DSP 控制器及其应用

章云　谢莉萍　熊红艳　编著



机 械 工 业 出 版 社

DSP 控制器是一款针对于工业控制而开发的 DSP 芯片，是一款 32 位的高性能单片机。本书首先概述 DSP 控制器的发展过程及其特点；接着介绍 DSP 控制器的总体结构，特别说明 DSP 控制器多组总线技术与并行机制的实现；然后分别介绍片内外设的结构、原理与使用方法；以及 DSP 控制器的指令系统；最后给出 2 个应用实例。

本书可供从事控制系统、通信系统、网络设备、仪器仪表、家用电器等相关领域的广大科技人员和教师阅读参考，也可作为相关专业研究生和本科生的教材。

图书在版编目（CIP）数据

DSP 控制器及其应用/章云等编著．-- 北京：机械工业出版社，2001.8

ISBN 7-111-09097-7

Ⅰ.D… Ⅱ.章… Ⅲ.单片式计算机-控制器

Ⅳ.TP368.1

中国版本图书馆 CIP 数据核字（2001）第 045268 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：贡克勤 版式设计：冉晓华 责任校对：程俊巧

封面设计：方 芬 责任印制：郭景龙

北京京丰印刷厂印刷·新华书店北京发行所发行

2001 年 8 月第 1 版·第 1 次印刷

787mm×1092mm 1/16·12.75 印张·312 千字

0 001—4 000 册

定价：22.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、68326677-2527

前　言

目前，各种各样的控制系统、通信系统、网络设备、仪器仪表等等，都是以微处理器为核心的。几十年来，随着大规模集成电路技术的不断发展，微处理器的性能越来越高、体积越来越小、系列越来越多。微处理器从过去单纯的中央处理单元，发展到将众多外围设备集成进片内形成单片机；由过去的8位机，发展到16位、32位机。DSP控制器就是一款高性能的单片机。

由于大规模集成电路工艺技术的突破，DSP控制器的价格已和普通单片机相接近，但其性能远远超过了普通单片机。特别是随着我国现代化建设的进程，对高性能的控制系统、通信系统、网络设备、仪器仪表，甚至高性能的家用电器的需求难以估量。为了实现高性能，就需要能快速地完成复杂算法，这是普通单片机的瓶颈。DSP控制器是由原来DSP处理器（数字信号处理）发展而来，它的突出特点就是采用多组总线技术实现了并行机制，有独立的加法器和乘法器，有灵活的寻址方式，从而可以非常快速地处理复杂算法。可以预见，在不远的将来DSP控制器将在控制系统、通信系统、网络设备、仪器仪表、甚至高性能的家用电器上得到非常广泛的应用。

在DSP领域中，美国得克萨斯仪器（Texas Instruments）公司的产品有着较强的竞争力。其中TMS320 DSP是其代表系列。1982年TI公司推出了TMS320系列的第一种产品——TMS32010。现在TMS320系列已有定点型的C1X、C2X、C2XX、C5X、C54X，浮点型的C3X、C4X，多处理器型的C8X。DSP控制器——TMS320C24X——是在TMS320C2XX的基础上集成了大量的片内外设而成的一款适用于工业控制的DSP芯片。目前该款芯片在国内控制界得到广泛的关注。

DSP控制器出现的时间不长，可以利用的资料较少，主要是各厂家的说明书与使用手册。为了使广大工程技术人员更好地进行DSP控制器的开发工作，本书作者在近几年从事DSP控制器的教学与研发工作的基础上写下了这本书。本书首先概述DSP控制器的发展过程及其特点；接着介绍DSP控制器的总体结构，特别说明DSP控制器多组总线技术与并行机制的实现；然后分别介绍片内外设的结构、原理与使用方法；跟着介绍DSP控制器的指令系统；最后给出2个应用实例。本书的第1章、第2章、第3章的3.1节和第5章由章云编写；第4章以及附录由谢莉萍编写；第3章除3.1节以外的部分由熊红艳编写；所有插图由谢莉萍和熊红艳完成。

曾岳南博士全面审阅了本书的内容并提出了宝贵的修改意见，在此向他表示感谢。得克萨斯（TI）仪器公司的中国大学计划在我院设立了DSPS开发实验室并提供了许多相关技术资料，也在此向他们表示感谢。本书的出版得到了广东省重点学科和广东工业大学研究生教材基金的资助。

由于作者水平有限，不足之处敬请读者批评指正。

作　者

2001年5月于广东工业大学自动化学院

目　录

前言

第 1 章　概述 …… 1

1.1　引言 …… 1
1.2　二进制、补码及其运算 …… 3
1.2.1　数制 …… 3
1.2.2　补码与加、乘运算 …… 4
1.3　DSP 控制器的基本原理 …… 9

第 2 章　总体结构 …… 11

2.1　总线结构 …… 12
2.2　中央处理单元 …… 14
2.2.1　输入比例部分 …… 15
2.2.2　中央算术逻辑部分 …… 15
2.2.3　乘法部分 …… 16
2.3　辅助寄存器算术单元 …… 16
2.4　状态寄存器 ST0 和 ST1 …… 17
2.5　存储器与 I/O 空间 …… 19
2.5.1　与外部存储器和 I/O 空间接口的信号 …… 19
2.5.2　程序存储器 …… 20
2.5.3　局部数据存储器 …… 21
2.5.4　全局数据存储器 …… 24
2.5.5　I/O 空间 …… 25
2.6　程序控制 …… 26
2.7　时钟源模块 …… 28
2.8　系统复位 …… 32

第 3 章　片内外设 …… 35

3.1　事件管理模块 …… 35
3.1.1　通用定时器 …… 37
3.1.2　比较单元与 PWM 发生器 …… 50
3.1.3　捕获单元 …… 61
3.1.4　正交编码脉冲电路 …… 65
3.2　模/数转换模块 …… 66
3.2.1　结构概述 …… 67
3.2.2　模/数转换控制与操作 …… 70
3.3　SCI 串行通信接口模块 …… 71
3.3.1　串行通信的工作原理 …… 71
3.3.2　串行通信接口模块 SCI 的结构 …… 74
3.3.3　多机通信 …… 83
3.4　SPI 串行外设接口模块 …… 84
3.4.1　串行外设接口结构与工作原理 …… 85
3.4.2　SPI 的多机通信 …… 91
3.4.3　SPI 引脚功能的选择 …… 94
3.5　数字 I/O 端口 …… 95
3.5.1　数字 I/O 端口概述 …… 95
3.5.2　数字 I/O 端口寄存器 …… 97
3.6　看门狗与实时时钟 …… 98
3.7　中断管理系统 …… 102
3.7.1　DSP 内核中断 …… 102
3.7.2　事件管理模块的中断 …… 106
3.7.3　系统模块中断 …… 110

第 4 章　指令系统 …… 112

4.1　寻址方式 …… 112
4.1.1　立即寻址 …… 112
4.1.2　直接寻址 …… 113
4.1.3　间接寻址 …… 114
4.2　句法格式 …… 117
4.2.1　汇编句法格式 …… 117
4.2.2　指令分类表 …… 119
4.3　传送指令 …… 123
4.4　算术操作指令 …… 142
4.5　逻辑运算指令 …… 155
4.6　分支转移指令 …… 161

第 5 章　应用实例 …… 170

5.1　基于空间矢量的通用变

频器 …… 170
5.2 快速傅里叶变换（FFT） …… 173
5.2.1 快速傅里叶变换的基本原理 …… 173
5.2.2 快速傅里叶变换的 DSP 实现 …… 177
附录 …… 182
附录 A DSP240 引脚说明 …… 182
附录 B 可编程寄存器汇总 …… 189
附录 C 指令汇总 …… 191
参考文献 …… 197

第1章 概 述

1.1 引言

随着1946年第一台电子计算机的诞生，一场数字化的技术革命悄然地引发。如果说当初计算机的出现纯粹是为了解决日益复杂的计算问题，那么现在计算机已无处不在。自动控制与计算机几乎是同步地发展着。自动控制系统的核心问题是如何寻找和实现最佳的控制律。在A/D、D/A以及I/O技术出现并成熟之后，最佳控制律的实现问题就变成了最佳控制律的运算（代数、微分、积分等运算）问题，从而计算机作为自动控制系统的核心部分就是自然之事。

计算机真正在自动控制系统中发挥重要作用还得缘起20世纪70年代微处理器（Microprocessor）的出现，它使得计算机在体积、价格上得以突破，为计算机在各种技术领域的应用提供了可能。微处理器即计算机的中央处理单元（CPU）和控制单元的集成，它配上一定的存储器、I/O接口和其它外设，就可构成自动控制系统的通用控制器。70年代Intel公司的8080，Motorola公司的M6800和Zilog公司的Z80是当时三个著名的8位微处理器。

随着大规模集成电路技术不断改进，一方面微处理器由8位向16位、32位甚至64位发展，在配上外围设备后便形成单板机或微型机（也称为个人计算机，Personal Computer），使得计算机不但在计算、控制中应用，而且使得计算机逐步地走入了家庭；另一方面将微处理器与外围设备集成到一块芯片形成单片机（Single _chip Microcomputer），以适应控制器体积越来越小的工程要求。

正是由于单片机的出现，计算机在控制领域的应用又得到了一次突破。单片机不但小巧、成本低，而且由于众多设备集成到了一块芯片上带来了功耗小和抗干扰能力强的优点。另外，它可以方便地组成各种智能式控制设备和仪器，做到机电仪一体化；也可以方便地实现多机和分布式控制，使整个控制系统的效率和可靠性大为提高。单片机有许多类型，有低档的1位、4位和8位单片机，也有高档的8位、16位单片机。其中Intel公司的MCS-51系列、Motorola公司的68系列和Zilog公司的Z8系列为大家所知。

DSP（Digital Signal Processor）实际上也是一种单片机，它同样是将中央处理单元、控制单元和外围设备集成到一块芯片上。DSP最早是针对数字信号处理，特别是语音、图像信号的各种处理而开发的。由于这类信号处理的算法复杂，要求DSP必须具有强大快速的运算能力。因此，DSP有别于普通的单片机，它采用了多组总线技术实现并行运行机制，从而极大地提高了运算速度，也提供了非常灵活的指令系统。近些年来，各种集成化单片DSP的性能不断得以改进，相应的软件和开发工具日臻完善，价格迅速下降，使得DSP在控制领域的应用倍受关注。

在DSP领域中，德州仪器（TI）公司的产品及其配套技术与开发工具最有强大的竞争力，其中TMS320 DSP是它的代表系列，本书以TMS320C24X进行介绍。TMS320C24X也称为DSP控制器，是TI公司专门针对电机、逆变器、机器人、数控机床等控制而设计的。它以

C2XLP 16bit 定点 DSP CPU 为内核，配置了完善的外围设备，包括事件管理模块（EV）、A/D 转换模块（ADC）、串行通信接口模块（SCI）、串行外设接口模块（SPI）、中断管理系统和系统监视模块，其中事件管理模块（EV）含有通用定时器、比较器、PWM 发生器、捕获器。

DSP 控制器的结构和主要特性如下：

□中央处理单元

- 32 位中央算术逻辑单元（CALU）。
- 32 位累加器。
- 16 位×16 位乘法器。
- 3 个比例移位器。
- 间接寻址用的 8 个 16 位辅助寄存器和它的辅助算术单元（ARAU）。

□存储器

- 544 字片内双口 RAM，其中 288 字用于数据，256 字用于程序/数据。
- 16K 字片内 ROM 或 FLASH EEPROM，用作程序存储器。
- 244K 字可寻址空间，程序存储空间 64K 字，数据存储空间 64K 字，I/O 空间 64K 字，还有 32K 字全局存储空间。
- 外部有 16 位地址总线，16 位数据总线，支持软件、硬件等待状态。

□程序控制

- 4 级流水线操作。
- 8 级硬件堆栈。
- 6 个外部中断：电源保护、复位、不可屏蔽中断（NMI）和 3 个可屏蔽中断。

□指令集

- 源代码与定点 TMS320C2X、C2XX、C5X 兼容。
- 单周期相乘/累加指令。
- 单指令重复操作。
- 程序/数据存储器中的块移动。
- 丰富的变址寻址能力。
- 具有基于 2 的 FFT 倒位序变址寻址能力。

□事件管理模块

- 3 个 16 位通用定时器。
- 3 个全比较/PWM 单元。
- 3 个简单比较/PWM 单元。
- 4 个捕获单元。

□2 个 8 通道 10 位 A/D 转换器

□串行异步数字通信接口模块（SCI）

□串行外设接口模块（SPI）

□中断管理系统

□由看门狗和实时中断定时器组成的系统监视模块

□28 个可独立编程的 I/O 引脚

□速度

●单周期指令执行时间为 50ns（20MIPS）。

□电源

● 5V 或 3.3V 静态 CMOS 工艺。

● 4 种降低功耗的方式。

1.2 二进制、补码及其运算

DSP 控制器就是一款高性能的单片机。运算是单片机（计算机）的主要任务，为此首先简单介绍二进制、补码及其运算的实现方法。

二进制数是计算机原理看似最简单的内容，但却是最容易弄混淆的。对于计算机，它的基本任务是：从某个“地方”取数，按某种规律运算，再把结果放到某个“地方”。这里“地方”是指存储器、I/O 接口等。因此，地址与数据是计算机中两个基本概念。由于计算机只存在“0”、“1”两种状态，因此在计算机中地址、数据都是二进制数表示。另外，数据存在着正数与负数、整数与小数，为了区别它们还必须产生用二进制数表示的编码系统。这样一来，在计算机中给出的一个二进制数到底是地址还是数据、是正数还是负数、是整数还是小数，常常让人眼花缭乱。本节着重介绍二进制数的编码系统及其特点，至于寻址问题将在第 4 章介绍。

1.2.1 数制

在日常生活中经常使用的是十进制数，由 0～9 十个符号组成，按照逢十进一的规则运算。由于计算机只存在“0”、“1”两种状态，故计算机中的数用二进制表示。二进制数只有 0 和 1 两个符号，按照逢二进一的规则运算。与二进制密切相关的是十六进制，它由 0～9、A～F 共 16 个符号组成，按照逢十六进一的规则运算。为了以示区别，二进制数以后缀“B”表示，十六进制数以后缀“H”表示。二进制数与十进制数、十六进制数与十进制数的关系如下：

$$[b_{n-1}b_{n-2}\cdots b_0 \cdot b_{-1}b_{-2}\cdots b_{-m}]_B = b_{n-1}\times 2^{n-1}+b_{n-2}\times 2^{n-2}+\cdots+b_0\times 2^0 + b_{-1}\times 2^{-1}+b_{-2}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m}$$

$$[h_{n-1}h_{n-2}\cdots h_0 \cdot h_{-1}h_{-2}\cdots h_{-m}]_H = h_{n-1}\times 16^{n-1}+h_{n-2}\times 16^{n-2}+\cdots+h_0\times 16^0 + h_{-1}\times 16^{-1}+h_{-2}\times 16^{-2}+\cdots+h_{-m}\times 16^{-m}$$

例 1.1 将下列二进制数、十六进制数转换为十进制。

$$1011.101_B = 1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3} = 11.625$$

$$1011.101_H = 1\times 16^3+0\times 16^2+1\times 16^1+1\times 16^0+1\times 16^{-1}+0\times 16^{-2}+1\times 16^{-3} = 4113.0627$$

由于 $2^4=16$，因此 1 位十六进制数可用 4 位二进制数表示，它们之间的关系见表 1.1。这样一来，二进制数与十六进制数的转换非常方便。

例 1.2 二进制数与十六进制数的转换。

$$[1AB.3E]_H=[0001\quad 1010\quad 1011.0011\quad 1110]_B$$

$$[10101101.11]_B=[1010\quad 1101.1100]_B=[AD.C]_H$$

计算机除了受“0”、“1”两种状态的限制外，还受到数据长度的限制。在计算机中数据总线的位数是有限的，一般有 8 位、16 位或 32 位。用 8 位二进制数表示的数据称为一个字节，

可以表示 0～255 的无符号整数；用 16 位二进制数表示的数据称为一个字，可以表示 0～65535 的无符号整数；用 32 位二进制数表示的数据称为一个双字，可以表示 $0\sim2^{32}-1$ 的无符号整数。一般情况下，用 n 位的二进制数表示无符号整数的范围是：$0\sim2^n-1$。由于数据位数的限制，在进行数据运算时要注意数据表示的范围，否则将会发生溢出错误。

表 1.1　十进制与二进制、十六进制的关系

十　进　制	十六进制	二　进　制	十　进　制	十六进制	二　进　制
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

除了要注意数据表示的范围外，还要关注数据的符号。数据有正、负两种，正好可用“0”、“1”、两种状态来表示，以“0”表示正数，以“1”表示负数。一般把符号位放在最高位，其余位表示数据。在计算机中，以这种格式表示的二进制数称为原码，n 位的原码表示有符号整数的范围是：$-2^{n-1}\sim2^{n-1}-1$。

定义 1（二进制原码）：整数 X 用 n 位二进制原码表示，其规则如下：

1. 若 $X\geqslant0$，它的最高位为 0，其余位是 X 的二进制数；

2. 若 $X<0$，它的最高位为 1，其余位是 $|X|$ 的二进制数。

例 1.3　用二进制数表示无符号、有符号整数。

无符号整数：$[0101\ 1101]_B=93$　　$[1101\ 1101]_B=221$

有符号整数：$[0101\ 1101]_{原}=93$　　$[1101\ 1101]_{原}=-93$

在计算机中，无符号整数的运算比较容易，因为一般情况下计算机中的加法器与乘法器都是按无符号整数设计的，这样一来只需注意数据的范围，不要发生溢出错误便可。对于有符号整数的运算就要麻烦一些，由于最高位是符号标志，如果直接参与运算，会带来不正确的结果。因此，对于有符号整数的运算分两步进行：先按规则运算符号位并寄存；然后屏蔽掉符号位，按无符号数进行运算，再把结果与寄存的符号位拼接成最后的结果。

可以看出，在计算机中直接进行有符号整数的运算是不方便的，有没有可能经过一定变换，使得有符号整数的运算能够一步完成？下面介绍的二进制补码就可以做到这一点。

1.2.2　补码与加、乘运算

1. 补码与整数加法　在计算机中加、乘运算是最基本的运算，对于有符号整数以及小数的加、乘运算能否有一套统一的编码系统来完成是一件非常有意义的事。下面首先从有符号整数的加法谈起，引入补码并同时介绍其特点，然后分别介绍补码形式的整数与小数的乘法运算。

前面说到，计算机的数据长度是有限的。对于 8 位计算机，超过 2^8 就会溢出，这跟数学

中的同余（mod）运算是一样的，即

$$X=[X+2^8] \mod 2^8$$

为了讨论方便，设 X 和 Y 是正整数，N 是大于 X 和 Y 的正整数，那么

$$\begin{aligned} X+(-Y)&=[N+X+(-Y)] \mod N \\ &=[X+(N-Y)] \mod N \end{aligned}$$

从上式可以看出，如果用 $N-Y$（这是大于 0 的数）代表（$-Y$），则有符号的加法运算转换成了无符号的加法运算。下面给出二进制补码表示的定义：

定义 2（二进制补码）：整数 X 用 n 位二进制补码表示，其规则如下：

1）若 $X\geqslant 0$，它的原码就是它的补码；

2）若 $X<0$，它的补码为 $2^n-|X|$。

原码和补码都可以表示有符号数，它们之间的区别在于：对于原码表示，其最高位仅仅是一个符号标志，不能直接参与数据的加法运算，否则会引起错误的结果；对于补码表示，则不一样，其最高位既是符号标志，也是一位可以参与运算的数据。按二进制补码表示的数据，无论正数还是负数可以直接进行加法运算，但其结果按二进制补码解释。

例 1.4 设 $n=8$，$X=66$，$Y=-24$，写出它们的原码和补码，并求 $X+Y=?$

1）按原码方式直接相加，其结果是错误的。

$$[X]_{原}=[0100\ 0010]_{原} \quad [Y]_{原}=[1001\ 1000]$$

$$\begin{aligned} [X]_{原}+[Y]_{原}&=[0100\ 0010]_{原}+[1001\ 1000]_{原} \\ &=[1101\ 1010]_{原}=-90 \end{aligned}$$

2）按补码方式直接相加，其结果是正确的。

$$[X]_{补}=[0100\ 0010]_{补} \quad [Y]_{补}=2^8-24=232=[1110\ 1000]_{补}$$

$$\begin{aligned} [X]_{补}+[Y]_{补}&=[0100\ 0010]_{补}+[1110\ 1000]_{补} \\ &=[1\ 0000\ 0000]+[0010\ 1010]_{补} \mod 2^8 \\ &=[0010\ 1010]_{补}=42 \end{aligned}$$

上面的讨论说明了采用二进制补码有益于有符号的加法运算，但是按定义 2 求负数的补码需要做一次减法，这与采用补码的初衷是相悖的，有无其它简易的方法求解补码呢？仔细的推敲可得到与定义 2 等价的另一种定义：

定义 3（二进制补码）：整数 X 用 n 位二进制补码表示，其规则如下：

1）若 $X\geqslant 0$，它的原码就是它的补码；

2）若 $X<0$，它的补码为：其原码的符号位不变，数据位求反加 1。

按照定义 3 可以很容易得到有符号整数的补码，从而避免了定义 2 求补码要做减法的困难。

2. 补码运算的溢出 要注意二进制补码表示也有一个数据范围的问题，n 位二进制补码可以表示 $-2^{n-1}\sim 2^{n-1}-1$ 范围内的数。在做二进制补码加法的时候，要特别注意溢出的问题。下面先看一个例题。

例 1.5 按二进制补码求 $X+Y=?$ 并注意符号位与数据的最高位的进位情况。

1）设 $n=8$，$X=110$，$Y=94$

$$[X]_{补}=[0110\ 1110]_{补} \quad [Y]_{补}=[0101\ 1110]_{补}$$

```
    0110 1110
+   0101 1110
-------------
    1100 1100
```

符号位没有进位，数据的最高位有进位。

$[X]_{补}+[Y]_{补}=[1100\ 1100]_{补}=-52$

2）设 $n=8, X=-92, Y=-102$

$[X]_{补}=[1010\ 0100]_{补}\quad [Y]_{补}=[1001\ 1010]_{补}$

```
     1010 0100
+    1001 1010
--------------
   1 0011 1110
```

符号位有进位，数据的最高位没有进位。

$[X]_{补}+[Y]_{补}=[0011\ 1110]_{补}=62$

3）设 $n=8, X=44, Y=82$

$[X]_{补}=[0010\ 1100]_{补}\quad [Y]_{补}=[0101\ 0010]_{补}$

```
    0010 1100
+   0101 0010
-------------
    0111 1100
```

符号位没有进位，数据的最高位没有进位。

$[X]_{补}+[Y]_{补}=[0111\ 1100]_{补}=126$

4）设 $n=8, X=-22, Y=-44$

$[X]_{补}=[1110\ 1010]_{补}\quad [Y]_{补}=[1101\ 0100]_{补}$

```
     1110 1010
+    1101 0100
--------------
   1 1011 1110
```

符号位有进位，数据的最高位有进位。

$[X]_{补}+[Y]_{补}=[1011\ 1110]_{补}=-66$

从前两个算式看，最后的结果是不正确的，因为两个正数相加不可能为负数，两个负数相加也不可能为正数。引起这个结果的原因是相加的结果超出了 8 位二进制补码表示的范围，发生了溢出。所以二进制补码的溢出，不能只看最高位是否有进位，而需要看最高两位（符号位与数据最高位）的进位情况。如果最高两位都有进位或都无进位，则不发生溢出，否则将产生溢出。不少计算机的状态标志寄存器专门有一个溢出位 OV，就是为补码运算所设。

3. 数据扩展与符号扩展 在计算机的计算中，为了提高计算精度，常常需要用多字节来表示一个数，这就存在数据扩展的问题。对于无符号整数，这种扩展比较容易，在前面补 0 即可。对于有符号整数仅仅补 0 是不够的。下面以一个实例说明补码形式的数据扩展并总结出它的规律。

例 1.6 设 $n=8$，$X=66$，$Y=-24$，写出它们的补码：

$[X]_{补}=[0100\ 0010]_{补}\quad [Y]_{补}=[1110\ 1000]_{补}$

设 $n=16$，$X=66$，$Y=-24$，写出它们的补码：

$[X]_{补}=[0000\ 0000\ 0100\ 0010]_{补}$

$[Y]_{补}=[1111\ 1111\ 1110\ 1000]_{补}$

仔细分析例 1.6 可以看出，采用二进制补码表示的数据扩展，实际上就是符号扩展，即正整数前面全部补 0，负整数前面全部补 1。正因为如此，在有的计算机中专门提供了符号扩展方式，就是为了方便实现二进制补码形式的数据扩展。

4. 整数补码的乘法 下面探讨二进制补码的乘法。设 X、Y 是 n 位有符号的整数，其范

围为-2^{n-1}到$2^{n-1}-1$，$N=2^n$。二进制补码的乘法有下面三种情况：

- $X\geqslant 0$，$Y\geqslant 0$。这与无符号乘法一样。
- $X=0$，$Y<0$。这是一种平凡的情况，按补码相乘的结果肯定是0。
- $X>0$，$Y<0$。用补码实现乘法的过程为：

$$\begin{aligned}\{X\times(N-|Y|)\} \mod N &=\{NX-N|Y|\} \mod N\\ &=\{N(X-1)+(N-X|Y|)\} \mod N\\ &=N-X|Y|\end{aligned}$$

注意到整数$X>0$，则$X-1\geqslant 0$。因此，上面的运算表明按照补码相乘其结果也是相应的补码。

- $X<0$，$Y<0$。用补码实现乘法的过程为：

$$\begin{aligned}\{(N-|X|)\times(N-|Y|)\} \mod N &=\{N^2-N(|X|+|Y|)+|X||Y|\} \mod N\\ &=\{N\{N-(|X|+|Y|)\}+XY\} \mod N\\ &=XY\end{aligned}$$

由于$|X|\leqslant 2^{n-1}$，$|Y|\leqslant 2^{n-1}$，所以$N-(|X|+|Y|)\geqslant 0$。因此，对于二个负的整数按照补码相乘其结果也是相应的补码（正整数的补码是其本身）。

综上所述，按照二进制补码进行的有符号整数的乘法毋须关注它的符号，其结果按补码进行解释。在进行二进制乘法运算时有一点需要注意，二个n位二进制整数相乘可以得到$2n$位二进制整数的乘积结果。为了得到正确的$2n$位二进制整数的乘积结果，需首先将n位二进制补码的乘数和被乘数扩展到$2n$位，然后再按二进制补码的方式相乘，根据上面的讨论，其结果肯定是正确的。总的一句话，要得到N位二进制补码的乘积，其乘数与被乘数应为N位二进制补码。

例1.7 设$X=21=[0001\ 0101]_{补}$，$Y=-3=[1111\ 1101]_{补}$，求$XY=?$

若只需8位二进制乘积结果，则

```
                 11111101
          ×      00010101
-------------------------
                 11111101
              11 111101
            1111 1101
-------------------------
00010100 11000001
```

所以 $XY=[1100\ 0001]_{补}=-63$

若需要16位二进制乘积结果，则先要将X、Y进行扩展，即

$X=21=[0000\ 0000\ 0001\ 0101]_{补}$，$Y=-3=[1111\ 1111\ 1111\ 1101]_{补}$

```
                                 11111111 11111101
                          ×      00000000 00010101
-------------------------------------------------
                                 11111111 11111101
                              11 11111111 111101
                            1111 11111111 1101
-------------------------------------------------
00000000 00010100 11111111 11000001
```

所以 $XY=[11111111\ 1100\ 0001]_{补}=-63$

5. 小数的补码与加、乘运算 前面较详细讨论了采用二进制补码的有符号整数的加、乘运算，那么对于有符号的小数，能否移植前面的结论？下面给出详细的讨论。

首先，有符号小数原码的表示可以从整数原码进行自然推广。即最高位为符号位为“0”表示正数，为“1”表示负数；其余位为数据位。不失一般性，设 n 位二进制小数由 m 位整数位（包括符号位）和 k 位小数位组成（$n=m+k$）。这种表示方式也成为 Q_k 方式。若 $k=0$，则退化为完全整数（Q_0）；若 $m=1$，则为完全小数，或称标准小数（若 $n=16$，为 Q_{15}）。

例 1.8 用 8 位二进制原码表示下面的十进制数（$m=4$，$k=4$）：

$$5.625=[0101.1010]_{原}$$

$$-3.25=[1011.0100]_{原}$$

有符号小数补码的定义可以从整数二进制补码的定义进行推广。对于定义 3 可以完全照搬，即对于正数，其原码就是它的补码；对于负数，将其原码的符号位不变，其余位求反加 1。而对于定义 2 有如下自然的推广。

定义 4（一般性的二进制补码）：设 x 是由 m 位整数位（包括符号位）和 k 位小数位组成的 n 位二进制数，$n=m+k$。它的补码表示如下：

1）若 $x\geqslant 0$，它的原码就是它的补码；

2）若 $x<0$，它的补码为 $2^m-|x|$。

例 1.9 用 8 位二进制补码表示下面的十进制数（$m=4$，$k=4$）。

采用定义 3 求：$5.625=[0101.1010]_{补}$　$-3.25=[1100.1100]_{补}$

采用定义 4 求：$-3.25=2^4-[0011.0100]_B=[10000.0000]_B-[0011.0100]_B$

$$=[1100.1100]_{补}$$

定义 3 和定义 4 都是二进制补码的一般性定义，它们是完全等效的，有兴趣的读者可自行证明。根据定义 3 或定义 4 给出的带小数二进制补码，其加法、乘法运算与整数补码的加法、乘法运算是否同样简单呢？答案是肯定的。

对于带小数的加法，首先要注意小数点位置要对齐，即需要相加的两个小数的格式要一致（m 与 k 的值要一样），不然的话要通过移位操作使它们一致。另外，在计算机中并没有小数点的硬件设置，小数点的位置是一种“假想”，也就是说计算机中的加法器（也包括乘法器）都是按无符号整数来运算的。基于这两点，下面简单推导有符号小数二进制补码加法实现的原理。

设 x、y 是由 m 位整数位（包括符号位）和 k 位小数位组成的 n（$n=m+k$）位二进制数，取 $X=x2^k$，$Y=y2^k$，即 X、Y 变为完全整数，也就是将小数点右移 k 位。按照前面讨论的结果知，完全整数可以采用补码形式直接相加；然后，将相加结果的小数点左移 k 位便是 x 与 y 相加的结果。不失一般性，设 $x>0$，$y<0$，则

$$[x+y]_{补}=[X+(2^n-|Y|)]\times 2^{-k}=[x2^k+(2^{m+k}-|y|)2^k]\times 2^{-k}$$

$$=[x+(2^m-|y|)]$$

这个推导说明，只要 x、y 的补码按照定义 3 或定义 4 给出，其加法可以直接相加。注意对于由 m 位整数位（包括符号位）和 k 位小数位组成的 n（$n=m+k$）位二进制数所表示的范围是 -2^{m-1} 到 $2^{m-1}-2^{-k}$，超出此范围将发生溢出。

有符号小数的乘法实现原理与有符号小数的加法是类似的。思路都是先把它看成是完全整数（相当于将小数点右移 k 位），然后按完全整数进行相乘，最后再确定小数点的位置。如

果相乘的两个数据的小数位都是 k 位，则最后的结果要将小数点左移 $2k$ 位；如果相乘两个数据的小数位分别是 k_1 位和 k_2 位（这一点与加法不同，两个数据可以有不同的数据格式），则最后的结果要将小数点左移 k_1+k_2 位。

不失一般性，设 $x>0$ 由 m_1 位整数位（包括符号位）和 k_1 位小数位组成，$y<0$ 由 m_2 位整数位（包括符号位）和 k_2 位小数位组成，$n=m_1+k_1=m_2+k_2$。取 $X=x2^{k_1}$，$Y=y2^{k_2}$，那么

$$\begin{aligned}[x\times y]_{补}&=2^{-(k_1+k_2)}[X\times Y]_{补}=[X\times(2^n-|Y|)]2^{-(k_1+k_2)}\\&=[x2^{k_1}\times(2^n-|y|2^{k_2})]\times 2^{-(k_1+k_2)}\\&=x\times(2^{m_2}-|y|)\end{aligned}$$

例 1.10 设 $x=5.25=[000101.01]_{补}$，$y=-0.75=[111111.01]_{补}$，求 $xy=?$

若只需 8 位二进制乘积结果，则

$$\begin{array}{rr}
 & 11111101\\
\times & 00010101\\
\hline
 & 11111101\\
11 & 111101\\
1111 & 1101\\
\hline
00010100 & 11000001
\end{array}$$

所以 $xy=[1100.0001]_{补}=-3.9375$

若需要 16 位二进制乘积结果，则先要将 x、y 进行扩展，即

$x=5.25=[0000\ 0000\ 0001\ 01.01]_{补}$，$y=-0.75=[11111111\ 1111\ 11.01]_{补}$

$$\begin{array}{rr}
 & 11111111\ 11111101\\
\times & 00000000\ 00010101\\
\hline
 & 11111111\ 11111101\\
11 & 11111111\ 111101\\
1111 & 11111111\ 1101\\
\hline
00000000\ 00010100 & 11111111\ 11000001
\end{array}$$

所以 $xy=[11111111\ 1100\ 00.01]_{补}=-3.9375$

上面讨论了二进制补码的整数、小数与有符号数的加乘运算原理，可以看出采用二进制补码带来了许多便利。因此，在实际应用时应尽可能转换成二进制补码的形式存储数据。在 DSP 控制器中有一个 16 位×16 位的乘法器，其结果是 32 位。由于该乘法器自动实现乘数与被乘数的数据扩展（16 位到 32 位），因此不需要用户编程干预。

1.3 DSP 控制器的基本原理

无论是微处理器、单片机还是 DSP 控制器，它们的工作原理是基本一致的。不外乎要做的工作都是：从存储器、I/O 接口等地方取数，按某种规律运算，再把结果放到存储器、I/O 接口等地方。因此，在其工作过程中数据流与地址流占统治地位。为了实现数据流、地址流有序地管理和控制，采用数据总线和地址总线是一种最佳的结构方式。数据总线和地址总线就像两条高速公路，数据信息与地址信息分别在其上快速地流动。中央处理单元（CPU）、程序存储器、数据存储器和内部外设等功能模块分别挂接在数据总线和地址总线上。中央处理

单元是控制中心，由它指挥当前时刻谁可以占用数据总线或地址总线，同时它还可以进行有关的运算；程序存储器是物理芯片与人的交接面，由人编写程序指令并写入到程序存储器中，体现了人的意志，中央处理单元只能根据程序的流程进行指挥不能随意发挥；数据存储器用于记录工作过程中的原始数据、中间结果和最后结论；内部外设是集成在芯片内部的与外部世界进行信息交换的功能模块，一般包含I/O、A/D、串行通信等。另外，数据总线和地址总线一般情况下都延伸到芯片外部（到引脚上）。图1.1说明了上述情况。

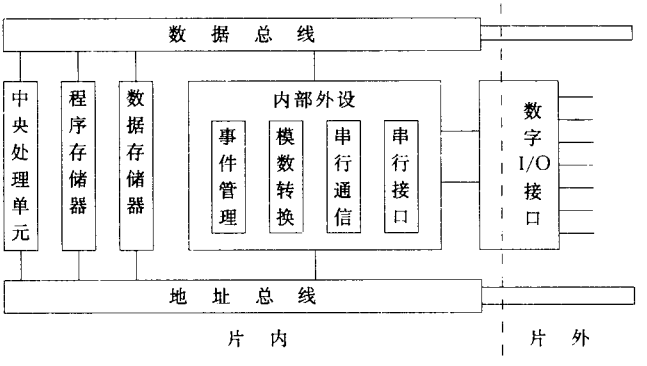


图1.1 DSP控制器的基本原理

一般的微处理器的数据总线和地址总线是单总线方式，相当于一辆车在只有一条道的高速公路上跑，这辆车分时地为大家服务。DSP控制器与此不同，采用多总线方式，相当于是多条道的高速公路，这样一来多辆车可以同时在其上行驶，极大地加快了运行速度。这实质上是一种并行机制。

数据和地址是贯穿任何一种微处理器设计、编程的两个基本概念，特别是地址，它就是数据源、专用寄存器、I/O的代表。每一个存储器、寄存器都有地址，这个容易为大家接受。对于可编程的功能模块（片内的或片外的）它也有地址，有时让人费神。准确地说，对可编程的功能模块的操作，实际上是对它的寄存器（控制的、数据的等）进行操作，这些寄存器必须有唯一地址，否则会引起工作混乱。对于片内外设的功能模块各寄存器的地址是由芯片厂家确定的，应仔细查看手册，不可更改。片外设功能模块各寄存器的地址与所连接的外部地址总线有关，这是设计者一个重要的设计任务，即给每个功能模块分配地址，一旦完成设计做好印制电路也就被固定了下来。

当设计者明确了存储器地址空间、I/O地址空间、片内或片外设功能模块各寄存器的地址后，程序设计的工作就有了一个明晰的轮廓，剩下的任务就是如何组织数据，采用什么控制律或算法，以何种流程来实现。

DSP控制器在工作时要注意如下问题：

●加电后，中央处理单元自动从复位地址（0000H）取出首条指令。

●指令处理周期分为：取指令、指令译码、取操作数、执行指令四个阶段。

●一条指令执行完后，顺序处理下一条指令，除非遇到分支指令或中断响应。

●数据的地址由指令的寻址方式和相应的操作数确定，这一点非常重要。

●对外部世界的访问有两种方式：通过I/O端口或者特定的映射寄存器，前者使用I/O传送指令，后者使用存储器传送指令。

第2章 总体结构

DSP 控制器是一款高性能的单片机。DSP 控制器的总体结构有许多独特的地方：一是采用多组总线结构实现并行处理机制，允许 CPU 同时进行程序指令和存储数据的访问；二是采

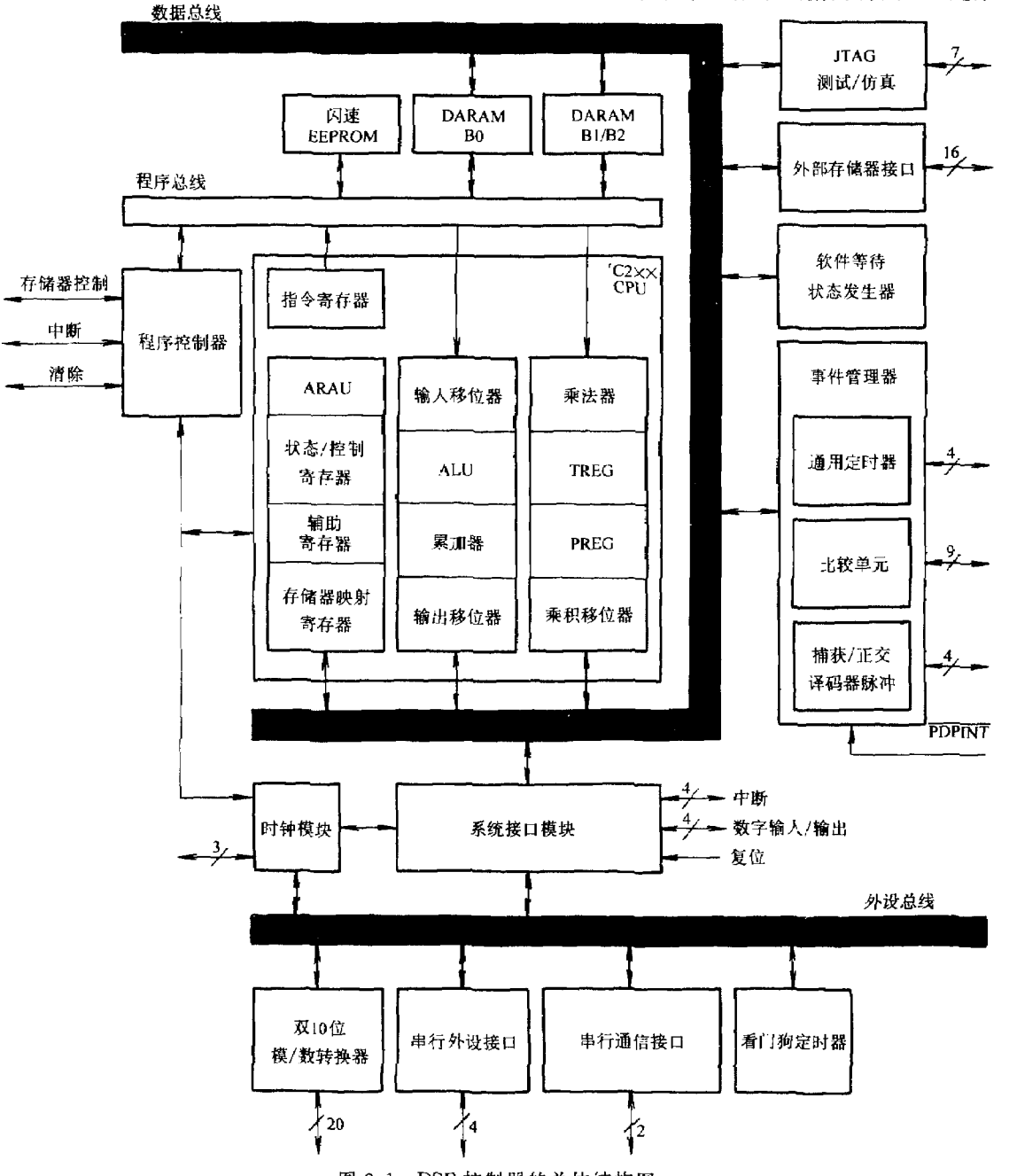


图 2.1 DSP 控制器的总体结构图

用独立的累加器和乘法器，使得复杂的乘法运算能快速进行；三是累加器和乘法器分别连接了比例移位器，使得许多复杂运算或者运算后的定标能在一条指令中完成；四是有丰富的寻址方式，可方便灵活的编程；五是有完善的片内外设，可以构成完整的单片系统。本章主要介绍 DSP 控制器的总体结构，包括总线结构、中央处理单元、存储器与 I/O 空间以及系统的复位。图 2.1 是 DSP 控制器的总体结构图。

2.1 总线结构

目前，在控制领域使用的各种微处理器芯片（各类 CPU、单片机等）的基本任务，就是从某个地方（内、外部存储器或外部接口）取得数据，经过算术或逻辑运算，然后放到相应的地方上。为了区别不同的数据源，需要给其赋予一个独立的地址。数据和地址是任何微处理器都要面对的两个基本要素。因此，在微处理器芯片中采用基于数据/地址总线的结构是最佳的选择。在总线上可以挂接中央算术逻辑单元（CALU）、存储器、定时器等功能模块。通过地址总线和某些控制信号线（与指令密切相关），使得在某一时刻仅仅让某个数据源占用数据总线。这样一来，在地址总线和控制总线的共同作用下，使得数据总线的数据得以有序的流动。

总线结构是计算机体系结构中最基本的结构，它提供了一种标准的接口方式。功能模块之间的信息交换，都可解释为“在什么地址存放数据”或“从什么地址取回数据”。数据与地址成为密不可分的一对伙伴。具备数据与地址接口方式的功能模块都可以挂接到数据/地址总线上。数据/地址总线是双向的，为了保证数据通畅流动，要在中央处理单元统一指挥下按节拍进行工作。

总线结构是各种微处理器芯片的总干道，它的性能（响应速度、位宽、负载能力等）在很大程度上决定了微处理器芯片的性能。为了提高处理速度，一方面可以通过新的工艺使得微处理器芯片能够采用更高频率的晶振以加快响应的速度；另一方面可以加宽数据总线（32 位或 64 位）以增加高精度复杂运算的指令。除此之外，加快处理速度的最佳方案是采用并行机制。一般情况下，总线的操作时序分为四个独立的阶段：取指令、指令译码、取操作数和执行指令，这四个阶段分别面向程序读、数据读和数据写。如果将数据/地址总线分开为三组数据/地址总线，分别对应程序读、数据读和数据写三种情况，这样一来就可以使总线操作时序的四个独立阶段并行处理，从而极大地加快微处理器芯片的处理速度。可以形象地理解，单总线方式就像是一辆车在只有一条道的高速公路上跑；而多组总线方式就像是多辆车在多条道的高速公路上跑。后者的运行速度和效率肯定要超过前者。

DSP 控制器就是采用了多组总线的结构，图 2.2 是 DSP 控制器的总线结构图。其中内部地址总线分为了三条总线：

- 程序读地址总线（PAB），提供读程序的地址；
- 数据读地址总线（DRAB），提供读数据存储器的地址；
- 数据写地址总线（DWAB），提供写数据存储器的地址。

内部数据总线也对应分为三条总线：

- 程序读数据总线（PRDB），将指令代码中的立即数以及表信息传送到 CPU；
- 数据读数据总线（DRDB），将数据存储器的数据传送到 CPU；
- 数据写数据总线（DWDB），将处理后的数据传送到数据存储器和程序存储器。