摘 要

随着人们对各种复杂信号或特殊信号的需求，随着不断发展的模拟和数字信号处理技术及其大规模集成电路工艺技术的发展而蕴育出来的一种新型的信号发生器---任意波形发生器。而通常情况通过下述两种方法来产生所需要的波形。一种方法是使用算法直接产生，这种方法能直接精确地计算出每个角度的波形值，所占的存储空间小。另外一种为查表法，使用这种方法需要较多的存储空间，但是实时性较好。本文我们来讨论第二种。基于 DDS 技术的任意波形发生器不仅能实现高稳定度、高精度、高分辨率的要求，还具有体积小、价格便宜的特点，是一种很有发展前途的信号源。

本论文在分析了现有波形发生器设计方案的基础上，合理地使用了 DDS 技术，以 TI 公司的 TMS320C5402 数字信号处理芯片为核心，结合应用背景设计了一种结构简便性能优良的任意波形发生器。论文中主要对 DSP 芯片控制的任意波形发生器的软硬件设计进行了相应的研究，着重分为几个部分对系统设计进行了介绍。

论文介绍了采用 VC++生成各种波形发生数据的原理及方法，整个系统以 DSP 作为核心控制器，数模转换部分采用 16 位高速的 AD7846，然后经过低通滤波器滤波即获得所需波形，后面的可编程增益运放 PGA205 和 PGA103 则可以对波形的幅度进行控制。其特点是界面友好，操作方便，产生的波形失真度小，频带宽，频率分辨率高。

关键词:任意波形，TMS320C5402，AD7846，低通滤波器

Abstract

With the desire of complicated and special signal, with the dewelopment of analog and digital signal processting, Arbitrary Waveform Generator has occurred. Generally speaking, it can generate waveform through two ways. One is using arithmetic, Another is looking up the waveform-list. The first method can calculate the waveform value accurately, The second method need much memory space, but it is high definition, high stability and low-cost, small volume. It is a promissing signal generator.

This paper analyses the design mathods of the Waveform Generator which occur in the market, uses Direct Digital Synthesis technology reasonably. Introduce a design method of Arbitrary Waveform Generator with DDS technology and framework of the software and hardware.

This paper introduces the design software and hardware. Introduce the theory of making various wave data. The whole system is controlled by DSP, D/A converter is the high speed AD7846, then pass the LPF, we can get the wave we need. The latter are PGA205 and PGA103, they are all programmable-gain amplifiers, They can control the waveform amplitude. The characteristics of this design are friendly interface, convenient operation, small distortion, wide frequerncy band, high frequency difference.

Key word: Arbitrary Waveform TMS320C5402 AD7846 LPF

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：[signature] 日期：2007年 6月22日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：[signature] 导师签名：[signature]

日期：2007年 6月22日

第一章 绪论

1.1 引 言

信号发生器是一种常用的信号源，广泛应用于电子电路、自动控制和科学试验等领域。它是一种为电子测量和计量工作提供符合严格技术要求的电信号设备。因此，信号发生器和示波器、电压表、频率计等仪器一样是最普通、最基本的，也是应用最广泛的电子仪器之一，几乎所有的电参量的测量都需要用到信号发生器。

在 70 年代前，信号发生器主要有两大类：正弦波发生器和脉冲波发生器。正弦波发生器只提供正弦波信号。通常使用的技术是——自由振荡器，工作频率即为输出频率，频率范围有限，一般从几 Hz 至约 1MHz。脉冲波发生器可产生高质量的方波和脉冲串，其频率范围一般低至 1Hz，高至 1 GHz，它被用在数字系统中模拟或取代数字信号，如时钟，数据等。介于两者之间的还有函数发生器，它提供正弦，余弦，方波，三角波，斜波等几种常用的特殊波形。产生有别于上述波形时，必须采用较复杂的电路或机电结合的方法，甚至采用模拟计算机构成专用设备。

在 70 年代后，微处理器的出现，利用信号处理器，A/D 和 D/A 以及软件使函数产生器的功能扩大，能够产生更复杂的波形，衰减振荡波，随机脉冲波，指数形脉冲等。当时的信号处理器是专门用于信号处理的微处理器，但时钟频率只有 1～2MHz, A/D 和 D/A 一般在 8 位左右，内部存储器约 2K，因此能够产生的正弦波的等效频宽不会超过 1MHz，要获得比较平滑和失真度低的波形，重复频率不能超过 10KHz，当时用模拟方式产生特殊波形，重复频率可以达到 1～10MHz，波形完整性好，用数字电路的函数发生器尚处于开发阶段，正式产品还不多。

80 年代情况有很大变化，随着现代信息事业的发展，测试对象不断丰富，现代通信系统和电子系统对测试系统提出了越来越高的要求，进而对信号发生器也提出了更高的要求，需要模拟工作现场的情况来对产品进行测试。而工作现场的信号往往是多种多样的，传统的模拟信号发生器显然是不能满足客观的需要的，这时就需要一种能产生用户定义的波形的仪器[1]。

经过这些年的发展，近年来一种新的电子测量仪器——任意波形发生器出现了，它可视为函数发生器的换代产品。任意波形发生器的功能远比函数发生器强，在前面讨论的难于产生的或不能产生的波形，都可以使用任意波形发生器。自然它也可以用来产生前面讨论的波形(方波，三角波，脉冲波)，但实际中还利用发生器来模拟更复杂的信号，甚至信号中的缺陷(如方波中的过冲和数字信号中的尖脉冲)都可通过控制来模拟，再者，任意波形发生器还可产生瞬变信号如阻尼正弦波等，它对存在的各种波形都可以模拟，只要可用数字形式存储，并送进波形存储器的波形都可以把他们模拟出来。

早期的信号源主要是基于模拟电路来实现，而任意波形发生器是以数字电路和计算机技术为基础的产品，因此可使它成为测试系统通用的高性能，多功能的激励源，因而将有很广阔的发展前景。

1.2 国内外发展状况

国外任意波形发生器的研制及生产技术已较为成熟，已有多种产品投放市场，目前任意波形发生器有三种产品结构形式:

• 独立仪器结构形式

独立仪器的结构形式是把任意波形发生器设计成单台仪器的形式，其优点是精度高，可单独工作。

• PC 总线插卡式

PC 总线插卡式是将任意波形发生卡直接插在 PC 机的总线扩展槽或扩展机箱中，利用 PC 机来控制任意波形发生器的工作状态，其优点是可以充分利用 PC 机的软硬件资源，在波形数据处理，波形参数的修改，计算等有明显优势。

• VXI 模块式

VXI 模块式是一种新型的模块化仪器，它必须插在 VXI 总线机箱上才能使用，而 VXI 总线机箱又通过 GPIB 或 RS-232C 等接口与计算机相连，VXI 卡式仪器对组成自动测试系统(ATE)特别有用，各个公司的 VXI 卡式仪器模块均可自由组合使用。

Agilent 和 Tektronix 两大公司在此领域进行了卓有成效的研究和开发，其代表产品无论在技术先进性还是市场占有率方面都在全世界享有卓越声誉。这两个公

司任意波形发生器的产品型号和性能特点分别如下表[1]所示:

表 1.1 国外任意波形发生器主要产品

生产厂家		型号	性能特点
Agilent 公司	独立仪器	HP33120A	最高采样速率：40MSPS 输出标准波形：正弦波、方波、三角波、锯齿波、噪声、直流电压、负锯齿波、指数函数波 任意波形：8－16000 个点 幅度分辨率： 12 位 调制： AM/FM/FSK/Burst
		HP33250A	最高采样速率： 200MSPS 输出标准波形：正弦波、方波、三角波、锯齿波、噪声、直流电压、负锯齿波、指数函数波 任意波形：1－64000 个点 幅度分辨率：12 位 调制：AM/FM/FSK/Burst
	VXI 模块	E1445A	尺寸：C 类型：消息基 最高采样速率： 40MSPS 输出标准波形：正弦波、方波、三角波、锯齿波、噪声、直流电压、负锯齿波、指数函数波 任意波形：256K 个点 幅度分辨率：13 位 调制：AM/FM/FSK/Burst

Tektronix 公司	独立仪器	AWG610	最高采样速率：2.6GSPS 任意波形：8M 个点 幅度分辨率：8 位 调制：AM/FM/FSK/Burst
		AWG2021	最高采样速率：250MSPS 输出标准波形：正弦波、方波、三角波、锯齿波 任意波形：256K 个点 幅度分辨率：8 位 调制：AM/FM/FSK/Burst
	VXI 模块	VX4792	尺寸：C 类型：消息基 最高采样速率： 250MSPS 输出标准波形：正弦波、方波、三角波、斜波、脉冲波、任意波、直流 最大频率范围：5－125M 任意波形存储深度：256K 个点 幅度分辨率：12 位
		VX4790A	尺寸：C 类型：消息基 最高采样速率： 25MSPS 输出波形：正弦波、方波、三角波、锯齿波、直流电压 幅度分辨率：12 位

我国的任意波形发生器近几年也发展迅速，取得可喜的成果。但是总体水平还是跟国外有差距，国内主要是一些 PC 仪器插卡，独立仪器和 VXI 系统的模块很少，另外种类和性能也有一定的差距，因此加强此类的研究尤其重要。

1.3 课题来源和主要工作

信号源是一种非常常用的实验仪器，随着科学技术的发展，信号源的应用领域越来越广，诸如电子自动化检测、视频信号、雷达、医学等等。生活中的许多实验仪器、设备也会用到一些特殊的信号，它们的工作频率不是很高，但是波形精度要求很高，像低转速角轴探测器等就是这种。用户给出的大多是波形的幅度数据或者直接给出实验波形，而非公式表示，如雷电等。

本课题就是针对角轴探测器这样的一些高精度、低速系统的设备设计的一种任意波形发生器。

课题的主要工作是研究数字信号处理器在任意波形发生器中应用的可行性。设计一种简便实用、性能优良的任意波形发生器，该任意波形发生器能产生正弦波，方波，三角波等常用的标准信号，还能根据用户的需要生成任意波形，整个系统采用 DSP 控制。

作者主要工作：

（1）根据课题要求，参阅有关波形发生器的相关资料和文献，了解技术的有关知识和疑难点，作出系统的整体规划和设计。

（2）对系统各功能电路进行深入地分析和理解，对系统中的难点进行重点讨论和设计，并采用模块化的思想详细划分各个模块的功能。

（3）查找系统中所涉及的各电子元器件的资料，进行分析、比较和选择，并完成电路的硬件设计和 PCB 布板，对系统的软件部分进行编程，并进行 CCS 软件仿真。

（4）对系统进行调测，并对设计进行总结和展望，提出一些改进的方法和思路。

论文还研究了在任意波形发生器设计时如何根据系统指标合理地确定设计方案，在系统的硬件设计时具体要注意的问题以及如何结合先进的电子设计自动化方法来进行电路的设计，还有相应的软件编制的问题等等，对于目前三种典型的任意波形发生器的结构——PC 总线插卡式、独立仪器、VXI 模块都有重要的参考价值。

第二章 任意波形发生器的理论分析

2.1 DDS 特点

数字合成(Direct Digital Synthesizer 简称 DDS)技术[1][2][3][4]是一种新型的频率合成技术，它从相位出发直接合成所需的波形。这种技术首先由 J.Tierney, C.M.Rader 和 B.Gold 于 1971 年提出，但限于当时的技术和工艺水平，DDS 技术仅仅限于理论研究，而没有应用到实际中去。近 20 年来，随着 VLSI(Very Large Scale Integration)，FPGA(Field Programmable Gates Array)以及高速 DSP 的发展，这种结构独特的频率合成技术得到了飞速发展。它以广泛应用于通讯、导航、雷达、电子对抗以及现代仪器仪表等电子系统中。DDS 技术同传统的频率合成技术相比，有以下几个突出的优点[2]：

◆极快的频率切换速度

DDS 是一个开环系统，无任何反馈环节，频率转换时间主要由 LPF 附加的时延决定。

◆极高的频率分辨率

由 $\Delta f = f_c/2^L$ 知，只要增加相位寄存器的位数即可获得任意小的频率分辨率。大多数 DDS 的分辨率在 Hz、mHz 甚至 uHz。

◆较低的相位噪声和低漂移

DDS 系统中合成信号的频率稳定度直接由参考源的频率稳定度决定，合成信号的相位噪声与参考源的相位噪声相同。

◆连续的相位变化

因为 DDS 是个开环系统，故当一个转换频率的指令加在 DDS 的数据输入端时，它会迅速合成所需的频率信号，在输出信号上没有叠加任何电流脉冲，输出变化是一个平稳的过渡过程，而且相位连续变化。

◆输出频带范围广

DDS 的最低输出频率是所有的时钟频率的最小分辨率或相位累加器的分辨率。奈奎斯特采样定理保证了在该时钟频率一半下的所有频率，DAC 都可以再现信号。

2.2 DDS 的原理[1][2][3][4]

直接数字法是采用直接数字合成(DIRECT DIGITAL SYNTHESIS)的方法实现信号产生。该技术具有频率转换速度快、频率分辨率高、易于控制的突出特点。直接数字合成技术近年来发展得很快，而要产生任意波形就必须采用直接数字合成技术。随着 DDS 技术的发展，出现了各种各样的直接数字合成的结构，但基本上可分成两种：

a 基于地址计数器的数字频率合成法

b 基于相位累加器的数字频率合成法

1. 基于地址计数器的直接数字合成法

(1)结构框图

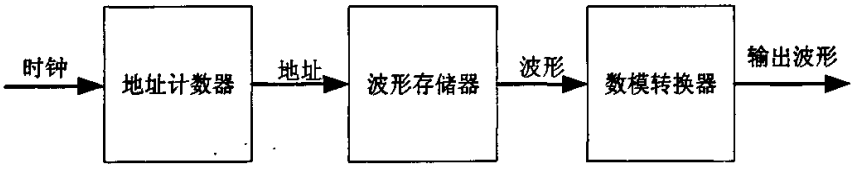


图 2—1 基于地址计数器的直接数字合成结构框图

这是一种最简单的直接数字合成方式，这种直接数字频率合成器的结构包括地址计数器，存储器和 D/A 转换器。

(2)工作原理

将波形数据存储于存储器中，而后用可程控的时钟信号为存储器提供扫描地址，与每个地址相对应的数据则代表波形在等间隔取样点上的幅度值。数据被送至 DAC，从而产生一个正比于其数字编码的电压值，每个电压值保持一个时钟周期，直至新的数据送至 DAC，经数/模转换后得到所需的模拟电压波形。在存储器里的数据产生的波形是对“取样波形”的阶梯近似。假定地址计数器的时钟频率为 f_c，波形一周期内有 n 个采样值，那么合成的波形频率为：

$$f_o = f_c / n \qquad (2-1)$$

如果改变地址计数器的时钟频率或 ROM 的地址步进大小，合成波形的频率都会随着改变。而要改变波形，只要在只读存储器中写入不同的数据。

2. 基于相位累加器的直接数字合成法

(1)结构框图

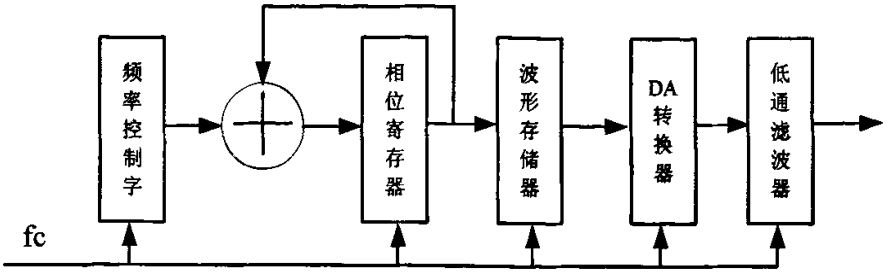


图 2－2 基于相位累加器的直接数字合成结构框图

这种结构中主要由相位累加器、数据 ROM 和 D/A 变换器组成，它是种全新的直接数字合成方式(目前国内外文献中提到的 DDS 一般是指这种方式，下面不作说明均指这种方式)。

(2)工作原理

将要产生的波形数据存入波形存储器，然后在参考时钟的作用下，对输入的频率数据进行累加，并且将累加器的输出一部分作为读取波形存储器的地址，将读出的波形数据经 D/A 转换为相应的电压信号，D/A 转换器输出的一系列的阶梯电压信号经低通滤波后便输出了光滑的合成波形的信号。以合成正弦波为例，通常我们考虑一个正弦波时习惯使用正弦波的幅度－时间表达式：

$S(t) = A\sin(\omega t + \varphi)$，正弦函数幅度的非线性使依据幅度产生任意频率的正弦波非常困难，但我们注意到，正弦波的相位是线性变化的，DDS 技术的关键就在于充分利用了正弦波相位线性变化这一特性，在 DDS 芯片中，其核心部件是相位累加器和 SIN 函数表，下面作一简单介绍：

• 相位累加器

相位累加器在功能上说实质是一个 N 位快速可循环累加器(目前，一般的 DDS 芯片中，相位累加器已达到 N=32 位)，N 位的相位累加器在每一个时钟来临时与频率控制所决定的相位增量 $A\varphi$(通常就是频率控制字 K，K 为二进制整数，1～2^N-1)累加一次，计数大于 2^N 时则自动溢出，保留后面 N 比特的数字于累加器中。每当相位累加器计数满后，可自动循环重新累加，所以输出相位可以保持连续变化，这就保证了输出正弦波的连续性。

• 正弦函数相位--幅度转换表(Sine Look-Up Table)

相位累加器的输出是随时间不断线行变化的用 N 位二进制数表达的相位信息，相位信息是无法直接利用的，必须设法把相位信息转换成幅度信息，在 DDS 技术中，人们把对应于不同相位的 Sine 函数的幅度值储存在 ROM 中，称此 ROM 为 SinLUT，相位累加器的输出相位作为 LUT 的寻址地址，LUT 相应地址单元存储的就是对应于该相位的正弦函数幅度值，为了保证 DDS 输出频率的分辨率，相位累加器的位数 N 要作得很多(如 32 位)，但是，要作出一个寻址能力为 2^{32} (4G)的 LUT，既不现实，也不必要。因此，可以只取相位累加器的高 A 位作为 LUT 的寻址信号，一般地，取 A=12 就可以完全满足精度的需要了。

(3). DDS 的技术性能及特点

根据上述原理：当参考时钟频率为 f_c，累加器的二进制位数为 N，频率控制字为 k，且波形存储器仅存储一个周期的合成信号的数据时，则合成输出的信号频率为：

$$f_O = \frac{k}{2^N} f_C \qquad (2-2)$$

频率分辨率：

$$\Delta f_O = \frac{f_C}{2^N} \qquad (2-3)$$

最低输出频率：

$$f_{O\min} = \frac{f_C}{2^N} \qquad (2-4)$$

最高输出频率：

$$f_{O\max} \le \frac{f_c}{2} \qquad \text{一般} f_{O\max} = 0.4 f_c \qquad (2-5)$$

从上面的讨论中可以看出 DDS 具有：输出频带宽，频率分辨率高，输出频率转换速度快，频率跳变时输出相位连续等优点。由于 DDS 采用全数字技术，其合成机理在根本上有别于传统的合成技术，从而为频率合成的设计开辟了广阔的前景，但是它的频谱是不纯净的，研究 DDS 的频谱对于选择芯片进行电路的正确设

计是很重要的。

2.3 DDS 的频谱分析[6][7][8][9][10][11][12]

1. 理想 DDS 的数学模型

理想 DDS 是 DDS 在理想情况下的数学模型，满足下列三个条件:

- 不考虑相位舍位，即 N=A
- 不考虑正弦波幅度量化的误差，即 D=∞
- 假定 DAC 是完全理想的

这样 DDS 就等效为一个理想的采样－保持电路，其中 D/A 转换器之前的部分（图 2－3 中）相当于一个采样周期为$T_c = 1/f_c$的理想采样器，DAC 相当于一个时宽为T_c的理想保持电路，对采样数据实现阶梯方式重构:

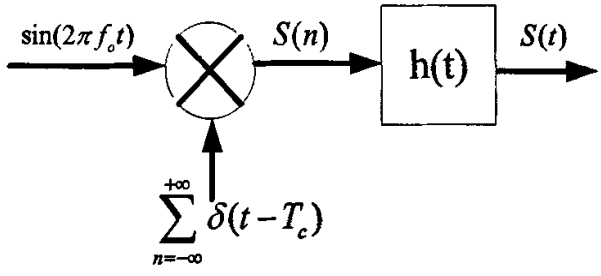


图 2－3 理想 DDS 的等效结构

2.理想 DDS 输出的频谱

由理想 DDS 的等效结构，理想采样序列

$$S(n) = \sin(2\pi f_o n T_c) = \sin[2\pi \frac{K f_c}{2^N} n T_c] = \sin[2\pi \frac{K}{2^N} n] \qquad (2-6)$$

由式 2－6 知采样序列 S(n)是周期为$\frac{2^N}{Gcd(2^N, K)}$的周期序列，Gcd (x, y)表示取最大公约数，而 S(n)的频谱是一些离散谱线，在这些离散谱线中，幅度不为 0 的只有f_o一根，故对理想的 DDS，其 S(n)在(0，$f_c/2$)范围内没有杂散。

$$S(t)=S(n)*h(t)=[\sin(2\pi f_O t)\bullet\sum_{n=-\infty}^{+\infty}\delta(t-nT_C)]*h(t) \qquad (2-7)$$

$$h(t)=\begin{cases}1 & (0\le t\le T_c)\\ 0 & (\text{其他})\end{cases}$$

对 S(t)作傅氏变换得到理想 DDS 的理想频谱函数：

$$S(\omega)=\pi\sum_{l=-\infty}^{+\infty}S_a(\frac{lf_c-f_0}{f_c}\pi)\bullet\exp(-j\frac{lf_c-f_0}{f_c}\pi)\bullet\delta(\omega+2\pi f_0-2\pi lf_c)$$
$$+\pi\sum_{l=-\infty}^{+\infty}S_a(\frac{lf_c+f_0}{f_c}\pi)\bullet\exp(-j\frac{lf_c+f_0}{f_c}\pi)\bullet\delta(\omega-2\pi f_0-2\pi lf_c) \qquad (2-8)$$

式中 $f_o=\dfrac{Kf_c}{2^N}$，$S_a(x)=\dfrac{\sin(x)}{x}$

S(t)为保持器的冲击响应，从 2-8 式可看出，理想 DAC 所完成的阶梯重构只改变了输出频谱的幅度和相位而未增加新的频率点。这样 S(n)的频谱结构即代表了 DDS 输出的频谱分布。据此可知，理想 DDS 的输出只在 $f=lf_c\pm f_0$ 处存在离散谱线，当 l=0 时，得到的就是主谱频率 f_0，下面以采样时钟 200M 和基本输出频率 40M 为例：

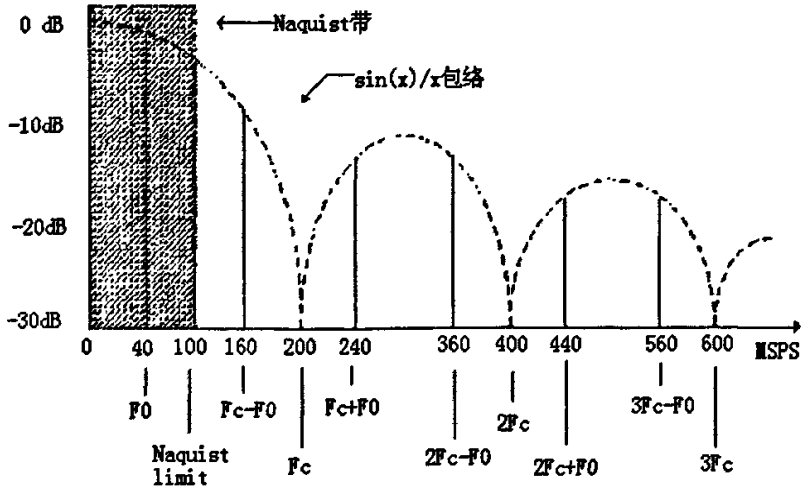


图 2—4 f_c=200MHz，f_o=40MHz 时理想 DDS 的频谱结构

由上图可见，$\omega_o = 2\pi f_o$，$\omega_c = 2\pi f_c$，$T_c = 1/f_c$，ω_o是输出频率谱线，而其他$\omega_c - \omega_o$，$\omega_c + \omega_o$，$2\omega_c - \omega_o$，$2\omega_c + \omega_o$，$3\omega_c - \omega_o$……等为副波谱线，这是通过取样后的正弦波来合成所需频率的必然结果，是由 DDS 的工作原理所决定的，因此在 D/A 后面必须设置一个低通滤波器，适当设计低通滤波器的截止频率，使其落在ω_o和$\omega_c - \omega_o$之间，即可有效地消除副波频率，而且从图中可看出，ω_o和$\omega_c - \omega_o$相差越大越好，从前面的推导过程可以看出，理想 DDS 杂散的主要来源是系统中的取样函数$\delta(t - nT_c)$，也就是说由参考时钟f_c引起的，这也说明了杂散是 DDS 固有的，并且参考时钟频率越高，即取样点越多，杂散的抑制相对就越大。

3.实际 DDS 输出的杂散分析

实际 DDS 并不满足理想 DDS 的 3 个条件，因而实际 DDS 将产生杂散信号，DDS 的杂散信号有 3 个来源，分别是：

- 相位截断误差$\varepsilon_p(n)$：在实际 DDS 电路中，为了取得高的频率分辨率，通常相位累加器的位数 N 取得很大，如 N=24,32,48 等，但受体积和成本的限制，ROM 的容量却远小于2^N，因此寻址 ROM 时，累加器输出的相位序列的低 B 位就被舍去，而只用其输出的高 A (A=N-B)位去寻址，这样就不可避免地引入误差。
- 幅度量化误差$\varepsilon_D(n)$：由于 ROM 中存储着正弦波样点的幅值编码和 D/A 有限字长，这样就不可避免地引入了幅度量化误差。
- 由 DAC 非线性引起的误差$\varepsilon_{DA}(n)$：实际的 DAC 存在非线性因素，主要有微分非线性，积分非线性，DAC 转换过程的尖峰电流，转换速率受限等，据此可以构造出 DDS 杂散(误差)来源模型，如下图所示：

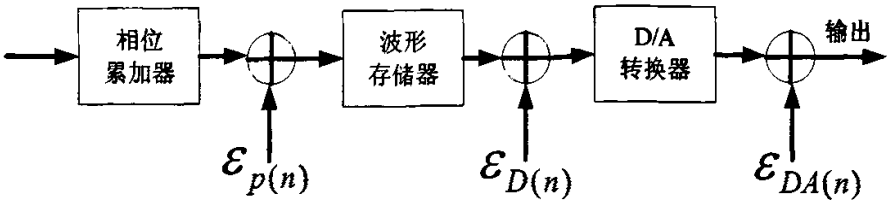


图 2—5 实际 DDS 的杂散(误差)模型

4.DDS 杂散的分析

(1)相位截断误差 $\varepsilon_{p(n)}$：若 DDS 相位累加器的字长为 N，频率控制字为 K,则相位累加器在 $t=nT_c$（T_c 为时钟周期，$T_C=1/f_c$）时刻，输出相位序列：

$$\varphi(n)=nk \bmod 2^N \qquad n=1,2,.. \tag{2—9}$$

这里 mod 表示模除运算，对任意两个整数 p, q, mod 运算定义为：

$$p \bmod q \triangleq p-\mathrm{int}(\frac{p}{q})\times q \tag{2—10}$$

上式中，int(.)表示取整运算。

在实际 DDS 中相位舍位后累加器输出的相位序列为：

$$\varphi_{\varepsilon}(n)=nk \bmod 2^N - nk \bmod 2^B \tag{2—11}$$

这样，由于相位舍位所产生的相位误差序列为：

$$\varepsilon_p(n)=\varphi(n)-\varphi_{\varepsilon}(n)=nk \bmod 2^B \tag{2—12}$$

由此可知 $\varepsilon_p(n)$ 是以 $\frac{2^B}{Gcd(2^B,K)}$ 为周期的序列，在频域上以 f_c 为周期，在

(0, $f_c/2$)内由 $\lambda=\frac{2^{B-1}}{Gcd(2^B,K)}$ 根离散谱线组成。从上式可以认为 $\varepsilon_p(n)$ 为 $\varepsilon_p(t)$ 的采样，$\varepsilon_p(t)$ 是幅值为 2^B,周期为 $\frac{2^B}{K}$ 的锯齿波。

(2)幅度量化误差 $\varepsilon_D(n)$

当 ROM 采用 D 位二进制数保存正弦函数值时，量化误差为:

$$\varepsilon_{D(n)}=\sin[2\pi\cdot\frac{K}{2^N}\cdot n]-\frac{1}{2^D}\cdot\mathrm{int}\{2^D\cdot\sin[2\pi\cdot\frac{2^B}{2^N}\cdot\mathrm{int}(\frac{K}{2^N}\cdot n)]\} \tag{2—13}$$

其中，int(.)表示取整运算。显然 $\varepsilon_{D(n)}$ 与 S(n)有相同的序列周期 $\frac{2^N}{Gcd(2^N,K)}$，因此

幅度量化误差在频谱中没有引入新的杂散成分，而是表现为均匀的噪声基底。通常在一个周期内 $\varepsilon_{D(n)}$ 被认为是在$(-2^{-D},2^{-D})$间均匀分布的噪声，则由量化引起的信噪比为：

$$SQR=1.76+6.02D(dB) \qquad (2-14)$$

由式2－14可见，量化位数D每增加一位，则SNR将提高6dB，所以在可能的条件下应尽可能采用存储位数高的存储器和高分辨率的DAC，但也不能盲目地追求高位数，因为当D增加到一定的位数时，杂散不再因之而改变。

(3)DAC的非线性误差 $\varepsilon_{DA(n)}$

在高频、超高频DDS系统中，DAC的非线性逐步成为DDS输出杂散的主要来源之一，DAC的非线性特性对DDS输出谱的影响表现为产生输出频率的谐波分量及这些谐波分量的镜像分量，即含有频率为：

$$F_S = lf_O + nf_S \qquad (l=0, \pm1, \pm2, \cdots, n=2, 3, \cdots) \qquad (2-15)$$

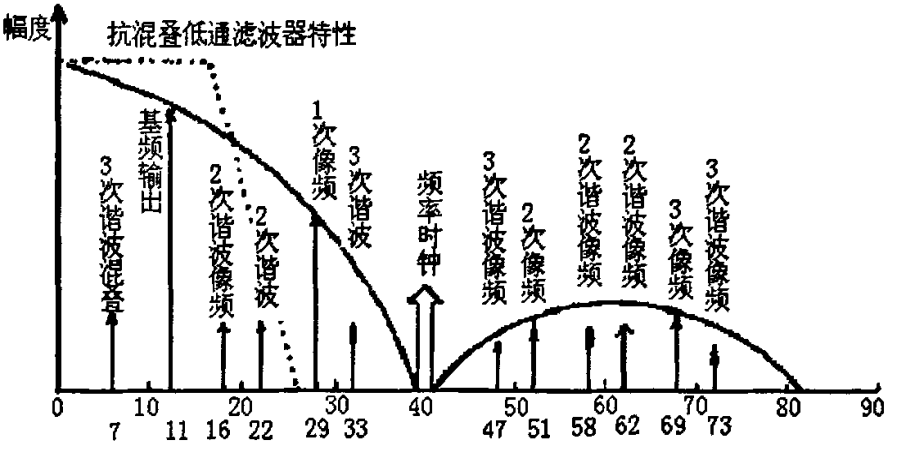


图2－6 DAC对DDS输出谱的影响

杂散电平由DAC的性能决定，当输出频率离 f_c/n (n=3,4,…)很近时，$f_c-(n-1)f_o$ 及 $(n+1)f_c-f_o$ 将离 f_o 很近，成为主谱周围幅度较大的杂散分量，实际应用中应当避免。

在现代VLSI技术条件下，通过加大ROM容量及数据位数，DDS由相位截断和数据量化引起的杂散噪声已经可以很容易做到-70dB以下的理论值，但是工作在

高速的 DDS 输出谱中仍然存在-40dB 左右的少数杂散谱线，这是由 DAC 的非理想特性引起的，因此 DAC 才是目前影响 DDS 频谱质量的决定因素，除了 DAC 有限分辨位数之外，DAC 的瞬间毛刺，DAC 的非线性，数字噪声馈通以及时钟的泄漏都是导致频谱劣化的因素，它们为 DDS 输出频谱增加了背景噪声和杂散，所以 DAC 的性能对于整个系统是很关键的。

2.4 主要技术指标

1.垂直分辨率(幅度分辨率--位数)

垂直分辨率是指幅度的分辨率由数模转换器的分辨率决定，任意波形发生器的特性很大程度上取决于其数模转换器的性能。对数摸转换器的速度和分辨率要权衡折衷，采用快速的数模转换器，其分辨率则较低。在 20M 取样点/秒的范围内，目前通常采用的是 10 位或 12 位的数模转换器，高于 20M 的采样率则选用 14 位或 16 位的数模转换器。通常，较高的垂直分辨率意味着再现的波形中量化误差较小，从而减少了失真。如果使用数字存储示波器捕捉到的波形，则应与数字存储示波器的分辨率一致。

2.水平分辨率(波形存储深度--字数/通道)

水平分辨率是波形存储器中可利用点的总数，大致可以表示任意波形发生器所能产生波形的复杂程度，在每秒 20M 个取样点时，要将每个存储单元的内容都送给数模转换器，并让数模转换器快速转换成相应的模拟量电压，只容许有 50ns 的时间。为满足这类需求，通常采用快速的 SRAM。水平分辨率点数越多，意味着再现的波形的失真越小。在低频应用时，因为要求在较长的一段时间内维持较高的取样速率，所以需要较长的波形存储器。

3.取样速率(采样速率--样点数/秒)

采样速率为准确再现信号所需的单位时间采样点数，其中最高取样速率决定了波形中最高的频率成分即任意波形发生器的输出带宽，Nyquist 定理指出:可还原的最高频率不大于二分之一的取样速率，在该频率点上，每周期只要有两个取样点就确定了原波形，实际上可采用的最高频率要比 Nyquist 频率低得多，具体取决于可容忍的失真程度。

2.5 本章小结

本章主要阐述了 DDS 的工作原理、特点和技术指标，也对 DDS 的频谱作了简要的分析说明。此章内容为任意波形发生器（AWG）的系统设计做必要的铺垫，为具体的设计工作打下理论基础。

第三章 任意波形发生器的系统设计

3.1 技术指标

1.波形

标准波形：正弦波，方波，三角波

任意波形：

- 波形深度：1--65536 个点
- 幅度分辨率：16 位
- 采样率：100KSPS 左右

2.输出波形的要求

要求输出波形频率在 3HZ～5KHZ 范围内可调，输出电压的峰峰最大值大于 25V，频率分辨率根据波形复杂程度不同尽可能的高。

本课题设计的任意波形发生器，不仅要求可以生成方波、三角波、正弦波等标准波形，而且还要可以生成用户所需要的任意波形，同时输出波形的频率和幅度均可编程控制。

3.2 系统整体设计思路

整个系统的设计思想是采用 VC++编写人机界面窗口，在 VC++编程实现的人机交互界面上按下鼠标左键可绘制任意波形，然后由相应程序对绘制的波形进行采样，采样数据会保存到数组文件，最后这些数据被传送给 DSP，DSP 经过整理后送到波形 RAM 中存储起来。

DSP 是整个系统的核心部分，对整个电路的工作进行控制。具体的控制程序在电脑上编写完成后经过 DSP 仿真器下载到程序存储器中。本课题中用的是扩展 FLASH，在 DSP 上电工作后，利用 DSP 提供的 boot 机制，再将程序下载到 DSP RAM 中运行。

波形数据在 DSP 的控制下从波形存储器中读出，送到 D/A 转换器中，经过数

模转换输出模拟的阶梯信号，再经过后面的低通滤波器滤波就得到圆滑的波形输出，再接下来是可编程增益运放电路可以对波形幅度进行编程控制，最终得到用户所需要的波形。其中涉及到的波形的频率控制和幅度控制将会在后续章节作清楚的说明介绍。

系统整体框架结构图如下图 3－1：

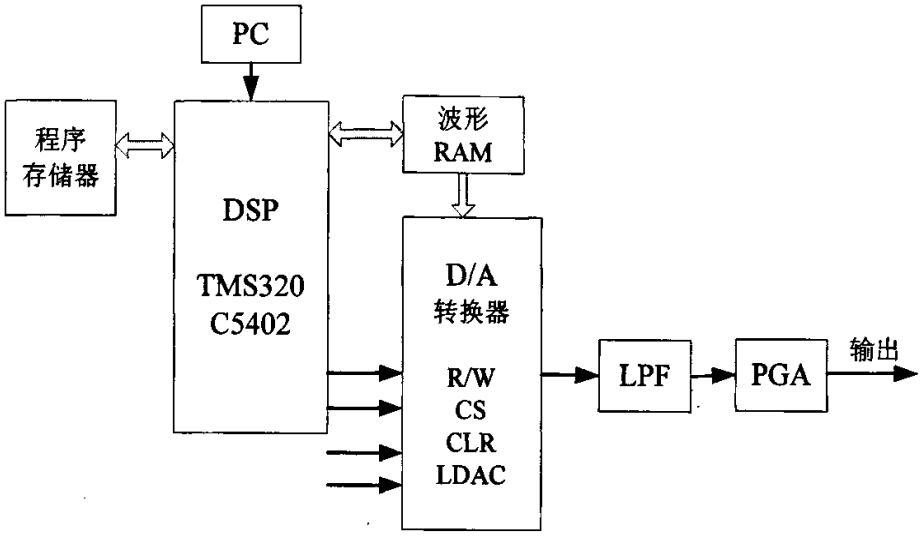


图 3－1 整体框架图

根据上述的设计思想，系统大概可以分成以下三个部分：

1.波形产生

2.系统控制

3.D/A 转换及滤波输出

下面针对这三个部分作系统级的阐述说明：

（1）波形的产生是由按下鼠标左键来绘制，在编程实现的 VC 人机窗口界面内按下鼠标左键来绘制任意波形，并对它采样、保存，目的是保存波形相关数据，这些数据会存储到一个系统生成的.dat 文件中，它们也将是要从电脑传送到数据存储器中的数据。数据最终被传送到 DSP，其具体传送方式以及程序设计将在后续章节中作介绍。

VC++实现的波形产生部分的程序和窗口界面也将在第五章软件章节中给出详细的介绍，这里就不详谈了。绘制任意波形的界面窗口如下图 3－2 所示：

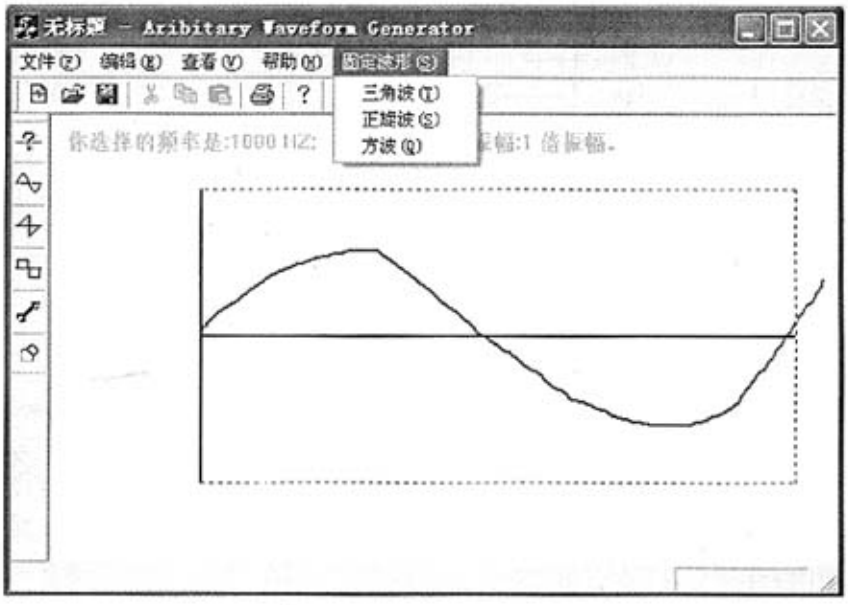


图 3—2 波形绘制窗口图

（2）系统控制部分就是对整个电路的工作进行控制，也是整个电路正常运行的关键。其核心就是 DSP 芯片 TMS320VC5402，它不但控制对波形数据的接受、传送，还控制电路中其他器件的工作状态。它对数据存储器、程序存储器、还有 D/A 转换器都进行协调。

首先是波形数据的获得，即 DSP 从 PC 接受波形数据，然后经过内部整理传送到波形存储器中保存起来，设计中用的是扩展数据存储器--IS61LV6416L。

其次你在电脑上编写的控制程序也要通过 DSP 仿真器下载到程序存储器，设计中用的是扩展 FLASH--AT49LV1024A。

另外 DSP 还控制 D/A 转换芯片 AD7846 的片选和读写引脚，这样就可以软件控制每次对 D/A 进行写数据时，都已经选通了 D/A。设计中用 DSP 的 XF 端子对 D/A 的片选/CS 控制，通过编程控制 DSP 的 XF 端子的高低电平，就可以做到对 D/A 转换芯片的选通控制。也可以控制读写的频率，从而对波形的输出频率进行控制。这部分将在第五章的第一节中作详细介绍。

以上提到的器件的具体连接和应用会在接下来的几节中芯片介绍和具体应用中为你作详细的说明。TMS320C5402 的控制部分结构框图如图 3—3：

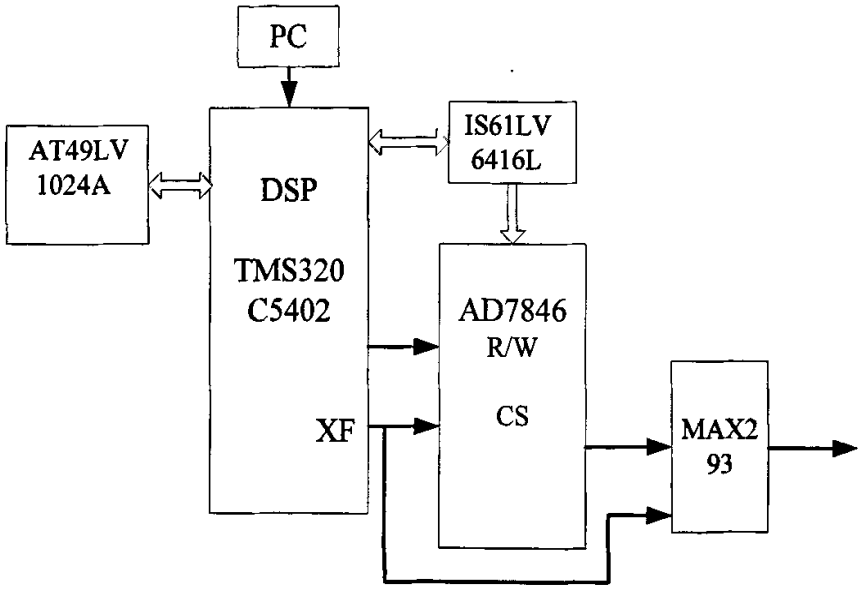


图 3－3 DSP 核心控制框图

（3）D/A 转换及其滤波输出部分，是指电路中采用 16 位高精度的数模转换芯片 AD7846 对数据存储器传送过来的数字信号进行转换，转换出模拟的阶梯信号，后面的高阶低通滤波电路对其滤波就得到平滑的模拟波形。设计中采用了 8 阶椭圆可调低通开关电容滤波器 MAX293。接下来是幅度增益控制部分，采用两块可编程增益块 PGA205 和 PGA103 串联实现对波形幅度控制，可以实现高达 800 倍的幅度信号。下面给出波形从 DA 出来经过滤波、增益控制、最后输出的简单示意框图：

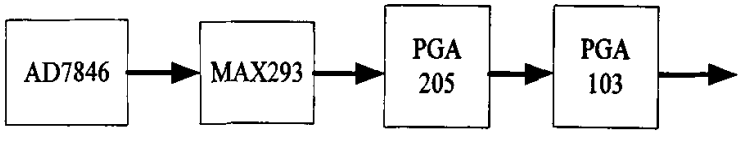


图 3－4 输出部分框图

3.3 电路硬件部分的设计

针对上一节中整个系统结构的介绍，本节主要对所设计电路中涉及到的主要芯片和核心部分电路做详细的说明和具体的介绍。下面把电路核心部分的框架结构图重画如下：

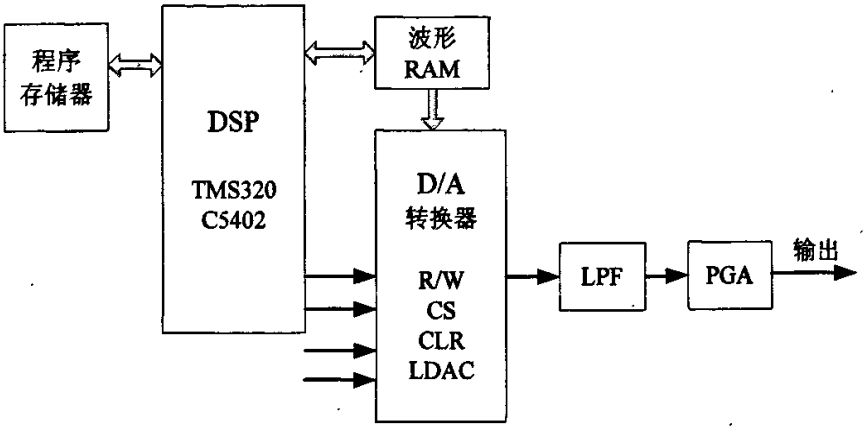


图 3—5 核心电路框图

电路的核心部分主要芯片有 DSP 芯片 TMS320C5402、D/A 转换芯片 AD7846、外扩数据 RAM IS61LV6416L、扩展 FLASH AT49LV1024A、低通滤波芯片 MAX293 以及幅度增益控制芯片 PGA205 和 PGA103。下面的章节将对它们作详细的器件说明和应用介绍。

3.3.1 DSP 芯片

DSP 采用了 TI 公司的 TMS320C5402,它有一组程序总线和三组数据总线,高度并行的算术逻辑单元 ALU、专用硬件逻辑片内存储器、增强型 HPI 口和高达 100MHz 的 CPU 频率,可以在一个周期里完成两个读和一个写操作.其他的一些设计中有关 DSP 的应用和应用中的具体连接都将在第四章中作详细的介绍，这里不再赘述了。下面讨论下 DSP 如何从 PC 获得波形数据，数据的获取可以有两种方法：一种通过仿真器下载到 DSP；另一种是通过串口把数据传送到 DSP。下面分别对它们进行介绍：

（1）用 C 编写一个数据转换程序，把十进制数据文件 waveform.dat 中的数据取出，并形成一个数组文件的形式，转换为数组头文件 waveform.h。则在 CCS 环境下，FLASH 的加载主程序包含此头文件，编译链接后通过仿真器下载到 DSP 中，运行 DSP，用户程序就被加载 FLASH 中。转换程序见附录 2。

（2）串口传送方式。本设计中为了实现与上位机之间的通信，设计中留了一个串行接口 RS-232。波形数据也可以通过串口来传送给 DSP，有同步串口通信和异步串口通信两种。若采用异步串口通信的话，PC 与 DSP 需要约定一个波特率，双方

按照这个约定的波特率收发数据，并依靠数据的起始位和数据长度通信，没有时钟同步信号，是一种简单易用的通信方式。

数据被一位一位的传送到DSP，DSP经过整理，每16位为一组，传送到数据RAM保存起来。这样就可以把波形数据一步步的传送到RAM中，实现波形RAM从PC上的数据获取。串口通信中涉及到一个电平转换芯片MAX3224，由它来进行逻辑电平转换，从而可以让PC和DSP进行数据通信。

MAX3224是EIA/TIA-232通信接口，有自动关断和唤醒、高数据速率以及增强的ESD保护等特点。作为RS-232发送器，能把CMOS逻辑电平转换为5.0V的EIA/TIA-232电平。数据传输速率可以保证250Kbps，还可以并联来驱动更多的接收器；作为RS-232接收器，能把RS-232信号转换为CMOS逻辑输出电平。它要求转换输出总是保持有效。DSP与MAX3224的连接如下图3－6所示：

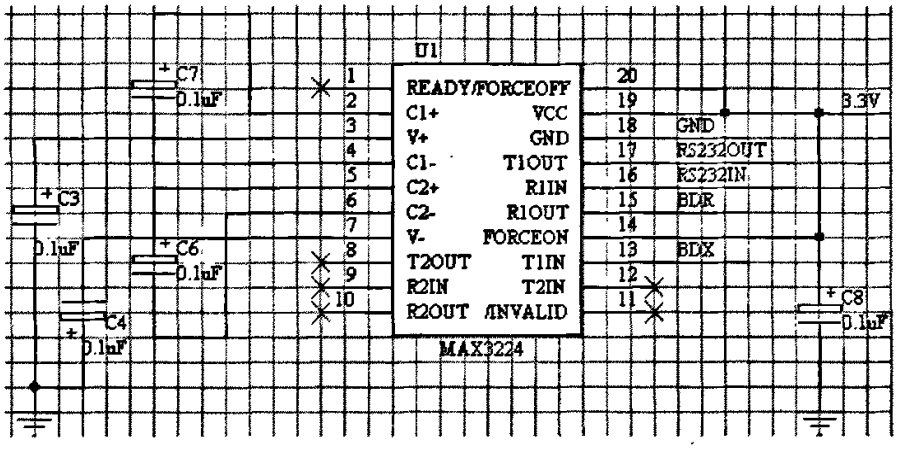


图3－6 MAX3224的连接图

3.3.2 D/A芯片[13][14]

1.AD7846的特点

AD7846是美国AD(ANALOG DEVICES)公司推出的一种采用LC^2MOS工艺制造的16位电压输出的D/A转换芯片，具有以下特点[15]：

- 16位精度，在全温度范围内具有16位精度，适用于闭环控制系统；
- 数据回读功能，对DAC寄存器中数据可以进行读取，降低了自动测试设备系统中软件设计的难度；

• 低功耗，典型功耗仅为 100mW；

• 接口电路简单，有单极性(0～5V，0～10V 输出范围)、双极性(±5V，±10V 输出范围)两种输出方式，无需额外电路。

AD7846 采用了分段式的体系结构，DAC 数据锁存中高四位数据用于选择 16 段电阻组成的电阻分压网络的输出电压，输出电压经放大器缓冲后输入到一个 12 位的 DAC，该 12 位 DAC 提供 12 位的转换精度，这种结构最终将提供 16 位的单调性。

除高精度外，AD7846 具有全面的微处理器接口，包含 16 位数据总线及多种控制接口，允许程序读出缓冲中的数据。LDAC 信号允许多 DAC 系统中的 DAC 芯片同时更新转换数据，CLR 信号也可根据 R/W 信号产生不同的清零结果使 DAC 在单极性或双极性输出时都初始化，并输出 0V 的电压。

2. AD7846 的基本结构和引脚功能

AD7846 的内部结构如图示[15]，其内部主要包括三个部分：控制逻辑电路、数模转换电路、4 到 16 段开关矩阵。

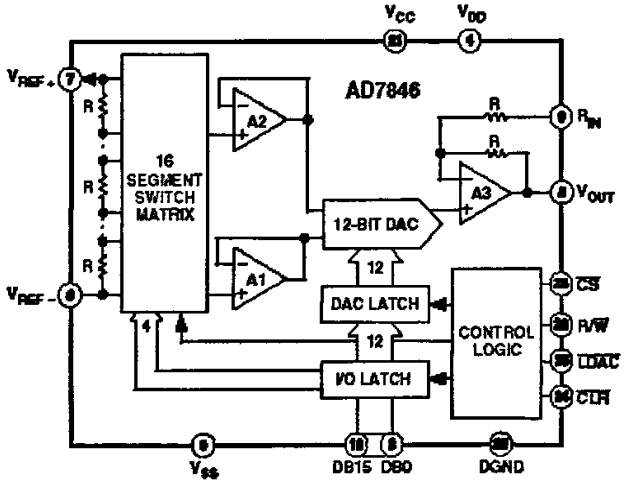


图 3－7 AD7846 内部电路

AD7846 数模转换电路的基本控制原理为：逻辑控制电路通过四个控制信号 CS、R/W、LDAC、CLR 来选择并进行相应的存储、初始化操作，锁存转换数据中的低 12 位数据。高 4 位数据控制 16 段开关网络，完成参考电压的设置，从而使输出的电压具有 16 位精度。其中，R/W 输入对转换的数据寄存器进行读写操作

控制，实现数据的回读处理。通过 CLR 与 R/W 控制信号的配合，可以对初始化输出值进行配置：在写状态下清零，输出数据为 000...000 的转换值；在读状态下清零，输出数据为 100...000。这个特性可以方便实现单双极性工作状态下 0V 电压输出。

D/A 采用了低功耗数模转换器 AD7846, 实现了高速同步数模转换. AD7846 有单极性(0～5V，0～10V 输出范围)、双极性(±5V，±10V 输出范围)两种工作方式。单极性工作时，需将 VREF+接设计所需的正参考电压，而将 VREF-接地；双极性工作则需将 VREF+、VREF-分别接设计所需的正负参考电压。由于 AD7846 有片内集成运放，如果将 RIN 脚接地,其输出范围为 2VREF-～2VREF+；如果将 RIN 脚与 VOUT 脚短接，则其输出范围为 VREF-～VREF+。具体如下表[15]所示：

Output Range	V_{REF+}	V_{REF-}	R_{IN}
0 V to +5 V	+5 V	0 V	V_{OUT}
0 V to +10 V	+5 V	0 V	0 V
+5 V to –5 V	+5 V	–5 V	V_{OUT}
+5 V to –5 V	+5 V	0 V	+5 V
+10 V to –10 V	+5 V	–5 V	0 V

表 3.1 D/A 输出电压范围表

AD7846 有四个数字逻辑控制引脚 CS、R/W、LDAC、CLR,它们不同的组合可控制 AD7846 工作的模式。系统设计中合理利用 DSP 的相关引脚来对 CS、R/W 进行软件编程控制，从而做到对数/模转换器 AD7846 的控制，使其工作于我们需要的最佳状态。表 3.2 给出 AD7846 的控制逻辑真值表：

表 3.2 D/A 逻辑控制表

CS	R/W	LDAC	CLR	功　能
1	X	X	X	使 DAC 的 I/O 锁存器呈高阻态
0	0	X	X	数据(DB15~DB0)装入 I/O 锁存器
0	1	X	X	I/O 锁存器中的数据输出到数据线上
X	X	0	1	I/O 锁存器中的数据装入 DAC 锁存器
X	0	X	0	DAC 锁存器装入数据 000...000
X	1	X	0	DAC 锁存器装入数据 100...000

AD7846 工作于±10V 的双极性模式，AD588 为其提供精密的±5V 电压，送

到 AD7846 的 VREF+与 VREF-端之间。参考电路连接如下图 3－8[15]所示：

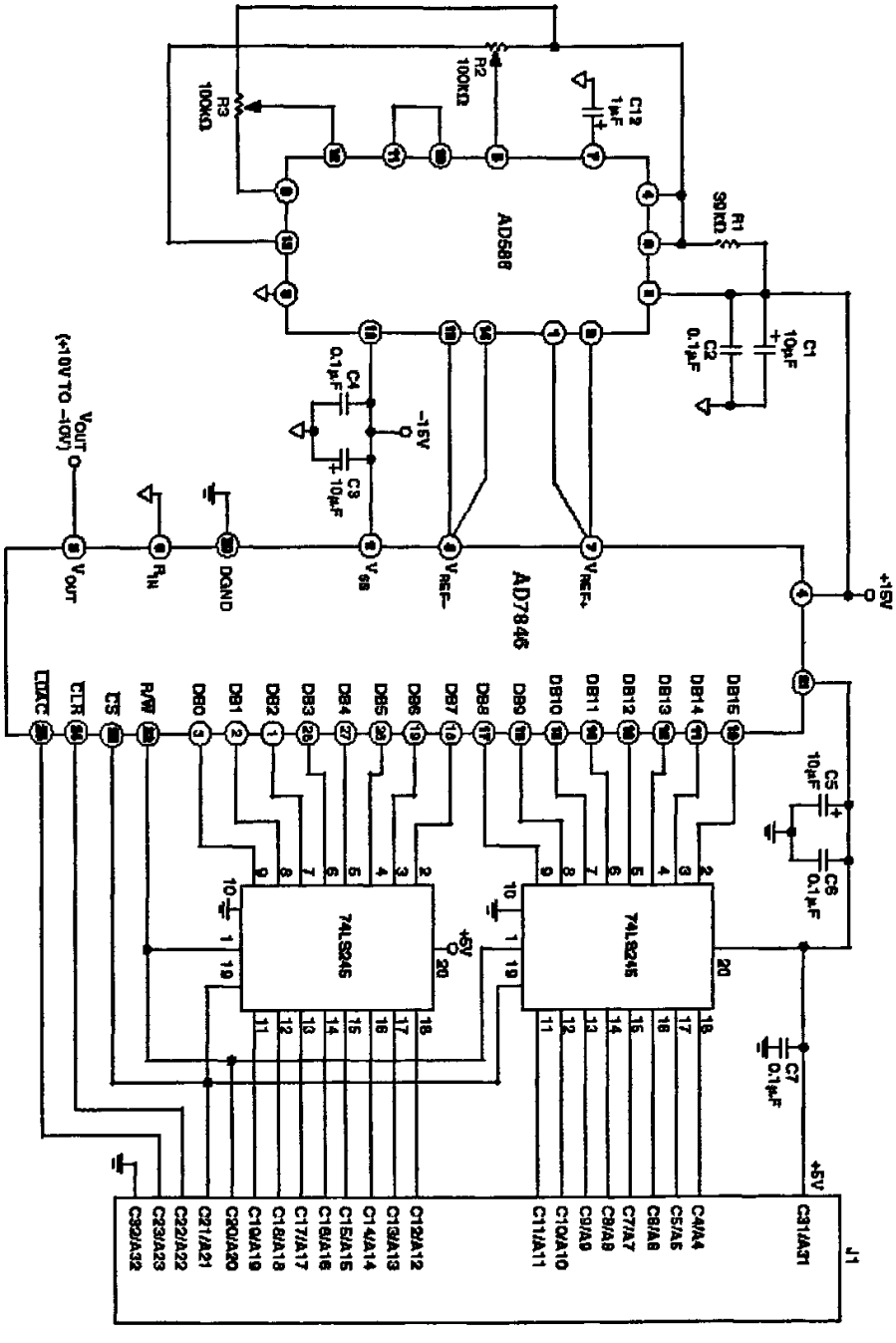


图 3－8 AD588 与 AD7846 的连接图

3.3.3 低通滤波器（LPF）

滤波器的设计是系统设计中的一个关键环节，是本课题的一个重要的部分，也是设计中的一个难点。

滤波器按不同的频域或时域特性要求，可分为巴特沃斯（Butterworth）型，契比雪夫(Chebyshev)型，贝赛尔(Bessel)型，椭圆型等标准型。相同的电路，通过选取不同的 R 和 C 参数可以实现不同的类型。其中，巴特沃斯型滤波器具有最平坦的通带幅频特性；契比雪夫型特点是通带内增益有波动，但这种滤波器的通带边界下降快；贝赛尔型通带边界下降较为缓慢，其相频特性接近线性；椭圆型的滤波特性很好，但模拟电路复杂，元件选择较为困难，实现难度大。

低通滤波器的设计部分采用的是开关电容滤波芯片 MAX293，因为查阅了一些有关任意波形发生器的相关资料，发现大多采用的是设计滤波电路来实现对 D/A 输出的后滤波，这样的话就存在几个问题：

首先是一个滤波电路只能滤一定频率的波形，不能随着波形输出频率的改变而改变，要满足任意波形输出的需要就必须设计很多这样的滤波电路，这样以来，电路将非常复杂。其次是任意波形的输出频率较低，这样滤波电路中的电容，电感的等器件的容值和感值都非常大，这样一来是电感的感值越大，一般体积也较大，整个电路的面积就很大。二来量值越大，也就相对越离散，可能市面上能买到的电感与你设计的电感值有很大偏差，这样设计出来的电路，滤波效果也受影响。最后，课题中采用的滤波器的拐角频率可以编程控制，并且完全适合课题的设计要求范围，是很好的一个选择。

MAX293 是 8 阶、椭圆、低通、开关电容滤波器。其拐角频率范围 0.1HZ～25KHZ。不需要外部电阻和电容。其过渡比为 1.5 时可以提供急剧滚降和－80db 的衰减带。有内部和外部两种时钟方式。时钟与拐角频率比为 100：1。

下面给出了它的引脚连接图 3－9 和频率响应图 3－10[16]：

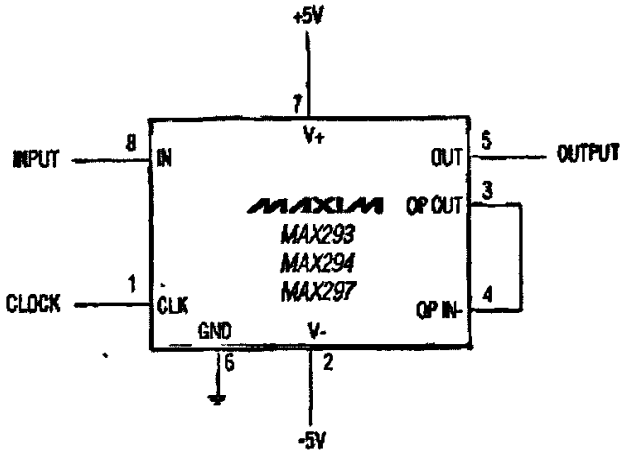


图 3－9 MAX293 引脚连接图

考虑到 AD7846 的数模转换时间为 6us～9us，典型时间是 6.5us。所以每个数据量经过转换后都要维持一个时间 t>6.5us，假设 t＝7us。这样的话，采样点数改变，可以改变输出波形的频率。

如果采样点数为 100 个，则波形频率 f＝1/(7us*100)=1.4khz；如果采样点数为 400 个，则频率为 0.35K 左右；如果采样点数为 50 个，则频率为 2.8K；同样，采样 10 个点，可以达到 14K；采样点数越少，频率就越高，不过波形的精度也就相对越低。

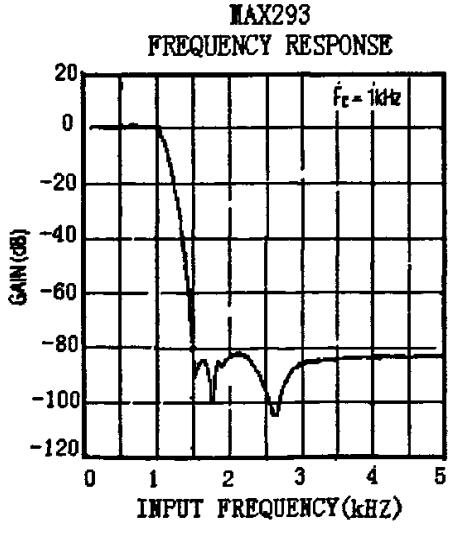


图 3－10 MAX293 的频率响应图

MAX293 开关电容滤波器可以通过 clk 端来控制滤波器的拐角频率。时钟与拐角频率比为 100：1。这个端子由 DSP 的 XF 端子来控制，通过编程可以实现周期为 8us 的周期信号来控制 MAX293。这样的话，周期信号的频率就是 125khz，它作为滤波器的 clk 的话，就是说滤波器的拐角频率就是 1.25khz。

如果每个数据量转换后的维持时间为 10us，则输出波形频率 1khz。同理，如果每个数据量转换后的维持时间大于 10us，则输出波形频率小于 1khz。

对于 1khz 的相对平滑、变化不是很急剧的波形信号来说，上述考虑还是挺合适的。而对于那些波形变化很剧烈的信号来说，则需要作适当的调整，因为它要求的滤波频率相对高些。

方案也是多种，可以在 DA 的选通端子和 MAX293 的 clk 之前分别加入分频器或者计数器，这样就可以对滤波器的拐角频率进行二次控制，一方面通过编程 XF 端子来实现对时钟信号的周期控制，另外一方面则是通过分频器或者计数器来对周期信号的频率进行分频，得到 MAX293 所需要的周期时钟信号。

下面举个例子来说明这个问题：

如果需要的滤波器拐角频率是 5Khz，则 MAX293 的时钟端子频率应该为 500Khz，可以设定 XF 的端子信号周期为 1us，则频率为 1Mhz，所以分频器 B 应该是 2 分频。对于 DA 来说，则考虑到 100K 的采样速率，需要分频器 A 是 10 分频。同理可知道，如果设定 XF 端子的信号周期是 2us 的话，则分频器 B 不需要，而分频器 A 需要 5 分频来与 DA 搭配。总之可以通过适当调整分频器 A 和分频器 B 来实现对滤波器的具体控制，达到需要的滤波频率。

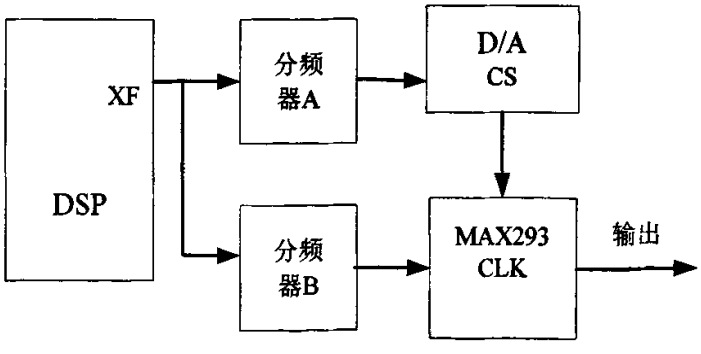


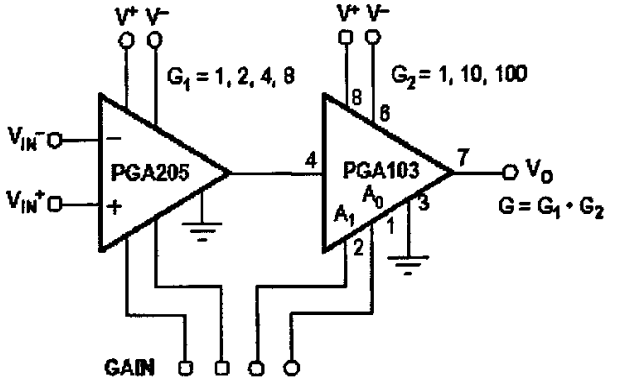
图 3—11 滤波器 MAX293 部分框图

3.3.4 幅度增益部分

可编程增益放大器采用的是美国 BB 公司的具有低增益误差的 PGA205，它可采用 4.5～18V 的电源工作，通过与 CMOS 与 TTL 兼容的输入端来设定增益，并能提供快速的稳定时间。

AD7846 在 16bit 分辨率条件下为±1LSB，在此 DAC 后端的 PGA 达到稳定状态的建立时间必须足够快，以便与具有相同分辨率 DAC 的转换速度相匹配。此外，所选择的 PGA 还必须具有尽可能低的噪声，因为它决定系统的信噪比(SNR)。为了解决这些问题，本设计中的放大器采用 PGA205 运算放大器，它具有满足设计要求的速度、精度和快速建立时间。

后端运放电路是由可编程增益运放 PGA205 和 PGA103 串联组成。这样的话，运放电路可提供从 G=1 到 G=800 的可编程增益放大。增益输入端具体输入值见如下的可编程增益真值表[17]：



GAIN				
1	0	0	0	0
2	0	1	0	0
4	1	0	0	0
8	1	1	0	0
10	0	0	0	1
20	0	1	0	1
40	1	0	0	1
80	1	1	0	1
100	0	0	1	0
200	0	1	1	0
400	1	0	1	0
800	1	1	1	0

图 3－12 可编程增益放大的连接和真值表

数字输入端可直接与通用的 CMOS 和 TTL 逻辑元件直接接口，逻辑输入端以接地端为基准。如果数字输入端不带锁存器，逻辑输入的改变将立即选择新的增益。逻辑输入的开关时间大约是 0.5μs。

增益改变的响应时间等于开关时间加上放大器稳定到与新选择的增益相对应的新输入电压所需要的时间。对于 0.01%的精度，当 G=1 时，稳定时间为 2.5μs，当 G=16 时，稳定时间为 5μs。下面给出电路设计中的器件连接示意图 3－13：

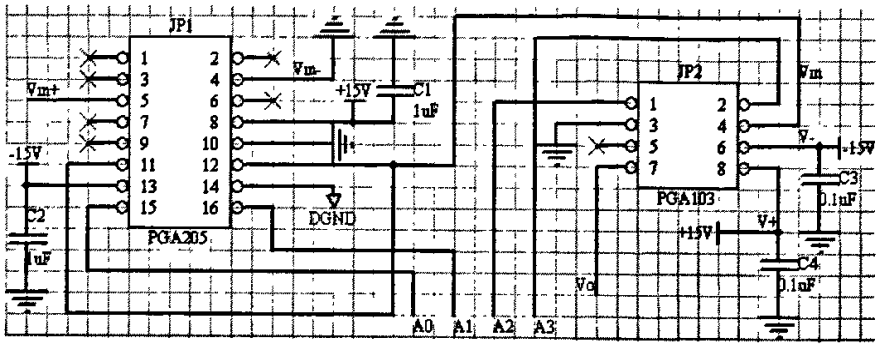


图 3－13 PGA205 与 PGA103 的连接图

3.3.5 存储器部分

1．程序存储器介绍[32]

设计中采用了外扩存储器 AT49LV1024A。它是单电源供电的 FLASH memory，64K 字的存储空间。采用 Atmel 非易失性的先进 CMOS 工艺技术。在工业温度范围之内，它能提供 45ns 快速读取时间，1.5s 的快速擦除周期，而功耗只有 72mw。它能 10000 次擦写，10×14mm VSOP 封装。功能框图和引脚介绍图如下：

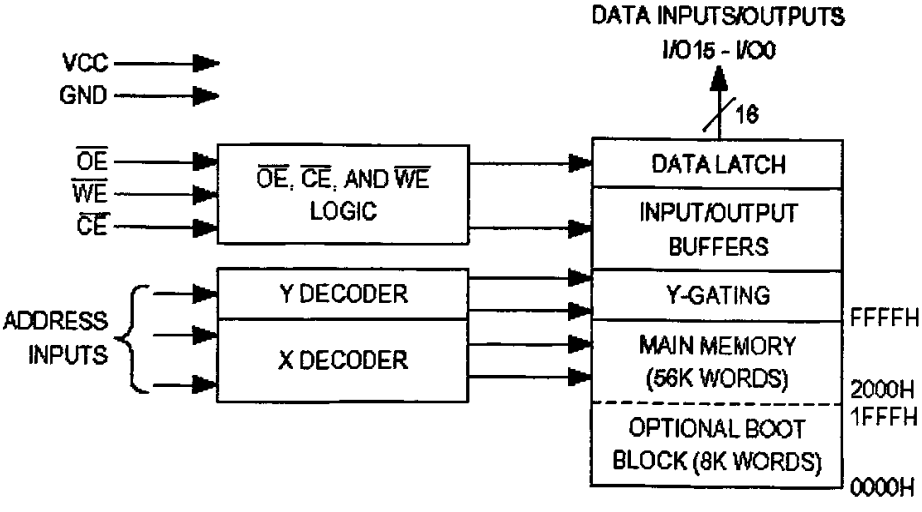
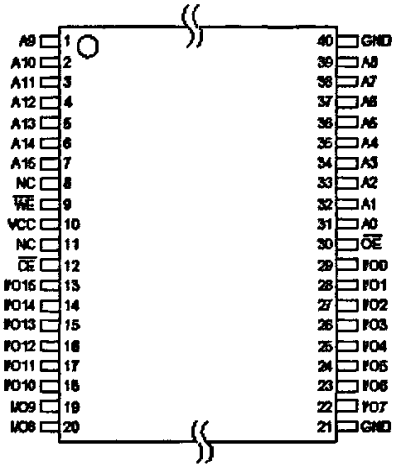


图 3－14 AT49LV1024A 功能框图

Pin Name	Function
A0 - A15	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O15	Data Inputs/Outputs
NC	No Connect

Pin Configurations



AT49BV/LV1024A VSOP Top View

图 3－15 AT49LV1024A 的引脚和封装图

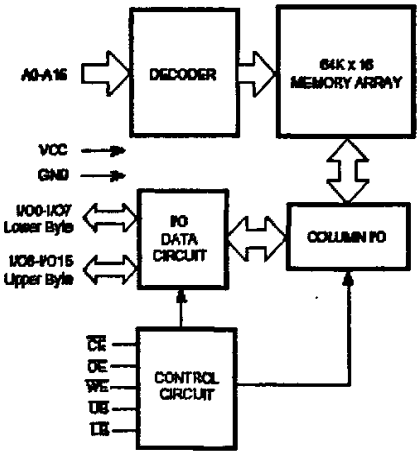
2．数据存储器的介绍[33]

本设计中采用了外扩存储器 IS61LV6416。IS61LV6416 是高速、64K 字、静态 RAM。它采用 ICSI 高性能 CMOS 技术，可以达到快至 8ns 的高速读取时间和低功耗。单一的 3.3V 电源供电，44pin TSOP-2 封装。

功耗：工作时典型值是 250mw （operation）

维持时典型值是 250uw （standby）

下图给出它的功能框图和引脚介绍：



PIN DESCRIPTIONS

A0-A15	Address Inputs
I/O0-I/O15	Data Inputs/Outputs
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
$\overline{LB}$	Lower-byte Control (I/O0-I/O7)
$\overline{UB}$	Upper-byte Control (I/O8-I/O15)
NC	No Connection
Vcc	Power
GND	Ground

图 3－16 IS61LV6416 功能框图和引脚说明图

3.4 本章小结

本章首先给出任意波形发生器的技术指标，然后根据指标给出了整个系统的设计思路，系统的分析了整个系统设计的大概框架结构。其次对核心部分的系统硬件依次作出了清晰的说明介绍，包括 MAX3224、DA 转换器 AD7846、开关电容滤波器 MAX293、幅度增益放大器 PGA205 和 PGA103、外扩 FLASH 和数据 RAM 的器件介绍以及他们在电路中的应用和连接。其重点内容是滤波器的设计部分和 DA 转换芯片的应用部分。

第四章 DSP 芯片在设计中的应用

4.1 DSP 芯片的介绍[5][18][19][20][21][22][23][24]

4.1.1 DSP 芯片发展简况[25]

自 1982 年美国 TI 公司推出第一个 DSP 芯片 TMS32010 以来，DSP 芯片有了很大的发展。DSP 芯片不仅在运算速度上有了很大的提高，而且在通用性和灵活性方面作了极大地改进。此外，DSP 芯片的成本、体积、重量和功耗也都有了很人程度的下降。随着 DSP 芯片应用领域的不断扩大，DSP 芯片已形成低、中、高三个档次：低端产品执行速度一般为 20～50MIPS，能维持适量存储和功耗，提供了较好的性能价格比，适用于仪器仪表和精密控制等；中端产品执行速度一般为 100～150MIPS，结构较为复杂，具有较高的处理速度和低的功耗，适用于无线电信设备和高速解调器等；高端产品执行速度一般为 1000 MIPS 以上，处理速度很高，产品结构多样化，适用于图像技术和智能通信基站等。对于种类繁多的 DSP 芯片，一般可按其工作的数据格式将其分为两大类：定点 DSP 芯片和浮点 DSP 芯片。定点 DSP 品种最多，处理速度为 20～2400MIPS；浮点 DSP 基本由 TI 公司和 AD 公司垄断，处理速度为 40M～1GFLOPS。

4.1.2 DSP 的选择

设计 DSP 系统时，首先要根据应用的需要选择合适的 DSP 处理器。选择 DSP 处理器主要考虑如下几个主要因素：

1. 数据类型

DSP 按照所支持的数据结构类型不同分为定点 DSP 和浮点 DSP 两大类。定点 DSP 进行算术处理时，使用的是小数点位置固定的有符号数或无符号数。浮点 DSP 进行算术操作时，使用的是带有指数的小数，小数点的位置随着具体数据的不同进行浮动。

定点 DSP 在硬件结构上比浮点 DSP 简单，具有价格低，速度快的特点，因而用得最多；而浮点 DSP 的优点是精度高，不需要进行定标和考虑有限字长效应，但其成本，功耗相对较高，速度较慢，适于对数据动态范围和精度要求高的特殊

应用。

2. 功耗

在简易设备的应用中，DSP 的功耗是选择 DSP 芯片的重要指标。大多数的 DSP 都具有功耗管理的能力。如在系统空闲时部分关闭系统时钟，关闭不需要的外设等。一些 DSP 处理器甚至提供了通过软件控制的方式改变处理器的时钟频率的能力。DSP 厂商还针对便携式应用提供了工作于 3.3v、2.5v 甚至 1.8v 电压下的 DSP 芯片。这些 DSP 的功耗比工作于 5v 下的 DSP 的功耗要小得多[25]。

此外，考虑到 DSP 数字信号处理芯片的价格和所需要的功能，本设计中采用了 TI 公司的 TMS320C5402，它是 TI(德州仪器)公司推出的高性能、低功耗的 16 位定点 DSP，是 TI 公司的第七代产品。

课题采用 DSP 芯片来作为整个系统的一个核心控制，还从另外两个方面考虑：一方面是考虑 DSP 的控制能力很强，在本课题中就得到了证实，对电路中的主要模块都进行编程控制，对系统的正常工作非常重要。另外一方面考虑到还可以充分利用 DSP 高速数据处理能力，对一些可以用算法实现或可用公式描述的波形信号进行产生、处理等。这样系统的功能将大大扩展，很有意义。

4.1.3 TMS320C5402 DSP 的体系结构及其主要特点[18][19][20][22][23][24]

当前设计中应用得比较广泛的是 TI 的 TMS320 系列 DSPs，TMS320C5402 是 TI 于 1999 年 10 月推出的性价比极高的定点数字信号处理器。其主要特点如下：

（1）144 PIN BGA，操作速率达 100MIPS

（2）哈佛结构取代传统的冯诺伊曼（Von Neumann）结构

由于 Von Neumann 结构使用单一公用的数据和指令总线，因此在高速运算时，往往在传输通道上会出现瓶颈效应。DSP 芯片内部一般采用哈佛(Harvard)结构，三条 16 位数据存储器总线和一条程序存储器总线。这种分离的程序总线和数据总线，可允许在一个机器周期内同时获取指令字(来自程序存储器)和操作数(来自数据存储器)，从而提高了执行速度。

（3）流水线技术

DSP 芯片的哈佛结构为流水线技术提供了方便。由于采用流水线技术，DSP 芯片可以单周期完成乘法累加运算，大大提高了运算速度。而 DSP 芯片的指令基

本上都是单周期指令，因此单周期指令执行时间可以作为衡量 DSP 芯片性能的一个主要指标。

（4）硬件乘法器

数字信号处理中最重要的一个基本运算是乘法累加运算，也是最主要和最耗时的运算，因此单周期的硬件乘法器是 DSP 芯片实现快速运算的保证。TMS320C5402 具有一个 17*17 乘法器，允许 16 位带/不带符号的乘法，提高了运算速度。

(5) 多种外设和接口

为了加强 DSP 芯片的通用性，DSP 芯片上增加了许多外设。可能包括的外设有：多路 DMA 通道、外部主机接口、外部存储器接口、芯片间高速链接口、外部中断、通信串口、定时器、可编程锁相环、A/D 转换器、JTAG 接口等。

（6）JTAG 接口

由于 DSP 芯片结构的复杂化、工作速度的提高、外部引脚的增多、封装面积减小而导致的引脚排列密集等原因，传统的并行仿真方式已不适合于 DSP 芯片的发展和应用开发。1991 年公布的 JTAG 接口标准满足了 IC 制造商和用户的要求，1993 年 JTAG 接口标准修订为 5 线接口。在片 JTAG 接口为 DSP 芯片的测试和仿真提供了很大的便利。

（7）程序的加载引导

加载引导是指器件在上电复位后执行一段引导程序，用于从端口(异步串口、I/O 口、主机接口)或外部 EPROM/FLASH 存储器中加载程序至高速 RAM 中运行。一般用 EPROM/FLASH 存储器存储程序，但是其访问速度较慢，而一些已有的高速 EPROM/FLASH 存储器价格昂贵且容量有限；同时高速大容量静态 RAM 价格又在不断下降，因此这种加载方式是一个有效的性价比解决方法。

TMS320C5402 有 144 个引脚，其中 20 根地址引脚，16 根数据引脚、两个多通道缓冲串行口、增强型 8 位并行主机接口、两个 16 位定时器、6 个通道的直接存储控制器、4K×16 位的片内 ROM, 16K×16 位的片内双访问 RAM 等等。

TMS320C542 有两个串行口一个是缓冲串行口 BSP(Buffered Serial Ports)，另一个是分时多任务的串行口 TDM(Time-division multiplexed serial port)。BSP 对应发送及接收的都是同步信号，格式化的同步信号以及双数据缓冲缓冲器可分别在不同传输率下同时完成。

下面给出 TMS320C5402 数字处理芯片的内部方框图：

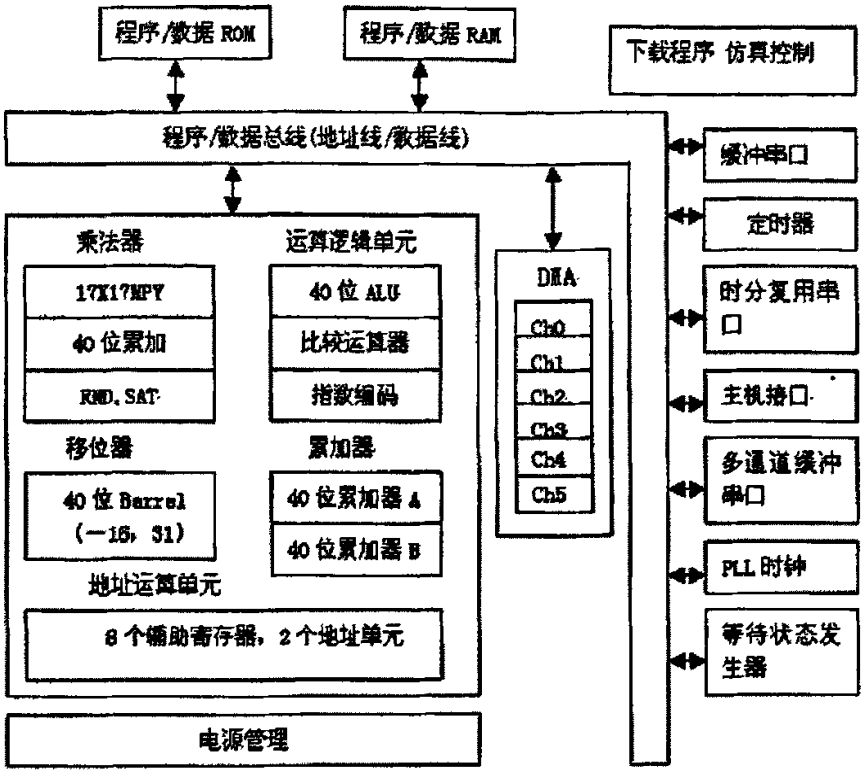


图 4－1 TMS320C5402 内部结构方框图

TMS320C542 含有一个 16 位可预定设置周期的定时器其计时可中断控制有一个计时输出 TOUT=P82 引脚可作为计时控制输出或时序输出等。

4.2 存储器扩展

C5402 系列 DSP 存储器分为三个独立选择的空间---程序、数据和 I/O，其中程序存储器存放待执行的指令和执行中所用的系数(常数)，可使用片内或片外的 RAM, ROM 或 FLASH 等来构成；数据存储器存放指令执行中产生的数据，可使用片内或片外的 RAM 和 ROM 来构成。I/O 存储器存放与映象外围接口相关的数据，也可以作为附加的数据存储空间使用。C5402 通过包含在处理器工作方式的状态寄存器(PMST)中的 3 个状态位，选择片内存储器作为程序空间或数据空间。这 3 个状态位是：

3 个状态位是：

(1) MP/MC 位。MP/MC=0，则片内 ROM 安排到程序空间；

MP/MC=1，则片内 ROM 不安排到程序空间。

(2) OVLY 位。OVLY=1，则片内 RAM 安排到程序和数据空间;

OVLY=0，则片内 RAM 只安排到数据存储空间。

(3) DROM 位。DROM=1，则部分片内 ROM 安排到数据空间;

DROM=0，则片内 ROM 不安排到数据空间。

程序设计者可根据不同的需求，相应的配置这 3 个位，使系统的存储空间满足应用要求。C5402 总共提供了 192K word 的地址范围，并可通过增强外设接口扩充。芯片上有 16K 的 DARAM 和 4K 的 P/D ROM。所有的存储空间又分为多个 8K 大小的块，支持分页寻址处理，这使得在对其中的一个块操作时，可同时对一另个块进行操作，增加了处理速度。而对本设计而言，由于波形本身的数据量不确定，可能会采样更多的数据点，另外为了将来的系统升级考虑，仅仅依靠片上的资源可能不够，因此需要外扩存储器。

4.2.1 程序存储器

C5402 片内 4K 字 ROM 通过设置 MP/MC 引脚可以设定是否将片内 ROM 映射到程序空间。程序存储器空间主要用于存放要执行的操作指令和执行中所用的系数表，其主要内容如表 4.1 所示：

表 4.1 片内 ROM 程序存储器映射

地址范围	内　　容
F800H--FFEFH	（1）自举加载程序（2）256 字 u 律扩展表（3）256 字 a 律扩展表（4）256 字 sin 函数表
FF00H--FF7FH	测试保留
FF80H--FFFFH	中断向量表

当 MP/MC=0，这 4K 字的 ROM 被映射到程序空间的地址范围为 F000H～FFFFH，其中高 2K 字 ROM 中的内容是由 TI 公司定义，如上图所示。当处理器复位时，复位、中断、以及陷阱向量将被映射到程序空间的 FF80H。复位后，这些

向量可以被重新映射到程序存储空间的任何一页的开头。

4.2.2 外扩数据存储器

数据存储器存放执行指令所用的数据。通过一片 IS61LV6416L 进行外部数据空间扩展，将扩展空间分为 2 页，每页 32K 字，一共 64K 字，映射到内存空间的 OX8000H—OXFFFFH。DSP 的数据空间选通信号(DSP_DS/)作为外扩数据存储器的片选信号，将存储器的 A15 端和 DSP 的 A16 端相连，用以实现数据空间的分页扩展。其电路如下图所示：

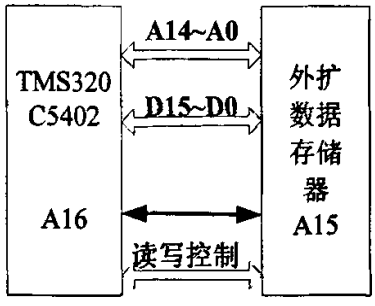


图 4—3 外扩数据 RAM

4.2.3 外扩 FLASH

在 TMS320C5402 DSP 系统的开发中，由于 DSP 片内只有 ROM 和 RAM 存储器，如要将用户代码写入 ROM 中，必须要由 DSP 芯片厂家来完成；但这样做用户就不能再更改代码，很不实用。由于 RAM 在 DSP 掉电后不能再保存数据，因此，常常利用 EPROM, Flash 等一些外部存储器来存放用户代码。在 DSP 上电工作后，利用 DSP 提供的 boot 机制，再将程序下载到 DSP RAM 中运行。如果使用 EPROM 外部存储器存放用户代码，需要用代码转换工具将用户代码转换为二进制目标文件，然后用编程器将其烧写进 EPROM；而如果使用 Flash 存储器存放用户代码，则可直接使用 DSP 仿真器和 CCS(Code Composer Studio)仿真环境进行在线编程，使用灵活方便，不再需要其它编程设备。我们就采用 Flash 存储器来实现多份用户代码的有选择加载。

本设计中，通过一片 AT49LV1024A 进行 FLASH 扩展，用来存放用户代码。AT49LV1024A Flash 存储器的容量为 64K 字，分为 2 页，每页 32K 字。映射到内

存空间的 OX8000H-OXFFFFH。DSP 的程序空间选通信号(DSP_PS/)作为外扩 FLASH 存储器的片选信号，将存储器的 A15 端和 DSP 的 A17 端相连，用以实现程序空间的分页扩展。其电路如下图 4—4 所示。

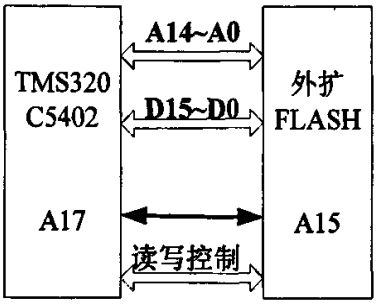


图 4—4 外扩 FALSH

系统上电后程序自举引导执行，进行系统测试。并将程序搬移到 RAM 中。自举引导的具体实现将在后续的第五章中介绍。

4.3 外围电路设计

外围电路包括：外围控制电路(电源、时钟电路、复位电路)，接口电路。

4.3.1 电源、复位电路设计

系统设计中 15V 电压信号是由外部直流稳压电源提供。而 C5402 工作电压有两种：内核电压为 1.8V，输入输出电压为 3.3V。系统设计中采用 TPS73HD318，该芯片的输入电压为 5V，它的输出电压分别为 3.3V 和 1.8V，每路电源的最大输出电流为 750mA，并且提供两个宽度为 200ms 的低电平复位脉冲。

TPS73HD318 是一种低压差、小静电流的双电压调节器。该芯片最大输出电流达到 750mA，输出电流能在两个调节器之间理想分配，从而可以为现今多数的 DSPs 提供电源电压。使用该芯片产生工作电压原理图见图 4—5。

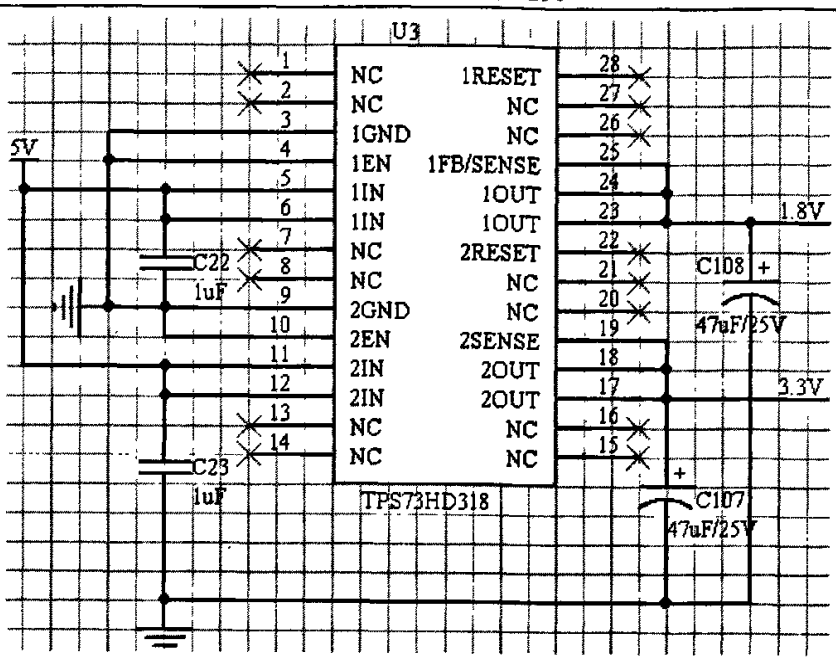


图 4－5 工作电压原理图

当 C5402 的 RS 位引脚上出现 2 个外部时钟周期以上的低电平，即可实现系统复位。C54x 复位有三种方式，即上电复位、手动复位和软件复位。前两种是通过硬件电路来实现的复位，后一种是通过指令方式来实现的复位。由于 DSP 系统的时钟频率较高，在运行很可能发生干扰和被干扰的现象，严重时系统可能会出现死机现象。为了克服这种情况，除了在软件上作一些保护措施外，硬件上也必须作相应的处理。设计中采用 MAX708 实现手动复位电路，具体电路如图 4－6。

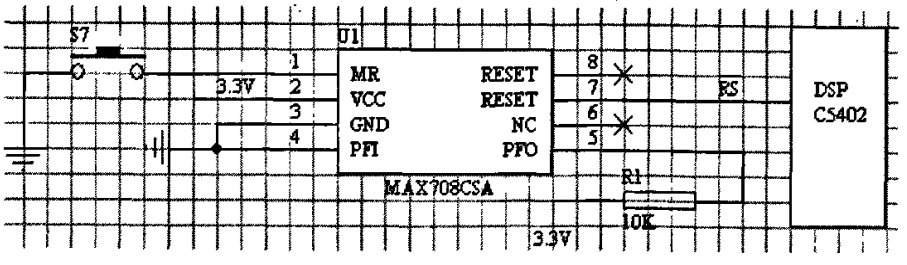


图 4－6 复位电路

系统有可能出现程序飞跑的现象。为了增加系统的可靠性，使之能在异常情况下重新复位，需采用看门狗。看门狗分为硬件看门狗和软件看门狗。在本系统中，为了简化硬件电路，采用了软件看门狗。将在第五章第三节予以介绍。

C5402 时钟发生器由一个内部振荡器和一个锁相环(PLL)组成。输入的参考时钟可由晶振和内部振荡器产生，也可以直接通过外部时钟源提供。通过 DSP 的内部锁相环电路，芯片可以被配置为 PLL 和 DIV 模式。PLL 模式可以产生 0.25～15 共 31 个乘系数；DIV 模式可以产生除以 2 或 4 两个系数。

系统一旦复位，时钟模式由引脚 CLKMDl, CLKMD2 和 CLKMD3 来决定，对应模式如下表[23]所示：

表 4.2 时钟模式表

CLKMD1	CLKMD2	CLKMD3	CLKMD 的复位数	时钟模式
0	0	0	E007h	PLL×15
0	0	1	9007h	PLL×10
0	1	0	4007h	PLL×5
1	0	0	1007h	PLL×2
1	1	0	F007h	PLL×1
1	1	1	0000h	1/2(PLL 无效)
1	0	1	F000h	1/4(PLL 无效)
0	1	1	——	保留

系统中 DSP 时钟源是通过外界在 X2 管脚 20MHz 的晶振产生的。通过设置时钟模式管脚(CLKMDI,CLKMD2,CLKMD3)为 100，选定 PLL 为 2 倍频模式，从而系统时钟频率为 40MHZ。电路如图 4—7：

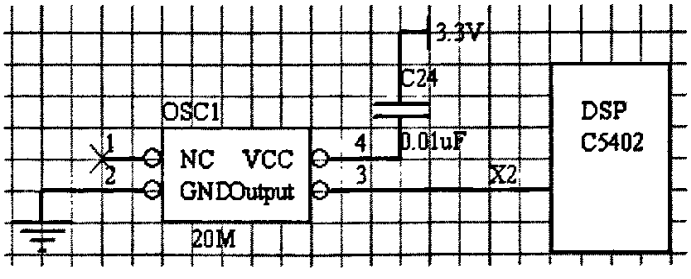


图 4—7 时钟产生电路

由于 DSP 的程序需要从外部低速 Flash ROM (AT49LV1024A)中调入，因此，可采用较低工作频率的 DSP 复位模式，这时，可将 CLKMDl～CLKMD3 设置为

111，则复位后 DSP 的工作频率为外部时钟 10MHz。待程序全部调入到 DSP 内部快速 RAM 后，再用软件重新配制 CLKMD=1007H，就可以使 DSP 工作在 2×20MHz=40MHz 的频率，甚至可以根据系统的需要工作于更高频率。

4.3.2 JTAG 仿真接口电路

JTAG(Joint Test Action Group)是 1985 年制定的检测 PCB 和 IC 芯片的一个标准，1990 年被修改后成为 IEEE 的一个标准，即 IEEE1149.1-1990。通过这个标准，可对具有 JTAG 口芯片的硬件电路进行边界扫描和故障检测。

具体 JTAG 口有如下 JTAG 引脚定义[22]:

TCK——测试时钟输入;

TDI——测试数据输入，数据通过 TDI 输入到 JTAG 口;

TDO——测试数据输出。数据通过 TDO 从 JTAG 口输出;

TMS——测试模式选择，TMS 用来设置 JTAG 口处于某种特定的测试模式;

可选引脚 TRST——测试复位，输入引脚，低电平有效。

通过程序将对 JTAG 口的控制指令和目标代码从 PC 的并口写入 JTAG 的 BSR 中。只要用 JTAG 指令将数据、地址及控制信号送到其 BSC 中，就可通过 BSC 对应的引脚将信号送给 Flash，实现对 Flash 的操作。将程序在线烧写到外扩的 FLASH 中，JTAG 的设计和连线关系如图 4—8 所示：

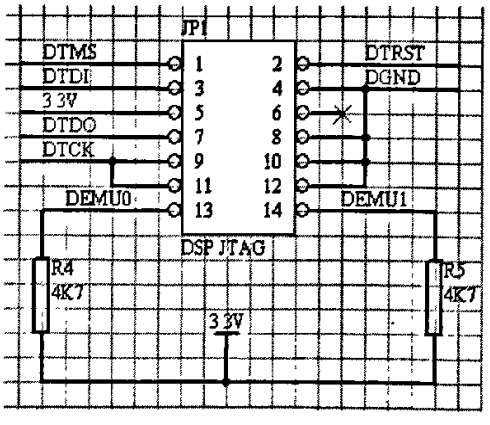


图 4—8 JTAG 仿真接口

4.4 DSK 简介

DSK 的全称是 DSP Starter Kit，即 DSP 初学者套件。DSK 为用户提供了一个开发 DSP 的方便的平台，由于 DSK 板上提供了基本的硬件支持，结合适当的软件开发工具，简化了开发工作，缩短了开发周期。在 DSK 板上主要包括如下资源：

- 100MHz VC5402 DSP 芯片
- 64K 字外部扩展 SRAM(64K*16)
- 256K 字的 FLASH 存储器（256K*16）
- 仿真器 JTAG 测试总线控制器 SN74ACT8990JTAG TBC 及与主机相连接的并行接口
- 模/数转换器 TI TLC320AD50 A/Dconverter 两个
- 电话接口（DAA）
- 麦克风/扬声器接口
- 并行口
- RS-232 串行口
- 扩展子板接口
- LED*4 (其中三个可供程序运行时点亮提示)

DSK 板上硬件资源分布如下图 4－9 所示：

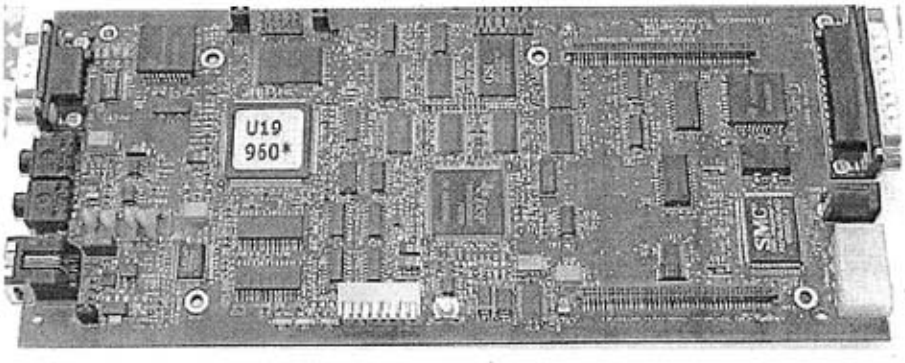


图 4－9 DSK 开发板电路板

为初学者设计和生产的DSK是一种廉价的开发工具。用户可以使用DSK来作DSP的实验，进行诸如控制系统、语音处理等应用；也可以用来编写和运行实时源代码，并对其作评估；还可以用来调试用户自己的系统。

在电脑上编写的程序可以通过DSP仿真器下载到开发板上进行调试，非常的简单方便。仿真器即扩展开发系统XDS（extended development system），可用来进行系统级的集成调试，是进行DSP芯片软硬件开发的最佳工具。

图4－10是DSP仿真器的实物图：

图4－10 DSP仿真器图

其主要特点有：

- 采用高速USB2.0标准接口，传输速度可达480MB/S，向下兼容USB1.1主机；
- 标准Jtag仿真接口，不占用用户资源；全面支持JTAG接口热插拔；
- 支持TI CCS2.X、支持CCS3.1集成开发环境,支持c语言和汇编语言；
- 仿真速度快，支持RTDX数据交换；
- 不占用目标系统资源；
- 自动适应目标板DSP电压；
- 可仿真调试TI公司 TMS320C2000、TMS320C3000、TMS320C5000、TMS320C6000、3X、C4X、C5X、C8X及OMAP、DM642等全系列DSP芯片。
- 支持多DSP调试，一套开发系统可以对板上的多个DSP芯片同时进行调试。

4.5 本章小结

本章首先对 DSP 芯片作了简短的介绍，接下来主要对 DSP 在设计中的具体应用作详细的说明，包括存储器的扩展、外围电源电路、时钟电路、复位电路、接口电路等都给出了具体的连接，对它们的引脚或者工作方式作了简单的阐述。最后是对 DSP 开发过程中用的开发板作了个说明。

第五章 软件设计

5.1 VC 人机界面的设计

Visual C++由微软公司所发布，是一种在 C++语言基础上的开发工具（也有人视其为整合开发环境---Integrated Develop Envirenment）,主要用来开发窗口应用程序。

“Visual”即“可视化”，“C++”成为“带类的 C”，Visual C++就是在 C 语言的基础上引入面向对象的机制而形成的一门程序设计语言[26]。

波形产生部分也是本设计的一个重要部分，一个核心内容，是设计中的一个亮点。

5.1.1 波形发生部分

本设计中就是用 VC++来实现波形产生部分，在 PC 上编程实现人机界面窗口，在窗口规定的区域内可以用鼠标来任意的绘制曲线，也就是任意波形。

主要分为几个部分：绘制曲线、图形保存、图形采样并存入数组。

1.绘制曲线：

```
class CPantdrwView::publicCView
{protected:
  int s_drag;                    //定义该成员变量用于表示鼠标被按下没有的 int
                                   变量，0 表示未按下，1 表示按下。
  HCURSOR s_cross;               //该变量用于保存光标的指针，HCURSOR 是光标
                                 类，该语句表示 s_cross，是光标类 HCURSOR 的对象。
  CPoint   s_pointend;           //MFC 基类 CPoint，是封装了系统坐标（x,y）的
                                 类，当鼠标按下时，将鼠标的位置由 CPoint 信息表示。
  CPoint   s_pointbegin;
protected:
```

```
  CPantodrwview();.....
}
CPantoDrwView::CPantoDrwView()
{
    // TODO: add construction code here
    s_drag=0;  //初始化，表示鼠标没有按下
    s_cross=AfxGetApp()->LoadCursor(IDC_Pantocross);
          //载入鼠标资源。将自己绘制的鼠标 IDC_Pantocross 载入到 s_cross。
}
```

在视图类的实现文件 CPantodrwview.cpp 中点右键，通过 appwizard(应用向导)加入鼠标左键按下时对应的消息相应函数：WM_LBUTTONDOWN。

```
void CPantoDrwView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CView::OnLButtonDown(nFlags, point);
    {s_pointbegin=point;       //point 变量是鼠标按下函数的参量，CPoint 型，
                                 指按下点坐标。
     s_pointend=point;        //将鼠标按下处的坐标送给 s_pointbegin 和 s_pointend
     SetCapture();
     s_drag=1;                //标示鼠标已经按下。
    }
}
```

同样，在鼠标的资源中加入消息相应函数：WM_MouseMove，该函数为按下鼠标移动过程中的消息相应函数。

并加入程序代码：

```
void CPantoDrwView::OnMouseMove(UINT nFlags, CPoint point)
{
```

```
// TODO: Add your message handler code here and/or call default
CView::OnMouseMove(nFlags, point);
{::SetCursor(s_cross);
//设置当前的图标为我们的绘图光标，前面的图标用于调用 win32 的 API 函数。
                    //s_cross 是我们前面定义好的 HCURSOR 型变量
if(s_drag)          //s_drag 也是前面定义好的 int 型变量表示是否处于画线状
                    态，即鼠标是否按下。
{CClientDC ClientDC(this);
                   //该语句用于创建一个客户区的环境句柄。CClientDC 是用户
                    区设备句柄类，从 MFC 设备句柄类 CDC 中派生。
ClientDC.SetROP2(R2_BLACK);
                   //该语句用于设置绘图颜色为黑色。
 s_pointend=point;     //将当前鼠标运动到的点赋给绘制直线的终点。
 ClientDC.MoveTo(s_pointbegin);
 ClientDC.LineTo(s_pointend);
                        //上述语句用于直线的绘制。
 s_pointbegin=point;
 //将前面绘制直线的终点，付给下一条直线的起点，用于下条直线绘制。
}
 }
```

在鼠标资源中加入鼠标释放消息处理函数 WM_OnLButtonUp();

程序代码如下：

```
void CPantoDrwView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CView::OnLButtonUp(nFlags, point);  //鼠标释放时的函数响应
    {if(s_drag)                         //判断是否处于画图状态
```

```
    {s_drag=0;                  //如果不处于画图状态，则中止鼠标信息。
     ::ReleaseCapture();        //停止对鼠标的捕捉。
    }
   CView::OnLButtonUp(nFlags, point);
    }
}
```

以上的程序完成后，编译运行，便完成了基本绘图程序。

2.图形保存

前面的程序通过在视图类 CPantodrwView 中进行程序的修改，实现了在应用程序中绘制任意曲线的功能。但是还未能实现对绘制的曲线的保存和其它的后台处理能力。若要实现绘制曲线的保存和处理能力，必须对程序的文档类即：CPantodrwDoc 基类作相应的修改。

参考前面的相应论述，文档一视图类程序中，数据的保存工作必须在文档类中实现。为了对绘制的曲线进行文档式保存，可以把任意的曲线看成一小段一小段的直线，对每条直线依次存储，而每条直线只需要存储起始点和结束点两点坐标即可。在文档类 CPantodrwDoc 中加入下面的新的直线类 CLine 的定义。

注意：在 VC 中增加新的类主要有两种方式，一个是在类视图（VC 开发界面）右边按右键快捷方式增加，这种方法将增加一个独立的类，并且为这个类生成了相应的头文件和源程序文件，另一种方法是在 VC 生成的现有基本类（初始化时，生成了五个基本类：CPantodrwDoc，CPantodrwView, CPantodrwApp, CAboutDlg, CMainFrame），在基本类内部定义和实现类。

本例中，新类即自定义的直线类 CLine 是在文档类 CPantodrwDoc 中定义的，显然是属于后者的方式。在程序文档类的头文件：PantoDrwDoc.h 中加入下面的定义：

```
//定义一个画图的类 CLine，该类从 CObject 中派生。
//注意的是类的定义以及对现成的类中的成员函数和成员变量的定义必须全部放在头文件中，而实现必须在 cpp 源文件中。
class CLine: public CObject
{protected:
```

```
    POINT S_beginp, S_endp;      //POINT 是用于返回窗口上的坐标的类，可以返
                                   回 x 坐标和 y 坐标
 CLine(){}                       //必须有的不带参数的构造函数
 CLine(POINT beginp, POINT endp)  //带参数的构造函数
      {S_beginp.x=beginp.x;
       S_beginp.y=beginp.y;      //利用 POINT 类返回 x 坐标和 y 坐标
       S_endp.x=endp.x;
S_endp.y=endp.y;
   }
void Draw(CDC *pDC);             //用于绘制直线
};
```

从上面类 CLine 的定义看出，定义了两个成员函数 S_beginp, S_endp，且都是 MFC 的坐标类 CPoint 的对象。即定义了直线的两个点 S_beginp, S_endp，能够完全的描述出直线的性质。

同时，仍在 PantoDrwDoc.h 中的基类 CPantoDrwDoc 类定义中加入一个数组类成员 S_LineArray，这个成员是 CLine 的对象的指针，即 CLine*的对象组成的数组。产生这个数组的格式和加入程序代码如下：

```
CTypedPtrArray<CObArray, CLine*> S_LineArray;
```

//这是 MFC 的样板实例，用于生成一个动态数组 S_LineArray，用于存放变量和对象组合，且这个数组指向的数据由后面参数（即 CLine*）中第一个指定的类派生，存放的数据是第二个参数指定类型的数据项。

由于使用了类样板，因此必须包含这个头文件，在 StdAfx.h 中加入：

#include<afxtempl.h> 类样本包含头文件。

同时，在文档类 PantoDrwDoc.h 中基类 CPantoDrwDoc 的定义加入下面的成员函数声明，以备后面的数据存储用。

```
public:
    void AddLine(POINT beginp, POINT endp);
    CLine *GetLine(int index);  //用于提取直线的相关数据。
```

```
int GetNumLines();          //用于计算直线个数的函数。
```

以上的这三个新加入的成员函数便是 CPantoDrwDoc 类的成员函数。

以下是相应的成员函数的实现，放在相应的源程序中。在文档类的实现程序 PantoDrwDoc.cpp 的最后加入下面的代码：

```
void CLine::Draw(CDC *pDC)      //这是对 CLine 的成员函数 Draw 的实现。
{pDC->MoveTo(S_beginp.x, S_beginp.y);
 pDC->LineTo(S_endp.x, S_endp.y);
}
//以上程序说明了怎么在设备情景对象中画图。
void CPantoDrwDoc::Addline (POINT beginp, POINT endp,)
{CLine *PLine=new CLine(beginp, endp);
 S_LineArray.Add(PLine);
}
```

//上述程序用于存储直线，PLine 为直线 CLine 类的变量，后面 CLine *PLine=new CLine(beginp, endp);调用了 CLine 类中的带参数构造函数 CLine(POINT beginp, POINT endp)。因为 S_LineArray 是由 CObArray 生成的 CLine 类数组，所以可以调用 CObArray 的成员函数 Add 加 PLine 到数组 S_LineArray。

后面写入下面的程序：（这段函数用于获得直线的数据）

```
CLine* PantoDrwDoc::GetLine (int index)
{if(index<0||index> S_LineArray.GetUpperBound())
     return 0;
return S_LineArray.GetAt(index);
}
```

S_LineArray 的成员函数：S_LineArray.GetAt(index); 其中，index 是一个正整数，该函数返回在编号为 index 的数组成员，即组成数组 S_LineArray 的相应 CLine 指针类对象。

接下来的程序用于返回直线数据的数量：

```
int CPantoDrwDoc::GetNumLines()
{return   S_LineArray.GetSize();
}
```

S_LineArray 的成员函数：S_LineArray.GetSize();是返回数组 S_LineArray 的成员总个数，即绘制曲线中包含的小段直线的总条数。

转到视图类 CPantoDrwView 的实现文件 PantoDrwView.cpp 中，在前面已经定义的在视图类鼠标移动函数 OnMouseMove（）的末尾加上下面的函数：

```
CPantoDrwDoc   *pDoc =GetDocument();
        //GetDocument()是 CPantoDrwView 的成员函数，作用是用于获得文档类
          CPantoDrwDoc 的指针。
pDoc ->AddLine(s_pointbegin, s_pointend);
```

//将视图中直线的起始和结束点 s_pointbegin, s_pointend 作为直线的相应数据调用文档类的 AddLine 函数加载到文档新的 CLine 类对象数组中。

加入该语句使得前面定义的鼠标移动函数 OnMouseMove()在鼠标按下并移动时，一方面根据鼠标移动的坐标点 point 不断传给视图类 CPantoDrwView 中定义的两个参数 s_pointbegin, s_pointend 并且，利用 MoveTo 和 LineTo 函数不断绘制小的直线段组成曲线。另一方面，把移动过程中的直线段参量 s_pointbegin, s_pointend 通过 pDoc -> AddLine(s_pointbegin, s_pointend)传递到文档类中，进行存储。

下面修改视图类 CPantoDrwView 的 OnDraw( )函数，当窗口需要重新绘制时，MFC 自动调用 OnDraw 函数。首先在视图类的实现文件 PantoDrwView 中，找到函数 void CPantoDrwView::OnDraw( )，并对 OnDraw 函数进行修改：

```
void CPantoDrwView: :OnDraw(CDC *pDC)
{CPantoDrwDoc *pDoc=GetDocument();
 ASSERT_VALID(pDoc);
 int index=pDoc->GetNumLines();              //获取文档类中存储的直线总数。
while(index--)
   pDoc->GetLine(index)->Draw(pDC);
}
```

//重新绘制所有的直线，这样，在窗口大小移动时，直线被重新绘制。发生上述情况时，MFC 调用默认的画刷清空屏幕，然后调用视图类的 OnDraw 函数重新绘制屏幕。

当完成上述操作步骤后，编译并运行可执行程序，可以看到，当拉动窗口改变其大小时，窗口内的线条不会消失，但还没有实现程序退出后对所画图形的保存。

而当程序退出后，要想保存文档，并在下一次程序启动时能重新载入，需要文档的序列化问题（serialize）。序列化能够使文件持续，但文件只能在磁盘文件中顺序读写，不能随机存取。如果希望随机存取，可以考虑 microsoft 中的 ODBC 和 DAO。

当生成的应用程序中使用菜单的 save 命令时，系统自动调用 serialize 函数。

打开 CPantoDrwDoc 中的 Serialize 函数，修改代码如下：

```
void CPantoDrwDoc::Serialize(CArchive& ar)
{if (ar.IsStoring())
                    //读取信息
   S_LineArray.Serialize(ar);
else
                    //保存信息
   S_LineArray.Serialize(ar);
}
```

注意：S_LineArray 是 SObArray 的生成对象，可以直接调用 Serialize 函数。

同时，为了支持 CLine 对象的序列化，必须将 DECLARE_SERIAL 宏加入到类定义中。返回 PantoDrw.h 文件中，对直线类 CLine 类定义中进行修改，在两个构造函数：CLine(){}　CLine(POINT beginp, POINT endp)之间加入：

```
    virtual void Serialize(CArchive &ar);
```

在 PantoDrwDoc.cpp 的最后加入 Serialize 的实现语句：

```
void CLine::Serialize (CArchive &ar)
{if(ar.IsStoring())
```

```
        ar<< S_beginp.x<<S_beginp.y<<S_endp.x<< S_endp.y;
 else
   {int nVersion=ar.GetObjectSchema();
    switch(nVersion)
    {case 1:
       ar>> S_beginp.x>>S_beginp.y>>S_endp.x>>S_endp.y;
       break;
     default:              //版本出错
       break;
     }
}
```

在 PantoDrwDoc.h 关于 CLine 类定义中加入：

```
    DECLARE_SERIAL(CLine)          //使类能够序列化。
```

在文档类 PantoDrwDoc.cpp 中加入宏，位置值 Draw 函数的实现前面加上

```
    IMPLEMENT_SERIAL(CLine, CObject,1)
```

完成上面的所有步骤后，便实现了对所绘制图形的保存功能。

3.图形采样并存入数组

从前面的程序可以知道：绘制的曲线是由一小段一小段的直线段组成，且每个直线段是前面在文档类 CPantodrwDoc 中定义的 CLine 类的对象。该类的对象包含两个成员变量： S_beginp, S_endp; 这两个变量都是 POINT 型的，即直线段在 windows 窗口中开始和结束的坐标点，而用 POINT 类变量来表示坐标，由可以进一步分为 POINT.x 型和 POINT.y 型，分别代表坐标点的 x 坐标值和 y 坐标值。如可以用 S_endp.x 和 S_endp.y 来表示该直线段结束点的 x 坐标和 y 坐标。

按照前面部分的介绍，用 VC 程序开发出的应用程序，窗口及坐标分布如下：假设应用窗口的大小为（600*500）象素。

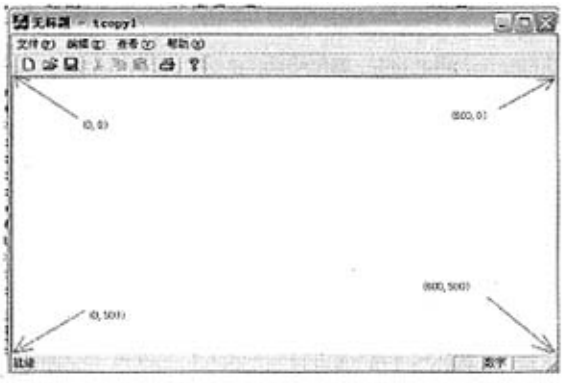


图 5－1 VC 应用窗口图

应用程序窗口内的坐标点，x,y 的值从左上角起始，开始均为 0，分别按照往左和往下逐步递增。

将绘制的曲线消息采集到一个数组 x[400]中，因为组成曲线的直线段足够短的话，可以近似地看成一个点，可以将每个直线段的起始或结束点保存，且将每个直线段起始或结束点的 x 坐标作为数组的下标 i,而 y 坐标为下标为 i 的数组元素的值 x[i]。即保存某个直线段的结束点 S_endp 时,可以转换为 S_endp.y= x[S_endp.x]。组成直线的 c 语言中，x[400]包含了 400 个成员：x[0]、x[1].....x[399]。

假设在窗口上绘制了图形后，要将 x 坐标在 100 至 500 范围内的图形部分作为信号一个周期波形存储到一个 400 个单元的数组 x[400]中。应该按照 x 坐标的顺序依次存储，x 坐标为 100 的点存储到第 1 个数组成员，即 x[0]中。x 坐标为 101 的点存储到第 2 个数组成员，即 x[1]中。依次类推，x 坐标为 i 的坐标保存在数组成员 x[i-100]中，且假设以 y 坐标为 150 的水平直线作为标准的 0 轴，图形高于该 0 轴的部分，存储在相应数组中的值为正值，且坐标离 0 轴的距离越远，数组成员的值越高，而低于 0 轴时，存储的值为负值，坐标离 0 轴的距离越远，数组成员的值越低。因此还必须对 y 坐标作相应的处理后，再存入数组中，处理计算式：150－S_endp.y，结合前面所述，该值（150－S_endp.y）将存入下标为（S_endp.x-100）的数组成员 x[i]中，i＝S_endp.x-100，x[i]=150－S_endp.y。

下面的示意图中，中间一条水平的蓝色直线为标准 0 轴，该 0 轴的 y 坐标为 150，x 坐标从 100 到 500。红色曲线为所绘制的波形，虚线框中的曲线部分将会保存在数组 x[400]中。

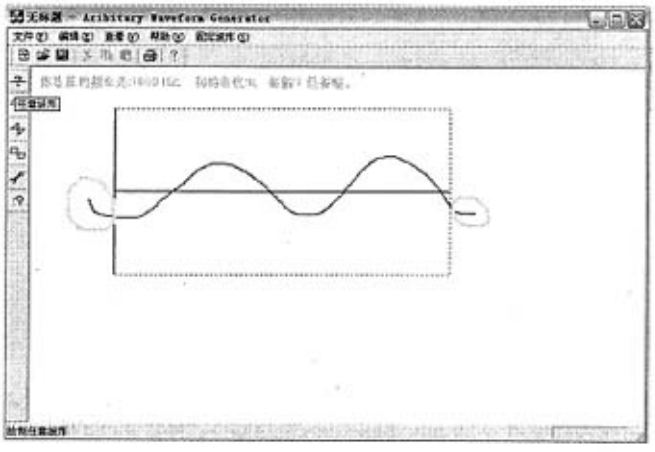


图 5—2 绘图示意图

如上图所示，该图显示了基于 VC 开发的用户界面软件，且软件界面的左边为工具栏，最上方显示问号的按钮，表示任意波形生成的按钮。

首先，用户在类似于画图板的界面中绘制任意形状的波形，当在界面中点击鼠标左键时，便能够绘制波形，且此时生成的波形为用户自定义的任意波形，如上图中的红色曲线，该界面软件能够将位于蓝色曲线框中的曲线部分进行获取，并采样。蓝色的实线，和虚线共同构成的框图，表示绘制波形的区域。其中，垂直的蓝色实线表示绘制图形的纵坐标，而水平的蓝色曲线表示横坐标，当绘制的波形在该坐标上方时，波形幅度为正值，反之，在下方时，波形幅度为负值。上下水平的蓝色虚线表示绘制波形的上限和下限。即是绘制的波形某点距离横轴的距离代表了该点的幅度，当波形到达上面或下面的蓝色虚线时分别代表幅度达到的最大值和最小值，如果波形的绘制超出了上下限，将被“截断”，即代以上下限表示。同理，垂直的两条直线也代表了波形采样在横坐标上的范围，如上图所示，黄色圆圈中的波形将会被剪掉。

绘制波形完成后，通过点击左边竖直工具栏最上方的问号型按钮，如图 5—2 所示，便会出现“任意波形”的提示，点击该按钮，软件进行相应的采样工作，并且生成和波形系数相关且硬件 DSP 可以识别的.dat 文件。采样点一共为 400 个，从图示的起始点到右边的终止点，在 VC 的实现过程中，一共开辟了 400 个单元的实型数组进行存储，同时在 DSP 中也开辟了长度为 400 的存储单元对波形进行存储。400 个单元的长度存储的波形参数，将会作为一个周期长度的波形，通过基于 DSP 为核心的硬件电路，进行输出。前面所述的.dat 文件是从 VC 的数组到 DSP

存储空间的转换连接的桥梁，生成的数据长度为 400 的.dat 文件如下图所示：

图 5—3 数据文件图

该文件是基于文本格式的文件。其中，dat 文件的第一行是用于 TMS 系列 DSP 读取数据所配置的识别变量，后面每行表示对用户输入波形在每个采样点处采集的波形，并用—100 到 100 的十进制数量化后的数据。

视图部分的一个优化问题就是在用鼠标绘制波形时可能会由于各种原因采样点数达不到要求的数量，针对这样的情况采取的措施就是在得到的点数之间插入几个数据点，需要插入几个点就将幅度间隔平均分成几部分，然后在起始点的基础上累加就得到了所有的点数值，从而得到自己需要的点数。图形也就得到了相应的优化，不会出现采样点数不够导致的波形畸变的情形。

除了用户自定义的任意波形外，该用户软件还能生成标准的波形，如方波、三角波和正弦波，这些标准波形的生成，只需要在软件界面左边的竖直工具栏上点击相应的按钮，便能生成。下面的图示展示了标准三角波、方波的生成示意。

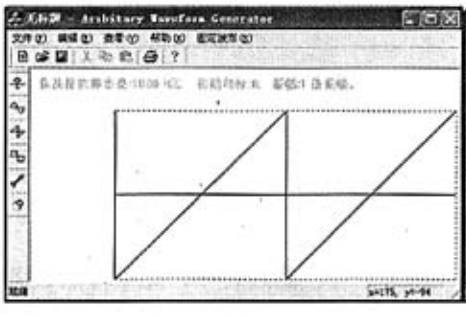


图 5—4 三角波

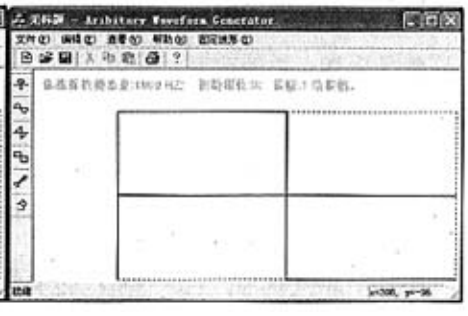


图 5—5 方波

用户还可以通过左边的工具栏进行波形参数的设置，波形设置的按钮为倒数第二个扳手型的按钮，点击后出现如下的对话框界面，如图所示可以对波形的频率、幅度和初始相位进行设置。

图 5—6 波形参数设置图

5.1.2 波形频率的控制

频率控制的实现流程是由用户软件输入，软件存储和控制实现的。其过程即由用户通过在基于 VC 的用户界面中，通过参数设置对话框进行设置。用户输入的频率信息，将会首先被视图层设置的变量 m_freq 获取，同时通过视图层中指向文档层的指针 pDoc。由 C++程序 cCPantoDrwDoc* pDoc = GetDocument()；…… pDoc->m_freq=m_freq；传向下面的文档层变量 m_freq，由文档层完成存储。

输出波形的频率改变可以通过以下几种方式来控制：

(1)就是采样点数的改变

(2)改变 DSP 往 DA 送数的时间间隔

(3)改变对采样点数的读取方式（跳读，也就是间隔读取波形数据）

第（1）种方法：

采样点数的改变，非常容易，就是在人机界面绘图的过程中，对采样点数进行相应的修改即可做到。不过点数减少的话，波形的精确度就会减低，这对任意波形来说是一个严重的问题。

第（2）种方法：

在DSP实现中调用相应的延时子程序，此法适用于设置的输出频率较低，且对波形要求较高的情况。据前所述，任意波形的波形幅度参数分别位于DSP的400个存储单元中，波形的输出是利用DSP的堆栈指针SP，分别从1到400遍历400个存储单元，将波形数据依次读出，并由DSP的输出引脚输出到数模转换模块中，根据量化数字信号的值输出相应的模拟波形。延时程序的作用是，根据由用户定义的波形频率值，设置一个类似于计数的程序，并且计数的门限值由频率决定，当每次SP读取一个存储单元的波形数据时，调用一次延时程序，程序开始计数，当计数从零，按照DSP机器周期的速度，一直增大到门限值时，便返回到主程序中，使SP指针继续读取下一个存储单元的数据。可见，当需要的输出波形频率越高时，延时子程序设置的门限值越小，通过它消耗的时间越短，反之门限值越大，子程序上消耗的时间越长。一种特殊情况是当所需要的频率最大时，系统设置为不调用延时程序，则DSP读取每一个存储单元的信息，是根据系统所能达到的最高的速率来进行，这个考虑到后面D/A转换的速率，会设置为一定的周期，来满足DA转换器和低通滤波器的要求。

利用DSP平台的汇编语言编写的延时程序核心部分如下所示：

```
stlm a, ar4              ; ar4 = loopCount-1
.....
loop:
stlm a,brc               ; initialize count register for block rpt
.....
rptb $1
nop
nop
nop
$1   nop
banz loop, *ar4-
```

以上程序，首先由DSP的主程序将根据需求频率计算出的门限值，放入累加器A中，同时，将累加器A的值送入寄存器AR4中，随即设置了相应的循环程序，并

且设置循环程序的门限值为 AR4，每循环一次，对 AR4 中的值减一，当 AR4 中的值减为 0 时，退出循环，回到主程序，由此实现了延时计数的功能。

此法本质上说，是在 DSP 往 DA 送数据的过程中加入一个计数周期为 N 的计数器，这样可以通过对 N 设定不同数值，来改变计数器的周期，从而改变 DSP 往 D/A 写数据的时间间隔，也就改变了 DA 转换后的数据保持时间，最终改变输出波形的周期。此法采样点数都会读取到，但是周期变长，波形频率降低。

第（3）种方法：

跟前面不同的是，在 SP 指遍历波形数据时，设置 SP 每次增加的步进值的方式，当步进值为 M（M 可以是 2、3、4.....）时，SP 在读取每个波形数据单元后，其值以 M 自增，读取后面第 M 的单元的波形数据值。不难看出，前面一种方法中，设置的步进值均是 M＝1，即遍历时，依次读取每个单元的数据，而第二中方法将会忽略两个读取单元间的 M-1 个单元的数据。同样的情况下，输出波形的周期变短，波形频率增大。

该方法 DSP 就对数据间隔的读取，采样点数不是每个都用到，从而会对波形细节造成损失，因此适合于对波形要求不高，但频率输出相对较高的情况。

5.1.3 波形幅度和初始相位的控制

和频率控制的实现类似，幅度和初始相位的控制也是由软件输入，软件存储和硬件控制实现的。其过程也是通过用户通过在基于 VC 的用户界面中，通过参数设置对话框进行设置。用户输入的幅度和初始相位信息，将会分别被视图层的设置的变量 m_phas 和 m_amp 获取，同时通过视图层中指向文档层的指针 pDoc，传向下面的文档层变量 pDoc->m_pha 和 pDoc->m_amp，由文档层完成存储。

5.2 DSP 软件开发环境 CCS

5.2.1 CCS 的特点和使用方法[27][28][29]

常用的用于 TMS320C5402 的 Code Composer Studio2. 0 是个高度集成的开发环境，用以满足复杂的 DSP 应用的要求。它综合了 TI 的代码产生工具，CCS IDE,DSP/BIOS II 实时内核和实时数据交换(RTDX)技术，把平均译码时间减少了 50%或更多。其组件包括：CCS 集成开发环境，包括调试器，编辑器，剖析器，

工程管理器和基于 C 的描述语言。基于标准 JTAG 接口，包含 RTDX 技术的 DSP/BIOS，高效 C 优化编译器，汇编语言编译器和链接工具，支持指令集结构的软件仿真器。CCS 主要特点有：

(1)将编辑，调试，项目管理，分析和探测点集成在一个环境里。

(2)包含 C 编译器，汇编优化器，和连接器(代码生成工具)。

(3)实时的基础软件(DSP/BIOS)。

(4)主机和目标机之间的实时数据交换(RTDX)。

(5)实时分析和数据可视化。

(6)允许同时插入主机和目标机，使开发者扩展其开发环境。

(7)第三方厂商的插件可以使目标机硬件的配置更容易。

(8)数据可视化。

下面就 Code Composer Studio 的使用作简单的介绍：

(1)项目的组成和建立

与一个 project 有关的文件有：C 语言的主调用流程文件（*. c)，汇编语言模块文件(*.asm)，连接器命令文件(*.cmd)，运行时支持库文件(*.lib)，编译器目标文件(*.obj)，头文件(*. h)，项目文件(*. mak)，可执行文件(*.out)等，如果要烧入 DSK 上的 Flash Rom，还要生成存储器映射文件(*.map)等。

建立一个项目时，在新建的项目中加入 C 语言文件、汇编语言文件，还有用于存储器映射的连接器命令文件，以及库文件(用于提供运行时的支持)等组成一个 project。如打开一个已有的项目，只需要打开该项目文件即可。

(2)文件说明

a. *.*或.c*文件：C 语言源程序，可以编译和链接

b. .a*或.s*文件：汇编源程序，可以编译和链接

c. .o*或.lib 文件：目标或库文件，可以链接

d. .h 文件：包含文件，不需要添加 header/include 文件，因为这些文件是自动加入项目列表的

e. .cmd 文件：链接命令文件，一个项目只需要一个.cmd 文件，否则，项目包含的文件就没有数量限制了。所有的文件都是用直接路径表示的，但它们是以相

对路径存储的。这样就很容易的可以将一个项目连接到另一个路径下。直接路径在每次打开一个项目时确定。存储的路径名，是相对于该项目的构造(make)文件的。

(3)项目的运行

项目建立好之后，首先要对其进行 build，系统检查 C 和汇编源文件，程序没有问题后，就可以用 file->load program 命令，使主机通过并口确认对 DSK 板 load，成功后即可用 debug->run 命令运行。

(4)watch window

watch window 是用于察看变量值的。可以用 view->watch window 打开，然后在 watch window 中，通过右键添加需要观察的变量名，也可以将光标置于要观察的变量名上，用右键打开 quick watch 察看变量值。

(5)断点设置

断点可以使程序的执行中断，当程序中断时，你可以察看程序状态，察看或更改变量，检查访问堆栈等等。但是要注意，不要在实现延迟的语句上设置断点，也不要在一个循环的最后一两句上设置断点，设置断点后，可以通过菜单，选择断点有效或无效。在 CCS 中也可以设置条件断点，可以在 GEL 文件中设置断点需满足的条件，当表达式结果为 0，处理器就会继续执行而不会刷新显示，否则就会像遇到一般的断点一样中断程序的运行。硬件断点和软件断点的设置都是通过 Debug->Breakpoints 打开断点类型窗口(Breakpoint type)设置的。

(6)探测点

探测点可以使特殊的窗口更新，或在算法的某特定点从外部文件读写。当探测点设置好后，一样可以使之有效或无效。一个窗口打开后，一般默认为断点刷新，但是也可以用连接的探测点(Connected Probe Point)均使窗口刷新，窗口刷新后，程序继续执行。

类似于 CCS 的文件输入输出功能，你也可以在代码的某点上用探测点读入或输出一串数据。当程序运行到探测点时，数据就从特殊存储区流向文件或者从文件流向存储器。硬件断点和硬件探测点都会影响到实时性，在仿真的 DSP 处理器上都不可行。

(7)存储器映射

CCS 调试器从存储器映射中可以知道哪一部分存储器空间是不能访问的。映射对应着连接命令文件(*.cmd)中的存储器定义。当你使存储器映射有效时，CCS 调试器，按照存储器映射检查每个访问，当你访问的是没有定义或受保护的存储器空间时，调试器就不访问目标而显示默认值。当 CCS 调试器将访问与存储器映射对照时，它将在软件中执行这一检查，而不是在硬件中。

存储器映射：存储器映射告诉 CCS 哪一块内存可以访问，哪一块不可以访问这种映射与你在链接命令文件中的定义相匹配。

5.2.2 CCS 的软件开发流程

利用 CCS 开发软件的大致流程如图 5－7[23]所示：

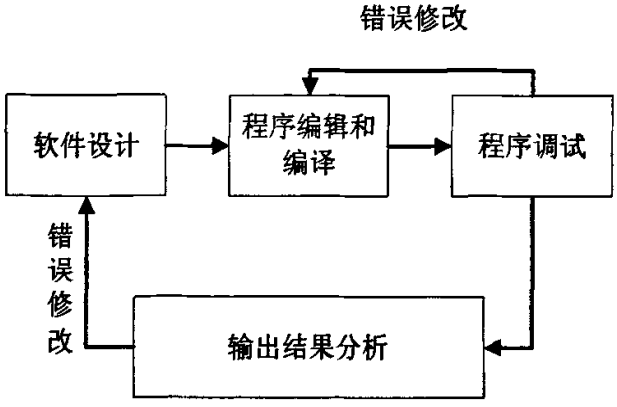


图 5－7 CCS 软件开发流程图

各个阶段完成功能如下：

（1）软件设计：程序模块化分，算法和流程确定，预测执行结果等。

（2）程序编辑和编译：创建工程文件，编写头文件，配置文件和源程序，使用汇编和 C 编译器进行编译，排除语法、变量定义等错误。

（3）程序调试：利用单步执行、断点、探针等手段调试程序。

（4）输出结果分析：利用 CCS 提供的工具分析 DSP 程序运行的结果，如图形显示数据或者统计运行时间等。若编写程序不能满足要求，则重新进行软件设计。

5.3 软件设计

软件设计包括主程序和子程序两部分。首先要对 DSP 进行初始化，即对存储器映射外围寄存器、状态寄存器、堆栈指针和处理器方式寄存器等进行相应的设置。其次是编写各个应用子程序。这样系统上电后才能工作在指定的方式下。接下来判断用户的输入，也就是根据用户的需要，看看是否发送波形，是发送常规波形还是任意波形，得出判断结果之后就进入到相应的子程序中执行，得到用户选定的波形输出。系统软件设计框图 5－8 如下：

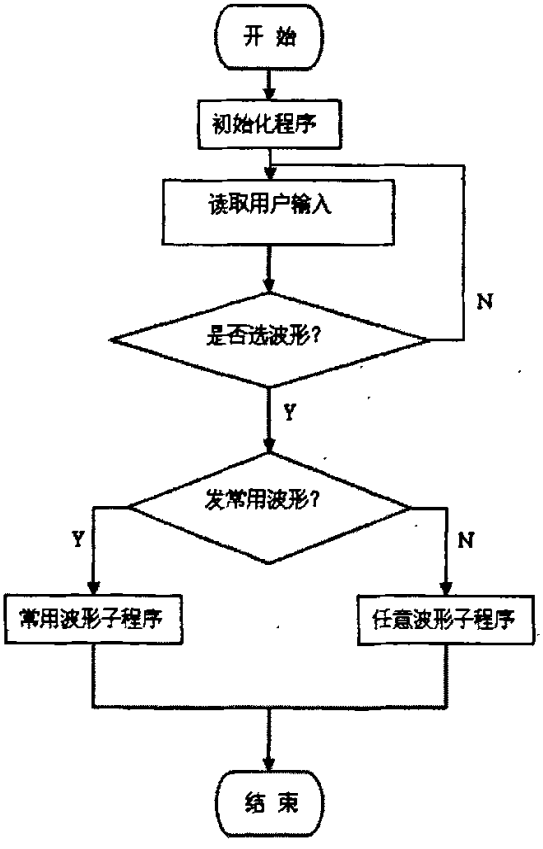


图 5－8 软件框图

5.3.1 系统初始化[29]

系统初始化程序包括影响DSP芯片的CPU运行的内部初始化和影响各个片内

外设工作的外设初始化，以及外围可编程器件的初始化的几个方面。DSP 系统上电后，DSP 芯片部分地进行了定义，因为 DSP 芯片的两个状态寄存器(ST0,ST1)的所有状态位都处于确定状态，因此，上电复位时，DSP 芯片即处于所谓的预定义状态，但上电复位后，用户可以通过编写初始化程序，对两个状态寄存器的一些状态位进行修改，以满足不同系统的系统配置和应用要求。主要包括：堆栈指针初始化(SP)、存储器映射外围控制寄存器（SWWSR 和 BSCR)初始化、状态寄存器初始化(ST0,ST1)、处理器方式状态寄存器初始化(PMST)。其流程如图 5—9：

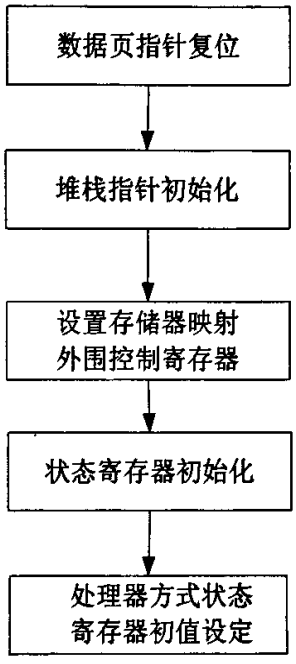


图 5—9 DSP 芯片系统初始化

5.3.2 WatchDog 设计

系统在受到干扰的情况下或者系统电源出现波动，有可能出现程序跑飞的现象。为了增加系统的可靠性，使之能在异常情况下重新复位，通常采用的方法是在系统中加入看门狗电路。

看门狗定时器的工作原理如下：系统正常工作时，周期性地刷新定时器，重新设置初值，使之不至于溢出而产生定时器中断。如果系统发生异常，则不能正常刷新定时器，于是定时器溢出产生中断，通常利用该定时器的中断使系统重新

复位，从而进入正常工作状态。看门狗分为硬件看门狗和软件看门狗。在设计中，可以使用硬件电路来复位，也可以采用软件看门狗。

系统启动时对软件看门狗初始化。初始化包括开相关中断、设置寄存器初值以及开始计数。初始化程序如下:

```
Dogreset:
    rsbx    cpl
    rsbx    intm            ;开所有中断
    st  #9h,  *(imr)        ;开定时器中断
    stm #l0h,  tcr
    stm #0c34fh, prd        ;每 0.1 秒溢出
    stm   #61h, tcr         ;开始计数
    ret
```

定时器中断服务程序用于系统复位。系统复位包括将系统的状态寄存器(st0,st1)恢复到复位的状态以及将模式状态寄存器恢复到脱机模式。中断复位程序如下:

```
timeisr:
    stm     #0ff80h,  pmst      ;恢复系统寄存器
    stm     #1800h,   st0       ;恢复系统寄存器
    stm     #2900h,   st1       ;恢复系统寄存器
    b        ff80h              ;软件复位
    rete
```

主程序设计时，应周期性地调用看门狗初始化程序，调用的时间间隔应不小于定时器的定时间隔，以避免定时器的溢出。

5.3.3 DSP 芯片控制程序设计

通过对 DSP 数字处理芯片的软件编写来控制其工作，集成开发环境 CCS 可以完成整个程序的设计，包括源代码的编写、编译、调试，同时配了仿真器，可以进行程序的调试与下载。

基于 TMS320C54X 系列的 DSP 程序设计，需要编写配置文件、中断文件以及

主程序文件。它们都可以在 DSP 集成开发环境中 CCS 中完成。

1.配置文件的编写

配置文件主要指的是 DSP 芯片的存储器的配置，文件以.cmd 为文件后缀名，该文件的功能是把实际的物理存储器按系统应用的需要配置到相应的地址空间中去。其中 MEMORY 和 SECTIONS 是两个关键的伪指令，MEMORY 伪指令是用来定义目标系统的存储空间的分配。书写格式如下:

```
MEMORY
    {
PAGE0:
PROG:origin=0x0080 length=2000
PAGE1:
DATA:origin=0x2090 length=4000
    }
```

PAGE0 和 PAGE1 分别用来指定程序空间和数据空间，origin 指定空间的起始地址，length 指定空间的长度。

SECTIONS 用来说明各个段分配到那个存储空间，书写格式为:

```
SECTIONS
{
    .text{}>PROG PAGE 0
    .cinit{}>PROG PAGE 0
    .pinit{}>PROG PAGE 0
    .vectors{}>VECS PAGE 0
    .switch{}>PROG PAGE 0
    .stack{}>DATA PAGE 1
    .bss{}>DATA PAGE 1
    .const{}>DATA PAGE 1
}
```

所谓段是指连续占用存储空间的一个数据或者代码块。编写汇编语言中的程序是以段的格式组织的，每行汇编语言都从属于一个段，并由汇编伪指令标明该段的属性。段是目标文件中可重新定位的最小单元，一个目标程序中的每个段通常是分开的和各不相同的。目标文件至少包含以下 3 个默认段：

.text 段(文本段), 通常包含可执行代码;

.data 段(数据段), 通常包含初始化的数据;

.bss 段(保留空间段), 通常为没有初始化的变量保留空间;

此外，汇编器和连接器可以建立、命名和连接自定义段，这些自定义段的使用与.data、.text、.bss 段的使用相同。

2.中断文件

中断服务程序的地址(中断向量)要装载到存储器的合适区域。一般这些向量都定位在 0x0 开始的程序存储器中。TMS320C5402 复位向量定位在 0x0，其他中断向量可以定位于任何 2K 字的程序存储器中，中断向量表的定位是与 PMST 寄存器的 IPTR 位有关。当程序中需要响应中断时，应设计中断向量表，并将其加入工程中去。本程序系统复位、波形数据输入和看门狗软件复位都需要设置响应中断。其程序见附录 2。

3.DSP 芯片主程序的编写

系统设计的主程序主要是采用 C 语言进行编写，因此需要在 CCS 里的程序项目中运行包含运行时支持库 rts.lib 文件，该库文件包含了标准 C/C++函数以及编译器用来管理 C/C++环境的函数。用户必须将所有 C/C++程序和目标代码链接，以便初始化时执行名为 BootLoader 的程序，BootLoader 程序负责如下的功能[20]：

(1) 设置状态和配置寄存器;

(2) 设置堆栈和二级系统堆栈;

(3) 处理.cinit 运行时初始化表和自动初始化全局变量;

(4) 调用所有全局对象构造器;

(5) 调用主程序;

(6) 当主程序(mian 函数)返回时调用 exit;

BootLoader 程序的入口点为_c_int00，在 rts500.1ib 中的 boot. obj 提供，该入

口地址通常用于设置 BootLoader 程序的起始地址。

设计中 TMS320C5402 的主要功能从 PC 获取数据进行整理、然后就是控制各个存储器和 D/A 的工作方式。整个 DSP 的程序存放在 FLASH 中，上电以后由 BootLoader 程序自动装载运行。

5.4 程序自举加载(Boot Loader)[30][31]

可编程的数字信号处理器芯片 C5402 从上电复位后到进入工作状态前的阶段叫自举加载阶段(Boot Loader)。当上电复位后，它的程序指针自动指到 ROM 中(从 F800H 到 FBFFH)的一个称为 Boot Loader 的程序，该程序是由生产商固化和升级管理的，主要完成一些数据的搬移和程序的重新定位工作。该程序根据环境选用相应的 Boot Loader 模式，将外部 FLASH 中的程序搬到 DSP 内部的 RAM 程序区中，并将程序指针移到执行程序的第一行处。

表 5.1 DSP 外部预载程序读取的模式状态表

D15-D8	D7-D4 模式	D3-D0 脉冲	代表在 FFFFH 地址设定预载控制模式
XXXXXXXX	0000	0000	8 位 BSP 标准形式内部 Bclkx 及 Bfsx
XXXXXXXX	0000	0100	16 位 BSP 标准形式内部 Bclkx 及 Bfsx
XXXXXXXX	0001	0000	8 位 BSP 标准形式外部 Bclkx 及 Bfsx
XXXXXXXX	0001	0100	16 位 BSP 标准形式外部 Bclkx 及 Bfsx
XXXXXXXX	0010	0000	8 位 TDM 标准形式内部 Tclkx 及 Tfsx
XXXXXXXX	0010	0100	16 位 TDM 标准形式内部 Tclkx 及 Tfsx
XXXXXXXX	0011	0000	8 位 TDM 标准形式外部 Tclkx 及 Tfsx
XXXXXXXX	0011	0100	8 位 TDM 标准形式外部 Tclkx 及 Tfsx
XXXXXXXX	0000	1000	8 位并行 I/O 读取模式
XXXXXXXX	0000	1100	16 位并行 I/O 读取模式
D15-D8	D7=D2 寻址	D1=D0 模式	代表在 FFFFH 地址设定预载控制模式
XXXXXXXX	6 位 SRC	01	8 位外部并行 EEPROM 读取程序数据
XXXXXXXX	6 位 SRC	10	16 位外部并行 EEPROM 读取程序数据
XXXXXXXX	6 位 ADR	11	预先暖栈模式 Warm Boot Mode

按照 Boot 时程序由外部 FLASH 存储器进入 DSP 片上 RAM 通道不同可分为 HPI、串行口、并行口等模式，按照数据进入 DSP 时字长不同可分为 8 位和 16 位模式。表 5.1 是 TMS320C5402 的外部预载程序读取的模式状态设定表。

C5402 Boot 过程如图 5－10 上电复位后，C5402 首先判断位于 IMR 和 IFR 寄存器的 INT2 标志位，若 INT2 有效则采取 HPI 模式，否则进一步判断 INT3 标志位；若 INT3 有效，则采用串行模式(Serial Mode)；否则读取 I/O 空间的 FFFFh 地址处，当该地址为有效地址，采用并行模式(Parallel Mode)；若该地址也无效，则读数据空间的 FFFFh 地址处，若该地址有效则仍采用并行模式，如果仍不满足条件时，进一步判断其他 Boot 模式。设计中采用了 8 位的并行加载方式。

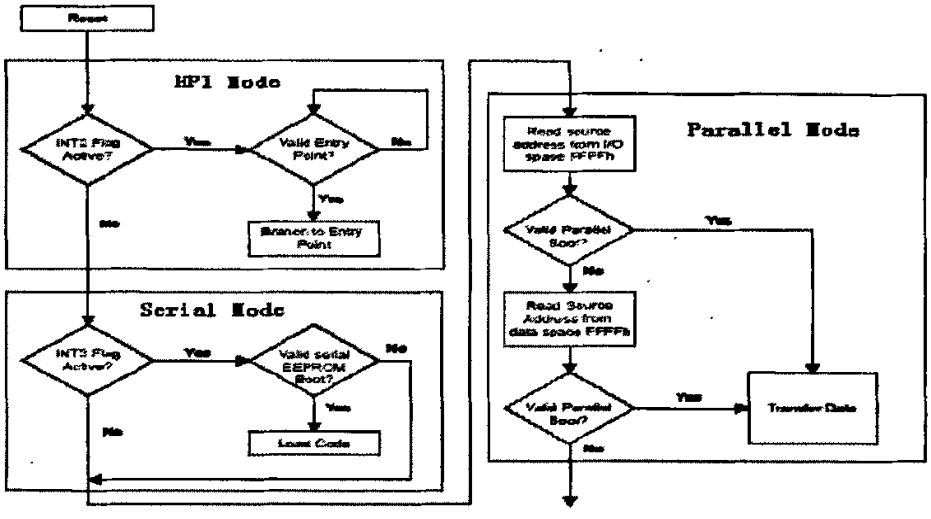


图 5－10 C5402 的 Boot Loader 部分流程图

Boot Loader 过程：

(1)生成 Boot 表

CCS 系统编译生成的.out 格式的文件不能直接通过 JTAG 口加载到 FLASH 中，而是要通过 CCS 自带的 hex500.exe 和一个特定的.cmd 文件将.out 文件生成个.hex 格式的文件，也叫 Boot 表。将已经编译好的名为 waveform.out 的文件加载到 FLASH 中，则需创建个.cmd 文件，取名为 waveform.cmd，其内容如下：

```
-waveform.out                /*输入的.out 文件*/
-waveform                    /*设置输出 ASCII 的十六进制格式文件*/
```

```
-map waveform.map          /*生成名为 waveform.map 的 map 表*/
-o waveform.hex            /*定义输出文件名*/
-bootorg PARALLEL          /*选择并行模式*/
-e 0x100                   /*定义入口地址*/
-boot                      /*生成 boot 表*/
-swwsr  0x7fff             /*设定 SWWSR 寄存器*/
-bscr   0x8802             /*设定 BSCR 寄存器*/
-memwidth  16              /*设定存储器字长*/
-romwidth  16
```

将 hex500.exe、waveform.cmd 和 waveform.out 放在同一目录下，执行窗口命令行：

```
hex500  waveform.cmd
```

即得到 waveform.hex 和 waveform.map，从 waveform.map 文件中可以容易的找到程序入口地址。

(2)加载程序到 FLASH 存储器

a.FLASH 擦除

在每次对 Flash 写入之前，要对其原来的内容进行擦除。Flash 的擦除包括扇区擦除(128 扇)、块擦除(每块 32 KW)和整片擦除三种。块擦除是对一个模块进行擦除，整片擦除是擦除整个 Flash 的内容。因此，对 Flash 的操作是以模块为基本单元的。本题使用第三种擦除方式。擦除步骤如下表：

表 5.2 FLASH 擦除

步骤	1	2	3	4	5	6
地址(0x)	D555	AAAA	D555	D555	AAAA	D555
数据(0x)	AA	55	80	AA	55	10

擦除过程中 FLASH DQ6 引脚出现 0 和 1 两种电平的跳变，当 DQ6 稳定为一个电位，表明擦除完成。

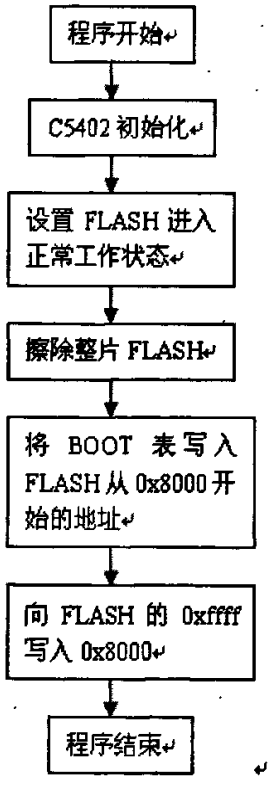
b.写入 FLASH

对 FLASH 写入程序由两个时序控制：一种是由 WE\控制的，另一种是由 CE\控制的。本题选用 WE\控制方式。写入步骤如下表：

表 5.3 写入 FLASH

步骤	1	2	3	4
地址（0x）	D555	AAAA	D5555	写入地址
数据（0x）	AA	55	A0	写入数据

将 Boot 表写入到 FLASH 中，同时仍可用 FLASH DQ6 端口作为写完成标志位。当 DQ6 电平稳定后，Boot 表被存储到 FLASH 中。这样就完成了一次完整的并行加载。即实现系统的脱机运行。程序流程如图 5－11。



对 CPU 初始化，关键设置 OVLY＝1，DROM＝0，MP/MC＝0 等

向 FLASH 编程，使之处于 可读写状态

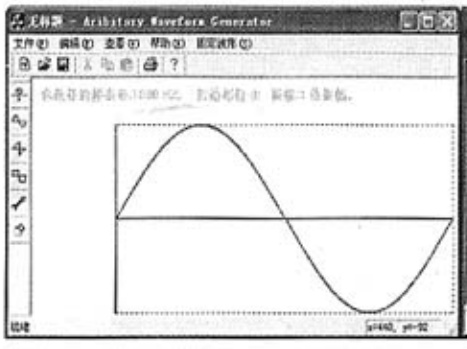
并口 BOOT 只能将程序写入 0X8000—0Xffff

FLASH 的 0Xffff 必须存储 BOOT 表首地址

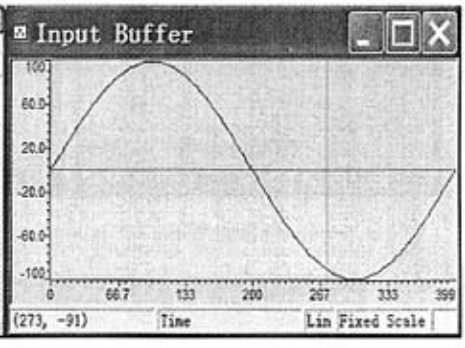
图 5－11 Boot Loder 流程

5.5 结果分析

本节将对波形的输入输出结果作出比较分析，并且给出它们的具体图示。首先，见下图中 VC 窗口绘制波形 5—12 与 DSP 的软件仿真输出波形 5—13，由图示，可以看出仿真结果无明显异常和失真，正弦波形的输出平滑流畅，采用了较多的采样点数，仿真结果非常好。



5-12 VC 生成的正弦波



5-13 DSP 仿真输入的正弦波

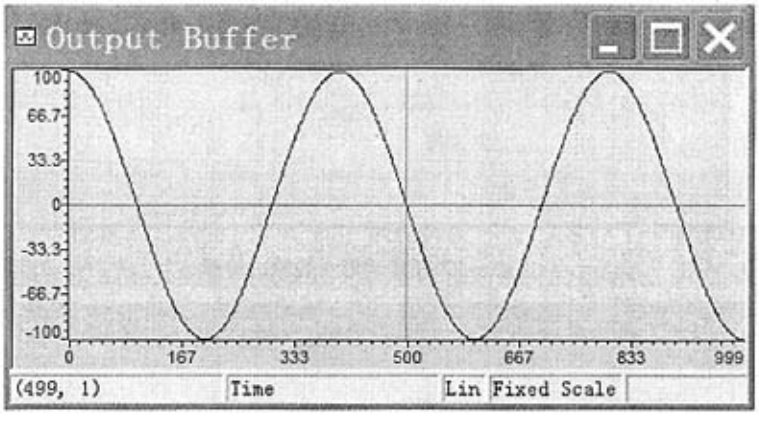
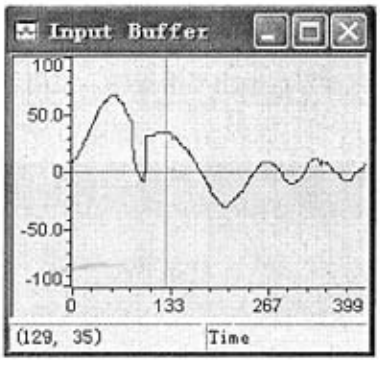


图 5—14 DSP 仿真输出的正弦波

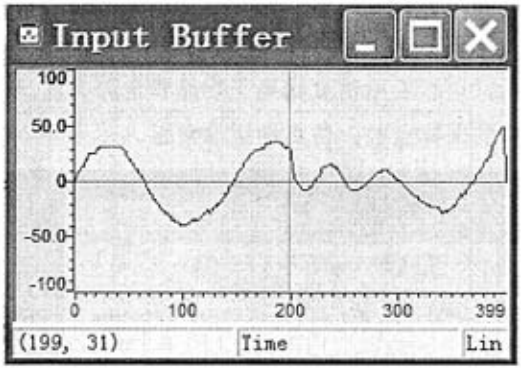
其中，DSP 仿真输入的波形图就是由 VC 生成的波形的采样数据依次读入获得，只要采样点数够多，它的波形图非常标准，不会出现偏差和失真。由图可见，DSP 仿真输出结果是输入数据点数的周期输出，波形连续不断，周期性复现，而且平滑流畅。

下面给出任意波形的输入输出结果图，其具体实现方法和理论都跟前述的正

弦波类似，唯一的区别就是它需按下鼠标左键绘制，而正弦波可以点击相关图标直接生成。任意波形的输入输出如图 5－15、5－16、5－17、5－18 所示：



5－15 任意波形 1



5－16 任意波形 2

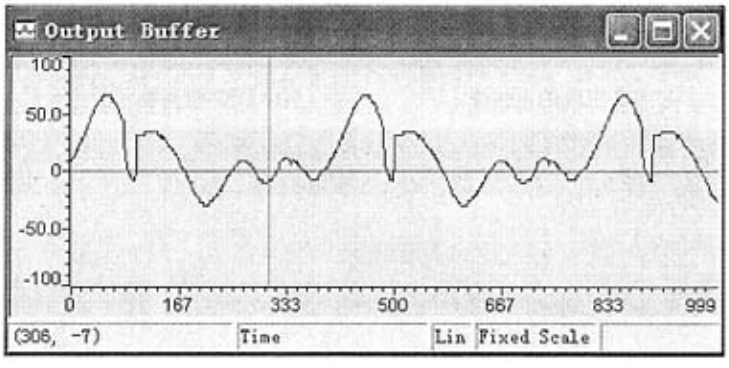


图 5－17 任意波形 1 的输出波形

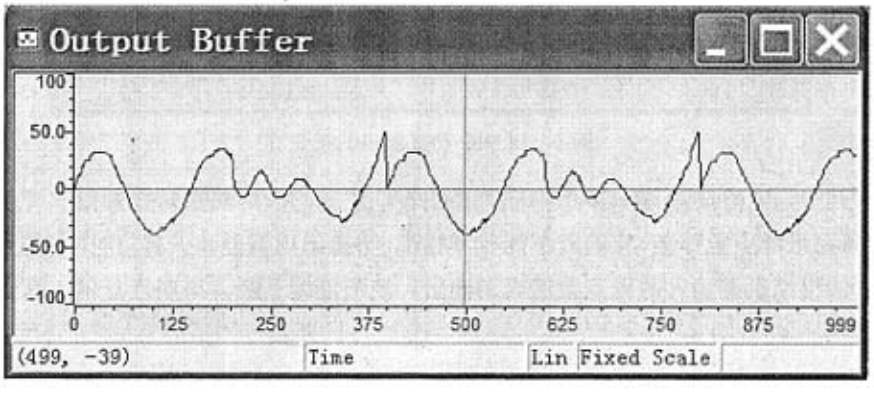


图 5－18 任意波形 2 的输出波形

下图给出任意波形的波形输入输出结果的调试窗口图 5－19：

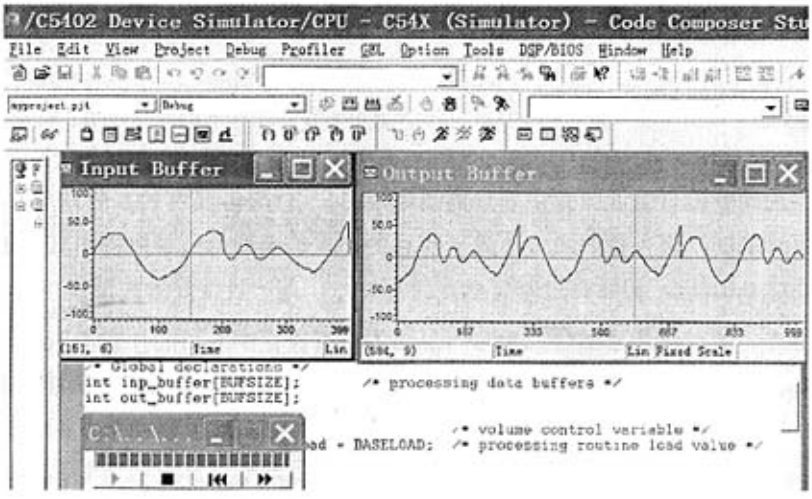


图 5－19 CCS 波形输入输出调试窗口图

5.6 本章小结

本章是系统的软件设计部分，首先是详细阐述了 VC 波形发生部分，包括波形的绘制、图形保存、图形的采样以及对它的优化等。其次对常规波形的生成和任意波形的绘制窗口以及频率、初相等的设置窗口都给出了图示。接下来介绍了对波形频率进行控制的三种方法。对 DSP 软件开发环境 CCS 作了个简单的介绍，接着就是对系统控制部分的软件设计逐一说明，包括系统的初始化、中断向量表、WatchDog、BootLoader 等。最后对波形结果进行了分析讨论。

第六章 总结及展望

6.1 总结

课题研究的任意波形发生器思路清晰明了，AD7846 具有较高的精度，并且采用了足够多的采样点数，设计达到了课题的要求。由理论推导可得出其频率分辨率可高达 0.01HZ，还具有控制灵活方便等特点，是一种很好的波形发生器。

此外，通过对课题的研究学习，总结了系统设计中应该的注意问题：

(1)高速系统特别是模拟数字混合系统要特别注意接地问题。除了电源端相连外，数字地和模拟地分开。另外，对于高速系统使用大面积地阻抗非常重要。印制电路板最好采用多层布线，其中的一层作为地层，且地层应尽量覆盖到高速器件的下方。

(2)妥善解决电源去耦问题对于高速数字系统也很重要。为防止电源输入端的电缆或连线引入分布电感，电源与器件尽量靠近，并在总的电源输入端跨接大容量的去耦电容。

(3)信号走线时，应避免数字、模拟信号交叉走线，如必须交叉，尽可能直角交叉。尽量采用多层布线，相邻层的走线尽量正交。

6.2 展望

电路的设计工作基本完成，功能基本实现，但是波形的输出频率相对比较低，为了增加系统的通用性，可以考虑采用精度够高、转换速率更快的数/模转换器，或者采用一些常用的 DDS 芯片自带的 D/A 转换器，这样就可以把波形频率提高上来而又同时兼顾到了精度。

系统还可以进一步优化，使其更加方便实用。可以在以下几个方面优化：

a) 是低通滤波器的改进。可以设计一些程控滤波器组，开关选通控制对不同的频率信号进行滤波。这样可以把频率范围扩的更宽，非常有意义。

b) 可以设计个简易的键盘，这样看起来系统更加完整。

c) 增加一个液晶显示屏，输出的波形信号可以直接显示到屏幕上。

参考文献

[1] 李晓明 基于AD9857的任意波形发生器的设计 电子科技大学 2002.3

[2] 薛文 DDS任意波形发生器的设计与实现 南京理工大学 2004.7

[3] 潘登 基于DDS技术的可编程任意波形发生器 武汉大学 2004.4

[4] 万天才，频率合成技术及发展，电子产品世界，1999年9月，51～52

[5] 吕庆 基于DSP的任意波形发生器的研究 西安电子科技大学 2004.1

[6] 张玉兴，彭清泉，DDS的背景杂散信号分析,电子科技大学学报,1997

[7] H.T.Nicholas, III H.Samueli. An Analysis of the Output Spectrum of Direct Digital Frequency Synthesizers in the Presence of Phase-Accumulator truncation. IEEE Proc.41 st AFCS,1987:495～502

[8] H.T.Nicholas III H.Samueli, B.Kim. The Optimization of Direct Digital Frequency Synthesizer Performance in the Presence of Finite of Word Length Effects. IEEE Proc.42th AFCS, 1998: 357363

[9] J.Vankka. Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis. IEEE Proc.50th AFCS, 1996: 942950

[10] M.J.Flanagan, G A.Zimmerman. Spur-reduced Digital Sinusoid Synthesis. IEEE Trans. On COM.,1995,43(7)

[11] 张骏凌，张玉兴，直接数字频率合成器中相位噪声分析，电子科技大学学报，1999(2)，24～27

[12].V.S.Reinhardt, Spur reduction techniques in direct digital synthesis, IEEE Proc,47th AFCS, 1993, 230～241

[13] 蒋毅 蒋明 并行D/A转换器AD7846及其接口设计 世界电子元器件 2006

[14] 邓重一 AD7846在自动测试设备中的应用 电子测试 2002

[15] Analog Device Inc,AD7846 data sheets

[16] Maxim Integrated Products,MAX293 data sheets

[17] BURR-BROWN products,PGA103 data sheets

[18] 颜友钧，朱宇光主编 DSP 应用技术教程 中国电力出版社 2002

[19] 汪春梅 孙洪波等编著 TMS320C5000 系列 DSP 系统设计与开发实例 电子工业出版社 2004.7

[20] 马双宝 基于 DSP 的人体皮肤测量系统的研究 武汉理工大学 2006.4

[21] 王丹 DSP 上的指纹识别模块的实现 电子技术应用 2004.2

[22] 清源科技 TMS320C54X DSP 硬件开发教程 机械工业出版社 2004.1

[23] 邹彦 主编 DSP 原理及应用 电子工业出版社 2005.1

[24] 孙宗瀛 谢鸿琳 编著 TMS320C5X DSP 原理设计与应用 清华大学出版社 2002.3

[25] 赵鹏飞 基于 TMS320C5402 DSP 的指纹识别系统研究 中北大学 2006.2

[26] 唐彬，刘超著 Visual C++案例开发集锦 电子工业出版社 2005

[27] 尹勇 欧光军 DSP 集成开发环境 CCS 开发指南 北京航空航天大学出版社 2003.11

[28] Texas instruments Developing a CCStudio 2.0 DSP/BIOS Application for FLASH Booting on the TMS320C5402 DSK 2001.8

[29] 刘益成 TMS320C54X DSP 应用程序设计与开发 北京航空航天大学出版社

[30] TMS320VC5402 and TMS320UC5402 Bootloader Texas instruments Application Report SPRA618 — February 2000

[31] TMS320C54x DSP Reference Set Texas instruments Application Report Literature Number: SPRU131F April 1999

[32] Atmel Device Inc, AT49LV1024A data sheets

[33] ISCI Device Inc, IS61LV6416 data sheets

致 谢

近三年的硕士学习生活即将结束，回顾自己在学业上的每一点进步，都离不开学校老师、同学们和亲人们的关心和帮助。在此，向他们表示最衷心和诚挚的谢意。

首先感谢我的导师唐广副教授，本文的工作从选题到研究的每一阶段自始至终都得到了唐老师的悉心指导。唐老师理论联系实际的工作作风，丰富、扎实的工程实践经验，敏捷、活跃的思维方式以及对学术问题准确、深刻的分析和把握，都使我在研究中受益非浅。他平易近人，脚踏实地的师者风范和对工作忘我的精神为我今后的学习和工作树立了榜样。在此特向为培养我而付出辛勤劳动的唐老师表示衷心的感谢！

我由衷的感谢兰家隆教授在本课题的论证阶段以及实施阶段给予了悉心指导和宝贵意见。

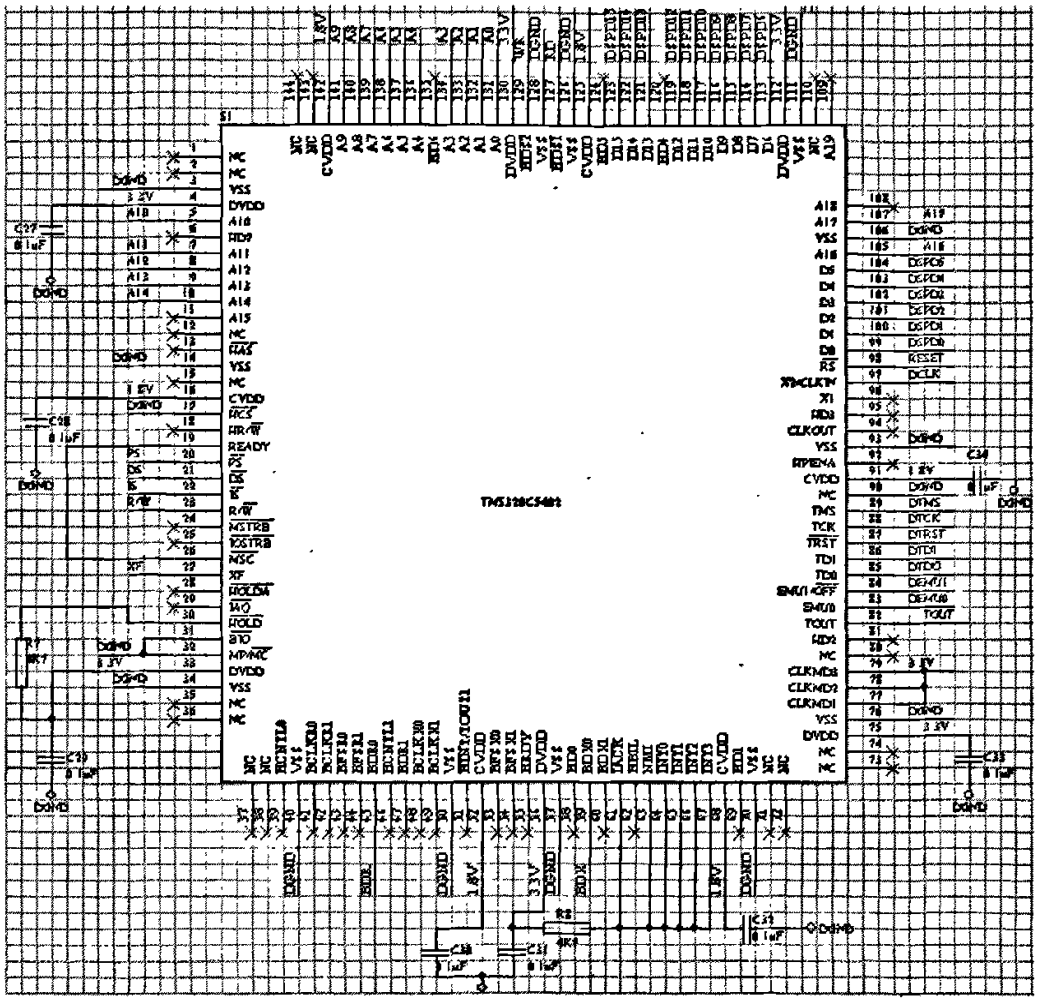
感谢杨宏、黎海明、韩波、黄海、李浩等师兄，和他们在学业上的探讨和交流使我受益非浅。

感谢王挺、邓普、刘武广、卫颜锋、王楠、汪金铭、汪润来、高腾飞、杨荣喜、彭玉吉、杜昊、郑萧、赵娜等同学在学习和生活中的帮助，和他们在一起使我度过了一段快乐、难忘的幸福时光。

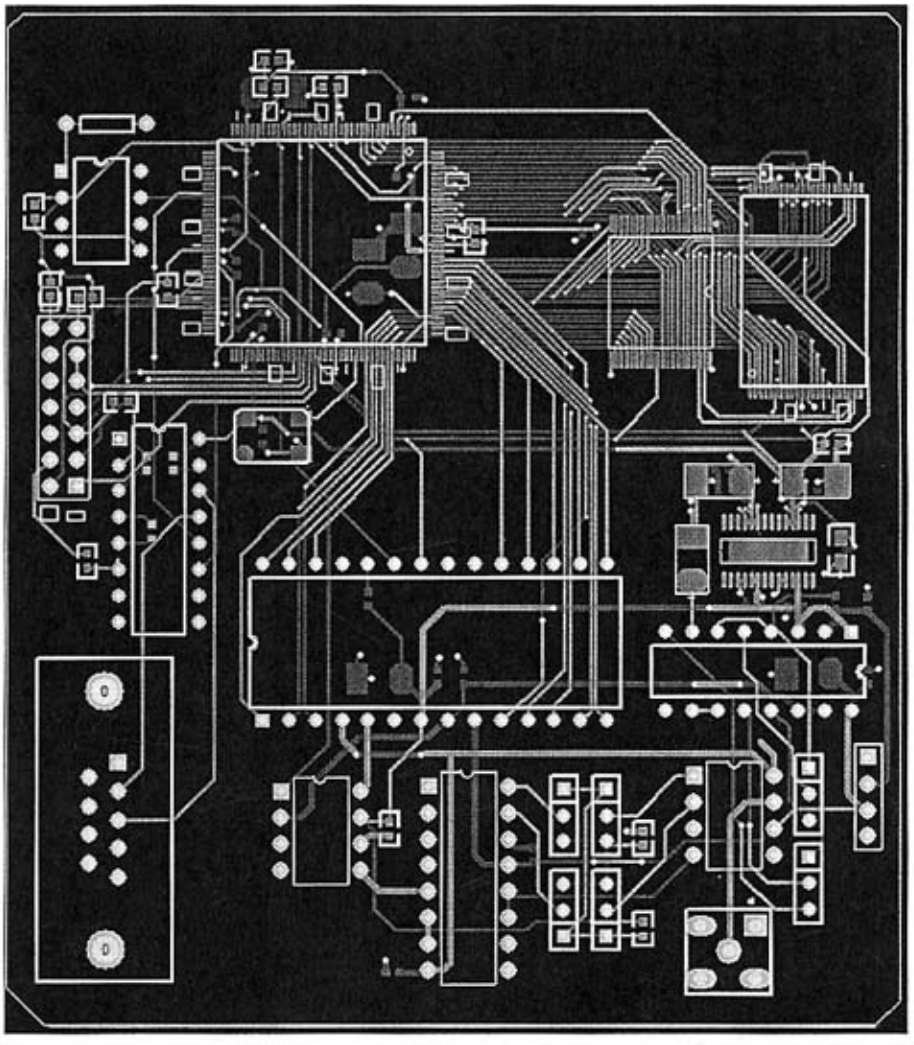
感谢父母对我多年来的养育之恩！感谢我的女友、姐姐、姐夫在生活上对我的帮助和支持！在此谨向他们致以深深的谢意！

附录一

C5402 的引脚连接图



PCB 版图



附录二

```
//系统初始化
c54init:
    _c_int00
    ld #0h,DP
    stm #2000h,SP
    ssbx INTM
    ssbx SXM
    STM #0x7208,SWWSR
    STM #0,SWCR
    STM #0xF800,BSCR
    STM #0,ST0
    STM #0x2b00,ST1              ;INTM=1 off interrupt
    STM #0xBfe4,PMST             ;Interrupt vector map Bf80
    STM #0xFFFF,IFR
    STM #0x0488,IMR              ;Timer0 enabel interrupt
    STM #0,CLKMD
clkcon: LDM CLKMD,A
      AND #0x01,A
      BC clkcon,ANEQ
      STM #0x13ff,CLKMD            ;PLL MUL=(1+1)*20=40MHz
      STM #0x0010,TCR1             ;timer1 stop
      STM #0x0010,TCR              ;TSS=1 Timer stop
      STM #12800,PRD
```

```
      STM #0x0020, TCR
//中断向量表
.include c54.inc
.sect ".vectors"
.ref _c_int00              ; main progrom
.align 0x80                ; must be aligned on page boundary
  RESET
  bd _c_int00
  stm #2000h, SP
.space 19*4*16
 nmi:   RETE                        ; enable interrupts and return from one
        NOP
        NOP
        NOP
sint17 .space 4*16
sint18 .space 4*16
sint19 .space 4*16
sint20 .space 4*16
sint21 .space 4*16
sint22 .space 4*16
sint23 .space 4*16
sint24 .space 4*16
sint25 .space 4*16
sint26 .space 4*16
sint27 .space 4*16
sint28 .space 4*16
```

```
sint29 .space 4*16
sint30 .space 4*16
  INT0  b INT0ISR
        NOP
        NOP
  int1:   RETE
        NOP
        NOP
        NOP
  int2:   RETE
        NOP
        NOP
        NOP
  tint0: b timeisr
        NOP
        NOP
  ......
  .end
//数据转换程序
#include<stdio.h>
#include<stdlib.h>
#include<iosteam.h>
#include<conio.h>
  viod main()
{
  char *str
```

```
FILE  *a
FILE  *b
a=fopen("..\\waveform.dat","r");   /*打开名为 waveform.dat 的文件*/
b=fopen("..\\waveform.h"."w");     /*打开名为 waveform.h 的文件*/
    for(int i=0;i<=9;i++)
   {
      for (int  j=1; j<=6,;i++)
       while(!feof (a))
       {
         fgets (str2,a);
          switch(j%6)
         {
          case 1:fputs ("0x", b);break;
          case 2:fputs ("str", b);break;   /*将取出的字符存入 b 文件*/
          case 3:fputs ("str", b);break;
          case 4:fputs ("str", b);break;
          case 5:fputs ("str", b);break;
          case 0:fputs(",", b);break;
          }
        }
      }
    }
  fclose(a);
  fclose(b);
  getch()
}
```

```
//Boot Loader 过程程序
//bootloader main
  void main()
 {
    init_board();
    flash_ready();
    flash_erase();
//flash write(0x8000,0xffff);
    for(i-0;i<=0x8000;i++)
   {
    flash_write((0x8000+i),flash_data[i]);
   }
    flash_write(0xFFFF,0x8000);
  }
//init 5402
  void init_baord(void)
 {
    *(volatile u16*)CLKMD=0x0000;
    while(*(volatile u16*)CLKMD&0x0001){};
    *(volatile u16*)CLKMD=0x9870;
    *(volatile u16*)PMST=0x00A0;
    *(volatile u16*)SWWSR=0x7FFF;
    *(volatile u16*)BSCR=0x8820;
//port.h
    ioport unsigned portaaaa;
    ioport unsigned port555;
```

```
    ioport unsigned portfffe;
    u16 u_temp;
    u16 u_dq7;
    u16 u_dq61;
    u16 u_dq62;
    u16 i;
//fhash ready state
    void flash_ready()
   {
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xd555)=0xaa;
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xaaaa)=0x55;
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xd555)=0xf0;
    u_temp=*(u16*)(0xfffe);
   }
//flash erase
  void flash_erase()
 {
    u_temp=portfffe;
    port555=0xaa;
    u_temp=portfffe;
    portaaaa=0x55;
    u_temp=portfffe;
    portd555=0x80;
```

```
    u_temp=portfffe;
    portd555=0xaa;
    u_temp=portfffe;
    portaaaa=0x55;
    u_temp=portfffe;
    port555=0x10;
    u_temp=portfffe;
   do
  {
    portfffe=0x11;
    u_dq7=port555;
  }while((u_dq7&0x0080)==0x0000);
 }
//flash write
  void flash_write(u16 u_addr,u16_val)
 {
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xd555)=0xaa;
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xaaaa)=0x55;
    u_temp=*(u16*)(0xfffe);
    *(u16*)(0xd555)=0xa0;
    u_temp=*(u16*)(0xfffe);
    *(u16*)(u_addr)=u_val;
    u_temp=*(u16*)(0xfffe);
   do
```

```
{
    *(u16*)(0xfffe)=0x11;
    u_dq61=*(u16*)(0x8000);
    *(u16*)(0xfffe)=0x11;
    u_dq62=*(u16*)(0x8000);
    *(u16*)(0xfffe)=0x11;
  }whlie((u_dq61&0x0040)!=u_dq62&0x0040);
}  //判断 DQ6 值，若稳定完成则退出
```

攻硕期间取得的研究成果

[1] 王新柳　唐广.　基于DSP的任意波形发生器的设计.　电子科技大学研究生学报.2006,NO.37:1-3