DSP原理与技术

中期分享

李尚文　崔 敖

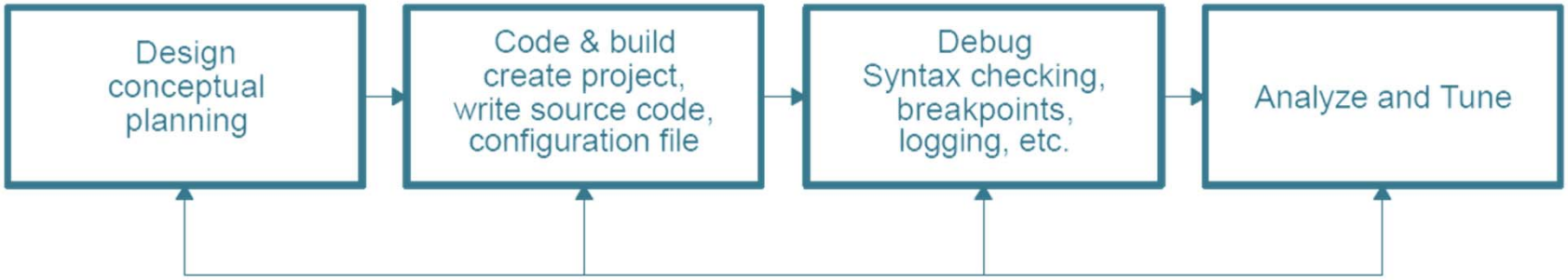
目录

第1部分

基础实验总结

1.| 基础实验总结

简化的CCS开发流程



可参考的文献资料：

[1] 教科书 《TMS320C54x DSP 结构、原理及应用》
[2] TI公司 《TMS320VC5402 Fixed Point Digital Signal Processor》
[3] TI公司 《TMS320C54x DSP Reference Set》
[4] TI公司 《Code Composer Studio Development Tools v3.3》

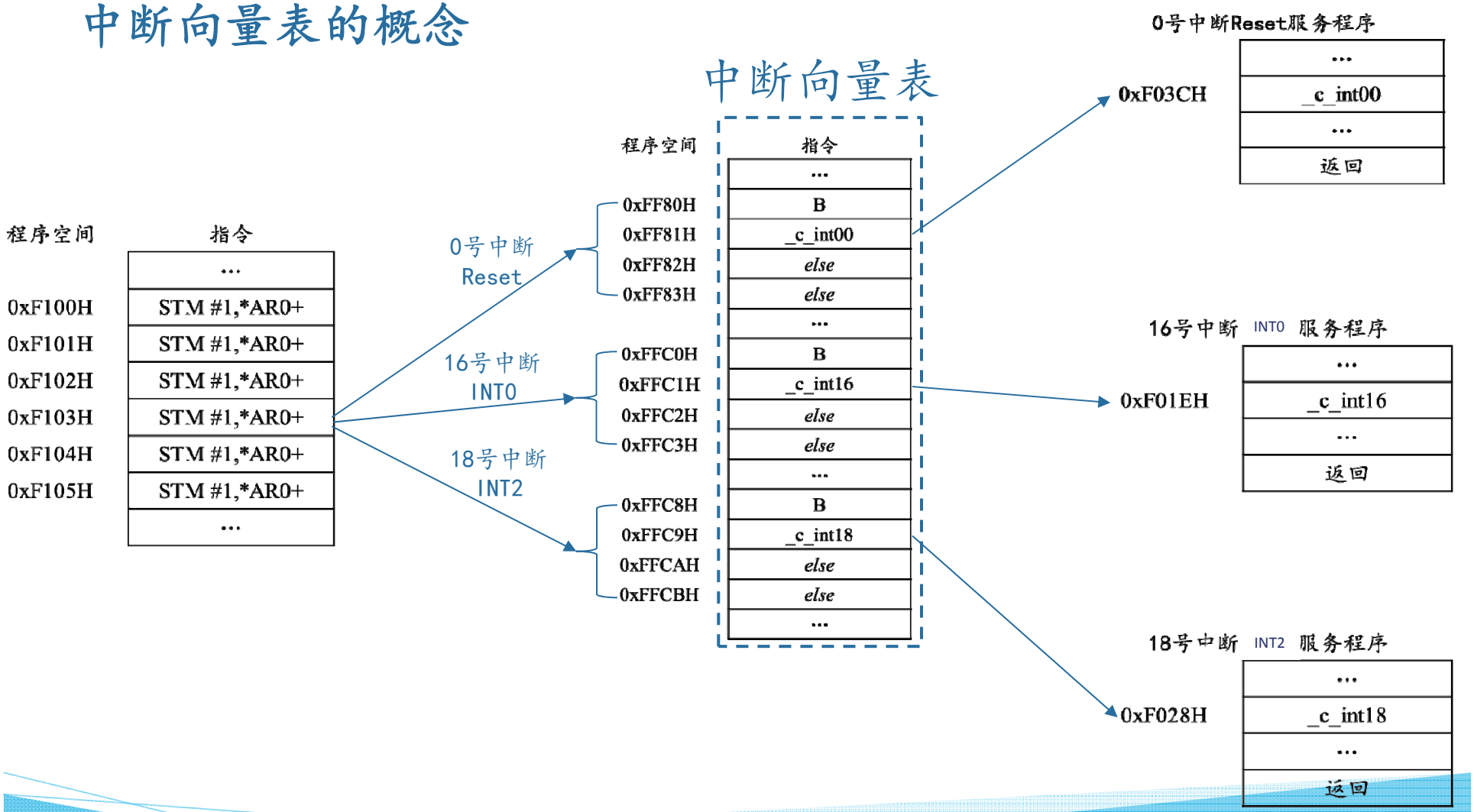
1.| 基础实验总结

数据寻址方式总结

寻址方式	用　途	举　例	指令含义
立即寻址	主要用于初始化	LD #10，A	立即数10 → A
绝对寻址	利用16位地址寻址存储单元	STL A，*（y）	将AL内容存入y所在的存储单元
累加器寻址	将累加器中的内容作为地址	READA x	将A的内容作为地址读程序存储器，并存入x存储单元
直接寻址	利用数据页指针和堆栈指针寻址	LD @x，A	(DP+x的低7位地址) → A
间接寻址	利用辅助寄存器作为地址指针	LD *AR1，A	((AR1)) → A
存储器映像寄存器寻址	快速寻址存储器映象寄存器	LDM ST1，B	(ST1) → B
堆栈寻址	压入/弹出数据存储器和存储器映像寄存器MMR	PSHM AG	(SP)-1 → SP, (AG) → (SP)

1.| 基础实验总结

中断向量表的概念



1. 基础实验总结

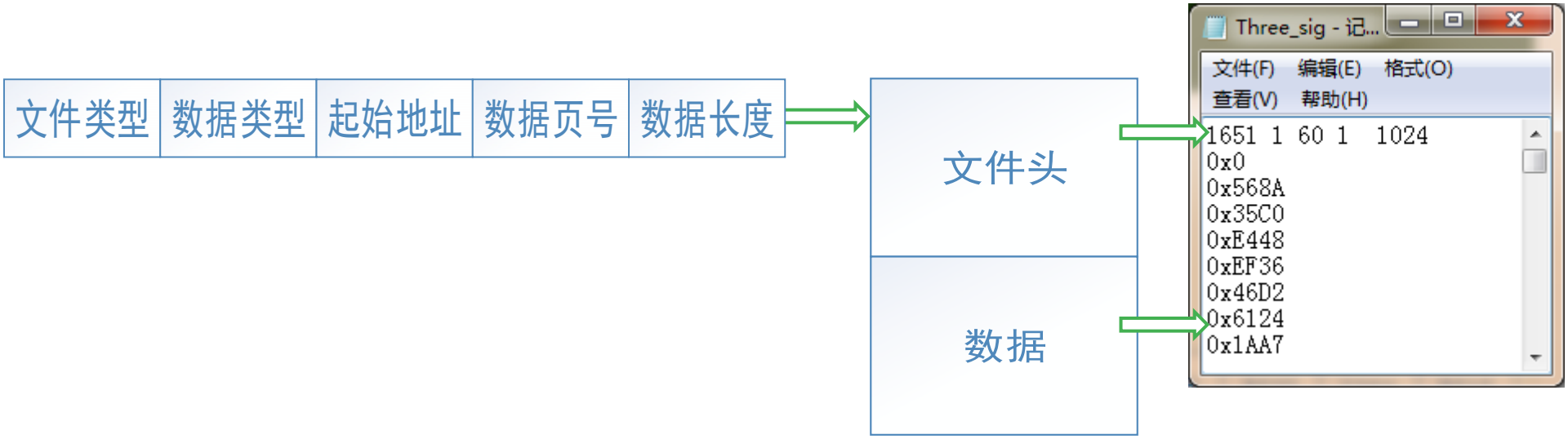
CCS中的.dat文件

CCS中的.dat文件主要有以下三种作用：

1. 利用.dat文件加载DSP特定的存储空间；
2. 利用.dat文件和Pin Connect功能实现外部中断的仿真；
3. 利用.dat文件和Port Connect功能实现外部IO接口的仿真；

1.| 基础实验总结

利用.dat文件加载DSP特定的存储空间



[文件类型] 固定为1651
[数据类型] 取值为1~4，分别对应Hex, int, long int, float
[起始地址] 存放数据内存区的首地址，十六进制数
[数据页号] 0为PM，1为DM，2为IO
[数据长度] 指明数据块长度，以字为单位，十六进制数

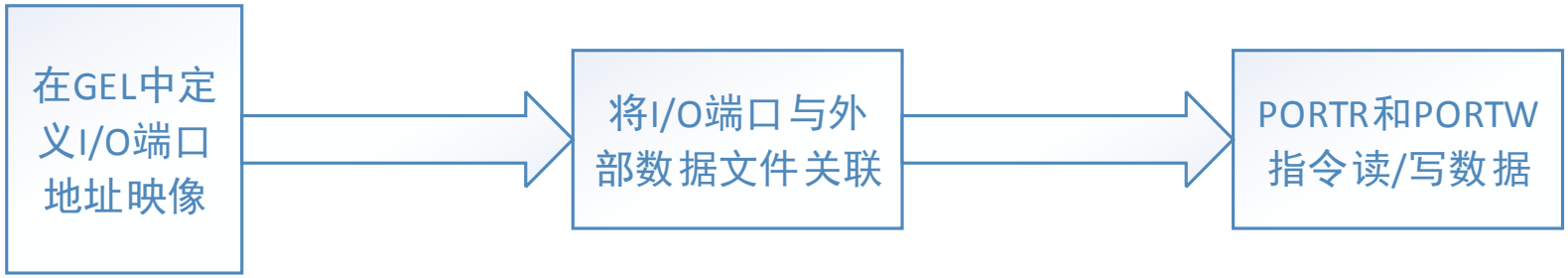
1.| 基础实验总结

利用.dat文件实现外部中断的仿真

文件内容	含义
100 120 300	分别在第100、120、300个时钟周期时产生中断；
100 +20 300	同上，分别在第100、120、300个时钟周期时产生中断；
5 (+10) rpt 3	分别在第5、15、25、35个时钟周期时产生中断；
5 (+10 +20) rpt 2	分别在第5、15、35、45、65个时钟周期时产生中断；
100 (+200) rpt EOS	分别在第100、300、500、700…个时钟周期时产生中断，无限循环。

1.| 基础实验总结

利用.dat文件实现外部IO接口的仿真



可通过在当前的GEL（通用扩展语句）文件中插入GEL函数定义I/O端口地址映像：

GEL_MapAdd(address, page, length, readable, writeable);

其中：

address　端口地址

page　存储空间值。0为程序空间，1为数据空间，2为I/O空间

length　地址空间长度。对于I/O口一般取为1

readable　可读标志。1为可读，0为不可读。

writeable　可写标志。1为可写，0为不可写。

1.| 基础实验总结

2的补码小数

16位整数相乘 → 乘积向左增长 →
- 超出器件数据范围
- 占用更多RAM
- 消耗更多机器周期
- 不利于后续递推运算

小数相乘 → 乘积向右增长 →
- 超出器件数据范围的为不敏感数据
- 占用更少RAM
- 利于后续递推运算

因此，定点DSP都采用小数乘法

1.| 基础实验总结

2的补码小数

MSB				...	LSB
-1.	½	¼	⅛	...	2^{-15}

生成方法：

~1		7FFFh
0.5	正数：乘以32768	4000h
0		0000h
-0.5	负数：绝对值乘以32768	C000h
-1	再取反加1	8000h

1.| 基础实验总结

这种带符号的小数在进行运算时，默认情况下DSP对小数的计算会产生冗余符号位，影响计算。解决冗余符号位的办法是：设定ST1中的FRCT（小数方式位）为1，在乘法器结果传送至累加器时就能自动地左移1位，自动地消去了两个相乘时产生的冗余符号位。

因此，在小数乘法编程时，应当事先设置FRCT位：

```
SSBX    FRCT
……
MPY     *AR2,*AR3,A
STH     A,@Z
```

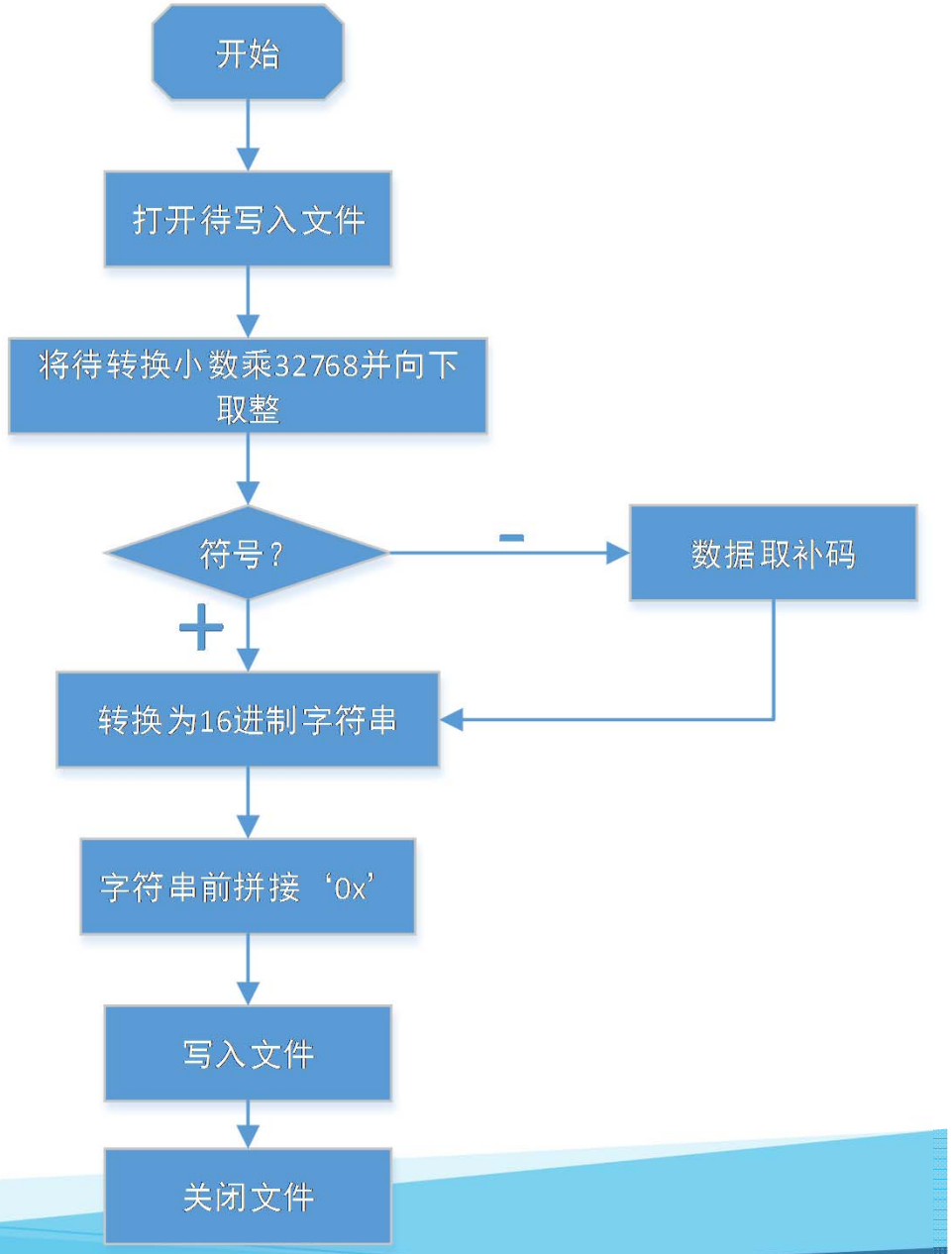
这样C54x就完成了Q15*Q15=Q15的运算

1.| 基础实验总结

生成16进制表示的2的补码小数算法流程：

可用到的Matlab函数：

floor	dec2hex
fopen	fprintf
fclose	等等…



1.| 基础实验总结

一般情况下，为了提高运算效率和方便起见，也可使用Matlab自带库中的量化器工具将数组转换为16进制数据。

量化器对象：

Quant=quantizer(mode, roundmode, overflowmode, format);

Mode: 设置量化器数据类型

Roundmode: 设置量化器输入数据的取整类型

Overflowmode: 设置量化器的溢出模式

Format: 设置量化器输出数据格式

生成2的补码小数量化器设置：

Quant=quantizer('fixed', 'floor', 'saturate', [16,15]);

HexNumber=num2hex(Quant, NormalNumber);

NormalNumber=hex2num(Quant, HexNumber);

1.| 基础实验总结

系数对称FIR滤波器的实现方法

系数对称FIR滤波器具有线性相位特性，因此应用较广。

一个N=8的FIR滤波器，若$a(n)=a(N-1-n)$,就是对称FIR滤波器，其输出方程为：

$$y(n)=a_0x(n)+a_1x(n-1)+a_2x(n-2)+a_3x(n-3)+$$
$$a_4x(n-4)+a_5x(n-5)+a_6x(n-6)+a_7x(n-7)$$

总共有8次乘法和7次加法。如果改写成：

$$y(n)=a_0\left[x(n)+x(n-7)\right]+a_1\left[x(n-1)+x(n-6)\right]+$$
$$a_2\left[x(n-2)+x(n-5)\right]+a_3\left[x(n-3)+x(n-4)\right]$$

则变成4次乘法和7次加法。可见乘法运算量较少了一半。这是对称FIR滤波器的一个优点。

1.| 基础实验总结

对称FIR滤波器C54x实现的要点如下：

1）在数据存储器中开辟两个循环缓冲区：New循环缓冲区中存放新数据；Old循环缓冲区中存放老数据。循环缓冲区长度均为N/2；（此处为举例方便，N取4）

2）初始化循环缓冲区指针分别指向New中第一个，Old中最后一个；

3）在程序存储器中设置系数表；

4）输入新数据，放入AR2所指向的存储器单元；

5）累加器清零，计算当前输出，每次相乘累加AR2与AR3循环减1；

6）修正AR2，指向New中最老数据，并将其所指内容加载至AR3所指；

7）修正AR3，保证其指向Old中最老数据；

8）Goto 4）

1.| 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=0

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=1　　[illegible] AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0000	← AR2
0x0081h	0x0001	← AR2
0x0082h	0	← AR2
0x0083h	0	← AR2

Addr.	Old循环缓冲区	
0x0088h	0	← AR3
0x0089h	0	← AR3
0x008Ah	0	← AR3
0x008Bh	0	← AR3

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=2 [illegible]AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0000	← AR2
0x0081h	0x0001	← AR2
0x0082h	0x0002	← AR2
0x0083h	0	← AR2

Addr.	Old循环缓冲区	
0x0088h	0	← AR3
0x0089h	0	← AR3
0x008Ah	0	← AR3
0x008Bh	0	← AR3

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=3

[illegible]AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0000	AR2
0x0081h	0x0001	AR2
0x0082h	0x0002	AR2
0x0083h	0x0003	AR2

Addr.	Old循环缓冲区	
0x0088h	0x0000	AR3
0x0089h	0	AR3
0x008Ah	0	AR3
0x008Bh	0	AR3

1.| 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=4　　输入样本，并将AR2指向的存储单元

1.| 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=5　　输入样本，并将AR2指向的存储单元

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=6

[illegible]AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0004	← AR2
0x0081h	0x0005	← AR2
0x0082h	0x0006	← AR2
0x0083h	0x0003	← AR2

Addr.	Old循环缓冲区	
0x0088h	0x0000	← AR3
0x0089h	0x0003	← AR3
0x008Ah	0x0002	← AR3
0x008Bh	0x0001	← AR3

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=7 [illegible]AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0004	← **AR2**
0x0081h	0x0005	← **AR2**
0x0082h	0x0006	← **AR2**
0x0083h	0x0007	← **AR2**

Addr.	Old循环缓冲区	
0x0088h	0x0004	← **AR3**
0x0089h	0x0003	← **AR3**
0x008Ah	0x0002	← **AR3**
0x008Bh	0x0001	← **AR3**

1. 基础实验总结

循环缓冲区示意（输入序列x[n]样本值假设如下）：

时刻	0	1	2	3	4	5	6	7	8
x[n]	0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007	0x0008

n=8

[illegible]AR3指向的存储单元

Addr.	New循环缓冲区	
0x0080h	0x0008	← AR2
0x0081h	0x0005	← AR2
0x0082h	0x0006	← AR2
0x0083h	0x0007	← AR2

Addr.	Old循环缓冲区	
0x0088h	0x0004	← AR3
0x0089h	0x0003	← AR3
0x008Ah	0x0002	← AR3
0x008Bh	0x0005	← AR3

1. 基础实验总结

IIR滤波器的实现方法

优点：

可以用较少的阶数获得很高的选择特性，所用的存储单元少，运算次数少，具有经济高效的特点。

缺点：

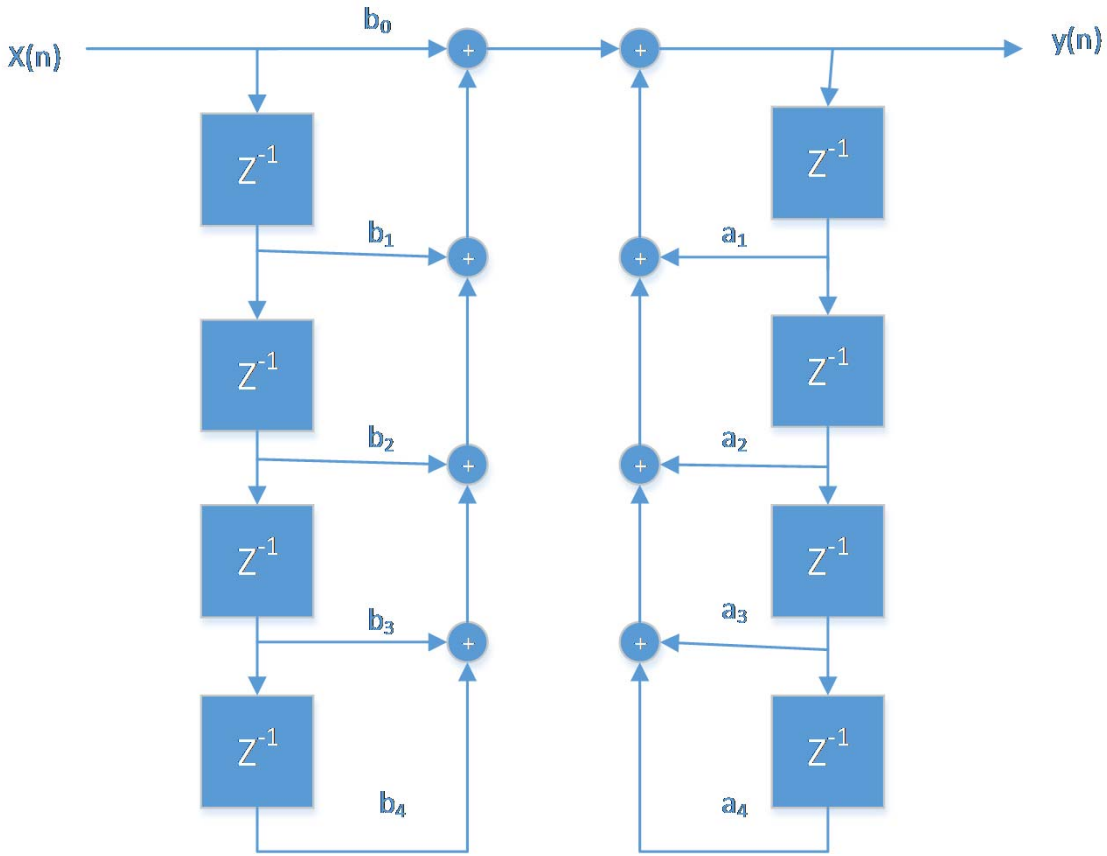
在有限精度的运算中，可能出现不稳定现象。且选择性越好，相位非线性越严重。

IIR滤波器一般传输函数

$$H(z)=\frac{\sum_{m=0}^{M} b_m z^{-m}}{1-\sum_{n=1}^{N} a_n z^{-n}}=\frac{Y(z)}{X(z)}$$

1.| 基础实验总结

以一个4阶IIR为例。说明IIR滤波器的直接形式C54x实现方法。



直接形式优点：

在迭代运算过程中先衰减后增益，系统的动态范围和鲁棒性都要好一些。

1.| 基础实验总结

差分方程为：

$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)+b_3x(n-3)+b_4x(n-4)+$$
$$a_1y(n-1)+a_2x(n-2)+a_3x(n-3)+a_4x(n-4)$$

在编程时，将变量和系数都存放在DARAM中，并采用循环缓冲区方式寻址，共需开辟四个循环缓冲区，用来存放变量和系数。

1.| 基础实验总结

IIR滤波器直接形式C54x实现的要点如下：

1）在数据存储器中开辟4个循环缓冲区：X循环缓冲区中存放X；Y循环缓冲区中存放Y。 B循环缓冲区中存放B系数； A循环缓冲区中存放A系数（4阶IIR滤波器中，各缓冲区长度N取5）；

2）在程序中将系数及变量初始值传入对应缓冲区；

3）初始化各辅助寄存器指向地址；

4）输入x(n),替换X缓冲区中最老数据；

5）计算前向通道；

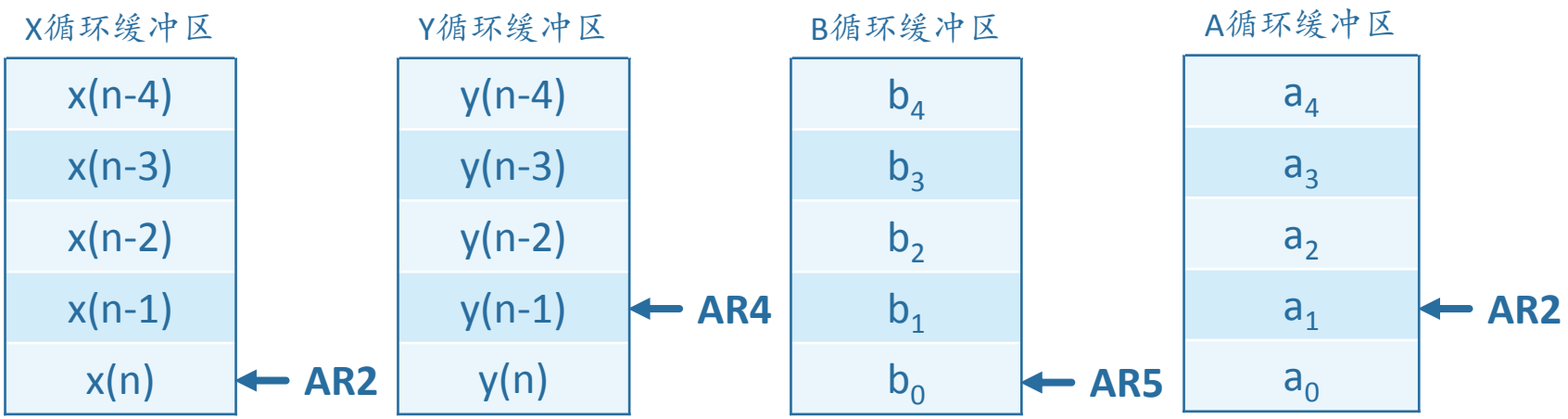
6）计算反馈通道；

7）输出由a_0系数衰减后的y(n),并用y(n)替换Y缓冲区中最老的数据；

8）修正AR3，保证其指向a_1；

9）Goto 4）

1.| 基础实验总结

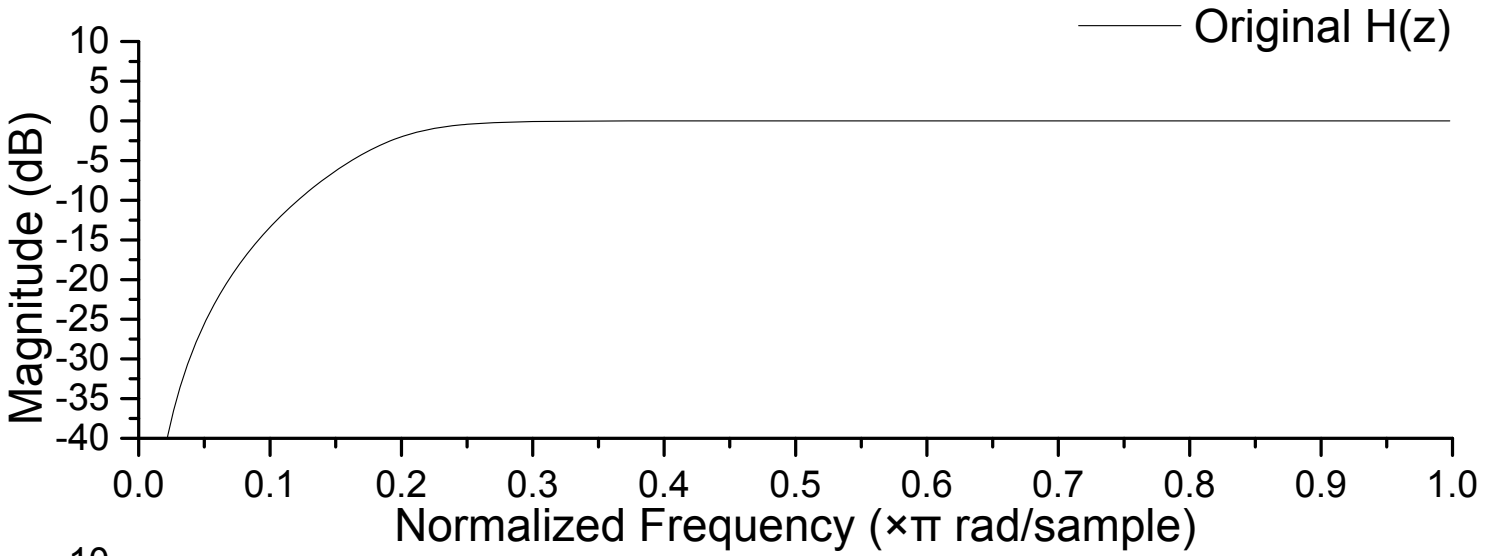

X循环缓冲区
x(n-4)
x(n-3)
x(n-2)
x(n-1)
x(n)
AR2
Y循环缓冲区
y(n-4)
y(n-3)
y(n-2)
y(n-1)
y(n)
AR4
B循环缓冲区
b_4
b_3
b_2
b_1
b_0
AR5
A循环缓冲区
a_4
a_3
a_2
a_1
a_0
AR2

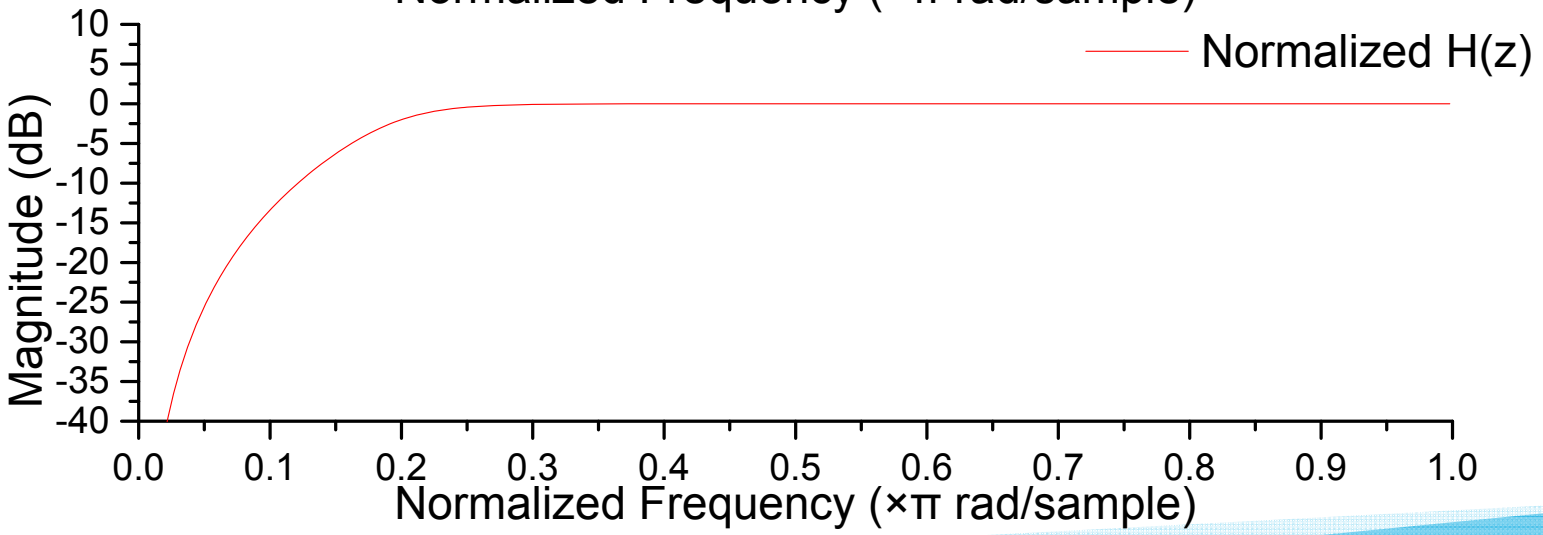
1. 基础实验总结

需要注意的是：

在设计IIR滤波器时，经常会出现系数大于1的情况，这种情况下由于极点在单位圆外会造成滤波器的不稳定，同时DSP计算字长有限，因此需要对滤波器系数归一化。这样以来由于$a_0<1$，则需要对每次的滤波器输出乘以a_0系数，相当一个衰减。

1.| 基础实验总结


Original H(z)
Magnitude (dB)
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Normalized Frequency (×π rad/sample)


Normalized H(z)
Magnitude (dB)
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Normalized Frequency (×π rad/sample)

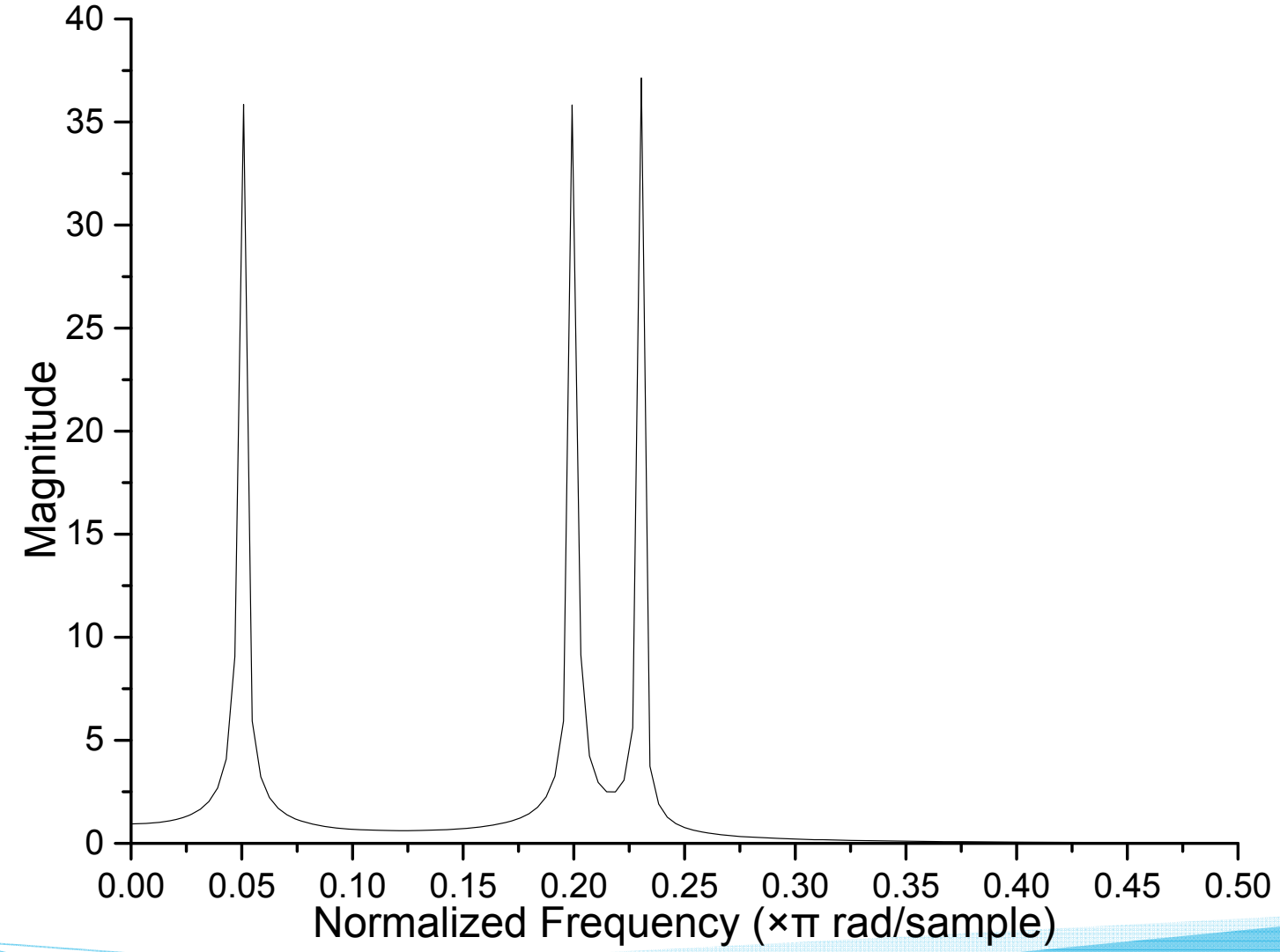
1.| 基础实验总结

```
table:  .word   0               ;X(N-4)
        .word   0               ;X(N-3)
        .word   0               ;X(N-2)
        .word   0               ;X(N-1)
        .word   0               ;Y(N-4)
        .word   0               ;Y(N-3)
        .word   0               ;Y(N-2)
        .word   0               ;Y(N-1)
        .word   3916;           b4=0.1195
        .word   -15667;         b3=-0.4781
        .word   23499;          b2=0.7171
        .word   -15667;         b1=-0.4781
        .word   3916;           b0=0.1195
        .word   1622;           a4=0.0495
        .word   -9264;          a3=-0.2827
        .word   20685;          a2=0.6313
        .word   -21636;         a1=-0.6603
        .word   9451;           a0=0.2884
```

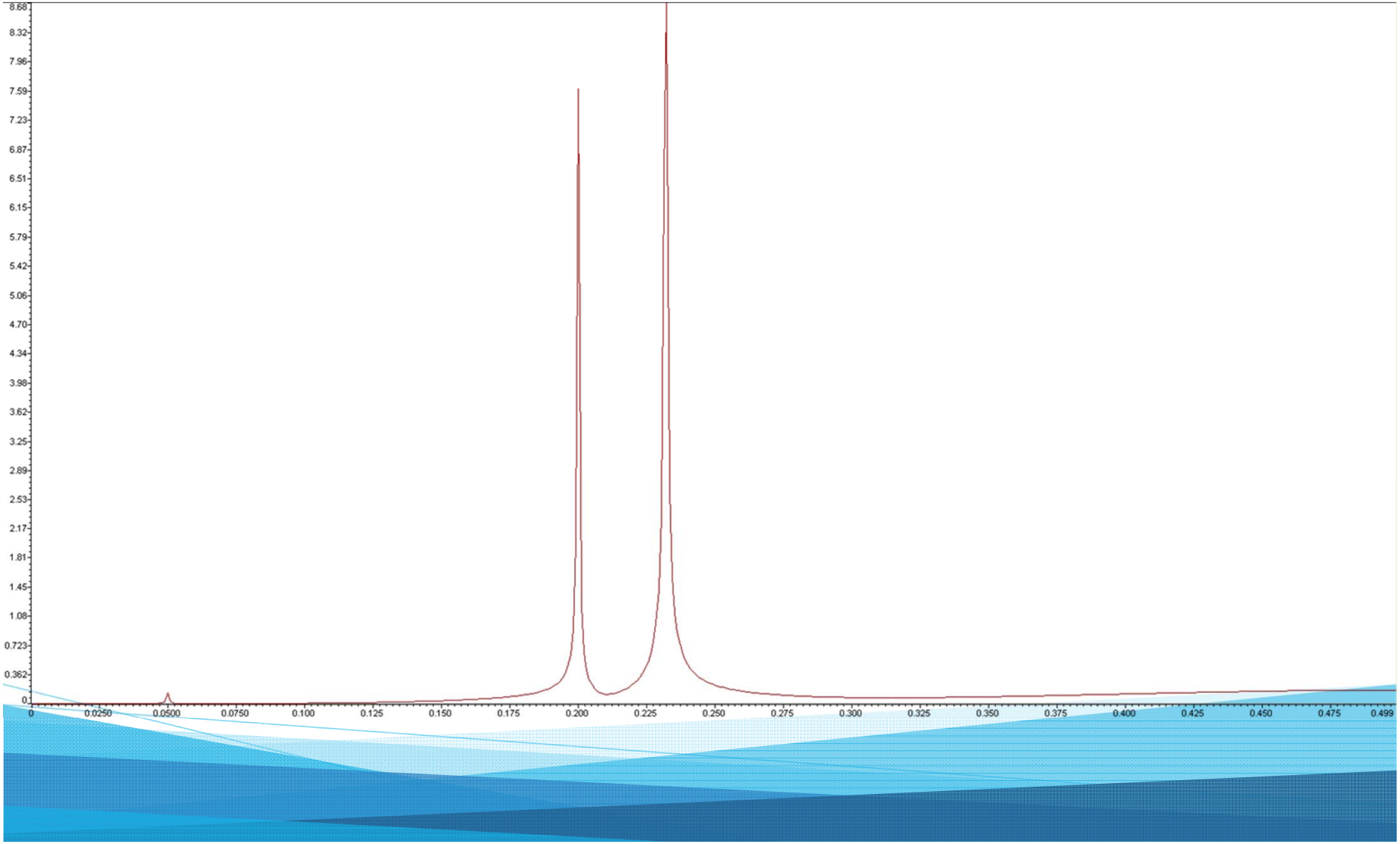
1.|基础实验总结

```
MAC     *AR2+0%,*AR5+0%,A     ;Forward Path
MAC     *AR2+0%,*AR5+0%,A
MAC     *AR2+0%,*AR5+0%,A
MAC     *AR2+0%,*AR5+0%,A
MAC     *AR2,*AR5+0%,A


MAC     *AR4+0%,*AR3+0%,A     ;Backward Path
MAC     *AR4+0%,*AR3+0%,A
MAC     *AR4+0%,*AR3+0%,A
MAC     *AR4+0%,*AR3+0%,A

MPYA    *AR3+0%               ;Output
STH     B,*AR1+
STH     B,*AR4
```

1.| 基础实验总结


40
35
30
25
20
15
10
5
0
Magnitude
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
Normalized Frequency (×π rad/sample)

1.| 基础实验总结


8.68
8.32
7.96
7.59
7.23
6.87
6.51
6.15
5.79
5.42
5.06
4.70
4.34
3.98
3.62
3.25
2.89
2.53
2.17
1.81
1.45
1.08
0.723
0.362
0
0
0.0250
0.0500
0.0750
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275
0.300
0.325
0.350
0.375
0.400
0.425
0.450
0.475
0.499

第2部分

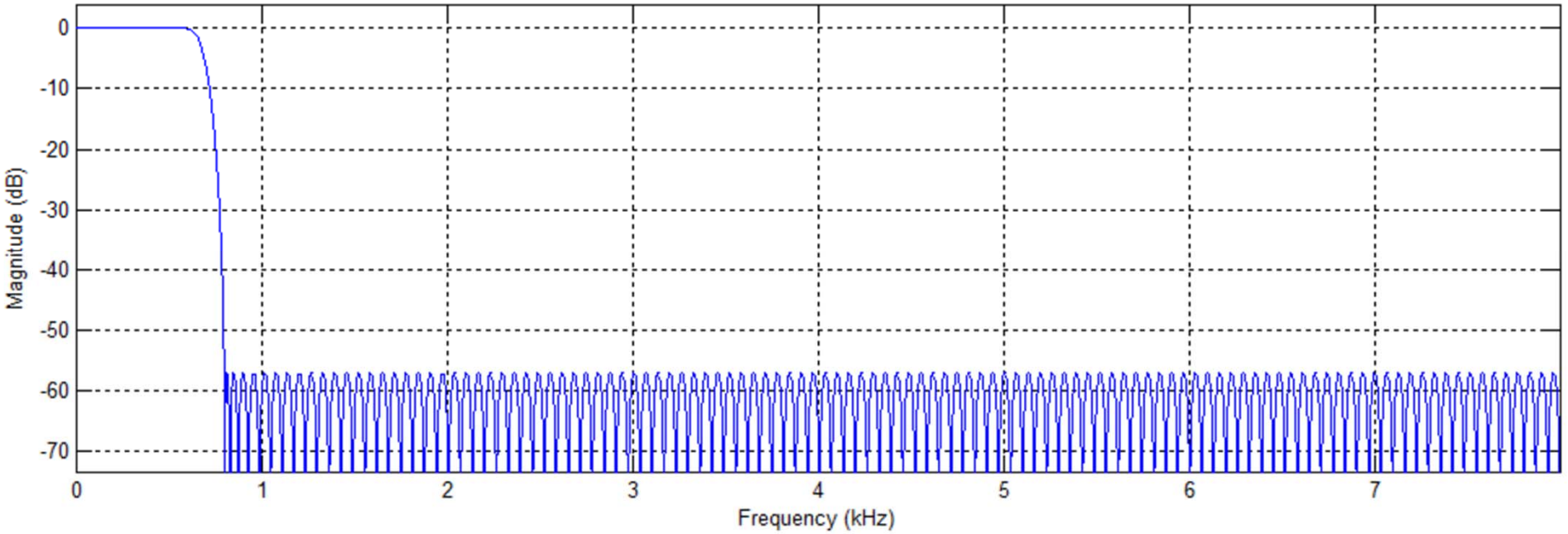
带噪声音滤波

2.| 带噪声音滤波

两段采样率为16KHz，时长为26秒的单声道音乐混合在一起，音乐A的频率范围为0~600Hz，音乐B的频率范围为1.5KHz~8KHz。使用DSP实现一个249阶，截止频率为600Hz的FIR滤波器对这个混合的音乐进行处理，以求输出尽可能为音乐A。音乐由地址为0x0010的IO端口进行采集，处理后数据由地址为0x0020的IO端口进行输出。

2.| 带噪声音滤波

Matlab设计出的滤波器频响



$$f_s = 16KHz \quad N = 249 \quad f_{pass} = 600Hz \quad f_{stop} = 800Hz$$

$$\eta = \frac{f_{stop}}{f_{pass}} \approx 1.33$$

2.| 带噪声音滤波

由Matlab生成输入.dat文件

```
clear;
MusicName='MusicNew201505270258.wav';          %设置.wav文件路径
[Music,Fs,BitLength]=wavread(MusicName);       %读取.wav文件
Quant=quantizer('fixed','floor','saturate',[16,15]);  %设置量化器
MusicData=num2hex(Quant,Music);                %转换音乐数据为16进制的2的补码小数
MusicFile='MusicDataNew201505270300.dat';      %设置用于音乐输入的.dat文件名
MusicID=fopen(MusicFile,'w');                  %打开.dat文件
for Index=1:1:size(MusicData,1)
    fprintf(MusicID,'0x%s\r\n',MusicData(Index,:));   %输出
end
fclose(MusicID);                               %关闭.dat文件
```

在GEL中定义输入输出IO端口

```
GEL_MapAdd(0x10,2,1,1,0);        /*IO Read Port*/
GEL_MapAdd(0x20,2,1,0,1);        /*IO Write Port*/
```

2.| 带噪声音滤波

使用CCS的Port Connect功能连接.dat文件

Port Address	Length	Page	Type	Filename
0x00000010	1	I/O	Read	C:\Documents and Settings\Administrator\My Documents\CCS Project\DeNoise201505301430\MusicDataNew201505270300.dat
0x00000020	1	I/O	Write	C:\Documents and Settings\Administrator\My Documents\CCS Project\DeNoise201505301430\MusicOutput.dat

使用软件中断方式模拟定时器中断输入输出数据

```
SecondCount:        STM     #(SampleRate-1),AR4
SampleCount:        INTR    #19
                    CALL    LPF
```

中断向量

```
TINT0_SINT3:    B       _c_int19_TINT0
                NOP
                NOP
```

2.| 带噪声音滤波

中断服务程序

```
_c_int19_TINT0:     PORTR       IO_Read,*(CurrentInput)
                    PORTW       *(CurrentOutput),IO_Write
                    RETE
```

程序常量预定义

```
IO_Read             .set        0010h
IO_Write            .set        0020h
FIR_Length          .set        250
MusicLength         .set        26
SampleRate          .set        16000
```

2.| 带噪声音滤波

程序初始化

```
_c_int00:       SSBX        INTM
                STM         #0000h,IFR
                STM         #0000h,IMR
                ST          #0,*(CurrentInput)
                ST          #0,*(CurrentOutput)
                STM         #Buffer_Old,AR0
                RPT         #(FIR_Length/2-1)
                ST          #0,*AR0+
                STM         #Buffer_New,AR0
                RPT         #(FIR_Length/2-1)
                ST          #0,*AR0+

                SSBX        FRCT
                STM         #Buffer_New,AR2
                STM         #(Buffer_Old+FIR_Length/2-1),AR3
                STM         #(FIR_Length/2),BK
                STM         #-1,AR0

                STM         #MusicLength-1,AR5
```

2.| 带噪声音滤波

主循环，26秒，每秒16000采样点

```
SecondCount:        STM     #(SampleRate-1),AR4
SampleCount:        INTR    #19
                    CALL    LPF
                    BANZ    SampleCount,*AR4-
                    BANZ    SecondCount,*AR5-
LOOP:               B       LOOP
```

LPF实现

```
LPF:                MVKD    *(CurrentInput),*AR2
                    RPTZ    B,#(FIR_Length/2-1)
                    FIRS    *AR2+0%,*AR3+0%,LPFCOEFF
                    STH     B,*(CurrentOutput)
                    MAR     *+AR2(1)%
                    MVDD    *AR2,*AR3+0%
                    RET
```

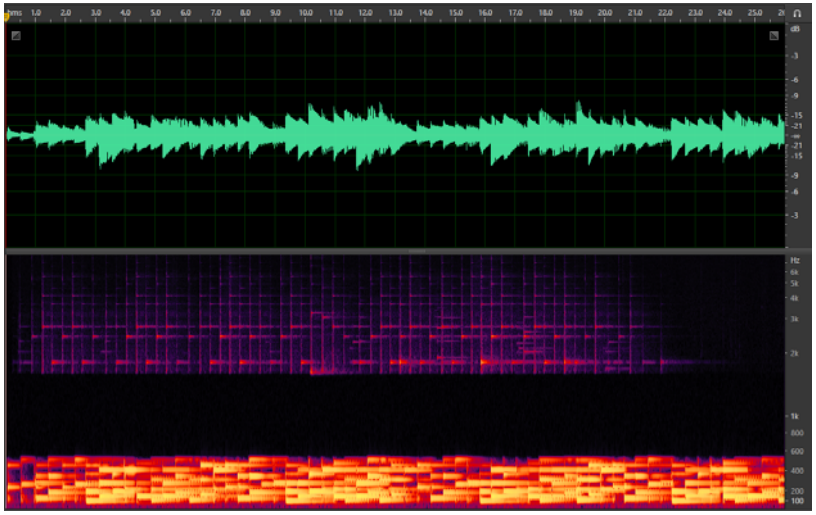
2.| 带噪声音滤波

程序内存分配

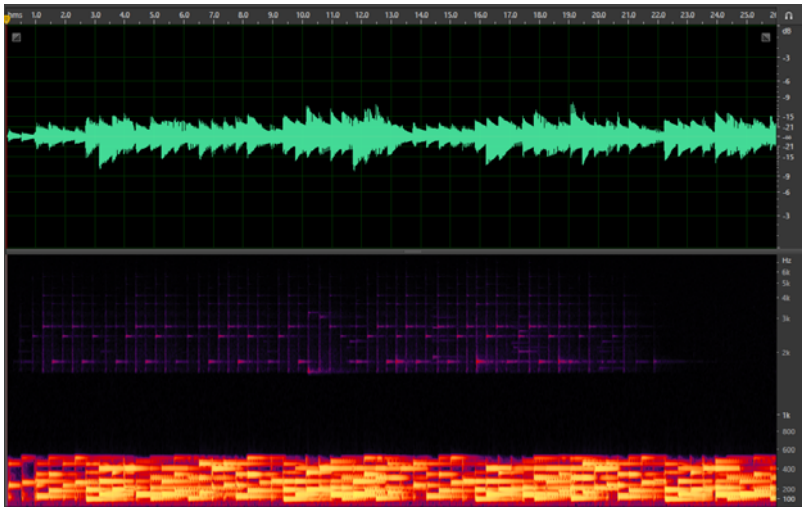
```
MEMORY
{
PAGE 0:
        PRORESERVED0:   origin = 00000h, length = 0080h
        PRODARAM:       origin = 03000h, length = 1000h
        PROROM:         origin = 0F000h, length = 0F00h
        PRORESERVED1:   origin = 0FF00h, length = 0080h
        INTVECTOR:      origin = 0FF80h, length = 0080h
PAGE 1:
        MMREG:          origin = 00000h, length = 0060h
        TEMPREG:        origin = 00060h, length = 0020h
        DATAREST:       origin = 00080h, length = 0f80h
        DATADARAM:      origin = 01000h, length = 2000h

}

SECTIONS
{
        LPFCOEFF:                       >PROROM         PAGE 0
        .text:                          >PROROM         PAGE 0
        .intvectors:                    >INTVECTOR      PAGE 0
        .bss:                           >DATADARAM      PAGE 1
        Buffer_Old: align (128) {}      >DATADARAM      PAGE 1
        Buffer_New: align (128) {}      >DATADARAM      PAGE 1
}
```

2.| 带噪声音滤波

实验结果

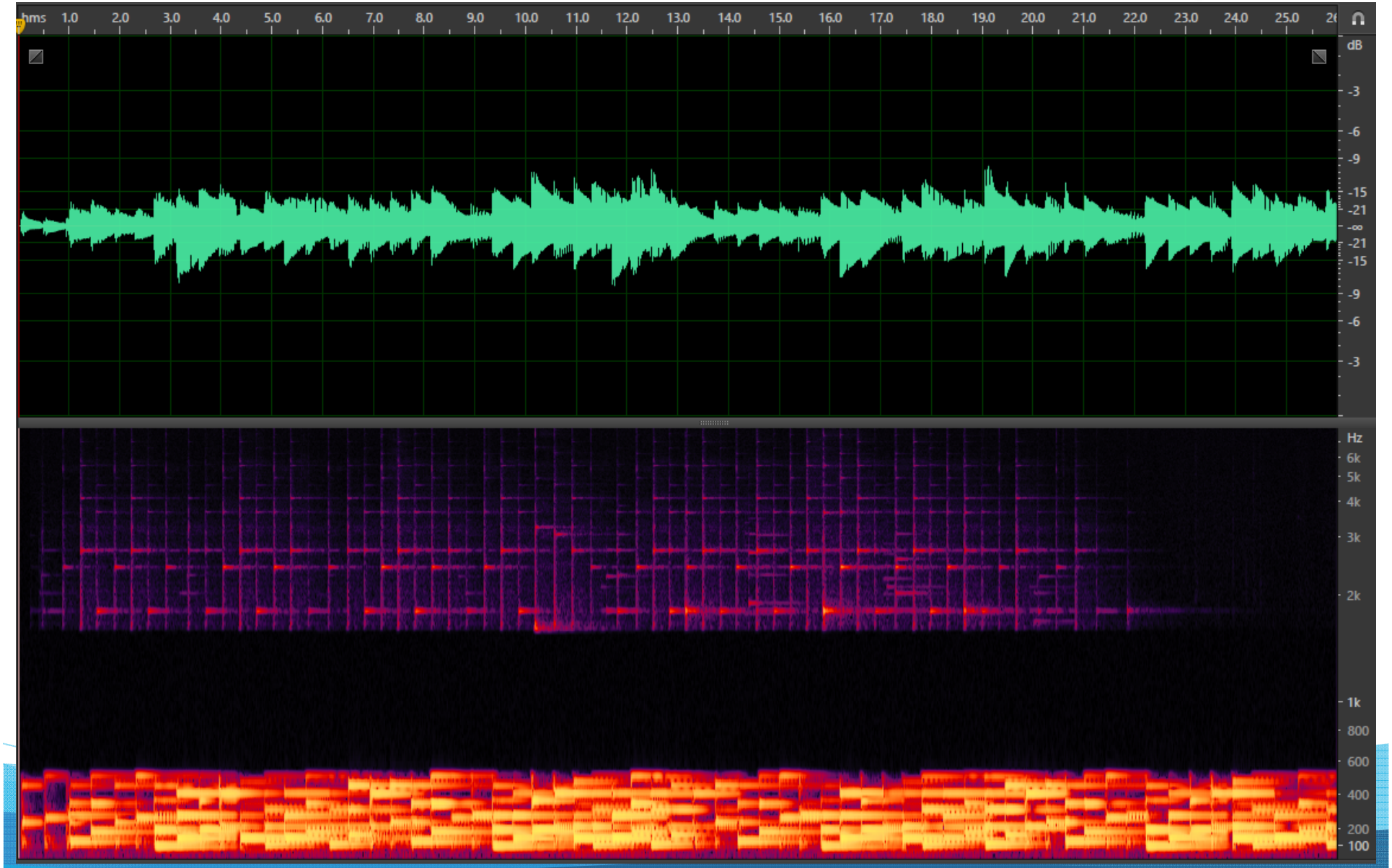


原始音频波形与时频图

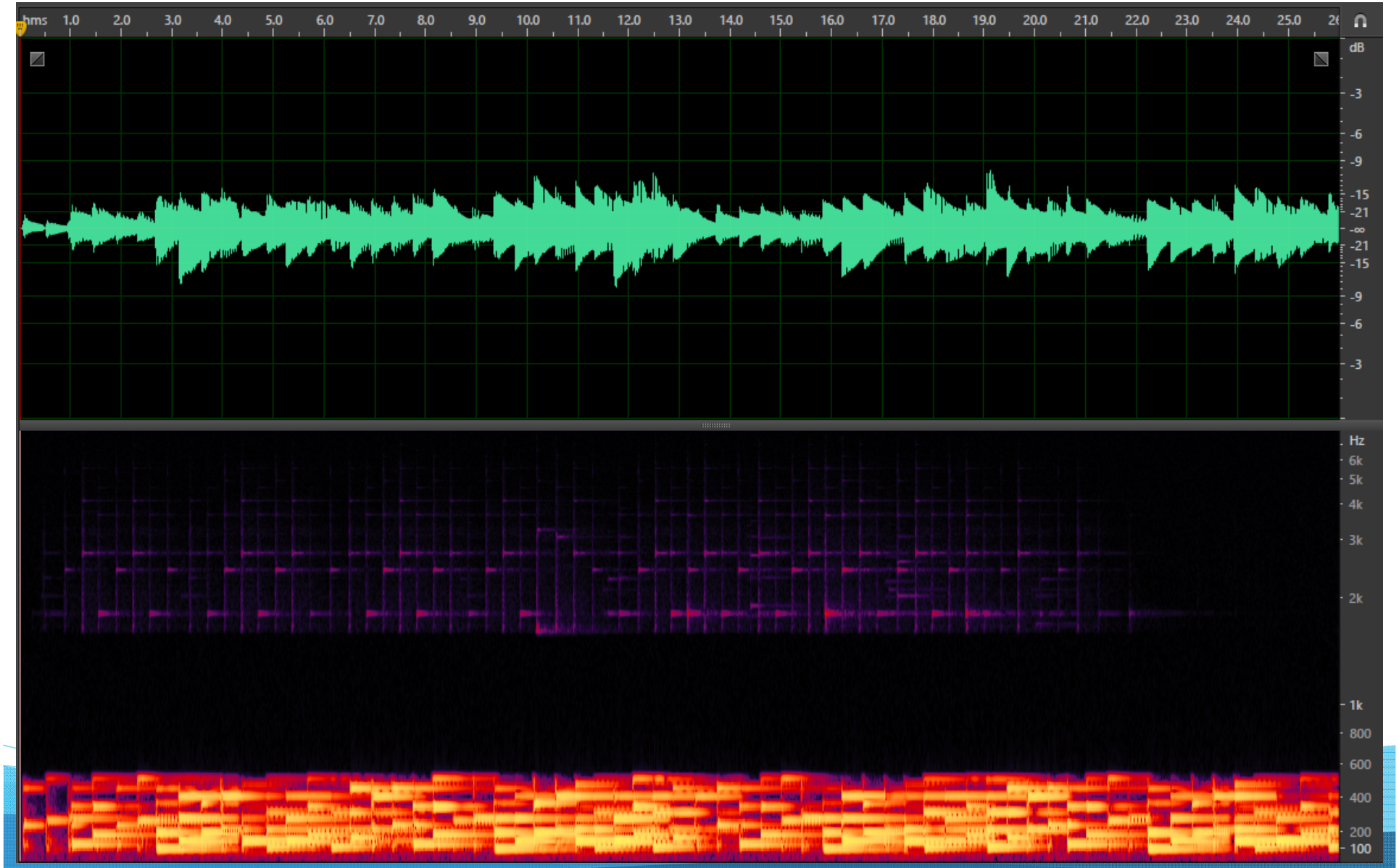


处理后音频波形与时频图

2.| 带噪声音滤波


hms 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0
dB
-3
-6
-9
-15
-21
-∞
-21
-15
-9
-6
-3
Hz
6k
5k
4k
3k
2k
1k
800
600
400
200
100

2.| 带噪声音滤波

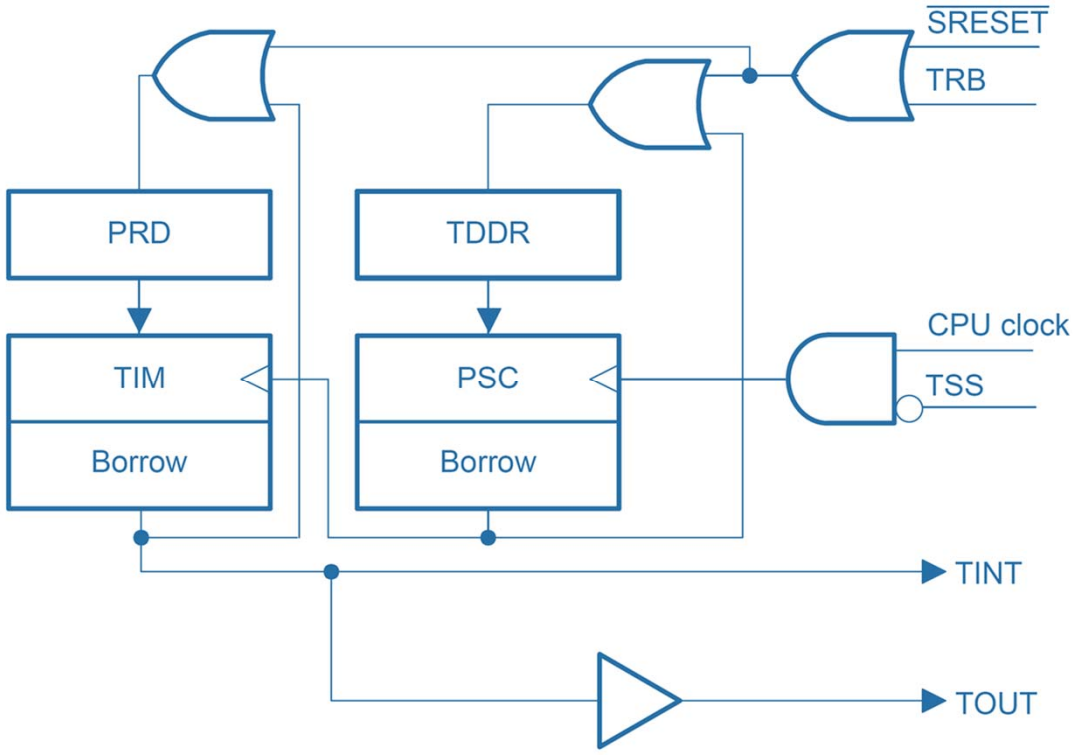

hms 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0
dB
-3
-6
-9
-15
-21
-∞
-21
-15
-9
-6
-3
Hz
6k
5k
4k
3k
2k
1k
800
600
400
200
100

第3部分

定时器的原理及应用

3. 定时器的原理及应用

定时器的原理框图



PRD: 定时器周期寄存器　　TTDR: 预定标分频系数

TIM: 定时器寄存器　　PSC: 预定标计数器

$$f_{TINT} = \frac{1}{t_c \times (TDDR + 1) \times (PRD + 1)}$$

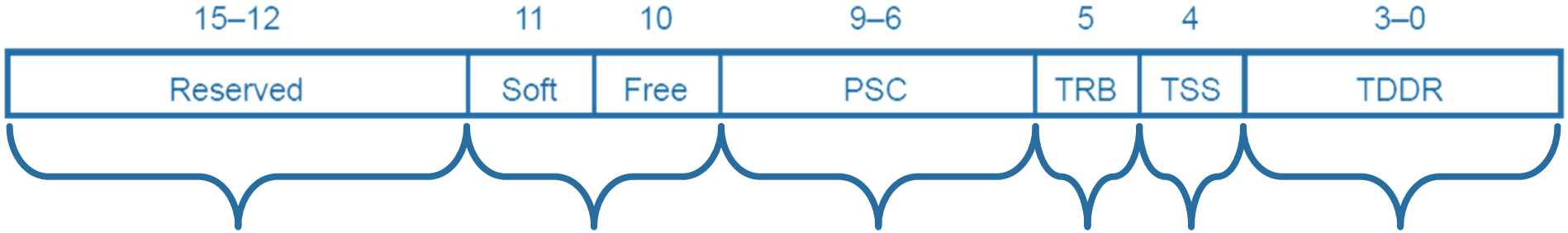
3.|定时器的原理及应用

定时器寄存器在存储器中的映射

Timer 0 Address	Timer 1 Address (C5402 only)	Register	Description
0024h	0030h	TIM	Timer register
0025h	0031h	PRD	Timer period register
0026h	0032h	TCR	Timer control register

3.|定时器的原理及应用

定时器控制寄存器TCR



保留位

软件中断定时器设置位

定时器预定标计数器

定时器重新加载位

定时停止状态位

定时器预定标分频位

3.|定时器的原理及应用

定时器初始化步骤

[1] 将TCR中的TSS位置1以关闭定时器
[2] 加载PRD
[3] 重新加载TCR（TDDR分频；TSS置0；TRB置1）以启动定时器

定时器中断初始化INTM=1

[1] 中断标志寄存器IFR的TINT对应位置1以清除中断标志
[2] 中断屏蔽寄存器IMR的TINT对应位置1以开放中断
[3] INTM=0，打开中断总开关

3.|定时器的原理及应用

举例：

一个DSP系统，外部晶振频率为10MHz，CLKMD[1..3]引脚电平为[0 0 1]，拟采用16kHz的采样频率对地址为0x0010的IO口进行采样（假设采样时外部设备已将数据准备好），如何使用定时器0进行采样？

CLKMD[1..3]引脚电平为[0 0 1]

$$f_c = 10MHz \times PLL = 100MHz$$

$$f_s = 16kHz \qquad N = \frac{f_c}{f_s} = 6250$$

3.|定时器的原理及应用

举例：

一个DSP系统，外部晶振频率为10MHz，CLKMD[1..3]引脚电平为[1 0 0]，拟采用16kHz的采样频率对地址为0x0010的IO口进行采样（假设采样时外部设备已将数据准备好），如何使用定时器0进行采样？

$$6250 = (TDDR + 1) \times (PRD + 1)$$

$$TDDR = 4 = (0100)_B$$

$$PRD = 1249 = (010011100001)_B$$

3.|定时器的原理及应用

定时器初始化步骤

[1] 将TCR中的TSS位置1以关闭定时器
[2] 加载PRD
[3] 重新加载TCR（TDDR分频；TSS置0；TRB置1）以启动定时器

定时器0初始化步骤

```
[1]    STM    0010h,TCR
[2]    STM    04E1h,PRD
[3]    STM    0024h,TCR
```

3.|定时器的原理及应用

定时器中断初始化INTM=1

[1] 中断标志寄存器IFR的TINT对应位置1以清除中断标志
[2] 中断屏蔽寄存器IMR的TINT对应位置1以开放中断
[3] INTM=0，打开中断总开关

定时器0中断初始化INTM=1

```
[1]    STM     0008h,IFR
[2]    STM     0008h,IMR
[3]    RSBX    INTM
```

3.|定时器的原理及应用

定时器0中断向量

```
[1]                         .ref      _c_int19_TINT0
[2] TINT0_SINT3:            B         _c_int19_TINT0
[3]                         NOP
[4]                         NOP
```

定时器0中断服务

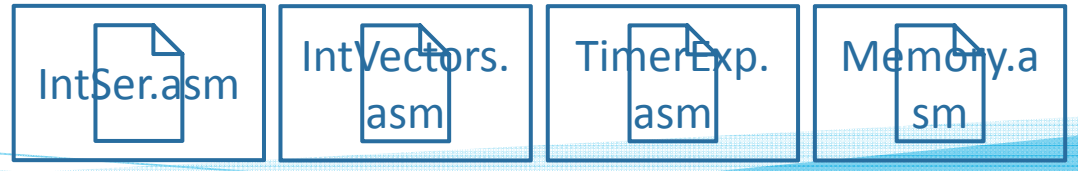
```
[1]IO_read                  .set      0010h
……
[2] _c_int19_TINT0:         PORTR   IO_read, *(CurrentInput)
[3]                         RETE
```

3.|定时器的原理及应用

仿真实验

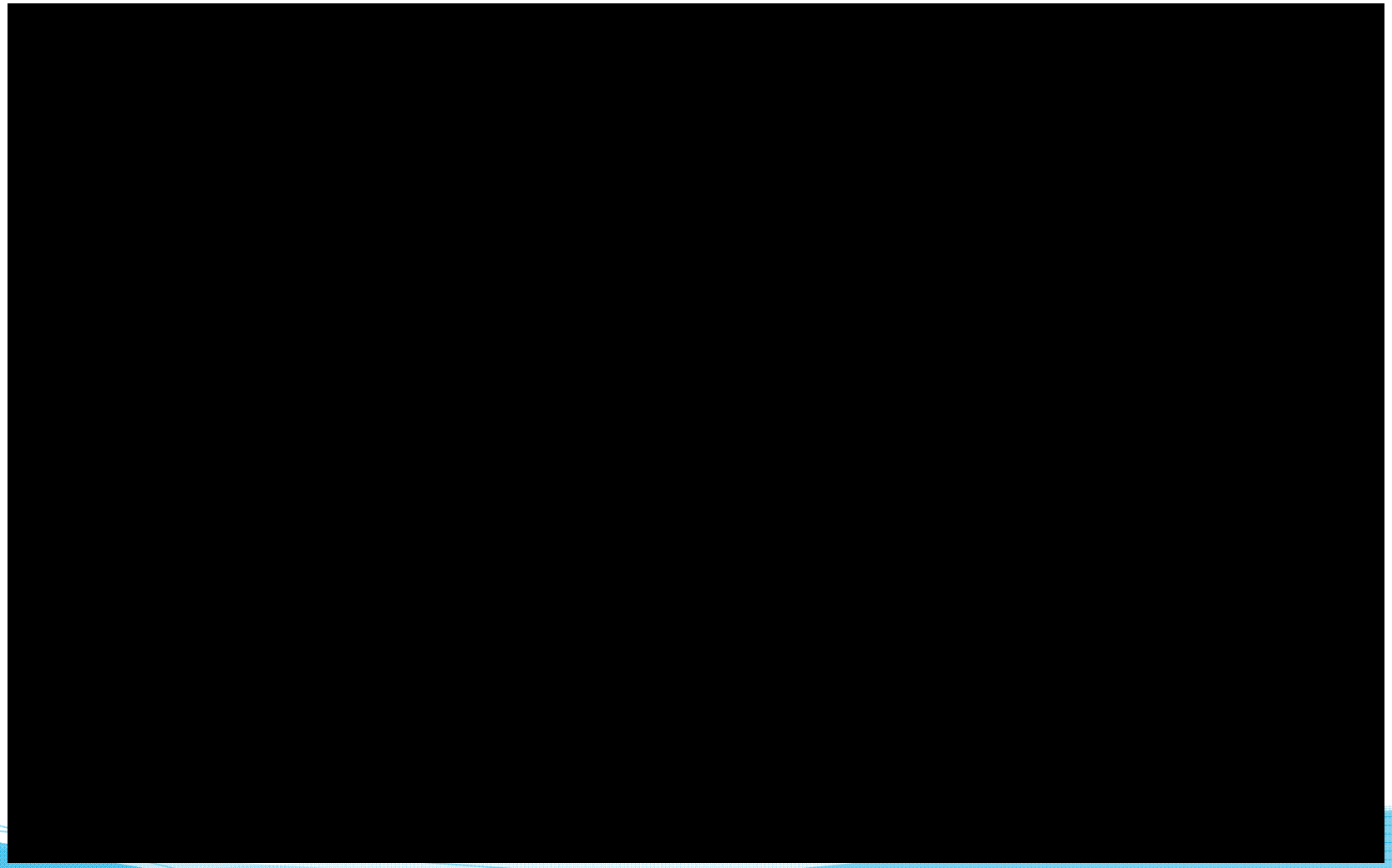
假设DSP系统主频为100MHz，采用16kHz的采样频率（逻辑对应6250时钟周期），使用定时器0对地址为0x0010的IO口进行采样。

实验中采用和系数对称FIR滤波器相似的200长度循环缓冲区（Buffer）存放数据，并且在每次采样之后按照时间先后顺序输出到显示缓冲区（DpBuffer），再使用CCS中的Graph功能查看DpBuffer，达到一个示波器的效果。



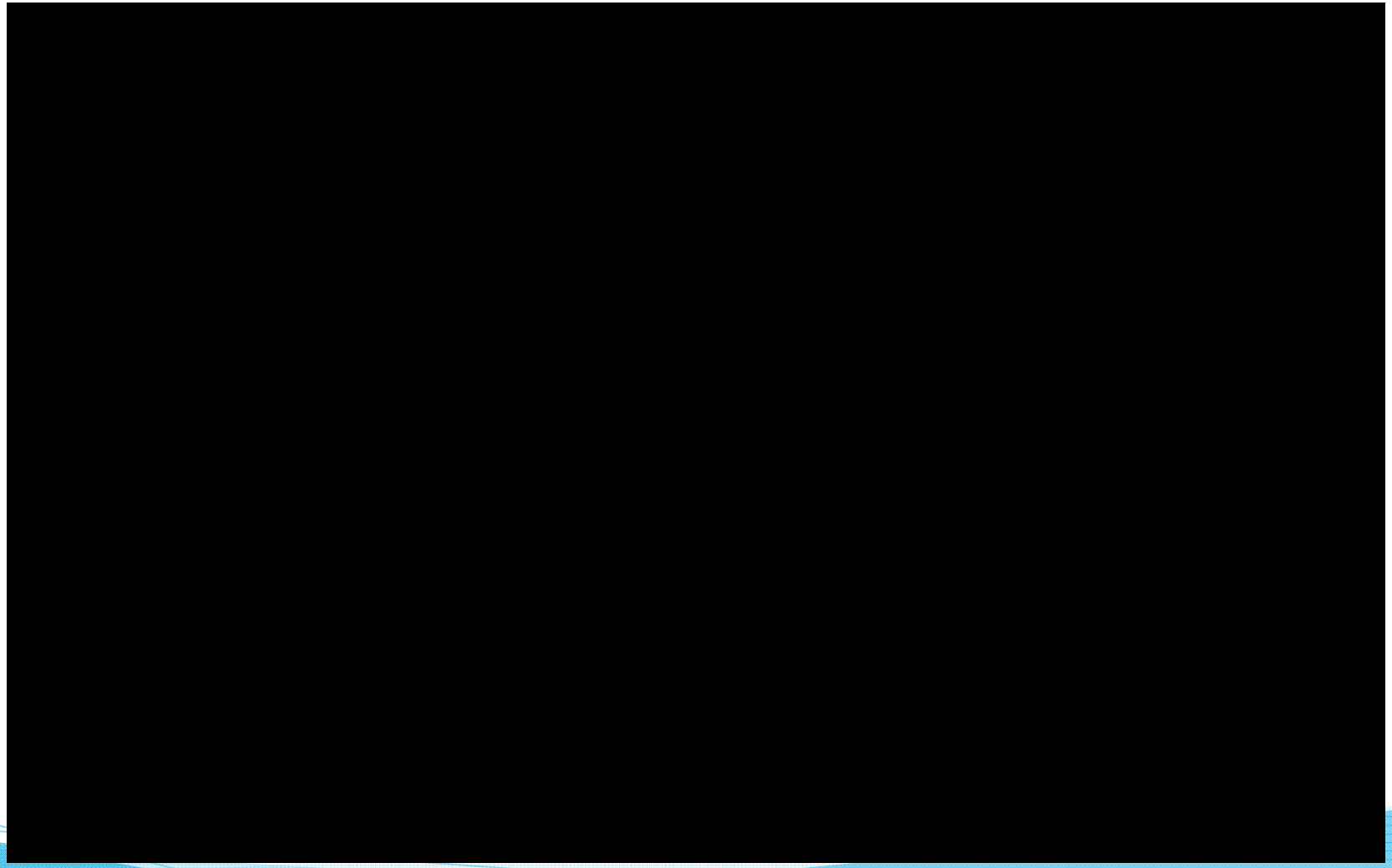
3.|定时器的原理及应用

对输入的正弦信号进行显示



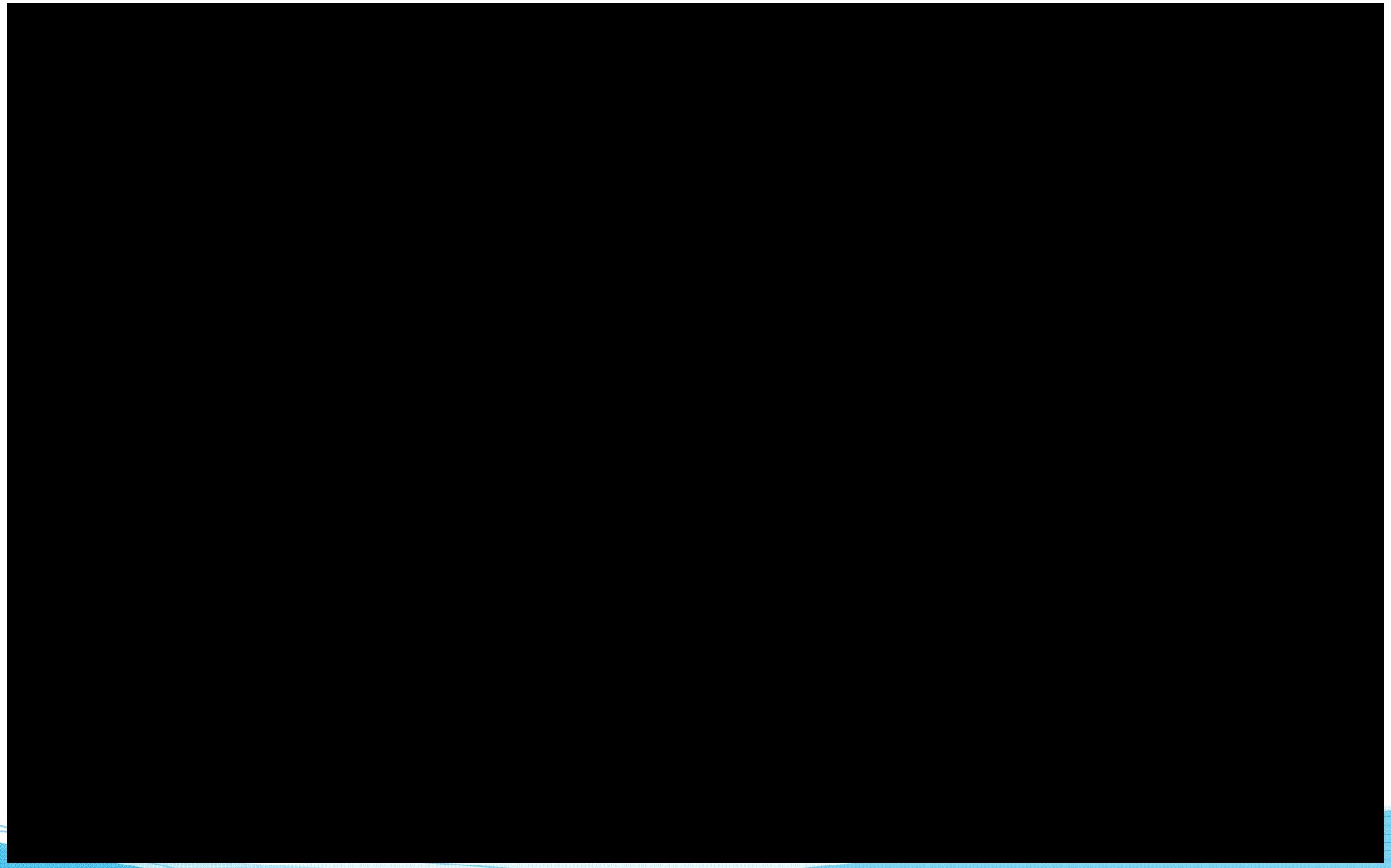
3.|定时器的原理及应用

对输入的声音信号进行显示



3.|定时器的原理及应用

Clock 计数



总结

1、学习并掌握了C54x系列DSP的原理及汇编编程方法；
2、了解并实现常用信号处理算法的DSP实现；
3、学习了C54x系列DSP定时器的原理及使用；

展望

根据TMS320VC5402芯片及相关芯片规范，完成一个简单音频处理系统电路及PCB设计；

参考文献

[1] INSTRUMENTS T. TMS320C54x DSP Reference Set Volume 1: CPU and Peripherals [M/OL]. 2001

Thank You!