร์เซ็นต์ดิสทอร์ชั่นของสัญญาณอยู่ในช่วงประมาณ 0.03-0.05 ซึ่งถือว่าอยู่ในช่วงที่ใช้งานได้ แต่หากสามารถวัดดิสทอร์ชั่นในย่านความถี่สูงได้ คาดว่าอาจเกิดความผิดเพี้ยนมากขึ้น สำหรับความผิดเพี้ยนของสัญญาณรูปคลื่นสี่เหลี่ยม สามารถดูได้จากค่า rise/fall time และ duty cycle จะพบว่าสัญญาณสี่เหลี่ยมค่อนข้างเห็นความผิดเพี้ยนมากกว่าสัญญาณอื่น ส่วนสัญญาณรูปสามเหลี่ยมสามารถตรวจสอบความผิดเพี้ยนได้จากค่า duty cycle (%) และสัญญาณรูปฟันเลื่อยตรวจสอบได้จากค่า rise/fall time

5.2 สรุปผลการทดลอง

การสร้างสัญญาณต่างๆใช้การสังเคราะห์สัญญาณจากระบบดิจิตอลโดยตรง และ ใช้ไมโครคอนโทรลเลอร์ควบคุมส่วนต่างๆ เช่น แอลซีดี, คีย์กด รวมทั้งควบคุมการเลือกแอดเดรสของ EPROM ซึ่งเก็บข้อมูลของสัญญาณต่างๆเพื่อให้มีความถี่เอาท์พุทตามต้องการ

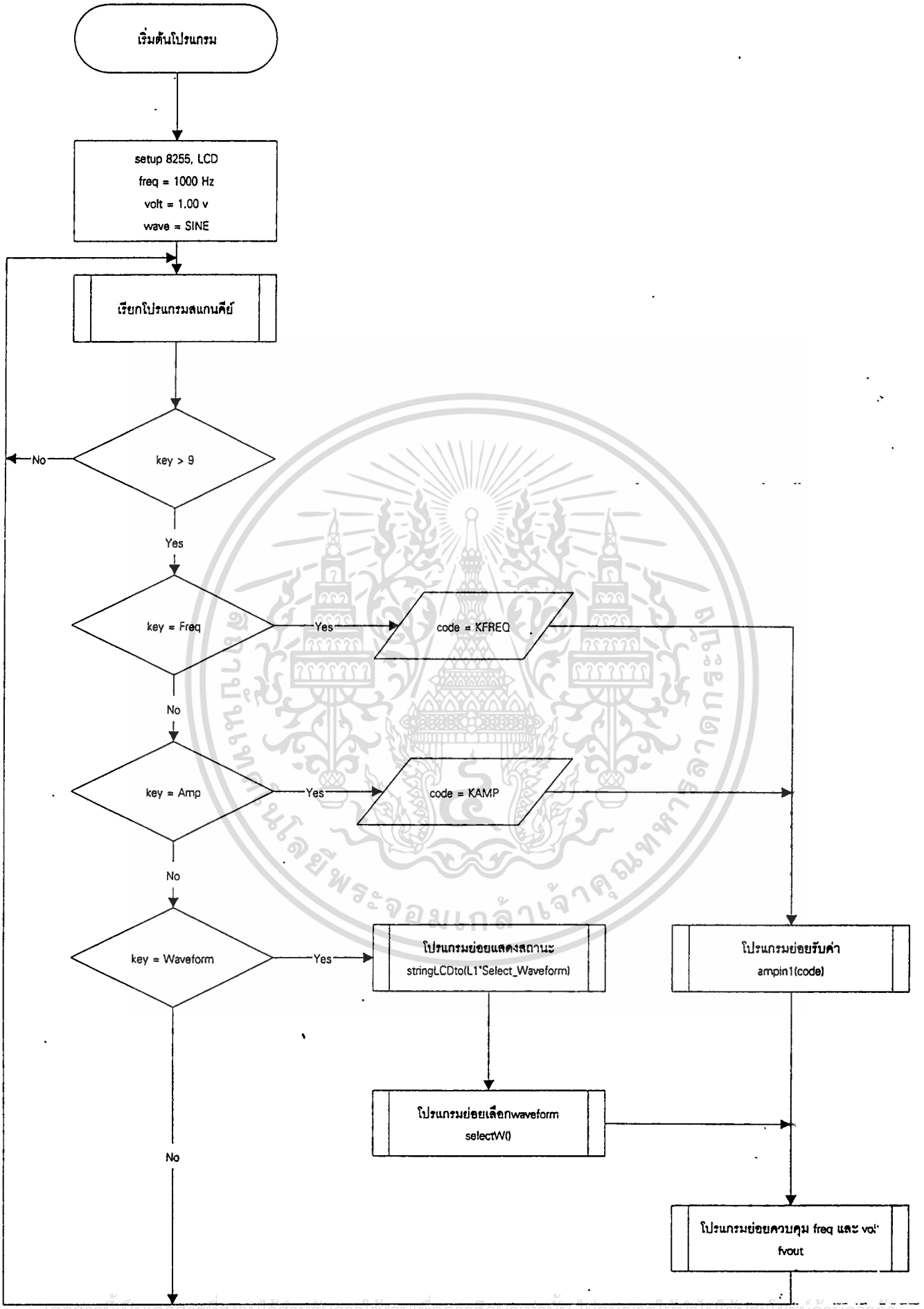
เมื่อไมโครคอนโทรลเลอร์เลือกแอดเดรสข้อมูล ในแต่ละค่าแอดเดรสจะเป็นค่าของสัญญาณ ซึ่งจะผ่านไปยังวงจรส่วนต่อๆไป เพื่อให้สัญญาณมีความสมบูรณ์ขึ้น

จากการทดลองค่าความถี่ที่ได้ ค่อนข้างแม่นยำ มีความคลาดเคลื่อนเพียงเล็กน้อยทุกค่าความถี่ ส่วนค่าแอมปลิจูดยังมีความคลาดเคลื่อนบ้าง มีความเพี้ยนของสัญญาณซายน์ต่ำ ในย่านความถี่ต่ำสัญญาณจะมีความเพี้ยนน้อย ในย่านความถี่สูง สัญญาณจะมีความเพี้ยนมากขึ้น ซึ่งเห็นได้ชัดในสัญญาณรูปสี่เหลี่ยม คือ มีรูปสัญญาณไม่คมชัด ในการเปลี่ยนแปลงระดับของสัญญาณจะมีความลาดเอียงอยู่มาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

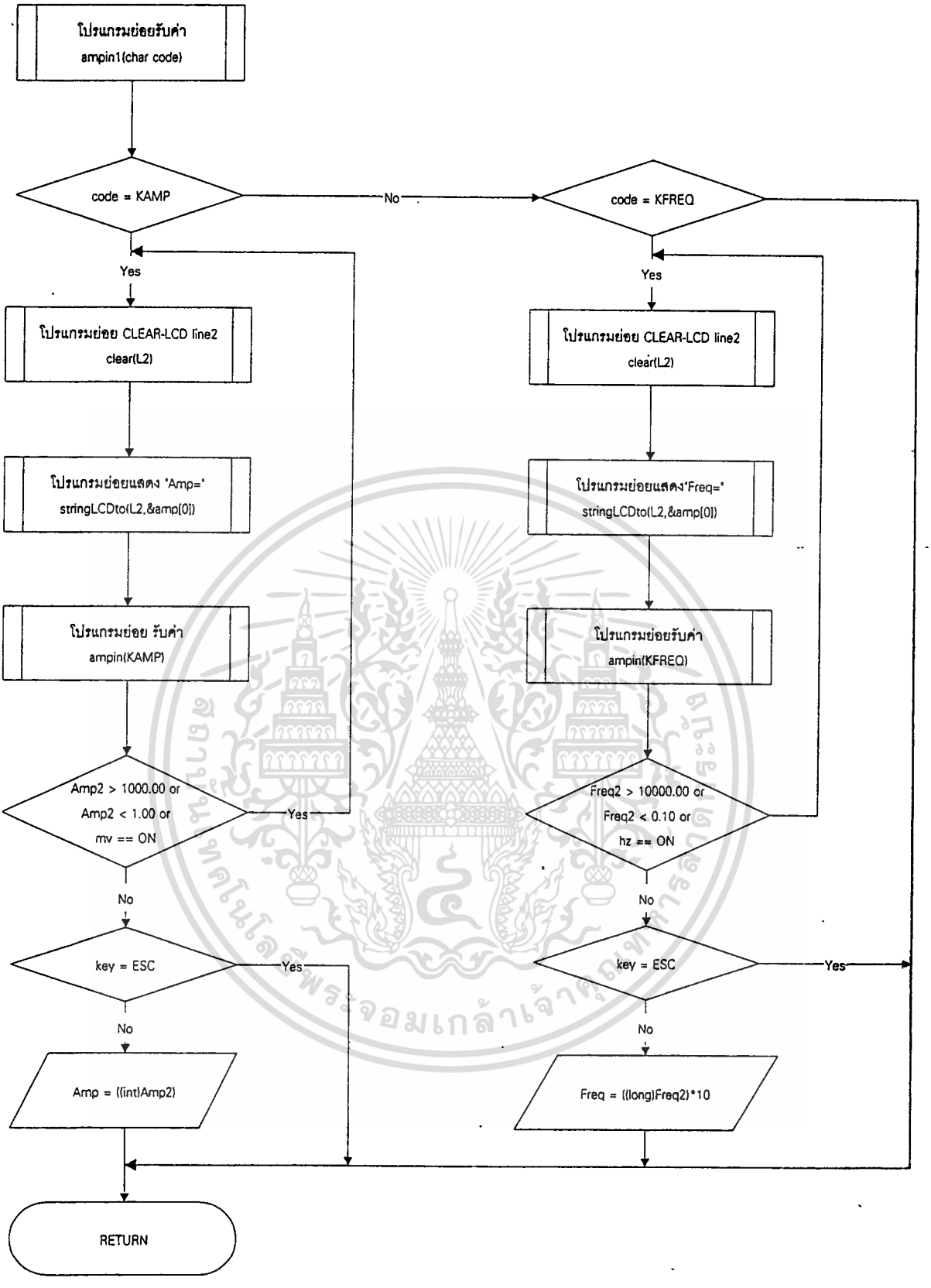
ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



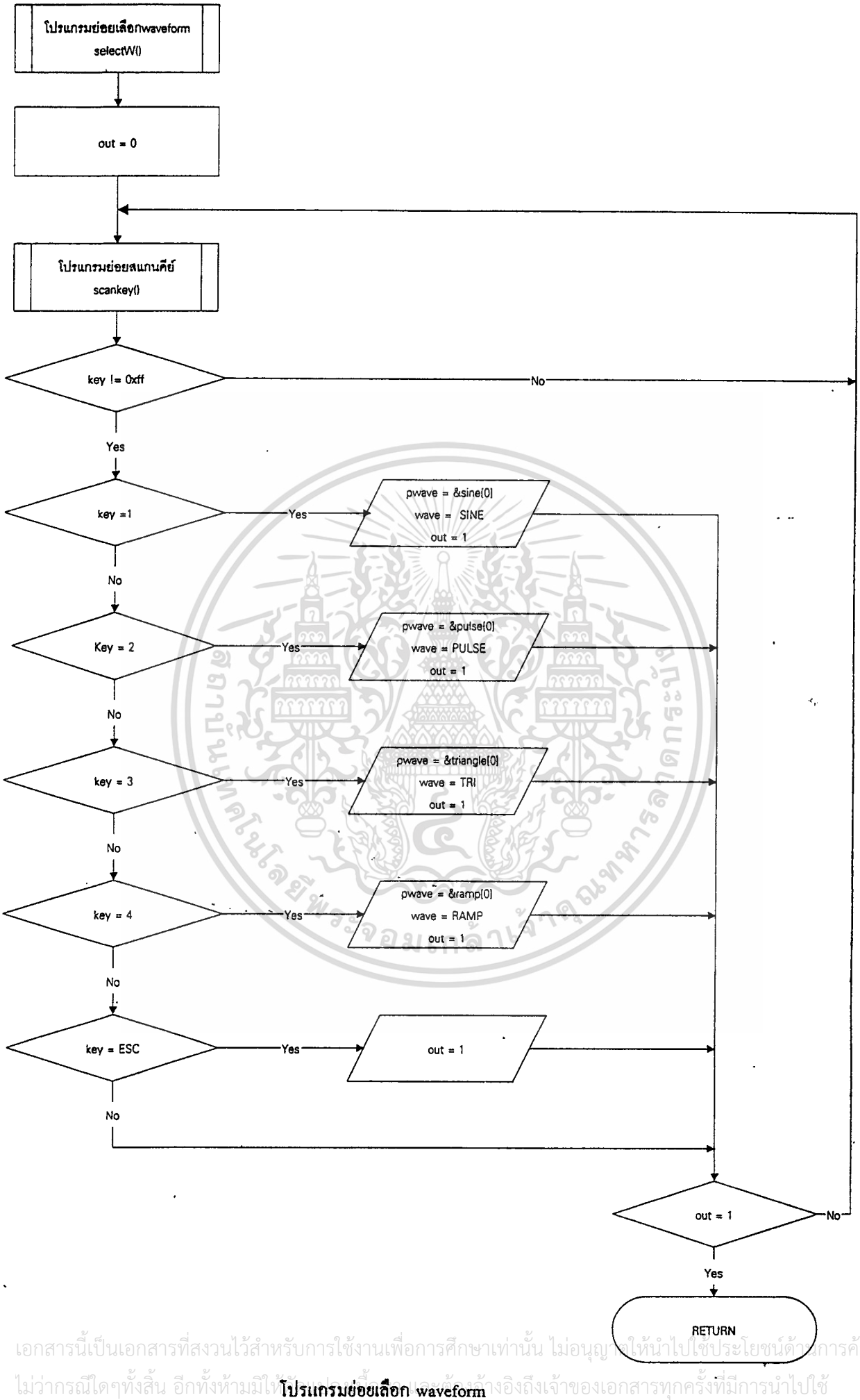
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมหลัก



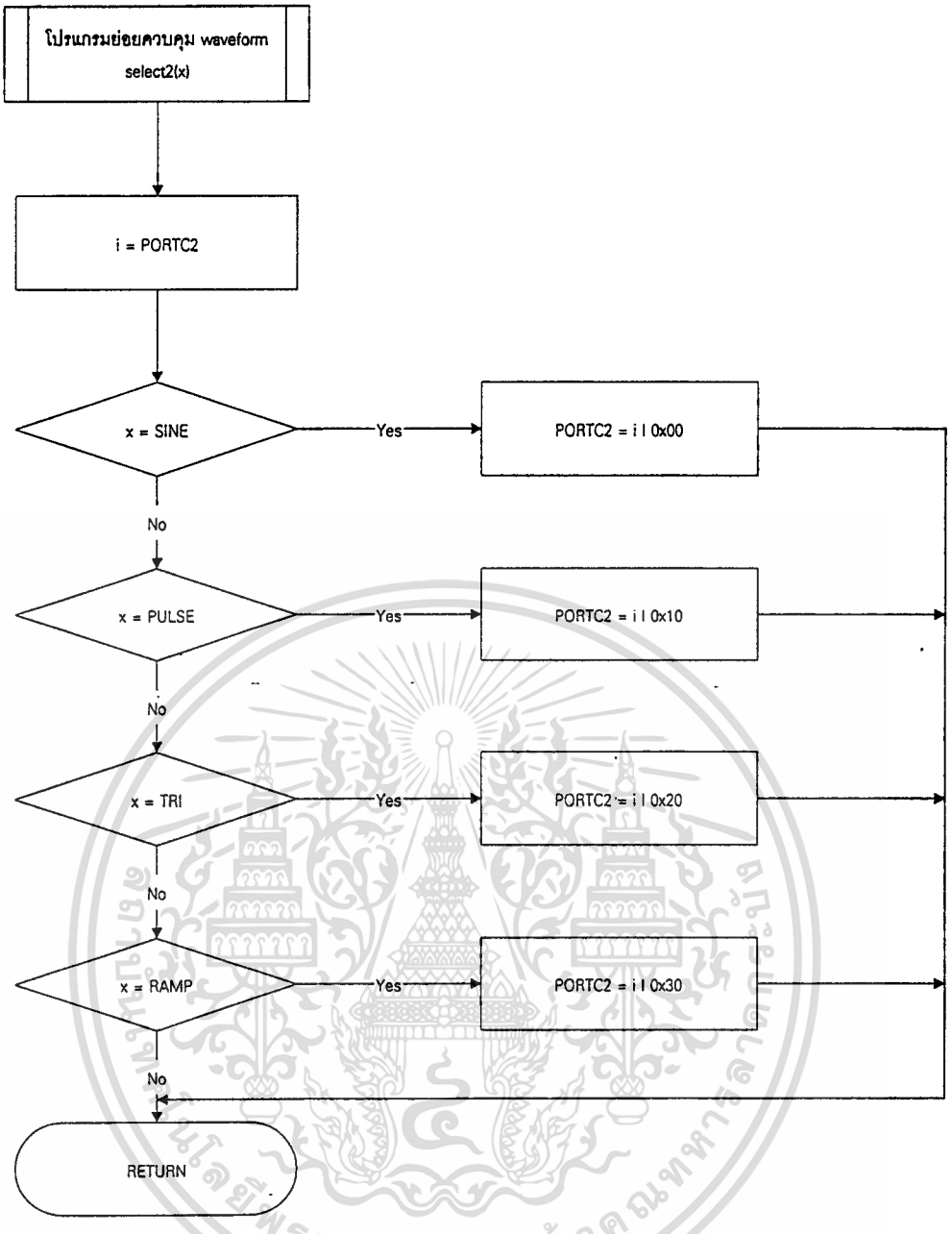
โปรแกรมย่อยรับค่า input

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



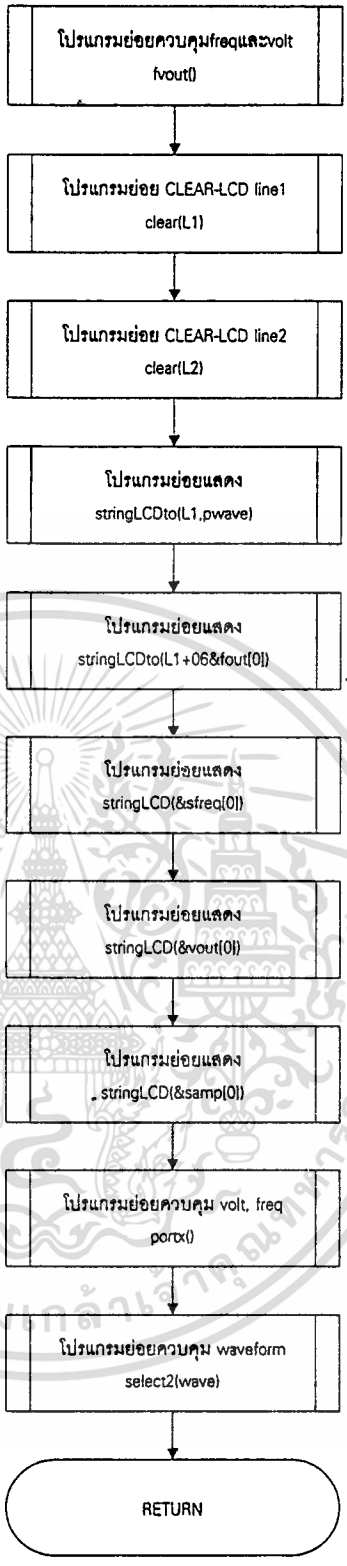
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมย่อยเลือก waveform



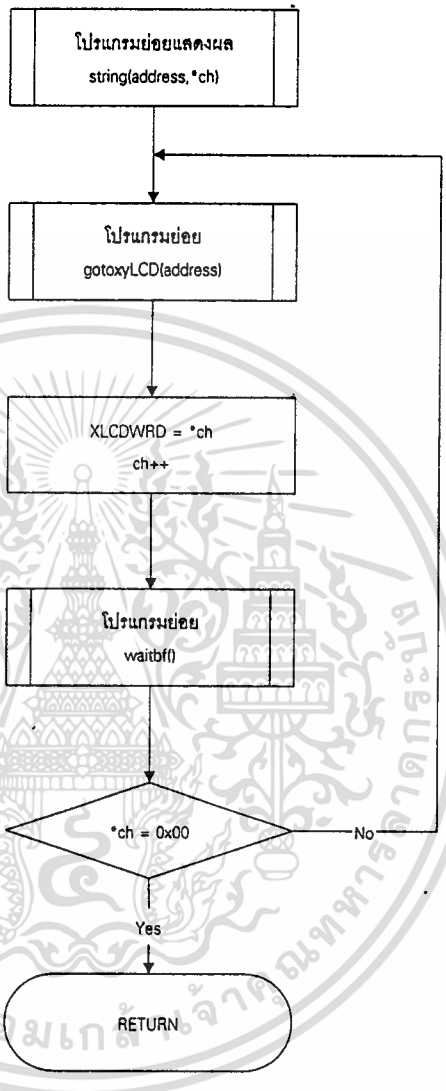
โปรแกรมย่อยควบคุม waveform

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



โปรแกรมย่อยส่วนควบคุม Frequency, Volt และ Waveform

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



โปรแกรมย่อยส่วนแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*--------------------------------------*/
/*      program function generator      */
/*--------------------------------------*/
#include  <8051.h>
#include  <intrpt.h>
#include  <stdlib.h>
#include  <stdio.h>
#include  <ant32.h>

#define   ON   1
#define   OFF  0
#define   L1   0x00
#define   L2   0x40

#define   SINE  0x0a
#define   PULSE 0x0b
#define   RAMP  0x0c
#define   TRI   0x0d

#define   KKHZ    0x0A
#define   KHZ     0x0B
#define   KUV     0x0C
#define   KV      KKHZ
#define   KMV     KHZ
#define   KCANCEL 0x0D
#define   KFREQ   0x0E
#define   KAMP    0x0F
#define   KWAVE   0x10
#define   KDOT    0x14
#define   KBP     0x15

#define   OK      0xFE
#define   ESC     0xFD
union Sbit{
  struct {
     unsigned B7:1;  /* the most significant bit */
     unsigned B6:1;
     unsigned B5:1;
     unsigned B4:1;
     unsigned B3:1;
     unsigned B2:1;
     unsigned B1:1;
     unsigned B0:1;
   }Bit;
   char allbit;
};
#define btmp (*(union Sbit *) 0x7F)

char key[]=
    { 0x74,
      0xE3,0xE4,0xE5,
      0xD3,0xD4,0xD5,
      0xB3,0xB4,0xB5,
      0XE6,0XD6,0xB6,0x76,
      0xE0,0xE1,0xE2,
      0xD0,0xD1,0xD2,
      0x73,0x75,
      0xB0,0xB1,0xB2,
```

```
        0x70,0x71,0x72
    };

char sine[]     = "Sine  ";
char pulse[]    = "Square";
char ramp[]     = "Ramp  ";
char triangle[] = "Triang";
char amp[]      = "AMP   ";
char freq[]     = "FREQ  ";
char timer[]    = "00:00:00";
char freqin[]   = "Freq In =";
char fout[]     = " =";
char vout[]     = " V=";
char select[]   = "Select Waveform";
char func[]     = "****Function****";
char gen[]      = "    Generator   ";
/*-------------------------------------------*/
void writeRTC();
void readRTC();
void sclkcom();
void sclkrw();
void delayRTC();
void rd1202();
void wr1202();
/*-------------------------------------------*/
void setupLCD();
void waitbf();
void gotoxyLCD(char address);
void stringLCD(char *ch);
void charLCD(char x);
void charLCDto(char address, char x);
void stringLCDto(char address,char *ch);
void clear(char line);
/*-------------------------------------------*/
void interrupt counter_isr(void);
void time_init();
void clock();
void outtimer(void);
/*-------------------------------------------*/
void selectW(void);
void select2(char x);
void select8255(void);
void ampin1(char code);
void ampin(char code);
void outportx();
/*------------ extern variable--------------*/
unsigned char keyin,kflag,cursor;
unsigned char tflag;
unsigned char hour,minute,sec,count;
char code,data;
/*------------ function generator------------*/
float Freq2,Amp2;
long  Freq,Amp;
char mv,khz,hz,volt;
char input[10];
char sfreq[10] ="1000Hz";
char samp[10]  ="1.00Vpp";
char stmp[10];
```

```
/*-------------------------------------------*/

/*[]=======================================[]*/
/*[]              Subroutine LCD           []*/
/*[]=======================================[]*/

void setupLCD()
{
    XLCDCOMN = 0x38;       /* LCD INITIALIZE */
    waitbf();              /* FUNCTION SET 2 LINE, 5*7 DOT, 8BIT
    XLCDCOMM = 0x0C;       /* DISPLAY ON/OFF */
    waitbf();
    XLCDCOMM = 0X06;       /* entry mode set, I/D=1, s=0 */
    waitbf();
    XLCDCOMM = 0X01;       /* clear display */
    waitbf();
}
void waitbf()
{
    asm("       push    dpl");
    asm("       push    dph");
    asm("       mov     dptr,#0FA01h");
    asm("rdy1:  movx    a,@dptr");
    asm("       jb      acc.7,rdy1");
    asm("       pop     dph");
    asm("       pop     dpl");
}
void gotoxyLCD(char address)
{
    btmp.allbit = address;
    btmp.Bit.B7 = ON;
    XLCDCOMM = btmp.allbit;
    waitbf();
}
void stringLCD(char *ch)
{
    for(;*ch != 0x00;) { /* *ch != nul */
        XLCDWRD = *ch;
        waitbf();
        ch++;
    }
}
void charLCD(char x)
{
     XLCDWRD = x;
     waitbf();
}
void charLCDto(char address,char x)
{
     gotoxyLCD(address);
     charLCD(x);
}
void stringLCDto(char address,char *ch)
{
     gotoxyLCD(address);
     for(;*ch != 0x00;) {/* *ch != nul */
         XLCDWRD = *ch;
         waitbf();
```

```
            ch++;
        }
    }
    void clear(char line)
    {
        char i;
        gotoxyLCD(line);
        for(i=0;i<20;i++) {
           XLCDWRD = ' ';
           waitbf();
        }
    }
    /*[]----------------------[]*/
    /*[]      Scan key        []*/
    /*[]----------------------[]*/
    void scankey(void)
    {
        /* use register r3,r4,r5,r6,dptr,a */

        asm("        mov     r3,#00");
        asm("        mov     r4,#07");
        asm("        mov     r5,#07");
        asm("        mov     dptr,#_keyin");
        asm("        mov     a,#0ffh");
        asm("        movx    @dptr,a");
        asm("scank1: lcall    scans");
        asm("        mov     dptr,#0f802h");
        asm("        movx    a,@dptr");
        asm("        anl     a,#0f0h");
        asm("        mov     r6,a");
        asm("        cjne    a,#0f0h,scank3");
        asm("        djnz    r5,scank2");
        asm("        mov     dptr,#_kflag");
        asm("        mov     a,#00");
        asm("        movx    @dptr,a");
        asm("scank2: inc      r3");
        asm("        djnz    r4,scank1");
        asm("        ljmp    escank");
        asm("scank3:         ");
        asm("        mov     a,r6");
        asm("        orl     a,r3");
        asm("        mov     r6,a");
        asm("        mov     r2,#0");
        asm("        mov     dptr,#_key");
        asm("scank4: movx     a,@dptr");
        asm("        xrl     a,r6");
        asm("        jz      scank5");
        asm("        inc     dptr");
        asm("        inc     r2");
        asm("        cjne    r2,#28,scank4");
        asm("scank5: mov      a,r2");
        asm("        mov     r6,a");
        asm("        mov     dptr,#_kflag");
        asm("        movx    a,@dptr");
        asm("        cjne    a,#01,scank6");
        asm("        ljmp    scank2");
        asm("scank6:         ");
        asm("        mov     a,#01");
```

```
    asm("        movx    @dptr,a");
    asm("        mov     dptr,#_keyin");
    asm("        mov     a,r6");
    asm("        movx    @dptr,a");
    asm("        ljmp    scank2"(;
/*--------------- Scans -------------------*/
    asm("scans: mov      dptr,#0f802h");
    asm("        mov     a,#0f0h");
    asm("        orl     a,r3");
    asm("        movx    @dptr,a");
    asm("        ret");
    asm("escank:         ");
}
void ampin1(char code)
{
     char i;
     if(code == KAMP)   {
        do {
       clear(L2);
       stringLCDto(L2,&amp[0]);
       ampin(KAMP);
        }while(Amp2 > 10.00 || Amp2 < 1.00 || mv == ON);
        if(keyin != ESC) {
      Amp = ((int)Amp2);
      for(i=0;i<10;i++) samp[i]=input[i];
        }
     }else {
        do {
       clear(L2);
       stringLCDto(L2,&freq[0]);
       ampin(KFREQ);
        }while(Freq2 > 10000.00 || Freq2 < 0.10 || hz==ON);
        if(keyin != ESC) {
       Freq2   = Freq2*10.00;
       Freq=((long)Freq2);
       for(i=0;i<10;i++) sfreq[i]=input[i];
        }
     }
     fvout();
}
void ampin(char code)
{
    char out,*p,*p2,i,dot;
    out=dot=hz=mv=OFF;
    p = &input[0];
    for(i=0;i<8;i++) { *p = 0x00; p++; }
    p = &input[0];
    if (code == KAMP) i=5;
    else i=6;           /* limit amount key */
    XLCDCOMM = 0x0F   /* show cursor display LCD */
    waitbf();
    do{
       scankey();
       if(keyin != 0xff) {
         if(keyin < 0x0a && p != &input[i]  {
             charLCD(keyin+0x30);
             *p = keyin+0x30;
             p++;
```

```
            } /* switch(inkey) */
          } /* keyin != 0xff */
        } while(out != ON);
        XLCDCOMM = 0x0c; /* off cursor display */
        waitbf();
}
void setup8255(void)
{
    CPORT1 = 0x88;
    CPORT2 = 0x80;
    PORTA2 = 0xE8;
    PORTB2 = 0x03;
    PORTC2 = 0x00;
}
void selectW(void)
{
    char out;
    out = OFF;
    do {
        scankey();
        if(keyin != 0xff) {
         switch(keyin) {
        case 0x01:
               pwave = &sine[0];
               wave = SINE;
               out = ON;
               break;
        case 0x02:
               pwave = &pulse[0];
               wave = PULSE;
               out = ON;
               break;
        case 0x03:
               pwave = &triangle[0];
               wave = TRI;
               out = ON;
               break;
        case 0x04:
               pwave = &ramp[0];
               wave = RAMP;
               out = ON;
               break;
        case KCANCEL:
               out = ON;
               break;
         } /* switch */
        }
    }while(out != ON);
    fout();
}
void select2(char x)
{
    unsigned char i;
    i = PORT2;
    switch(x) {
        case SINE:   PORTC2 = i | 0x00; break:
        case PULSE:  PORTC2 = i | 0x10; break;
        case TRI:    PORTC2 = i | 0x20; break;
```

```
} else switch(keyin) {
case KCANCEL: keyin = ESC;
       out = ON;
       clear(L2);
       break;
case KBP: if(p != &input[0]) {
         p--;
         *p = 0x00;
         cursor = XLCDBUSY;
         cursor = (cursor & 0x7f)-1;
         gotoxyLCD(cursor);
         charLCD(0x20);
         gotoxyLCD(cursor);
       }
      break;
case KDQT: if(dot != ON) {
          *p = '.';
          charLCD('.');
          p++;
          dot = ON;
        }
       break;
case KV:   if(code == KAMP) {
          if (input[4] != 0x00) {
          input[4] = 0x00;
          p = &input[4];
          }
          Amp2 = atof(&input[0])*100;
          *p = 'V';p++;*p='p';p++;*p='p';
          volt =KV;
        }
        else{
          Freq2 = atof(&input[0])*100;
          *p='K';p++;*p='H';p++;*p'z';
          khz = ON;
        }
        keyin = OK;
        out = ON;
        break;
case KMV:  if(code == KAMP) {
          if (input[3] != 0x00) {
          input[3] = 0x00;
          p= &input[3];
          }
          Amp2 = atof(&input[0]);
          *p ='m';p++;*p='V';p++;*p='p';
          mv = dot;
          volt=KMV;
        }
        else {
          Freq2 = atof(&input[0])/10;
          *p ='H';p++;*p='Z';
          hz = dot;
          khz = OFF;
        }
        keyin = OK;
        out = ON;
        break;
```

```
            case RAMP:    PORTC2 = i | 0x30; break;
        }
        i = PORTB;
        if(volt == KV)
          PORTB = i | 0x00;  /* 0.00V - 9.99V */
        else
          PORTB = i | 0x10;  /* 000mV - 999mV */
    }
    void outportx()
    {
        asm("       mov     dptr,#_Amp+3");
        asm("       movx    a,@dptr");
        asm("       mov     dptr,#0f800h");
        asm("       movx    @dptr,a");
        asm("       mov     dptr,#_Amp+2");
        asm("       movx    a,@dptr");
        asm("       mov     dptr,#0f801h");
        asm("       movx    @dptr,a");
        asm("       mov     dptr,#_Freq+3");
        asm("       movx    a,@dptr");
        asm("       mov     dptr,#0fC00h");
        asm("       movx    @dptr,a");
        asm("       mov     dptr,#_Freq+2");
        asm("       movx    a,@dptr");
        asm("       mov     dptr,#0fC01h");
        asm("       movx    @dptr,a");
        asm("       mov     dptr,#_Freq+1");
        asm("       movx    a,@dptr");
        asm("       mov     dptr,#0fC02h");
        asm("       movx    @dptr,a");
    }
    fout()
    {
        clear(L1);
        clear(L2);
        stringLCDto(L1,pwave);
        stringLCDto(L1+0x06,&fout[0]);
        stringLCD(&sfreq[0]);
        stringLCDto(L2+0x06,&vout[0]);
        stringLCD(&samp[0]);
        outportx();
        select2(wave);
    }

    main()
    {
        unsigned int i,j;

        P1_BITS.B6 = 0;
        P1_BITS.B5 = 1;
        hour = minute = sec = count = 0;
        keyin = 0xff;
        setup8255();
        setupLCD();
        Amp2  = Amp = 1.00;      /* 1.00 V */
        Freq2 = Freq = 1000;    /* 1000 Hz */
        volt  = KV;
        pwave = &sine[0];
```

```
        wave = SINE;
        stringLCDto(L1,&func[0]);
        stringLCDto(L2,&gen[0]);
        for(i=0;i<5;i++);
            for(j=0;j<30000;j++);

        fout();
        for(;;) {
            scankey();
            if(keyin != 0xff && keyin > 0x09) {
             switch(keyin) {
           case KWAVE: stringLCDto(L1,&select[0]);
                   selectW();
                   fout();
                   break;
           case KFREQ: ampin1(KFREQ);
                   outportx();
                   select2(wave);
                   break;
           case KAMP:  ampin1(KAMP);
                   outportx();
                   select2(wave);
                   break;
             } /* switch */
            }
        }
}
```

เอกสารอ้างอิง

1] Arthur H. Seidman, *Integrated Circuit Applications Handbook,* Seidman

2] Matthew Mahoney, *DSP-Based Testing of Analog and Mixed-Signal Circuits,* The Computer Society of The IEEE, 1987

3] บริษัท ศิลารีเสิร์ช จำกัด, *คู่มือการใช้งาน ANT-32 V3.0 USER MANUAL*

4] สุนทร วิทูสุรพจน์, *การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051,* บริษัทซีเอ็ดยูเคชั่น จำกัด(มหาชน), 2537

5] สมควร เมืองรมย์, ดิษฐกรณ์ กันเรียน , *โครงงานเครื่องกำเนิดสัญญาณซายน์ระบบดิจิตอล,* เซมิคอนดัคเตอร์ อิเลคทรอนิคส์, ฉบับที่ 114, 2535

6] มัณฑนา ปราการสมุทร, *การเขียนชุดคำสั่งภาษาซี,* ไฮเทคพรินติ้ง, 2534

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้