华南農業大學

DSP 技术与应用课程设计

自动化测量信号的周期

课程设计报告书

姓名：陈嘉维、罗晟宾、邱致远

班级：2020 级电子信息类工科 2、5 班

学号：202034310204　202034310214

201934910319

指导老师：高鹏

日期：2023.5.6 - 2023.5.7

华南农业大学电子工程学院（人工智能学院）

摘　　要

周期性是一种重要的自然规律。大自然中无处不在的循环现象，国际经济周而复始的波动变化，故障的机器设备所发出的振动信号，凡此种种，无不体现出周期律的强大作用。了解各种现象的周期性并准确测量其周期值，将有助于我们掌握规律、预测趋势、诊断错误，最终使我们趋利避害、长久获益。如果伴随着周期现象释放某种可捕获的信号，通过对信号的处理和分析，我们就有望获得关于信号周期的信息。

本次课程设计中对这一问题进行了简化。在本次课程设计中，我们要完成的任务是人工实时产生一个单频信号，并自动化的测量信号的周期，目的是训练使用 F28335 平台进行数字信号处理的工程实践能力。

关键词：周期性 自动化 信号的周期 F28335 数字信号处理

Automated measurement of signal cycles

Chen Jiawei Luo Shengbin Qiu Zhiyuan

College of Electronic Engineering, South China Agricultural University, Guangzhou, 510642, China

Abstract： Periodicity is an important natural law. The ubiquitous cyclic phenomena in nature, the cyclical changes in the international economy, and the vibration signals emitted by faulty machinery and equipment all reflect the powerful role of the periodic law. Understanding the periodicity of various phenomena and accurately measuring their periodic values will help us grasp patterns, predict trends, diagnose errors, and ultimately enable us to seek benefits and avoid harm for a long time.

If a captured signal is released along with the periodic phenomenon, through signal processing and analysis, we have the potential to obtain information about the signal period.

This issue has been simplified in the course design. In this course design, the task we need to complete is to manually generate a single frequency signal in real time and automatically measure the signal cycle, with the aim of training the engineering practical ability to use the F28335 platform for digital signal processing.

Key words: Periodicity Automatically signal cycle F28335 digital signal processing

目录

一、系统分析

1. 实验原理

（1）信号的产生

使用定时器（CPUTimer），定时周期为 100ms，每次中断生成正弦信号的一个数据点。即信号的采样间隔为 100ms（采样频率为 10Hz）。

（2）信号的控制

四个按键（SW4/SW5/SW2/SW3）分别控制四个 LED 灯（LED0/LED1/LED2/LED3）的亮灭。每次按下的动作使得相应的 LED 灯切换亮与灭的状态。当四个 LED 灯的状态为“亮亮灭灭”时，信号的周期为 12s（二进制数 1100 对应的十进制数是 12）；当四个 LED 灯的状态为“灭亮亮亮”时，信号的周期为 7s（二进制数 0111 对应的十进制数是 7）；等等。采用这种方式，信号的周期可以即时设置为 1~15 秒，最小单位为 1s。

（3）测量信号的周期

频谱分析法的基本原理：对累积一定时间长度的信号进行快速傅里叶变换（FFT），计算信号的幅度谱，搜索最强的谱线，确定其对应的频率。频率的倒数即为周期。

（4）测量代码的执行时间

在仿真器和 CCS 软件构筑的调试环境下，可以使用断点、单步运行和时钟功能，测量某段代码运行所消耗的时钟周期数，再根据器件的主频（时钟周期的倒数）即可折算成时间。但这种方法的测量结果不准确（往往偏大）。

另一种方法是，在所要测量的代码前后反转某个 GPIO 的输出，将 GPIO 的物理管脚引接到示波器，从而在示波器上测量代码的执行时间。这种方法的测量结果非常准确，但并不是很便利。

本次实验中，我们对这两种方法分别进行了实现，并且进行了对比。

2. F28335 芯片

（1）架构和性能

F28335 芯片采用 32 位 RISC 架构，主频可达 150MHz，具有高性能的浮点运算单元（FPU），能够快速高效地处理复杂的算法和运算。此外，该芯片还内置了 256KB 的 Flash 存储器和 18KB 的 RAM 存储器，可以满足大部分应用的存储需求。

（2）外设资源

F28335 芯片具有丰富的外设资源，包括 16 个 PWM 模块、12 路 ADC 模块、6 个定时器、4 个捕获/比较模块等，能够满足各种控制和计算应用的需求。其中，PWM 模块可用于驱动各种电机、灯光等设备，ADC 模块可实现高精度的模拟信号采集，定时器和捕获/比较模块可用于定时控制和信号处理。

（3）通信接口

F28335 芯片支持多种通信接口，包括 CAN、SPI、SCI 等，可以方便地与其他设备进行通信。其中，CAN 接口可用于汽车电子、工业控制等领域的通信，SPI 接口可用于外部 Flash 存储器等设备的通信，SCI 接口可用于串口通信等应用。

（4）保护机制和故障检测

F28335 芯片具有丰富的保护机制和故障检测功能，能够提高系统的可靠性和稳定性。其中，芯片内置了多个保护电路，如过流保护、过压保护、过温保护等，能够有效地保护芯片和系统。此外，芯片还具有多个故障检测功能，如失速检测、过零检测等，能够及时发现和处理系统故障。

二、程序设计

1. 软件调试

（1）任务 1：信号的产生和控制

main 函数中相应代码：

```
while(1)
    {
        T = get_key();
        get_UART();
        isSend(cosine);
    }
```

在 main 函数的 while（1）函数中，通过按键检测函数，检测使用者按下开发板的按键（SW4/SW5/SW2/SW3），串口检测函数检测是否向开发板发送命令‘t’（单个字符‘t’），当接收到发送的‘t’时，开发板会通过串口发送周期和相应时间的 cosine 值。

按键检测函数：

```
int get_key()
{
    key=KEY_Scan(0);
    switch(key)
    {
        case KEY2_PRESS:
            LED4_TOGGLE;
            D4=modify(D4);
            break;
        case KEY3_PRESS:
            LED5_TOGGLE;
            D5=modify(D5);
            break;
        case KEY4_PRESS:
            LED2_TOGGLE;
            D2=modify(D2);
            break;
        case KEY5_PRESS:
            LED3_TOGGLE;
            D3=modify(D3);
            break;
```

```
    }
    return calculate_C(D2,D3,D4,D5);
}
```

按键检测函数，检测在开发板中键入的按键，检测到四个按键其中有按键键入就会翻转 LED 灯的亮灭，并通过 modify 函数相应赋值给 D2D3D4D5，四位二进制数分别对应如下：D2 为二进制的第 4 位，D3 为二进制的第 3 位，D4 为二进制的第 2 位，D5 为二进制的第 1 位。并以返回他们的十进制值到 main 函数中。四个灯的亮灭可以反映信号的周期。

```
int modify(int byte)
{
    if(byte==1)
        {byte=0;}
    else
        {byte=1;}
    return byte;
}
//Modify 函数，检测到相应的按键键入就将相应位的值置 1。
int calculate_C(int D_2,int D_3,int D_4,int D_5)
{
    int Cir;
    Cir = D_5 + 2*D_4 + 2*2*D_3 + 2*2*2*D_2 ;
    return Cir;
}
calculate_C//函数将四位二进制数转化为十进制数返回。
#define D2 C[0]
#define D3 C[1]
#define D4 C[2]
#define D5 C[3]
//4 位的 C 数组分别宏定义为 D2D3D4D5。
//串口检测函数：
void get_UART()
{
    ReceivedChar = SciaRegs.SCIRXBUF.all;
}
//该子函数中，SciaRegs.SCIRXBUF.all 用于接收电脑发送的不定长度的字符串，然后存放于缓冲区中，定义的 ReceivedChar 用于接收所发送的字符串。
void isSend()
{
    if(ReceivedChar==0x74) //判断是否是't'字符
    {
```

```
32.         symbol=1;
33.     }
34.     else if(ReceivedChar==0x54) //判断是否是'T'字符
35.     {
36.         symbol=2;
37.     }
38.     else
39.     {
40.         symbol=0;
41.     }
42. }
```

判断接收的 ReceivedChar 字符串是“t”、“T”还是其他类型，将相应的 symbol 赋值为 1、2、0。如下图所示，在 ASCII 码表中，0x54 为字符“T”，0x74 为字符“t”。

二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(space)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u

图 1

定时器中断函数：

```
1. interrupt void TIM0_IRQn(void)
2. {
3.     EALLOW;
4.     LED1_TOGGLE;
5.     phi += 0.1;
6.     if(phi>2*M_PI)
7.         phi=0;
8.     omiga = 2 * M_PI / T;
```

```
    cosine = cos(omiga * phi);
    temp[flag]=cosine;
    flag++;
    if(flag>=512)
    {
        flag=0;
    }
    PieCtrlRegs.PIEACK.bit.ACK1=1;
    EDIS;
}
```

代码释义：通过定时器中断函数产生 cos 值信号。定时器每隔 100ms 进入执行中断函数，每执行一次中断函数 phi 的值就会增加 0.1，会产生一个周期为 2Π的 cos 函数，每次中断生成正弦信号的一个数据点，最后将 cos 函数数据点的值装进 temp[]的数值，temp 数组的值会根据时间不断迭代更新以反映最新周期的 cos 值。

（2）任务 2：信号的产生和控制

将任务 1 生成的装载 cosine 值的数组 temp，装入 FFT( )函数，FFT 函数源于 TI 官方的库文件，为官方的 FFT 函数，对官方函数进行改写运用，整段代码如下；原本的 cosine 值经过 FFT 算法函数处理后变为其相应的频谱幅值，重新装进新的数组 RFFT_f32，其对应的横坐标为 RFFT_f32_mag，然后再循环遍历找出 RFFT_f32 的最大值，用频谱最大值对应的横坐标（相应的 RFFT_f32_mag 值）来推测出原信号的频率 f 与周期 T，运算思路为 $f = i \times \frac{fs}{N}$。

```
void FFT(double origin_signal[])
{
    int i;
    for(i=0; i < RFFT_SIZE; i++)
    {
        RFFTin1Buff[i] = 0.0f;
    }//对传入数组进行清零

    for(i=0; i < RFFT_SIZE; i++)
    {
        RFFTin1Buff[i]=origin_signal[i];
    }//对传入数组进行赋值
    rfft.FFTSize   = RFFT_SIZE;
    rfft.FFTStages = RFFT_STAGES;
```

```
    rfft.InBuf      = &RFFTin1Buff[0];  //输入缓冲区
    rfft.OutBuf     = &RFFToutBuff[0];  //输出缓冲区
    rfft.CosSinBuf = &RFFTF32Coef[0];  //旋转因子缓冲区
    rfft.MagBuf     = &RFFTmagBuff[0];  //幅度缓冲区
    RFFT_f32_sincostable(&rfft);
    for (i=0; i < RFFT_SIZE; i++)
    {
        RFFToutBuff[i] = 0;
    }
    for (i=0; i < RFFT_SIZE/2; i++)
    {
        RFFTmagBuff[i] = 0;
    }

    RFFT_f32(&rfft);                    //计算 FFT 的幅值

    RFFT_f32_mag(&rfft);
    MAX=nmax(RFFTmagBuff,RFFT_SIZE/2);
    fre=MAX*10.0/RFFT_SIZE;
}
```

这段代码的前 4 个 for 循环都在为 RFFTin1Buff 数组赋值，其中的 RFFT_f32(&rfft);该语句调用了 TI 官方的库文件运算出 FFT 值 MAX=nmax(RFFTmagBuff,RFFT_SIZE/2)；该语句用以算出 FFT 的最大幅值的横坐标 i。fre=MAX*10.0/RFFT_SIZE;对应了 $f = i \times \frac{fs}{N}$ 这段公式，f 则为对应了计算机运算出预测原信号的频率。

```
int nmax(double arr[], int n)
{
    int i = 0, MAX = 0, l = 0;
    for (i = 1; i < n; i++)
    {
        if (arr[i] > MAX)
        {
            MAX = arr[i];
            l = i;
        }
    }
    return l;
}
```

该段代码功能为将所有 FFT 的幅值循环遍历比对找出其中的最大值。

2. 硬件调试

（1）任务 1：信号的产生和控制

在开发板中，D7_LED 的持续闪烁代表程序正在正常运行，如下图所示。

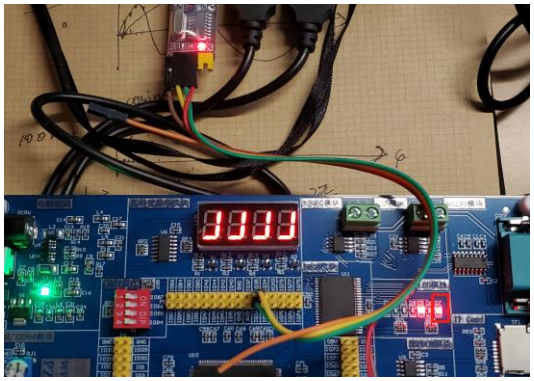


图 2

其左侧的四个灯 D2—D5 代表四位二进制数，D2 代表最高位，D5 代表最低位，如下图 3 所示。

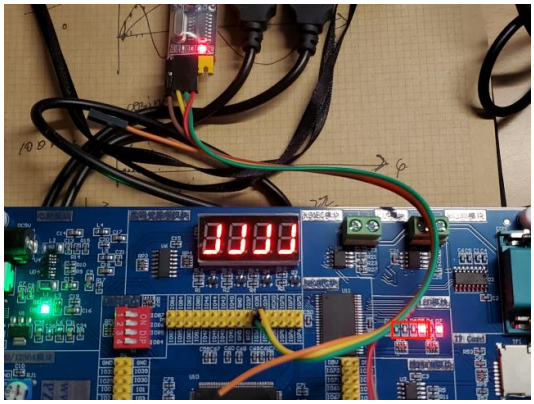


图 3

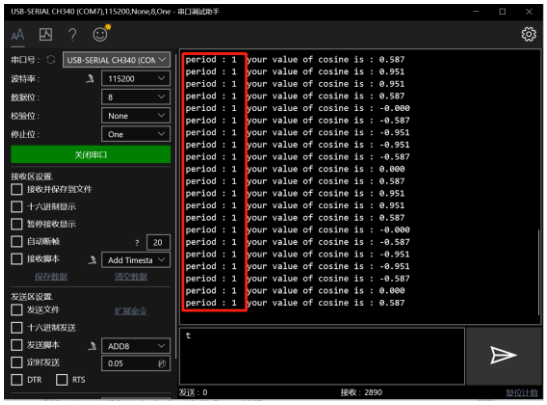


图 4

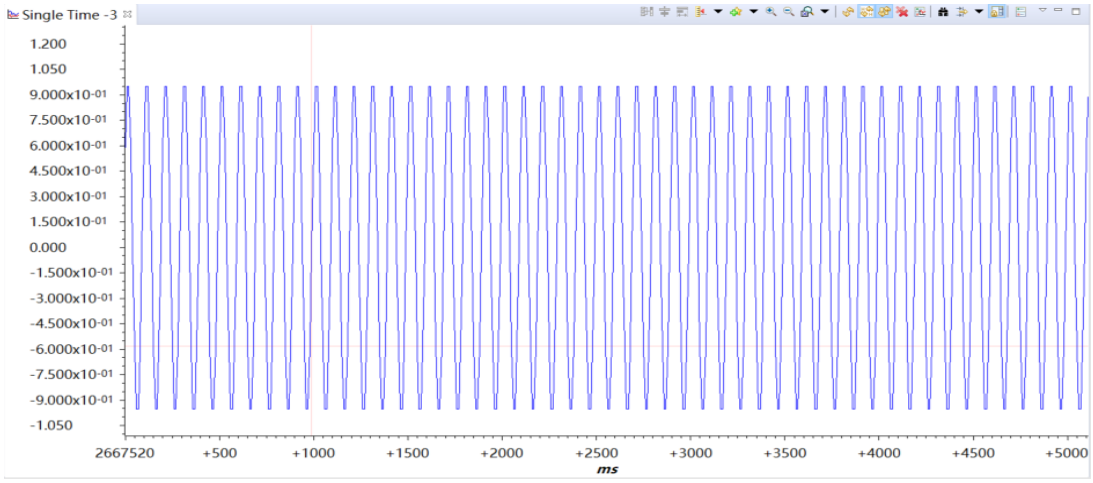
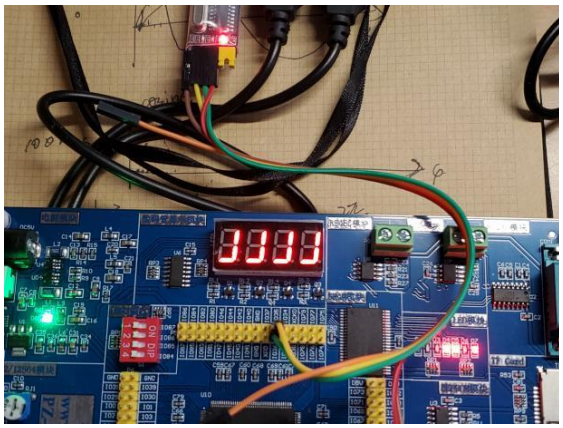


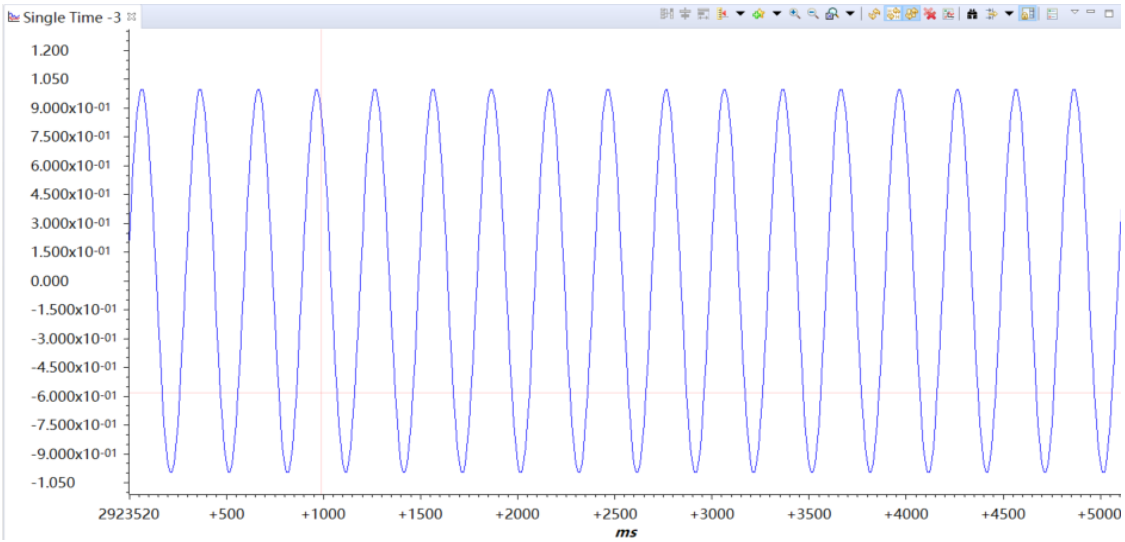
图 5

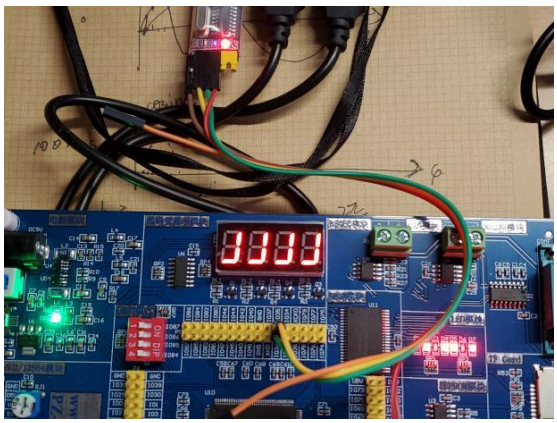
上图 3 中四个灯显示的二进制数代表了周期数 T=1，图 4 为接收到“t”后，上位机串口软件显示开发板的 cosine 周期与相应的值，图 5 为周期 1 时的 cosine 图像。如下为其他周期的开发板 LED 灯情况、上位机串口软件显示及相应周期的 cosine 图像。



```
period : 3  your value of cosine is : -0.000
period : 3  your value of cosine is : -0.207
period : 3  your value of cosine is : -0.406
period : 3  your value of cosine is : -0.587

t
```

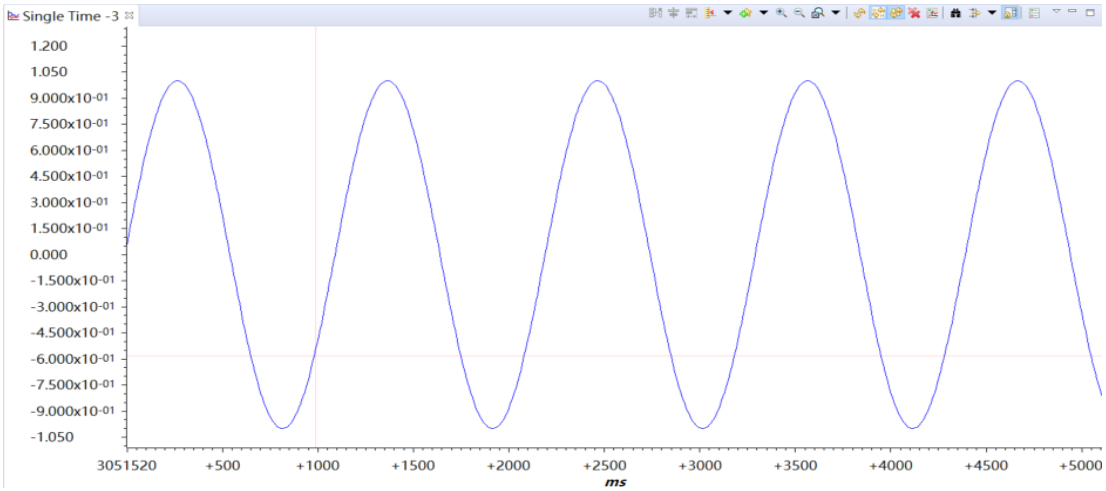
串口调试助手

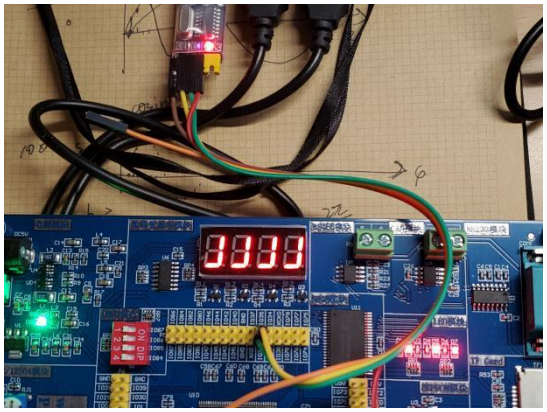




```
period : 11  your value of cosine is : -0.755
period : 11  your value of cosine is : -0.791
period : 11  your value of cosine is : -0.825
period : 11  your value of cosine is : -0.856
period : 11  your value of cosine is : -0.884

t
```

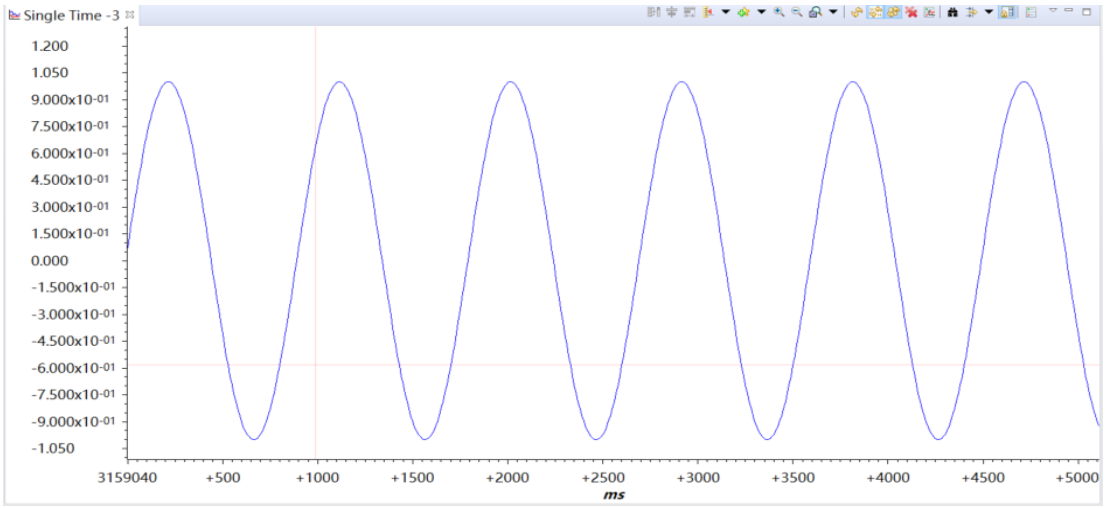
串口调试助手





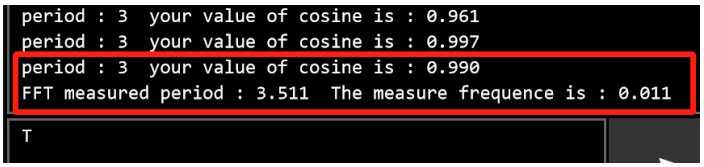
```
period : 9  your value of cosine is : -0.994
period : 9  your value of cosine is : -0.999
period : 9  your value of cosine is : -0.999
period : 9  your value of cosine is : -0.994

t
```

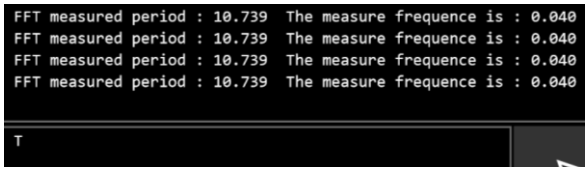
串口调试助手



（2）任务 2：测量信号的周期

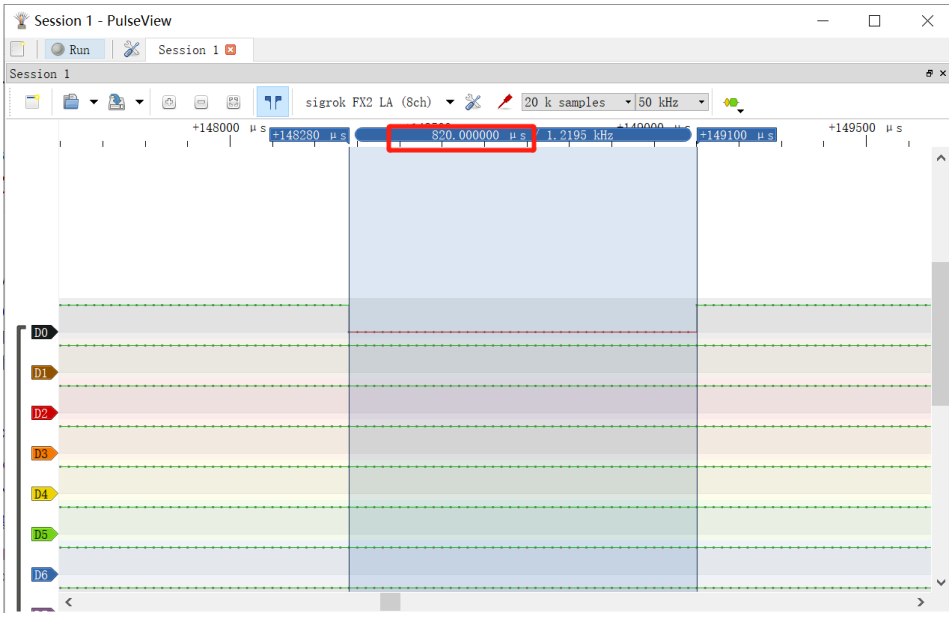
当上位机通过 SCI 向开发板发送命令‘T’（单个字符‘T’）时，F28335 芯片将信号的周期估计值、FFT 算法的执行时间发送回上位机，可通过串口调试助手显示，如下图所示。左图为原信号周期为 3，FFT 运算后所得周期为 3.511，频率为 0.011；右图为原信号周期为 11，FFT 运算后所得周期为 10.799，频率为 0.040。





（3）任务 3：测量代码的执行时间

方法一：修改代码在前后加上//LED1 TOGGLE;当代码开始执行时，LED1 对应引脚的电平就会翻转一次；当代码执行反比时，LED1 对应引脚的电平也会翻转一次；用逻辑分析仪接上 LED1 的引脚（GPIO_68），可查看其电平变化波形，当翻转两次电平时，所耗费时长为 820 μs，测量结果非常准确。如下图所示。



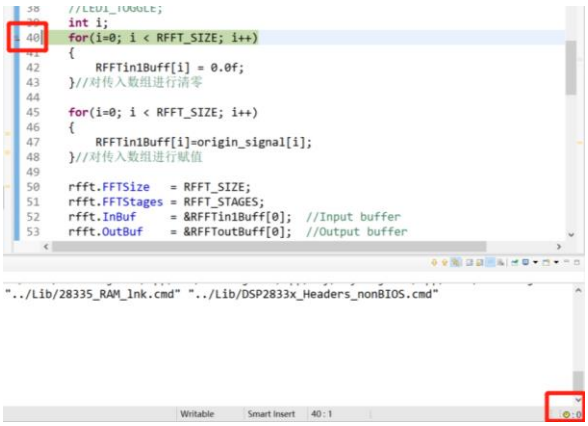

```
1. void FFT(double origin_signal[])
2. {
3. //LED1 TOGGLE;
4.      int i;
5.      for(i=0; i < RFFT_SIZE; i++)
6.      {
7.          RFFTin1Buff[i] = 0.0f;
8.      }//对传入数组进行清零
9.      for(i=0; i < RFFT_SIZE; i++)
10.     {
11.         RFFTin1Buff[i]=origin_signal[i];
12.     }//对传入数组进行赋值
13.     rfft.FFTSize   = RFFT_SIZE;
14.     rfft.FFTStages = RFFT_STAGES;
```

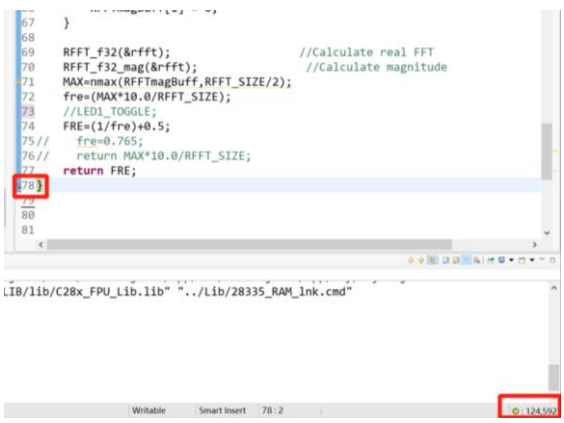
```
15.     rfft.InBuf      = &RFFTin1Buff[0];  //输入缓冲区
16.     rfft.OutBuf     = &RFFToutBuff[0];  //输出缓冲区
17.     rfft.CosSinBuf = &RFFTF32Coef[0];  //旋转因子缓冲区
18.     rfft.MagBuf     = &RFFTmagBuff[0];  //幅度缓冲区
19.     RFFT_f32_sincostable(&rfft);
20.     for (i=0; i < RFFT_SIZE; i++)
21.     {
22.         RFFToutBuff[i] = 0;
23.     }
24.     for (i=0; i < RFFT_SIZE/2; i++)
25.     {
26.         RFFTmagBuff[i] = 0;
27.     }
28.
29.     RFFT_f32(&rfft);                    //计算 FFT 的幅值
30.
31.     RFFT_f32_mag(&rfft);
32.     MAX=nmax(RFFTmagBuff,RFFT_SIZE/2);
33.     fre=MAX*10.0/RFFT_SIZE;
34. //LED1 TOGGLE;
35. }
```

方法二：在仿真器和 CCS 软件构筑的调试环境下，使用断点、单步运行和时钟功能，测量某段代码运行所消耗的时钟周期数，再根据器件的主频（时钟周期的倒数）折算成时间。如下两图所示在程序的起始与结束都打上了断点，右下脚显示运行该段程序所消耗的时钟周期。

1/150MHz*124592*1000≈830.6μs。

对比以上两种方法，所测量出的结果相近，其中方法一较为准确，因而测量出该段代码的执行时间为 820 μ s。



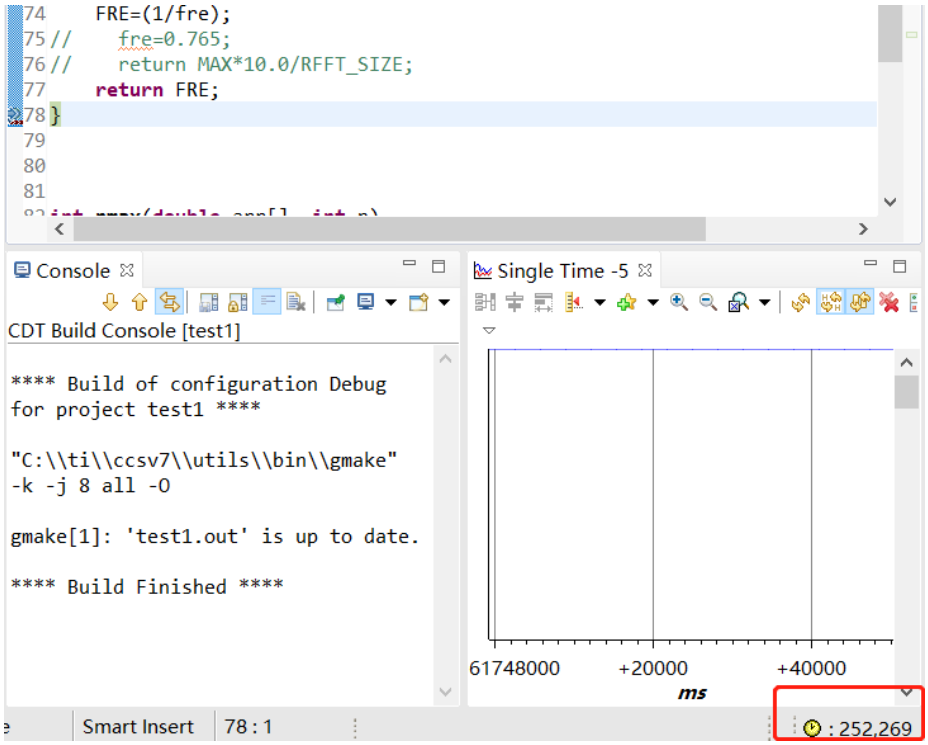


三、结果分析

结合频谱分析的理论，对信号的周期的测量结果进行分析，阐述准确性和精度与哪些因素有关，尝试改善测量结果。

分析：

每个数据之间的频率差 = 采样频率/点数。FFT 的变换区间 N 误差主要来自于用 FFT 作频谱分析时，得到的是离散谱，而信号（周期信号除外） 是连续谱，只有当 N 较大时，离散谱的包络才能逼近于连续谱，因此 N 要适当选择大一些，如果点数太少；采样频率太低会导致频谱分辨率低，无法精确表达某些过低的频率。因此遇到低频信号时，可以增加采样点数。但是同时也会带来运行速度变慢的问题。比如将本实验中用的 512 点数据用于处理，改为 1024 点数据处理，那么 FFT 的运行时间将翻倍。



不难看出，消耗的时钟周期翻倍，最终算出来的结果大约为 1.6ms

周期信号的频谱是离散谱，只有用整数倍周期的长度作 FFT 得到的离散谱才能代表周期信号的频谱。对模拟信号进行频谱分析时，首先要按照采样定理将其变成时域离散信号。如果是模拟周期信号，也应该选取整数倍周期的长度，经过采样后形成周期序列，按照周期序列的频谱分析进行。

四、实验总结

在这次课程设计中，我们收获颇丰。一方面，我们既进一步了解 DSP 这门课程的内容，强化了对课本上知识的应用，还通过自主查找网络上的资料，自学学习到了许多课本上没有的专业知识；另一方面，强调动手操作调试的课程设计内容也很好地训练了我们自我动手做项目的能力。

本次课程设计还提高了我们的综合素质。首先，在课程设计的过程中，组员们通过独立或合作解决各种问题进一步培养了独立思考问题、共同合作的能力。其次，这次课程设计也给了我们动手操作的机会，在进行课程设计的过程中，我们复习了以前学习过的知识，并掌握了一些知识在实际应用时采用的技巧等。这次作业也使我们培养了团队精神，并且在以后的工作中会更加注重团队的合作。

总得来说，这次课程设计对我们来说是一次非常有意义的经历。

五、附录

任务分配表

姓名	学号	分工	任务
陈嘉维	202034310204	组长	主代码编写整合，联合调试
罗晟宾	202034310214	组员	测量代码使用时间，撰写实验报告全文
邱致远	201934910319	组员	查找资料，调用官方 FFT 库函数，调整报告格式