

IQMATH 的使用

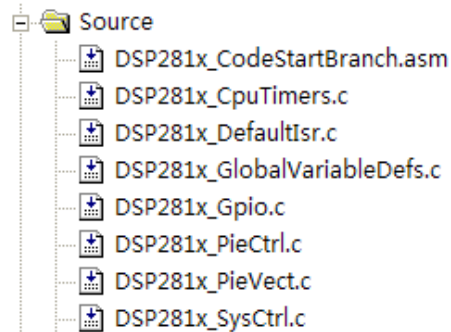
Hdu freescale lab

yu554377214@126.com

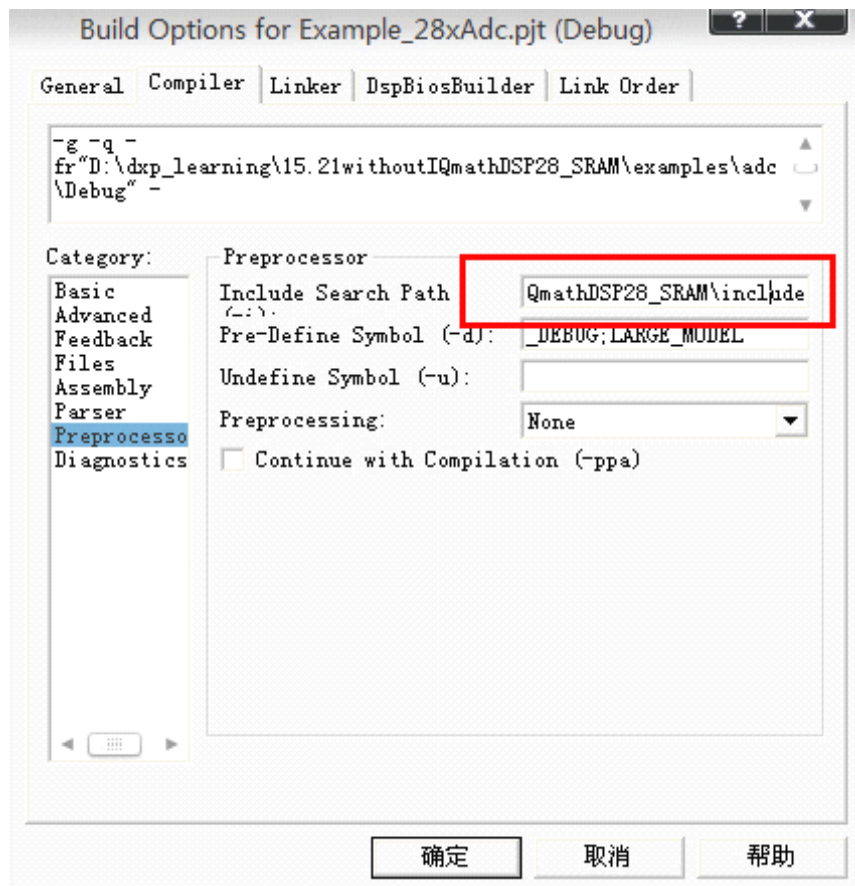
在 F2812 的 ROM 中，有 $3K \times 16$ 位被保留用于存放数学公式表以及未来的开发。主要应用于高速度和高精度的实时计算，比同等程度的 ANSI C 语言效率更高，同时可以节省用户更多的设计和调试时间。

为了应用 IQmath，首先要从 TI 官方网站下载 IQmath 库，文档名称为 SPRC087。我们主要应用库里面的：IQmath.cmd，IQmathLib.h，IQmath.lib。

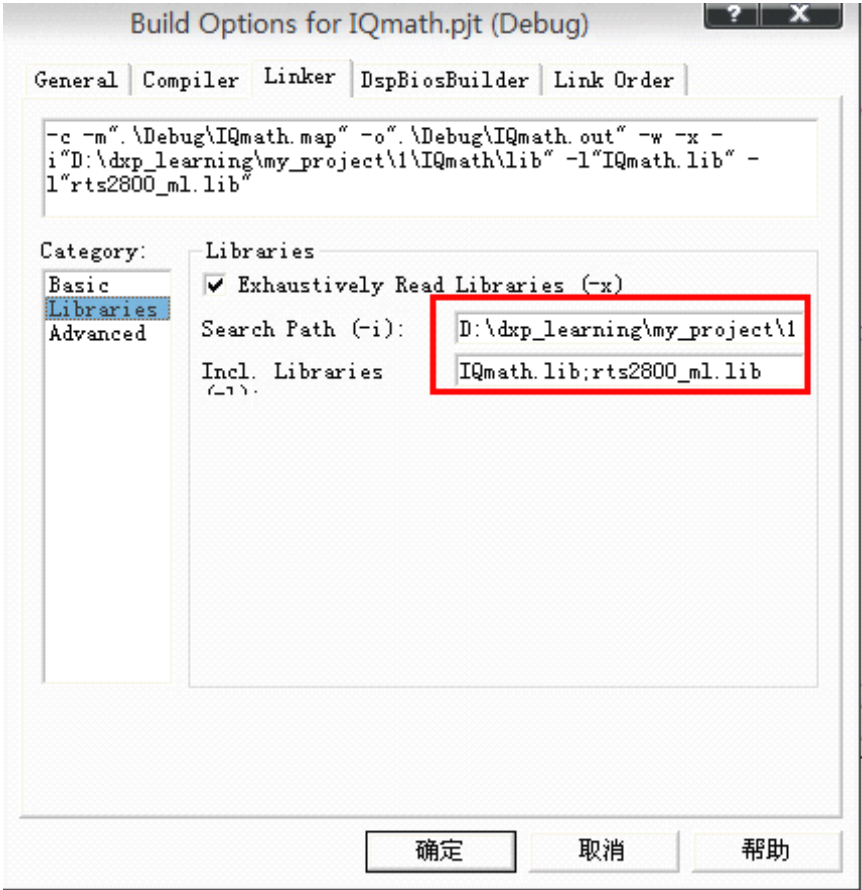
1.在ccs 中新建一个空的工程iqmath.pjt。添加如下源文件（源文件根据自己的需求添加）。



2.然后编写头文件所在目录（具体目录，你们可能与我的不相同，我将 2812 的所有头文件都放在一个文件夹下，这样方便以后使用）

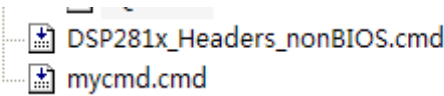


3.然后添加库文件



添加lib文件用，文件名必须一致，然后中间用”;”隔开。

4.添加下面2个cmd 文件。



具体说明：

IQMATH 段和查找表

一些IQmath 函数需要使用查找表。这些表中很多嵌入在了 28x 设备的boot rom 中。IQmathTables 段中包含一些经常使用的 IQmath 函数的查找表，包括 IQdiv, IQsin, IQcos, IQsqrt 和 IQatan2。IQmathTables 存在于所有 28x 系列的 boot room 中，因此，在CMD 文件中，这个段必须被识别为“NOLOAD”类型。

Device	Assembly Section Name (i.e. .sect)	Boot ROM Location
281x	IQmathTables	0x3FF000

IQmath Linker Command File Example: 281x Devices

```
MEMORY
{
  PAGE 0:
    PRAMH0 (RW)      : origin = 0x3f8000, length = 0x001000
  PAGE 1:
    IQTABLES (R)      : origin = 0x3FF000, length = 0x000b50
    DRAMH0 (RW)       : origin = 0x3f9000, length = 0x001000
}
SECTIONS
{
  IQmathTables       : load = IQTABLES, type = NOLOAD, PAGE = 1
  IQmathTablesRam    : load = DRAMH0, PAGE = 1
  IQmath             : load = PRAMH0, PAGE = 0
}
```

按照上述事例，新建一个cmd输入以下代码，或者将文档里的 mycmd.cmd 加到工程里面去。

```
-stack 400H

MEMORY
{
  PAGE 0 :

    RAMM0      : origin = 0x000000, length = 0x000400

    BEGIN      : origin = 0x3F8000, length = 0x000002
    RAMH0      : origin = 0x3F8002, length = 0x001FFE
    RESET      : origin = 0x3FFFC0, length = 0x000002
    VECTORS     : origin = 0x3FFFC2, length = 0x00003E

    SRAM       : origin = 0x0080000, length = 0x020000

    IQTABLES(R) : origin = 0x3FF000, length = 0x000b50

  PAGE 1 :

    RAMM1      : origin = 0x000400, length = 0x000400
    RAMLO      : origin = 0x008000, length = 0x001000
    RAML1      : origin = 0x009000, length = 0x001000
}
```

```

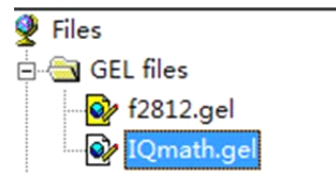
SECTIONS
{
    /* Setup for "boot to H0" mode:
       The codestart section (found in DSP28_CodeStartBranch.asm)
       re-directs execution to the start of user code.
       Place this section at the start of H0 */
    vectors          : > VECTORS          PAGE = 0
    codestart         : > BEGIN,           PAGE = 0
    ramfuncs          : > RAMH0           PAGE = 0
    .text             : > RAMH0,          PAGE = 0
    .cinit             : > RAMH0,          PAGE = 0
    .pinit             : > RAMH0,          PAGE = 0
    .switch            : > RAMM0,          PAGE = 0
    .reset             : > RESET,          PAGE = 0, TYPE = DSECT /* not used, */

    .stack            : > RAMM1,           PAGE = 1
    .ebss              : > RAML1,           PAGE = 1
    .econst            : > RAML1,           PAGE = 1
    .esysmem           : > RAML1,           PAGE = 1

    IQmathTables       : > IQTABLES,        PAGE = 0, type = NOLOAD
    IQmath              : > RAMH0,          PAGE = 0
}

```

5.添加GEL文件，这个是用CCS调试的时候用的



到此Iqmath是添加完成了，接下来就是Iqmath的应用了！

一个例题：

现在来编写 main.c 函数。

如图所示:

PS: 我加入了 GPIO 程序，就一个简单的翻动作。因为之前我编译的时候没有添加进去，仅仅一个 `mycmd.cmd` 文件，编译还是能通过的。***IQ 指令也能正常执行***，也能通过 watch 窗口观察数值变化，但是实际用示波器观察端口值不会翻转变

化。这也是我发现的一个不知名 bug！所以加入一段 GPIO 的程序来测试下。

```

#include "DSP281x_Device.h"
#include "IQmathLib.h"
#define GLOBAL_Q 15 //全局Q15格式
#define PI 3.14159
#define ad_in 55000 //ADRESULEn中的值
float sinout_flt, ad_flt;
_iq sinout_iq, ad_iq;
_iq a, b;

void main(void)
{
    InitSysCtrl();
    InitGpio();
    //InitXintf();
    DINT;
    IER=0X0000;
    IFR=0X0000;
    GpioDataRegs.GPADAT.all=0xffff;

    sinout_iq=_IQsin(_IQmpy(_IQ(0.25), _IQ(PI))); //求sin( $\pi/4$ )



    ad_iq=_IQdiv(_IQ(ad_in), _IQ(21840)); //这里是计算一个AD的值, 21840=65520/3

    sinout_flt=_IQtoF(sinout_iq); //转换浮点数
    ad_flt=_IQtoF(ad_iq);

    for(;;)
    {
        GpioDataRegs.GPADAT.all=~GpioDataRegs.GPADAT.all; //电平翻转
        for(a=0; a<2000; a++) //简易延时
        {
            for(b=0; b<500; b++)
            {
            }
        }
    }
}

```

编译完后将程序烧入 DSP 中运行，在观察窗口可见（我的工程中 GLOBAL_Q 为15）

Watch Window		
Name	Value	Radix
sinout_iq	23169	dec
sinout_flt	0.7070618	float
ad_iq	82520	dec
ad_flt	2.518311	float
<div>  Watch Locals  Watch 1 </div>		

另一个例子

```
#include "DSP281x_Examples.h"

#define GLOBAL_Q 16
#include "IQmathLib.h"

#define PI 3.14159

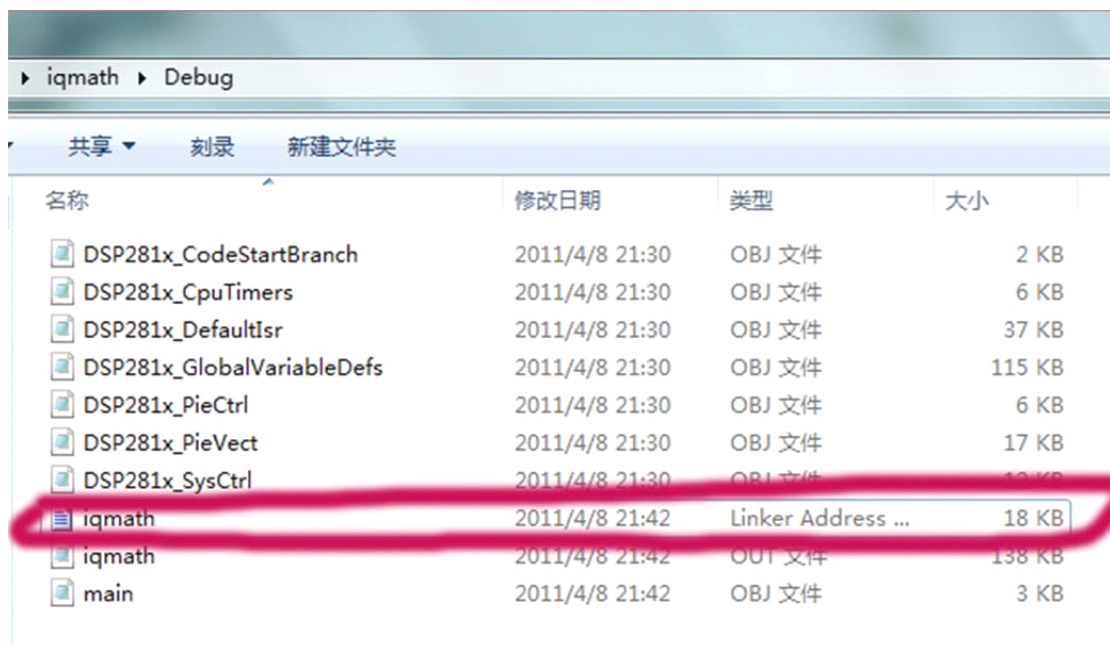
long GlobalQ = GLOBAL_Q;

#define PWM_PERIOD 500
#define PWM_DUTY 0.5
_iq Ua;
_iq a = _IQ(1);
void main()
{
    InitSysCtrl();
    DINT;
    IER = 0x0000;
    IFR = 0x0000;

    Ua = _IQmpy(_IQ(PWM_PERIOD), _IQ(PWM_DUTY));

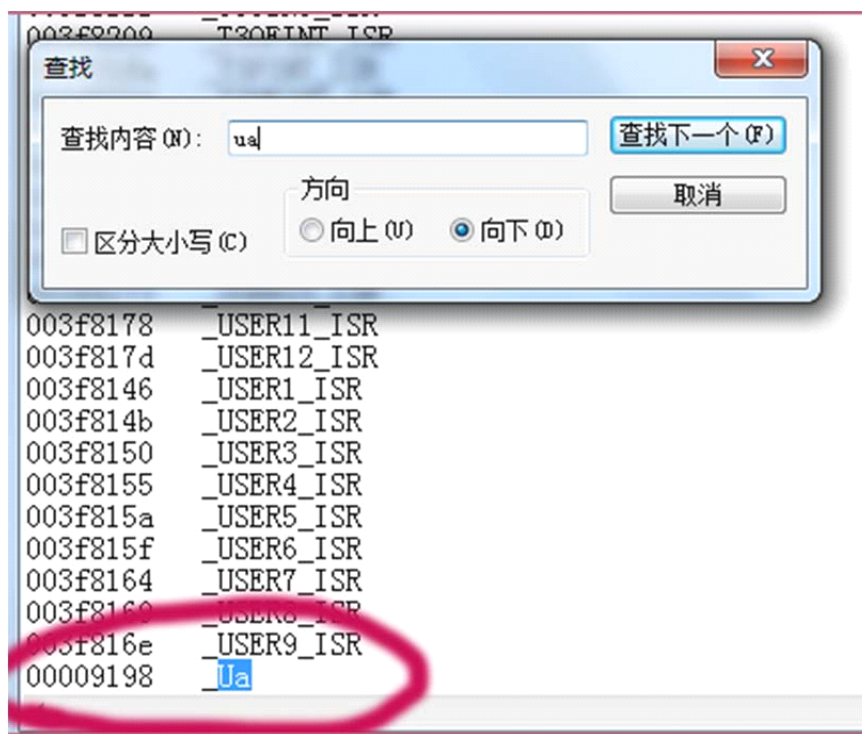
    for(;;)
    {
    }
}
```

在你的工程所在的文件夹里，进入 DEBUG 文件夹。



iqmath \> Debug				
共享 刻录 新建文件夹				
名称	修改日期	类型	大小	
DSP281x_CodeStartBranch	2011/4/8 21:30	OBJ 文件	2 KB	
DSP281x_CpuTimers	2011/4/8 21:30	OBJ 文件	6 KB	
DSP281x_DefaultIsr	2011/4/8 21:30	OBJ 文件	37 KB	
DSP281x_GlobalVariableDefs	2011/4/8 21:30	OBJ 文件	115 KB	
DSP281x_PieCtrl	2011/4/8 21:30	OBJ 文件	6 KB	
DSP281x_PieVect	2011/4/8 21:30	OBJ 文件	17 KB	
DSP281x_SysCtrl	2011/4/8 21:30	OBJ 文件	12 KB	
iqmath	2011/4/8 21:42	Linker Address ...	18 KB	
iqmath	2011/4/8 21:42	OUT 文件	138 KB	
main	2011/4/8 21:42	OBJ 文件	3 KB	

打开上图圈圈中的文件。



搜索 Ua，即可找到程序烧入 DSP 后，该变量在内存中的具体位置。

在 View 菜单中选择 Memory 选项，输入 0x00009198，回车，便可看到下图。

0x0000919a			
0x00009190	CpuTimer0		
0x00009190	0xE46F	0xF3F9	0x8442
0x00009193	0x67A8	0xA1EE	0xDA60
0x00009196	0x2E42	0xDD9F	
0x00009198	Ua		
0x00009198	0x0000	0x00FA	
0x0000919A	a		
0x0000919A	0x0000	0x0001	
0x0000919C	GlobalQ		
0x0000919C	0x0010	0x0000	
0x0000919E	_unlock		
0x0000919E	0x8090	0x003F	
0x000091A0	_lock		
0x000091A0	0x8090	0x003F	0x228A
0x000091A3	0x5A95	0x0EF7	0x6251

Ua 的值是 250，GLOBAL_Q 的格式为 16，相当于一个 32 位数中，小数点在第 16 位，故 Ua 的“原本”（即在内存中的实际存储）的值为 $250 \ll 16$ ，2 进制表示为 0x00FA0000。同理，变量 a 的值“原本”为 0x00010000，但通过 IQ 表示后，即为 1。

Q 格式的选择

IQmath 一共提供了 30 种 Q 格式，具体选择哪种格式要兼顾精度和值的大小依据下表而定：

Data Type	Range		Resolution/Precision
	Min	Max	
_iq30	-2	1.999 999 999	0.000 000 001
_iq29	-4	3.999 999 998	0.000 000 002
_iq28	-8	7.999 999 996	0.000 000 004
_iq27	-16	15.999 999 993	0.000 000 007
_iq26	-32	31.999 999 985	0.000 000 015
_iq25	-64	63.999 999 970	0.000 000 030
_iq24	-128	127.999 999 940	0.000 000 060
_iq23	-256	255.999 999 981	0.000 000 119
_iq22	-512	511.999 999 762	0.000 000 238
_iq21	-1024	1023.999 999 523	0.000 000 477
_iq20	-2048	2047.999 999 046	0.000 000 954
_iq19	-4096	4095.999 998 093	0.000 001 907
_iq18	-8192	8191.999 996 185	0.000 003 815
_iq17	-16384	16383.999 992 371	0.000 007 629
_iq16	-32768	32767.999 984 741	0.000 015 259
_iq15	-65536	65535.999 969 482	0.000 030 518
_iq14	-131072	131071.999 938 965	0.000 061 035
_iq13	-262144	262143.999 877 930	0.000 122 070
_iq12	-524288	524287.999 755 859	0.000 244 141
_iq11	-1048576	1048575.999 511 719	0.000 488 281
_iq10	-2097152	2097151.999 023 437	0.000 976 563
_iq9	-4194304	4194303.998 046 875	0.001 953 125
_iq8	-8388608	8388607.996 093 750	0.003 906 250
_iq7	-16777216	16777215.992 187 500	0.007 812 500
_iq6	-33554432	33554431.984 375 000	0.015 625 000
_iq5	-67108864	67108863.968 750 000	0.031 250 000
_iq4	-134217728	134217727.937 500 000	0.062 500 000
_iq3	-268435456	268435455.875 000 000	0.125 000 000
_iq2	-536870912	536870911.750 000 000	0.250 000 000
_iq1	-1073741824	1 073741823.500 000 000	0.500 000 000

例如将数 5.0 转为 Q 格式，只能从 _iq1~_iq28 里面选择，而不能转化为 _iq29 和 _iq30 表示，因为 _iq29 能转化的最大值为 3.999999998，否则会发生溢出。所以在定 Q 格式时要对数的范围做一下估计。也正是由于这个原因，有些三角函数不能采用 _iq30 格式。

使用IQmath GEL 文件来调试

IQmath GEL 文件包含 GEL 函数来帮助在 watch 窗口中观察 IQ 变量，并且允许通过对话框来设置 IQ 变量的值。

步骤1：定义“GlobalQ”变量 在用户的一个源文件中，添加如下语句：

```
long GlobalQ = GLOBAL_Q;
```

这个变量被GEL函数使用

步骤2：

装载IQmath.gel 到用户工程

中。 步骤3：

要在watch 窗口中观察IQ 变量，在watch 窗口中输入以下命令。

```
_IQ(VarName)      ;GLOBAL_Q value;
_IQN(VarName)     ;N=1 to 30
```

Name	Value	Type	Radix
 _IQ(sinout_iq)	0.7070923	float	float
 _IQ(a)	3.848679	float	float
			

选择GLOBAL_Q 值

在一个应用在需要考虑到数据的精度和范围，高精度必然会使数据的范围变小，因此系统设计者必须权衡精度和范围，然后设置GLOBAL_Q 的值。

情况1：

默认的 GLOBAL_Q 是 24，可以在 IQmathLib.h 文件中修改这个值，用户可以自己选择 Q1 到 Q29 作为 GLOBAL_Q 的值，修改这个值后，意味着所有的 GLOBAL_Q 函数将会使用这个值来表示多有 IQ 数据，除非这个符号在源文件中被重载。

IQmathLib.h : Selecting GLOBAL_Q format

```
#ifndef GLOBAL_Q
#define GLOBAL_Q 24 /* Q1 to Q29 */
#endif
```

情况2:

一个完整的系统可能包含各种模块，每个模块又可能要求有不同的数据精度，因此每个模块可能都要单独设置GLOBAL_Q 的值。

MODULE6.C : Selecting Local Q format

```
#define GLOBAL_Q 27 /* Set the Local Q value */
#include <IQmathLib.h>
```

记住，要在包含头文件之前定义 GLOBAL_Q 的值。

IQmath 命名规则

每种IQmath 函数都有2 种类型

GLOBAL_Q 函数：

- _IQsin(A) /* High Precision SIN */
- _IQcos(A) /* High Precision COS */
- _IQrmpy(A,B) /* IQ multiply with rounding */
- _IQmpy(A,B) /* IQ multiply */

Q 格式特殊函数

- _IQ29sin(A) /* High Precision SIN: input/output are in Q29 */
- _IQ28sin(A) /* High Precision SIN: input/output are in Q28 */
- _IQ27sin(A) /* High Precision SIN: input/output are in Q27 */
- _IQ26sin(A) /* High Precision SIN: input/output are in Q26 */
- _IQ25sin(A) /* High Precision SIN: input/output are in Q25 */
- _IQ24sin(A) /* High Precision SIN: input/output are in Q24 */