西北工業大學

嵌入式系统及应用
开放性实验报告

得 分：

题　　目：	波形发生器
专业名称：	信息安全
学生姓名：	Higor
班　　级：	09061401
时　　间：	2017-05-25

波形发生器

一、总体设计方案

1.1 题目描述

本次嵌入式开放性实验的要求为：使用 ARM 开发板上硬件资源，编程实现波形发生器功能，通过按键 1 可切换波形的形态（正弦波、三角波、方波），通过按键 2 可改变波形的频率。

1.2 系统组成

按题目要求，本次开放性实验需要实现的为基于 LPC2378 的波形发生器，使之能够产生正弦波、三角波、方波等信号波形，并可通过按键调整波形类型与频率等。结合 LPC2378 的片上资源，确定系统组成如图 1-1 所示。该设计方案相对于题目要求增加了波形可调节的参数，并增加了一个 OLED 显示模块，用于显示当前输出的波形与波形参数，使系统的人机交互过程变的方便和谐。

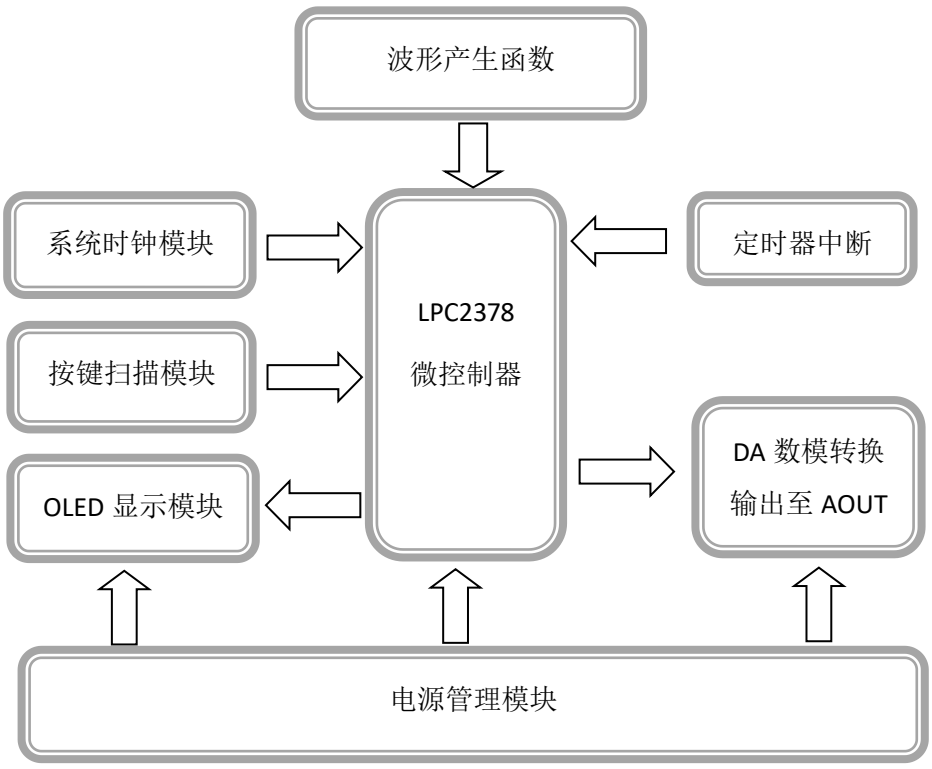

图 1-1 系统整体组成框图

1.3 各组成模块功能

1.3.1 波形产生函数模块

波形产生函数模块的功能是产生设定频率、幅值、偏移、占空比的波形，包括方波产生函数、正弦波产生函数、三角波产生函数、锯齿波产生函数。这些函数是通过编程实现的，产生的数据量将作为DA转换的输入。

1.3.2 DA模数转换模块

DA 模数转换模块功能是将波形产生函数模块产生的波形数字值转换为模拟电压量，通过不断将波形产生模块产生的值传入DA转换器，便可以产生连续的指定波形。

1.3.3 定时中断模块

定时中断的功能是产生频率比较高的周期性的定时中断，用于定时地将波形产生函数模块产生的波形数据值送入DA模数转换模块，输出到对应的模拟输出脚 AOUT。定时器中断的频率决定波形的最小分辨率，理论上定时器中断的频率越高，产生的波形的分辨率越高，波形的最大频率也会相应变大。但实际中还需要考虑控制器的响应能力，应该在控制器能够及时准确响应的基础上，尽量减小中断的周期即增大中断频率。

1.3.4 按键扫描模块

按键扫描模块是人机接口的一部分，作为人机交互的输入。通过周期性地扫描按键的状态获取用户的输入，调整系统的输出。比如改变波形类型、频率、幅值等。

1.3.5 OLED显示模块

OLED 显示模块是人机接口的一部分，作为系统状态的显示输出。将系统的状态与参数显示在一个显示屏上，结合按键使系统的人机交互过程变的十分方便和谐。

1.3.6 系统时钟模块

系统时钟模块为系统的基础且重要的模块，作为系统的时钟输入。由外部晶振和控制器内部的锁相环路PLL等组成。

1.3.7 电源管理模块

电源管理为开发板的一部分，将5V的电源供电稳压出3.3V的电源给微控制器与相关外设供电。

二、具体设计

2.1 波形产生函数

波形产生函数的目的为产生指定参数的波形，通过在定时器中断内周期性地调用波形产生函数，并将结果送入 DA 转换模块，即可在输出引脚上产生相应的波形。定时器中断的频率即波形产生函数的被调用频率将影响最终波形的频率，为了修改移植方便，需定义如说明 2-1 所示的几个配置参数：

说明 2-1 波形产生配置参数

```
#define V_REF              3.3          //芯片的 DA 参考电压
#define DA_MAX_OUT         1023.0       //DA 最大的输出值
#define CALL_FREQUENCY     10000.0      //波形生成函数被调用的频率
```

波形的相关参数是可以改变的，如频率、幅值等。因此需设置相应的参数，在调用波形产生函数时传入参数产生不同参数的波形。与设置波形相关的参数如下，它们都可以通过按键修改，并显示在 OLED 显示屏上：

说明 2-2 波形参数

```
//0：方波、1：正弦波、2：三角波、4：锯齿波
uint8_t wave_type=0;      //波形类型
float amplitude=3.3;      //波形幅值
float frequency=100;      //波形频率
float offset=0;           //波形偏置
float duty=50;            //方波占空比
```

2.1.1 方波产生函数

方波只有“高”和“低”这两个值。在产生函数内只需将自变量与周期的一半进行比较，小于输出最大值，大于输出最小值即可得到常规的方波。如图 2-1 所示。

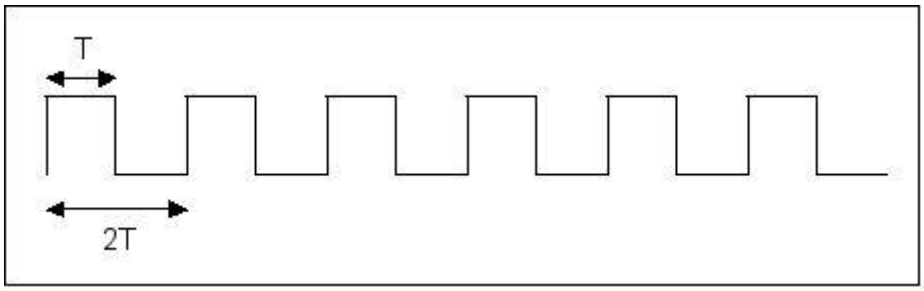

图 2-1 方波示意图

在方波基础上，方波产生函数增加了几个可调的参数，用于产生不同参数的矩形波（包括方波）。可调的参数包括频率，幅值、偏移、占空比。

最终的方波产生函数如程序 2-1 所示：

程序 2-1 方波产生函数

```
//方波
uint32_t wave_square(float frequency,\
                float amplitude,float offset,float duty)
{
    static float step = 1;
    static float x = 0;
    static uint32_t curdata;
    duty = duty/100.0;
    if(x <= CALL_FREQUENCY*duty)curdata =\
    offset/V_REF*DA_MAX_OUT+amplitude/V_REF*DA_MAX_OUT;
    else if(x >= CALL_FREQUENCY) x = 0;
    else curdata = offset/V_REF*DA_MAX_OUT;
    x += step*frequency;
    if(curdata>DA_MAX_OUT)curdata = DA_MAX_OUT;
    return curdata;
}
```

2.1.2 正弦波产生函数

正弦波信号的波形是数学上的正弦曲线。因此在产生正弦波信号可以以 sin 函数为基础，产生正弦波。如图 2-2 所示。

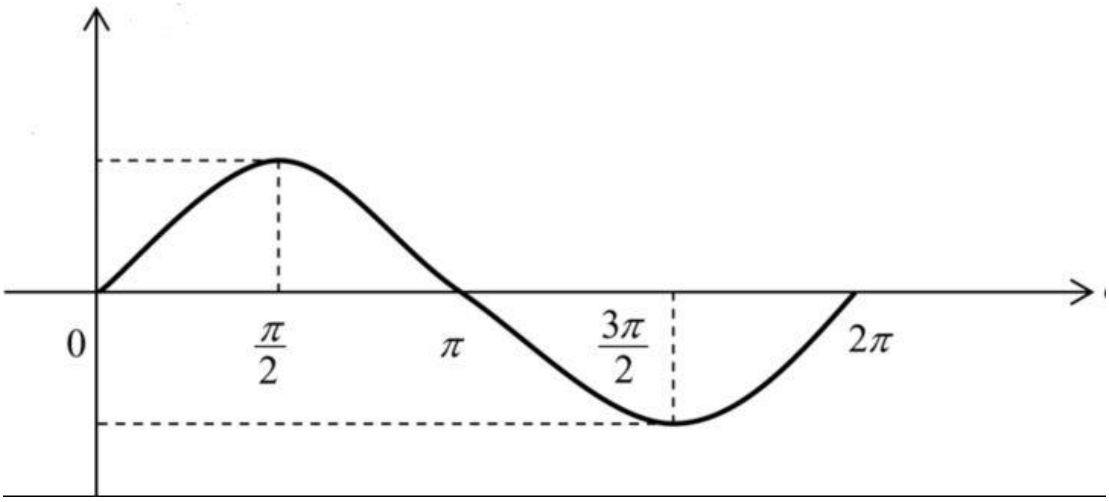

图 2-2 正弦波示意图

在正弦波产生函数中，也设置了可调参数，可调的参数包括频率，幅值、偏移。

最终的正弦波产生函数如程序 2-2 所示：

程序 2-2 正弦波产生函数

```
//正弦波
#define PI            3.1415926
uint32_t wave_sin(float frequency,\
                  float amplitude,float offset)
{
    static float step = 2.0*PI;
    static float x = 0;
    static uint32_t curdata;
    curdata = offset/V_REF*DA_MAX_OUT +
    amplitude/V_REF*(((sin(x)+1.0)/2.0)*DA_MAX_OUT);
    x += (step*frequency)/(CALL_FREQUENCY);
    if(curdata>DA_MAX_OUT)curdata = DA_MAX_OUT;
    return curdata;
}
```

2.1.3 三角波产生函数

三角波的形状形似三角形，波形值在波形的前半周期内以一定的斜率上升，后半周期又以相同大小的斜率减小。因此在两个阶段实现两个相同斜率的递增与递减便可以产生如图 2-3 所示的三角波形。

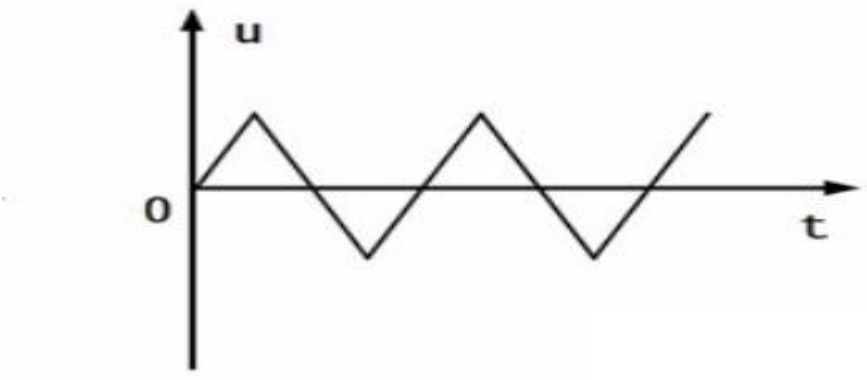

图 2-3 三角波示意图

在三角波产生函数中，也设置了可调参数，可调的参数包括频率，幅值、偏移。

最终的三角波产生函数如程序 2-3 所示：

程序 2-3 三角波产生函数

```
//三角波
uint32_t wave_triangle(float frequency,\
                       float amplitude,float offset)
{
    static float step = 1;
    static float x = 0;
    static uint32_t curdata;
    if(x <= CALL_FREQUENCY/2.0)curdata =\
    offset/V_REF*DA_MAX_OUT+amplitude/V_REF*\
         (x/(CALL_FREQUENCY/2))*DA_MAX_OUT;
    else if(x >= CALL_FREQUENCY) x = 0;
    else curdata = offset/V_REF*DA_MAX_OUT +\
         amplitude/V_REF*DA_MAX_OUT - amplitude/V_REF*\
         (x/(CALL_FREQUENCY/2)-1.0)*DA_MAX_OUT;
    x += step*frequency;
    if(curdata>DA_MAX_OUT)curdata = DA_MAX_OUT;
    return curdata;
}
```

2.1.4 锯齿波产生函数

锯齿波的形状形似锯齿，波形值在波形的周期内以一定的斜率上升，在周期末急剧减小至最小值。因此在周期内对波形值递增，在周期末急剧减至最小值便可以产生如图 2-4 所示的波形。

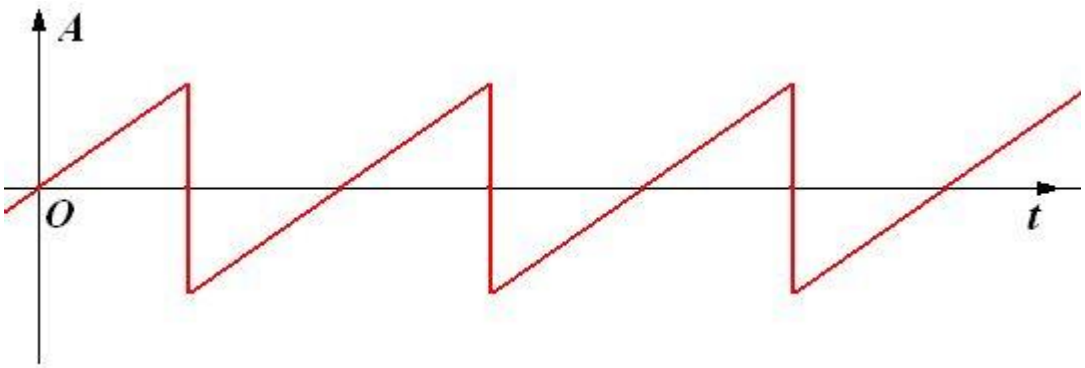

图 2-4 锯齿波示意图

在锯齿波产生函数中，也设置了可调参数，可调的参数包括频率，幅值、偏移。

最终的锯齿波产生函数如程序 2-4 所示：

程序 2-4 三角波产生函数

```
//锯齿波
uint32_t wave_sawtooth(float frequency,\
                       float amplitude,float offset)
{
    static float step = 1;
    static float x = 0;
    static uint32_t curdata;
    if(x >= CALL_FREQUENCY) x = 0;
    else curdata = offset/V_REF*DA_MAX_OUT +\
         amplitude/V_REF*(x/CALL_FREQUENCY)*DA_MAX_OUT;
    x += step*frequency;
    if(curdata>DA_MAX_OUT)curdata = DA_MAX_OUT;
    return curdata;
}
```

2.2 DA 数模转换模块

DA 模数转换模块功能是将波形产生函数模块产生的波形数字值转换为模拟电压量，通过不断将波形产生模块产生的值传入 DA 转换器，便可以产生连续的指定波形。

实际实现是利用 LPC2378 片上的 DA 资源。通过这只 PINSEL1 寄存器选择 P0.26 引脚为 AOUT 模拟输出功能，当 DA 外设寄存器 DACR 写入新值后，经过所选的设定时间，管脚 P0.26 的电压为 VALUE/1024×VREF(相对于 VSSA,一般为 3.3V)。DA 初始化代码如程序 2-5 所示。

程序 2-5 DA 初始化函数

```
//DA 初始化
void DA_Init(void)
{
    //P0.26 功能引脚设定为 DA
    PINSEL1 = (PINSEL1&0xFFCFFFFF)|0x00200000;
}
```

因此 DA 数模转换模块就是讲波形产生函数产生的数字量转换为模拟量并输出到具体的引脚上。

2.3 定时中断模块

定时中断的功能是产生频率比较高的周期性的定时中断，用于定时地将波形产生函数模块产生的波形数据值送入 DA 模数转换模块。同时也用于产生用于按键定时扫描和系统定时显示的时间基准标志，这个时间基准为 10ms，而中断的周期为 us 级别，因此需要在中断服务函数内设置一个计数值，当计数值大于一定次数后设置 10ms 时间标志。

定时中断的频率会直接影响波形产生函数的调用频率，即会影响波形的频率，为了波形的频率与设置的值不至于有大的偏离，需要定时器有一个比较稳定的且与预期频率一致的中断频率。中断频率需要结合 LPC2378 的中断响应能力确定。

通过在 LPC2378 定时器的实际测试，发现 LPC2378 在定时器中断周期在 50us 以下时，中断的响应会变得比实际周期慢，导致实际的中断频率比预期的地。但定时中断的频率准确是后续产生波形频率正确的前提，结合实验要求最终设定定时中断周期为 100us，即 10Khz。定时器初始化代码如程序 2-6 所示，定时器中断服务函数如程序 2-7 所示。

程序 2-6 定时器 0 初始化函数

```
//TIMER0 与 VIC 初始化
void Timer_Init(uint32_t time_us)
{
    T0TC   = 0;          //定时器 0 的计数器清零
    T0PR   = 0;          //时钟不预分频
    T0MCR  = 0x03;       //设置 T0MR0 匹配后复位 T0TC，并产生中断标志
    T0MR0  = (PCLK_TIMER0/1000000)*time_us ;  //TIM0 的 PCLK
    T0TCR  = 0x01;       //启动定时器
    /* 设置 IRQ TIM0 中断对应的是 VIC 中断通道的中断向量 4*/
    VICIntSelect = VICIntSelect & (~(1<<4));
    VICVectPriority4 = 0x00;
    VICVectAddr4 = (unsigned int)IRQ_Timer0;
    VICIntEnable |= 1 << 0x04;
}
```

程序 2-7 定时器 0 中断服务函数

```
//TIMER0 中断服务函数
void __irq IRQ_Timer0 (void)
{
    static uint8_t  count=0;
    static uint32_t da_out;
    count++;
    if(count>100)
    {
        time_10ms_flag = 1;
        count = 0;
    }
    if(wave_type==0)da_out = wave_square( frequency,
amplitude, offset,duty);
    else if(wave_type==1)da_out = wave_sin( frequency,
amplitude, offset);
    else if(wave_type==2)da_out = wave_triangle( frequency,
amplitude, offset);
    else if(wave_type==3)da_out = wave_sawtooth( frequency,
amplitude, offset);
    DACR = da_out<<6;
    system_time += 0.0001;
    T0IR = 0x01;                //清除中断标志
    VICVectAddr = 0x00;         //通知 VIC 中断处理结束，更新 VIC
}
```

2.4 按键扫描模块

按键扫描模块是人机接口的一部分，作为人机交互的输入。通过周期性地扫描按键的状态获取用户的输入，调整系统的输出。比如改变波形类型、频率、幅值等。

本次实验共使用了 5 个按键，对应的引脚分别为 P1.24、P1.25、P1.26、P1.27、P2.10。其原理图如图 2-5 所示。由原理图可知，按键在未被按下时对应引脚上为高电平，按下后为低电平。通过扫描这几个引脚的状态便可以获取到按键的状态，做相应的系统参数调整。

另外按键扫描的实现为定时扫描，连续若干次引脚都为低电平才认为按键被按下，避免了通过延时的方式进行按键的消抖。

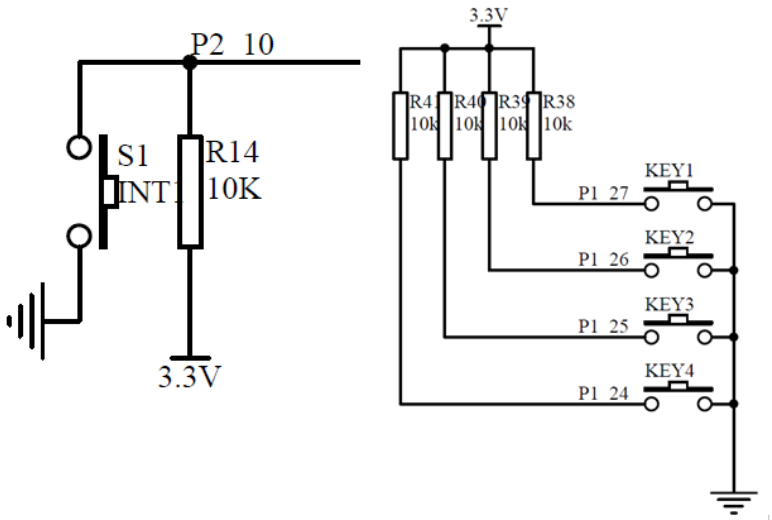

图 2-5 按键原理图

相关按键在程序内定义如说明 2-3 所示。

说明 2-3 按键相关定义

```
#define Key_1     ((~IOPIN1>>27)&0x01)
#define Key_2     ((~IOPIN1>>26)&0x01)
#define Key_3     ((~IOPIN1>>25)&0x01)
#define Key_4     ((~IOPIN1>>24)&0x01)
#define Key_5     ((~FIO2PIN>>10)&0x01)
```

按键的扫描首先需要将按键相应的引脚设置为输入，之后再周期性地检查相应引脚的数据状态。对应的按键初始化函数如程序 2-8 所示，按键扫描函数如程序 2-9 所示。

程序 2-8 按键初始化函数

```
//按键 IO 初始化
void Key_Init(void)
{
    PINSEL10 = 0;              //禁止 ETM
    PINSEL3 &= 0xFF00FFFF;     //选择 P1.24~P1.27 为 GPIO
    IODIR1 &= 0xF0FFFFFF;      //P1.24~P1.27 输入
    PINSEL4 &= 0xFFCFFFFF;     //选择 P2.10 为 GPIO
    FIO2DIR &= 0xFFFFFBFF;     //P2.10 输入
    FIO2MASK &= 0xFFFFFBFF;
    Key_Scan();                //初始化 KeyUp_old,KeyDown_old
}
```

程序 2-9 按键扫描函数

```
//按键 IO 扫描
void Key_Scan(void)
{
    static uint8_t keysta_old=0;
    uint8_t keysta_new=0x00;
    keysta_new|=Key_5;
    keysta_new=keysta_new<<1;
    keysta_new|=Key_4;
    keysta_new=keysta_new<<1;
    keysta_new|=Key_3;
    keysta_new=keysta_new<<1;
    keysta_new|=Key_2;
    keysta_new=keysta_new<<1;
    keysta_new|=Key_1;
    KEY_Scan_STA=keysta_new&keysta_old;
    KEY_Scan_STA|=KEY_Scan_STA_Update_MASK;      //添加标识位
    keysta_old=keysta_new;
}
```

最终按键扫描实现为 10ms 周期性扫描，连续若干次扫描到按键按下才设定按键按下有效。长按一段时间（300ms）后，参数会每隔 50ms 变化一次，即连按效果。每次只有一个按键有效，若多个按键按下，最后一个按下的按键有效。

每个按键说明如下：

KEY1：切换输出波形类型（方波、正弦波、三角波、锯齿波）

KEY2：切换当前调整的参数（频率、幅值、偏移、占空比）

KEY3：以设置的步进值减小当前参数

KEY4：以设置的步进值增加当前参数

KEY5（INT）：改变 KEY3、KEY4 调整的参数（波形参数或是步进值）

2.5 OLED 显示模块

OLED 显示模块是人机接口的一部分，作为系统状态的显示输出。将系统的状态与参数显示在一个显示屏上，结合按键使系统的人机交互过程变的十分方便和谐。

本次实验选用的 OLED 为如图 2-6 所示的 OLED12864。该 OLED 由 SPI 驱动，实现上通过软件模拟 SPI 实现。

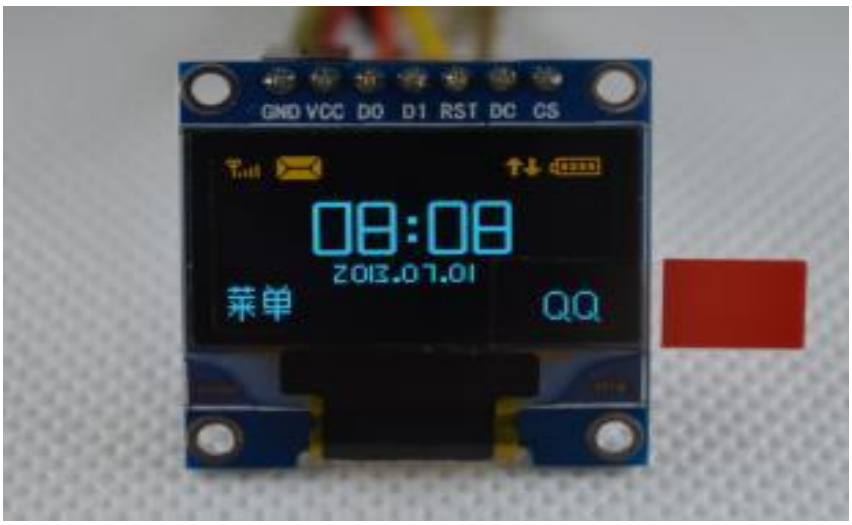

图 2-6 OLED12864 实物图与引脚定义

OLED 与 LPC2378 的连接见说明 2-4。相关的显示接口见所附源码。

说明 2-4 OLED 与 LPC2378 引脚连接

```
//OLED_CLK_D0          P2.7    OLED_MISO_D1       P2.6
//OLED_RST             P2.5    OLED_MOSI_DS       P2.4
//OLED_CS              P2.3
#define OLED_CLK_D0(x)         if(x)FIO2SET=1<<7;\
                               else FIO2CLR=1<<7;
#define OLED_MISO_D1(x)        if(x)FIO2SET=1<<6;\
                               else FIO2CLR=1<<6;
#define OLED_RST(x)            if(x)FIO2SET=1<<5;\
                               else FIO2CLR=1<<5;
#define OLED_MOSI_DS(x)        if(x)FIO2SET=1<<4;\
                               else FIO2CLR=1<<4;
#define OLED_CS(x)             if(x)FIO2SET=1<<3;\
                               else FIO2CLR=1<<3;
```

系统启动后将显示如图 2-7 的启动画面。

图 2-7 启动画面

系统启动后，将在显示屏顶部显示如图 2-8 所示的波形选择栏。分别代表方波、正弦波、三角波、锯齿波。

图 2-8 波形选择栏

除了以上图片，OLED 还会显示系统的一些参数，如波形的参数：频率、幅值、偏移、占空比。结合按键构成了本系统的人机交互接口。

所有的图片与文字的显示都是通过实现将对应图像或文字的图形码存在代码内实现的。

2.6 系统时钟模块

系统时钟模块为系统的基础且重要的模块，作为系统的时钟输入。由外部晶振和控制器内部的锁相环路 PLL 等组成。开发板上的外部晶振为 12Mhz。

本次实验设置 Fcclk 为 72Mhz，Fpclk_timer0 为 72Mhz，其他相关外设频率为 18Mhz。

2.7 电源管理模块

电源管理为开发板的一部分，将 5V 的电源供电稳压出 3.3V 的电源给微控制器与相关外设供电。

DA 数模转换的参考电压为 3.3V，因此 DA 输出的模拟量最大为 3.3V。

2.8 相关寄存器描述

2.8.1 GPIO 相关寄存器：

PINSELn：管脚功能选择寄存器，控制管脚的功能。

IOnDIR：GPIO 端口方向控制寄存器。

IOnMASK：端口的高速屏蔽寄存器。写、置位、清零和读端口改变或返回该寄存器中仅由 0 使能的位。

IOnSET：端口置位寄存器。该寄存器控制输出管脚的状态。写 1 使相应的端口管脚产生高电平。写 0 没有影响。

IOnCLR：端口输出清零寄存器。该寄存器控制输出管脚的状态。写 1 到相应端口的管脚产生低电平。写 0 没有影响。

IOnPIN:端口管脚值寄存器。数字端口管脚的当前状态可从该寄存器中读出。

2.8.2 DA 数模转换相关寄存器

DACR：DA 控制寄存器，该读/写寄存器包含转换成模拟值的数字值和用来平衡性能和功耗的位。

2.8.3 定时器相关寄存器

TOTC：定时器 0 计数器。32 位的 TC 每隔(PR+1)个 PCLK 周期递增一次。

TOPC：预分频计数器。32 位的 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当达到了 PR 的值时，TC 增加，PC 被清除。

TOMCR:匹配控制寄存器。MCR 用来控制在匹配出现时是否产生中断和 TC 是否复位。

TOMR0:匹配寄存器 0。MR0 可通过 MCR 设定为在匹配时复位 TC,停止 TC 和 PC 和/或产生中断。

TOTCR：定时器控制寄存器。TCR 用来控制定时器计数器功能。定时器计数器可以通过 TCR 来禁能或复位。

TOIR：中断寄存器。可向 IR 写入相应的值来清除中断。

2.8.4 向量中断控制器 VIC 相关寄存器

VICIntSelect：中断选择寄存器。该寄存器将 32 个中断请求的每个都分配为 FIQ 或 IRQ。

VICVectPriorityn：向量优先级 n 寄存器。指定了相应的向量 IRQ 的优先级。

VICVectAddrn：向量地址 n 寄存器。保存了相应的向量 IRQ 的中断服务程序(ISR)地址。

VICIntEnable：中断使能寄存器。该寄存器控制将 32 个中断请求和软件中断中的哪几个使能为 FIQ 或 IRQ。

VICVectAddr：向量地址寄存器。当发生 IRQ 中断时，向量地址寄存器保存当前有效中断的地址。

2.9 源代码

实验源代码另附，集成开发环境为 MDK Keil 5.14。

三、实验结果分析

3.1 实验结果

3.1.1 波形发生结果

方波波形如图 3-1 所示。其中设定的波形参数为频率 100hz，幅值 3.3/2V，偏移 0V，占空比 50%。

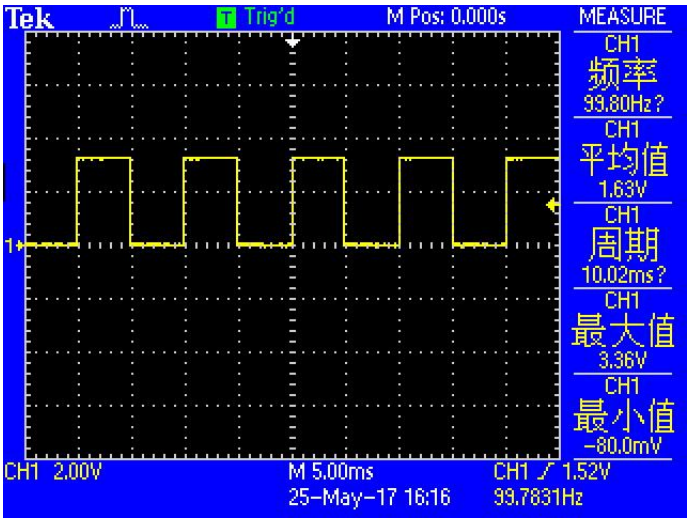

图 3-1 方波波形

正弦波波形如图 3-2 所示。其中设定的波形参数为频率 100hz，幅值 3.3/2V，偏移 0V。

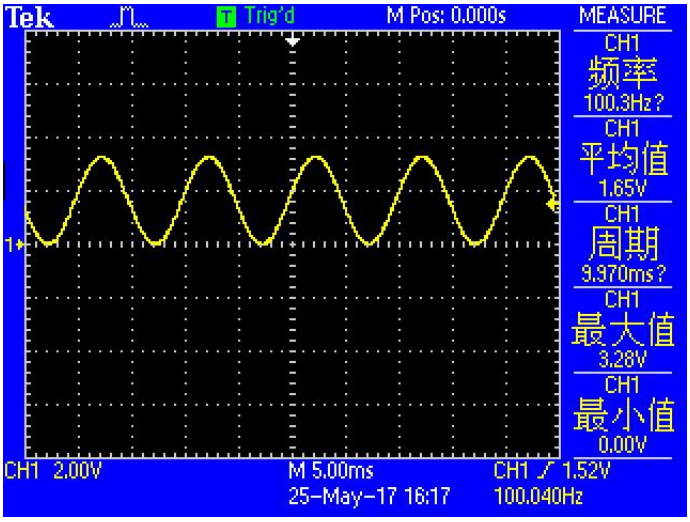

图 3-2 正弦波波形

三角波波形如图 3-3 所示。其中设定的波形参数为频率 100hz，幅值 3.3/2V，偏移 0V。

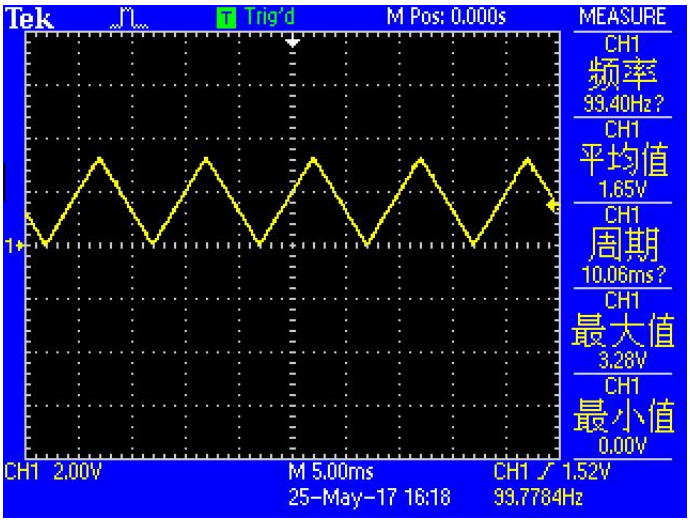

图 3-3 三角波波形

锯齿波波形如图 3-4 所示。其中设定的波形参数为频率 100hz，幅值 3.3/2V，偏移 0V。

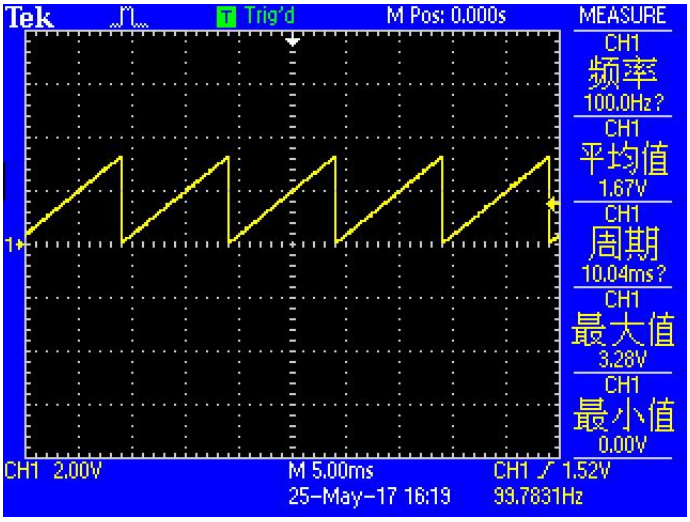

图 3-4 锯齿波波形

以上是产生的四种波形，分别是方波、正弦波、三角波、锯齿波。

实验题目还要求需要实现通过按键修改波形的频率，下面以方波为例演示通过按键修改过频率后的波形。

修改过频率的方波波形如图 3-5 所示。其中设定的波形参数为频率 200hz，幅值 3.3/2V，偏移 0V，占空比 50%。

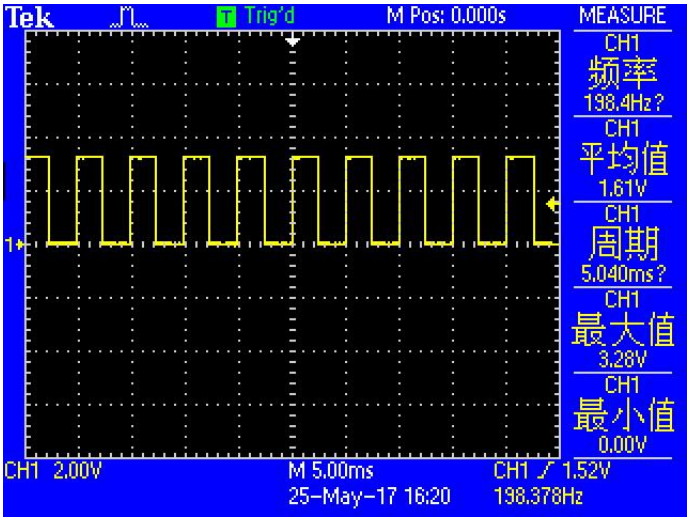

图 3-5 修改频率后的方波波形

在题目要求之外，还实现了对波形的幅值、偏移、占空比进行调节。以方波为例演示对这些参数的调节效果。

经过多项参数调节的矩形波波形如图 3-6。其中设定的波形参数为频率 100hz，幅值 2.3/2V，偏移 1V，占空比 70%。

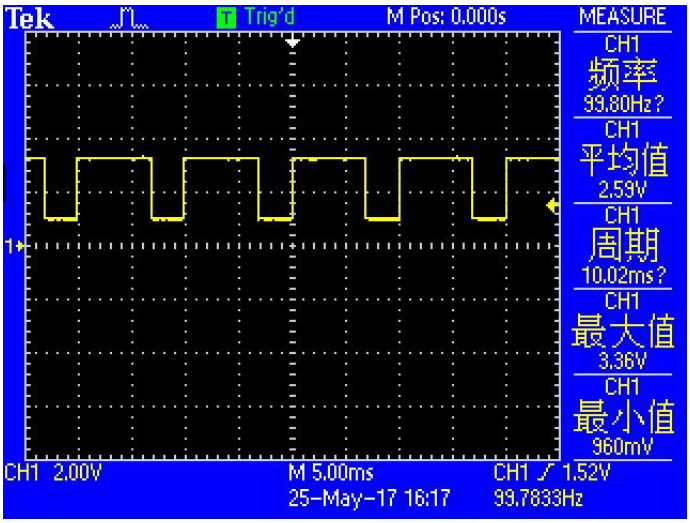

图 3-6 修改多项参数的矩形波波形

3.1.2 OLED 显示结果

系统启动页面实际显示效果如图 3-7 所示。

图 3-7 系统启动页面

系统启动后实际显示效果如图 3-8 所示。

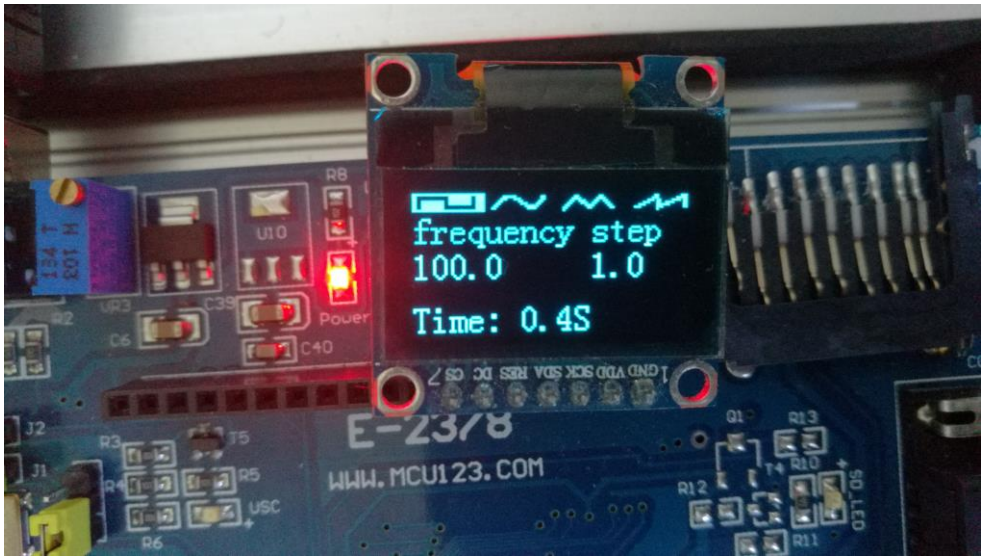

图 3-8 系统运行时页面

3.2 结果分析

经过测试，系统的各个组成模块按预期实现了相应的功能。通过 LPC2378 开发板产生了方波、正弦波、三角波、锯齿波。并且可以通过按键进行波形的参数调节，如波形的频率、幅值、偏移、占空比。

定时器中断产生的时间基准比较精准，因此在此时间基准基础上产生的波形的频率基本和预期设置的一致。体现了此波形发生器的可用性、准确性。

由按键与 OLED 组成的人机交互接口工作正常，使得波形的选择与参数设置变得更加人性化，使系统变得更加易用。

四、总结

本次开放实验，实现了使用 LPC2378 开发板上硬件资源，编程实现波形发生器功能，通过按键 1 可切换波形的形态（正弦波、三角波、方波），通过按键 2 可改变波形的频率。

在以上题目要求基础上，还增加了波形类型，加入了锯齿波。同时增加了波形可调节的参数，除了题目要求的改变频率以外，增加了对波形幅度、偏移、占空比的调整。适应更多样化的信号波形需求。

另外本系统增加了 OLED 显示的功能，在实现波形发生器的功能基础上，使人机交互变得更加方便和谐。OLED 显示了当前输出的波形类型，以及当前输出波形的参数，结合按键组成了本系统的人机交互接口，使得本系统更加易用。更加符合实际场景中的应用需求。

通过本次嵌入式开放实验，我得以将课堂上学到的知识结合实践加以运用，加深了我对嵌入式系统尤其是以 LPC2378 为代表的的 ARM 开发板的理解。在这个过程中，我熟悉了嵌入式开发环境，掌握了嵌入式开发的流程与方法。收获了难得的实践经验。

感谢自动化实验中心提供良好的实验环境与实验工具，感谢老师在各次实验中的指导与帮助。