

轱辘轩辕

昵称： 轱辘轩辕
园龄： 8年8个月
粉丝： 9
关注： 4
[+加关注](#)

	2023年5月					
日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

搜索

找找看

谷歌搜

常用链接

我的随笔

博客园 首页 新随笔 联系 管理 订阅 

随笔- 7 文章- 8 评论- 2 阅读- 46527

STM32的DSP库的应用

前些天理解了fft变换的理论和对其工程应用进行了实例分析，详见我的名为《C语言实现fft理论基础与工程应用的实例分析》的博客，用C语言编写的fft算法比较容易看懂，但带来的缺点就是执行效率低，对于要求实时操作（例如电机控制）的反应速度不够灵敏。本篇内容将简要分析STM32自带的DSP库文件，其用汇编语言编写，代码执行效率明显优于C语言，ST公司封装好了了库文件，我们不必看懂其汇编代码，只要会调用接口函数即可。

1.代码分析

首先我们需要在一个已经建立好的工程文件里添加如下编译路径：

```
..\Libraries\STM32F10x_DSP_Lib\inc
..\Moudle\DSP_test
```

工程需要添加的文件如下图：

我的评论
我的参与
最新评论
我的标签
更多链接

随笔档案

2014年11月(4)

2014年10月(3)

文章分类

C++(1)

Linux(2)

uC/OS-III(3)

传感器系列(2)

电源类(1)

功率放大器(1)

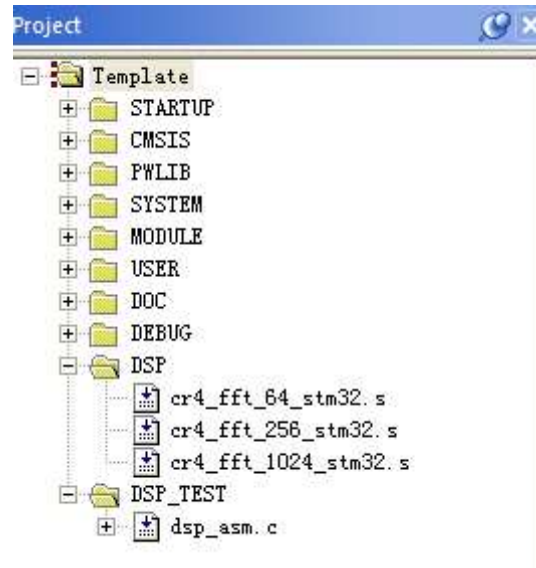
算法集锦(4)

运算放大器(1)

阅读排行榜

1. STM32的DSP库的应用(9438)
2. OPA2134+LM3886立体声功放(7291)
3. C语言实现fft理论基础与工程应用的实例分析(6388)
4. LMS自适应维纳滤波器(5704)
5. 任务内建消息队列——OSTask Q? ? ? () (3437)

评论排行榜



为了产生fft变换信号，我们可以自己产生个采样信号：使用三角函数生成采样点，供FFT计算， $f_x = 4000 * \sin(\pi/2 * i * 50.0/F_s) + 4000 * \sin(\pi/2 * i * 2500.0/F_s) + 4000 * \sin(\pi/2 * i * 2550.0/F_s)$ 。

模拟采样数据,采样数据中包含3种频率正弦波：50Hz，2500Hz，2550Hz，IBUFIN数组中，每个单元数据高字(高16位)中存储采样数据的实部，低字(低16位)存储采样数据的虚部(总是为0)。

其中dsp_asm_init()函数的作用是产生采样信号，实际工程应用中我们使用的是ADC采样的处理值。dsp_asm_test()函数的作用是进行FFT变换，并计算各次谐波幅值。具体代码参见下面：

```

1  /*
2  ****
3  *
4  *
5  *
6  *
7  *
8  *
9  *
10 *
11 ****
12 */

MICIRUM BOARD SUPPORT PACKAGE

(c) Copyright 2007; Micrium, Inc.; Weston, FL

All rights reserved. Protected by international copyright laws.
Knowledge of the source code may NOT be used to develop a similar product.
Please help us continue to provide the Embedded community with the finest
software available. Your honesty is greatly appreciated.

```

1. LMS自适应维纳滤波器(2)

推荐排行榜

1. C语言实现fft理论基础与工程应用的实例分析(3)

2. LMS自适应维纳滤波器(2)

3. STM32的DSP库的应用(2)

最新评论

1. Re:LMS自适应维纳滤波器

@lmk92 你好，博主最下面分享的源文件可以找到吗。我现在需要完整的程序，如果可以找到，希望可以告诉我，有偿...

--1234555

2. Re:LMS自适应维纳滤波器

博主您好，请问您的博文中用到的图片，是哪本书的，您是否可以推荐一下教程

--lmk92

```

13
14 /*
15 ****
16 *
17 *                                BOARD SUPPORT PACKAGE
18 *
19 *                                ST Microelectronics STM32
20 *                                with the
21 *                                STM3210B-EVAL Evaluation Board
22 *
23 * Filename      : bsp.c
24 * Version       : V1.00
25 * Programmer(s) : Brian Nagel
26 ****
27 */
28
29 /*
30 ****
31 *                                INCLUDE FILES
32 ****
33 */
34
35 #define  DSP_ASM
36 #include "stm32f10x.h"
37 #include "dsp_asm.h"
38 #include "stm32_dsp.h"
39 #include "table_fft.h"
40 #include <stdio.h>
41 #include <math.h>
42
43
44 /*
45 ****
46 *                                LOCAL CONSTANTS
47 ****
48 */
49 #define PI2 6.28318530717959
50 #define NPT_1024 1024
51 // #define NPT_256 256

```

```
52 // #define NPT_1024 1024
53
54 // N=64, Fs/N=50Hz, Max(Valid)=1600Hz
55 #ifdef NPT_64
56 #define NPT 64
57 #define Fs 3200
58 #endif
59
60 // N=256, Fs/N=25Hz, Max(Valid)=3200Hz
61 #ifdef NPT_256
62 #define NPT 256
63 #define Fs 6400
64 #endif
65
66 // N=1024, Fs/N=5Hz, Max(Valid)=2560Hz
67 #ifdef NPT_1024
68 #define NPT 1024
69 #define Fs 5120
70 #endif
71
72
73 /*
74 ****
75 *
76 ****
77 */
78
79
80 /*
81 ****
82 *
83 ****
84 */
85
86
87 /*
88 ****
89 *
90 ****
```

```

91 */
92 long lBUFIN[NPT];          /* Complex input vector */
93 long lBUFOUT[NPT];         /* Complex output vector */
94 long lBUFMAG[NPT]; /* Magnitude vector */
95 /*
96 ****
97 *                                LOCAL FUNCTION PROTOTYPES
98 ****
99 */
100 void dsp_asm_powerMag(void);
101
102 /*
103 ****
104 *                                LOCAL CONFIGURATION ERRORS
105 ****
106 */
107
108
109 /*
110 ****
111 ****
112 **                                Global Functions
113 ****
114 ****
115 */
116
117 void dsp_asm_init()
118 {
119     u16 i=0;
120     float fx;
121     for(i=0;i<NPT;i++)
122     {
123         fx = 4000 * sin(PI2*i*50.0/Fs) + 4000 * sin(PI2*i*2500.0/Fs) + 4000*sin(PI2*i*2550.0/Fs);
124         lBUFIN[i] = ((long)fx)<<16;
125     }
126 }
127
128 void dsp_asm_test()
129 {

```

```
130
131 #ifdef NPT_64
132     cr4_fft_64_stm32(lBUFOUT, lBUFIN, NPT);
133 #endif
134
135 #ifdef NPT_256
136     cr4_fft_256_stm32(lBUFOUT, lBUFIN, NPT);
137 #endif
138
139 #ifdef NPT_1024
140     cr4_fft_1024_stm32(lBUFOUT, lBUFIN, NPT);
141 #endif
142
143 // 计算幅值
144 dsp_asm_powerMag();
145
146 }
147
148 void dsp_asm_powerMag(void)
149 {
150     s16 lX,lY;
151     u32 i;
152     for(i=0;i<NPT/2;i++)
153     {
154         lX = (lBUFOUT[i] << 16) >> 16;
155         lY = (lBUFOUT[i] >> 16);
156         {
157             float X    = NPT * ((float)lX) /32768;
158             float Y    = NPT * ((float)lY) /32768;
159             float Mag = sqrt(X*X + Y*Y)/NPT;
160             lBUFMAG[i] = (u32)(Mag * 65536);
161         }
162     }
163 }
```



```

1  /*
2  ****
3  *
4  *
5  *
6  *
7  *
8  *
9  *
10 *
11 ****
12 */
13
14 /*
15 ****
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 * Filename      : bsp.h
24 * Version       : V1.00
25 * Programmer(s) : Brian Nagel
26 ****
27 */
28
29 #ifndef __DSP_ASM_H__
30 #define __DSP_ASM_H__
31
32 /*
33 ****
34 *
35 ****
36 */
37
38 #ifdef DSP_ASM
39 #define DSP_EXT

```

```
40 #else
41 #define DSP_EXT extern
42 #endif
43
44 /*
45 ****
46 * INCLUDE FILES
47 ****
48 */
49
50
51
52
53 /*
54 ****
55 * GLOBAL VARIABLES
56 ****
57 */
58
59
60 /*
61 ****
62 * MACRO'S
63 ****
64 */
65
66
67 /*
68 ****
69 * FUNCTION PROTOTYPES
70 ****
71 */
72
73 void dsp_asm_test(void);
74 void dsp_asm_init(void);
75
76 #endif /* End of module include.
```



着重分析下dsp_asm_powerMag()函数的作用，其函数就是求幅值，首先定义的的一个16位的有符号的数据IX 和IY 这两个只是中间变量，然后定义的i，是32位的无符号型。语句的目的是 $\text{Mag} = \sqrt{X^2 + Y^2} / \text{NPT}$ 。但直接这么写不符合DSP的计算习惯也就是不符合浮点运算的习惯。因此语句在for函数i写道 $\text{IX} = (\text{IBUFOUT}[i] \ll 16) \gg 16$ 就是取32位的i的低16位数据， $\text{IY} = (\text{IBUFOUT}[i] \gg 16)$ 是取高16位数据。下面的两句

```
float X = NPT * ((float)IX) / 32768;
```

```
float Y = NPT * ((float)IY) / 32768
```

目的就是把数据浮点化，至于为什么是除以32768。可以这么说，浮点化就好像10进制里面的科学计数法。32768=2的15次。除以32768也就是去除了浮点数后面的那个基数，只剩下前面的。比如1991 改写成 1.991×10 的三次幂，再除以10的三次方，只剩下1.991，便于余下的运算。至于最后一句要乘以65536是因为我们定义的数据和我们要求得的数据都是无符号32位的，之前已经把32位的数据拆开又分别浮点化了又开了个根号，所以再把它变回来 只需要乘以2的16次，也就是65536.比如说问你什么时候生日，你说是19911030，然而DSP是不习惯这么干的，他需要把它拆开为1991和1030。再写成 1.991×10 的3次方和 1.030×10 的3次方。然后才能进行其他的运算。

这里是ST公司采用了DSP专用芯片（主要是指TI）的写法，也就是说尽管DSP的芯片类型很多，数据变量的定义也各有差异，但原理是一样的，最终还是要采用DSP习惯的运算方式。至于为什么一定要采用浮点运算，因为机器是傻子，然而TI公司的工程师是天才。

main函数中我们只需在while（1）前加上dsp_asm_init(); dsp_asm_test();即可。

2, 实验现象

注意FFT运算结果的对称性，也即256点的运算结果，只有前面128点的数据是有效可用的。

① $N=64, F_s/N=50\text{Hz}, \text{Max(Valid)}=1600\text{Hz}$ ，64点FFT，采样率3200Hz,频率分辨率50Hz，测量最大有效频率1600Hz

64点FFT运算结果图（局部）：

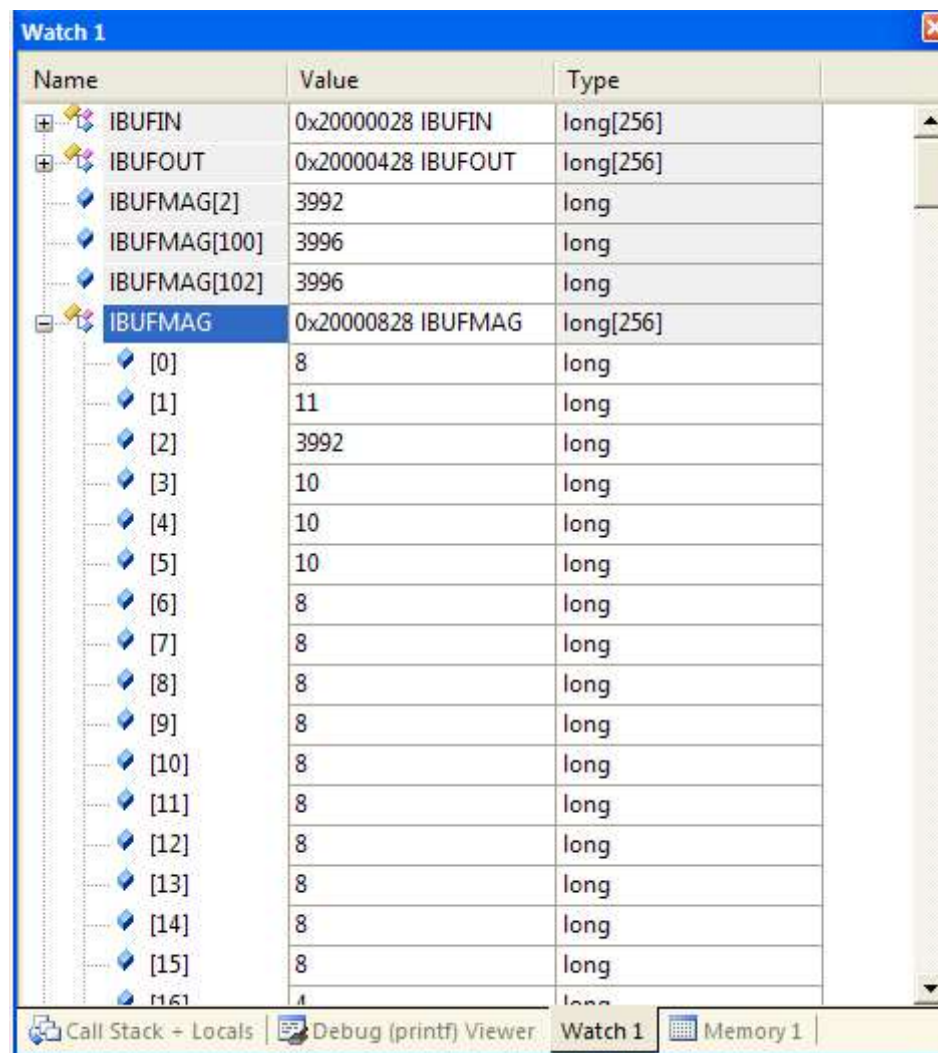
Name	Value	Type
IBUFIN	0x20000028 IBUFIN	long[64]
IBUFOUT	0x20000128 IBUFOUT	long[64]
IBUFMAG[1]	3992	long
IBUFMAG[50]	0	long
IBUFMAG[510]	0	long
IBUFMAG	0x20000228 IBUFMAG	long[64]
[0]	6	long
[1]	3992	long
[2]	7	long
[3]	7	long
[4]	2	long
[5]	4	long
[6]	4	long
[7]	6	long
[8]	4	long
[9]	6	long
[10]	0	long
[11]	6	long
[12]	2	long
[13]	4002	long
[14]	4004	long
[15]	2	long
[16]	2	long

上图中，数组下标X对应的谐波频率为： $N \times F_s / 64 = N \times 3200 / 64 = N \times 50\text{Hz}$ 。

IBUFMAG[1] 对应 50Hz谐波幅值。

上图中由于FFT分辨率50HZ，最大只能识别1600Hz谐波，导致结果中出现错误的的数据。

② $N=256, F_s/N=25\text{Hz}, \text{Max(Valid)}=3200\text{Hz}$ ，256点FFT，采样率6400Hz,频率分辨率25Hz，测量最大有效频率3200Hz
256点FFT运算结果图（局部）：



Name	Value	Type
IBUFIN	0x20000028 IBUFIN	long[256]
IBUFOUT	0x20000428 IBUFOUT	long[256]
IBUFMAG[2]	3992	long
IBUFMAG[100]	3996	long
IBUFMAG[102]	3996	long
IBUFMAG	0x20000828 IBUFMAG	long[256]
[0]	8	long
[1]	11	long
[2]	3992	long
[3]	10	long
[4]	10	long
[5]	10	long
[6]	8	long
[7]	8	long
[8]	8	long
[9]	8	long
[10]	8	long
[11]	8	long
[12]	8	long
[13]	8	long
[14]	8	long
[15]	8	long

上图中，数组下标X对应的谐波频率为： $N \times F_s / 256 = N \times 6400 / 256 = N \times 25\text{Hz}$.

IBUFMAG[2] 对应 $2 \times 25 = 50\text{Hz}$ 谐波幅值

IBUFMAG[100] 对应 $100 \times 25 = 2500\text{Hz}$ 谐波幅值

IBUFMAG[102] 对应 $102 \times 25 = 2550\text{Hz}$ 谐波幅值

③ $N=1024, F_s/N=5\text{Hz}, \text{Max(Valid)}=2560\text{Hz}$, 1024点FFT, 采样率5120Hz, 频率分辨率5Hz, 测量最大有效频率2560Hz

1024点FFT运算结果图（局部）：

Name	Value	Type
IBUFIN	0x20000028 IBUFIN	long[1024]
IBUFOUT	0x20001028 IBUFOUT	long[1024]
IBUFMAG[10]	3992	long
IBUFMAG[500]	4000	long
IBUFMAG[510]	3998	long
IBUFMAG	0x20002028 IBUFMAG	long[1024]
[0]	8	long
[1]	14	long
[2]	12	long
[3]	14	long
[4]	12	long
[5]	13	long
[6]	11	long
[7]	8	long
[8]	11	long
[9]	11	long
[10]	3992	long
[11]	8	long
[12]	10	long
[13]	8	long
[14]	10	long
[15]	8	long
[16]	10	long

上图中，数组下标X对应的谐波频率为： $N \times F_s / 1024 = N \times 5120 / 1024 = N \times 5 \text{Hz}$.

IBUFMAG[10] 对应 $10 \times 5 = 50 \text{Hz}$ 谐波幅值

IBUFMAG[500] 对应 $500 \times 5 = 2500 \text{Hz}$ 谐波幅值

IBUFMAG[510] 对应 $510 \times 5 = 2550 \text{Hz}$ 谐波幅值

总结：该工程中模拟信号源为： $4000 * \sin(\text{PI}2*i*50.0/\text{Fs}) + 4000 * \sin(\text{PI}2*i*2500.0/\text{Fs}) + 4000*\sin(\text{PI}2*i*2550.0/\text{Fs})$ 。

信号为1个50Hz、1个2500Hz、1个2550Hz的正弦波混合信号，幅值为均为4000。

3，工程源码下载地址

LL STM32-DSP-fft.rar
115网盘礼包码：5lbdd57l2hkq
<http://115.com/lb/5lbdd57l2hkq>

分类: [算法集锦](#)

好文要顶

关注我

收藏该文



轱辘轩辕
粉丝 - 9 关注 - 4

20

[+加关注](#)

« 上一篇: [C语言实现fft理论基础与工程应用的实例分析](#)
» 下一篇: [开关电源基础知识](#)

posted @ 2014-10-26 22:45 [轱辘轩辕](#) 阅读(9439) 评论(0) [编辑](#) [收藏](#) [举报](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

- 【推荐】园子的商业化努力-AI人才服务：招募AI导师，一起探索AI领域的机会
- 【推荐】中国云计算领导者：阿里云轻量应用服务器2核2G低至108元/年
- 【推荐】第五届金蝶云苍穹低代码开发大赛正式启动，百万奖金等你拿！

编辑推荐:

- 记一次 Visual Studio 2022 卡死分析
- 异常体系与项目实践
- .NET 通过源码深究依赖注入原理
- [趣话计算机底层技术] 一个故事看懂各种锁
- 记一次 .NET 某医院门诊软件 卡死分析

阅读排行:

- 【源码解读】asp.net core源码启动流程精细解读
- 我写了本开源书:《3D编程模式》
- IntelliJ IDEA一站式配置【全】(提高开发效率)
- 记一次 Oracle 下的 SQL 优化过程
- Create Vite App 支持 OpenTiny 啦

Copyright © 2023 轱辘轩辕

Powered by .NET 7.0 on Kubernetes