基于 NiosII 的等精度数字频率计的设计

（Author:itspy）

摘　要：NiosII 处理器是一个具有很大灵活性的 32 软核处理器，比较容易在片上实现 SOPC。本文主要提供了一种基于 NiosII 软核处理器的等精度数字频率计的设计的实现方法。

关键词： FPGA、NiosII、等精度、频率计

Abstract NiosII embedded procesor is a general RISC one with configuration by user,which is flexible and powerful proccessor.With the improvement of property in PLD,it is very easy to design and realize the SOPC,and the user can not only design flexibly,but also do field system modification.Basing on NiosII embedded processor,this paper provides a sample of designing Equal-Precision Cymometer.

Keywords:FPGA、NiosII、Equal-Precision、Cymometer

引言：

Nios II 嵌入式处理器是 Altera 公司于 2004 年 6 月推出的第 2 代用于可编程逻辑器件的可配置的软核处理器，性能超过 200 DMIPS。基于哈佛结构的 RISC 通用嵌入式处理器软核。NiosII 处理器系统的外设配置具有很大的灵活性，设计者可以根据十几系统需求来添加必要的外设，Nios II 能与用户逻辑相结合，编程至 Altera 的 FPGA 中。它特别为可编程逻辑进行了优化设计，也为可编程单芯片系统(SoPC)设计了一套综合解决方案。NiosII CPU 具有以下特点：具备完整的 32 位操作指令集，32 位数据通道核地址空间，带有 32 个通用寄存器；支持 32 个外部中断源；单指令的 32 位于 32 位乘法核除法指令；带有单指令桶形移位寄存器；可以访问各种内外设，提供与片外存储器核外设的接口；

Nios II 处理器系列包括 3 种内核：一种是高性能的内核(Nios II/f)；一种是低成本内核(Nios II/e)；一种是性能/成本折中的标准内核(Nios II/s)。

Nios II 处理器支持 256 个具有固定或可变时钟周期操作的定制指令；允许 Nios II 设计人员利用扩展 CPU 指令集，通过提升那些对时间敏感的应用软件的运行速度，来提高系统性能。

方案论证与选择：

方案一：

可以利用单片机来实现是比较容易的，但是现今的单片机，处理频率一般不是很高，资源可利用少，而且相对来说单片机易受外部条件的干扰，功耗也高，所以在精度与其他性能方面都难以实现高性能标准。

方案二：

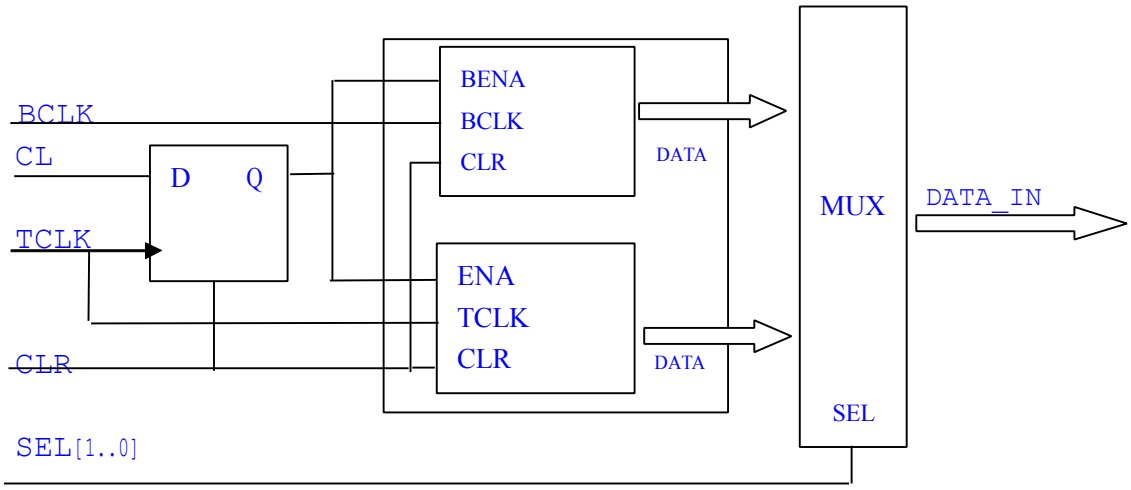
可以利用 ARM 实现。可以说 ARM 是性能较好的，但相对 FPGA 来说其处理速度还是稍逊一筹的，而且由于 FPGA 可以让用户通过软件编程的方式添加或者更改软件，从而在硬件不用变化的情况下，实现其他更多的功能，在灵活性上远优于 ARM 处理器。

方案三：

利用 FPGA 实现。所谓 FPGA 是 Field Programmable Gate Array 的简称。FPGA 的出现为现代电子产品嵌入式的设计带来了更大的灵活性，更容易实现片上系统（SOC）。通过在 FPGA 上潜入 NIOSII 处理器，在 IDE 环境下采用 C，或 C++对各种逻辑器件进行控制。 另外在 FPGA 上采用 VHDL 或者 VERILOG 等硬件描述语言编写各种逻辑器件的驱动，从而实现各种功能。

实现原理：

下图为在 FPGA 中实现的一个频率测量模块。



其原理为：

设计两个输出数据都为 32 位的 BZH、TF 计数器分别用来测量标准频率和待测频率计数。待测频率作为一个 D 触发器的时钟信号，CL 是作为 D 触发器的预置门控信号，D 触发器的输出信号为 ENA 是同时对 BZH、TF 计数器的使能。

当 CL=1 时，两计数器同时开始计数。当 CL=0 时，都停止工作。这样得到一下测量频率的公式：

Fx=(Fs/Ns)*Nx

Fx:为所测 TCLK 的频率　　Fs:为标准频率

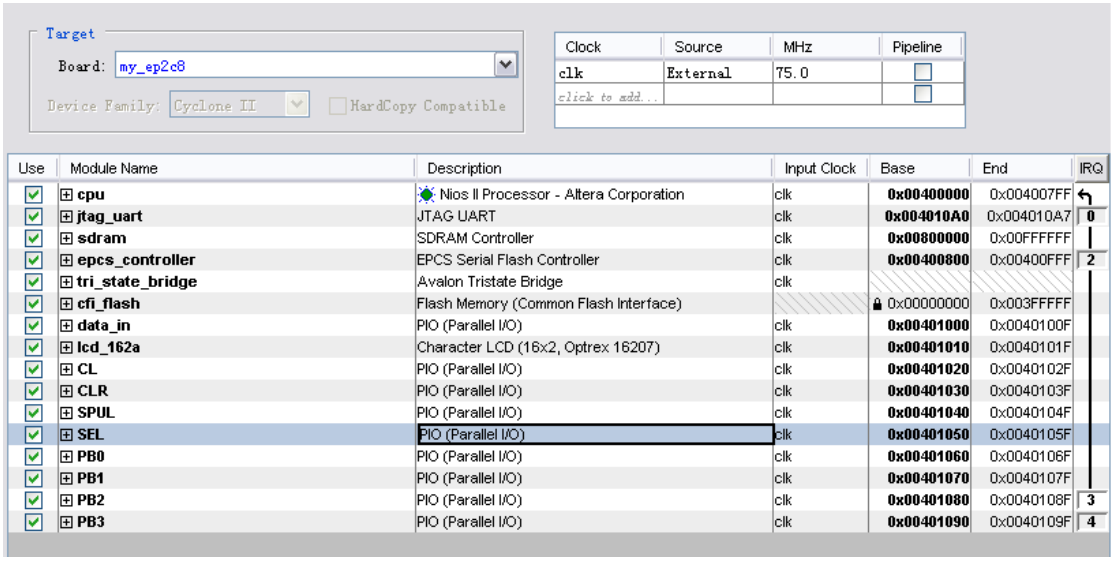
Ns:BZH 中的计数值　　Nx:TF 中的计数值

由于在 CL 门控信号控制下，ENA 使能信号都是所测 TCLK 信号的整数倍。而且 CL 的宽度的变化以及随即的出现时间所造成的误差最多只有 BCLK 信号的一个时钟周期。也就是说，待测结果不受 CL 门控信号的影响，最终的理论误差只有标准信号 BCLK 信号的一个周期。这就是等精度频率测量的原理.

对于频率占空比的测量于频率测量类似.即通过构造逻辑控制使 BZH、TF 两个计数器分别测量高低电平的计数数据,如 N1 表示高电平的计数,N2 表示低电平的计数,这样所求占空比为：

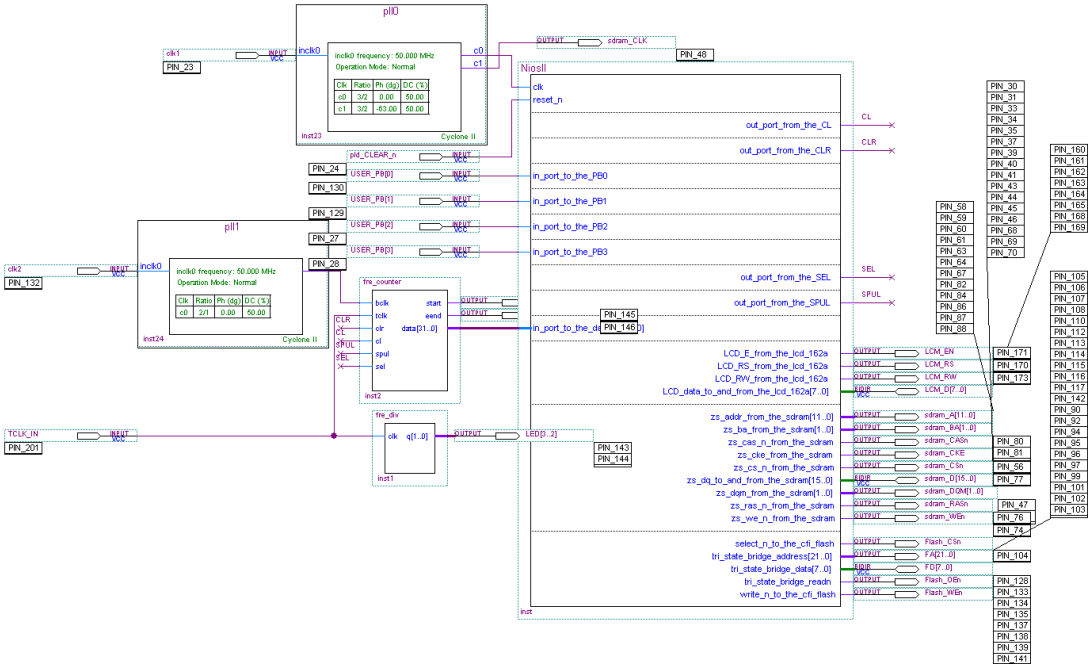
Rate=N1/(N1+N2)*100%

本系统在 SOPC BUILDER 中加入一下 IP 核有：
NIOSII 处理器(fast)、JTAG、EPCS_CONTROLLER、
LCD16207、FLASH、TRI_STATE_BRIDGE 以及普通 PIO 口
各 IP 核的作用分别如下：
NiosII 处理器用来对系统各外设的逻辑控制。
JTAG 用来下载和调试程序。
SDRAM 用做程序运行时所要的内存。
EPCS_CONTROLLER 用来存放 FPGA 的配置数据也即引导数据。
FLASH 用来存放最终的软件的内容（程序的数据）
LCD 用来显示所测量的结果 ， IO 口用来做键盘控制
SOPC 配置情况如下图所示：



在 QuartusII 中加入两个 PLL，一个 PLL 提供 NIOSII CPU 和 SDRAM 的工作频率，为 75MHZ，另一个作为标准频率 BCLK 计数频率，为 100MHZ。这样再通过 NIOSII 在 IDE 环境下对频率测量模块控制，且对其结果进行处理后送到 LCD 显示，最终实现整个系统要求。

QuartusII 下系统的最终框图如下所示：



程序的编写：

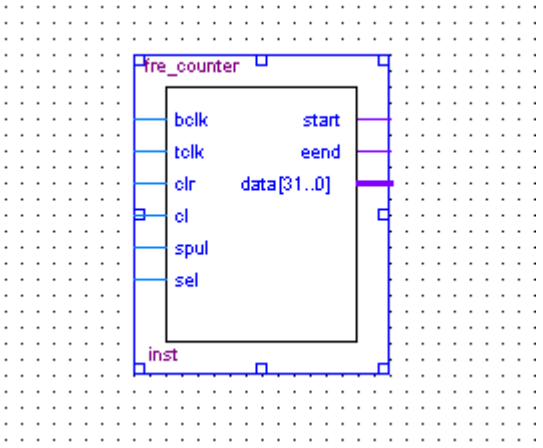
由于 IDE 支持标准的 C 编译环境，这样可以通过 C 编程直接对由 SOPC 生成的设备进行访问和控制。最后将分别将生成软件数据保存到 FLASH 中去，将 FPGA 配置文件存到 EPCS 器件中去，从而当系统在上电的时候，自动加载程序，从而启动系统运行。

结束语：

本文基于 NIOSII 的等精度数字频率计的设计，采用 VHDL 编写底层模块，C 语言来编上层应用程序。然后将程序下到硬件中去，最后用示波器提供频率信号来检验系统的功能。检测结果达到预期的高精度要求，且系统运行稳定，很好的实现了系统所要求的功能。

参考文献：

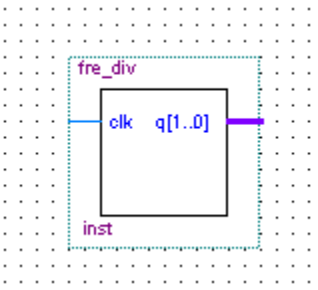
[1]NiosII software Developer's Handbook
http://www.altera.com
[2]潘松 黄继业 编著 EDA 技术实用教程（第二版）科学出版社




```
--频率测量和脉宽测量模块
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity fre_counter IS
  port(bclk: in std_logic;
       tclk: in std_logic;
       clr : in std_logic;  --当 SPUL 为高电平时，CL 为预知门控信号，用于测频计数
       cl  : in std_logic;  --的时间控制，当 SPUL 为低时，CL 为测脉宽控制信号，
       spul: in std_logic;  --CL 高电平时测高电平脉冲
       start:out std_logic; --开始测量信号
       eend: out std_logic; --结束测量信号
       sel: in std_logic;   --数据输出选择信号
       data:out std_logic_vector(31 downto 0));
end fre_counter;
architecture bhv of fre_counter is
signal bzq: std_logic_vector(31 downto 0);
signal tsq: std_logic_vector(31 downto 0);
signal ena: std_logic;
signal ma,clk1,clk2,clk3: std_logic;
signal q1,q2,q3,bena,pul: std_logic;
signal ss:std_logic_vector(1 downto 0);
begin
  start<=ena;
  data<=bzq when sel='1' else
        tsq;
bzh: process(bclk,clr)          --标准标准计数器
     begin
       if clr='1' then bzq<=(others=>'0');
       elsif rising_edge(bclk) then
         if bena='1' then bzq<=bzq+1; end if;
       end if;
     end process;
tf:  process(tclk,clr,ena)      --待测频率计数器
     begin
       if clr='1' then tsq<=(others=>'0');
       elsif rising_edge(tclk) then
          if ena='1' then tsq<=tsq+1; end if;
```

```
      end if;
    end process;
process(tclk,clr)
begin
  if clr='1' then ena<='0';
  elsif rising_edge(tclk) then ena<=cl; end if;
end process;
ma<=(tclk and cl) or not (tclk or cl);     --脉宽测量逻辑
clk1<=not ma; clk2<=ma and q1; clk3<=not clk2; ss<=q2&q3;
dd1: process(clk1,clr)
    begin
      if clr='1' then q1<='0';
      elsif rising_edge(clk1) then q1<='1'; end if;
end  process;
dd2:process(clk2,clr)
    begin
      if clr='1' then q2<='0';
      elsif rising_edge(clk2) then q2<='1'; end if;
end  process;
dd3:process(clk3,clr)
    begin
      if clr='1' then q3<='0';
      elsif rising_edge(clk3) then q3<='1'; end if;
end  process;
pul<='1' when ss="10" else '0';  --pul ='1' 允许计数,'0' 禁止计数
eend<='1' when ss="11" else '0';
bena<=ena when spul='1' else    --spul='1'为频率测量,'0'为脉宽测量
     pul when spul='0' else
     pul;
end bhv;
==========================================================================
```

VHDL 源程序：




```
--  待测信号 TCLK 的分频
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity fre_div is
  port( clk: in std_logic;
        q: out std_logic_vector( 1 downto 0));
end fre_div;
architecture bhv of fre_div is
signal q1: std_logic_vector(19 downto 0);
begin
  process(clk)
  begin
    if rising_edge(clk) then
       q1<=q1+1;
    end if;
  end process;
  q(0)<=q1(9);    --1024 分频
  q(1)<=q1(19);   --1024*1024 分频
end bhv;
```

==========

C 程序：

```
//*************************************************************//
// Title :  基于NiosII的等精度数字频率计的设计
// Author:  itspy
// Discription:
// 基于FPGA/Niosii 的等精度频率测量/
// 标准频率：100MHZ NiosII频率：75MHZ/
// 精度为：理论值为1e8/
// FPGA底层模块由2个PLL、1个频率测量模块、1个分频计模块/
// NiosII构成.1 LCD16207、4 Key、4 LED作为外围/
// Date : 04-DEC-07/
//*************************************************************//

//C 程序代码(模块文件由VHDL编写)：

#include "stdio.h"
#include "system.h"
#include "alt_types.h"
#include "sys/alt_irq.h"
#include "altera_avalon_pio_regs.h"


static float Fs=100000000.0; //标准频率fs
static alt_u32 Ns=0;      //标准频率计数器BZQ中计数值
static alt_u32 Nx=0;      //待测频率计数器TSQ中计数值
static alt_u8 Test_Mode=1; //测量模式
static alt_u8 Operation=1; //频率测量或者占空测量运行或暂停标志

void Init();
void Test_Freq();
void Freq_Test();
void Duty_Test();
void Key_Scan();
void Delay(alt_u32 dly);
static void pb2_ISR(void *context,alt_u32 id);  //中断服务程序
static void pb3_ISR(void *context,alt_u32 id);  //


int main()
{
  Init();
  while(1){
    if(Operation==1){
    Key_Scan();
```

```
    Test_Mode?Freq_Test():Duty_Test();
    }
  }
}


void Init()  // 初始化设置
{
  FILE *lcd;
  IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PB2_BASE,0x1);
  IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PB3_BASE,0x1);//  PB2的优先级为3;
PB3的优先级为2;
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PB2_BASE,0x0);
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PB3_BASE,0x0);
  alt_irq_register(PB2_IRQ,0,pb2_ISR); //注册中断服务程序
  alt_irq_register(PB3_IRQ,0,pb3_ISR); //
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1); //清零BZQ,TSQ计数器
  IOWR_ALTERA_AVALON_PIO_DATA(CL_BASE,0x0);//初始化CL预置门,0x0 禁止
计数，0x1使能计数
  IOWR_ALTERA_AVALON_PIO_DATA(SPUL_BASE,0x1); //0x1为频率测,0x0为占
空比测量
  IOWR_ALTERA_AVALON_PIO_DATA(SEL_BASE,0x0); //选通读Ns值
  lcd=fopen("/dev/lcd_162a","w");
  if(lcd!=NULL){
    fprintf(lcd,"   Cymometer   \n");
    Delay(5000000);
    fprintf(lcd,"  Initialized  \n");
    Delay(5000000);
    fprintf(lcd," FrequencyTest \n");
    Delay(500000);
    fprintf(lcd,"\n");
    fclose(lcd);
  }
}

static void pb2_ISR(void *context,alt_u32 id)
{ //中断服务程序1，暂停当前测量
  IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PB2_BASE,0x1);
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PB2_BASE,0x0);
  Operation=0;
}

static void pb3_ISR(void *context,alt_u32 id)
{ //中断服务程序2，开始当前测量
```

```
  volatile int *edge_cap_ptr=(volatile int*)context;
  *edge_cap_ptr=IORD_ALTERA_AVALON_PIO_EDGE_CAP(PB3_BASE);
  IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PB3_BASE,0x1);
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PB3_BASE,0x0);
  Operation=1;
}

void Freq_Test() //频率测量
{
  FILE *lcd;
  IOWR_ALTERA_AVALON_PIO_DATA(SPUL_BASE,01);
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1);
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x0);
  IOWR_ALTERA_AVALON_PIO_DATA(CL_BASE,0x1); //开始
  Delay(5000000); //读数时间
  IOWR_ALTERA_AVALON_PIO_DATA(CL_BASE,0x0); //结束
  IOWR_ALTERA_AVALON_PIO_DATA(SEL_BASE,0x0); //读BZQ计数值
  Ns=IORD_ALTERA_AVALON_PIO_DATA(DATA_IN_BASE);
  IOWR_ALTERA_AVALON_PIO_DATA(SEL_BASE,0x1); //读TSQ计数值
  Nx=IORD_ALTERA_AVALON_PIO_DATA(DATA_IN_BASE);
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1); //读完清零
  lcd=fopen("/dev/lcd_162a","w");
  if(lcd!=NULL){
    if(Nx==0){
      fprintf(lcd,"    NoSignal    \n      !!!      \n");
    }
    else{
      fprintf(lcd,"FrequencyResult\n");
      fprintf(lcd,"%.2fHz\n",(1.0*Fs/Ns*Nx));
    }
    fclose(lcd);
  }
  return;
}


void Duty_Test() //占空比测量
{ FILE *lcd;
  IOWR_ALTERA_AVALON_PIO_DATA(SEL_BASE,0x0); //以下都是读BZQ计数器值
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1); //清零
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x0); //准备计数
  IOWR_ALTERA_AVALON_PIO_DATA(CL_BASE,0x1); //高电平开始计数
  Delay(2000000); //读数时延时间，TCLK低电平到来时，停止计数
  Ns=IORD_ALTERA_AVALON_PIO_DATA(DATA_IN_BASE); //读高电平计数值
```

```
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1); //清零
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x0); //准备计数
  IOWR_ALTERA_AVALON_PIO_DATA(CL_BASE,0x0); //低电平开始计数
  Delay(2000000); //读数时延时间，TCLK高点平到来时，停止计数
  Nx=IORD_ALTERA_AVALON_PIO_DATA(DATA_IN_BASE);
  IOWR_ALTERA_AVALON_PIO_DATA(CLR_BASE,0x1); //读完清零
  lcd=fopen("/dev/lcd_162a","w");
  if(lcd!=NULL){
    if(Nx==0){
      fprintf(lcd,"    NoSignal    \n      !!!      \n");
    }
    else{
      fprintf(lcd," DutyCircleRate \n");
      fprintf(lcd,"%.2f%c\n",(1.0*Ns/(Nx+Ns)),'%');
    }
    fclose(lcd);
  }
  return;
}

void Key_Scan() //扫描键盘子程序
{ FILE *lcd;
  if(!IORD_ALTERA_AVALON_PIO_DATA(PB0_BASE)){
    Delay(2000000);
    if(!IORD_ALTERA_AVALON_PIO_DATA(PB0_BASE)){
      IOWR_ALTERA_AVALON_PIO_DATA(SPUL_BASE,0x0);
      Test_Mode=0;
      lcd=fopen("/dev/lcd_162a","w");
      if(lcd!=NULL){
        fprintf(lcd," DutyCircleTest \n");
        fclose(lcd);
      }
      while(!IORD_ALTERA_AVALON_PIO_DATA(PB0_BASE));
//键PB0未放开，则在此处停止
    }
  }
  if(!IORD_ALTERA_AVALON_PIO_DATA(PB1_BASE)){
    Delay(2000000);
    if(!IORD_ALTERA_AVALON_PIO_DATA(PB1_BASE)){
      IOWR_ALTERA_AVALON_PIO_DATA(SPUL_BASE,0x1);
      Test_Mode=1;
      lcd=fopen("/dev/lcd_162a","w");
      if(lcd!=NULL){
        fprintf(lcd," FrequencyTest \n");
```

```
        fclose(lcd);
      }
      while(!IORD_ALTERA_AVALON_PIO_DATA(PB1_BASE));
//键PB0未放开，则在此处停止
    }
  }
  return;
}
void Delay(alt_u32 dly)
{
  while(--dly);
}
```