ตู้ไฟ LED 3 มิติประกอบเสียงเพลง
3D LED LIGHT MUSIC BOX

ปรัชญา แก้วหินลาย
PRACHYA KAEWHINLAI

ปิยวัฒน์ ฉิมทิม
PHIYAWAT CHIMTIM

อดิพงษ์ คันธนาสินชัย
ADIPONG KANTANASINCHAI

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2555

ตู้ไฟ LED 3 มิติประกอบเสียงเพลง

3D LED LIGHT MUSIC BOX

ปรัชญา แก้วหินลาย
PRACHYA KAEWHINLAI

ปิยวัฒน์ ฉิมทิม
PHIYAWAT CHIMTIM

อดิพงษ์ คันธนาสินชัย
ADIPONG KANTANASINCHAI

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2555

ปริญญานิพนธ์ปีการศึกษา 2555

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ตู้ไฟ LED 3 มิติประกอบเสียงเพลง

3D LED LIGHT MUSIC BOX

ผู้จัดทำ นาย ปรัชญา แก้วหินลาย รหัสประจำตัว 52010684

นาย ปิยวัฒน์ ฉิมทิม รหัสประจำตัว 52010724

นาย อดิพงษ์ คันธนาสินชัย รหัสประจำตัว 52011370

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....

(ดร.อิทธิภูมิ บุญพิคำ)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์ ตู้ไฟ LED 3 มิติประกอบเสียงเพลง
นักศึกษา นาย ปรัชญา แก้วหินลาย รหัสประจำตัว 52010684
นาย ปิยวัฒน์ ฉิมทิม รหัสประจำตัว 52010724
นาย อดิพงษ์ คันธนาสินชัย รหัสประจำตัว 52011370
ปริญญา วิศวกรรมศาสตร์บัณฑิต
สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์
ปีการศึกษา 2555
อาจารย์ที่ปรึกษาปริญญานิพนธ์ **ดร.**อิทธิภูมิ บุญพิคำ

บทคัดย่อ

ปริญญานิพนธ์ “ตู้ไฟ LED 3 มิติประกอบเสียงเพลง” นี้ แสดงตู้ไฟ LED 3 มิติหรือรูปลูกบาศก์ $5\times5\times5$ (รวม 125 LED) ที่แสดงรูปแบบการกระพริบของไฟตามจังหวะเพลง

ลูกบาศก์ LED นี้ประกอบด้วย 125 LED ที่จัดเรียงตัวกัน 5 ชั้น โดยแต่ละชั้นประกอบด้วย 25 LEDส่วนแสดงผลนี้จะต่อรวมเข้าด้วยกัน คือแทนที่จะใช้ 125 การเชื่อมต่อแต่ใช้ 30 การเชื่อมต่อแทน นั่นคือ 1 การเชื่อมต่อสำหรับชั้น LED ทั้ง 5 และ 25 การเชื่อมต่อระหว่าง 25 LED ในแต่ละชั้น

สำหรับส่วนอินพุทและเอาท์พุท(I/O) ใช้ 8 I/O เท่านั้นเพื่อควบคุมตัวขับ LED เหล่านี้ จากขา 14 ขาของไมโครคอนโทรลเลอร์ PIC 16F688 ที่ได้รับการโปรแกรมเพื่อควบคุมการปิดเปิด LED

สำหรับส่วนควบคุม LED Cube นี้ให้แสดงผลตามเสียงเพลงใช้วงจรแบนพาสฟิลเตอร์แยกความถี่เสียงทุ้มและแหลมเพื่อนำไปคุมไฟแต่ละดวงซึ่งมีดวงละสองสีโดยทำงานร่วมกับสัญญาณจากไมโครคอนโทรลเลอร์

Thesis Title	3D LED LIGHT MUSIC BOX
Student	Mr. Prachya Kaewhinlai Student ID.52010684
	Mr. Phiyawat Chimtim Student ID.52010724
	Mr. Adipong Kantanasinchai Student ID.52011370
Degree	Bachelor of Engineering
Program	Electronics Engineering
Year	2012
Thesis Advisor	Dr.Ittibhoom Boonphikum

ABSTRACT

"3D LED LIGHT MUSIC BOX" displays 3D LEDs or a $5 \times 5 \times 5$ cube (125 LEDs total) showing light animation patterns according to rhythms of music

The LEDs cube is made up from 125 LEDs arranged into 5 layers of 25 LEDs. These LED are multiplexed so instead of requiring 125 connections, it requires only 30 connections, one to each of the 5 layers and 25 LEDs in a layer.

As for I/O, only 8 I/O lines are needed to control these LEDs using 14 pin PIC 16F688 microcontrollers to control on/off LEDs

For control LED Cube to show as a music using band pass filters separate bass and treble frequency to control LED Each LED has two colors that it works with signals from the microcontroller.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ทั้งนี้เพราะได้รับคำแนะนำและคำปรึกษาจาก ดร. อิทธิภูมิ บุญพิคำ ซึ่งเป็นอาจารย์ที่ปรึกษาในการทำปริญญานิพนธ์ ในครั้งนี้ ผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

นอกจากนี้ขอขอบพระคุณทุกๆท่านที่ให้ความช่วยเหลือ ตลอดจนให้คำแนะนำต่างๆจนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จโดยสมบูรณ์

คุณค่าและประโยชน์อันพึงมีในรายงานฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

ปรัชญา แก้วหินลาย
ปิยวัฒน์ ฉิมทิม
อดิพงษ์ คันธนาสินชัย

สารบัญ

สารบัญรูปภาพ, ตารางและกราฟ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบันโลกเราได้มีการประยุกต์เทคโนโลยีเกี่ยวกับหลอดไฟ LED ไปในรูปแบบต่างๆ ทั้งด้านคมนาคม ด้านโฆษณา และด้านความสวยงาม จากการที่คณะผู้จัดทำโครงงานได้มีโอกาสไปฝึกงานที่บริษัทที่เกี่ยวข้องกับผลิตภัณฑ์เกี่ยวกับหลอด LED จึงได้เล็งเห็นถึงโอกาสการเจริญเติบโตของธุรกิจและผลิตภัณฑ์เกี่ยวกับสินค้าจากหลอด LED คณะผู้จัดทำโครงงานจึงมีความเห็นที่จะนำหลอดไฟ LED มาประยุกต์ใช้งานทำเป็นตู้ไฟทรงสี่เหลี่ยมจัตุรัส และเพื่อพัฒนารูปแบบลักษณะโคมไฟตกแต่งที่มีการใช้งานในปัจจุบัน และสามารถออกแบบไฟให้แสดงผลออกมาเป็นรูปร่างหรือลายลักษณ์อักษร เป็นภาพเคลื่อนไหวตามจังหวะเสียงดนตรีเพื่อสร้างรูปแบบใหม่ๆ ในการสร้างตู้ไฟประดับตกแต่ง

1.2 วัตถุประสงค์

- เพื่อให้ได้ตู้ไฟประดับตกแต่งที่ควบคุมการแสดงผลโดยการเคลื่อนไหวตามเสียงดนตรี
- เพื่อพัฒนาตู้ไฟประดับตกแต่งจากรูปแบบต่างๆที่มีอยู่ในปัจจุบัน
- เพื่อเป็นการศึกษาประโยชน์ของหลอดไฟ LED โดยการประยุกต์ใช้นำมาสร้างสีสันสวยงาม
- เพื่อศึกษาการเขียนโปรแกรมไมโครคอนโทรลเลอร์ในการนำมาใช้ควบคุม LED Cube
- เพื่อสร้างรูปแบบผลิตภัณฑ์ใหม่ๆที่เกิดจากหลอด LED ซึ่งมีโอกาสขยายการเจริญเติบโต การพัฒนาทางด้านแสงสีประดับตกแต่ง

1.3 ขอบเขตของโครงงาน

จากที่ได้สร้างตู้ไฟ LED สี่เหลี่ยมจัตุรัสขนาด 5x5x5 ซึ่งประกอบด้วยส่วนของวงจรและส่วนของโปรแกรมควบคุมรูปแบบการแสดงไฟ โปรแกรมควบคุมอยู่ในรูป HEX ไฟล์ โดยส่วนอินพุทและเอาท์พุท (I/O) ใช้ 8 I/O เท่านั้นเพื่อควบคุมตัวขับ LED จากขา 14 ขาของไมโครคอนโทรลเลอร์ PIC 16F688 ที่ได้รับการโปรแกรมเพื่อควบคุมการปิดเปิด LED ในปริญญานิพนธ์นี้จึงนำตู้ไฟดังกล่าวไปทำการออกแบบการควบคุมการแสดงผลจากเสียงเพลงโดยใช้วงจรแบนพาสฟิลเตอร์แยกสัญญาณเสียงต่ำและสูงเพื่อนำมาประยุกต์คุมการแสดงผลของหลอดไฟสองสีเพื่อให้ตอบสนองต่อความถี่สองความถี่ดังกล่าว

1.4 ผลที่คาดว่าจะได้รับ

1. นำมาประยุกต์เพื่อใช้งานเพื่อออกแบบตู้ไฟ โคมไฟที่ไปประดับตามร้านอาหาร
2. สามารถนำไปใช้งานได้จริงภายในที่พักอาศัยและร้านค้าและร้านอาหารที่มีเสียงเพลงในการตกแต่งให้สวยงาม
3. ความรู้และทักษะที่เกิดขึ้นกับผู้จัดทำโครงการ ซึ่งมาจากการปฏิบัติงาน
4. ได้รับความรู้ในการศึกษาการทำงานของไมโครคอนโทรลเลอร์และการเขียนโปรแกรม
5. เป็นต้นแบบในการพัฒนาประโยชน์ของ LED และไมโครคอนโทรลเลอร์ต่อไป

1.5 โครงสร้างของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้จะได้นำเสนอถึง ทฤษฎีความรู้พื้นฐานที่เกี่ยวข้อง การออกแบบสร้างและควบคุมการทำงานของ LED Cube 5x5x5 โดยใช้การควบคุมจากไมโครคอนโทรลเลอร์และส่วนวงจรควบคุมหลอด LED ให้มีการตอบสนองตามเสียงเพลง โดยแบ่งเป็นบทต่างๆ ดังนี้

บทที่ 1. บทนำ
บทที่ 2. ทฤษฎีความรู้ที่เกี่ยวข้อง
บทที่ 3. การออกแบบและการประกอบ
บทที่ 4. การทดลองและผลการทดลอง
บทที่ 5. สรุปและวิจารณ์ผลการทดลอง

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

2.1 รูปแบบของตู้ไฟ LED 3 มิติประกอบเสียงเพลง

ไฟ LEDs 3 มิติหรือรูปลูกบาศก์ 5×5×5 (รวม 125 LED) ส่วนแสดงผลนี้จะต่อรวมเข้าด้วยกันกับส่วนควบคุม คือแทนที่จะใช้ 125 การเชื่อมต่อแต่ใช้ 30 การเชื่อมต่อแทน นั่นคือ 1 การเชื่อมต่อสำหรับชั้น LEDs ทั้ง 5 และ 25 การเชื่อมต่อระหว่าง 25 LEDs ในแต่ละชั้น

สำหรับส่วนอินพุทและเอาท์พุท(I/O) ใช้ 8 I/O เท่านั้นเพื่อควบคุมตัวขับ LED (STP16DP05) จากขา 14 ขาของไมโครคอนโทรลเลอร์ PIC 16F688 ที่ได้รับการโปรแกรมเพื่อควบคุมการปิดเปิด LED ให้เป็นไปตามรูปแบบของเพลง ไมโครคอนโทรลเลอร์นี้ให้สัญญาณนาฬิกาภายใน 8 MHz และความจุ 4K คำ

สำหรับส่วนควบคุม LED Cube นี้ให้แสดงผลตามเสียงเพลงใช้วงจรแบนพาสฟิลเตอร์แยกความถี่เสียงทุ้มและแหลมเพื่อนำไปคุมไฟแต่ละดวงซึ่งมีดวงละสองสีโดยทำงานร่วมกับสัญญาณจากไมโครคอนโทรลเลอร์

2.2 ภาษาแอสแซมบลี้ (Assembly Language)

ภาษาแอสแซมบลี้ เป็นภาษาที่ใช้สัญลักษณ์ในการสื่อความหมาย ภาษาแอสแซมบลีมีลักษณะคำสั่งที่ขึ้นกับเครื่อง คอมพิวเตอร์ที่ใช้งานและมีการแปลคำสั่งให้เป็นภาษาเครื่อง นอกจากภาษาเครื่อง และภาษาแอสแซมบลีแล้ว ยังมีภาษาระดับสูง เช่น Basic Cobol FORTRANซึ่งเป็นภาษา ที่มีคำสั่งใกล้เคียงกับภาษาอังกฤษมาก ทำให้ผู้เขียนโปรแกรม สามารถเขียนโปรแกรมได้สะดวกและรวดเร็วแต่ว่า โปรแกรมที่เขียนด้วยภาษาระดับสูงต้องใช้เนื้อที่เก็บในหน่วยความจำเป็นจำนวนมากอีกทั้งทำงานได้ช้ากว่า ภาษาแอสแซมบลีดังนั้นภาษาระดับสูงจึงไม่นิยมนำมาประยุกต์ใช้กับการทำงาน ที่ระบบการควบคุม ที่มีความสำคัญมาก ภาษาแอสแซมบลี เหมาะกับโปรแกรมที่ใช้เนื้อที่ในหน่วยความจำไม่มากนัก ทั้งทำงานได้รวดเร็วและ ในการควบคุม การทำงานของเครื่องคอมพิวเตอร์ได้โดยตรง

2.2.1 การควบคุมส่วนแสดงผล LED Cube

วิธี Data lookup table เพื่อขับ LED cube คือดูจำนวนข้อมูลที่ต้องการ หาข้อมูลนี้ถูก packed แล้ว ต้องใช้ 16 bytes เพื่อรักษาสถานะของ cube ไว้(125/8=15.625) เนื่องจาก 16F688 PIC มีหน่วยความจำโปรแกรมเพียง 4K words และรหัสหลักใช้ประมาณ 1455 คำ ทำให้เหลือประมาณ 2600 คำสำหรับข้อมูลตาราง หรือมีที่ว่างสำหรับสถานนะ cube ที่สมบูรณ์ 168 ตัวที่จะเก็บไว้ในหน่วยความจำได้ ดังนั้นหากเราต้องการแสดงคำว่า “PROJECTKMITL” ให้เคลื่อนจากหลังไปด้านหน้าของ cube นี้ จะต้องใช้ 16 bytesX5 frames ต่อตัวอักษรX 11 ตัวอักษร= 880 bytes

เป็นที่ขัดเจนแล้วว่าวิธี table lookup นั้นไม่ได้ให้เราทำอะไรมากนักก่อนที่จะไม่มีที่ว่างโปรแกรมในหน่วยความจำ เพื่อแก้ปัญหานี้ เราควบคุม cube โดยใช้ภาษาคำสั่ง macro ที่ควบคุม virtual drawing processor โดยใช้รหัสที่เขียนด้วยคำสั่ง macro เพื่อกำหนดให้กับ cube drawing processor ให้มีการแสดงผลเช่นเดียวกันใน 60 byte.

คำสั่งสร้างขึ้นโดยใช้ assembler Macros, ด้วยเหตุนี้คำสั่ง mnemonics ตั้งต่างไปจากกลุ่มคำสั่ง PIC และ MPASM assembler reserved words. คำสั่งชื่อ Macro และชื่อ register เป็นตัวอักษรใหญ่และแตกต่างกันสำหรับตัวอักษรใหญ่และตัวอักษรเล็กด้วย คำสั่ง Macro จะถูกแปลงเป็นคำสั่ง retlw PIC assembler 1 หรือ 2 ชุด ที่เวลา assembly เมื่อ คำสั่ง PIC LED cube ทำการอ่านกลับตามข้อมูลตาราง จะถอดรหัสคำสั่งตาม bit-fields และเรียกหน้าที่ของ assembler เพื่อดำเนินการการทำงานของ drawing processor ที่ต้องการ

Drawing Processor นั้นมีด้วยกัน 72 คำสั่ง 8 registers และ 33 level user stack. กลุ่มคำสั่งนี้คล้ายกับ PIC assembler และเราต้องมีความรู้เกี่ยวกับโปรแกรมภาษา PIC assembly และ Microchip MPASM/MPLAB IDE หากเราต้องการเขียนรูปแบบของเราเอง นอกจากนี้การความเข้าใจการทำงาน postfix stack จะเป็นสิ่งดีด้วย

เนื่องจากรหัส DP นั้นสร้างจาก MPASM macros และจบลงด้วยการ assembled กับรหัสโปรแกรม PIC ที่เหลืออยู่ ผู้ใช้งาน MPASM ปกติมีจำนวนรูปแบบสำหรับนำมาใช้ เมื่อต้องการเขียนรหัสสำหรับ DP

สำหรับโครงงานนี้คำสั่ง Voxel คือคำสั่งที่ควบคุม LED แต่ละหลอดภายใน cube ที่ตำแหน่งนั้นกำหนดโดยตำแหน่ง x, y และ z

- Registers

DP มี 8 register สำหรับโดยปกติจะมี 4 register และสำหรับแบบพิเศษนี้จะมีอีกอีก 4 register ทำให้ register ทั้งหมดนั้นกว้าง 8 bits และสามารถรับค่าในช่วง 0-255 อย่างไรก็ตาม คำสั่งที่ใช้ค่าต่างๆใน function registers พิเศษ นั้นคาดว่าค่าเหล่านี้จะอยู่ในช่วงที่แสดงในตางรางด้านล่าง ค่านอกเหนือช่วงค่าที่แสดงอาจเป็นผลไม่สามารถบอกการทำงานของรหัสได้

Register name	Function	Range
R0	General purpose register	0-255
R1	General purpose register	0-255
R2	General purpose register	0-255
R3	General purpose register	0-255
RHOLD	Hold time value for SHOW command (x 10mS)	0-255
RX	X co-ordinate for voxel and drawing functions	0-4
RY	Y co-ordinate for voxel and drawing functions	0-4
RZ	Z co-ordinate for voxel and drawing functions	0-4

-Timer

มี user timer เดียวที่นับจากค่า preloaded ลงจนถึง 0 ในเวลา 1 วินาที

- Stacks

DP มี 2 stacks คือ user stack และ return address stack. Return address stack เก็บ return address สำหรับคำสั่ง Jump to Subroutine (JSR) และไม่สามารถเข้าถึงโดยตรงได้ ส่วน user stack นั้นมีถึง 33 ระดับ และสามารถนำมาใช้เพื่อ push/pull user register ใดๆ ได้ นอกจากนี้ยังใช้งานทางคณิตศาสตร์และตรรกศาสตร์ โดยเก็บค่าเหล่านี้ให้อยู่ใน stack แล้วดำเนินการ ผลจะถูกดึงกลับไปยัง stack นี้ ตอนท้าย ผู้ใช้ stack DUP, DROP, SWAP, OVER และ ROT ยังเกิดขึ้นด้วย

- Flags

- Fcarry-Set หาก carry เกิดขึ้นระหว่างเพิ่มคำสั่งจะ ลบออกหากไม่มี borrow เกิดขึ้นระหว่างการลบ (เช่นเดียวกับการทำงานของ underlying PIC)
- Fzero-Set หากผลเป็นศูนย์ Fzero flage จะถูกลบออกหากผลไม่เป็นศูนย์ Fzero flage จะถูกแก้ไขโดยคำสั่ง TSTVOX, CMP, Timer และ external switch

- Instruction Set (กลุ่มคำสั่ง)

r - register name
k - literal, constant data or label

Mnemonic	Description	Flags	Register
NOOP	No operation		
MSET	Modify operations will set voxel to on		
MCLR	Modify operations will set voxel to off		
MINV	Modify operations will invert current voxel value		
SETALL	Turn on all voxels in cube (independant of MSET, MCLR and MINV instructions)		
CLRALL	Turn off all voxels in cube (independant of MSET, MCLR and MINV instructions)		
INVALL	Invert all voxels in cube (independant of MSET, MCLR and MINV instructions)		
SHOW	Transfer drawing buffer to display and load value in RHOLD register into hold timer		
VOX k,k,k	Load RX, RY, RZ and modify voxel [0 <= k <= 4		RX,RY,RZ

VOXM	Modify voxel at current RX, RY, RZ co-ordinates		
TSTVOX	Test voxel at current RX, RY, RZ co-ordinates. Fzero clear if voxel on : Fzero set if voxel off	Fzero	
CHYR r	Draw ASCII character specified in register r in the Y (vertical) plane		
CHZR r	Draw ASCII character specified in register r in the Z (horizontal) plane		
CHY k	Draw ASCII character value k in the Y (vertical) plane [32 <= k <= 95]		
CHZ k	Draw ASCII character value k in the Z (horizontal) plane [32 <= k <= 95]		
LINE k,k,k,k	Modify a line of voxels, specify x inc, y inc, z inc, length		RX,RY,RZ
LINEX	Modify line of voxels across the whole X axis, located at RY,RZ		
LINEY	Modify line of voxels across the whole Y axis, located at RX,RZ		
LINEZ	Modify line of voxels across the whole Z axis, located at RX,RY		
PLANEX	Modify all voxels in the YZ plane, located in the X-axis at RX		
PLANEY	Modify all voxels in the XZ plane, located in the Y-axis at RY		
PLANEZ	Modify all voxels in the XY plane, located in the Z-axis at RZ		
ROTATEX	Rotate entire cube along a line at y=2, z=2 in the x-axis		
SHXL	Shift entire drawing buffer left one voxel.		
SHXR	Shift entire drawing buffer right one voxel.		
SHYU	Shift entire drawing buffer up one voxel		
SHYD	Shift entire drawing buffer down one voxel		
SHZF	Shift entire drawing buffer forward one voxel		
SHZB	Shift entire drawing buffer back one voxel		
DECX	Decrement RX register, modulo 5	Fzero	RX
DECY	Decrement RY register, modulo 5	Fzero	RY
DECZ	Decrement RZ register, modulo 5	Fzero	RZ
INCX	Increment RX register, modulo 5	Fzero	RX
INCY	Increment RY register, modulo 5	Fzero	RY
INCZ	Increment RZ register, modulo 5	Fzero	RZ
DECR r	Decrement register, modulo 256	Fzero	r

DECRSZ r	Decrement register, modulo 256, Skip next instruction if result is zero		r
INCR r	Increment register, modulo 256	Fzero	r
INCRSZ r	Increment register, modulo 256, Skip next instruction if result is zero		r
PUSHR r	Push register contents onto top of stack		
PULLR r	Pull top of stack and place contents into register		r
PUSHXYZ	Push registers RX, RY, RZ on to stack		
PULLXYZ	Pull registers RX, RY, RZ from stack		RX,RY,RZ
DROP	Pull entry from top of stack and discard it (a --)		
SWAP	Swap top two entries on stack. (a b -- b a)		
DUP	Duplicate entry on top of stack. (a b -- a a b)		
OVER	Operates on the stack : (a b -- a b a)		
ROT	Operates on the stack : (a b c -- b c a)		
TSTZ	Test value on top of stack and condition Fzero flag (a -- a) : a==0 Fzero set, a != 0 Fzero clear	Fzero	
ADD	pulls two values from the stack, adds them together and pushes result back onto stack	Fzero Fcarry	
SUB	pulls two values from the stack, subtracts them and pushes result back onto stack *order is (TopOfStack-1) - (TopOfStack) -> TopOfStack*	Fzero Fcarry	
AND	pulls two values from the stack, performs a bitwise 'AND' result is pushed back onto stack	Fzero	
OR	pulls two values from the stack, performs a bitwise 'OR' result is pushed back onto stack	Fzero	
XOR	pulls two values from the stack, performs a bitwise 'XOR' result is pushed back onto stack	Fzero	
NOT	pull value from top of stack, perform bitwise 'NOT' operation on byte and push result back on to stack	Fzero	
CMP r, k	Compare register contents with k. If contents of r == k then Fzero Set, else Fzero Cleared.	Fzero	
LDXYZ k,k,k	Load RX, RY, RZ [0 <= k <= 4] (*see also VOX k,k,k*)		RX,RY,RZ
LDR r,k	Load register with value k [0 <= k <= 255]		r
LDRAND r,k	Load register with random number in the range [0 <= Random Number < k]		r
LDTMR k	Load timer with period in seconds [1 <= k <= 255		

ADDTRND k	Add random number in the range [0 <= Random Number < k] to contents of Timer		
SKIPZ	Skip next instruction if Fzero flag is set		
SKIPNZ	Skip next instruction if Fzero flag is clear		
SKIPC	Skip next instruction if Fcarry flag is set		
SKIPNC	Skip next instruction if Fcarry flag is clear		
SKIPTOUT	Skip next instruction if Timer == 0		
JUMP k	Jump to program address k		
JSR k	Jump to subroutine at address k		
RET	Return from subroutine		
RANDSEED	Seed random number generator with non-zero value from TMR0		
SYNCEXT k	Wait for a falling edge on SW1 input before continuing program execution or timer out	Fzero	
TSTSW	Test SW1 input. Set Fzero flag if switch active (pressed), Clear Fzero flag if switch not active	Fzero	

- SHOW

คำสั่งวาดรูปทั้งหมดทำงานบน drawing buffer ขณะที่ LED cube แสดงรายละเอียดของ display buffer เพื่อส่งผ่าน drawing buffer สู่ display buffer เราต้องใช้คำสั่ง SHOW เมื่อใช้คำสั่ง show จะรอให้ hold delay ก่อนหน้านี้เสร็จสิ้น หากยังทำงานอยู่ หลังจากนั้นตั้งค่า flag ส่งผ่าน buffer และรอ display driver คัดลอก drawing buffer เข้าใน display buffer และล้าง transfer flag ตอนนี้ค่าใน RHOLD ถูกส่งผ่านไปยัง hold time และ instruction exits, RHOLD timer คือคาบรอ X10mS หาก RHOLD คือ 0 buffer นี้จะถูกส่งผ่านที่ cube display refresh ต่อไป

คำสั่งนี้จะกั้นการทำงานของโปรแกรม จะไม่ดำเนินต่อไปจนกระทั่ง hold timer หมดเวลาลง และ interrupt display driver ได้ส่งผ่านไปยัง buffer นี้แล้ว

- SYMCEXT

คำสั่ง sync ภายนอกจะรอจนกระทั่งขอบลงของอินพุท S1 มาถึง หรือหมดเวลาลงก่อนดำเนินโปรแกรมต่อไป

หากค่า timeout ถูกตั้งค่าเป็น 0 คำสั่งจะรอไปไม่มีกำหนด จนกว่าขอบลงจะมาถึงก่อนดำเนินการใดๆต่อไป หากค่า timeout ถูกตั้งค่าระหว่าง 1 และ 255 คำสั่งจะรอจนกระทั่งตรวจจับขอบลงได้หรือค่า timer

-คำสั่ง SHXL, SHXR, SHYU, SHYD, SHZF, SHZB

คำสั่ง SHXL, SHXR, SHYU, SHYD, SHZF, SHZB สำหรับนำ drawing buffer one voxel เข้าสู่ direction specified voxels ที่ขอบข้าวของ shift นั้นจะถูกล้าง(ตั้งค่าเป็น off)

รูปที่ 2.1 ตัวอย่างผลของคำสั่ง SHXL, SHXR, SHYU, SHYD, SHZF, SHZB

- LINE x_inc, y_inc, z_inc, length

คำสั่งนี้ใช้แก้ไข line ของ voxels

- x_inc, y_inc และ z_inc สารถมีค่าได้เป็น -1, 0 หรือ 1
- จุดเริ่มของเส้นคือค่าปัจจุบันใน RX, RY, RZ
- หลังจากแก้ไข voxel ที่ RX, RY, RZ แล้ว
  RX = RX + x_inc : RY = RY + y_inc : RZ= RZ + z_inc : Length = Length -1
  จะทำซ้ำจนกระทั่ง Length == 0
- คำสั่ง Line จะทิ้งค่า RX, RY และ RZ ให้เป็นจุดสิ้นสุดสำหรับเส้นนี้ ให้คำสั่งเส้นที่ตามมาวาดเส้นใหม่เริ่มจากจุดปลายของเส้นก่อนหน้านี้
- การทำงาน เพิ่ม/ลด บน registers คือ modulo 5 ดังนั้นการลดจะ roll ด้านล่างจาก 0 กลับไปยัง 4 และการเพิ่ม rolls over

MSET
LDXYZ 0,0,0
LINE 1, 1, 1, 5

plots a line through the following voxels

RX	RY	RZ
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4

MSET
LDXYZ 0,4,2
LINE 0, -1, 1, 2

plots a line through the following voxels

RX	RY	RZ
0	4	2
0	3	3

MSET
LDXYZ 2,2,2
LINE -1, 0, 1, 4

plots a line through the following voxels

RX	RY	RZ	
2	2	2	
1	2	3	
0	2	4	
4	2	0	*<- rollover*

MSET
LDXYZ 0,0,0
LINE 0, 0, 1, 5

plots a line through the following voxels

RX	RY	RZ
0	0	0
0	0	1
0	0	2
0	0	3
0	0	4

รูปที่ 2.2 ตัวอย่างคำสั่ง LINE x_inc, y_inc, z_inc, length

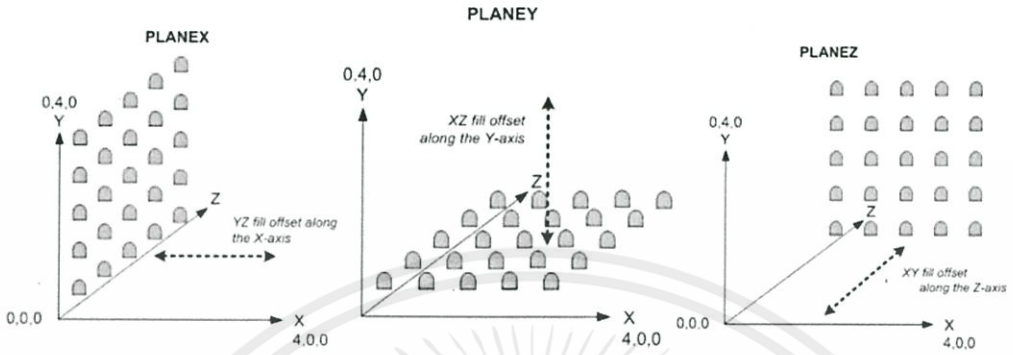
- LINEX, LINEY, LINEZ

คำส่งเหล่านี้วาด length line เต็มตามแกนหนึ่งใน drawing buffer
Voxels ในเส้นจะถูกตั้งค่า ล้าง หรือ inverted ตามโหมดปัจจุบันที่ตั้งค่าโดยคำสั่ง MCLR, MSET หรือ MINV ล่าสุด

- คำสั่งทั้ง 3 นี้ทำงานได้เร็วกว่าคำสั่ง LINELINEX
  อยู่ที่ระนาบ Y-Z โดยค่า RY และ RZ ปัจจุบัน
- LINEY
  อยู่ที่ระนาบ X-Y โดยค่า RX และ RZ ปัจจุบัน
- LINEZ
  อยู่ที่ระนาบ X-Y โดยค่า RX และ RY ปัจจุบัน

-PLANEX, PLANEY, PLANEZ

คำสั่ง PLANE เติมเติมระนาบทั้งหมดใน drawing buffer
Voxels ในระนาบจะถูกตั้งค่า ล้าง หรือ inverted ตามโหมดปัจจุบันที่ตั้งค่าโดยคำสั่ง MCLR, MSET หรือ MINV ล่าสุด



(a) (b) (c)

(a) ค่า RX กำหนดตำแหน่งตามแนวแกน X

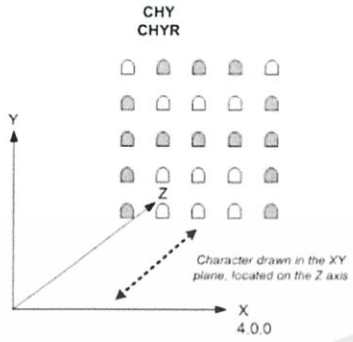
(b) ค่า RY กำหนดตำแหน่งตามแนวแกน Y

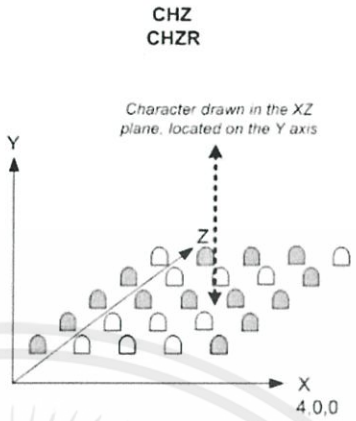
(c) ค่า RZ กำหนดตำแหน่งตามแนวแกน Z

รูปที่ 2.3 ตัวอย่าง PLANEX, PLANEY, PLANEZ

- CHZ, CHZR, CHY, CHYR

การตั้งค่าตัวอักษรขนาด 5X5 ได้ถูกกำหนดสำหรับตัวอักษร ASCII ในช่วง 32 ถึง 95 ซึ่งครอบคลุมตัวเลข 0-9 และตัวอักษรใหญ่ A-Z พร้อมกับสัญลักษณ์เกือบทั้งหมด





ตัวอักษร “A” ในระนาบ XY

ตัวอักษร “A” ในระนาบ XZ

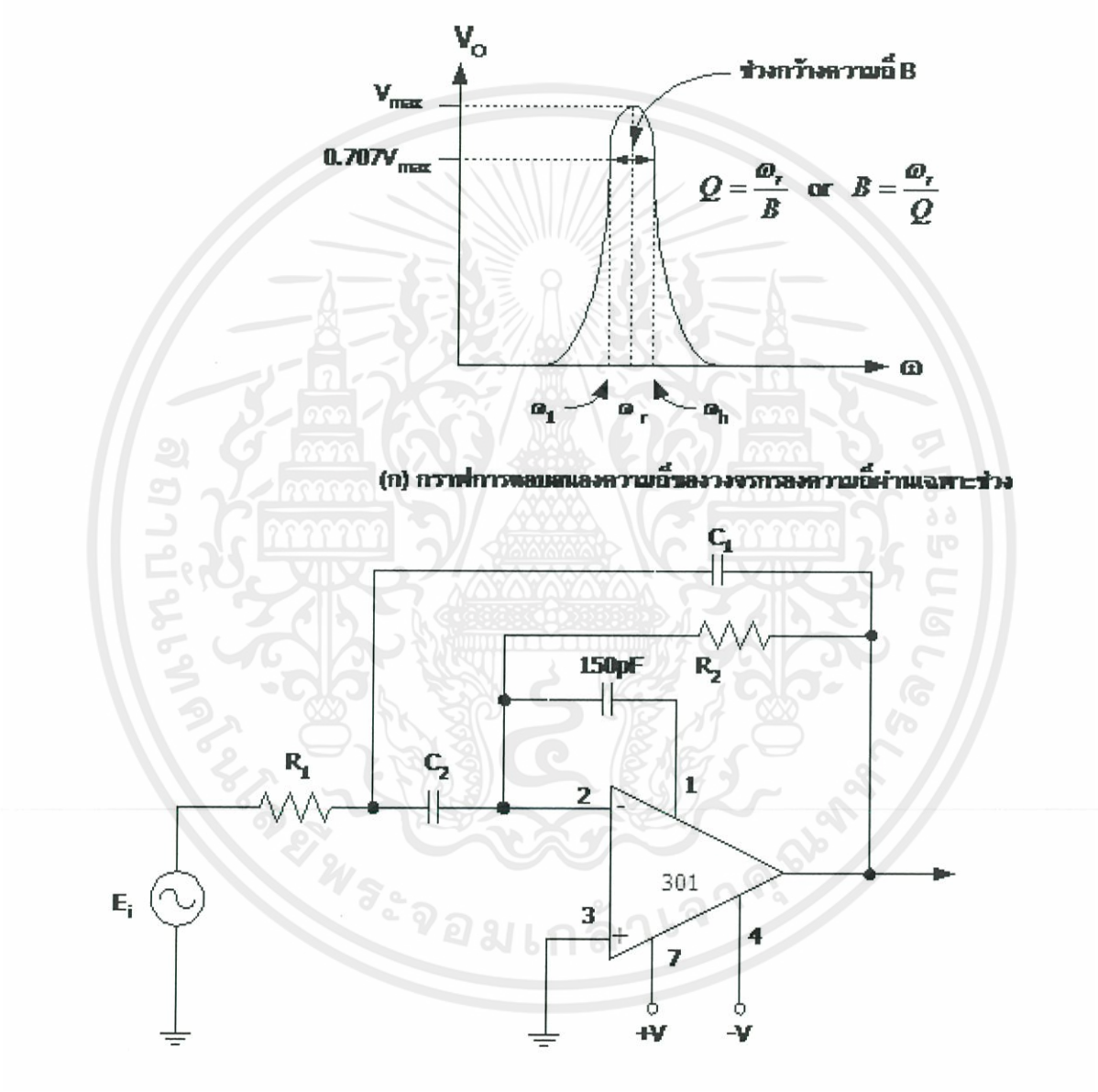
ค่าตัวอักษรตำแหน่ง RZ ในระนาบ Z

ค่าตัวอักษรตำแหน่ง R Y ในระนาบ Y

รูปที่ 2.4 CHZ, CHZR, CHY, CHYR

2.3 วงจรกรองสัญญาณช่วงความถี่ (Band Pass Filter: BPF)

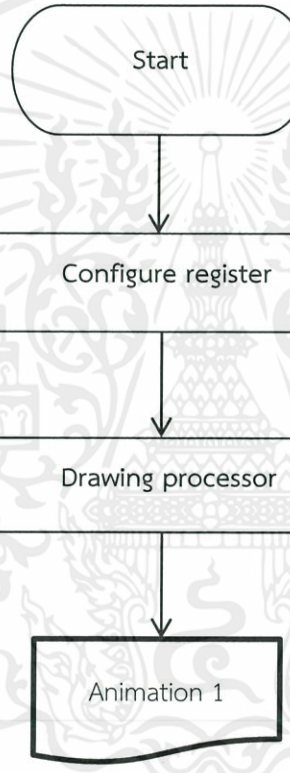
เนื่องจากเป็นวงจรที่มีลักษณะคล้ายกับการนำเอาวงจรกรองสัญญาณความถี่ต่ำ และความถี่สูง มาต่อร่วมกัน (Cascade) ดังนั้น วงจรกรองความถี่ผ่านเฉพาะช่วง จะยอมให้สัญญาณผ่านไปได้เฉพาะช่วงที่กำหนดเท่านั้น ความถี่ที่นอกเหนือจากที่กำหนดจะถูกจำกัดโดยการลดทอนให้หมดไป



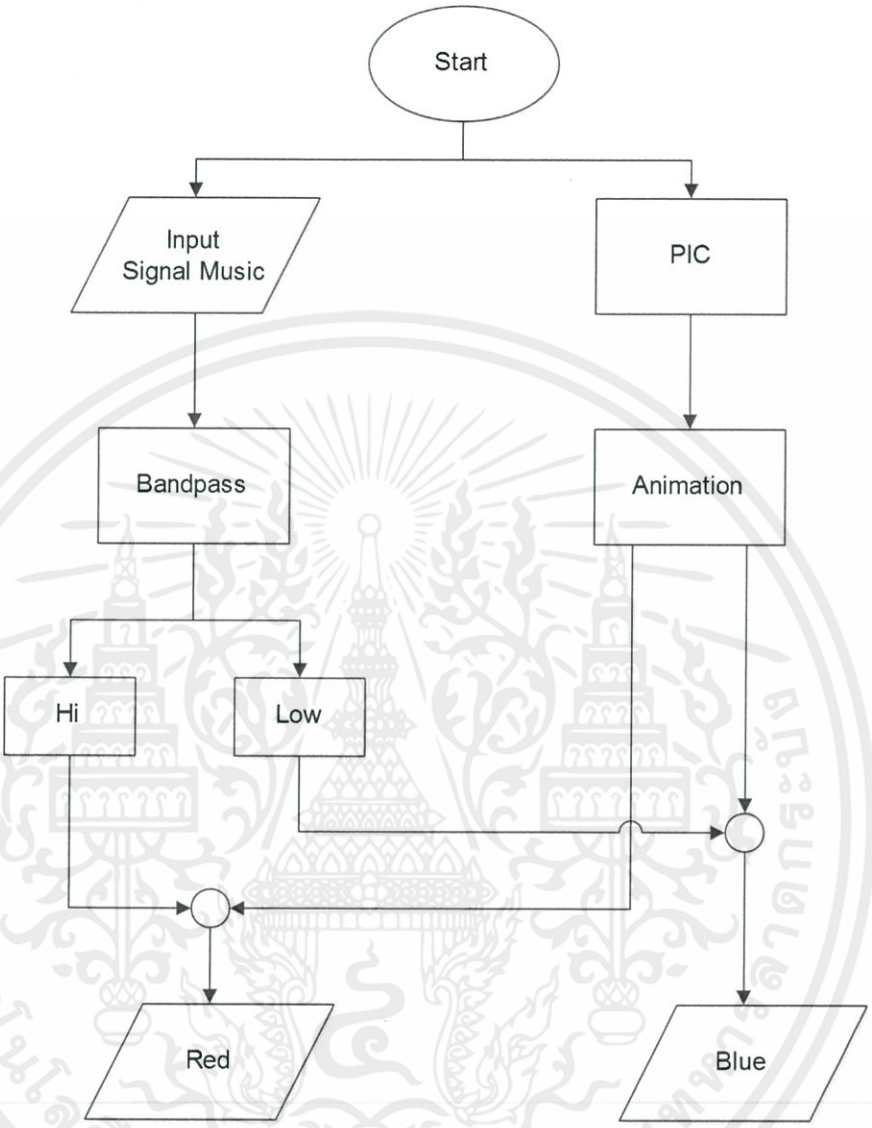
รูปที่ 2.5 วงจรกรองสัญญาณช่วงความถี่และกราฟแสดงผลตอบสนองอัตราขยายเชิงความถี่

บทที่ 3
การออกแบบและการประกอบ

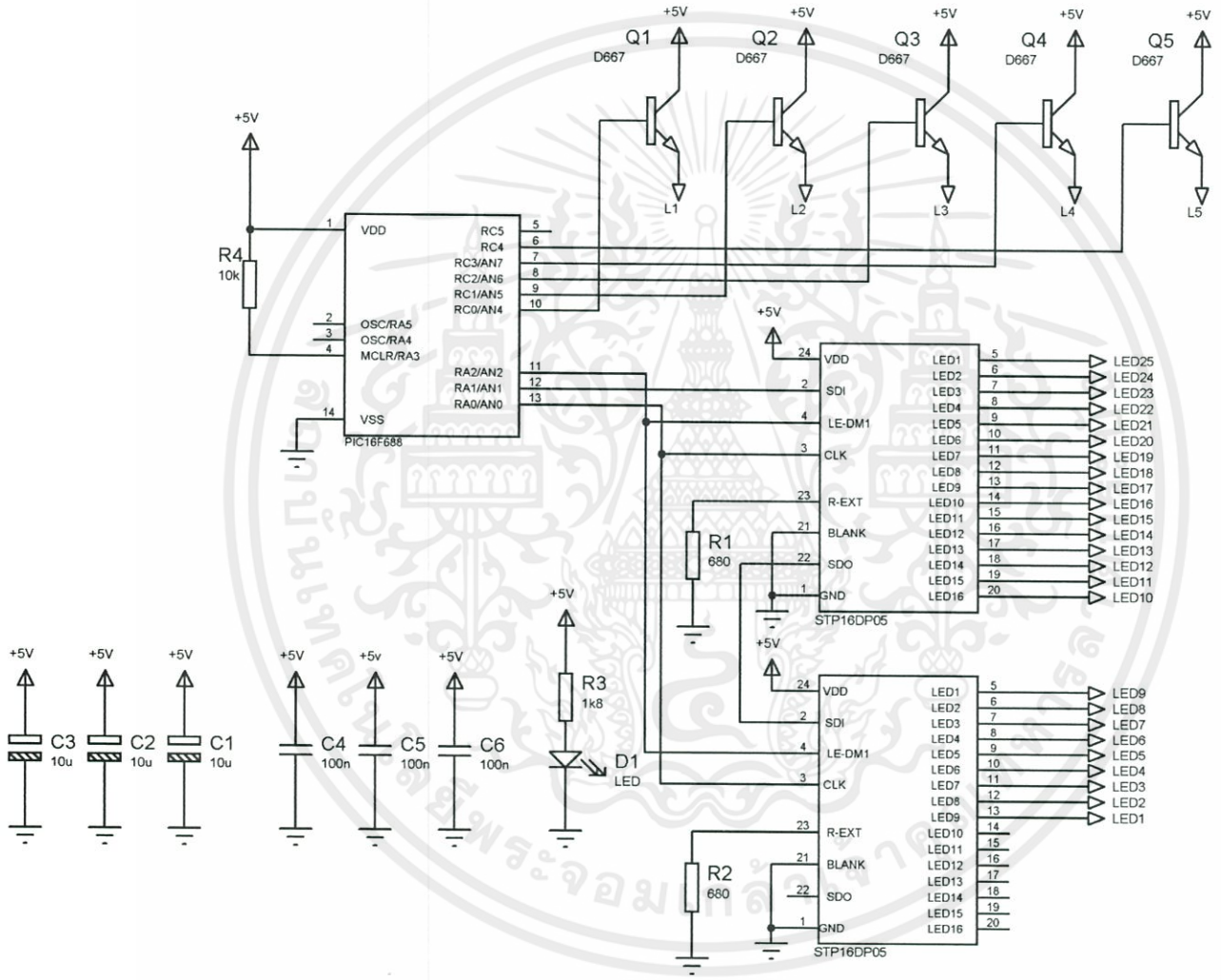
3.1 Software Flowcharts,Full Flowcharts & schematics



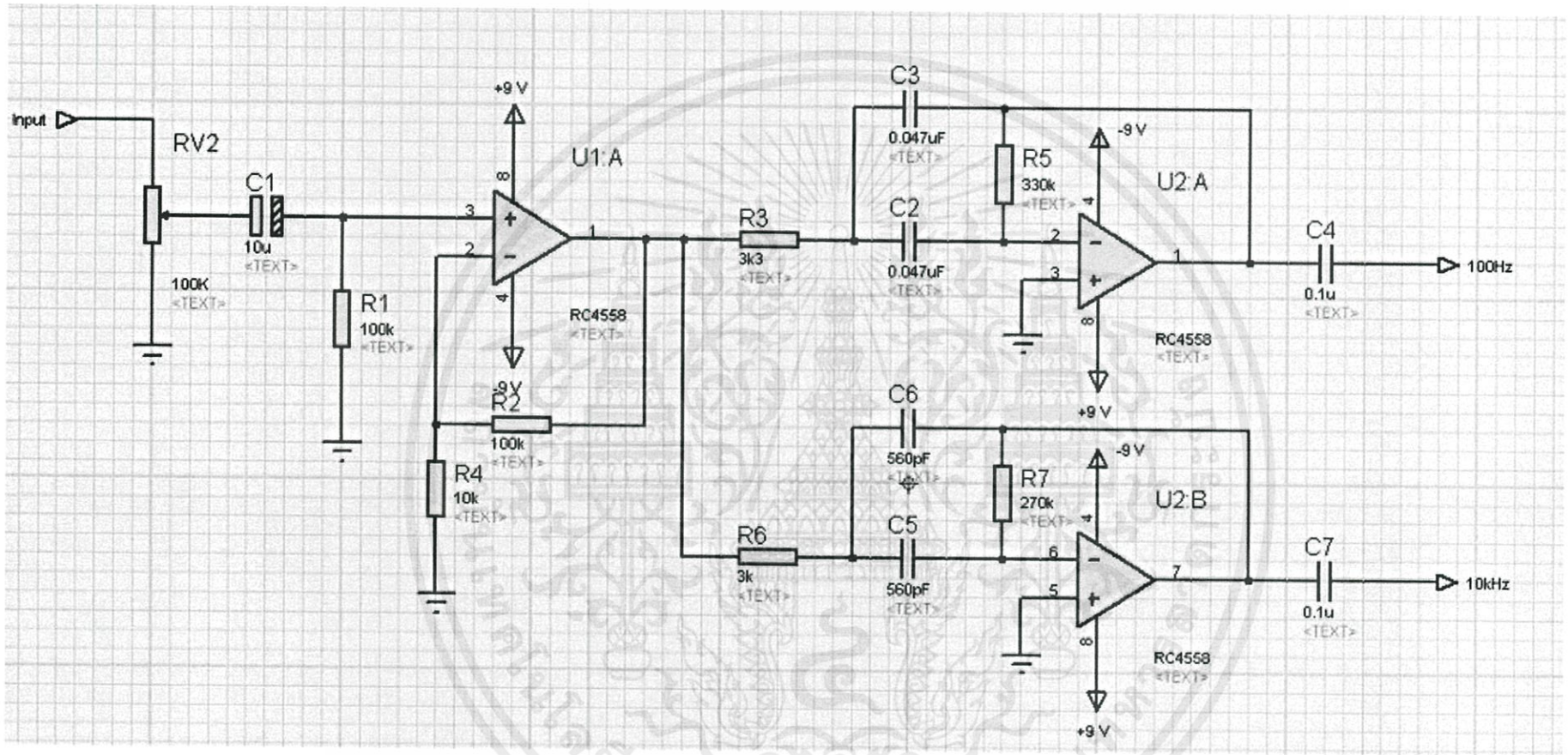
รูปที่ 3.1.1 Software Flowcharts



รูปที่ 3.1.2 Full Flowcharts



รูปที่ 3.1.3 schematics ของ LED Cube วงจรเดิม



รูปที่ 3.1.4 schematics ของวงจร Band Pass filter ที่ใช้

การออกแบบ

3.1.1 ตัวอย่างคำสั่งคอนโทรลเลอร์ที่เขียนควบคุมหลอดไฟ LED

-Configure register

```
list      p=16f688
#include <P16F688.inc>
   __CONFIG    _CP_OFF & _CPD_OFF & _BOD_ON &
_PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
& _MCLRE_ON & _FCMEN_OFF & _IESO_OFF
```

- Drawing processor

```
RUN.IN.SEQUENCE
        JSR     PROJECTS4C
RUN.IN.SEQ.LOOP
        JSR     HELIX
        JSR     WAVEX
        JSR     WAVEY
        JSR     WAVEZ
        JUMP  RUN.IN.SEQ.LOOP
```

```
RUN.IN.SEQUENCE
        JSR     PROJECTS4C
RUN.IN.SEQ.LOOP
        JSR     HELIX
        JSR     WAVEX
        JSR     WAVEY
        JSR     WAVEZ
JUMP  RUN.IN.SEQ.LOOP
```

-PROJECTS text animation

```
PROJECTS4C
        LDR RZ,4
        MSET
        LDR R0, 'P'
        JSR WRITE
        LDR R0, 'R'
        JSR WRITE
        LDR R0, 'O'
        JSR WRITE
        LDR R0, 'J'
        JSR WRITE
        LDR R0, 'E'
        JSR WRITE
        LDR R0, 'C'
```

-WAVES (Animation 2-4)

```
WAVEX       JSR   WAVE.setup
        JUMP  WAVEX.loop
WAVEY       JSR   WAVE.setup
        JUMP  WAVEY.loop
WAVEZ       JSR   WAVE.setup
        JUMP  WAVEZ.loop
WAVE.setup  CLRALL
        MSET
        LDRAND  RHOLD,8
        PUSHR RHOLD
        LDR RHOLD,8
        PUSHR RHOLD
        ADD
        PULLR RHOLD
        LDXYZ 0,2,0
        LDR R0,1
        LDTMR 4
        ADDTRND 6
        RET
```

-WAVEX.loop

```
WAVEX.loop
        LINEZ
        SHOW
        SHXR
        PUSHR R0
        PUSHR RY
        ADD
        PULLR RY
        CMP RY,4
        SKIPNZ
        LDR R0,-1
        CMP RY,0
        SKIPNZ
        LDR R0,1
        SKIPTOUT
        JUMP WAVEX.loop
          RET
```

-WAVEXY.loop

```
WAVEY.loop
        LINEX
        SHOW
        SHZB
        PUSHR R0
        PUSHR RY
        ADD
        PULLR RY
        CMP RY,4
        SKIPNZ
        LDR R0,-1
        CMP RY,0
        SKIPNZ
        LDR R0,1
        SKIPTOUT
JUMP WAVEY.loop
```

-WAVEXZ.loop

```
WAVEZ.loop
        LINEY
        SHOW
        SHZB

        PUSHR R0
        PUSHR RX
        ADD
        PULLR RX
        CMP RX,4
        SKIPNZ
        LDR R0,-1
        CMP RX,0
        SKIPNZ
        LDR R0,1
        SKIPTOUT
        JUMP WAVEZ.loop
        RET
```

-Char.base

EX:ตัวเลข 4

```
;4
        retlw   b'00110'
        retlw   b'01010'
        retlw   b'11111'
        retlw   b'00010'
        retlw   b'00010'
```

EX:ตัวอักษร ตัว K

```
; K
        retlw   b'10011'
        retlw   b'10100'
        retlw   b'11000'
        retlw   b'10100'
        retlw   b'10011'
```

3.2 รายละเอียดการออกแบบวงจร

LED cube สร้างจาก LEDs 125 ตัวจัดเรียงกันเป็น 5 ชั้น โดยแต่ละชั้นประกอบด้วย LEDs 25 ตัว ส่วนแสดงผลจะถูก multiplexed คือแทนที่ต้องใช้การเชื่อมต่อ LEDs ทั้งหมด 125 ตัว เราใช้ 1 การเชื่อมต่อสำหรับชั้นแต่ละชั้นของ LEDs และ 25 การเชื่อมต่อ LEDs 25 ตัวที่อยู่ในชั้นหนึ่งๆ จึงรวมกันเป็นการเชื่อมต่อทั้งหมด 30 เราเริ่มต้นการทำงาน cube นี้ใหม่ด้วยโปรแกรม interrupt routine ที่แต่ละชั้นจะทำงานเป็นเวลา 2 ms เท่านั้น นั่นคือ cube ทั้งหมดจะถูกกำหนดให้ทำงานใหม่ทุกๆ 10 mS(100 Hz) ผลคือจะไม่เห็น flicker ในส่วนแสดงผล

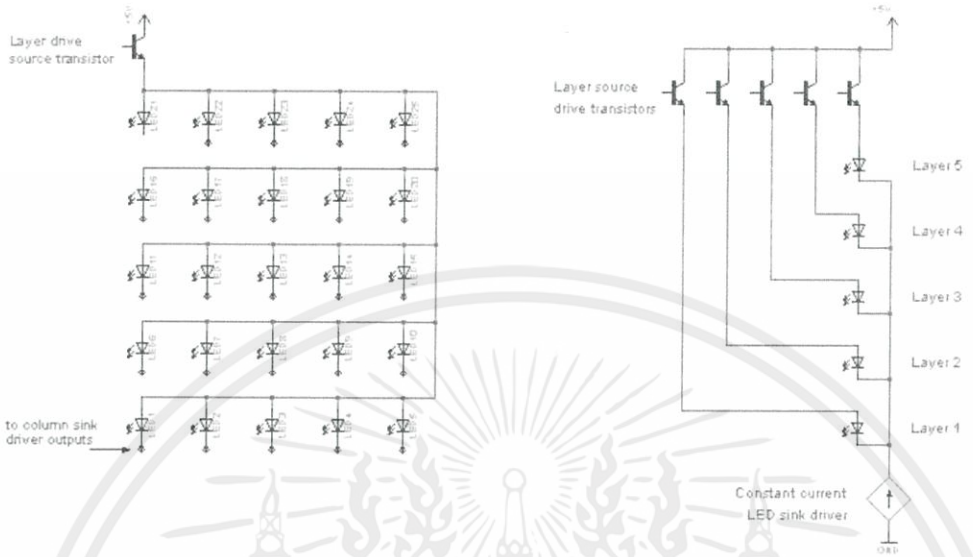
ใช้ 8 I/O lines เท่านั้นเพื่อควบคุม LED drivers สำหรับ cube นี้ ที่ให้ขา 14 ขาของ PIC 16F688 ซึ่งเป็นไมโครคอนโทรลเลอร์เป็นตัวควบคุม cube ทั้งหมด ไมโครคอนโทรลเลอร์นี้มีสัญญาณนาฬิกา 8 MHz อยู่ภายใน และมีความจำ 6 K-words

ชั้น LED แต่ละชั้นจัดเรียงตัวกันในรูปแมทริกซ์ขนาด 5X5 และควบคุมโดยทรานซิสเตอร์ 1 ตัว ใน รูปแบบ emitter follower ที่ต่อเข้ากับแอโนดของ LED เมื่อสัญญาณเอาท์พุทควบคุมชั้น LED ที่สอดคล้องกันจาก PIC มีค่าลอจิกเป็น “high” แรงดันไฟฟ้าที่เบสของทรานซิสเตอร์นี้จะถูกรักษาไว้ที่ +5V และแรงดันไฟฟ้าที่อีมิตเตอร์จะมีค่าต่ำกว่าค่าแรงดันเบสนี้ประมาณ 0.7 โวลต์ ทรานซิสเตอร์ที่ใช้คือ D667 NPN transistors เพราะมีค่า IC rating มากกว่า 1 Amp.

แคโทดของ LEDs นี้ต่อเข้ากับ IC2 และ IC3 (STP16DP05)ซึ่งคือตัวขับกระแสคงที่ 16 bit และแรงดันไฟฟ้าต่ำเบอร์ STP16DP05 ตั้งค่ากระแส LED โดยใช้ตัวต้านทาน 1 ตัวต่อเข้ากับ RSET input ของ IC นี้(ขา 23) ตัวต้านทานค่า 1K8(R1 และ R2) ตั้งค่ากระแส LED ไว้ที่ประมาณ 33 mA สามารถเปลี่ยนเพื่อเปลี่ยนค่าแหล่งจ่ายกระแสให้แก่ LEDs ได้

ชั้นต่างๆของ cube

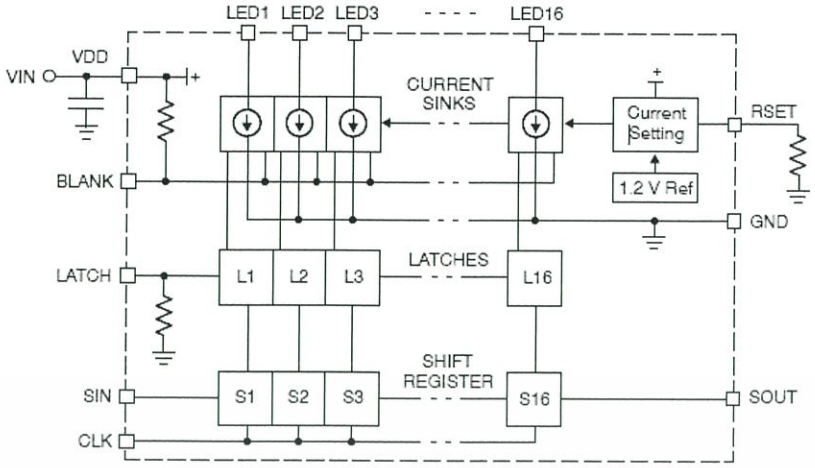
แต่ละคอลัมน์ของ LED cube



รูปที่ 3.2 ชั้นและคอลัมน์ต่างๆของLED Cube

ข้อดีของการใช้ IC ที่เป็นตัวขับกระแสให้คงที่คือสามารถใช้ LED เกือบทุกประเภทและกระแสยังคงคงที่ขึ้นกับแรงดังไฟฟ้าตรง LED หากไม่ต้องการเปลี่ยนแปลงกระแสเอาท์พุทนี้ แค่ทำการเปลี่ยนตัวต้านทานตั้งค่ากระแส 2 ตัวเท่านั้น

เอาท์พุทของตัวขับกระแส(IC2/IC3) นั้นถูกควบคุมโดยข้อมูล LED ที่โหลดเข้ามาโดย microcontroller PIC ตัวขับ ICs เหล่านี้แต่ละตัวประกอบด้วย shift register และ output latch อย่างละ 16 ตัว PIC จะนำ LED data 1 บิต ไปยังอินพุทอนุกรมของ IC2(SIN) หลังจากนั้น PIC จะสร้างพัลส์บนอินพุทของ ของ CLOCK ของ IC driver ทั้งสอง เพื่อส่งข้อมูลเข้าไป driver ICs ทั้งสองนี้ต่อกันแบบ cascaded(SOUT ของ IC2 ป้อนขา SIN ของ IC3) ดังนั้น PIC เพียง clocks ข้อมูล 25 bits เท่านั้น เมื่อ /5 data bits ถูกส่งไปยัง driver ICs สัญญาณ LATCH จะถูกส่งเพื่อวางข้อมูลไว้บนเอาท์พุทของ sink ปัจจุบัน หลังจากนั้น PIC จะตั้งค่าเอาท์พุทของทรานซิสเตอร์ที่เป็นตัวขับ LED ในชั้นที่สอดคล้องกันให้มีสภาวะเป็น “high” ทำให้ LEDs ในชั้นที่ต้องการสว่าง

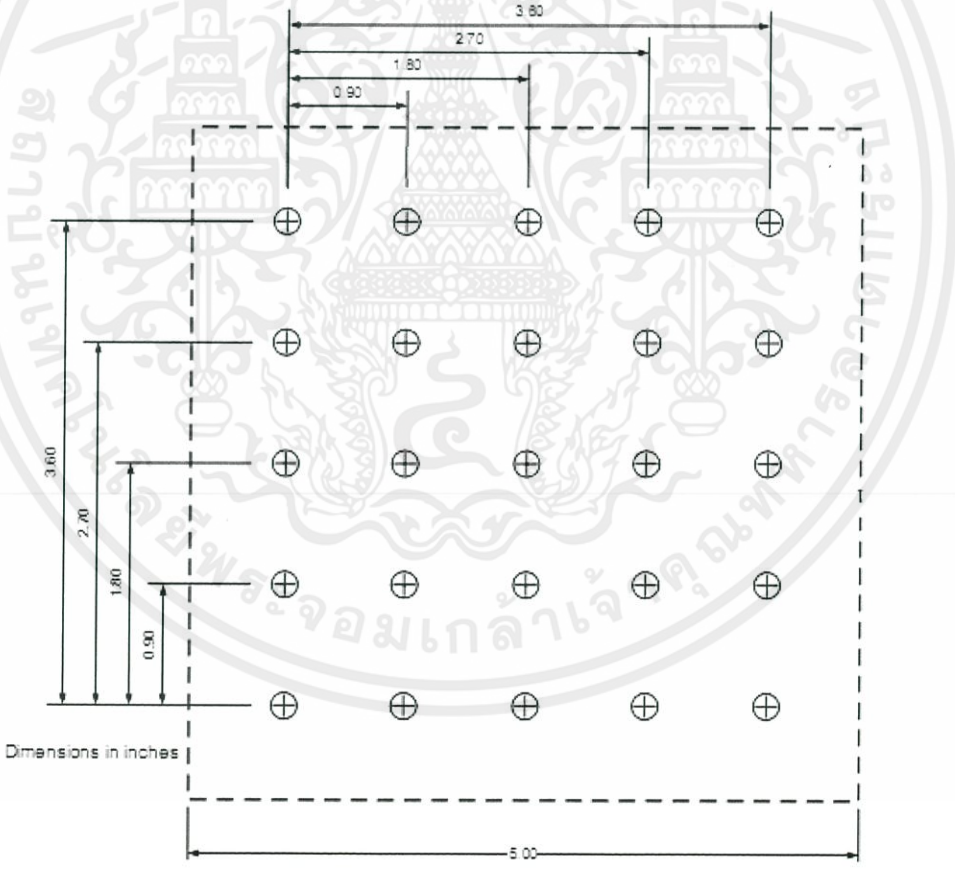


รูปที่ 3.3 แผนผังการทำงานของ IC2/IC3

3.3 การสร้าง LED 5x5x5

3.3.1 สร้างแบบสำหรับเจาะรูบนไม้อัดเพื่อบัดกรีขาหลอดในแต่ละชั้น

ต้องแน่ใจว่าระยะห่างจริงและระยะห่างจากแผนผังที่พิมพ์ออกมานั้นเท่ากัน



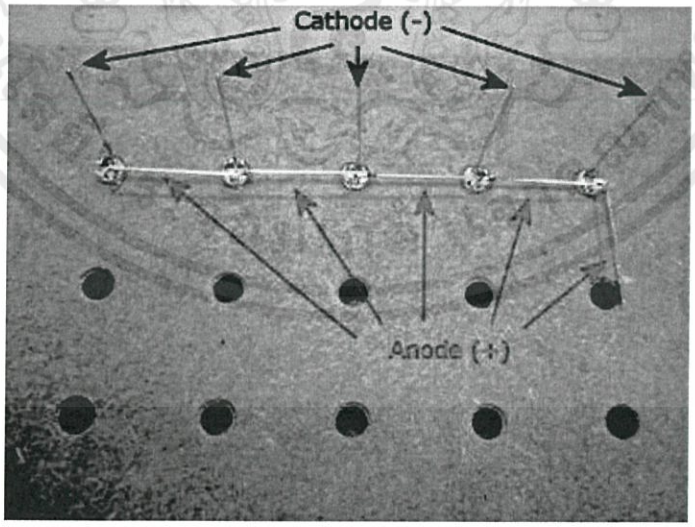
รูปที่ 3.4 แบบสำหรับเจาะรู้แผ่นไม้เพื่อบัดกรีขาหลอด LED

ตัดตามเส้นประ ติดกาวแผ่นแผนผังนี้ลงบนแผ่นไม้อัดใช้ตะปูแหลมทิ่มมาร์ค
ตำแหน่งที่จะเจาะรู จากนั้นทำการเจาะรู

รูปที่ 3.5 เจาะรู้แผ่นไม้สำหรับเพื่อบัดกรีขาหลอด LED

3.3.2 บัดกรีขาหลอด

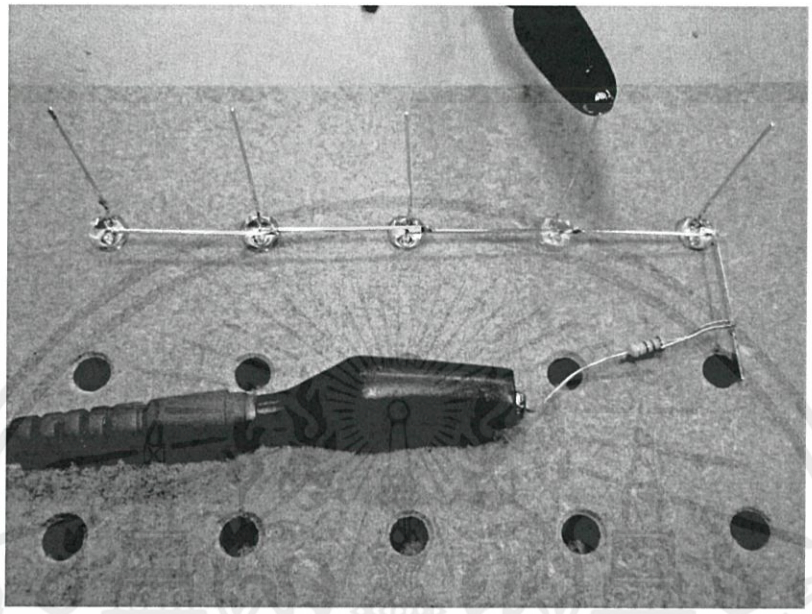
ค่อยๆ ดัดขา LED แต่ละตัว ควรยึดขาแคโทดของ LED ออก เพื่อให้เบี่ยงออกทางด้านข้างของตัว LED (ยังขนาดกับตัว LED) ส่วนขาแอโนดนั้นควรหักทำมุม 90 องศากับตัว LED โดยทั่วไปแล้ว ขา LED ที่สั้นกว่านั้นคือแคโทด จากนั้นควรตรวจสอบด้วยตัวเองอีกครั้ง



รูปที่ 3.6 บัดกรีขาหลอด LED

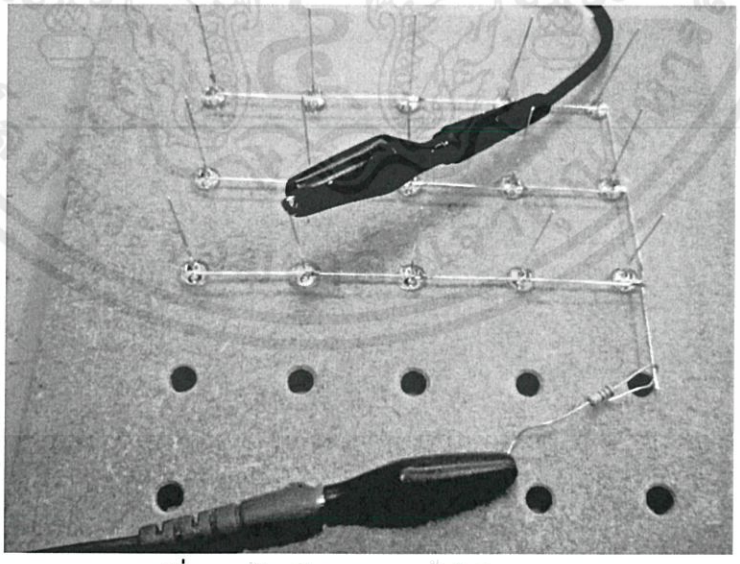
แอโนดของ LED นี้ อยู่ในแนวขนานกับแผ่นรองรับและเชื่อมต่อเข้าด้วยกันทั้งหมด ส่วนแคโทดนั้นตั้งฉากกับแผ่นรองรับ LED นี้ ติด LEDs ทั้ง 5 ในแถวหนึ่งตามรูป LED ตัวสุดท้ายควร มี

ขาแอโนดงอไปรอบๆ สำหรับเชื่อมต่อกับ LED ในแถวต่อไป ทำการบัดกรีขาแอโนดของ LED แต่ละตัวเข้ากับขา LED อีกตัว พร้อมทดสอบ LED แต่ละตัวว่าใช้งานได้ หรือไม่กลับขั้วแคโทดและแอโนด



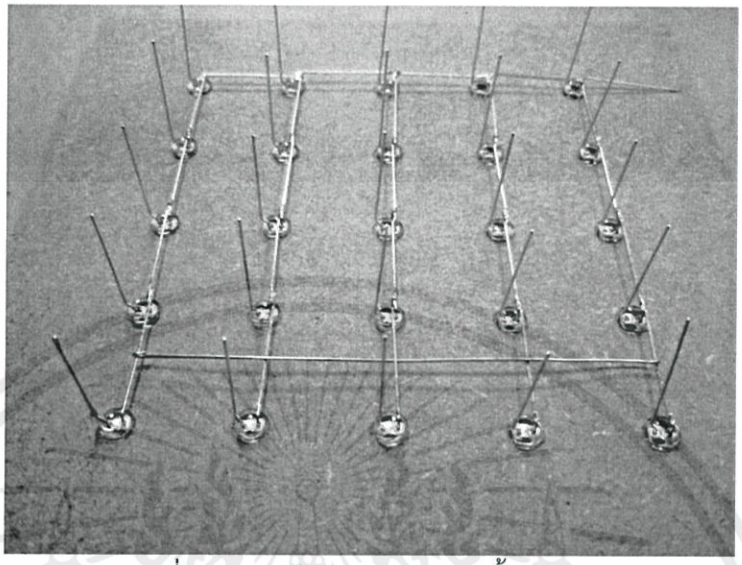
รูปที่ 3.7 ตรวจสอบหลอด LED ว่าทำงานหรือไม่

ต่อขาแอโนดเข้ากับ LED ตัวขวาสุด และ LED ในแถวต่อไป เรายังมีขาแอโนดจาก LED ทางด้านซ้ายมาเจอกัน ณ จุดนี้ บัดกรีขาทั้งสามเข้าด้วยกันตามรูป ให้แน่ใจว่าขาของ LED จากแถวก่อนหน้านี้ไม่บังขาแคโทดในแนวดิ่ง



รูปที่ 3.8 บัดกรี LED แถวต่อไป

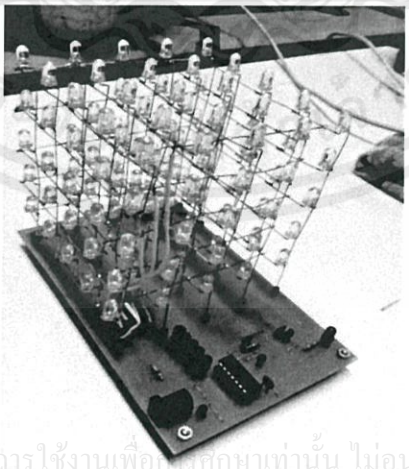
ติด LED บนแถวที่เหลือทั้งหมด บักรีและทดสอบอีกครั้ง



รูปที่ 3.9 บัดกรี LED จนครบ 1 ชั้น

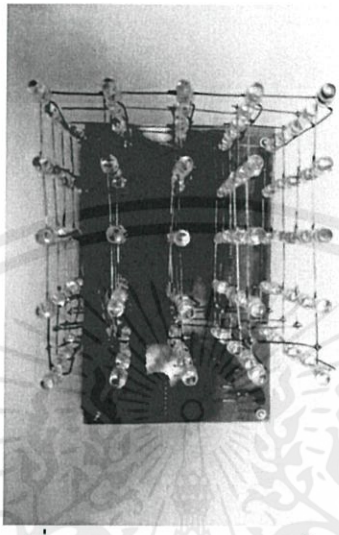
เมื่อติด LED ได้ 5 แถวแล้ว บัดกรีเส้นลวดที่เชื่อมต่อแถวทั้ง 5 แถว เพื่อจัดให้แถวทั้งหมดอยู่แนวเดียวกัน เส้นลวดนี้ยังทำหน้าที่การเชื่อมต่อทางไฟฟ้าได้ หลังจากนั้นยกชั้นการเชื่อมต่อ LEDs ออกจากฐานแบบอย่างระวัง แล้วนำไปวางไว้ด้านข้าง ทำแบบเดียวกันกับชั้นต่อไปจนครบทั้ง 5 ชั้น

เมื่อทุกชั้นอยู่ในตำแหน่งแล้ว ให้นำเส้นลวด 5 เส้นต่อเข้ากับตัวต่อของแต่ละชั้นตามรูป ให้แน่ใจว่าเส้นลวดแต่ละเส้นไม่สัมผัสกันเมื่อต่อสูงขึ้น หัวต่อเหล่านี้จะอยู่ห่างกันบนแผ่น PCB เพื่อหลีกเลี่ยงการลัดวงจร การเชื่อมต่อแสดงด้วยสัญลักษณ์ L1 ถึง L5 โดยที่ L1 เป็นชั้นล่างสุด และ L5 เป็นชั้นบนสุด เมื่อติดลูกบาศก์นี้ลงบน PCB และทำการต่อสายทั้งหมดแล้ว ให้ดูที่ลูกบาศก์อีกครั้งเพื่อให้แน่ใจว่าไม่มีขาใดโค้งงอและสัมผัสกันมิฉะนั้นแล้วจะทำให้ ICs ที่เป็นตัวขับ LED เสียได้



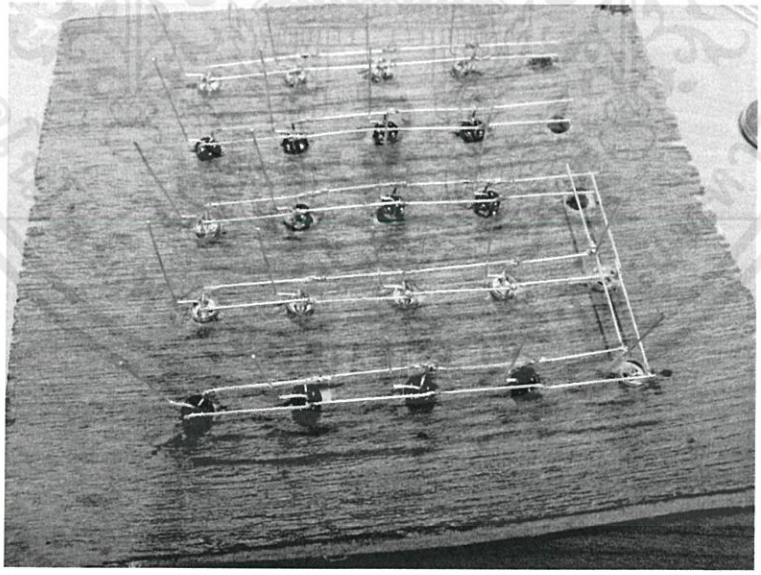
รูปที่ 3.10 บัดกรี LED ครบทุกชั้นและลงบอร์ด

เมื่อประกอบเสร็จแล้วเช็คหลอดว่าตรงดีหรือป่าว เช็คขาหลอดอย่าให้ลัดวงจรอีกรอบพร้อมตรวจสอบความสวยงาม



รูปที่ 3.11 ตรวจสอบความเรียบร้อย

ในภาคการศึกษานี้ได้เปลี่ยนหลอด LED Cube เป็นแบบ 3 ขา (+ - +) เพื่อให้ตอบสนองหลอดละสองสีเพื่อตอบสนองตามความถี่เสียงสองความถี่ที่แยกเอาไว้ โดยใช้หลักการต่อหลอดแบบก่อนหน้า ดังรูป

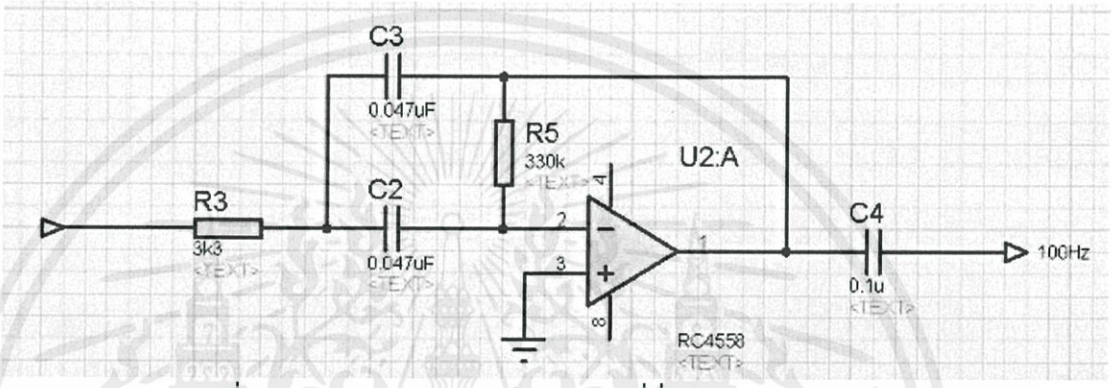


รูปที่ 3.12 บัดกรี LED 3 ขา

3.4 ชุดควบคุม LED ให้ตอบสนองตามเสียงเพลง

จะใช้วงจรแบนพาสฟิลเตอร์แยกความถี่สองวงจรคือแยะเสียงต่ำและสูงคือ 100 Hz และ 10 KHz แล้วนำเอ้าท์พุทมาขยายและนำไปควบคุมออปแอมป์ที่มาจากไมโครคอนโทรลเลอร์ให้ออปแอมป์ทำงานหรือไม่ทำงานหรือเป็นสวิตช์ที่คอยปิดเปิดหลอด LED ให้วิ่งตามเสียงเพลงนั่นเอง ซึ่งก็จะยังวิ่งเป็นรูปร่างที่ไมโครคอนโทรลเลอร์กำหนดไว้อยู่

3.4.1 แบนพาสฟิลเตอร์แยกความถี่ต่ำ



รูปที่ 3.13 แบนพาสฟิลเตอร์แยกความถี่ต่ำผ่าน

ใช้วิธีคำนวณแบบ Low pass filter และ High pass filter ผสมกัน

-ส่วน Low pass จะมี R3 และ C2

$$\omega_c = \frac{1}{RC} = 2\pi f_c$$

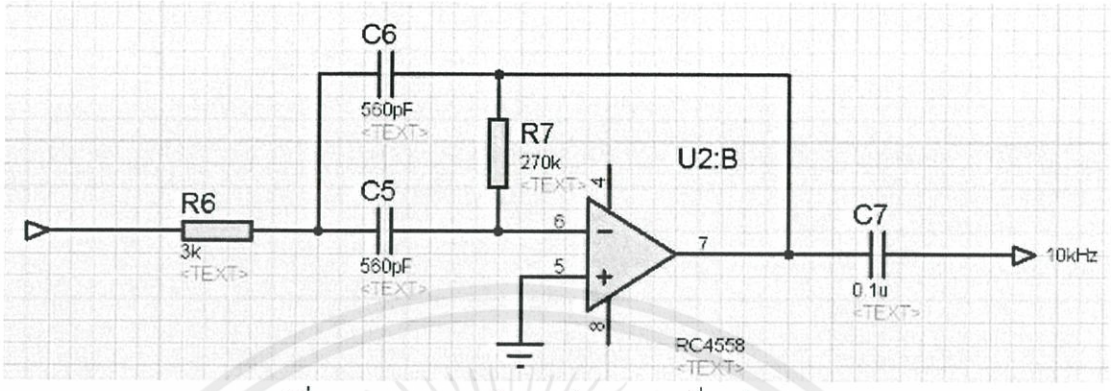
จากการกำหนดความถี่คัทออฟประมาน 1000 Hz กำหนดค่า C2 เท่ากับ 0.047uf จะได้ค่า R3 ประมาณ 3300 Ohm

-ส่วน High pass จะมี R5 และ C3

ใช้สมการการคิดแบบเดียวกับกัน Low Pass

จากการกำหนดความถี่คัทออฟประมาน 10 Hz กำหนดค่า C3 เท่ากับ 0.047uf จะได้ค่า R5 ประมาณ 330 K-Ohm

3.4.2 แบนพาสฟิลเตอร์แยกความถี่สูง



รูปที่ 3.14 แบนพาสฟิลเตอร์แยกความถี่สูงผ่าน

ใช้วิธีคำนวณแบบ Low pass filter และ High pass filter ผสมกัน

-ส่วน Low pass จะมี R3 และ C2

$$\omega_c = \frac{1}{RC} = 2\pi f_c$$

จากการกำหนดความถี่คัทออฟประมาน 10 KHz กำหนดค่า C5 เท่ากับ 560 pf จะได้ค่า R6 ประมาณ 3 K-Ohm

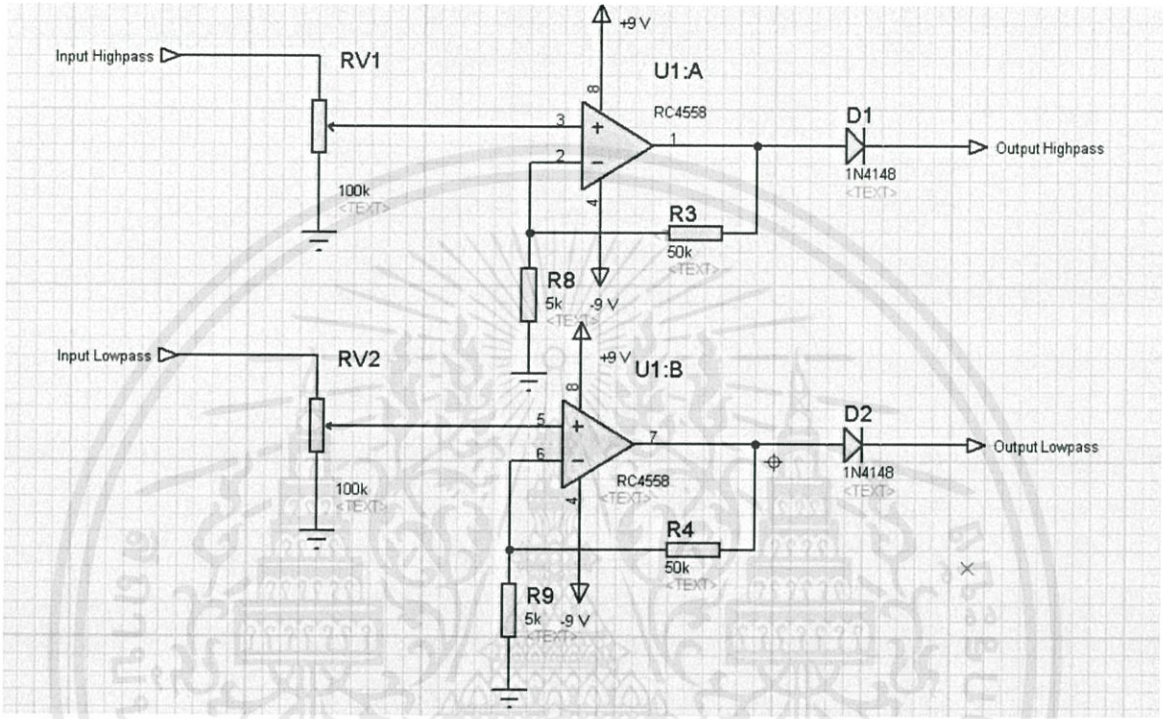
-ส่วน High pass จะมี R7 และ C6

ใช้สมการการคิดแบบเดียวกับกัน Low Pass

จากการกำหนดความถี่คัทออฟประมาน 1100 Hz กำหนดค่า C6 เท่ากับ 560pf จะได้ค่า R7 ประมาณ 330 K-Ohm

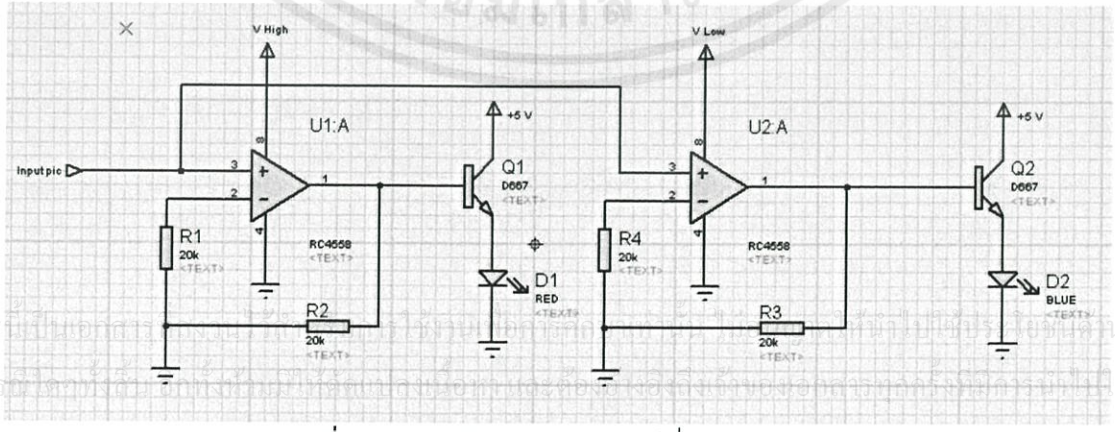
3.4.3 วงจรปรับ gain และขยายสัญญาณ

เมื่อได้สัญญาณเสียงที่แยกมาแล้วนำสัญญาณมาปรับ gain ก่อนนำไปขยายด้วยวงจรขยายไม่กลับเฟสในอัตตราส่วน 1:10



รูปที่ 3.14 ขยายสัญญาณเสียง

จากนั้นนำสัญญาณที่ได้ไปคุมออปแอมป์ที่ทางออกของสัญญาณจากไมโครคอนโทรลเลอร์ โดยนำสัญญาณที่ได้นี้ไปคุมไฟเลี้ยงออปแอมป์ดังกล่าวเพื่อเปิดปิดไฟตามเสียงเพลง



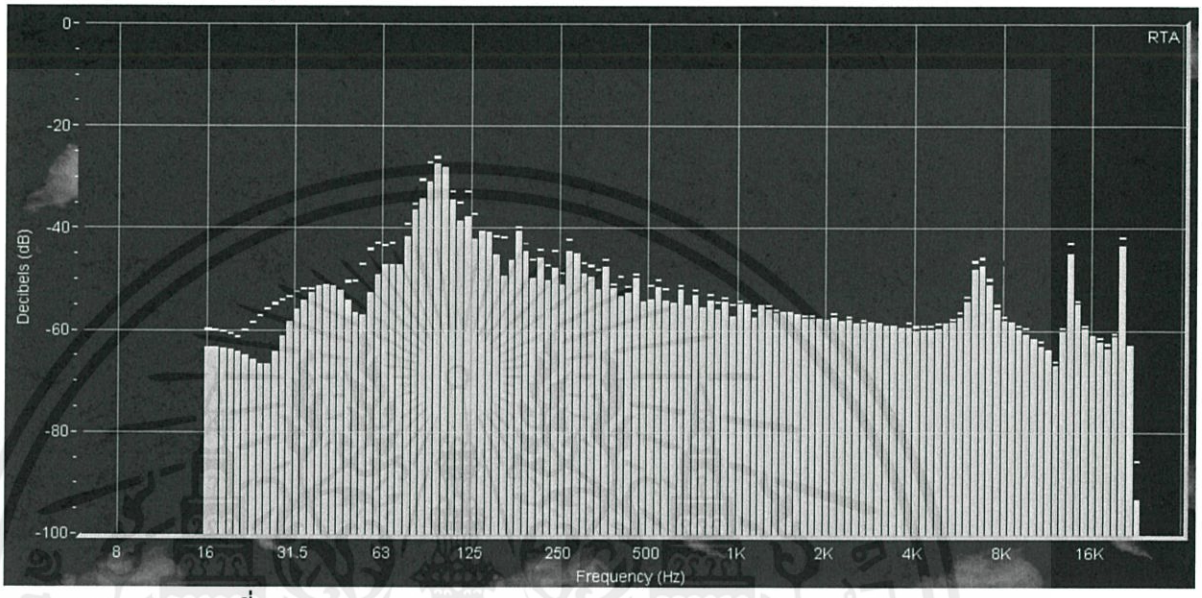
รูปที่ 3.15 คุมสัญญาณออปแอมป์ที่ควบคุม LED

3.5 ขั้นตอนการดำเนินงาน

ขั้นตอนการดำเนินงาน	พ.ย. 2555	ธ.ค. 2555	ม.ค. 2555	ก.พ. 2555
ค้นคว้า ข้อมูล และหลักการทำงานของ cube 3D				
ศึกษาวิธีนำสัญญาณเสียงมาควบคุมการทำงานของ LED				
สร้างวงจรนำสัญญาณเสียงมาควบคุมการทำงานของ LED				
รวบรวมข้อมูลโดยรวมของชิ้นงาน				
ทำการทดสอบ ชิ้นงานเป็นไปตามเป้าหมาย				
เริ่มทำรายงาน และ เก็บรายละเอียดหน้าต่างฐานข้อมูล				
สรุปผลการทดลอง และรายงานเสร็จสมบูรณ์				

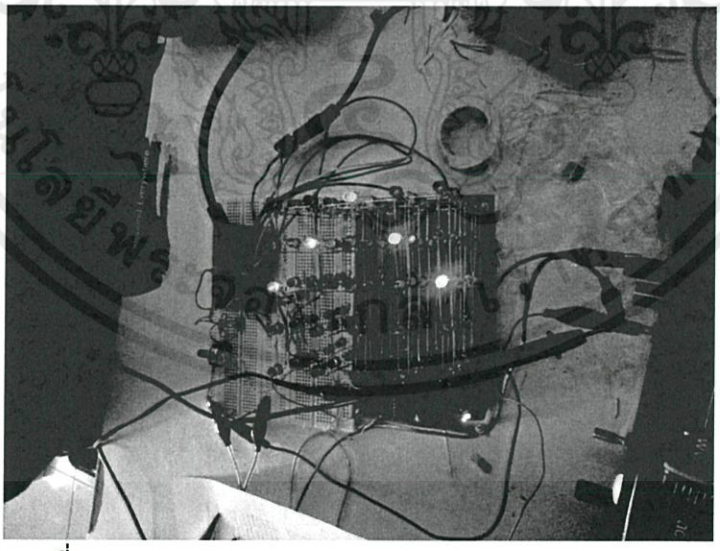
บทที่ 4
การทดลองและผลการทดลอง

-output of band pass filter 100 Hz



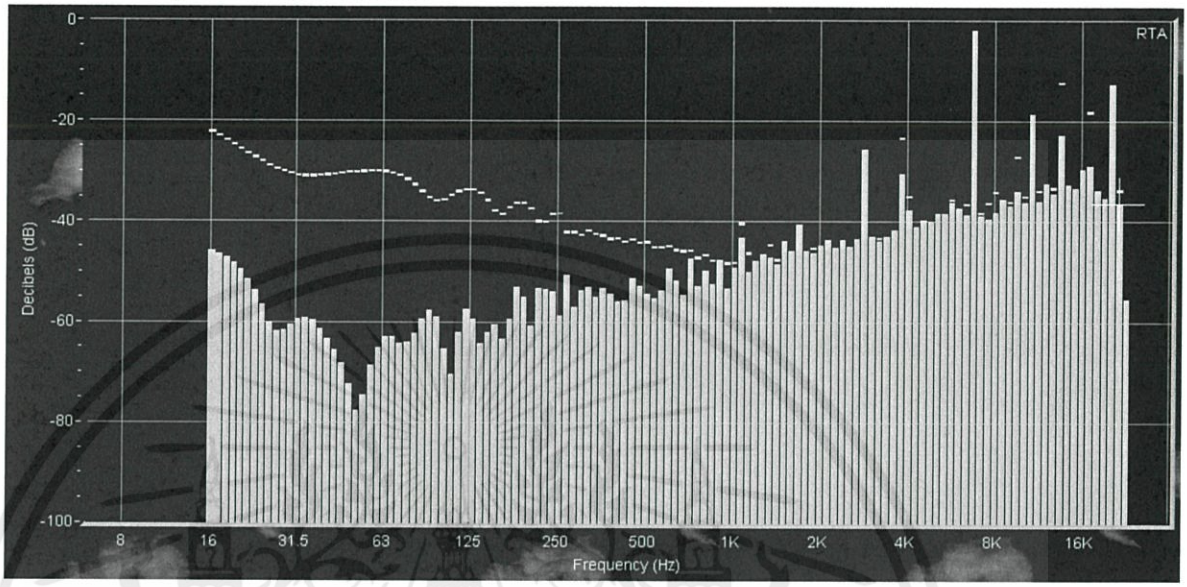
ภาพที่ 4.1 output of band pass filter 100 Hz

-การแสดงผลของหลอด LED Low pass frequency



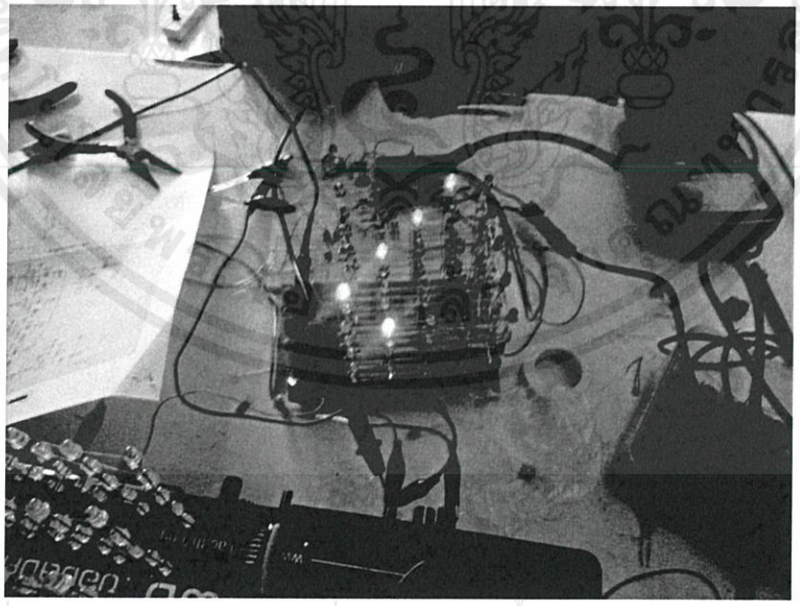
ภาพที่ 4.2 การแสดงผลของหลอด LED Low pass frequency

-output of band pass filter 10KHz



ภาพที่ 4.3 output of band pass filter 10 KHz

-การแสดงผลของหลอด LED High pass Frequency



ภาพที่ 4.4 การแสดงผลของหลอด LED High pass frequency

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1สรุปผลการดำเนินงาน

จากการทดลอง LED cube ผลการทดลองที่ได้เป็นไปตามที่ผู้จัดทำได้ออกแบบไว้ วงจรสามารถทำงานได้ตามคำสั่งที่เขียนให้ไมโครคอนโทรลเลอร์ และตอบสนองต่อสียงเพลง จากการตรวจสอบสัญญาณที่บริเวณต่างๆ ปรากฏว่ามีสัญญาณเป็นไปตามทฤษฎีแต่ยังมีสัญญาณรบกวนอยู่บ้าง

5.2สิ่งที่ได้รับจากการทำปริญญานิพนธ์

5.2.1 ทำให้ได้เรียนรู้การทำงานและพื้นฐานการเขียนคำสั่งไมโครคอนโทรลเลอร์ และวงจรแบนพาสฟิลเตอร์

5.2.2 ฝึกให้เรียนรู้การทำงานเป็นขั้นตอน

5.2.3 ฝึกความสามัคคีในการทำงานกับผู้อื่น

5.2.4 ฝึกความรับผิดชอบ ต่องานที่ได้รับมอบหมาย

5.2.5 ได้ชิ้นงานที่เป็นต้นแบบและสามารถนำไปใช้งานได้จริง

5.3ปัญหาที่พบ

5.3.1 ผู้ทดลองใช้แหล่งจ่ายไฟเกิน ทำให้ไอซีขับหลอด LED เกิดความเสียหาย

5.3.2 ปัญหาเกี่ยวกับการเขียนโค้ดแอสเซมบลี

5.3.3 ปัญหาเรื่องสัญญาณที่ออกจากแบนพาสฟิลเตอร์ที่จะนำมาควบคุมทรานซิสเตอร์ขับหลอด

หนังสืออ้างอิง

[1] วัชรินทร์ เคารพ. 2546. คู่มือการทดลอง PIC16F877 และ PIC18F458. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร: สำนักพิมพ์อีทีทีม, 2546.

[2] สมบูรณ์ เนียมกล่ำ. เรียนรู้และประยุกต์ใช้งาน PIC Microcontroller. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร: สำนักพิมพ์ เอดิสันแพรสโปรดักส์, 2549.

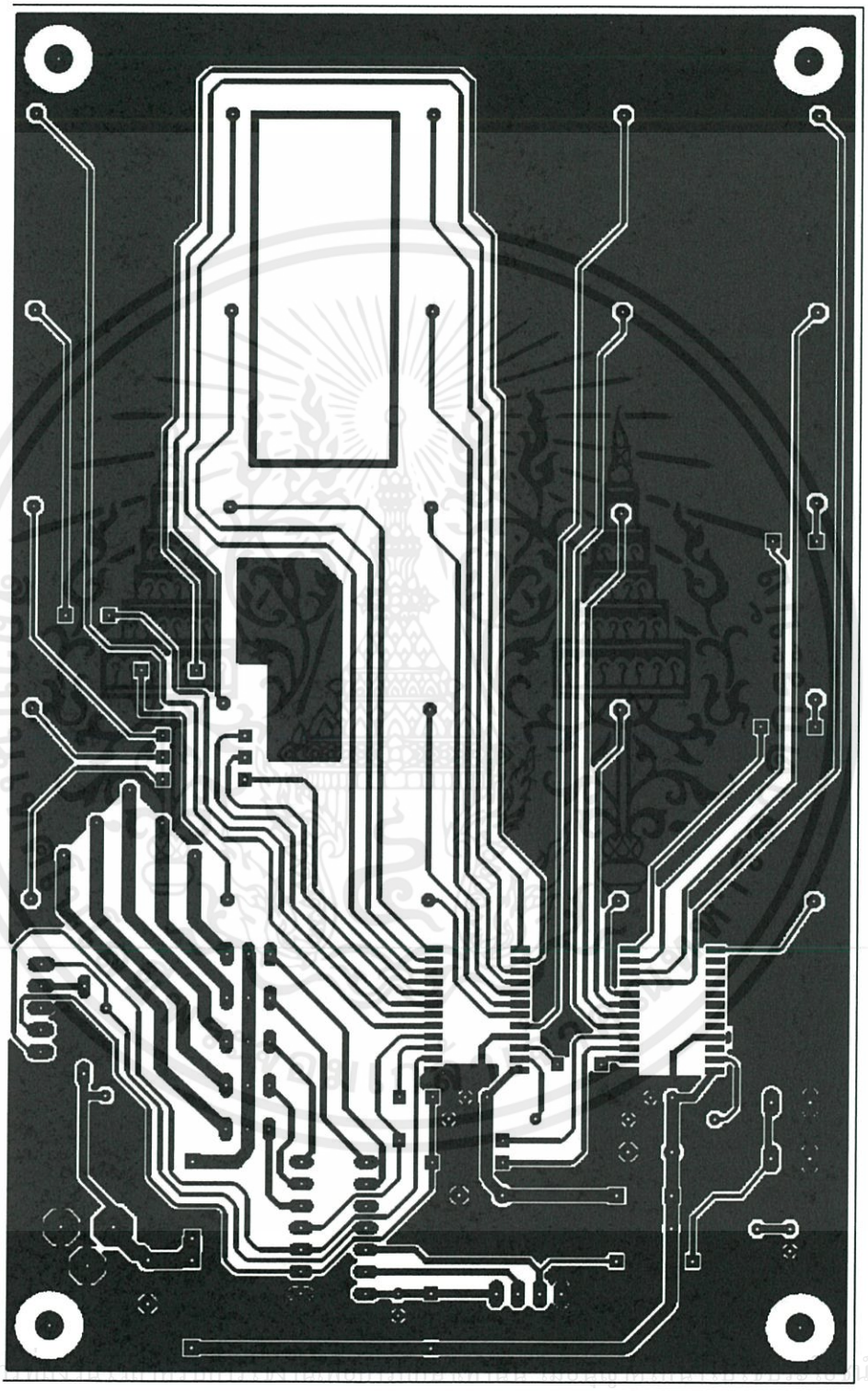
[3] พิชัย ภักดีพานิชเจริญ. คู่มือการออกแบบจงจรกรองสัญญาณความถี่.กรุงเทพมหานคร: สำนักพิมพ์ ฟิสิกส์เซนเตอร์.

[4] ESRChannelWU. 2011. 8x8x8 LED Cube. [Online]. Avilable: http://www.youtube.com/watch?v=85_wCuAPO-o.html

[5] PPOEPVLIEG . 2010. Music LED Box – DIY. [Online]. Avilable: http://www.youtube.com/watch?v=HPhm8VTCWpg.html

ภาคผนวก

ภาพลายPCBวงจร

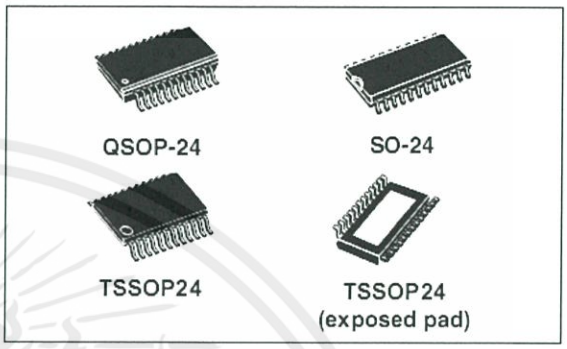


Datasheet

- STP16DP05

Features

- Low voltage power supply down to 3 V
- 16 constant current output channels
- Adjustable output current through external resistor
- Short and open output error detection
- Serial data IN/Parallel data OUT
- 3.3 V micro driver-able
- Output current: 5-100 mA
- 30 MHz clock frequency
- Available in high thermal efficiency TSSOP exposed pad
- ESD protection 2.5 kV HBM, 200 V MM



Description

The STP16DP05 is a monolithic, low voltage, low current power 16-bit shift register designed for LED panel displays. The device contains a 16-bit serial-in, parallel-out shift register that feeds a 16-bit D-type storage register. In the output stage, sixteen regulated current sources were designed to provide 5-100 mA constant current to drive the LEDs.

The STP16DP05 features open and short LED detections on the outputs.The STP16DP05 is backward compatible with STP16C/L596.The detection circuit checks 3 different conditions that can occur on the output line: short to GND, short to V_O or open line.

The data detection results are loaded in the shift register and shifted out via the serial line output.

The detection functionality is implemented without increasing the pin count number, through a secondary function of the output enable and latch pin (DM1 and DM2 respectively), a dedicated logic sequence allows the device to enter or leave from detection mode. Through an external resistor, users can adjust the STP16DP05 output current, controlling in this way the light intensity of LEDs, in addition, user can adjust LED's brightness intensity from 0% to 100% via $\overline{OE}$/DM2 pin.

The STP16DP05 guarantees a 20 V output driving capability, allowing users to connect more LEDs in series. The high clock frequency, 30 MHz, makes the device suitable for high data rate transmission. The 3.3 V voltage supply is well useful for applications that interface any 3.3V micro. Compared with a standard TSSOP package, the TSSOP exposed pad increases heat dissipation capability by a 2.5 factor.

Table 1. Device summary

Order codes	Package	Packaging
STP16DP05MTR	SO-24 (tape and reel)	1000 parts per reel
STP16DP05TTR	TSSOP24 (tape and reel)	2500 parts per reel
STP16DP05XTTR	TSSOP24 exposed pad (tape and reel)	2500 parts per reel
STP16DP05PTR	QSOP-24	2500 parts per reel

-2SD667

2SD667, 2SD667A

Absolute Maximum Ratings (Ta = 25°C)

Item	Symbol	2SD667	2SD667A	Unit
Collector to base voltage	V_{CBO}	120	120	V
Collector to emitter voltage	V_{CEO}	80	100	V
Emitter to base voltage	V_{EBO}	5	5	V
Collector current	I_C	1	1	A
Collector peak current	$i_{C(peak)}$	2	2	A
Collector power dissipation	P_C	0.9	0.9	W
Junction temperature	Tj	150	150	°C
Storage temperature	Tstg	–55 to +150	–50 to +150	°C

Electrical Characteristics (Ta = 25°C)

		2SD667			2SD667A				
Item	Symbol	Min	Typ	Max	Min	Typ	Max	Unit	Test conditions
Collector to base breakdown voltage	$V_{(BR)CBO}$	120	—	—	120	—	—	V	I_C = 10 µA, I_E = 0
Collector to emitter breakdown voltage	$V_{(BR)CEO}$	80	—	—	100	—	—	V	I_C = 1 mA, R_{BE} = ∞
Emitter to base breakdown voltage	$V_{(BR)EBO}$	5	—	—	5	—	—	V	I_E = 10 µA, I_C = 0
Collector cutoff current	I_{CBO}	—	—	10	—	—	10	µA	V_{CB} = 100 V, I_E = 0
DC current transfer ratio	h_{FE1}*1	60	—	320	60	—	200		V_{CE} = 5 V, I_C = 150 mA*2
	h_{FE2}	30	—	—	30	—	—		V_{CE} = 5 V, I_C = 500 mA*2
Collector to emitter saturation voltage	$V_{CE(sat)}$	—	—	1	—	—	1	V	I_C = 500 mA, I_B = 50 mA*2
Base to emitter voltage	V_{BE}	—	—	1.5	—	—	1.5	V	V_{CE} = 5 V, I_C = 150 mA*2
Gain bandwidth product	f_T	—	140	—	—	140	—	MHz	V_{CE} = 5 V, I_C = 150 mA*2
Collector output capacitance	Cob	—	12	—	—	12	—	pF	V_{CB} = 10 V, I_E = 0, f = 1 MHz

Notes: 1. The 2SD667 and 2SD667A are grouped by h_{FE1} as follows.

2. Pulse test

	B	C	D
2SD667	60 to 120	100 to 200	160 to 320
2SD667A	60 to 120	100 to 200	

-PIC16F688

High-Performance RISC CPU:

- Only 35 instructions to learn:
  - All single-cycle instructions except branches
- Operating speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 125 kHz
  - Software tunable
  - Two-Speed Start-Up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control option
- Enhanced Low-Current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with weak pull-up or input only pin
- Programmable code protection
- High-Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years

Low-Power Features:

- Standby Current:
  - 50 nA @ 2.0V, typical
- Operating Current:
  - 11 µA @ 32 kHz, 2.0V, typical
  - 220 µA @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 µA @ 2.0V, typical

Peripheral Features:

- 12 I/O pins with individual direction control:
  - High-current source/sink for direct LED drive
  - Interrupt-on-change pin
  - Individually programmable weak pull-ups
  - Ultra Low-Power Wake-up
- Analog Comparator module with:
  - Two analog comparators
  - Programmable On-chip Voltage Reference (CV_{REF}) module (% of V_{DD})
  - Comparator inputs and outputs externally accessible
- A/D Converter:
  - 10-bit resolution and 8 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Timer1 Gate (count enable)
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator if INTOSC mode selected
- Enhanced USART Module:
  - Supports RS-485, RS-232, and LIN 1.2
  - Auto-Baud Detect
  - Auto-wake-up on Start bit
- In-Circuit Serial Programming™ (ICSP™) via two pins

-RC4558

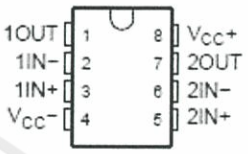
DUAL GENERAL-PURPOSE OPERATIONAL AMPLIFIER

Check for Samples: RC4558

FEATURES

- Continuous Short-Circuit Protection
- Wide Common-Mode and Differential Voltage Ranges
- No Frequency Compensation Required
- Low Power Consumption
- No Latch-Up
- Unity-Gain Bandwidth . . . 3 MHz Typ
- Gain and Phase Match Between Amplifiers
- Low Noise . . . 8 nV/$\sqrt{Hz}$ Typ at 1 kHz

D, DGK, P, PS, OR PW PACKAGE
(TOP VIEW)



DESCRIPTION/ORDERING INFORMATION

The RC4558 device is a dual general-purpose operational amplifier, with each half electrically similar to the µA741, except that offset null capability is not provided.

The high common-mode input voltage range and the absence of latch-up make this amplifier ideal for voltage-follower applications. The device is short-circuit protected, and the internal frequency compensation ensures stability without external components.

Table 1. ORDERING INFORMATION

T_A	PACKAGE[1]		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	MSOP/VSSOP – DGK	Reel of 2500	RC4558DGKR	YR_[2]
	PDIP – P	Tube of 50	RC4558P	RC4558P
	SOIC – D	Tube of 75	RC4558D	RC4558
		Reel of 2500	RC4558DRG3	
	SOP – PS	Reel of 2000	RC4558PSR	R4558
	TSSOP – PW	Tube of 150	RC4558PW	R4558
		Reel of 2000	RC4558PWR	
–40°C to 85°C	MSOP/VSSOP – DGK	Reel of 2500	RC4558IDGKR	YS_[2]
	PDIP – P	Tube of 50	RC4558IP	RC4558IP
	SOIC – D	Tube of 75	RC4558ID	R4558I
		Reel of 2500	RC4558IDR	
	TSSOP – PW	Tube of 150	RC4558IPW	R4558I
		Reel of 2000	RC4558IPWR	

(1) Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.
(2) The actual top-side marking has one additional character that designates the assembly/test site.