DSP 课程设计报告

设计题目：256 点 FFT

院　　系：计算机科学学院

专　　业：自动化

年　　级：2008 级

姓　　名：

学　　号：

指导教师：

2011年 11月 28日

256点FFT的实现

一、 设计目的

1、 加深对DFT算法原理和基本性质的理解；

2、 熟悉FFT的算法原理和FFT子程序的算法流程和应用；

3、 学习用FFT对连续信号和时域信号进行频谱分析的方法；

4、 学习DSP中FFT的设计和编程思想；

5、 学习使用CCS的波形观察器观察波形和频谱情况；

二、 设计内容

给定256采样点,求频谱,统计运行时间并在PC上显示。

三、 设计原理

快速傅里叶变换（FFT）是一种高效实现离散傅里叶变换（DFT）的快速算法，是数字信号处理中最为重要的工具之一，它在声学，语音，电信和信号处理等领域有着广泛的应用。

快速傅里叶变换 FFT

旋转因子 WN有如下的特性。

对称性： WNk+N/2=-WNk

周期性：WNn(N-k)=WNk(N-n)=WN-nk

利用这些特性，既可以使 DFT中有些项合并，减少了乘法积项，又可以将长序列的 DFT分解成几个短序列的 DFT。FFT就是利用了旋转因子的对称性和周期性来减少运算量的。

FFT的算法是将长序列的 DFT分解成短序列的 DFT。例如：N为偶数时，先将 N点的 DFT分解为两个 N/2 点的 DFT，使复数乘法减少一半；再将每个 N/2 点的 DFT分解成 N/4 点的 DFT，使复数乘又减少一半，继续进行分解可以大大减少计算量。最小变换的点数称为基数，对于基数为 2 的 FFT算法，它的最小变换是 2 点 DFT。

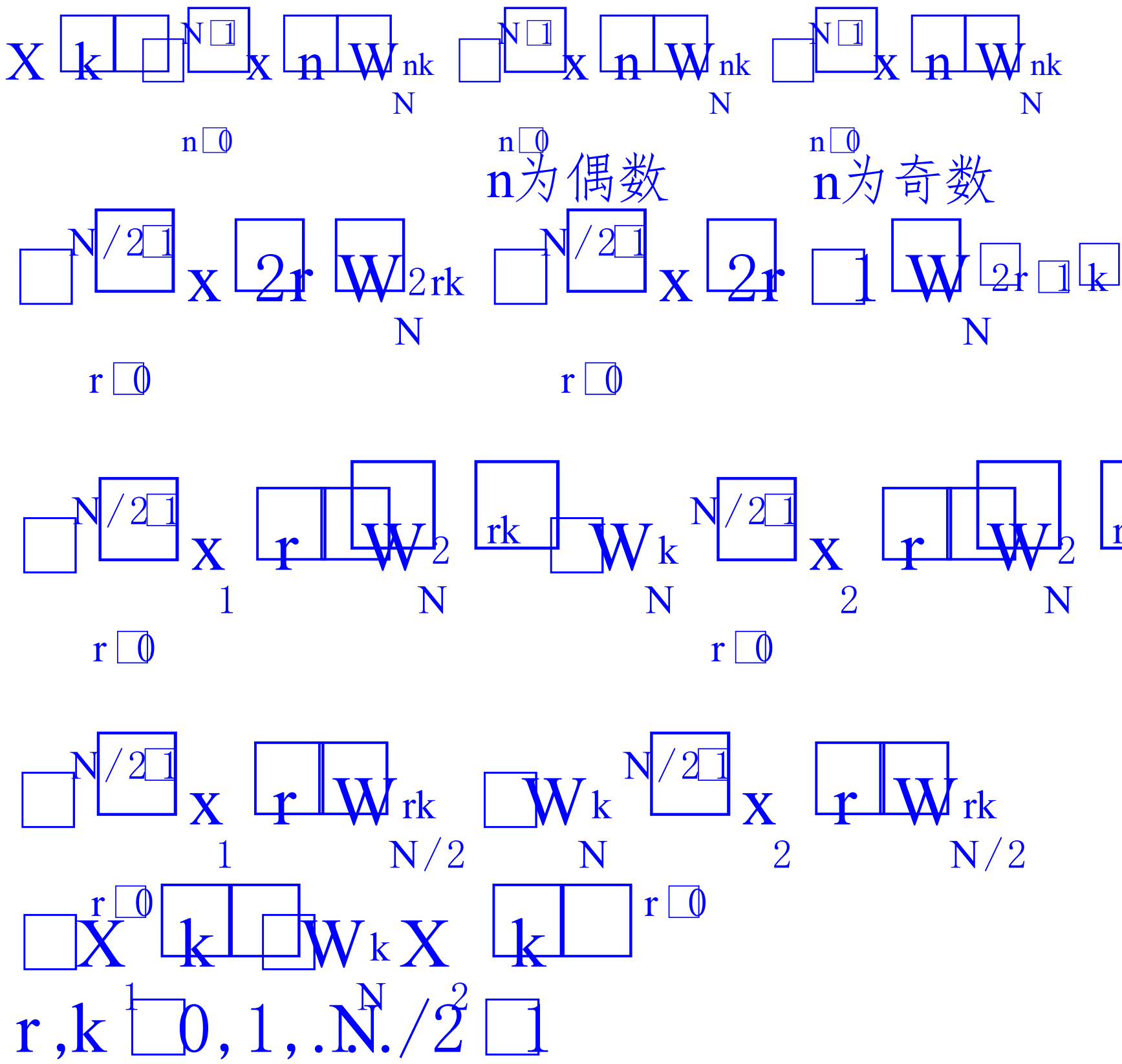
一般而言，FFT算法分为按时间抽取的 FFT（DIT　F F T）和按频率抽取的 F F T（DIF FFT）T 两大类。DIF FFT算法是在时域内将每一级输入序列依次按奇 / 偶分成 2 个短序列进行计算。而 DIF FFT算法是在频域内将每一级输入序列依次奇 / 偶分成 2 个短序列进行计算。两者的区别是旋转因子出现的位置不同，得算法是一样的。在 DIF FFT算法中，旋转因子出现在输入端，而在 DIF FFT算法中它出现在输入端。

假定序列 x(n)的点数 N是 2 的幂，按照 DIF FFT算法可将其分为偶序列和奇序列。

偶序列：x(2r)=x1(r)

奇序列：x(2r+1)=x2(r)

其中：r=0,1,2,…,N/2-1　　则 x(n)的 DFT表示为



式中，x1(k)和 x2(k)分别为 x1(r)和 x2(r)的 N/2 的 DFT。

由于对称性，

WNk+N/2=-WNk 因此，N点 DFT可分为两部分：

前半部分：x(k)=x1(k)+WkNx2(k)　　（4）

后半部分：　x(N/2+k)=x1(k)-WkNx2(k)　k=0,1,…,N/2-1　　（5）

从式（4）和式（5）可以看出，只要求出 0~N/2-1 区间 x1(k)和 x2(k)的值，就可求出 0~N-1区间 x(k)的 N 点值。

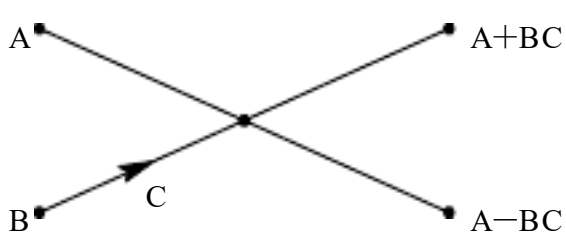
以同样的方式进行抽取，可以求得 N/4 点的 DFT，重复抽取过程，就可以使 N点的 DFT用上组 2 点的 DFT 来计算，这样就可以大减少运算量。

基 2 DIF FFT的蝶形运算如图(a)所示。设蝶形输入为 X1(K)和 X2((K)，输出为 x(k)和 x(N/2+K)，则有

$$x(k)=x1(k)+WkNx2(k) \qquad (6)$$

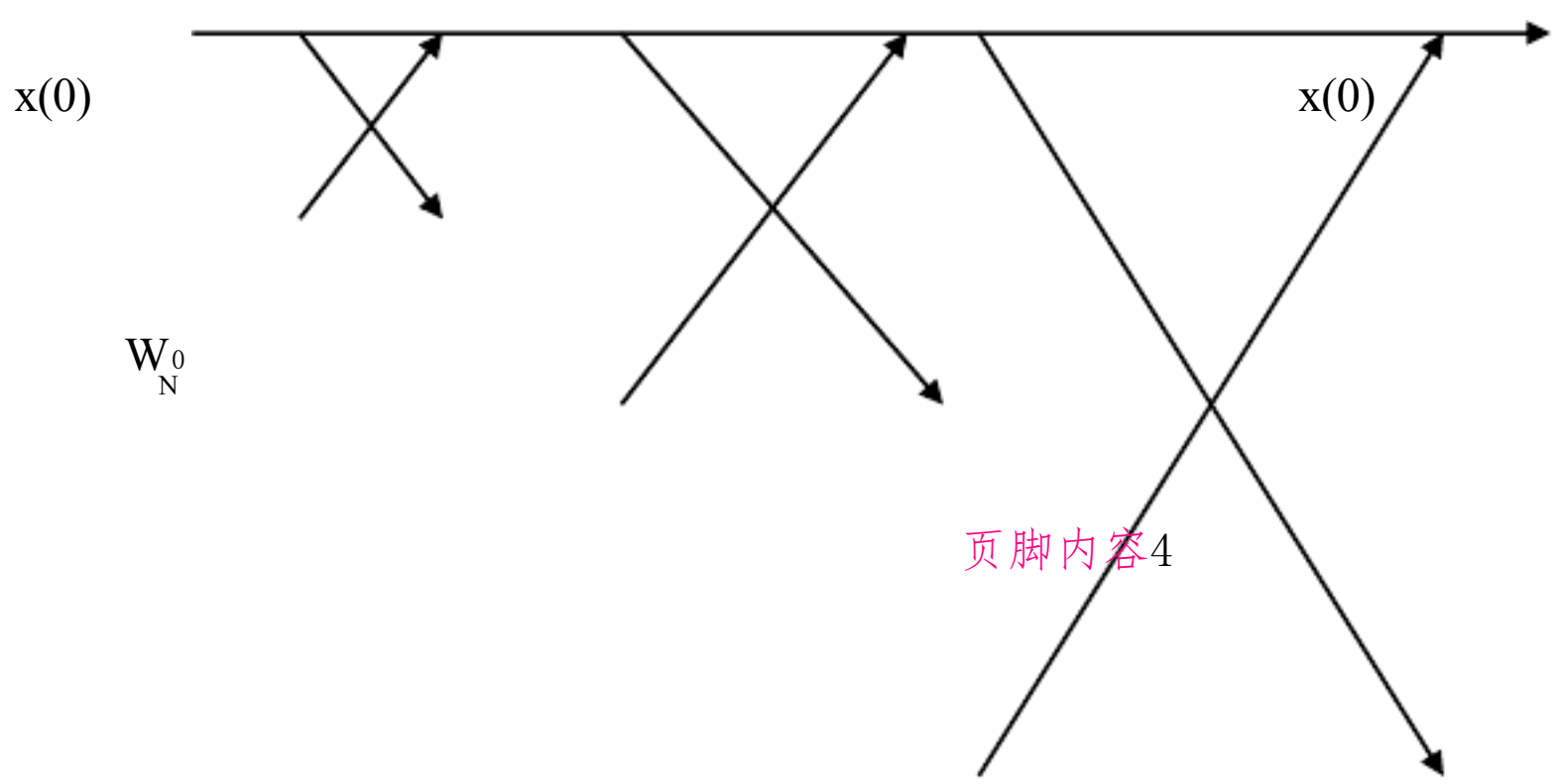
$$x(N/2+k)=x1(k)-WkNx2(k) \qquad (7)$$

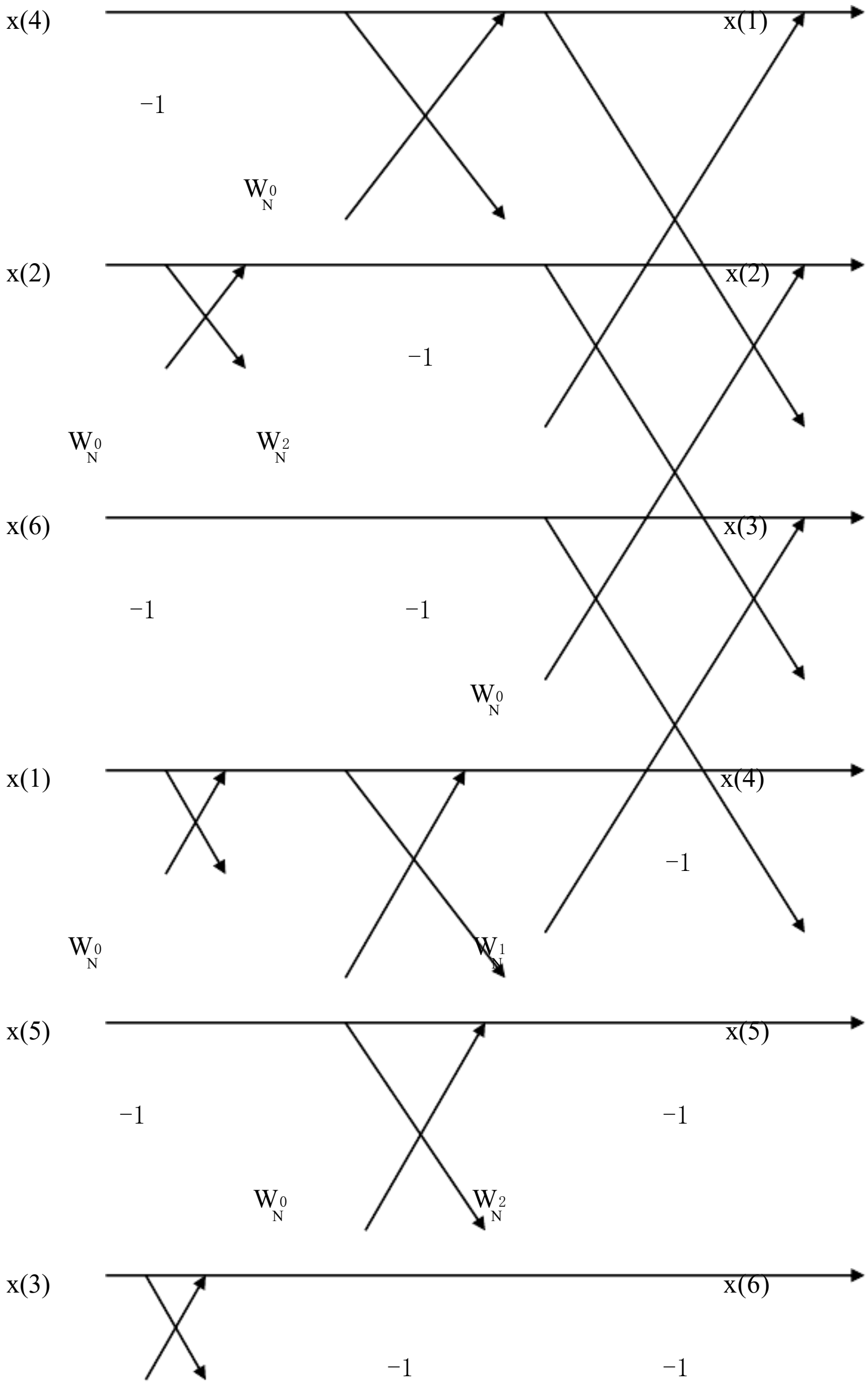
在基数为 2 的 FFT中，设 N=2M，共有 M级运算，每级有 N/2 个 2 点 FFT蝶形运算，因此，N点 FFT总共有 MN/2个蝶形运算。

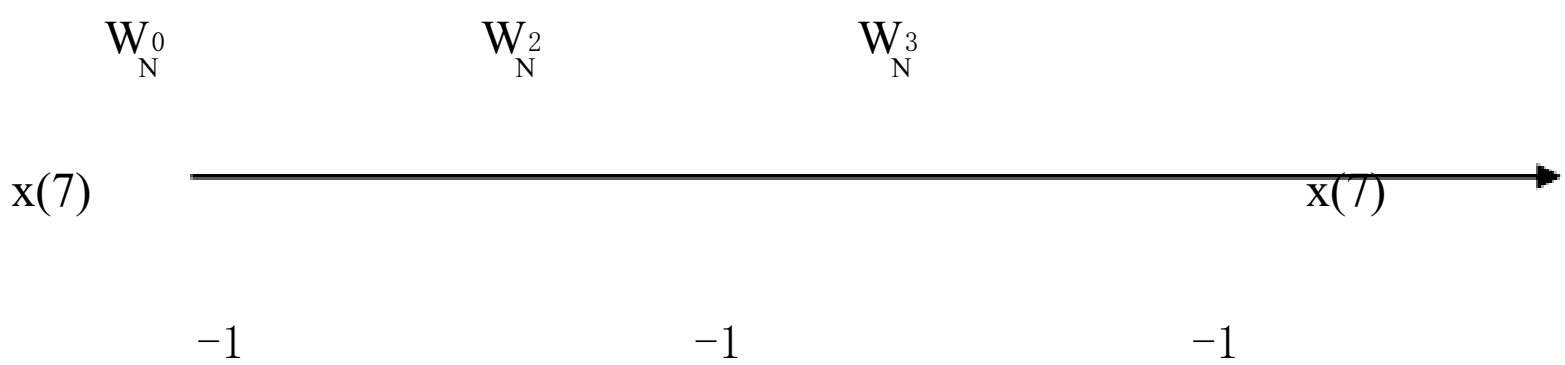


图(a) 基 2 DIF FFT的蝶形运算

例如：基数为 2 的 FFT，当 N=8时，共需要 3 级，12个基 2 DIT FFT的蝶形运算。其信号流程如图(b)所示。




x(4)
x(1)
−1
W_N^0
x(2)
x(2)
−1
W_N^0
W_N^2
x(6)
x(3)
−1
−1
W_N^0
x(1)
x(4)
−1
W_N^0
W_N^1
x(5)
x(5)
−1
−1
W_N^0
W_N^2
x(3)
x(6)
−1
−1

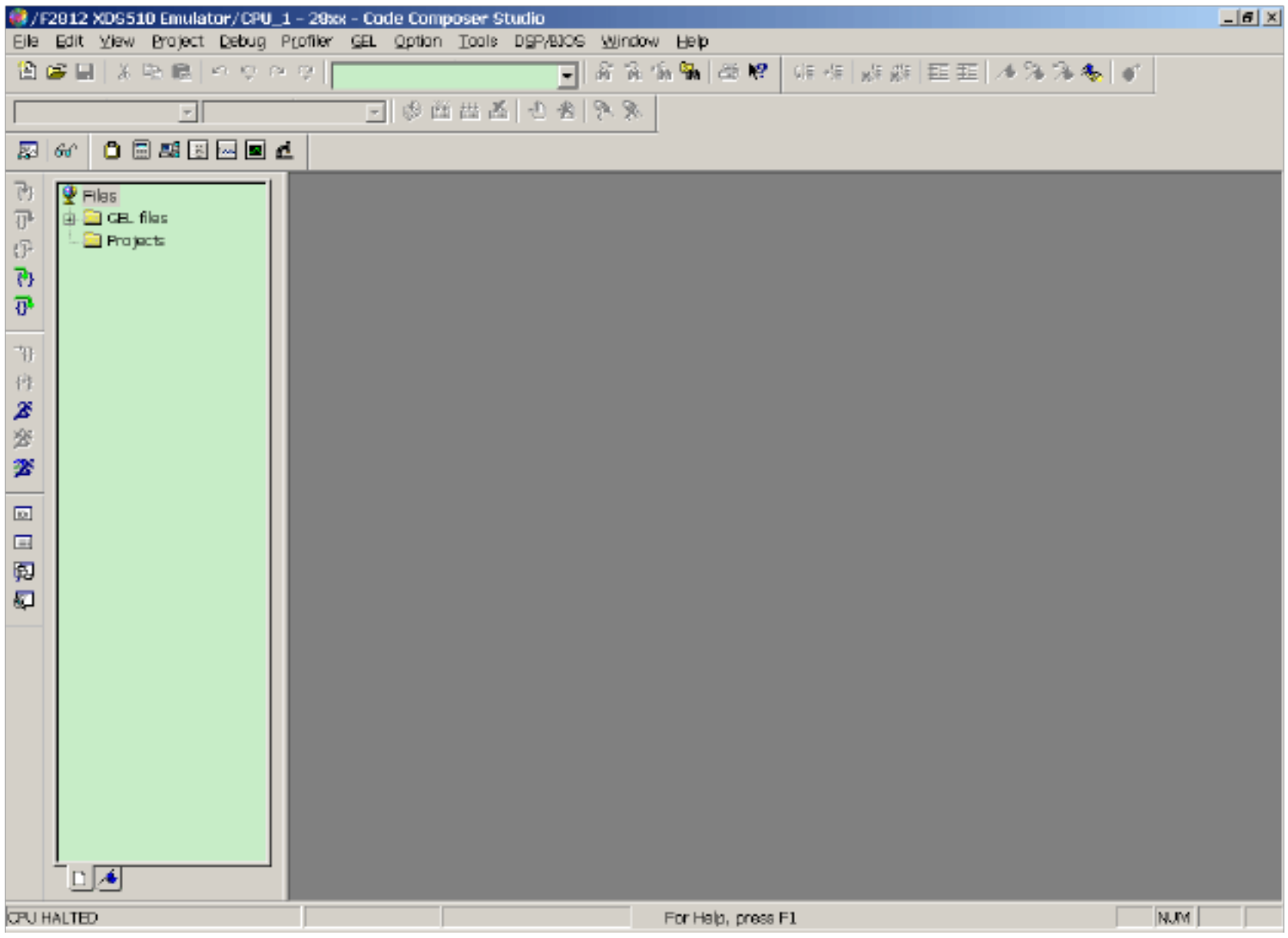


图(b) 8 点基 2 DIF FFT 蝶形运算

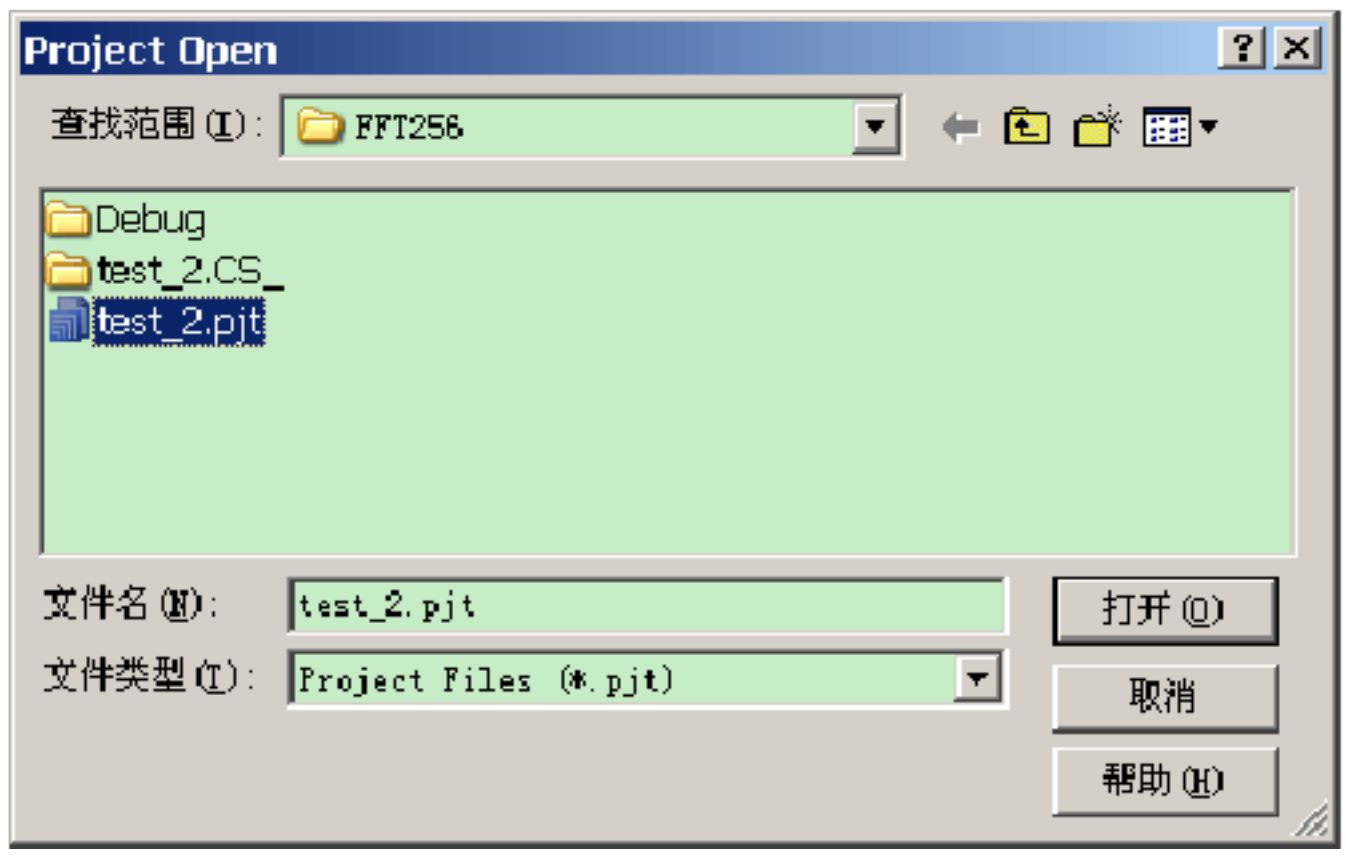
从图(b)可以看出，输入是经过比特反转的倒位序列，称为位码倒置，其排列顺序为 x(0),x(4),x(2),x(6),x(1),x(5),x(3),x(7)，输出是按自然顺序排列，其顺序为 x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7)

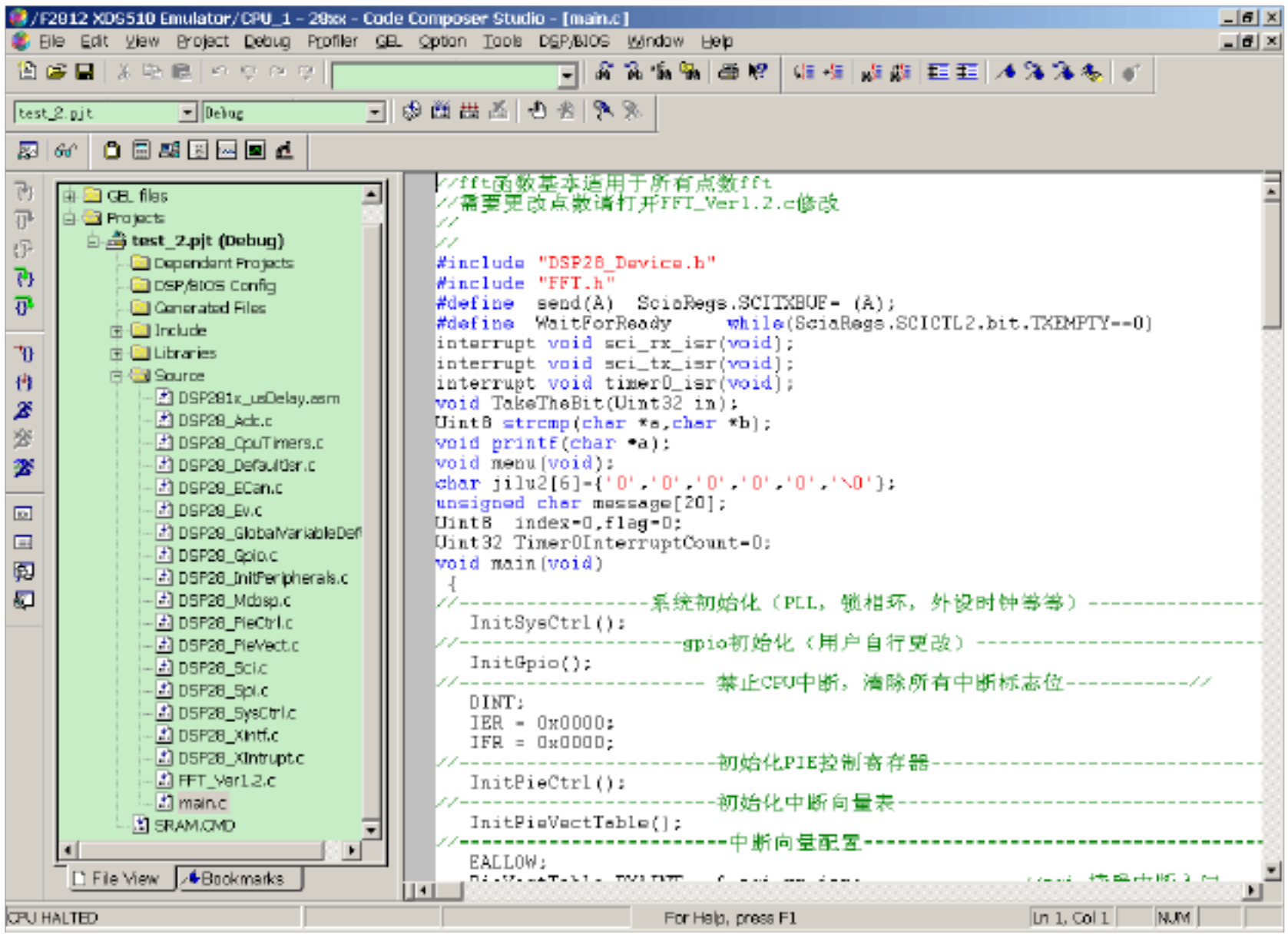
四、设计步骤

1、 启动 CSS

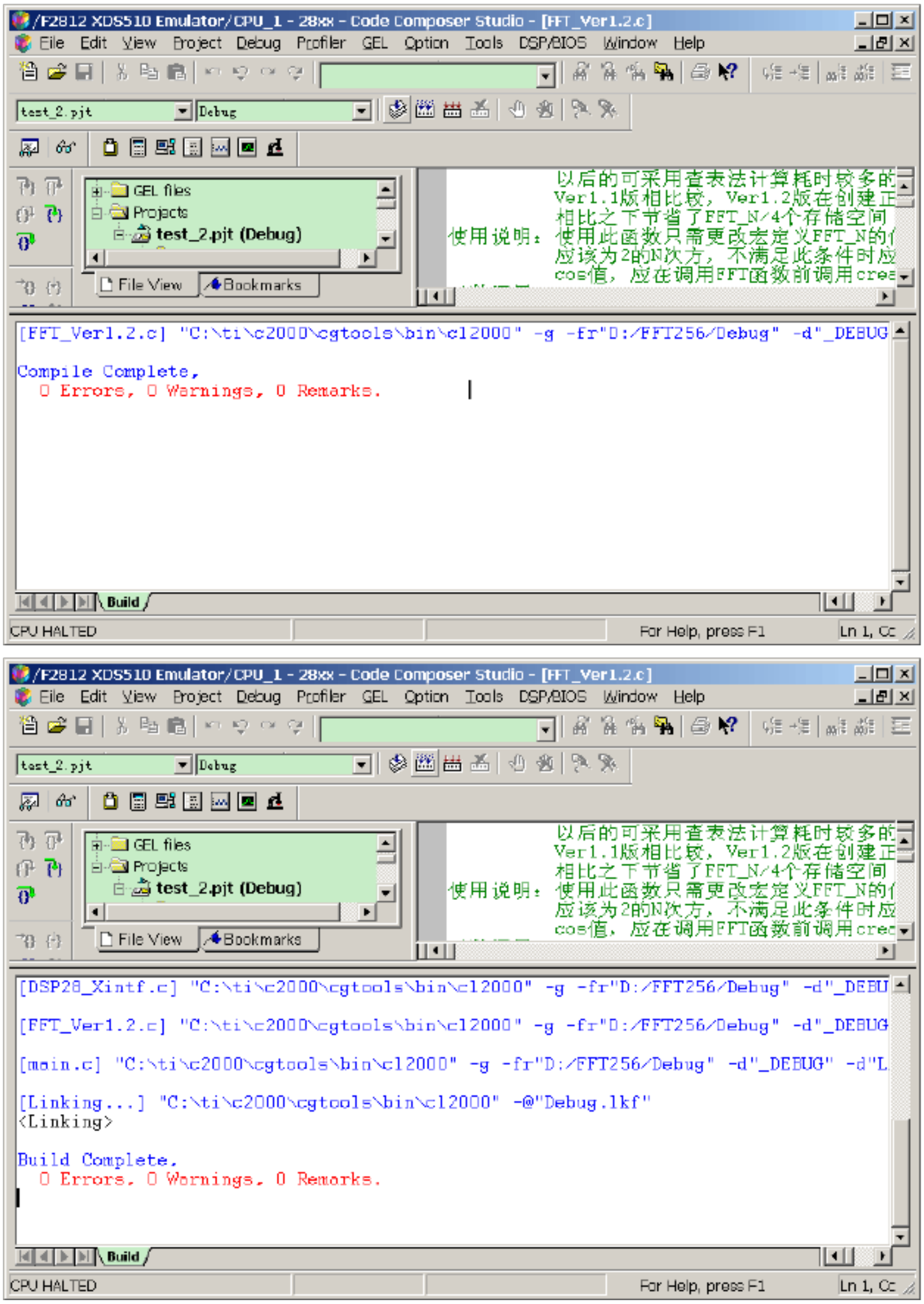

/F2812 XDS510 Emulator/CPU_1 - 28xx - Code Composer Studio
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
Files
GEL files
Projects
CPU HALTED
For Help, press F1
NUM

2、 加载工程项目。

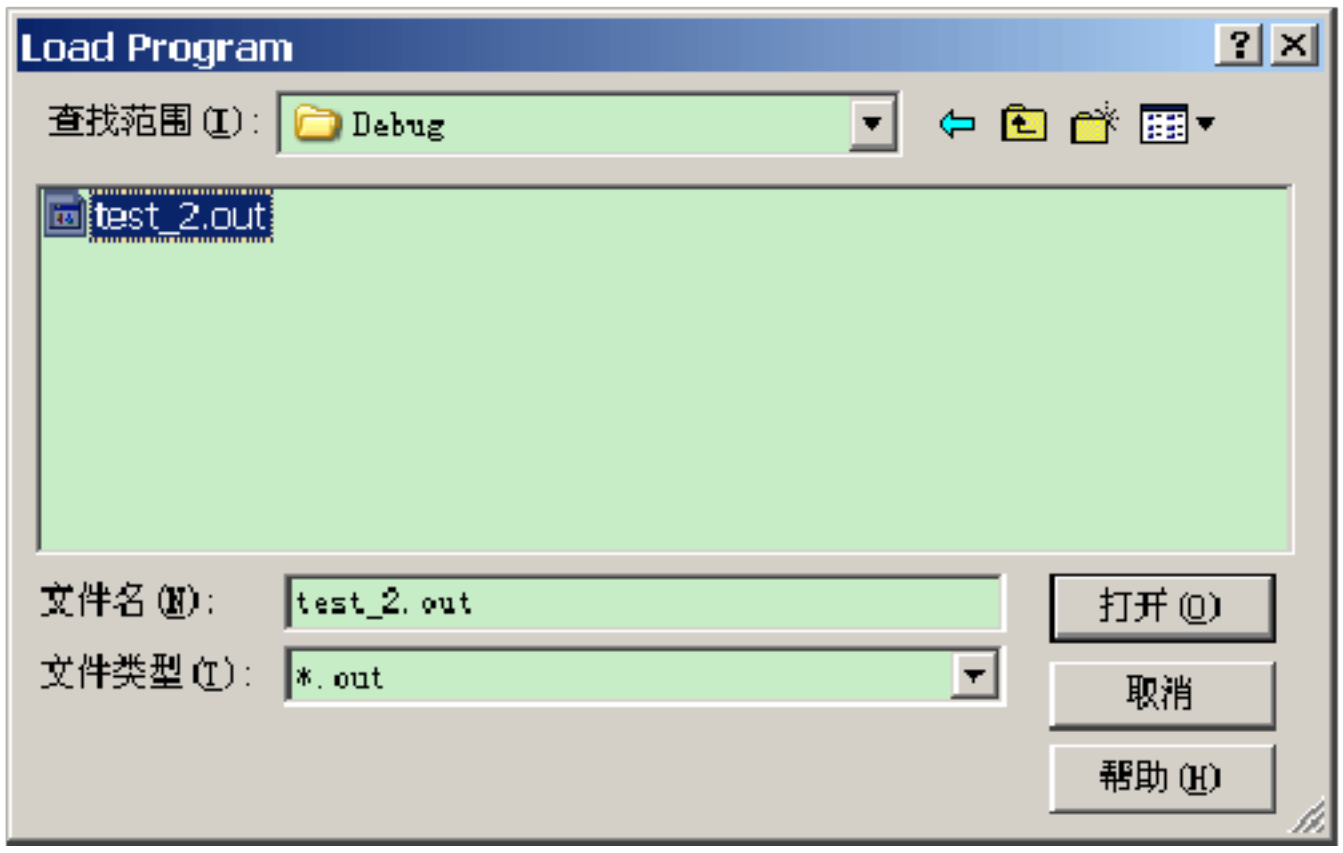

Project Open
查找范围(I):
FFT256
Debug
test_2.CS_
test_2.pjt
文件名(N):
test_2.pjt
文件类型(T):
Project Files (*.pjt)
打开(O)
取消
帮助(H)

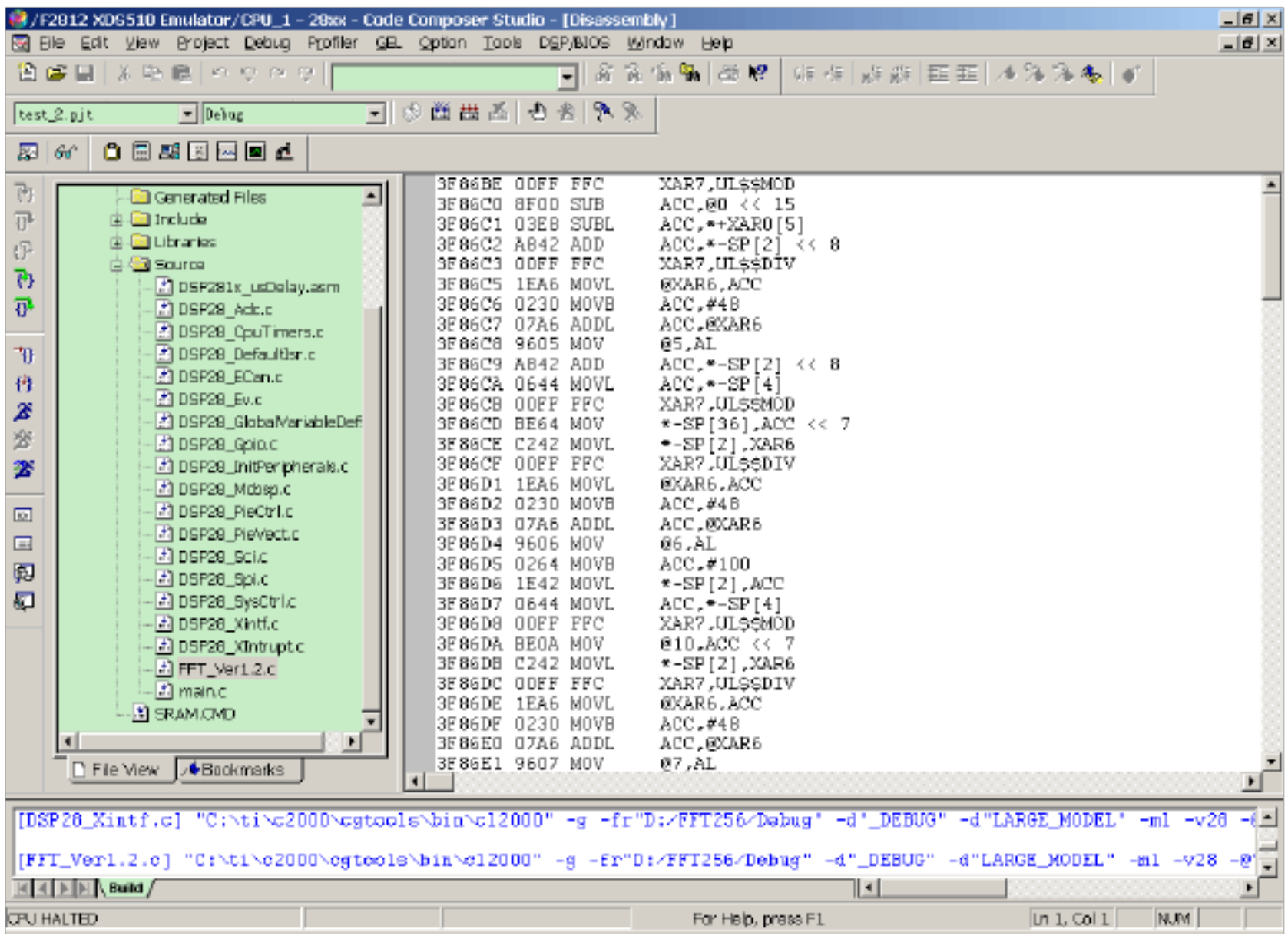

/F2812 XDS510 Emulator/CPU_1 - 28xx - Code Composer Studio - [main.c]
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
test_2.pjt
Debug
GEL files
Projects
test_2.pjt (Debug)
Dependent Projects
DSP/BIOS Config
Generated Files
Include
Libraries
Source
DSP281x_usDelay.asm
DSP28_Adc.c
DSP28_CpuTimers.c
DSP28_DefaultIsr.c
DSP28_ECan.c
DSP28_Ev.c
DSP28_GlobalVariableDef
DSP28_Gpio.c
DSP28_InitPeripherals.c
DSP28_Mdsp.c
DSP28_PieCtrl.c
DSP28_PieVect.c
DSP28_Sci.c
DSP28_Spi.c
DSP28_SysCtrl.c
DSP28_Xintf.c
DSP28_Xintrupt.c
FFT_Ver1.2.c
main.c
SRAM.CMD
File View
Bookmarks
//fft函数基本适用于所有点数fft
//需要更改点数请打开FFT_Ver1.2.c修改
#include "DSP28_Device.h"
#include "FFT.h"
#define send(A) SciaRegs.SCITXBUF= (A);
#define WaitForReady while(SciaRegs.SCICTL2.bit.TXEMPTY==0)
interrupt void sci_rx_isr(void);
interrupt void sci_tx_isr(void);
interrupt void timer0_isr(void);
void TakeTheBit(Uint32 in);
Uint8 strcmp(char *a,char *b);
void printf(char *a);
void menu(void);
char jilu2[6]={'0','0','0','0','0','\0'};
unsigned char message[20];
Uint8 index=0,flag=0;
Uint32 Timer0InterruptCount=0;
void main(void)
{
//----------------系统初始化（PLL，锁相环，外设时钟等等）----------------
InitSysCtrl();
//--------------------gpio初始化（用户自行更改）-----------------------
InitGpio();
//---------------------- 禁止CPU中断，清除所有中断标志位-----------//
DINT;
IER = 0x0000;
IFR = 0x0000;
//-----------------------初始化PIE控制寄存器-----------------------------
InitPieCtrl();
//-----------------------初始化中断向量表-------------------------------
InitPieVectTable();
//-----------------------中断向量配置-----------------------------------
EALLOW;
CPU HALTED
For Help, press F1
Ln 1, Col 1
NUM

3、 编译。


/F2812 XDS510 Emulator/CPU_1 - 28xx - Code Composer Studio - [FFT_Ver1.2.c]
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
test_2.pjt
Debug
GEL files
Projects
test_2.pjt (Debug)
File View
Bookmarks
使用说明：
以后的可采用查表法计算耗时较多的
Ver1.1版相比较，Ver1.2版在创建正
相比之下节省了FFT_N/4个存储空间
使用此函数只需更改宏定义FFT_N的
应该为2的N次方，不满足此条件时应
cos值，应在调用FFT函数前调用cre
[FFT_Ver1.2.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBUG
Compile Complete,
0 Errors, 0 Warnings, 0 Remarks.
Build
CPU HALTED
For Help, press F1
Ln 1, Cc
[DSP28_Xintf.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBU
[FFT_Ver1.2.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBUG
[main.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBUG" -d"L
[Linking...] "C:\ti\c2000\cgtools\bin\cl2000" -@"Debug.lkf"
<Linking>
Build Complete,
0 Errors, 0 Warnings, 0 Remarks.

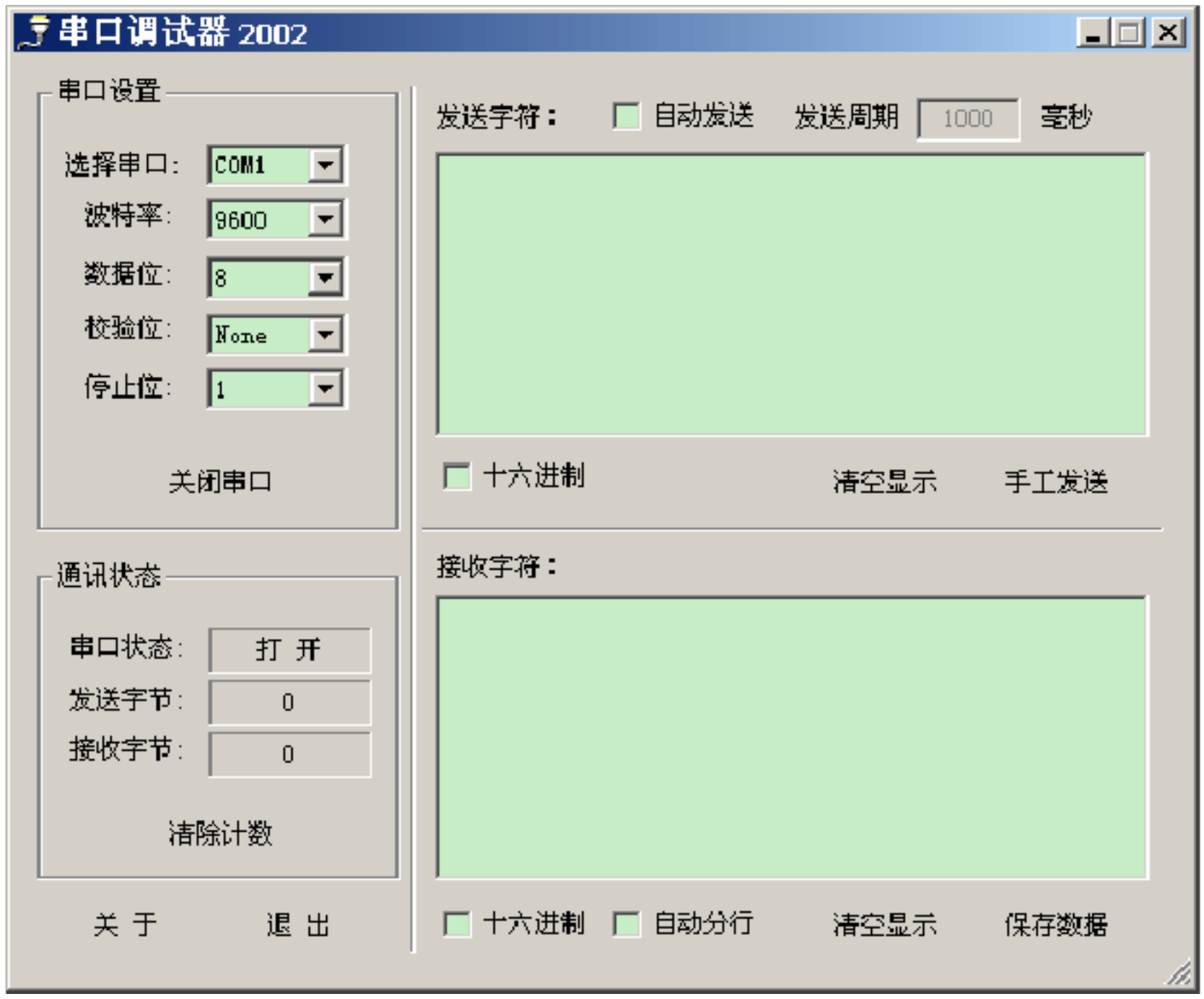
4、 加载.out 文件


Load Program
查找范围(I):
Debug
test_2.out
文件名(N):
test_2.out
文件类型(T):
*.out
打开(O)
取消
帮助(H)


/F2812 XDS510 Emulator/CPU_1 - 28xx - Code Composer Studio - [Disassembly]
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
test_2.pjt
Debug
Generated Files
Include
Libraries
Source
DSP281x_usDelay.asm
DSP28_Adc.c
DSP28_CpuTimers.c
DSP28_DefaultIsr.c
DSP28_ECan.c
DSP28_Ev.c
DSP28_GlobalVariableDef
DSP28_Gpio.c
DSP28_InitPeripherals.c
DSP28_Mcbsp.c
DSP28_PieCtrl.c
DSP28_PieVect.c
DSP28_Sci.c
DSP28_Spi.c
DSP28_SysCtrl.c
DSP28_Xintf.c
DSP28_XIntrupt.c
FFT_Ver1.2.c
main.c
SRAM.CMD
File View
Bookmarks
3F86BE 00FF FFC XAR7,UL$$MOD
3F86C0 8F00 SUB ACC,@0 << 15
3F86C1 03E8 SUBL ACC,*+XAR0[5]
3F86C2 AB42 ADD ACC,*-SP[2] << 8
3F86C3 00FF FFC XAR7,UL$$DIV
3F86C5 1EA6 MOVL @XAR6,ACC
3F86C6 023D MOVB ACC,#48
3F86C7 07A6 ADDL ACC,@XAR6
3F86C8 9605 MOV @5,AL
3F86C9 AB42 ADD ACC,*-SP[2] << 8
3F86CA 0644 MOVL ACC,*-SP[4]
3F86CB 00FF FFC XAR7,UL$$MOD
3F86CD BE64 MOV *-SP[36],ACC << 7
3F86CE C242 MOVL *-SP[2],XAR6
3F86CF 00FF FFC XAR7,UL$$DIV
3F86D1 1EA6 MOVL @XAR6,ACC
3F86D2 023D MOVB ACC,#48
3F86D3 07A6 ADDL ACC,@XAR6
3F86D4 9606 MOV @6,AL
3F86D5 0264 MOVB ACC,#100
3F86D6 1E42 MOVL *-SP[2],ACC
3F86D7 0644 MOVL ACC,*-SP[4]
3F86D8 00FF FFC XAR7,UL$$MOD
3F86DA BE0A MOV @10,ACC << 7
3F86DB C242 MOVL *-SP[2],XAR6
3F86DC 00FF FFC XAR7,UL$$DIV
3F86DE 1EA6 MOVL @XAR6,ACC
3F86DF 023D MOVB ACC,#48
3F86E0 07A6 ADDL ACC,@XAR6
3F86E1 9607 MOV @7,AL
[DSP28_Xintf.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBUG" -d"LARGE_MODEL" -ml -v28
[FFT_Ver1.2.c] "C:\ti\c2000\cgtools\bin\cl2000" -g -fr"D:/FFT256/Debug" -d"_DEBUG" -d"LARGE_MODEL" -ml -v28
Build
CPU HALTED
For Help, press F1
Ln 1, Col 1
NUM

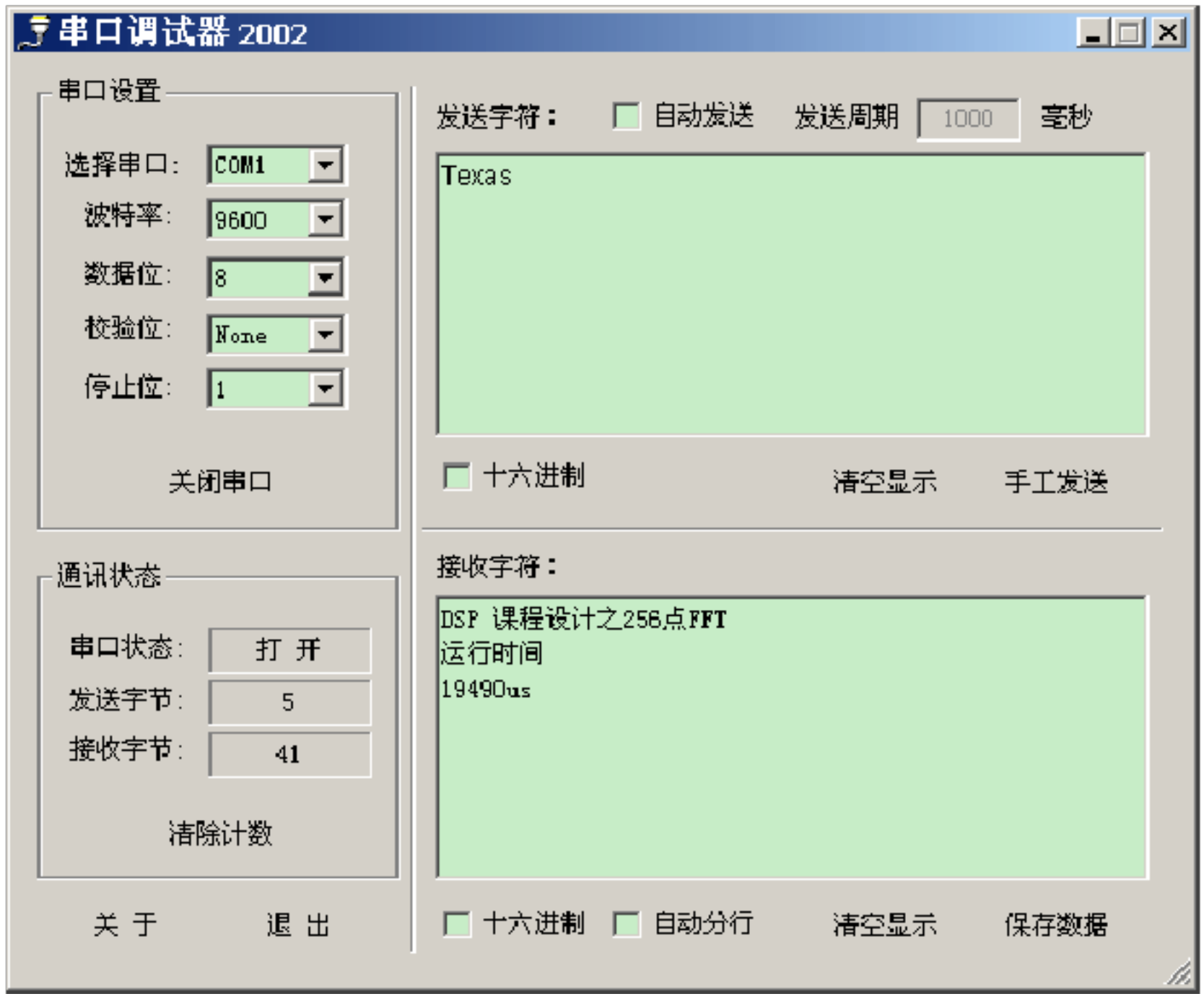
5、 打开串口调试。


串口调试器 2002
串口设置
选择串口:
COM1
波特率:
38400
数据位:
8
校验位:
None
停止位:
1
打开串口
通讯状态
串口状态:
关 闭
发送字节:
0
接收字节:
0
清除计数
关 于
退 出
发送字符:
自动发送
发送周期
1000
毫秒
十六进制
清空显示
手工发送
接收字符:
十六进制
自动分行
清空显示
保存数据


串口调试器 2002
串口设置
选择串口:
COM1
波特率:
9600
数据位:
8
校验位:
None
停止位:
1
关闭串口
通讯状态
串口状态:
打 开
发送字节:
0
接收字节:
0
清除计数
关 于
退 出
发送字符:
自动发送
发送周期
1000
毫秒
十六进制
清空显示
手工发送
接收字符:
十六进制
自动分行
清空显示
保存数据

6、 运行。

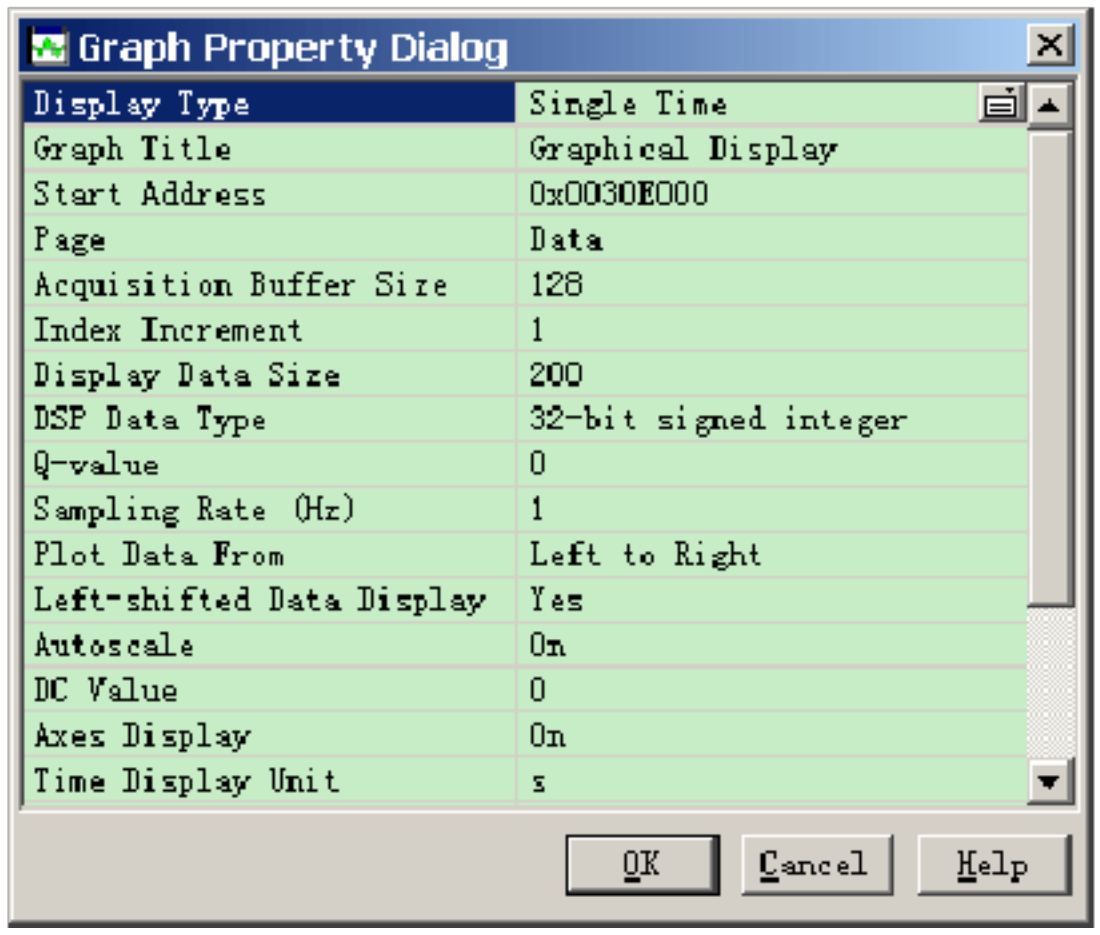
7、 输入 Texas点手工发送。



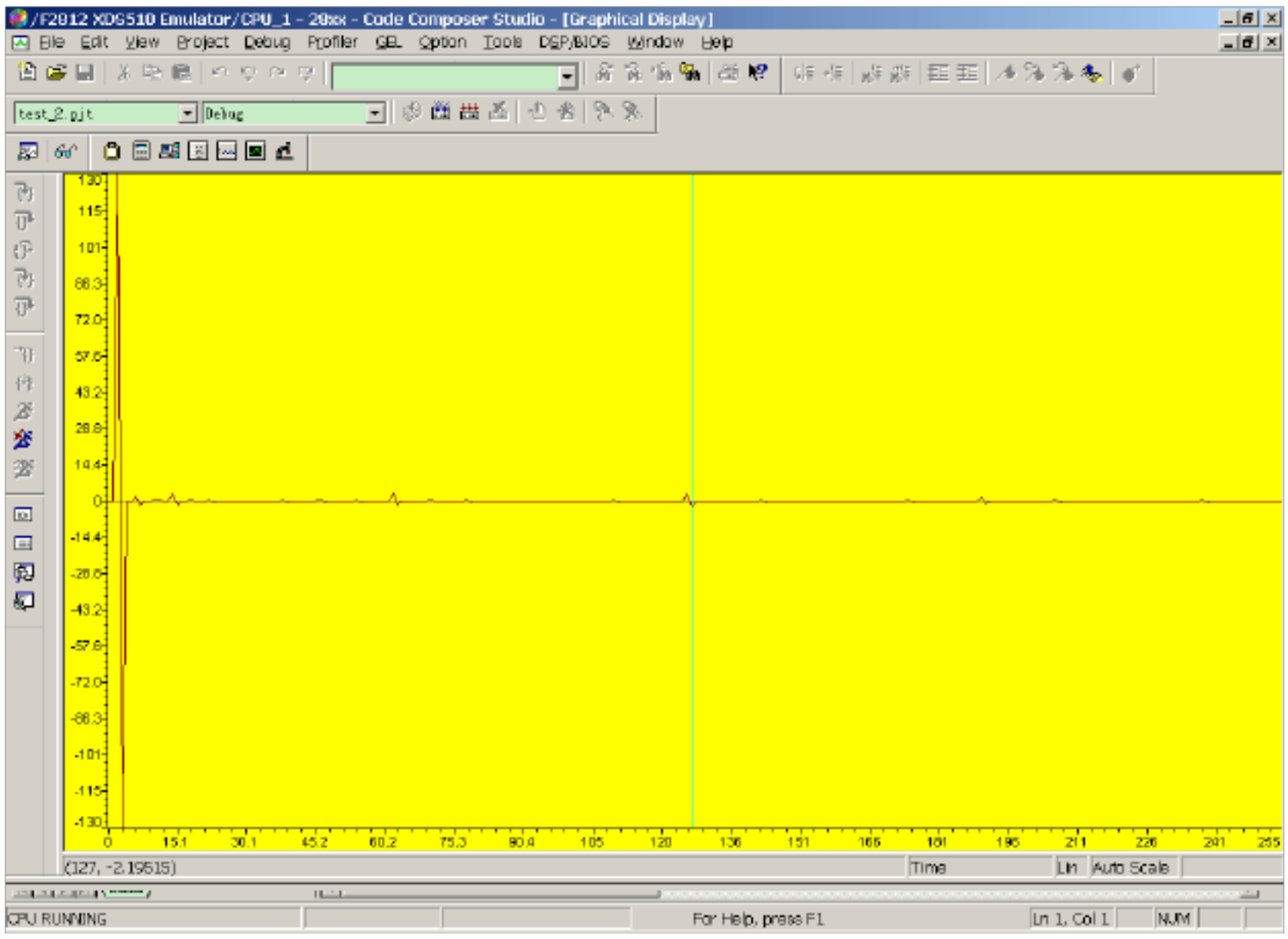
8、 观察 s 的频谱。

s[i].real=sin(2*3.1415926535897*i/FFT_N); 实部为正弦波 FFT_N 点采样，赋值为 1

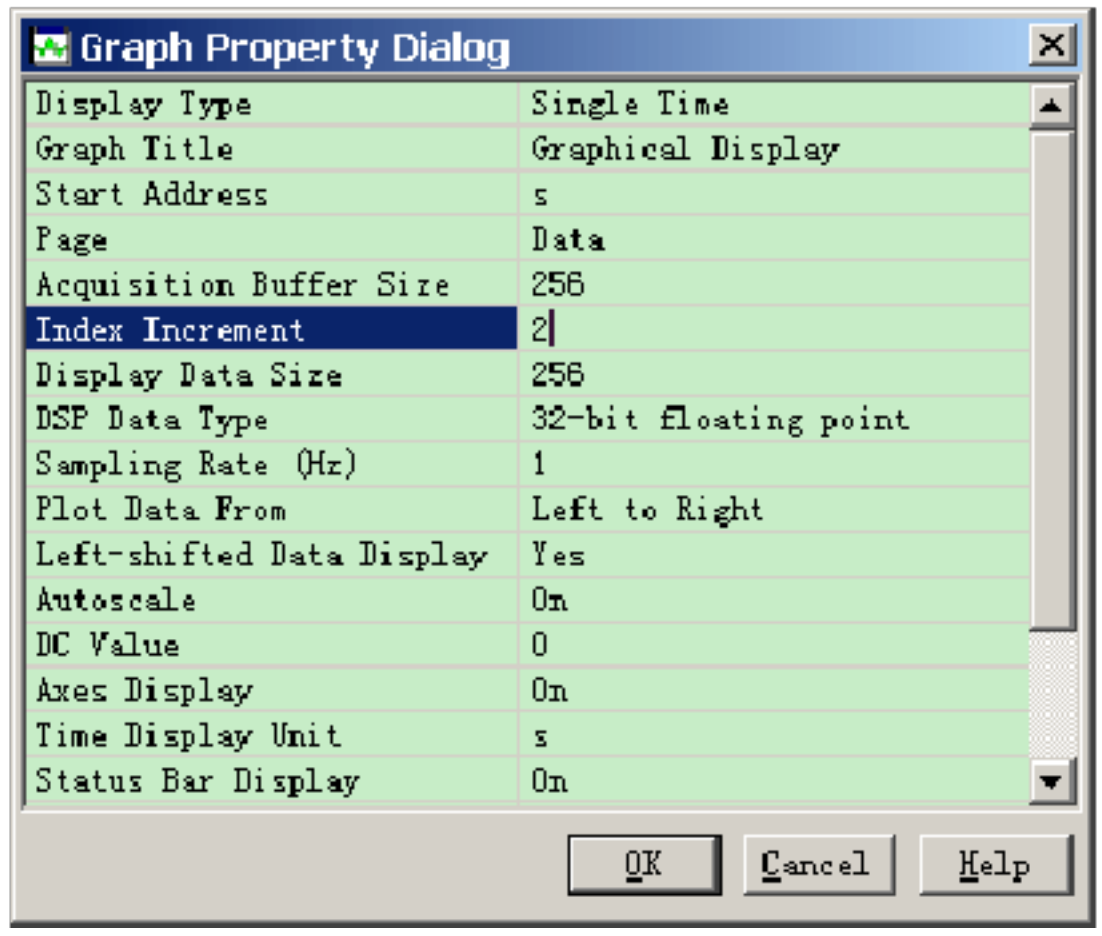
s[i].imag=0; //虚部为 0

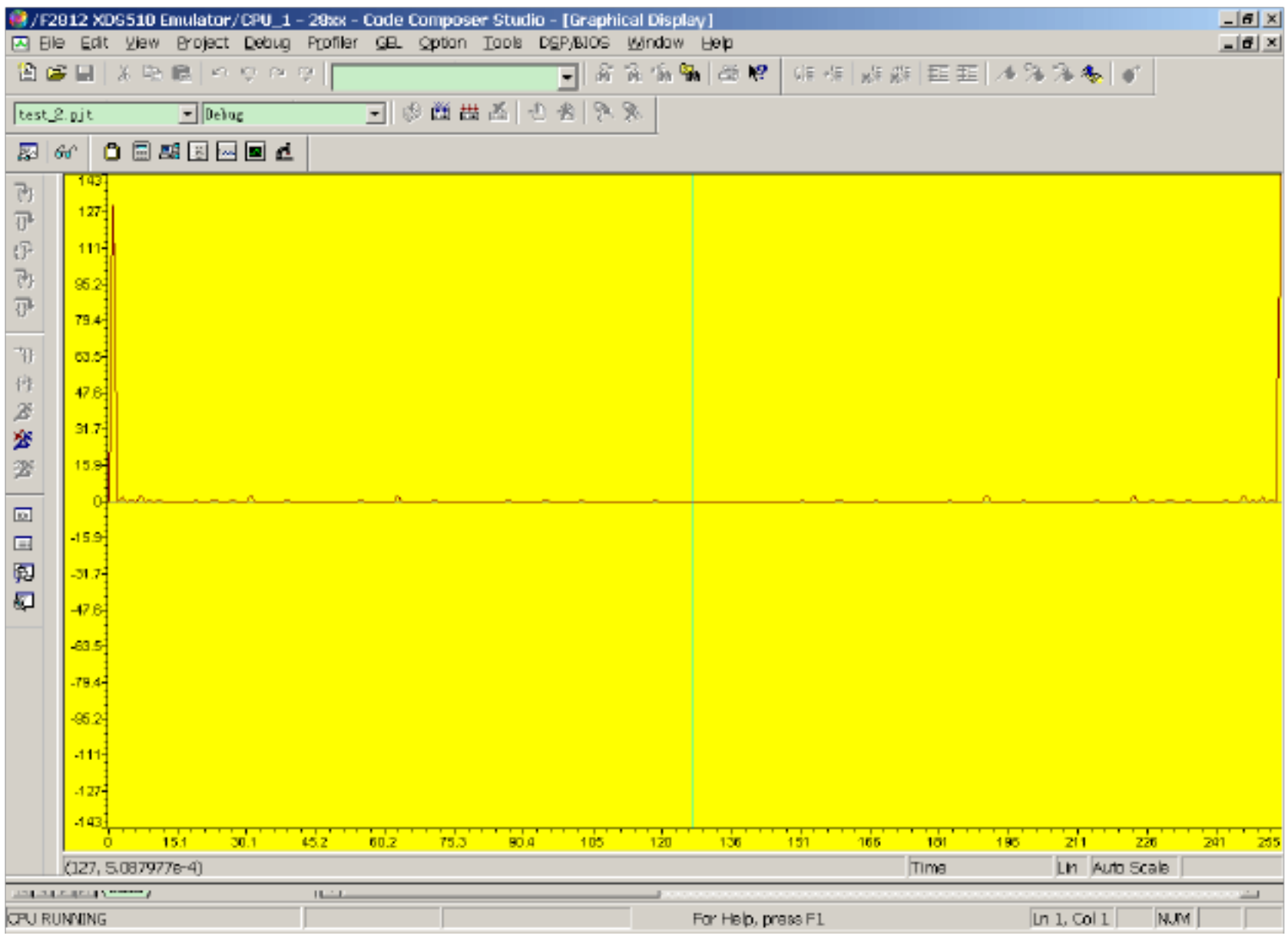


正弦波形图，由于定义了虚部，0的下方有波形。

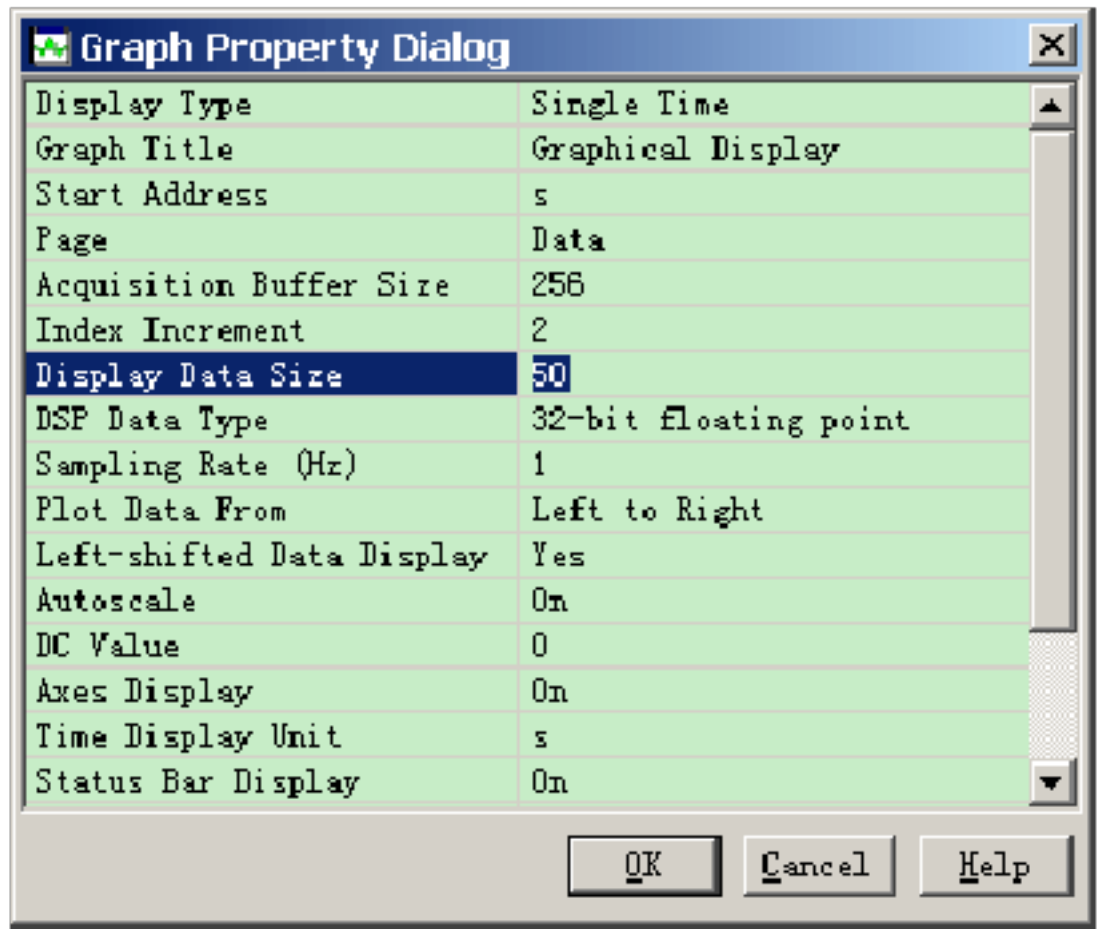


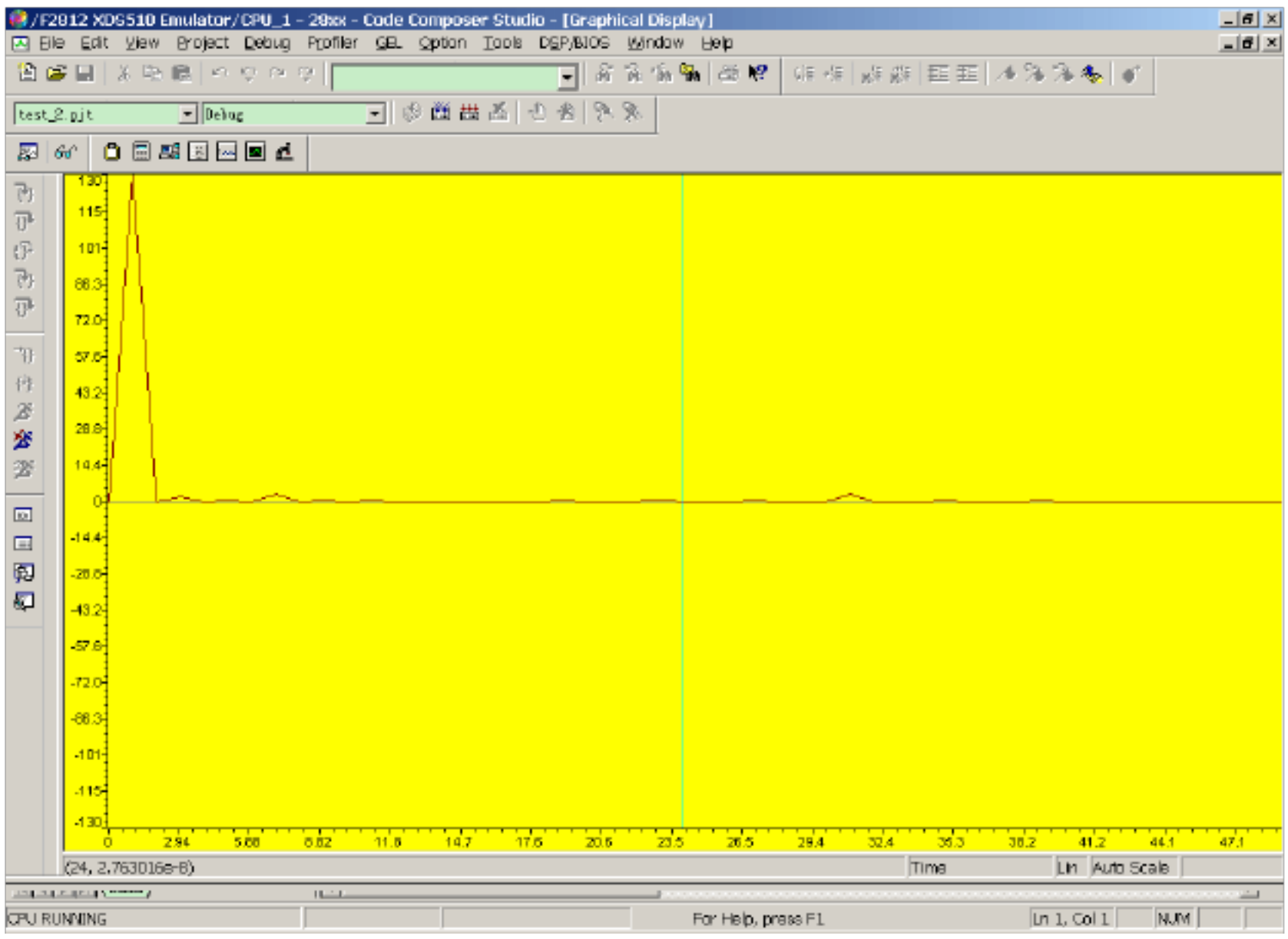
9、 调整，只显示实部。


Graph Property Dialog
Display Type
Single Time
Graph Title
Graphical Display
Start Address
s
Page
Data
Acquisition Buffer Size
256
Index Increment
2
Display Data Size
256
DSP Data Type
32-bit floating point
Sampling Rate (Hz)
1
Plot Data From
Left to Right
Left-shifted Data Display
Yes
Autoscale
On
DC Value
0
Axes Display
On
Time Display Unit
s
Status Bar Display
On
OK
Cancel
Help


/F2812 XDS510 Emulator/CPU_1 - 28xx - Code Composer Studio - [Graphical Display]
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
test_2.pjt
Debug
143
127
111
95.2
79.4
63.5
47.6
31.7
15.9
0
-15.9
-31.7
-47.6
-63.5
-79.4
-95.2
-111
-127
-143
0 15.1 30.1 45.2 60.2 75.3 90.4 105 120 136 151 165 181 196 211 226 241 255
(127, 5.087977e-4)
Time
Lin
Auto Scale
CPU RUNNING
For Help, press F1
Ln 1, Col 1
NUM

10、调整显示大小，显示更好。





五、 程序主要代码

1、main( )函数

```
#define   send(A) SciaRegs.SCITXBUF= (A);
      #define   WaitForReady     while(SciaRegs.SCICTL2.bit.TXEMPTY==0)
interrupt void sci_rx_isr(void);
interrupt void sci_tx_isr(void);
interrupt void timer0_isr(void);
       void TakeTheBit(Uint32 in);
Uint8 strcmp(char *a,char *b);
void printf(char *a);
void menu(void);


unsigned char message[20];
Uint8  index=0,flag=0;
Uint32 Timer0InterruptCount=0;
```

```
void main(void)

  {

//-----------系统初始化（PLL，锁相环，外设时钟等等）----------------------//

    InitSysCtrl();

//-------------------gpio初始化-------------------------------//

    InitGpio();

//---------------禁止CPU中断，清除所有中断标志位-----------//

    DINT;

    IER = 0x0000;

    IFR = 0x0000;

//---------------初始化PIE控制寄存器-----------------------------------//

    InitPieCtrl();

//---------------初始化中断向量表-------------------------------------//

    InitPieVectTable();

//=========================          中断向量配置

=======================================//

    EALLOW;
```