Project 13 DAC

โจทย์

สร้างวงจรและเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์ PIC16F88 เพื่อสร้างสัญญาณแอนะล็อกเป็นคลื่นไซน์ที่มีความถี่ 10 Hz โดยใช้สัญญาณดิจิทัลที่มีความละเอียด 10 บิต

ขั้นตอนการปฏิบัติ

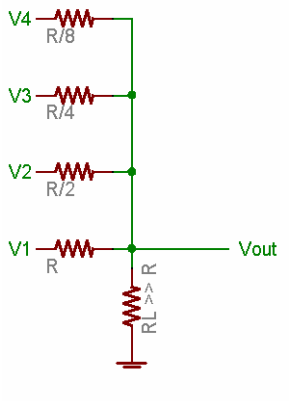
1) การออกแบบวงจร

วงจรแปลงสัญญาณดิจิทัลเป็นสัญญาณแอนะล็อก หรือที่เราเรียกว่า Digital to Analog Converter (DAC) สามารถสร้างขึ้นเองได้ง่าย โดยใช้เพียงตัวต้านทานหลายตัวต่อเข้าด้วยกันเป็นวงจร ซึ่งมีสองวงจร ได้แก่

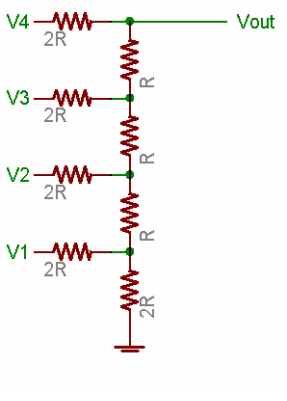
- แบบรวมกระแส (weighted resistor)
- แบบ R-2R ladder

รูปที่ 1 แสดงโครงสร้างของวงจร DAC แบบรวมกระแสและแบบ R-2R สำหรับสัญญาณดิจิทัลขนาด 4 บิต วงจรแบบรวมกระแสมีข้อดีคือ สามารถทำการคำนวณค่าเอาท์พุทได้ง่ายกว่าและใช้จำนวนตัวต้านทานน้อยกว่าแบบ R-2R ladder แต่ว่าต้องใช้ตัวต้านทานหลายขนาด (ได้แก่ R, R/2, R/4, …) ในขณะที่วงจรแบบ R-2R ใช้ตัวต้านทานเพียงสองขนาดคือ R และ 2R เท่านั้น จึงทำให้มีความสะดวกในแง่ของการจัดหา รายละเอียดของวงจรทั้งสองแบบนี้ได้สามารถอ่านเพิ่มเติมได้จากเว็บไซท์

http://www.elecnet.chandra.ac.th/learn/tipntrick/atd/default.htm



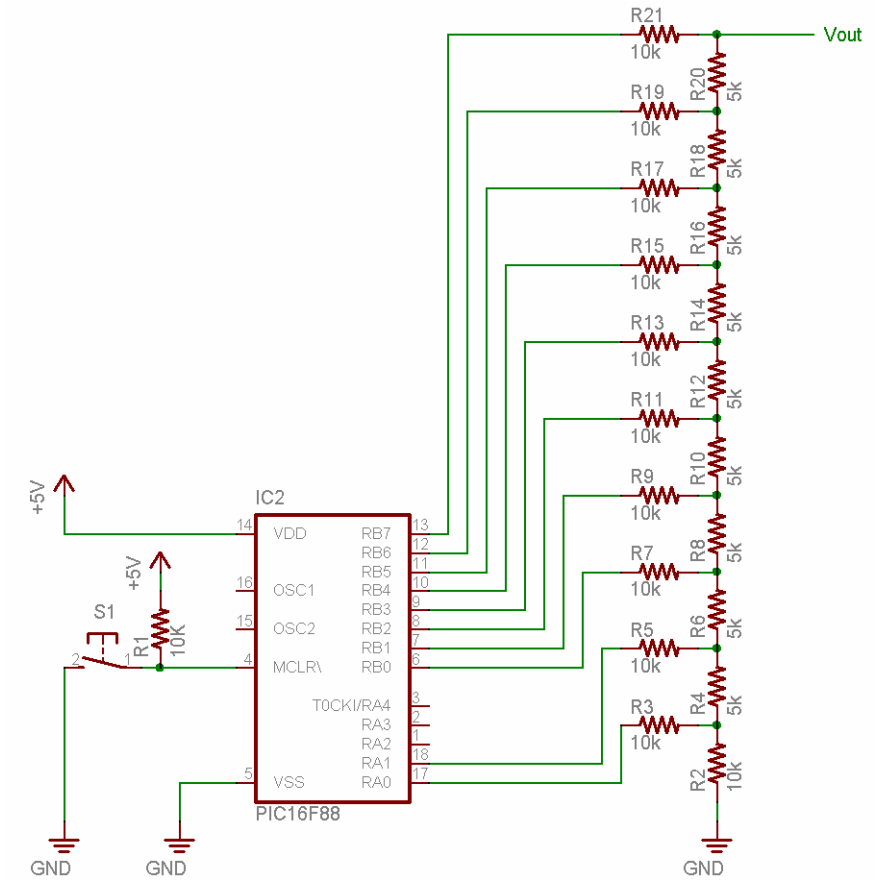
แบบรวมกระแส



แบบ R-2R ladder

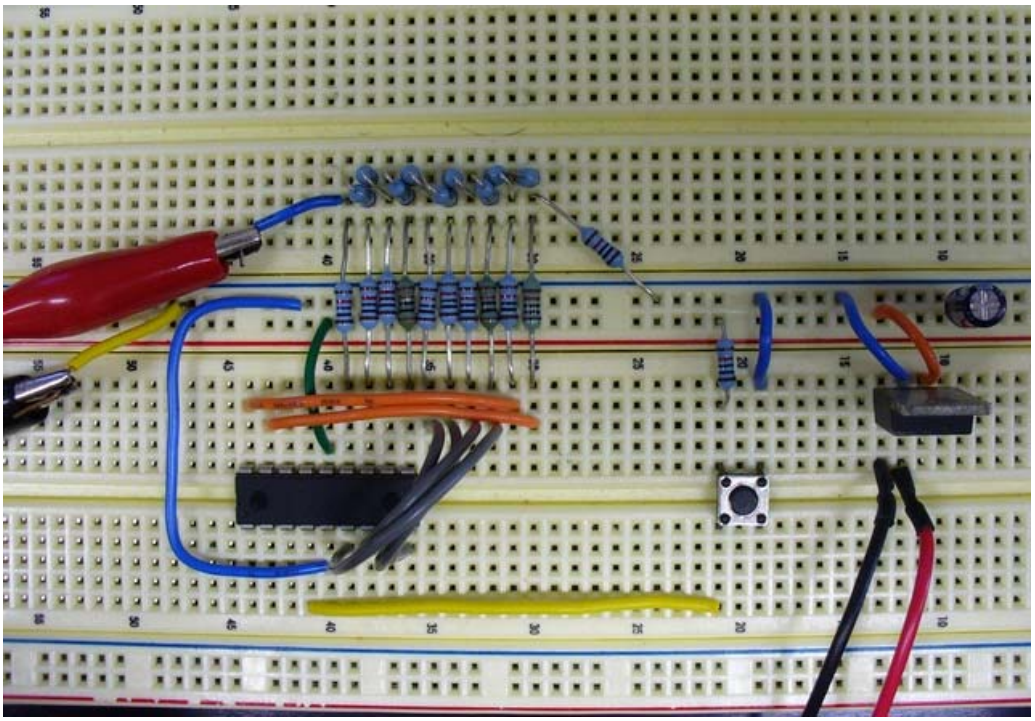
รูปที่ 1 โครงสร้างของวงจร DAC แบบรวมกระแสและแบบ R-2R

ในโพรเจกต์นี้เราเลือกใช้วงจรแบบ R-2R สำหรับสัญญาณดิจิทัลขนาด 10 บิต โดยเลือกต่อสัญญาณจากพอร์ต PORTB:PORTB<0:1> ของ PIC16F88 ดังแสดงในรูปที่ 2



รูปที่ 2 แผนผังวงจรสำหรับสร้างสัญญาณแอนะล็อก

2) การต่อวงจร



รูปที่ 3 รูปแสดงการต่อวงจร

3) การเขียนโปรแกรม

ในการสร้างสัญญาณคลื่นไซน์ เราสามารถใช้การคำนวณฟังก์ชัน sin() ได้เนื่องจากเราสามารถใช้ math library ซึ่งคอมไพเลอร์ได้จัดไว้ให้ได้ แต่ว่าการคำนวณฟังก์ชันไซน์เป็นการคำนวณแบบ floating point ซึ่งใช้เวลาในการประมวลผลมาก อีกวิธีการหนึ่งก็คือการคำนวณฟังก์ชันไซน์ไว้ล่วงหน้าจากโปรแกรมภายนอก แล้วทำการบันทึกค่าไว้ในรูปของตารางเอาไว้เรียกดูภายหลัง ซึ่งเราเรียกตารางในลักษณะนี้ว่าเป็น lookup table

เราสามารถสร้าง lookup table ของฟังก์ชัน sin() ที่มีขนาดเท่ากับ 100 โดยใช้โปรแกรม Excel ดังแสดงในรูปที่ 4 คอลัมน์ A เป็นค่าอันดับจาก 0-100 (จริงๆ แล้วใช้เพียงค่า 0-99 เท่านั้น ส่วนค่า 100 เอาไว้แสดงให้ดูว่าจะมีค่าเท่ากับที่ค่า 0 เท่านั้น) แล้วทำการสร้างค่า B ซึ่งเป็นจากการแบ่ง 0 ถึง 2π เป็น 100 จุด ซึ่งคำนวณได้จาก

$$B = A \cdot 2\pi / 100$$

ค่า C ได้จากการคำนวณฟังก์ชัน sin() และแปลงให้คำตอบอยู่ในช่วงจาก 0 ถึง 1023 (สำหรับ 10 บิต) ซึ่งคำนวณได้จาก

$$C = \left(\sin\left(B\right)+1\right)\cdot 1023/2$$

และค่า D ได้จากการปัดเศษค่า C ให้กลายเป็นเลขจำนวนเต็ม เพื่อให้สามารถจัดเก็บในรูปตัวแปร unsigned int ได้

A	B	C	D
0	0.00	511.50	512
1	0.06	543.62	544
2	0.13	575.61	576
3	0.19	607.35	607
4	0.25	638.70	639
…	…	…	…
98	6.16	447.39	447
99	6.22	479.38	479
100	6.28	511.50	512

รูปที่ 4 Lookup table **สำหรับฟังก์ชัน** sin()

<u>โปรแกรม project13.c</u>

```
#include<pic.h>
#include<stdio.h>

#define PI 3.14159265358979
#define XMAX 3481
#define XMIN 1518
#define YMAX 3471
#define YMIN 1479

__CONFIG(UNPROTECT & LVPDIS & BOREN & MCLREN & PWRTDIS & WDTDIS & INTIO);

void main (void)
{
   // Setup oscillator
   IRCF2   =  1;
   IRCF1 =  1;
   IRCF0 =  1;              // IRCF<2:0> = 111: internal RC oscillator frequency = 8 MHz
   SCS1  =  0;
   SCS0  =  0;              // SCS<1:0> = 00: Oscilator mode defined by FOSC<2:0>
   while(!IOFS);            // Wait for the frequency to become stable

   // Set up output ports <PORTB:RA0:RA1> (10 bits)
   ANSEL = 0x00;            // Select every bit of port A as digital I/O
   TRISB = 0x00;            // Set PortB as output
   TRISA0 = 0;              // Set port RA0 as output
   TRISA1 = 0;              // Set port RA1 as output
   PORTB = 0x00;            // Initialize PortB
   RA0 = 0;                 // Initialize RA0
   RA1 = 0;                 // Initialize RA1

   // Setup Timer1
   T1CON = 0b00100100;      // Prescale = 1:4, Clock source = Fosc/4 (1 count = 2 us)
```

```
   TMR1IF = 0;              // Clear Timer1 interrupt flag
   TMR1IE = 1;              // Enable Timer1 interrupt
   PEIE = 1;                // Enable Peripheral interrupt
   GIE = 1;                 // Enable Global interrupt
   TMR1IF = 1;              // Force to enter interrupt subroutine for the first time

   while(1)
   {

   }
}

static void interrupt isr(void)
{
   unsigned int aout;             // Analog output
   unsigned int i;
   const unsigned int sintable[] = {     512, 544, 576, 607, 639,
                                         670, 700, 729, 758, 786,
                                         812, 838, 862, 884, 906,
                                         925, 943, 960, 974, 987,
                                         998, 1007, 1014, 1019, 1022,
                                         1023, 1022, 1019, 1014, 1007,
                                         998, 987, 974, 960, 943,
                                         925, 906, 884, 862, 838,
                                         812, 786, 758, 729, 700,
                                         670, 639, 607, 576, 544,
                                         512, 479, 447, 416, 384,
                                         353, 323, 294, 265, 237,
                                         211, 185, 161, 139, 117,
                                         98, 80, 63, 49, 36,
                                         25, 16, 9, 4, 1,
                                         0, 1, 4, 9, 16,
                                         25, 36, 49, 63, 80,
                                         98, 117, 139, 161, 185,
                                         211, 237, 265, 294, 323,
                                         353, 384, 416, 447, 479
                                   }; // Lookup table for sine function (100 points, range
from 0 to 1023)

   if(TMR1IF)
   {
      TMR1IF = 0; // Clear Timer1 interrupt flag
      TMR1ON = 0; // Turn off Timer1
      TMR1H = 0xFE;
      TMR1L = 0x0C;
      // Set Timer1 = 65,536 - 500 = 65,036 = 0xFE0C
      // This results in interrupt every 500 counts or 1 ms)
      TMR1ON = 1;    // Turn on Timer1

      if (i < 98) i++; else i = 0;
      aout = sintable[i];
      PORTB = aout >> 2;
      RA0 = aout & 0x0001;
      RA1 = aout & 0x0010;
   }
}
```

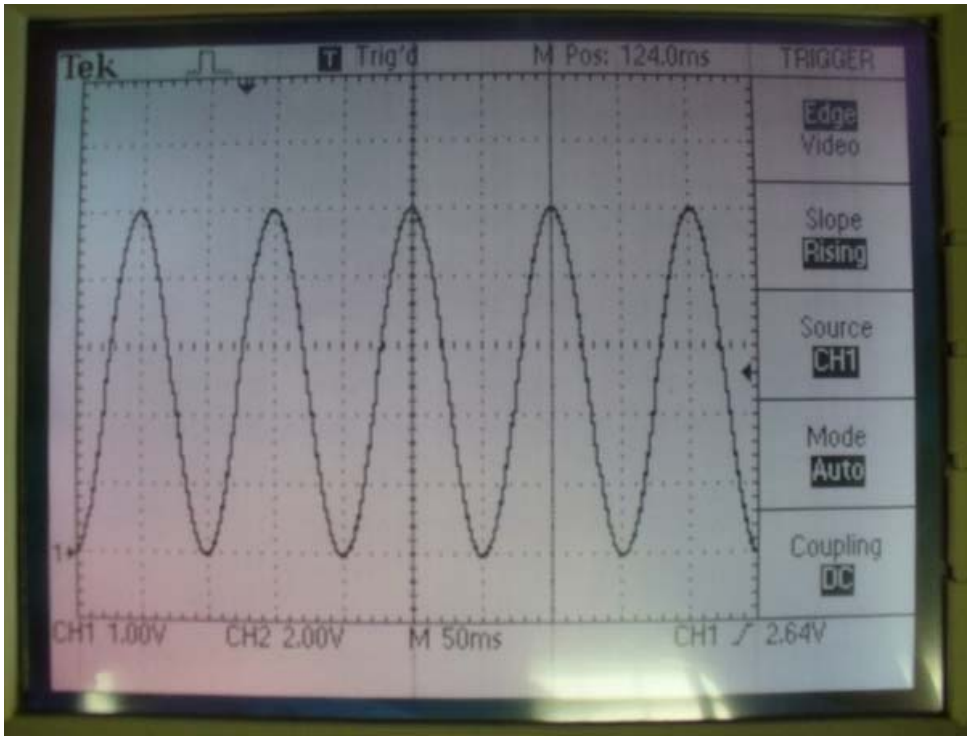
คำอธิบายโปรแกรม

เนื่องจากเรามีค่าของฟังก์ชัน sin() จัดเก็บไว้ใน lookup table จำนวน 100 ค่า (0 - 2π) ดังนั้นหากต้องการความถี่ 10 Hz หรือคาบเท่ากับ 100 ms แต่ละค่าจึงต้องมีระยะเวลาห่างกัน 100ms/100 = 1ms

โปรแกรม project13.c มีการใช้ Timer0 interrupt เพื่อให้สัญญาณที่ได้ออกมามีความแม่นยำ โดยในเราตั้งค่า Timer0 ให้มีการ interrupt ทุกๆ 1 ms เมื่อมีการ interrupt เกิดขึ้น โปรแกรมจะทำการอ่านค่าจาก lookup table ซึ่งเก็บไว้ในตัวแปรอาเรย์ชื่อ sintable[] ขอให้สังเกตว่าในการประกาศตัวแปรนี้มีคำว่า const นำหน้าเพื่อบอกให้รู้ต้องการที่จะเก็บค่า lookup table ไว้ในส่วน ROM ที่เก็บโปรแกรม เพื่อไม่ให้ใช้เนื้อที่ในส่วน RAM ซึ่งมีขนาดเล็ก

4) การทดสอบการทำงาน

เราสามารถใช้ oscilloscope ในการแสดงสัญญาณแอนะล็อกที่สร้างขึ้นจากโปรแกรม ผลการทดลองแสดงดังรูปที่ 5 จะเห็นว่าคลื่นไซน์ที่ได้มีคาบเท่ากับ 2 ช่อง หรือ 100 ms หรือความถี่เท่ากับ 10 Hz ตามต้องการ



รูปที่ 5 รูปแสดงผลการทดลอง