东南大学

《微机实验及课程设计》

课程设计报告

参数可调波形发生器

姓　　名： 邹滨阳　　**学　　号：** 08022305

同组姓名： 邓卓霖　　**学　　号：** 08022405

专　　业： 自动化　　**实 验 室：** 金智楼

实验时间： **2024** 年 **5** 月 **7** 日　　**报告时间：** **2024** 年 **5** 月 **20** 日

评定成绩：　　**审阅教师：**

一， 小组总体完成情况

表 1 小组人员及分工

姓名	学号	实际完成的设计/开发/实验工作，承担任务占比例
邹滨阳	08022305	60%（共同商量程序整体框架，负责生成三角函数，三次函数波形的函数）
邓卓霖	08022405	40%（共同商量程序整体框架，负责读取参数和交互界面的设计）

二， 实验目的要求

1.实验目的

(1) 了解波形发生器的基本原理；

(2) 进一步了解 8255 可编程并行 I/O 接口、0832DA 转换、0809AD 转换、8253 可编程计 数器等芯片的电路设计、硬件连接、软件编程方法；

(3) 掌握多芯片综合应用构成复杂系统的设计、调试方法，锻炼在原理与接口综合应用方面 分析问题与解决问题的能力。

2.实验要求

编制一个参数在线可调的波形发生程序，由 D / A 输出,构成参数在线可调的波形发生器， 并用示波器观察波形。

(1) 函数波形可选 f(t)=asin(bt)，其中 a 、 b 两个参数可调,且调节效果明显；

(2) 参数调节采用键盘实现参数调整；

(3) 由开关选择标准信号波形输出（a=2V,b=100πRAD/s,即 50Hz 正弦波）。

3.提高要求

产生周期性三次可调函数 $f(x) = at^3 + bt^2 + ct + d$

三， 总体原理与设计

1，实现方法

通过调用 int 中断在显示器上显示提示信息提醒用户进行操作，包括产生何种波形、 调节 a/b 的方式等，然后将得到的用户信息存储到指定的寄存器中，接着调用输出子程序产生波形。 一些具体的子功能包括：

(1) 设置提示信息并在进程相应位置调用显示；

(2) 若是通过键盘接收 a、b 信息需对 a、b 大小进行判断，防止溢出；

(3) 将输入的 a、b 根据十六进制 ASCII 码值转化成对应的数并存储在对应变量中；

(4) 通过等时间间隔读取正弦数字量表并通过 D/A 转化后产生连续正弦波；

(5) 通过 a 值大小改变输出正弦波的幅值；

(6) 通过 b 值大小改变输出正弦波的频率；

(7) 设计开关以实现不同波形的产生；

(8) 设计交互实现退出该程序。

2，a、b 参数获取

首先，是读取 a 的值。我们设计了一个 input_a 函数，在这个函数中，首先将要显示的字符串 "string1" 的地址加载到 DX 寄存器中，然后设置 AH 寄存器以调用 MS-DOS 的“显示字符串”功能，

来显示提示信息。接下来，通过调用 21h 号中断来显示这个字符串。然后，AH 寄存器被设置为 01h，以调用 MS-DOS 的“读取字符”功能。通过再次调用 21h 号中断，程序等待用户输入一个字符。由于 ASCII 码中 '0' 的值是 30h，所以程序通过将读取到的字符与 '0' 进行减法来获取参数 a 的数值，并将结果保存到变量 a 中，最后返回。

然后就是读取 b 的值，我们同样设计了一个 input_b 函数，首先它通过将要显示的字符串 "string3" 的地址加载到 DX 寄存器中，并通过调用 21h 号中断来显示这个字符串，也就是 b 的提示信息。然后，它加载另一个字符串 "input" 的地址，并设置 AH 寄存器以调用 MS-DOS 的“缓冲输入”功能，这允许用户输入一行字符串并将其存储在指定的缓冲区中。通过再次调用 21h 号中断，程序等待用户输入。随后，它初始化一个变量 right 为 1，用于标记输入的正确性。接着，清除了进位标志位和各个寄存器。程序通过访问输入缓冲区中的数据，并将 ASCII 码转换为对应的数值。在这个过程中，它通过乘以 10 来逐位转换字符串中的数字，最终得到参数 b 的数值。如果输入有误（例如输入了非数字字符），则将 right 标记设为 0。最后，将结果保存到变量 output 中，并返回。

3.产生标准正弦波

正弦波的生成实质上是通过在非常短的时间内连续输出一系列离散电压值来模拟正弦信号的图像。首先，我们需要定义这个“极短的时间段”，这里采用了软件定时的方法。通过使用指令 nop 进行延时，使得在延时期间输出的电压保持恒定不变。其次，我们需要确定每个时间段内输出的电压值大小。在这里，电压值的大小是通过读取预先定义的正弦波数字量表实现的。代码中的 create 过程通过循环遍历正弦波数据表，将每个数据值输出到 D/A 转换器，以模拟正弦波形。在每次输出后，通过调用 delay 过程进行延时，然后继续下一个数据的输出，直到完成一组数据的输出。

通过 8 位 D/A 转换芯片 0832 将查表得到 的的数字量转换模拟量并输出，其关系为：

$U = -\frac{N}{256} \times U_{REF}$其中 N 为数字量大小，U 为输出电压，UREF 为参考电压。

4.

上述提到的正弦数字量表即对标准正弦波进行离散周期采样得到，可以用 Python 进行行编程计算获得：

```
sin db 080h, 084h, 088h, 08Ch, 090h, 094h, 098h, 09Bh
       db 09Fh, 0A3h, 0A7h, 0ABh, 0AFh, 0B2h, 0B6h, 0BAh
       db 0BDh, 0C1h, 0C4h, 0C7h, 0CBh, 0CEh, 0D1h, 0D4h
       db 0D7h, 0DAh, 0DDh, 0DFh, 0E2h, 0E5h, 0E7h, 0E9h
       db 0EBh, 0EEh, 0F0h, 0F1h, 0F3h, 0F5h, 0F6h, 0F8h
       db 0F9h, 0FAh, 0FBh, 0FCh, 0FDh, 0FEh, 0FEh, 0FFh
       db 0FFh, 0FFh, 0FFh, 0FFh, 0FFh, 0FFh, 0FEh, 0FEh
       db 0FDh, 0FDh, 0FCh, 0FBh, 0FAh, 0F8h, 0F7h, 0F6h
       db 0F4h, 0F2h, 0F1h, 0EFh, 0EDh, 0EAh, 0E8h, 0E6h
       db 0E3h, 0E1h, 0DEh, 0DBh, 0D9h, 0D6h, 0D3h, 0D0h
       db 0CCh, 0C9h, 0C6h, 0C2h, 0BFh, 0BBh, 0B8h, 0B4h
       db 0B1h, 0ADh, 0A9h, 0A5h, 0A1h, 099h, 095h, 092h
       db 08Eh, 08Ah, 086h, 082h, 07Dh, 079h, 075h, 071h
       db 06Dh, 069h, 066h, 062h, 05Eh, 05Ah, 056h, 052h
       db 04Eh, 04Bh, 047h, 044h, 040h, 03Dh, 039h, 036h
       db 033h, 02Fh, 02Ch, 029h, 026h, 024h, 021h, 01Eh
       db 01Ch, 019h, 017h, 015h, 012h, 010h, 00Eh, 00Dh
       db 00Bh, 009h, 008h, 007h, 005h, 004h, 003h, 002h
       db 002h, 001h, 001h, 000h, 000h, 000h, 000h, 000h
```

```python
import numpy as np

# 定义函数 y=128sin(x)+128
1 个用法
def waveform(x):
    return 127.5 * np.sin(x) + 127.5

# 生成等间隔的 x 值
x_values = np.linspace( start: 0, 2*np.pi,  num: 198)

# 计算 y 值并转换为整数
y_values = np.round(waveform(x_values)).astype(int)

# 将 y 值转换为十六进制并输出
hex_values = [hex(y)[2:] for y in y_values]

print(hex_values)
```

```
db 000h, 001h, 001h, 002h, 003h, 004h, 005h, 006h
db 007h, 009h, 00Ah, 00Ch, 00Eh, 00Fh, 011h, 014h
db 016h, 018h, 01Ah, 01Dh, 020h, 022h, 025h, 028h
db 02Bh, 02Eh, 031h, 034h, 038h, 03Bh, 03Eh, 042h
db 045h, 049h, 04Dh, 050h, 054h, 058h, 05Ch, 060h
db 064h, 067h, 06Bh, 06Fh, 073h, 077h, 07Bh, 07Fh ;正弦波数据
```

4.实现波形变换

我们通过改变变量 a 和 b 来调整生成的波形。在主过程中，变量 a 被用于调整波形的振幅。通过将当前读取的正弦波数据乘以 a 的值，可以调整输出的电压大小，进而影响波形的振幅。如果 a 的值较大，那么每个正弦波数据乘以 a 后输出的电压也会相应增大，从而增加波形的振幅；反之，如果 a 的值较小，则波形的振幅会减小。

另外，变量 b 控制着波形的时间轴。在主过程中，每个正弦波数据输出到 D/A 转换器后，会调用延时过程，然后处理下一个数据。输出的结果会直接反映在波形的变化速度和频率上。通过增大 b，就会使得输出间隔较小，则波形的变化速度会较快，频率会增加；反之，减小 b 会导致输出间隔较大，则波形的变化速度会减慢，频率会降低。

此外在编写 delay 函数当中，我们发现当 nop 比较少时候，时间延迟随 nop 数量变化不成线性变化，而 nop 比较多时，时间延迟随 nop 的数量变化成线性变化，说明除了 nop 还有其他影响时间变化的因素，而只有在 nop 比较多的时候才可以尽量减少这个误差。

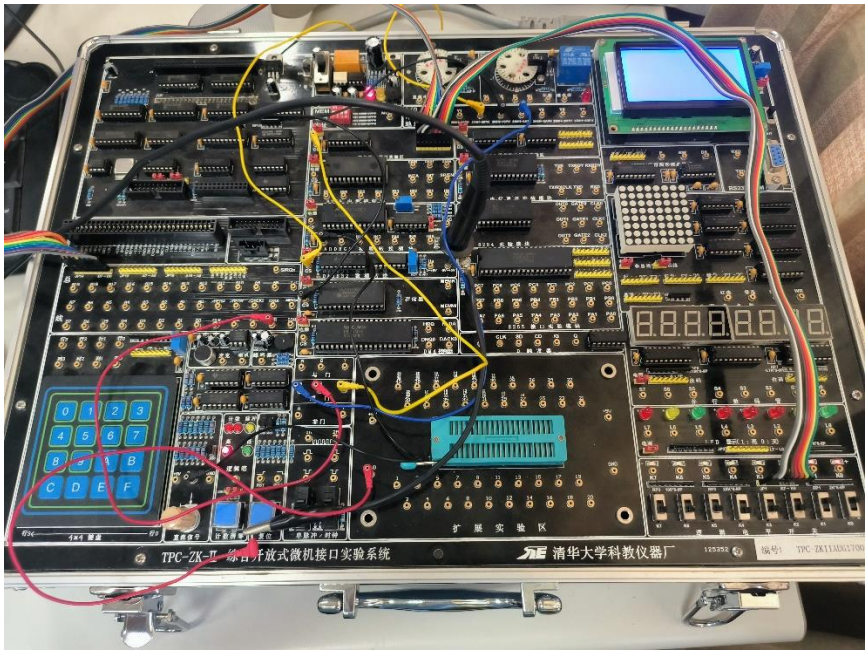
因此，通过调整这两个变量的值，可以有效地控制生成的波形的振幅和频率，从而实现不同形态的波形输出。

5. 功能切换设计

利用开关读取相关数据，如果读取到的结果是 01，说明是参数可调的三角函数产生器，如果是 10，说明是参数可调的三次函数产生器。

6. 硬件连接设计

如图所示：

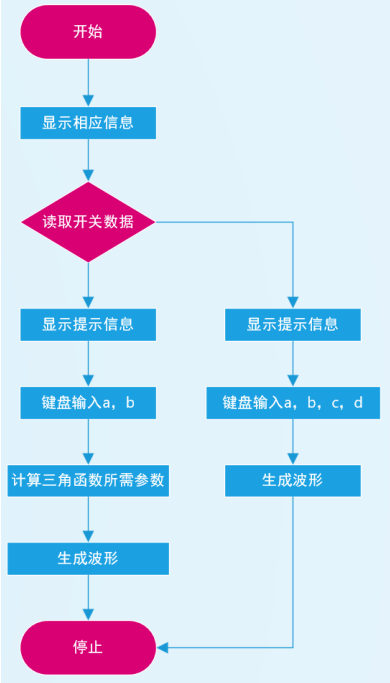


利用开关读取数据，并利用简单输入输出进行读取，从而决定工作方式，接着利用数模转换模块进行输入。

四， 程序流程和编程实现

1.主程序模块

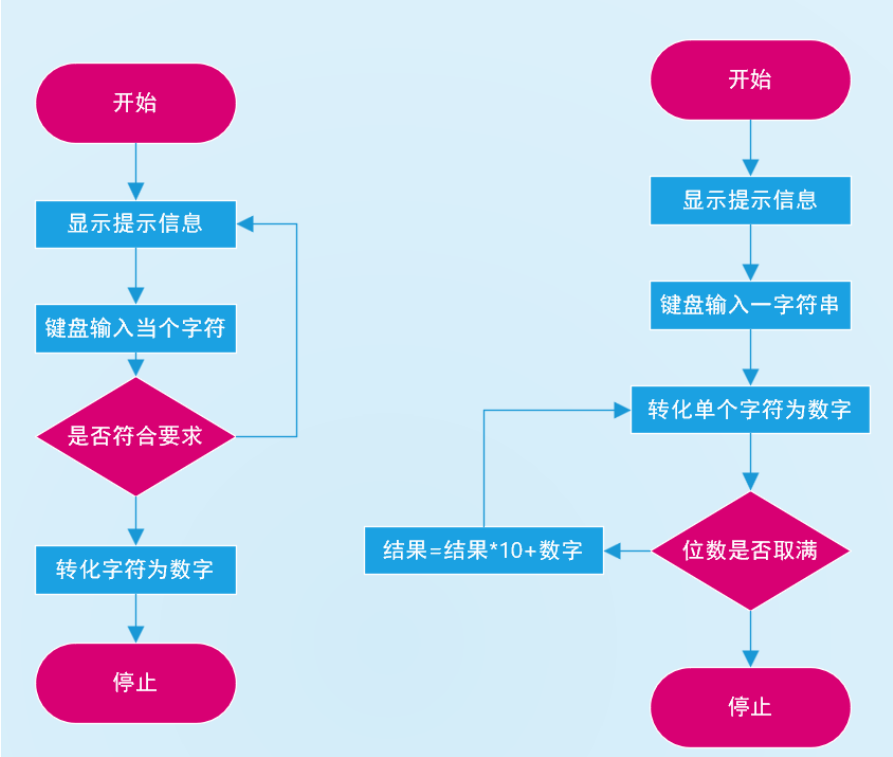
（1）流程图



（2）代码解读

```
start:    mov ax,stacks          ; 将 'stacks' 常量或标签的地址移动到寄存器 ax
          mov ss,ax              ; 将 ax 寄存器的值移动到堆栈段寄存器 ss
          mov ax,data            ; 将 'data' 常量或标签的地址移动到寄存器 ax
          mov ds,ax              ; 将 ax 寄存器的值移动到数据段寄存器 ds
          mov dx,ls244           ; 将常量 ls244 移动到寄存器 dx（用于输入端口地址）
          in al,dx               ; 从 dx 指定的端口地址读取数据到寄存器 al
          cmp al,01H            ; 比较 al 寄存器的值与 01H（十六进制的 1）
          jne exit1              ; 如果 al 不等于 01H，则跳转到标签 exit1
          call input_a           ; 调用子程序 input_a
          call input_b           ; 调用子程序 input_b
          call cumulate          ; 调用子程序 cumulate 计算三角函数需要参数
          call create            ; 调用子程序 create 生成三角函数
          exit1:                ; 标签 exit1
          cmp al,02H            ; 比较 al 寄存器的值与 02H（十六进制的 2）
          call input_a1          ; 调用子程序 input_a1
          call input_b1          ; 调用子程序 input_b1
          call input_c1          ; 调用子程序 input_c1
          call input_d1          ; 调用子程序 input_d1.键盘输入模块
          call sancehans         ; 调用子程序 三次函数产生器
```

（1）流程图



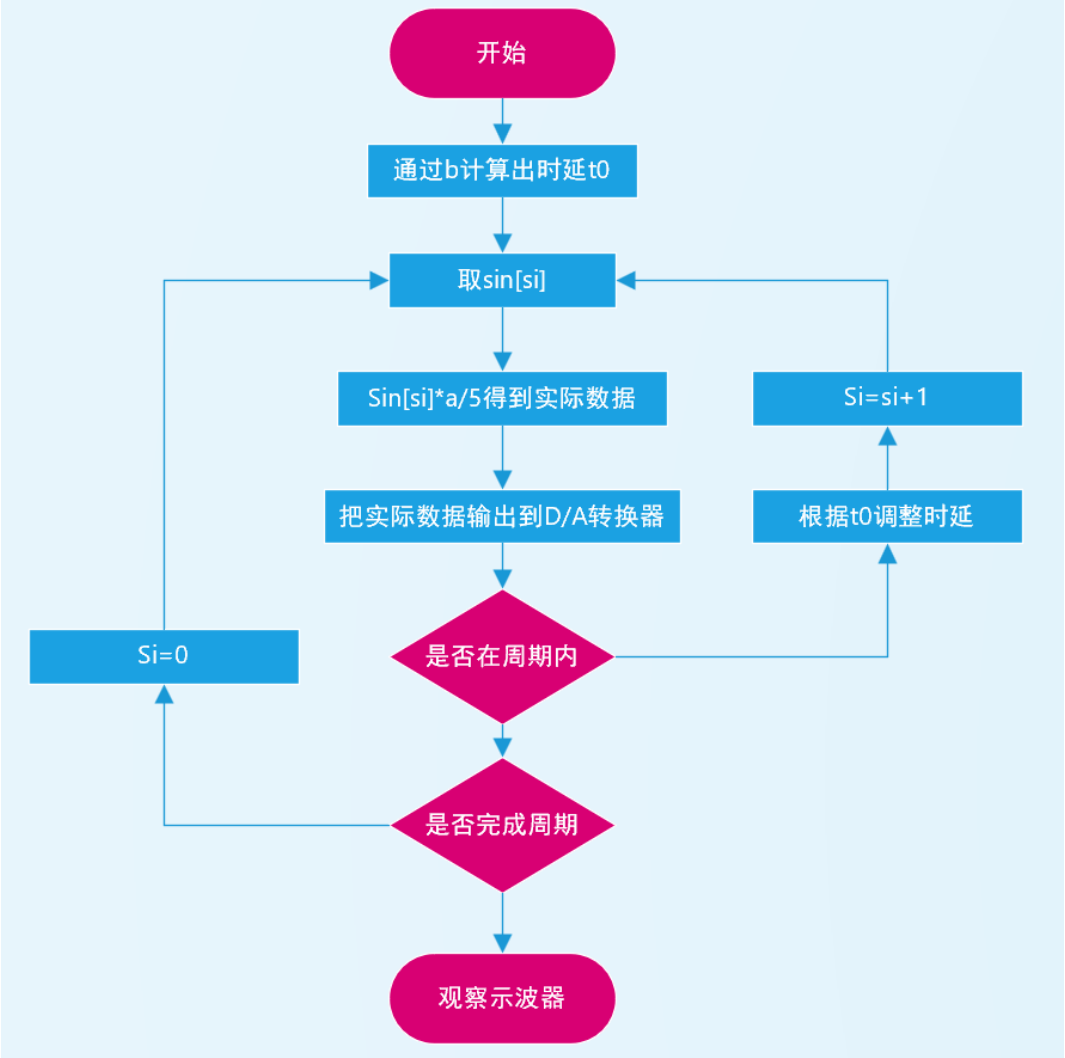
（2）代码解读

```
input_a proc near
    mov dx,offset string1    ; 将 string1 的偏移地址移动到 DX 寄存器中（假设 string1 是以
null 结尾的字符串）
    mov ah,09h               ; 将 AH 设置为 09h，以便使用 DOS 中断 21h 显示字符串
    int 21h                  ; 调用 DOS 中断以显示 DX 中的字符串（即 string1）
    mov ah,01h               ; 将 AH 设置为 01h，以从标准输入读取字符
    int 21h                  ; 调用 DOS 中断以从标准输入读取字符，并将其存储在 AL 寄存器中
    sub al,30h               ; 通过减去 30h 将 ASCII 字符转换为其对应的数字值
    mov a,al                 ; 将数字值存储在变量 'a' 中
    ret                      ; 返回该过程

input_a endp

input_b proc near
    mov dx,offset string3    ; 将 string3 的偏移地址移动到 DX 寄存器中（假设 string3 是以
null 结尾的字符串）
    mov ah,09h               ; 将 AH 设置为 09h，以便使用 DOS 中断 21h 显示字符串
    int 21h                  ; 调用 DOS 中断以显示 DX 中的字符串（即 string3）
    mov dx,offset input      ; 将输入缓冲区的偏移地址移动到 DX 寄存器中
    mov ah,0ah               ; 将 AH 设置为 0Ah，以从标准输入读取一行输入
    int 21h                  ; 调用 DOS 中断以读取一行输入，并将其存储在 DS:DX 处的缓冲区中
（即 input）
    mov right,1              ; 将右标志初始化为 1（假设成功）
```

```
    clc                     ; 清除进位标志
    xor ax,ax               ; 清除 AX 寄存器
    xor bx,bx               ; 清除 BX 寄存器
    xor dx,dx               ; 清除 DX 寄存器
    mov cl,input[1]         ; 将输入的长度（不包括回车符）移动到 CL 寄存器中
    mov si,2                ; 将 SI 寄存器初始化为 2（索引，用于访问输入缓冲区中的字符）
again:
    mov bl,input[si]        ; 将输入缓冲区中的 ASCII 字符移动到 BL 寄存器中
    cmp bl,30h              ; 将 BL 与 ASCII 值 '0' 进行比较
    jb error1               ; 如果低于（'jb' - 如果低于则跳转），表示字符不是数字，跳转到
error1
    cmp bl,39h              ; 将 BL 与 ASCII 值 '9' 进行比较
    ja error1               ; 如果高于（'ja' - 如果高于则跳转），表示字符不是数字，跳转到
error1
    sub bl,30h              ; 通过减去 30h 将 ASCII 字符转换为其对应的数字值
    dec cl                  ; 减少字符计数
    cmp cl,0                ; 将 CL 与 0 进行比较
    jz over                 ; 如果为零，则跳转到 over
    mov ax,1                ; 将 AX 设置为 1
    push cx                 ; 将 CX 的当前值压入堆栈
loop2:                      ; 循环的标签
    mul ten                 ; 将 AX 乘以 10
    loop loop2              ; 循环，直到 CX 变为零
    pop cx                  ; 从堆栈中恢复 CX 的值
    mul bx                  ; 将 AX 乘以 BX
    add output,ax           ; 将结果添加到输出中
    inc si                  ; 将 SI 增加以指向输入缓冲区中的下一个字符
    jmp again               ; 跳转到 again 继续处理下一个字符
over:                       ; 所有字符处理完毕时的标签
    add output,bx           ; 将最终结果添加到输出中
    jnc over1               ; 如果没有进位，则跳转到 over1
error1:                     ; 错误处理的标签
    mov right,0             ; 将右标志设置为 0 表示错误
over1:                      ; 处理结束的标签
    ret                     ; 从该过程返回
input_b endp
```

3.正弦波输出和延时模块

（1）流程图



（2）代码解读

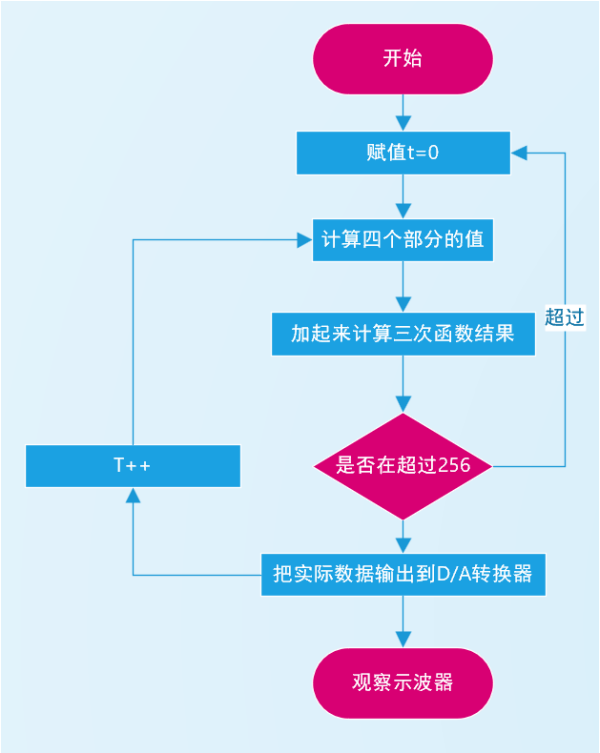
```
cumulate proc near
    push ax             ; 保存 AX 寄存器的值到栈中
    push dx             ; 保存 DX 寄存器的值到栈中
    push cx             ; 保存 CX 寄存器的值到栈中
    mov ax,chu1         ; 将变量 chu1 的值移动到 AX 寄存器中
    mov cx,chu2         ; 将变量 chu2 的值移动到 CX 寄存器中
    mul cx              ; 将 AX 和 CX 中的值相乘，结果存放在 AX 中
    div output          ; 将 AX 中的值除以 output 的值，商存放在 AX 中，余数存放在 DX 中
    mov yanshi,ax       ; 将 AX 中的值移动到变量 yanshi 中
    pop ax              ; 从栈中弹出值到 AX 寄存器中
    pop dx              ; 从栈中弹出值到 DX 寄存器中
    pop cx              ; 从栈中弹出值到 CX 寄存器中
    ret                 ; 返回
cumulate endp
create proc near
l1:    mov si,offset sin        ; 将正弦波数据的偏移地址设置为 SI
```

```
    mov bh,200                ; 设置一组输出 200 个数据
lll:    mov al,[si]             ; 将数据输出到 D/A 转换器
    push cx                   ; 保存 CX 寄存器的值到栈中
    mov ah,0                  ; 清空 AH 寄存器
    mul a                     ; 将 AL 寄存器中的值与变量 a 相乘
    mov cx,5                  ; 将 CX 寄存器设置为 5
    div cl                    ; 将 AX 寄存器中的值除以 CL 寄存器中的值
    pop cx                    ; 弹出栈中的值到 CX 寄存器
    mov dx,io0832a            ; 将 IO 端口的地址移动到 DX 寄存器
    out dx,al                 ; 将 AL 的值输出到 DX 指定的 IO 端口
    mov ah,06h                ; 将 AH 寄存器设置为 06h，用于 DOS 延时功能
    mov dl,0ffh               ; 将 DL 寄存器设置为 0xff
    int 21h                   ; 调用 DOS 中断以实现延时
    call delay                ; 调用延时函数
    inc si                    ; 取下一个数据
    dec bh                    ; 减少输出数据组数
    jnz lll                   ; 若未取完 200 个数据则转向 lll
    jmp  ll                   ; 跳转到 ll 标签，重新开始
    ret                       ; 返回

delay proc near
    mov cx,yanshi             ; 设置延时循环的次数
delay1:
    nop                       ; 空指令，用于延时
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    loop delay1               ; 循环延时
    ret                       ; 返回
```

```
delay endp
```

3.三次函数输出

（1）流程图



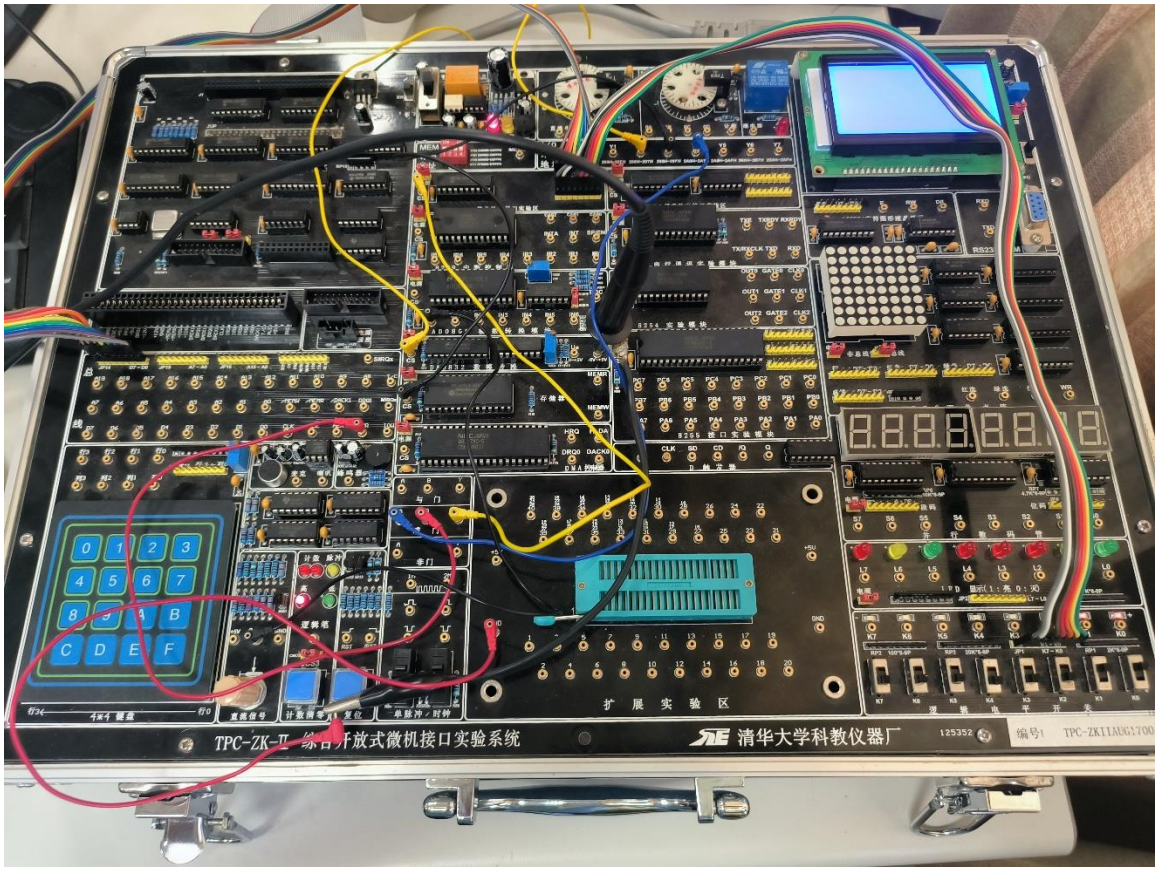
（2）代码解读

```
sancehans proc near ; 定义一个名为 sancehans 的近过程（子程序）
loop1: mov t1,0          ; 将寄存器 t1 置为 0，这通常用于循环计数器或波形的初始值
loop2:
    mov ax,t1            ; 将 t1 的值复制到 ax 寄存器
    mul t1               ; 将 ax 与 t1 相乘，结果仍在 ax 中，用于计算三次项 t1^3
    mov t2,ax            ; 将 ax 的值（即 t1^2）复制到 t2 寄存器
    mul t1               ; 再次将 ax 与 t1 相乘，得到 t1^3，结果仍在 ax 中
    mov t3,ax            ; 将 ax 的值（即 t1^3）复制到 t3 寄存器
    mov ax,t1            ; 将 t1 的值复制回 ax 寄存器
    mul c1               ; 将 ax（t1）与常数 c1 相乘，结果仍在 ax 中
    mov t1,ax            ; 将 ax 的值（即 t1*c1）复制到 t1 寄存器
    mov ax,t2            ; 将 t2 的值（即 t1^2）复制到 ax 寄存器
    mul b1               ; 将 ax（t1^2）与常数 b1 相乘，结果仍在 ax 中
    mov t2,ax            ; 将 ax 的值（即 t1^2*b1）复制到 t2 寄存器
    mov ax,t3            ; 将 t3 的值（即 t1^3）复制到 ax 寄存器
    mul a1               ; 将 ax（t1^3）与常数 a1 相乘，结果仍在 ax 中
    mov t3,ax            ; 将 ax 的值（即 t1^3*a1）复制到 t3 寄存器
    mov ax,d1            ; 将常数 d1 的值复制到 ax 寄存器
    add ax,t1            ; 将 ax（d1）与 t1 相加，得到线性项 d1 + t1*c1
    add ax,t2            ; 将 ax 与 t2 相加，得到二次项 d1 + t1*c1 + t1^2*b1
```

```
    add ax,t3              ; 将 ax 与 t3 相加，得到三次函数 d1 + t1*c1 + t1^2*b1 + t1^3*a1
    cmp ax,256             ; 比较 ax 的值与 256，用于确定是否达到输出的最大值
    ja loop1               ; 如果 ax 大于 256，则跳转到 loop1，继续循环
    mov dx,io0832a         ; 将 io0832a 的地址复制到 dx 寄存器，用于数模转换的端口地址
    out dx,al              ; 将 al 寄存器的值输出到 dx 指定的端口，进行数模转换
    inc t1                  ; 将 t1 寄存器的值增加 1，用于生成波形的下一个点
    jmp loop2               ; 无条件跳转到 loop2，继续循环生成波形
    ret                     ; 从子程序返回到调用它的程序
sancehans endp             ; 结束 sancehans 子程序的定义
```

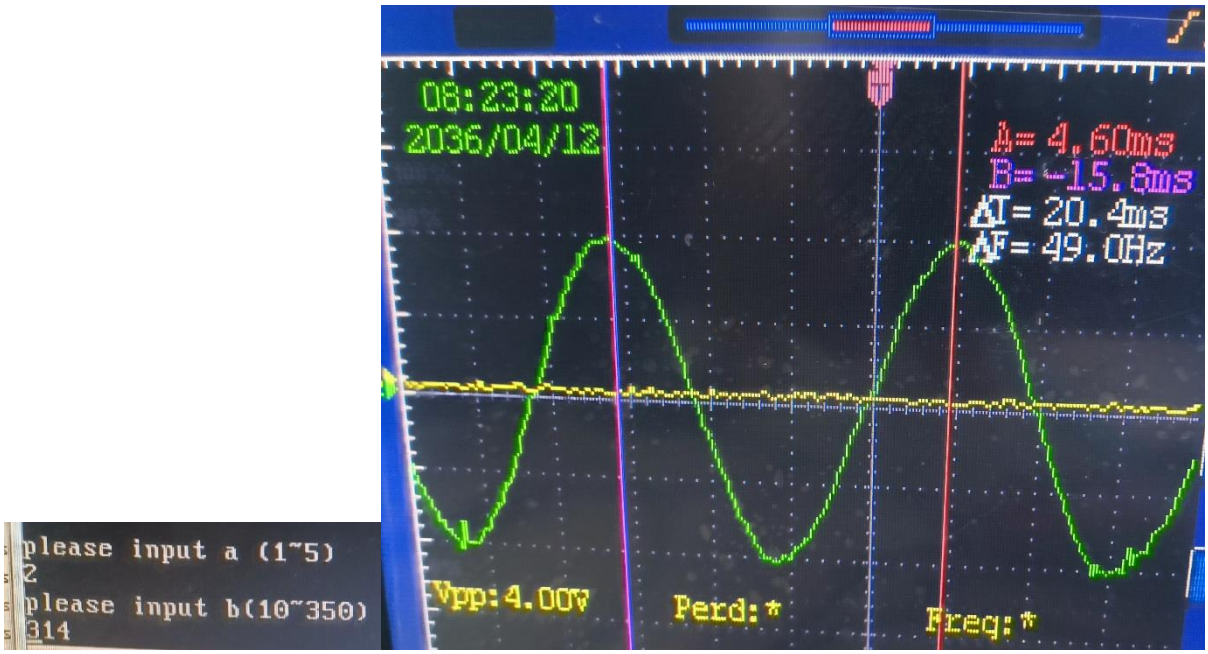
五，测试方案和测试结果

（1）实物连接图



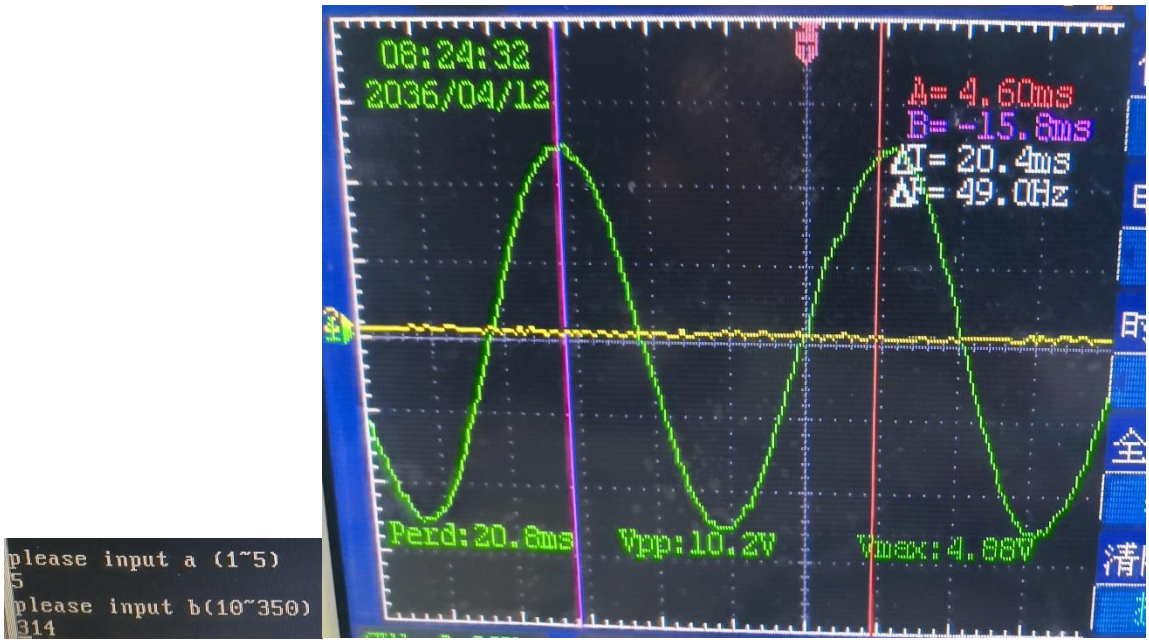
（2）正弦波测试

输入标准正弦波也就是 a=2V b=314



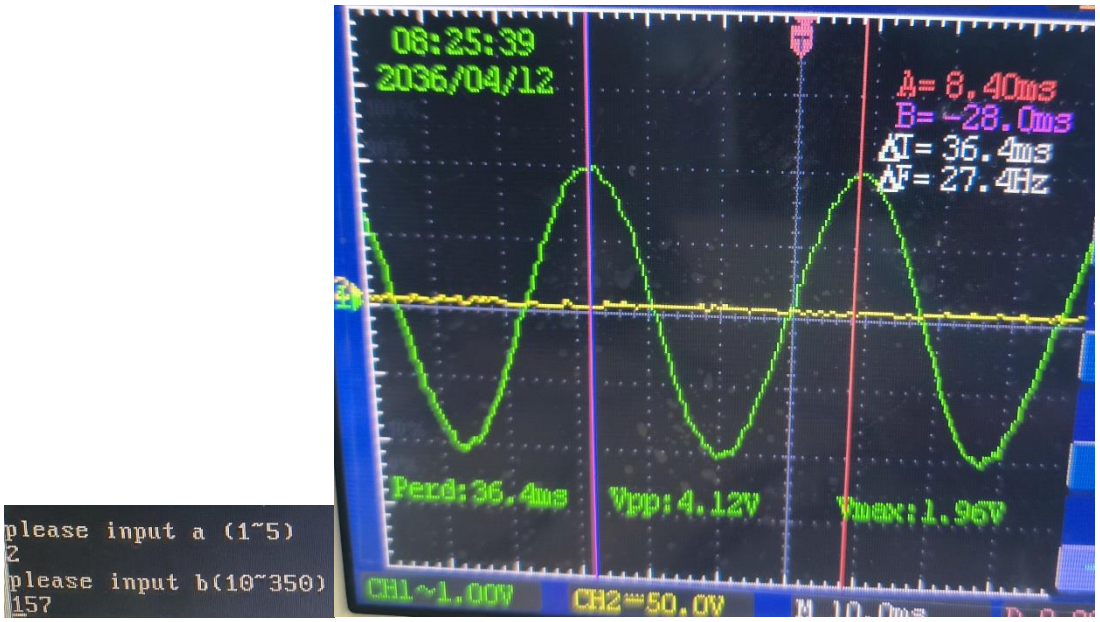
我们可以看到周期为 20ms，频率为 50Hz，峰峰值为 4V 符合标准波形的要求

调整输入 a 为 5，b 为 314



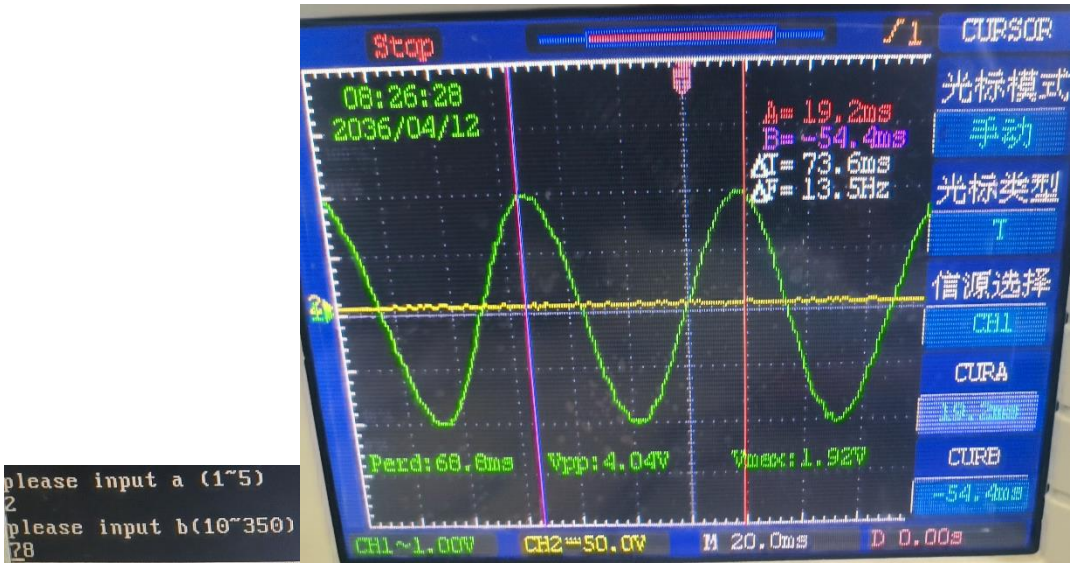
我们可以看到峰峰值为 10.2V，频率为 49Hz，可以看到调整 a 导致了非常明显的波形变化

调整输入 a 为 2，b 为 157



我们可以看到峰峰值为 4.12V，频率为 27.4Hz，与所要求的 25Hz 差别不大，可以看到调整 b 导致的频率变化基本正确。

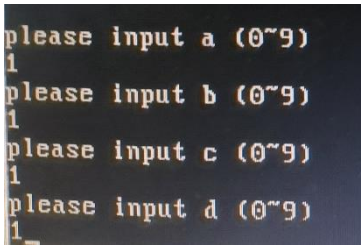
调整输入 a 为 2，b 为 78

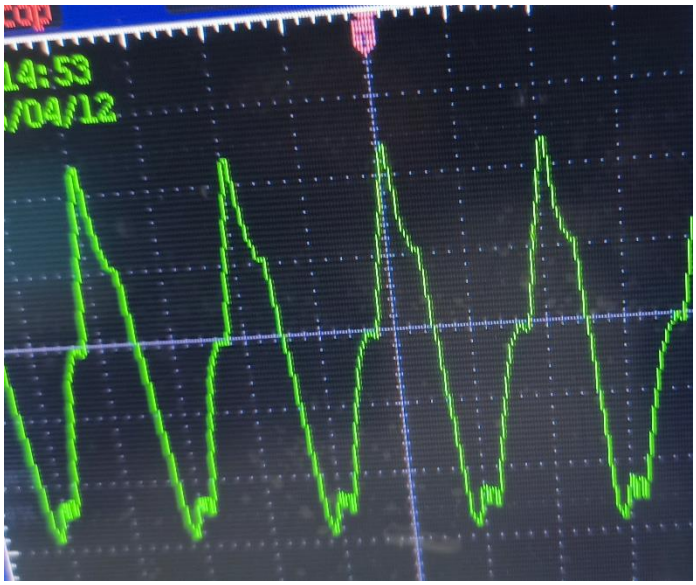


我们可以看到峰峰值为 4.04V，频率为 13.5Hz，与所要求的 12.4Hz 差别不大，也可以证明调整 b 导致的频率变化基本正确。

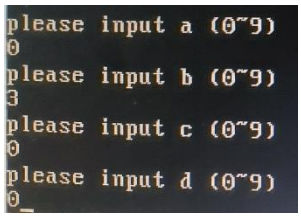
（3）三次函数测试

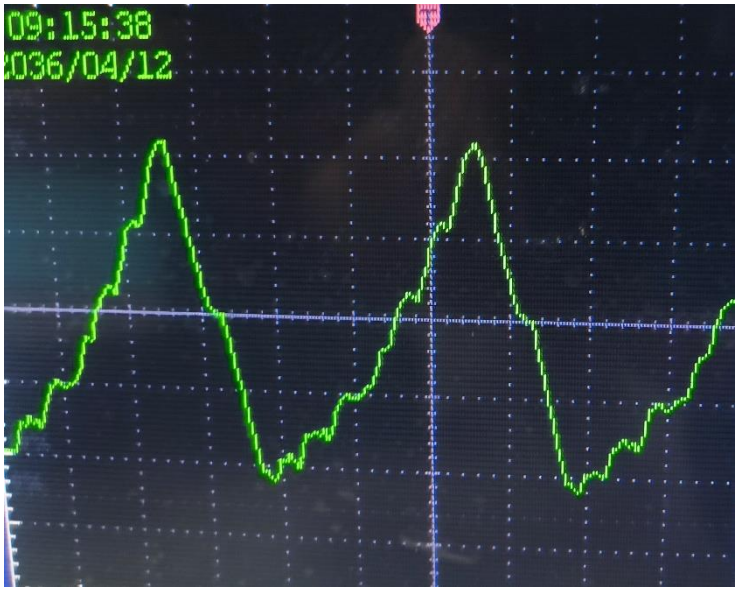
产生基本三次函数：



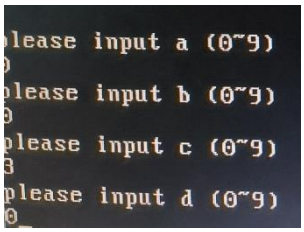


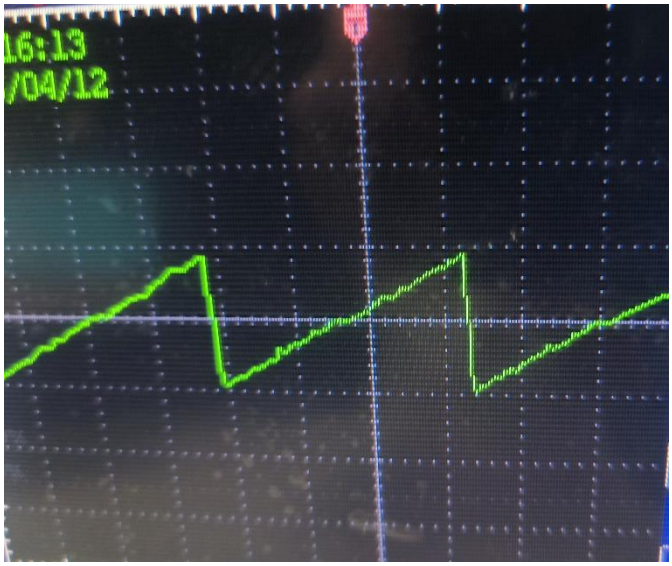
在三次函数的基础上产生二次函数





在三次函数的基础上产生一次函数





我们看到能根据我们输入的数据产生不同的波形，基本满足要求

六，分析与总结认识

通过本次实验，我们深入理解了波形发生器的基本原理，并且掌握了如何利用 8255 可编程并行 I/O 接口、0832DA 数模转换、0809AD 模数转换、8253 可编程计数器等芯片来设计电路、连接硬件以及编写相应的软件程序。实验中，我们成功实现了一个参数在线可调的波形发生程序，能够通过 D/A 转换输出并用示波器观察到波形，这不仅锻炼了我们的动手能力，也加强了我们分析问题与解决问题的综合能力。

实验要求我们编制的程序能够生成特定函数波形，如 f(t)=asin(bt)，并允许参数 a 和 b 的在线调节。通过键盘输入，我们可以调整波形的幅度和频率，这一过程涉及到 ASCII 码值的转换和参数的验证，确保了程序的健壮性。我们还设计了特定的提示信息和错误处理机制，以提高用户体验并防止非法输入。

在硬件连接方面，我们通过开关读取数据，并利用简单的输入输出操作来决定工作方式，然后通过数模转换模块进行输入，这一设计体现了系统的灵活性和可扩展性。在软件实现方面，我们采用了模块化的设计方法，每个模块负责特定的功能，如参数获取、波形生成、延时控制等，这样的设计使得程序易于理解和维护。

测试结果表明，我们的波形发生器能够根据输入参数正确地产生正弦波和三次函数波形，且参数调整对波形的影响明显。无论是标准正弦波的生成，还是参数调整后波形的变化，或是三次函数波形的产生，我们的系统都表现出了良好的性能和稳定性。

总结来说，本次实验不仅加深了我们对波形发生器原理的理解，而且提升了我们在硬件电路设计、软件编程、系统集成和测试方面的实践能力。通过实验，我们学会了如何将理论知识应用到实际问题中，如何解决实际问题，并在此过程中获得了宝贵的经验和技能。这些经验将对我们未来的学习和工作产生积极的影响。